



SAPIENZA  
UNIVERSITÀ DI ROMA

---

# Algoritmi

## Esercizi 1 — Algoritmi di base

---

Esercizi svolti sui capitoli 1–5: rappresentazione dei numeri, pseudocodice, ordinamento, ricerca e algoritmi numerici. Ogni esercizio ha la soluzione completa.

**Fabio Furini**

[fabiofurini.github.io](https://fabiofurini.github.io)

Dipartimento di Ingegneria informatica, automatica e gestionale — Sapienza Università di Roma

Versione online e materiale scaricabile:

[fabiofurini.github.io/algoritmi/](https://fabiofurini.github.io/algoritmi/)  
[github.com/fabiofurini/algoritmi](https://github.com/fabiofurini/algoritmi)

Anno accademico 2026–2027

---

# Indice

---

|   |                             |     |
|---|-----------------------------|-----|
| 1 | Rappresentazione dei numeri | 2   |
| 2 | Pseudocodice                | 16  |
| 3 | Algoritmi di ordinamento    | 33  |
| 4 | Ricerca lineare e binaria   | 99  |
| 5 | Algoritmi numerici          | 118 |



# Rappresentazione dei numeri

Negli esercizi di questo capitolo i numeri sono rappresentati soltanto con le definizioni e le osservazioni del capitolo «Rappresentazione dei numeri», riassunte dalle seguenti formule.

**Numeri naturali** con  $k$  cifre in base  $b$ :

$$(d_{k-1} \dots d_1 d_0)_b = \sum_{i=0}^{k-1} d_i b^i \quad (\text{valore di una sequenza di cifre})$$

$$[0, b^k - 1] \quad (\text{intervallo di rappresentazione})$$

$$\lfloor \log_b n \rfloor + 1 \quad (\text{numero minimo di cifre di } n \geq 1)$$

**Numeri interi** con  $k$  bit:

$$[-(2^{k-1} - 1), 2^{k-1} - 1] \quad (\text{intervallo in modulo e segno})$$

$$[-2^{k-1}, 2^{k-1} - 1] \quad (\text{intervallo in complemento a due})$$

$$n < 0 \mapsto 2^k + n \quad (\text{complemento a due di un } n \text{ negativo})$$

Nel complemento a due, la rappresentazione di un  $n$  negativo si ottiene anche invertendo i bit di  $|n|$  e aggiungendo 1; una somma va in overflow se e solo se due operandi con lo stesso segno danno un risultato con il bit di segno opposto.

**Numeri reali** in IEEE 754 a singola precisione (1 bit di segno  $s$ , 8 bit di esponente polarizzato  $E$ , 23 bit di parte frazionaria  $f$ ), per  $1 \leq E \leq 254$ :

$$x = (-1)^s \cdot (1.f_1 f_2 \dots f_{23})_2 \cdot 2^{E-127}$$

Doppia precisione: 1 + 11 + 52 bit, polarizzazione 1023.

**Esercizio 1.1 — Rappresentazione dei numeri (E1)**

Si consideri la rappresentazione dei numeri naturali nel sistema binario (base  $b = 2$ ): intervalli di rappresentazione e numero minimo di cifre.

**Domande**

1. Calcolare il più grande numero naturale rappresentabile con  $k = 6$  cifre in binario e determinare il relativo intervallo di rappresentazione.
2. Per i seguenti valori di  $k$  (numero di cifre), determinare l'intervallo di rappresentazione  $[0, b^k - 1]$  in base  $b = 2$ :

$$k = 4, \quad k = 8, \quad k = 10, \quad k = 16$$

3. Determinare il numero minimo di cifre necessarie per rappresentare ciascuno dei seguenti numeri in binario, usando la formula  $\lfloor \log_b n \rfloor + 1$ :

$$n = 64, \quad n = 128, \quad n = 256, \quad n = 512$$

4. Dati  $k = 12$  bit:

- (a) determinare l'intervallo di rappresentazione;
- (b) verificare che  $n = 2,048$  si può rappresentare con  $k = 12$  bit;
- (c) calcolare il numero minimo di bit necessari per rappresentare  $n = 2,048$ .

**Soluzione**

1. Il più grande numero naturale rappresentabile con  $k = 6$  cifre in base  $b = 2$  è:

$$b^k - 1 = 2^6 - 1 = 64 - 1 = 63$$

Esso corrisponde alla sequenza binaria  $(111111)_2$ . Il relativo intervallo di rappresentazione è:

$$[0, 2^6 - 1] = [0, 63]$$

2. Intervalli di rappresentazione per i diversi valori di  $k$ :

$$k = 4 : [0, 2^4 - 1] = [0, 16 - 1] = [0, 15]$$

$$k = 8 : [0, 2^8 - 1] = [0, 256 - 1] = [0, 255]$$

$$k = 10 : [0, 2^{10} - 1] = [0, 1,024 - 1] = [0, 1,023]$$

$$k = 16 : [0, 2^{16} - 1] = [0, 65,536 - 1] = [0, 65,535]$$

3. Il numero minimo di cifre per ciascun numero:

$$n = 64 = 2^6 : \lfloor \log_2 64 \rfloor + 1 = \lfloor 6 \rfloor + 1 = 6 + 1 = 7$$

$$n = 128 = 2^7 : \lfloor \log_2 128 \rfloor + 1 = \lfloor 7 \rfloor + 1 = 7 + 1 = 8$$

$$n = 256 = 2^8 : \lfloor \log_2 256 \rfloor + 1 = \lfloor 8 \rfloor + 1 = 8 + 1 = 9$$

$$n = 512 = 2^9 : \lfloor \log_2 512 \rfloor + 1 = \lfloor 9 \rfloor + 1 = 9 + 1 = 10$$

Per le potenze di 2, cioè  $n = 2^m$ , si ha  $\log_2 n = m$ , e quindi il numero minimo di bit è  $m + 1$ .

4. Per  $k = 12$  bit:

(a) l'intervallo di rappresentazione è:

$$[0, 2^{12} - 1] = [0, 4,096 - 1] = [0, 4,095]$$

(b) poiché  $n = 2,048 = 2^{11}$  e  $2,048 \leq 4,095$ , si ha  $2,048 \in [0, 4,095]$ : il numero  $n = 2,048$  si può rappresentare con  $k = 12$  bit;

(c) il numero minimo di bit necessari per rappresentare  $n = 2,048 = 2^{11}$  è:

$$\lfloor \log_2 2,048 \rfloor + 1 = \lfloor 11 \rfloor + 1 = 12$$

Servono esattamente 12 bit per rappresentare  $2,048 = 2^{11}$ , perché la sua rappresentazione binaria è una cifra 1 seguita da undici cifre 0:

$$2,048 = (100000000000)_2$$

### Esercizio 1.2 — Rappresentazione dei numeri (E2)

Si considerino gli intervalli di rappresentazione e il numero minimo di cifre dei numeri naturali nel sistema binario (base  $b = 2$ ).

#### Domande

1. Completare la seguente tabella degli intervalli di rappresentazione in binario:

| $k$ (cifre) | $b$ (base) | $b^k - 1$ | intervallo di rappresentazione |
|-------------|------------|-----------|--------------------------------|
| 5           | 2          |           |                                |
| 8           | 2          |           |                                |
| 10          | 2          |           |                                |
| 12          | 2          |           |                                |

2. Per ogni numero naturale  $n$  della tabella seguente, calcolare il numero minimo di cifre necessarie per rappresentarlo in base  $b = 2$ , usando la formula  $\lfloor \log_b n \rfloor + 1$ :

| $n$ | $b$ | $\log_b n$ | $\lfloor \log_b n \rfloor + 1$ |
|-----|-----|------------|--------------------------------|
| 7   | 2   |            |                                |
| 15  | 2   |            |                                |
| 31  | 2   |            |                                |
| 100 | 2   |            |                                |
| 255 | 2   |            |                                |

3. Usando la relazione tra byte e bit, completare la tabella:

| byte | bit ( $k$ ) | $2^k$       | intervallo di rappresentazione |
|------|-------------|-------------|--------------------------------|
| 1    | 8           | $2^8 = 256$ | $[0, 255]$                     |
| 2    |             |             |                                |
| 4    |             |             |                                |

4. Convertire  $(11011011)_2$  in base dieci, scrivendo la somma completa  $\sum_{i=0}^{k-1} d_i \cdot 2^i$ .

### Soluzione

1. Intervalli di rappresentazione in binario:

| $k$ (cifre) | $b$ (base) | $b^k - 1$            | intervallo di rappresentazione |
|-------------|------------|----------------------|--------------------------------|
| 5           | 2          | $2^5 - 1 = 31$       | $[0, 31]$                      |
| 8           | 2          | $2^8 - 1 = 255$      | $[0, 255]$                     |
| 10          | 2          | $2^{10} - 1 = 1,023$ | $[0, 1,023]$                   |
| 12          | 2          | $2^{12} - 1 = 4,095$ | $[0, 4,095]$                   |

2. Numero minimo di cifre per ciascun numero naturale:

| $n$ | $b$ | $\log_b n$                 | $\lfloor \log_b n \rfloor + 1$  |
|-----|-----|----------------------------|---------------------------------|
| 7   | 2   | $\log_2 7 \approx 2.807$   | $\lfloor 2.807 \rfloor + 1 = 3$ |
| 15  | 2   | $\log_2 15 \approx 3.907$  | $\lfloor 3.907 \rfloor + 1 = 4$ |
| 31  | 2   | $\log_2 31 \approx 4.954$  | $\lfloor 4.954 \rfloor + 1 = 5$ |
| 100 | 2   | $\log_2 100 \approx 6.644$ | $\lfloor 6.644 \rfloor + 1 = 7$ |
| 255 | 2   | $\log_2 255 \approx 7.994$ | $\lfloor 7.994 \rfloor + 1 = 8$ |

Per i numeri della forma  $2^k - 1$  servono esattamente  $k$  bit:  $2^k - 1$  è il più grande numero naturale con  $k$  bit.

3. Poiché 1 byte = 8 bit:

| byte | bit ( $k$ ) | $2^k$                    | intervallo di rappresentazione |
|------|-------------|--------------------------|--------------------------------|
| 1    | 8           | $2^8 = 256$              | $[0, 255]$                     |
| 2    | 16          | $2^{16} = 65,536$        | $[0, 65,535]$                  |
| 4    | 32          | $2^{32} = 4,294,967,296$ | $[0, 4,294,967,295]$           |

4. La sequenza ha  $k = 8$  cifre:  $d_7 = 1, d_6 = 1, d_5 = 0, d_4 = 1, d_3 = 1, d_2 = 0, d_1 = 1, d_0 = 1$ . Quindi:

$$\begin{aligned}
 (11011011)_2 &= \sum_{i=0}^7 d_i \cdot 2^i \\
 &= d_0 \cdot 2^0 + d_1 \cdot 2^1 + d_2 \cdot 2^2 + d_3 \cdot 2^3 + d_4 \cdot 2^4 + d_5 \cdot 2^5 + d_6 \cdot 2^6 + d_7 \cdot 2^7 \\
 &= 1 \cdot 1 + 1 \cdot 2 + 0 \cdot 4 + 1 \cdot 8 + 1 \cdot 16 + 0 \cdot 32 + 1 \cdot 64 + 1 \cdot 128 \\
 &= 1 + 2 + 0 + 8 + 16 + 0 + 64 + 128 \\
 &= 219
 \end{aligned}$$

Pertanto,  $(11011011)_2 = (219)_{10}$ .

### Esercizio 1.3 — Rappresentazione dei numeri (E3)

Si considerino l'enumerazione completa delle sequenze di cifre in binario per un dato numero di bit e la rappresentazione binaria di alcuni numeri naturali.

#### Domande

1. Completare la tabella di tutte le sequenze di  $k = 5$  cifre in binario (base 2) e dei loro valori decimali. Mostrare il calcolo dettagliato almeno per le prime tre e le ultime tre sequenze:

| Sequenza di cifre | $b$      | $k$      | rappresentazione decimale |
|-------------------|----------|----------|---------------------------|
| $(00000)_2$       | 2        | 5        |                           |
| $(00001)_2$       | 2        | 5        |                           |
| $(00010)_2$       | 2        | 5        |                           |
| $(00011)_2$       | 2        | 5        |                           |
| $\vdots$          | $\vdots$ | $\vdots$ | $\vdots$                  |
| $(11101)_2$       | 2        | 5        |                           |
| $(11110)_2$       | 2        | 5        |                           |
| $(11111)_2$       | 2        | 5        |                           |

2. Quante sequenze diverse sono possibili con  $k = 5$  bit? Verificare la risposta con la formula  $b^k$ , dove  $b = 2$ .
3. Trovare la rappresentazione binaria dei seguenti numeri decimali, determinando quali potenze di 2 sommano al valore dato:

$$n = 21, \quad n = 50, \quad n = 63$$

4. Per ciascuna sequenza seguente, stabilire se rappresenta una potenza di 2 e, in caso affermativo, quale potenza:

$$(10000)_2, \quad (10101)_2, \quad (100000000)_2$$

### Soluzione

1. Tutte le sequenze con  $k = 5$  bit:

| Sequenza di cifre | $b$      | $k$      | rappresentazione decimale                                                                |
|-------------------|----------|----------|------------------------------------------------------------------------------------------|
| $(00000)_2$       | 2        | 5        | $(0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0)_{10} = (0)_{10}$  |
| $(00001)_2$       | 2        | 5        | $(0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0)_{10} = (1)_{10}$  |
| $(00010)_2$       | 2        | 5        | $(0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0)_{10} = (2)_{10}$  |
| $(00011)_2$       | 2        | 5        | $(3)_{10}$                                                                               |
| $(00100)_2$       | 2        | 5        | $(4)_{10}$                                                                               |
| $(00101)_2$       | 2        | 5        | $(5)_{10}$                                                                               |
| $(00110)_2$       | 2        | 5        | $(6)_{10}$                                                                               |
| $(00111)_2$       | 2        | 5        | $(7)_{10}$                                                                               |
| $(01000)_2$       | 2        | 5        | $(8)_{10}$                                                                               |
| $(01001)_2$       | 2        | 5        | $(9)_{10}$                                                                               |
| $\vdots$          | $\vdots$ | $\vdots$ |                                                                                          |
| $(11101)_2$       | 2        | 5        | $(1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0)_{10} = (29)_{10}$ |
| $(11110)_2$       | 2        | 5        | $(1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0)_{10} = (30)_{10}$ |
| $(11111)_2$       | 2        | 5        | $(1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0)_{10} = (31)_{10}$ |

2. Il numero totale di sequenze diverse con  $k = 5$  bit in base  $b = 2$  è:

$$b^k = 2^5 = 32$$

Infatti, le sequenze vanno da  $(00000)_2 = 0$  a  $(11111)_2 = 31$ , il che dà esattamente 32 valori diversi (da 0 a 31 inclusi).

3. Rappresentazione binaria dei numeri decimali:

- $n = 21$ : cerchiamo le potenze di 2 che sommano a 21, partendo dalla più grande potenza non maggiore di 21:

$$21 = 16 + 5 = 16 + 4 + 1 = 2^4 + 2^2 + 2^0$$

Quindi  $(21)_{10} = (10101)_2$ .

- $n = 50$ :

$$50 = 32 + 18 = 32 + 16 + 2 = 2^5 + 2^4 + 2^1$$

Quindi  $(50)_{10} = (110010)_2$ .

- $n = 63$ :

$$63 = 32 + 31 = 32 + 16 + 8 + 4 + 2 + 1 = 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$$

In alternativa,  $63 = 2^6 - 1$  è il più grande numero naturale con 6 bit, quindi tutti i suoi 6 bit valgono 1. Quindi  $(63)_{10} = (111111)_2$ .

## 4. Potenze di 2:

- $(10000)_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 2^4 = 16$ : è una potenza di 2, precisamente  $2^4$ .
- $(10101)_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 16 + 4 + 1 = 21$ : non è una potenza di 2 (è la somma di tre potenze di 2 diverse).
- $(100000000)_2 = 1 \cdot 2^8 = 256$ : è una potenza di 2, precisamente  $2^8$ .

Una sequenza in binario rappresenta una potenza di 2 se e solo se ha esattamente una cifra uguale a 1 e tutte le altre cifre uguali a 0.

## Esercizio 1.4 — Rappresentazione dei numeri (E4)

Si consideri la rappresentazione in complemento a due dei numeri interi in binario.

## Domande

1. Determinare l'intervallo di rappresentazione degli interi in complemento a due con i seguenti numeri di bit:

$$k = 4, \quad k = 8, \quad k = 16, \quad k = 32$$

2. Completare la seguente tabella con tutte le rappresentazioni in complemento a due con  $k = 4$  bit:

| Binario             | Decimale | Bit di segno |
|---------------------|----------|--------------|
| (0000) <sub>2</sub> |          |              |
| (0001) <sub>2</sub> |          |              |
| (0010) <sub>2</sub> |          |              |
| (0011) <sub>2</sub> |          |              |
| (0100) <sub>2</sub> |          |              |
| (0101) <sub>2</sub> |          |              |
| (0110) <sub>2</sub> |          |              |
| (0111) <sub>2</sub> |          |              |
| (1000) <sub>2</sub> |          |              |
| (1001) <sub>2</sub> |          |              |
| (1010) <sub>2</sub> |          |              |
| (1011) <sub>2</sub> |          |              |
| (1100) <sub>2</sub> |          |              |
| (1101) <sub>2</sub> |          |              |
| (1110) <sub>2</sub> |          |              |
| (1111) <sub>2</sub> |          |              |

3. Calcolare la rappresentazione in complemento a due di  $-5$  con 8 bit in tre passi: (1) rappresentazione binaria di 5, (2) inversione di tutti i bit, (3) somma di 1.
4. Verificare che con  $k = 5$  bit in complemento a due si può rappresentare l'intero  $-16$  ma non l'intero 16.

### Soluzione

1. Intervalli di rappresentazione, con la formula  $[-2^{k-1}, 2^{k-1} - 1]$ :

$$k = 4 : [-2^{4-1}, 2^{4-1} - 1] = [-2^3, 2^3 - 1] = [-8, 7]$$

$$k = 8 : [-2^{8-1}, 2^{8-1} - 1] = [-2^7, 2^7 - 1] = [-128, 127]$$

$$k = 16 : [-2^{16-1}, 2^{16-1} - 1] = [-2^{15}, 2^{15} - 1] = [-32,768, 32,767]$$

$$k = 32 : [-2^{32-1}, 2^{32-1} - 1] = [-2^{31}, 2^{31} - 1] = [-2,147,483,648, 2,147,483,647]$$

2. Tutte le rappresentazioni in complemento a due con  $k = 4$  bit: una sequenza con bit di segno 1 e valore naturale  $r$  rappresenta  $r - 16$ .

| Binario    | Decimale       | Bit di segno     |
|------------|----------------|------------------|
| $(0000)_2$ | 0              | 0 (non negativo) |
| $(0001)_2$ | 1              | 0 (non negativo) |
| $(0010)_2$ | 2              | 0 (non negativo) |
| $(0011)_2$ | 3              | 0 (non negativo) |
| $(0100)_2$ | 4              | 0 (non negativo) |
| $(0101)_2$ | 5              | 0 (non negativo) |
| $(0110)_2$ | 6              | 0 (non negativo) |
| $(0111)_2$ | 7              | 0 (non negativo) |
| $(1000)_2$ | $8 - 16 = -8$  | 1 (negativo)     |
| $(1001)_2$ | $9 - 16 = -7$  | 1 (negativo)     |
| $(1010)_2$ | $10 - 16 = -6$ | 1 (negativo)     |
| $(1011)_2$ | $11 - 16 = -5$ | 1 (negativo)     |
| $(1100)_2$ | $12 - 16 = -4$ | 1 (negativo)     |
| $(1101)_2$ | $13 - 16 = -3$ | 1 (negativo)     |
| $(1110)_2$ | $14 - 16 = -2$ | 1 (negativo)     |
| $(1111)_2$ | $15 - 16 = -1$ | 1 (negativo)     |

L'intervallo di rappresentazione con 4 bit è  $[-8, 7]$ .

3. Rappresentazione in complemento a due di  $-5$  con 8 bit:

- Passo 1: rappresentazione binaria di 5 con 8 bit:  $(00000101)_2$ .

- Passo 2: inversione di tutti i bit:  $(11111010)_2$ .
- Passo 3: somma di 1 (la prima riga contiene i riporti; il riporto sopra una colonna proviene dalla colonna alla sua destra):

|         |   |   |   |   |   |   |   |
|---------|---|---|---|---|---|---|---|
| riporto | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|         | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
|         | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| somma   | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

Pertanto, la rappresentazione in complemento a due di  $-5$  con 8 bit è  $(11111011)_2$ .  
Verifica:

$$2^8 - 5 = 256 - 5 = 251 = (11111011)_2$$

4. Con  $k = 5$  bit l'intervallo di rappresentazione è:

$$[-2^{5-1}, 2^{5-1} - 1] = [-2^4, 2^4 - 1] = [-16, 15]$$

- Poiché  $-16 \in [-16, 15]$ , l'intero  $-16$  si può rappresentare, e la sua rappresentazione è:

$$2^5 - 16 = 32 - 16 = 16 = (10000)_2$$

- Poiché  $16 > 15$ , si ha  $16 \notin [-16, 15]$ : l'intero 16 non si può rappresentare con 5 bit in complemento a due.

Il complemento a due rappresenta un intero negativo in più rispetto a quelli positivi, perché l'intervallo  $[-2^{k-1}, 2^{k-1} - 1]$  è asimmetrico.

### Esercizio 1.5 — Rappresentazione dei numeri (E5)

Si considerino la somma di numeri interi in complemento a due, il rilevamento degli overflow e il confronto tra modulo e segno e complemento a due.

#### Domande

1. Eseguire le seguenti somme in complemento a due con  $k = 5$  bit, mostrando le rappresentazioni binarie e la somma in colonna, e verificare i risultati:

$$(-7) + 3, \quad (-8) + 8, \quad 5 + 6$$

2. Stabilire se si verifica un overflow nelle seguenti somme con  $k = 5$  bit, e spiegarne il motivo:

$$7 + 7, \quad (-8) + (-8), \quad 7 + (-8), \quad 9 + 8$$

3. Confrontare le rappresentazioni in modulo e segno e in complemento a due con  $k = 4$  bit:

- (a) qual è l'intervallo di rappresentazione di ciascuna?
- (b) quante rappresentazioni ha lo zero in ciascun sistema?
- (c) quale sistema è migliore per le operazioni aritmetiche, e perché?
4. Con  $k = 8$  bit in complemento a due, calcolare le rappresentazioni di  $-64$  e  $-128$ , usando la formula  $2^k + n$  per gli interi negativi  $n$ .

### Soluzione

Nelle somme in colonna, la prima riga contiene i riporti: il riporto scritto sopra una colonna proviene dalla colonna alla sua destra, e la colonna più a sinistra riceve il riporto uscente dalla posizione più significativa, che viene scartato.

1. Con  $k = 5$  bit l'intervallo di rappresentazione è  $[-16, 15]$ .

- $(-7) + 3 = -4$ . Rappresentazioni:  $-7 \rightarrow 2^5 - 7 = 25 = (11001)_2$  e  $3 = (00011)_2$ .

|         |   |   |   |   |   |
|---------|---|---|---|---|---|
| riporto | 0 | 0 | 0 | 1 | 1 |
|         |   | 1 | 1 | 0 | 0 |
|         |   | 0 | 0 | 0 | 1 |
| somma   | 0 | 1 | 1 | 1 | 0 |

Il risultato  $(11100)_2$  ha valore naturale 28, quindi rappresenta  $28 - 32 = -4$ .

- $(-8) + 8 = 0$ . Rappresentazioni:  $-8 \rightarrow 2^5 - 8 = 24 = (11000)_2$  e  $8 = (01000)_2$ .

|         |   |   |   |   |   |
|---------|---|---|---|---|---|
| riporto | 1 | 1 | 0 | 0 | 0 |
|         |   | 1 | 1 | 0 | 0 |
|         |   | 0 | 1 | 0 | 0 |
| somma   | 1 | 0 | 0 | 0 | 0 |

Il risultato è  $(00000)_2 = 0$ .

- $5 + 6 = 11$ . Rappresentazioni:  $5 = (00101)_2$  e  $6 = (00110)_2$ .

|         |   |   |   |   |   |
|---------|---|---|---|---|---|
| riporto | 0 | 0 | 1 | 0 | 0 |
|         |   | 0 | 0 | 1 | 0 |
|         |   | 0 | 0 | 1 | 1 |
| somma   | 0 | 0 | 1 | 0 | 1 |

Il risultato è  $(01011)_2 = 11$ .

2. Rilevamento dell'overflow con  $k = 5$  bit (intervallo  $[-16, 15]$ ):

- $7 + 7$  (atteso 14). Rappresentazioni:  $7 = (00111)_2$ .

|         |   |   |   |   |   |
|---------|---|---|---|---|---|
| riporto | 0 | 0 | 1 | 1 | 1 |
|         |   | 0 | 0 | 1 | 1 |
|         |   | 0 | 0 | 1 | 1 |
| somma   | 0 | 0 | 1 | 1 | 0 |

Risultato:  $(01110)_2 = 14$ : entrambi gli operandi sono positivi e il risultato è positivo: nessun overflow ( $14 \leq 15$ ).

- $(-8) + (-8)$  (atteso  $-16$ ). Rappresentazioni:  $-8 \rightarrow 2^5 - 8 = 24 = (11000)_2$ .

|         |   |   |   |   |   |   |
|---------|---|---|---|---|---|---|
| riporto | 1 | 1 | 0 | 0 | 0 |   |
|         |   | 1 | 1 | 0 | 0 | 0 |
|         |   | 1 | 1 | 0 | 0 | 0 |
| somma   | 1 | 1 | 0 | 0 | 0 | 0 |

Risultato:  $(10000)_2 = -16$ : entrambi gli operandi sono negativi e il risultato è negativo: nessun overflow ( $-16 \geq -16$ ). Il riporto uscente viene scartato.

- $7 + (-8)$  (atteso  $-1$ ). Rappresentazioni:  $7 = (00111)_2$  e  $-8 \rightarrow 2^5 - 8 = 24 = (11000)_2$ .

|         |   |   |   |   |   |   |
|---------|---|---|---|---|---|---|
| riporto | 0 | 0 | 0 | 0 | 0 |   |
|         |   | 0 | 0 | 1 | 1 | 1 |
|         |   | 1 | 1 | 0 | 0 | 0 |
| somma   | 0 | 1 | 1 | 1 | 1 | 1 |

Risultato:  $(11111)_2 = -1$ : gli operandi hanno segni diversi, quindi non può verificarsi un overflow.

- $9 + 8$  (atteso  $17$ ). Rappresentazioni:  $9 = (01001)_2$  e  $8 = (01000)_2$ .

|         |   |   |   |   |   |   |
|---------|---|---|---|---|---|---|
| riporto | 0 | 1 | 0 | 0 | 0 |   |
|         |   | 0 | 1 | 0 | 0 | 1 |
|         |   | 0 | 1 | 0 | 0 | 0 |
| somma   | 0 | 1 | 0 | 0 | 0 | 1 |

Risultato:  $(10001)_2 = -15$ : entrambi gli operandi sono positivi, ma il risultato ha bit di segno 1: overflow. Infatti  $9 + 8 = 17 > 15$  non si può rappresentare con 5 bit.

Regola: si ha overflow solo quando si sommano due interi con lo stesso segno e il risultato ha il bit di segno opposto.

### 3. Confronto tra modulo e segno e complemento a due con $k = 4$ bit:

- (a) intervalli di rappresentazione:

$$\text{modulo e segno: } [-(2^{4-1} - 1), 2^{4-1} - 1] = [-7, 7]$$

$$\text{complemento a due: } [-2^{4-1}, 2^{4-1} - 1] = [-8, 7]$$

il complemento a due rappresenta un intero in più,  $-8$ ;

- (b) il modulo e segno ha due rappresentazioni dello zero,  $(0000)_2 = +0$  e  $(1000)_2 = -0$ ; il complemento a due ne ha una sola,  $(0000)_2 = 0$ ;
- (c) il complemento a due è migliore per le operazioni aritmetiche, perché:
- lo zero ha una rappresentazione unica;
  - somma e sottrazione si eseguono come per i numeri naturali (modulo  $2^k$ ), con gli stessi circuiti usati per i numeri senza segno;

- il bit di segno non richiede un trattamento speciale, quindi i circuiti sono più semplici ed efficienti.

4. Rappresentazioni con  $k = 8$  bit:

$$-64: \quad 2^k + n = 2^8 + (-64) = 256 - 64 = 192 = (11000000)_2$$

$$-128: \quad 2^k + n = 2^8 + (-128) = 256 - 128 = 128 = (10000000)_2$$

L'intero  $-128 = -2^{8-1}$  è il più piccolo intero rappresentabile con 8 bit in complemento a due.

### Esercizio 1.6 — Rappresentazione dei numeri (E6)

Si consideri la rappresentazione dei numeri reali in virgola mobile, secondo lo standard IEEE 754.

#### Domande

1. Scrivere il numero reale 13.25 in binario e in notazione scientifica binaria normalizzata, e fornire i suoi 32 bit in IEEE 754 a singola precisione.
2. Determinare il numero reale rappresentato dai seguenti 32 bit in IEEE 754 a singola precisione:

0 01111100 010000000000000000000000

3. Usando il formato a doppia precisione (1 bit di segno, 11 bit di esponente con polarizzazione 1023, 52 bit di parte frazionaria), calcolare il più grande numero finito e il più piccolo numero normalizzato positivo. I valori  $E = 0$  e  $E = 2^{11} - 1 = 2,047$  dell'esponente polarizzato sono riservati.
4. In doppia precisione, sommare dieci volte il numero 0.1 a una variabile inizializzata a 0 dà 0.9999999999999999 e non 1. Spiegare perché.

#### Soluzione

1. Rappresentazione di 13.25:

- Passo 1 (binario):  $13.25 = 8 + 4 + 1 + 0.25 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$ , quindi:

$$13.25 = (1101.01)_2$$

- Passo 2 (normalizzazione): la virgola si sposta di tre posizioni a sinistra:

$$(1101.01)_2 = (1.10101)_2 \cdot 2^3$$

quindi  $s = 0$ ,  $e = 3$  e la parte frazionaria inizia con 10101.



## Pseudocodice

### Esercizio 2.1 — Pseudocodice (E1)

Dato un array  $a[1..n]$  di  $n \geq 1$  numeri interi, si consideri il problema di determinare il valore minimo dell'array.

#### Domande

1. Scrivere lo pseudocodice di un algoritmo che trova il valore minimo in un array  $a$  di  $n$  numeri interi. L'algoritmo deve:
  - ricevere in input un array  $a[1..n]$  di  $n \geq 1$  numeri interi;
  - restituire il valore minimo  $\min$  dell'array;
  - usare una sola scansione dell'array.
2. Eseguire l'algoritmo passo per passo sulle seguenti istanze:
  - Istanza 1:  $a = [9, 12, 3, 16, 19, 11, 12]$  con  $n = 7$ ;
  - Istanza 2:  $a = [15, 7, 22, 4]$  con  $n = 4$ ;
  - Istanza 3:  $a = [100]$  con  $n = 1$ .

Per ogni istanza, mostrare lo stato iniziale delle variabili, il loro stato dopo ogni iterazione e il risultato finale.

3. Scrivere lo pseudocodice di un algoritmo che trova sia il valore massimo sia il valore minimo con una sola scansione dell'array.

#### Soluzione

1. Pseudocodice per trovare il valore minimo in un array:

**Algorithm:** FindMinimum**Input** : Un array  $a[1..n]$  di  $n \geq 1$  numeri interi**Output:** Il valore minimo nell'array  $a$ 

```

1 function FindMinimum( $a, n$ )
2    $min := a[1]$ 
3   for  $i := 2$  to  $n$  do
4     if  $a[i] < min$  then
5        $min \leftarrow a[i]$ 
6     end
7   end
8   return  $min$ 
9 end

```

- Inizializziamo  $min$  ad  $a[1]$ , il primo elemento (esiste, poiché  $n \geq 1$ ).
- Scorriamo ogni elemento restante dell'array, confrontandolo con il minimo corrente.
- Se l'elemento corrente è più piccolo, aggiorniamo il minimo.
- Infine, restituiamo il valore minimo trovato.

2. Esecuzione passo per passo sulle istanze.

**Istanza 1:**  $a = [9, 12, 3, 16, 19, 11, 12]$ ,  $n = 7$

| $i$ | $a[i]$ | $a[i] < min?$ | $min$ |
|-----|--------|---------------|-------|
| –   | –      | –             | 9     |
| 2   | 12     | No            | 9     |
| 3   | 3      | Sì            | 3     |
| 4   | 16     | No            | 3     |
| 5   | 19     | No            | 3     |
| 6   | 11     | No            | 3     |
| 7   | 12     | No            | 3     |

**Restituisce:**  $min = 3$

**Istanza 2:**  $a = [15, 7, 22, 4]$ ,  $n = 4$

| $i$ | $a[i]$ | $a[i] < min?$ | $min$ |
|-----|--------|---------------|-------|
| –   | –      | –             | 15    |
| 2   | 7      | Sì            | 7     |
| 3   | 22     | No            | 7     |
| 4   | 4      | Sì            | 4     |

**Restituisce:**  $min = 4$

**Istanza 3:**  $a = [100]$ ,  $n = 1$

| $i$                                                   | $a[i]$ | $a[i] < min?$ | $min$ |
|-------------------------------------------------------|--------|---------------|-------|
| –                                                     | –      | –             | 100   |
| nessuna iterazione: il ciclo parte da $i = 2 > n = 1$ |        |               |       |

**Restituisce:**  $min = 100$

3. Pseudocodice per trovare sia il massimo sia il minimo con una sola scansione:

**Algorithm:** FindMaxMin

**Input** : Un array  $a[1..n]$  di  $n \geq 1$  numeri interi

**Output:** La coppia  $(max, min)$ , dove  $max$  è il massimo e  $min$  è il minimo di  $a$

```

1 function FindMaxMin( $a, n$ )
2    $max := a[1]$ 
3    $min := a[1]$ 
4   for  $i := 2$  to  $n$  do
5     if  $a[i] > max$  then
6        $max \leftarrow a[i]$ 
7     end
8     if  $a[i] < min$  then
9        $min \leftarrow a[i]$ 
10    end
11  end
12  return  $max, min$ 
13 end

```

Ad esempio, sull'Istanza 1 ( $a = [9, 12, 3, 16, 19, 11, 12]$ ) l'algoritmo restituisce  $(19, 3)$ .

## Esercizio 2.2 — Pseudocodice (E2)

Dato un array  $a[1..n]$  di  $n \geq 1$  numeri interi, si consideri il problema di calcolare la somma dei suoi valori.

### Domande

1. Scrivere lo pseudocodice di un algoritmo che calcola la somma di tutti gli elementi di un array  $a$  di  $n$  numeri interi. L'algoritmo deve:
  - ricevere in input un array  $a[1..n]$  di  $n \geq 1$  numeri interi;

- restituire la somma  $sum$  di tutti gli elementi;
  - inizializzare una variabile  $sum$  a 0 e sommarle ogni elemento.
2. Eseguire l'algoritmo passo per passo sulle seguenti istanze:
- Istanza 1:  $a = [5, 12, 3, 8]$  con  $n = 4$ ;
  - Istanza 2:  $a = [10, 20, 30]$  con  $n = 3$ ;
  - Istanza 3:  $a = [7]$  con  $n = 1$ .

Per ogni istanza, mostrare il valore iniziale della somma, il suo valore dopo l'aggiunta di ogni elemento e il risultato finale.

3. Modificare lo pseudocodice per calcolare sia la somma sia la media  $avg = \frac{sum}{n}$  degli elementi, con una sola scansione dell'array.

### Soluzione

1. Pseudocodice per calcolare la somma di tutti gli elementi di un array:

#### Algorithm: ArraySum

**Input** : Un array  $a[1..n]$  di  $n \geq 1$  numeri interi

**Output**: La somma di tutti gli elementi dell'array  $a$

```

1 function ArraySum( $a, n$ )
2    $sum := 0$ 
3   for  $i := 1$  to  $n$  do
4      $sum \leftarrow sum + a[i]$ 
5   end
6   return  $sum$ 
7 end
```

- Inizializziamo  $sum$  a 0.
  - Scorriamo ogni elemento dell'array, aggiungendolo alla somma parziale  $sum$ .
  - Infine, restituiamo la somma totale  $sum$ .
2. Esecuzione passo per passo sulle istanze.

**Istanza 1:**  $a = [5, 12, 3, 8]$ ,  $n = 4$

| $i$ | $a[i]$ | $sum$ |
|-----|--------|-------|
| —   | —      | 0     |
| 1   | 5      | 5     |
| 2   | 12     | 17    |
| 3   | 3      | 20    |
| 4   | 8      | 28    |

**Restituisce:**  $sum = 28$

**Istanza 2:**  $a = [10, 20, 30]$ ,  $n = 3$

| $i$ | $a[i]$ | $sum$ |
|-----|--------|-------|
| –   | –      | 0     |
| 1   | 10     | 10    |
| 2   | 20     | 30    |
| 3   | 30     | 60    |

**Restituisce:**  $sum = 60$

**Istanza 3:**  $a = [7]$ ,  $n = 1$

| $i$ | $a[i]$ | $sum$ |
|-----|--------|-------|
| –   | –      | 0     |
| 1   | 7      | 7     |

**Restituisce:**  $sum = 7$

3. Pseudocodice per calcolare sia la somma sia la media:

**Algorithm:** ArraySumAndAverage

**Input :** Un array  $a[1..n]$  di  $n \geq 1$  numeri interi

**Output:** La coppia  $(sum, avg)$ , dove  $sum$  è il totale e  $avg$  è la media degli elementi

```

1 function ArraySumAndAverage( $a, n$ )
2    $sum := 0$ 
3   for  $i := 1$  to  $n$  do
4      $sum \leftarrow sum + a[i]$ 
5   end
6    $avg := sum/n$ 
7   return  $sum, avg$ 
8 end
```

La preconditione  $n \geq 1$  garantisce che la divisione  $sum/n$  sia definita. Ad esempio, sull'Istanza 1 ( $a = [5, 12, 3, 8]$ ,  $n = 4$ ) l'algoritmo restituisce  $(28, 7)$ , poiché la somma è 28 e la media è  $28/4 = 7$ .

**Esercizio 2.3 — Pseudocodice (E3)**

Dato un array bidimensionale  $b[1..m][1..n]$  di  $m \times n$  numeri interi e un valore di soglia  $k$ , si consideri il problema di contare il numero dei suoi valori strettamente maggiori di  $k$ .

**Domande**

1. Scrivere lo pseudocodice di un algoritmo che conta quanti elementi di  $b$  sono strettamente maggiori della soglia  $k$ . L'algoritmo deve:
  - ricevere in input un array bidimensionale  $b[1..m][1..n]$ , le dimensioni  $m$  e  $n$ , e la soglia  $k$ ;
  - restituire il numero di elementi che soddisfano  $b[i][j] > k$ ;
  - scorrere tutte le posizioni dell'array con due cicli annidati.
2. Eseguire l'algoritmo passo per passo sulle seguenti istanze, mostrando il valore di *count* dopo ogni aggiornamento:
  - Istanza 1:  $b = \begin{bmatrix} 3 & 7 & 1 \\ 5 & 2 & 8 \end{bmatrix}$ ,  $m = 2$ ,  $n = 3$ ,  $k = 4$ ;
  - Istanza 2:  $b = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ ,  $m = 3$ ,  $n = 3$ ,  $k = 5$ .
3. Modificare lo pseudocodice in modo che, oltre al conteggio, l'algoritmo restituisca anche un array di coppie  $(i, j)$  che elenca le posizioni di tutti gli elementi dell'array bidimensionale maggiori di  $k$ .

**Soluzione**

1. Pseudocodice per contare gli elementi maggiori di una soglia:

**Algorithm:** CountGreaterThan

**Input** : Un array bidimensionale  $b[1..m][1..n]$ , le dimensioni  $m$  e  $n$ , e una soglia  $k$

**Output**: Il numero  $count$  di elementi con  $b[i][j] > k$

```

1 function CountGreaterThan( $b, m, n, k$ )
2    $count := 0$ 
3   for  $i := 1$  to  $m$  do
4     for  $j := 1$  to  $n$  do
5       if  $b[i][j] > k$  then
6          $count \leftarrow count + 1$ 
7       end
8     end
9   end
10  return  $count$ 
11 end

```

- Inizializziamo  $count$  a 0.
- Il ciclo esterno scorre le righe  $i = 1, 2, \dots, m$ , il ciclo interno le colonne  $j = 1, 2, \dots, n$ .
- Per ogni elemento  $b[i][j]$ , se supera  $k$  incrementiamo  $count$ .
- Infine, restituiamo  $count$ .

2. Esecuzione passo per passo sulle istanze.

**Istanza 1:**  $b = \begin{bmatrix} 3 & 7 & 1 \\ 5 & 2 & 8 \end{bmatrix}$ ,  $m = 2$ ,  $n = 3$ ,  $k = 4$

| $i$ | $j$ | $b[i][j]$ | $b[i][j] > 4?$ | $count$ |
|-----|-----|-----------|----------------|---------|
| –   | –   | –         | –              | 0       |
| 1   | 1   | 3         | No             | 0       |
| 1   | 2   | 7         | Sì             | 1       |
| 1   | 3   | 1         | No             | 1       |
| 2   | 1   | 5         | Sì             | 2       |
| 2   | 2   | 2         | No             | 2       |
| 2   | 3   | 8         | Sì             | 3       |

**Restituisce:**  $count = 3$  (elementi 7, 5, 8)

**Istanza 2:**  $b = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ ,  $m = 3$ ,  $n = 3$ ,  $k = 5$

| $i$ | $j$ | $b[i][j]$ | $b[i][j] > 5?$ | $count$ |
|-----|-----|-----------|----------------|---------|
| —   | —   | —         | —              | 0       |
| 1   | 1   | 1         | No             | 0       |
| 1   | 2   | 2         | No             | 0       |
| 1   | 3   | 3         | No             | 0       |
| 2   | 1   | 4         | No             | 0       |
| 2   | 2   | 5         | No             | 0       |
| 2   | 3   | 6         | Sì             | 1       |
| 3   | 1   | 7         | Sì             | 2       |
| 3   | 2   | 8         | Sì             | 3       |
| 3   | 3   | 9         | Sì             | 4       |

**Restituisce:**  $count = 4$  (elementi 6, 7, 8, 9)

3. Pseudocodice che restituisce anche le posizioni degli elementi maggiori di  $k$ . L'array  $pos$  è dichiarato con  $m \cdot n$  elementi, il massimo numero possibile di posizioni; ne vengono usati solo i primi  $count$  elementi.

**Algorithm:** CountAndListGreaterThan

**Input :** Un array bidimensionale  $b[1..m][1..n]$ , le dimensioni  $m$  e  $n$ , e una soglia  $k$

**Output:** Il numero  $count$  e l'array  $pos[1..count]$  delle coppie  $(i, j)$  con  $b[i][j] > k$

```

1 function CountAndListGreaterThan( $b, m, n, k$ )
2    $count := 0$ 
3    $pos := \text{new array}[1..m \cdot n]$ 
4   for  $i := 1$  to  $m$  do
5     for  $j := 1$  to  $n$  do
6       if  $b[i][j] > k$  then
7          $count \leftarrow count + 1$ 
8          $pos[count] \leftarrow (i, j)$ 
9       end
10    end
11  end
12  return  $count, pos[1..count]$ 
13 end

```

Per l'Istanza 1, l'algoritmo restituisce  $count = 3$  e  $pos = [(1, 2), (2, 1), (2, 3)]$ ; per l'Istanza 2, restituisce  $count = 4$  e  $pos = [(2, 3), (3, 1), (3, 2), (3, 3)]$ .

**Esercizio 2.4 — Pseudocodice (E4)**

Dato un array bidimensionale  $b[1..m][1..n]$  di  $m \times n$  numeri interi, con  $m \geq 1$  e  $n \geq 1$ , si consideri il problema di trovare il valore minimo di ciascuna colonna.

**Domande**

1. Scrivere lo pseudocodice di un algoritmo che calcola il minimo di ciascuna colonna dell'array bidimensionale  $b$ . L'algoritmo deve:
  - ricevere in input un array bidimensionale  $b[1..m][1..n]$  e le dimensioni  $m, n$ ;
  - restituire un array  $colMin[1..n]$ , dove l'elemento  $colMin[j]$  è il minimo della colonna  $j$ ;
  - scorrere le colonne e le righe con due cicli annidati.
2. Eseguire l'algoritmo passo per passo sulle seguenti istanze, mostrando ogni aggiornamento dell'array  $colMin$ :
  - Istanza 1:  $b = \begin{bmatrix} 4 & 9 & 2 \\ 7 & 1 & 6 \\ 3 & 5 & 8 \end{bmatrix}$ ,  $m = 3$ ,  $n = 3$ ;
  - Istanza 2:  $b = \begin{bmatrix} 8 & 3 \\ 5 & 7 \\ 2 & 4 \end{bmatrix}$ ,  $m = 3$ ,  $n = 2$ .
3. Modificare lo pseudocodice in modo che, oltre al valore minimo di ciascuna colonna, l'algoritmo restituisca anche l'indice di riga  $rowIdx[j]$  della prima posizione in cui si trova il minimo della colonna  $j$ .

**Soluzione**

1. Pseudocodice per calcolare i minimi delle colonne:

**Algorithm:** ColumnMinima

**Input** : Un array bidimensionale  $b[1..m][1..n]$ , le dimensioni  $m \geq 1$  e  $n \geq 1$

**Output**: Un array  $colMin[1..n]$ , dove  $colMin[j]$  è il minimo della colonna  $j$

```

1 function ColumnMinima(b, m, n)
2   colMin := new array[1..n]
3   for j := 1 to n do
4     colMin[j] ← b[1][j]
5     for i := 2 to m do
6       if b[i][j] < colMin[j] then
7         colMin[j] ← b[i][j]
8       end
9     end
10  end
11  return colMin
12 end

```

- Il ciclo esterno scorre le colonne  $j = 1, 2, \dots, n$ .
- Inizializziamo  $colMin[j]$  a  $b[1][j]$ , il primo elemento della colonna  $j$ .
- Il ciclo interno scorre le righe  $i = 2, 3, \dots, m$ , confrontando ogni elemento con il minimo corrente della colonna, e aggiorna  $colMin[j]$  quando trova un elemento più piccolo.
- Infine, restituiamo l'array  $colMin$ .

2. Esecuzione passo per passo sulle istanze.

**Istanza 1:**  $b = \begin{bmatrix} 4 & 9 & 2 \\ 7 & 1 & 6 \\ 3 & 5 & 8 \end{bmatrix}$ ,  $m = 3$ ,  $n = 3$

Colonna  $j = 1$

| $i$ | $b[i][1]$ | $<? colMin[1]$ |   |
|-----|-----------|----------------|---|
| —   | —         | —              | 4 |
| 2   | 7         | No             | 4 |
| 3   | 3         | Sì             | 3 |

Colonna  $j = 2$

| $i$ | $b[i][2]$ | $<? colMin[2]$ |   |
|-----|-----------|----------------|---|
| —   | —         | —              | 9 |
| 2   | 1         | Sì             | 1 |
| 3   | 5         | No             | 1 |

Colonna  $j = 3$ 

| $i$ | $b[i][3]$ | $<? \text{ colMin}[3]$ |   |
|-----|-----------|------------------------|---|
| –   | –         | –                      | 2 |
| 2   | 6         | No                     | 2 |
| 3   | 8         | No                     | 2 |

La colonna  $<?$  riporta l'esito del confronto  $b[i][j] < \text{colMin}[j]$ .

**Restituisce:**  $\text{colMin} = [3, 1, 2]$

**Istanza 2:**  $b = \begin{bmatrix} 8 & 3 \\ 5 & 7 \\ 2 & 4 \end{bmatrix}$ ,  $m = 3$ ,  $n = 2$

Colonna  $j = 1$ 

| $i$ | $b[i][1]$ | $<? \text{ colMin}[1]$ |   |
|-----|-----------|------------------------|---|
| –   | –         | –                      | 8 |
| 2   | 5         | Sì                     | 5 |
| 3   | 2         | Sì                     | 2 |

Colonna  $j = 2$ 

| $i$ | $b[i][2]$ | $<? \text{ colMin}[2]$ |   |
|-----|-----------|------------------------|---|
| –   | –         | –                      | 3 |
| 2   | 7         | No                     | 3 |
| 3   | 4         | No                     | 3 |

**Restituisce:**  $\text{colMin} = [2, 3]$

3. Pseudocodice che restituisce anche l'indice di riga del minimo di ciascuna colonna.  
L'aggiornamento usa il confronto stretto  $<$ , quindi in caso di parità viene mantenuta la prima riga.

**Algorithm:** ColumnMinimaWithIndex

**Input** : Un array bidimensionale  $b[1..m][1..n]$  e le dimensioni  $m \geq 1$  e  $n \geq 1$

**Output**: Gli array  $colMin[1..n]$  e  $rowIdx[1..n]$

```

1 function ColumnMinimaWithIndex( $b, m, n$ )
2    $colMin := \text{new array}[1..n]$ 
3    $rowIdx := \text{new array}[1..n]$ 
4   for  $j := 1$  to  $n$  do
5      $colMin[j] \leftarrow b[1][j]$ 
6      $rowIdx[j] \leftarrow 1$ 
7     for  $i := 2$  to  $m$  do
8       if  $b[i][j] < colMin[j]$  then
9          $colMin[j] \leftarrow b[i][j]$ 
10         $rowIdx[j] \leftarrow i$ 
11      end
12    end
13  end
14  return  $colMin, rowIdx$ 
15 end

```

Per l'Istanza 1, l'algoritmo restituisce  $colMin = [3, 1, 2]$  e  $rowIdx = [3, 2, 1]$ ; per l'Istanza 2, restituisce  $colMin = [2, 3]$  e  $rowIdx = [3, 1]$ .

**Esercizio 2.5 — Pseudocodice (E5)**

Dato un array bidimensionale  $b[1..m][1..n]$  di  $m \times n$  numeri interi, si consideri il problema di calcolarne l'array bidimensionale trasposto, cioè l'array bidimensionale  $c[1..n][1..m]$  con  $c[j][i] = b[i][j]$ .

**Domande**

- Scrivere lo pseudocodice di un algoritmo che calcola la trasposta di un array bidimensionale  $b[1..m][1..n]$ . L'algoritmo deve:
  - ricevere in input un array bidimensionale  $b[1..m][1..n]$  e le dimensioni  $m, n$ ;
  - allocare un nuovo array bidimensionale  $c[1..n][1..m]$  per il risultato;
  - restituire l'array bidimensionale trasposto  $c$ , tale che  $c[j][i] = b[i][j]$ .
- Eseguire l'algoritmo passo per passo sulle seguenti istanze, mostrando l'array bidimensionale  $c$  che viene riempito elemento per elemento:
  - Istanza 1:  $b = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ ,  $m = 2$ ,  $n = 3$ ;

- Istanza 2:  $b = \begin{bmatrix} 7 & 2 \\ 3 & 8 \\ 1 & 5 \end{bmatrix}$ ,  $m = 3$ ,  $n = 2$ .

3. Per un array bidimensionale quadrato ( $m = n$ ), scrivere lo pseudocodice di un algoritmo che esegue la trasposizione sul posto (senza allocare un nuovo array bidimensionale), scambiando gli elementi  $b[i][j]$  e  $b[j][i]$  per ogni  $i < j$ .

### Soluzione

1. Pseudocodice per la trasposizione di un array bidimensionale:

#### Algorithm: Transpose

**Input** : Un array bidimensionale  $b[1..m][1..n]$ , le dimensioni  $m$  e  $n$

**Output**: L'array bidimensionale trasposto  $c[1..n][1..m]$

```

1 function Transpose( $b, m, n$ )
2    $c := \text{new array}[1..n][1..m]$ 
3   for  $i := 1$  to  $m$  do
4     for  $j := 1$  to  $n$  do
5        $c[j][i] \leftarrow b[i][j]$ 
6     end
7   end
8   return  $c$ 
9 end

```

- Allochiamo  $c$  come un nuovo array bidimensionale  $n \times m$ .
- Il ciclo esterno scorre le righe  $i = 1, 2, \dots, m$  di  $b$ , il ciclo interno le colonne  $j = 1, 2, \dots, n$  di  $b$ .
- Ogni elemento  $b[i][j]$  viene copiato nella posizione  $c[j][i]$ ; ogni elemento di  $c$  viene assegnato esattamente una volta.
- Infine, restituiamo l'array bidimensionale trasposto  $c$ .

2. Esecuzione passo per passo sulle istanze.

**Istanza 1:**  $b = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ ,  $m = 2$ ,  $n = 3$ , quindi  $c$  è  $3 \times 2$

| $i$ | $j$ | $b[i][j]$ | assegnamento           |
|-----|-----|-----------|------------------------|
| 1   | 1   | 1         | $c[1][1] \leftarrow 1$ |
| 1   | 2   | 2         | $c[2][1] \leftarrow 2$ |
| 1   | 3   | 3         | $c[3][1] \leftarrow 3$ |
| 2   | 1   | 4         | $c[1][2] \leftarrow 4$ |
| 2   | 2   | 5         | $c[2][2] \leftarrow 5$ |
| 2   | 3   | 6         | $c[3][2] \leftarrow 6$ |

**Restituisce:**  $c = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$

**Istanza 2:**  $b = \begin{bmatrix} 7 & 2 \\ 3 & 8 \\ 1 & 5 \end{bmatrix}$ ,  $m = 3$ ,  $n = 2$ , quindi  $c$  è  $2 \times 3$

| $i$ | $j$ | $b[i][j]$ | assegnamento           |
|-----|-----|-----------|------------------------|
| 1   | 1   | 7         | $c[1][1] \leftarrow 7$ |
| 1   | 2   | 2         | $c[2][1] \leftarrow 2$ |
| 2   | 1   | 3         | $c[1][2] \leftarrow 3$ |
| 2   | 2   | 8         | $c[2][2] \leftarrow 8$ |
| 3   | 1   | 1         | $c[1][3] \leftarrow 1$ |
| 3   | 2   | 5         | $c[2][3] \leftarrow 5$ |

**Restituisce:**  $c = \begin{bmatrix} 7 & 3 & 1 \\ 2 & 8 & 5 \end{bmatrix}$

3. Trasposizione sul posto di un array bidimensionale quadrato ( $m = n$ ):

**Algorithm:** TransposeInPlace

**Input :** Un array bidimensionale quadrato  $b[1..n][1..n]$ , la dimensione  $n$

**Output:** L'array bidimensionale  $b$ , trasposto sul posto

```

1 function TransposeInPlace( $b, n$ )
2   for  $i := 1$  to  $n - 1$  do
3     for  $j := i + 1$  to  $n$  do
4        $tmp := b[i][j]$ 
5        $b[i][j] \leftarrow b[j][i]$ 
6        $b[j][i] \leftarrow tmp$ 
7     end
8   end
9   return  $b$ 
10 end
```

Il ciclo interno parte da  $j = i + 1$ , in modo che ogni coppia di posizioni simmetriche venga scambiata esattamente una volta (scambiarla due volte ripristinerebbe l'array originale) e gli elementi  $b[i][i]$  sulla diagonale non vengano mai spostati. Per  $i = n$  il ciclo interno sarebbe vuoto, quindi il ciclo esterno si ferma a  $n - 1$ . Ad esempio,

su  $b = \begin{bmatrix} 4 & 1 \\ 9 & 6 \end{bmatrix}$  ( $n = 2$ ) l'unica iterazione scambia  $b[1][2] = 1$  e  $b[2][1] = 9$ , dando

$b = \begin{bmatrix} 4 & 9 \\ 1 & 6 \end{bmatrix}$ .

**Esercizio 2.6 — Pseudocodice (E6)**

Dati un array bidimensionale  $a[1..m][1..p]$  di  $m \times p$  numeri interi e un array bidimensionale  $b[1..p][1..n]$  di  $p \times n$  numeri interi, si consideri il problema di calcolarne il prodotto, cioè l'array bidimensionale  $c[1..m][1..n]$  con:

$$c[i][j] = \sum_{k=1}^p a[i][k] \cdot b[k][j]$$

**Domande**

1. Scrivere lo pseudocodice di un algoritmo che calcola il prodotto  $c = a \cdot b$ . L'algoritmo deve:
  - ricevere in input gli array bidimensionali  $a[1..m][1..p]$  e  $b[1..p][1..n]$ , e le dimensioni  $m, p, n$ ;
  - allocare un nuovo array bidimensionale  $c[1..m][1..n]$ , inizializzato a zero;
  - usare tre cicli annidati (su  $i, j, k$ ) per calcolare il prodotto con la formula precedente;
  - restituire l'array bidimensionale prodotto  $c$ .
2. Eseguire l'algoritmo passo per passo sulle seguenti istanze, mostrando i valori di  $k$  e  $c[i][j]$  a ogni passo del ciclo più interno:
  - Istanza 1:  $a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ ,  $b = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ ,  $m = p = n = 2$ ;
  - Istanza 2:  $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & 3 \end{bmatrix}$ ,  $b = \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix}$ ,  $m = 3, p = 2, n = 2$ .

**Soluzione**

1. Pseudocodice per il prodotto di due array bidimensionali:

**Algorithm:** MatrixProduct

**Input** : Gli array bidimensionali  $a[1..m][1..p]$  e  $b[1..p][1..n]$ , le dimensioni  $m, p, n$

**Output**: L'array bidimensionale prodotto  $c[1..m][1..n]$

```

1 function MatrixProduct( $a, b, m, p, n$ )
2    $c := \text{new array}[1..m][1..n]$ 
3   for  $i := 1$  to  $m$  do
4     for  $j := 1$  to  $n$  do
5        $c[i][j] \leftarrow 0$ 
6     end
7   end
8   for  $i := 1$  to  $m$  do
9     for  $j := 1$  to  $n$  do
10      for  $k := 1$  to  $p$  do
11         $c[i][j] \leftarrow c[i][j] + a[i][k] \cdot b[k][j]$ 
12      end
13    end
14  end
15  return  $c$ 
16 end

```

- Allochiamo  $c$  come array bidimensionale  $m \times n$  e inizializziamo a zero tutti i suoi elementi con due cicli annidati.
- Il ciclo esterno scorre le righe di  $c$ ,  $i = 1, 2, \dots, m$ , e il ciclo intermedio le sue colonne,  $j = 1, 2, \dots, n$ .
- Il ciclo interno calcola la somma  $\sum_{k=1}^p a[i][k] \cdot b[k][j]$ , aggiungendo un prodotto a ogni iterazione.
- Infine, restituiamo  $c$ .

2. Esecuzione passo per passo sulle istanze.

**Istanza 1:**  $a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ ,  $b = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ ,  $m = p = n = 2$

| $i$ | $j$ | $k$ | $a[i][k]$ | $b[k][j]$ | prodotto         | $c[i][j]$ (dopo) |
|-----|-----|-----|-----------|-----------|------------------|------------------|
| 1   | 1   | 1   | 1         | 5         | $1 \cdot 5 = 5$  | 5                |
| 1   | 1   | 2   | 2         | 7         | $2 \cdot 7 = 14$ | 19               |
| 1   | 2   | 1   | 1         | 6         | $1 \cdot 6 = 6$  | 6                |
| 1   | 2   | 2   | 2         | 8         | $2 \cdot 8 = 16$ | 22               |
| 2   | 1   | 1   | 3         | 5         | $3 \cdot 5 = 15$ | 15               |
| 2   | 1   | 2   | 4         | 7         | $4 \cdot 7 = 28$ | 43               |
| 2   | 2   | 1   | 3         | 6         | $3 \cdot 6 = 18$ | 18               |
| 2   | 2   | 2   | 4         | 8         | $4 \cdot 8 = 32$ | 50               |

**Restituisce:**  $c = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$

**Istanza 2:**  $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & 3 \end{bmatrix}$ ,  $b = \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix}$ ,  $m = 3$ ,  $p = 2$ ,  $n = 2$

| $i$ | $j$ | $k$ | $a[i][k]$ | $b[k][j]$ | prodotto | $c[i][j]$ (dopo) |
|-----|-----|-----|-----------|-----------|----------|------------------|
| 1   | 1   | 1   | 1         | 4         | 4        | 4                |
| 1   | 1   | 2   | 0         | 6         | 0        | 4                |
| 1   | 2   | 1   | 1         | 5         | 5        | 5                |
| 1   | 2   | 2   | 0         | 7         | 0        | 5                |
| 2   | 1   | 1   | 0         | 4         | 0        | 0                |
| 2   | 1   | 2   | 1         | 6         | 6        | 6                |
| 2   | 2   | 1   | 0         | 5         | 0        | 0                |
| 2   | 2   | 2   | 1         | 7         | 7        | 7                |
| 3   | 1   | 1   | 2         | 4         | 8        | 8                |
| 3   | 1   | 2   | 3         | 6         | 18       | 26               |
| 3   | 2   | 1   | 2         | 5         | 10       | 10               |
| 3   | 2   | 2   | 3         | 7         | 21       | 31               |

**Restituisce:**  $c = \begin{bmatrix} 4 & 5 \\ 6 & 7 \\ 26 & 31 \end{bmatrix}$

Le prime due righe di  $a$  formano l'identità, quindi le prime due righe di  $c$  coincidono con le due righe di  $b$ .

# Algoritmi di ordinamento

Gli esercizi di questo capitolo usano i tre algoritmi del capitolo «Algoritmi di ordinamento» (Insertion Sort, Exchange Sort e Selection Sort) nella loro versione in pseudocodice, con array indicizzati a partire da 1. In tutte le soluzioni valgono le seguenti convenzioni.

- In una tabella di traccia, ogni riga mostra lo stato dell'array *dopo* l'operazione descritta nell'ultima colonna. La chiave corrente di Insertion Sort è evidenziata in **rosso**; le celle modificate da uno spostamento, da un inserimento o da uno scambio sono evidenziate in **blu**.
- Un **confronto tra chiavi** è la valutazione di un confronto tra due chiavi:  $a[j] > key$  in Insertion Sort (valutato solo quando  $j \geq 1$ ),  $a[i] > a[j]$  in Exchange Sort,  $a[j] < a[\text{minIdx}]$  in Selection Sort.
- Un **passo** è l'esecuzione di una riga dello pseudocodice; la valutazione della condizione di un ciclo **while** conta come un passo ogni volta che viene valutata, compresa l'ultima, quella che termina il ciclo.
- Per ogni  $n \in \mathbb{Z}_{\geq 1}$ :

$$\sum_{k=1}^{n-1} k = 1 + 2 + \cdots + (n-1) = \frac{n(n-1)}{2}.$$

## Esercizio 3.1 — Algoritmi di ordinamento (E1)

Consideriamo gli array

$$a = [3, 1, 2] \quad \text{e} \quad b = [2, 1, 2]$$

e i riarrangiamenti di un array, come nella definizione del problema dell'ordinamento.

### Domande

1. Elencare tutti i riarrangiamenti di  $a$  e di  $b$ . Quali di essi sono ordinati in ordine non decrescente?
2. Spiegare perché l'output del problema dell'ordinamento non è un array qualunque ordinato in ordine non decrescente: fornire un esempio di array ordinato in ordine non decrescente che non è un output corretto per  $a$ .

3. Quanti riarrangiamenti ha un array di  $n$  elementi distinti? Calcolare il numero per  $n = 5$ ,  $n = 10$  e  $n = 20$ .
4. Consideriamo l'algoritmo **BruteForceSort** che genera uno a uno i riarrangiamenti dell'array e si ferma quando ne trova uno ordinato. Quanti riarrangiamenti può esaminare nel caso peggiore per un array di  $n$  elementi distinti? Confrontare con i  $\frac{n(n-1)}{2}$  spostamenti eseguiti nel caso peggiore da Insertion Sort.

### Soluzione

1. L'array  $a = [3, 1, 2]$  ha  $3! = 6$  riarrangiamenti:

$$[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1].$$

Solo  $[1, 2, 3]$  è ordinato in ordine non decrescente.

L'array  $b = [2, 1, 2]$  contiene due volte il valore 2, quindi scambiare i due 2 dà lo stesso array. I suoi riarrangiamenti distinti sono soltanto 3:

$$[1, 2, 2], [2, 1, 2], [2, 2, 1],$$

e solo  $[1, 2, 2]$  è ordinato. In entrambi i casi il riarrangiamento ordinato è **unico**: l'output del problema dell'ordinamento è univocamente determinato dall'input.

2. L'output deve essere un **riarrangiamento dell'input** ordinato. Ad esempio, gli array  $[1, 1, 1]$  e  $[1, 2, 4]$  sono ordinati, ma non sono un output corretto per  $a = [3, 1, 2]$ , perché non contengono gli stessi elementi. Un "algoritmo" che sovrascrive tutte le componenti con 0 produce un array ordinato, ma non risolve il problema dell'ordinamento. Per questo, quando si dimostra che un algoritmo di ordinamento è corretto, bisogna mostrare anche che l'array finale contiene gli elementi originali.
3. Un array di  $n$  elementi distinti ha

$$n! = n \cdot (n-1) \cdots 2 \cdot 1$$

riarrangiamenti: ci sono  $n$  scelte per la prima posizione,  $n-1$  per la seconda, e così via. Quindi

$$5! = 120, \quad 10! = 3\,628\,800, \quad 20! = 2\,432\,902\,008\,176\,640\,000 \approx 2.4 \cdot 10^{18}.$$

4. Nel caso peggiore il riarrangiamento ordinato è l'ultimo generato, quindi **BruteForceSort** può esaminare tutti gli  $n!$  riarrangiamenti (e verificare ciascuno di essi richiede fino a  $n-1$  confronti). Insertion Sort esegue al più  $\frac{n(n-1)}{2}$  spostamenti:

| $n$ | $n!$                      | $\frac{n(n-1)}{2}$ |
|-----|---------------------------|--------------------|
| 5   | 120                       | 10                 |
| 10  | 3 628 800                 | 45                 |
| 20  | 2 432 902 008 176 640 000 | 190                |

Poiché  $n! \geq 2^{n-1}$  per ogni  $n \geq 1$ , il numero di riarrangiamenti esaminati dall'approccio a forza bruta cresce almeno in modo esponenziale con  $n$ , mentre il numero di spostamenti di Insertion Sort cresce solo in modo quadratico. Già per  $n = 20$ , anche esaminando  $10^9$  riarrangiamenti al secondo, **BruteForceSort** potrebbe impiegare

$$\frac{2.43 \cdot 10^{18}}{10^9} \text{ s} \approx 2.43 \cdot 10^9 \text{ s} \approx 77 \text{ anni.}$$

### Esercizio 3.2 — Algoritmi di ordinamento (E2)

Consideriamo il problema di **verificare** se un array  $a[1..n]$  di  $n \geq 1$  numeri naturali è ordinato in ordine non decrescente, e il seguente algoritmo.

**Algorithm:** IsSorted

**Input** : Un array  $a[1..n]$  di  $n$  numeri naturali

**Output:** **true** se  $a[1] \leq a[2] \leq \dots \leq a[n]$ , **false** altrimenti

```

1 function IsSorted( $a, n$ )
2    $i := 1$ 
3   while  $i \leq n - 1$  do
4     if  $a[i] > a[i + 1]$  then
5       return false
6     end
7      $i \leftarrow i + 1$ 
8   end
9   return true
10 end
```

#### Domande

1. Applicare **IsSorted** agli array  $[1, 3, 3, 7, 9]$ ,  $[2, 5, 4, 8]$ ,  $[9, 1, 2, 3]$  e  $[1]$ . Per ciascun array indicare l'output e il numero di passi eseguiti.
2. **IsSorted** è un algoritmo basato su confronti? Quanti confronti tra chiavi esegue nel caso peggiore?
3. Sia  $S(n)$  il numero di passi eseguiti da **IsSorted** nel caso peggiore, per un array di  $n$  elementi. Calcolare  $S(n)$  e confrontare il numero di confronti tra chiavi di **IsSorted** con il numero di confronti tra chiavi nel caso peggiore degli algoritmi di ordinamento del capitolo.
4. Dimostrare che **IsSorted** restituisce **true** se e solo se l'array  $a[1..n]$  è ordinato in ordine non decrescente.

#### Soluzione

1. Executions:

- $[1, 3, 3, 7, 9]$  ( $n = 5$ ): i confronti  $1 \leq 3$ ,  $3 \leq 3$ ,  $3 \leq 7$ ,  $7 \leq 9$  sono tutti soddisfatti;

il ciclo termina quando  $i = 5$  e l'output è **true**. Passi: riga 2 una volta, riga 3 5 volte, righe 4 e 7 4 volte ciascuna, riga 9 una volta:

$$1 + 5 + 4 + 4 + 1 = 15.$$

- $[2, 5, 4, 8]$  ( $n = 4$ ):  $a[1] = 2 \leq 5$ ; poi  $a[2] = 5 > a[3] = 4$ , quindi l'output è **false** per  $i = 2$ . Passi: riga 2 una volta, riga 3 due volte, riga 4 due volte, riga 7 una volta, riga 5 una volta:

$$1 + 2 + 2 + 1 + 1 = 7.$$

- $[9, 1, 2, 3]$  ( $n = 4$ ):  $a[1] = 9 > a[2] = 1$ , quindi l'output è **false** per  $i = 1$ . Passi: le righe 2, 3, 4, 5 una volta ciascuna:

$$1 + 1 + 1 + 1 = 4.$$

- $[1]$  ( $n = 1$ ): la condizione  $1 \leq 0$  è falsa e l'output è **true**. Passi: le righe 2, 3, 9 una volta ciascuna:

$$1 + 1 + 1 = 3.$$

2. Sì: l'algoritmo accede alle chiavi soltanto per confrontarne due (riga 4); non le modifica mai e non esegue mai operazioni aritmetiche su di esse. Nel caso peggiore esegue  $n - 1$  confronti tra chiavi, uno per ogni coppia di posizioni consecutive.

3. Il caso peggiore è quello in cui l'algoritmo non ritorna mai alla riga 5, cioè quando l'array è ordinato. Il numero di volte in cui ciascuna riga viene eseguita è:

- la riga 2 viene eseguita 1 volta;
- la riga 3 viene eseguita  $n$  volte;
- la riga 4 viene eseguita  $n - 1$  volte;
- la riga 5 non viene eseguita;
- la riga 7 viene eseguita  $n - 1$  volte;
- la riga 9 viene eseguita 1 volta.

Quindi

$$S(n) = 1 + n + (n - 1) + (n - 1) + 1 = 3n,$$

in accordo con il punto 1 ( $15 = 3 \cdot 5$  per l'array ordinato di 5 elementi,  $3 = 3 \cdot 1$  per  $n = 1$ ).

Verificare se un array è ordinato richiede al più  $n - 1$  confronti tra chiavi, un numero che cresce linearmente con  $n$ , mentre Insertion Sort, Exchange Sort e Selection Sort eseguono  $\frac{n(n-1)}{2}$  confronti tra chiavi nel caso peggiore, un numero che cresce quadraticamente con  $n$ . Per  $n = 100$ : 99 confronti contro 4950.

4. L'algoritmo restituisce **false** soltanto alla riga 5, e soltanto dopo aver trovato un indice  $i$  con  $a[i] > a[i + 1]$ : in questo caso l'array non è ordinato, perché due elementi consecutivi sono nell'ordine sbagliato. Quindi, se l'array è ordinato, l'algoritmo non

restituisce mai **false**; poiché  $i$  aumenta di 1 a ogni iterazione, il ciclo termina dopo  $n - 1$  iterazioni e l'algoritmo restituisce **true** alla riga 9.

Viceversa, se l'algoritmo restituisce **true**, il ciclo è stato completato, quindi il confronto alla riga 4 era falso per ogni  $i \in \{1, 2, \dots, n - 1\}$ :

$$a[1] \leq a[2], \quad a[2] \leq a[3], \quad \dots, \quad a[n-1] \leq a[n].$$

Concatenando queste disuguaglianze,  $a[1] \leq a[2] \leq \dots \leq a[n]$ : l'array è ordinato. Dunque **IsSorted** restituisce **true** se e solo se l'array è ordinato in ordine non decrescente.

### Esercizio 3.3 — Algoritmi di ordinamento (E3)

Consideriamo l'array di record  $(key, label)$ , in cui si confrontano solo le chiavi,

$$a = [(3, A), (1, B), (3, C), (2, D), (1, E)],$$

e gli algoritmi Insertion Sort e Selection Sort del capitolo (Insertion Sort sposta  $a[j]$  a destra finché  $a[j] > key$ ; Selection Sort scambia  $a[i]$  con il minimo di  $a[i..n]$ , scegliendo il primo minimo trovato).

#### Domande

1. Applicare Insertion Sort ad  $a$ , mostrando l'array dopo ogni iterazione del ciclo esterno. Il risultato è stabile?
2. Applicare Selection Sort ad  $a$ , mostrando l'array dopo ogni iterazione del ciclo esterno. Il risultato è stabile?
3. Uno studente sostituisce la condizione  $a[j] > key$  di Insertion Sort con  $a[j] \geq key$ . Il nuovo algoritmo è ancora corretto? È stabile?
4. Insertion Sort e Selection Sort sono in-place? Descrivere un metodo di ordinamento che non è in-place.

#### Soluzione

1. Insertion Sort (la chiave inserita è evidenziata in **rosso**):

| $i$      | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ |
|----------|--------|--------|--------|--------|--------|
| iniziale | (3, A) | (1, B) | (3, C) | (2, D) | (1, E) |
| 2        | (1, B) | (3, A) | (3, C) | (2, D) | (1, E) |
| 3        | (1, B) | (3, A) | (3, C) | (2, D) | (1, E) |
| 4        | (1, B) | (2, D) | (3, A) | (3, C) | (1, E) |
| 5        | (1, B) | (1, E) | (2, D) | (3, A) | (3, C) |

Per  $i = 3$ ,  $a[2] = (3, A)$  non è  $> 3$ , quindi  $(3, C)$  resta dopo  $(3, A)$ ; per  $i = 5$ , gli spostamenti si fermano in  $(1, B)$ , perché  $1 > 1$  è falso. Nell'output  $B$  precede  $E$  e  $A$  precede  $C$ , come nell'input: il risultato è **stabile**. In generale, Insertion Sort è stabile,

perché una chiave non viene mai spostata prima di una chiave uguale (la condizione di spostamento è una disuguaglianza stretta).

2. Selection Sort (le due posizioni scambiate sono evidenziate in **blu**):

| $i$      | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $minIdx$            |
|----------|--------|--------|--------|--------|--------|---------------------|
| iniziale | (3, A) | (1, B) | (3, C) | (2, D) | (1, E) |                     |
| 1        | (1, B) | (3, A) | (3, C) | (2, D) | (1, E) | 2                   |
| 2        | (1, B) | (1, E) | (3, C) | (2, D) | (3, A) | 5                   |
| 3        | (1, B) | (1, E) | (2, D) | (3, C) | (3, A) | 4                   |
| 4        | (1, B) | (1, E) | (2, D) | (3, C) | (3, A) | 4 (nessuno scambio) |

Le chiavi sono ordinate, ma (3, C) precede (3, A), mentre nell'input A precedeva C: il risultato **non è stabile**. Lo scambio per  $i = 2$  ha spostato (3, A) dalla posizione 2 alla posizione 5, “saltando” (3, C). Quindi Selection Sort non è stabile (un controesempio ancora più piccolo è [(2, A), (2, B), (1, C)], che diventa [(1, C), (2, B), (2, A)]).

3. Con  $a[j] \geq key$  l'algoritmo è ancora **corretto**: le chiavi sono ancora ordinate, perché  $key$  viene inserita dopo tutti gli elementi  $< key$  e prima di tutti gli elementi  $\geq key$ . Tuttavia **non è stabile**, perché  $key$  viene spostata prima delle chiavi uguali. Su  $a$ :

$$[(3, A), (1, B), (3, C), (2, D), (1, E)] \rightarrow [(1, E), (1, B), (2, D), (3, C), (3, A)],$$

dove entrambe le coppie di chiavi uguali sono invertite. Inoltre esegue più spostamenti del necessario quando ci sono chiavi uguali.

4. Entrambi gli algoritmi sono **in-place**: riarrangiano gli elementi all'interno dell'array  $a$  e usano soltanto un numero costante di variabili ausiliarie ( $i, j, key$  per Insertion Sort;  $i, j, minIdx, tmp$  per Selection Sort), indipendentemente da  $n$ .

Un metodo che **non** è in-place: creare un nuovo array  $b[1..n]$ ; per  $k \in \{1, 2, \dots, n\}$ , trovare il minimo tra gli elementi di  $a$  non ancora copiati, copiarlo in  $b[k]$  e marcarlo come usato (ad esempio con un array di  $n$  flag booleani); infine restituire  $b$ . Questo metodo usa memoria aggiuntiva proporzionale a  $n$  (l'array  $b$  e i flag), quindi non è in-place.

### Esercizio 3.4 — Algoritmi di ordinamento (E4)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [5, 2, 4, 6, 1, 3]$
2.  $a = [8, 3, 7, 4, 9, 2]$
3.  $a = [6, 5, 4, 3, 2, 1]$
4.  $a = [1, 2, 3, 4, 5, 6]$
5.  $a = [10, 5, 15, 3]$

## Domande

1. Applicare l'algoritmo **InsertionSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni iterazione del ciclo esterno, indicare la chiave da inserire, ogni spostamento eseguito dal ciclo interno e lo stato dell'array dopo l'inserimento.
2. Per ciascun array, contare i confronti tra chiavi  $a[j] > key$  e gli spostamenti eseguiti dall'algoritmo.

## Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [5, 2, 4, 6, 1, 3]$  ( $n = 6$ ).

Traccia dettagliata:

| $i$ | $j$ | $key$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                      |
|-----|-----|-------|--------|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| –   | –   | –     | 5      | 2      | 4      | 6      | 1      | 3      | stato iniziale                                              |
| 2   | 1   | 2     | 5      | 2      | 4      | 6      | 1      | 3      | $key \leftarrow a[2] = 2; j \leftarrow 1$                   |
|     | 0   | 2     | 5      | 5      | 4      | 6      | 1      | 3      | $a[1] = 5 > 2$ : sposta $a[2] \leftarrow 5; j \leftarrow 0$ |
|     | –   | 2     | 2      | 5      | 4      | 6      | 1      | 3      | $j = 0$ : esci; inserisci $a[1] \leftarrow 2$               |
| 3   | 2   | 4     | 2      | 5      | 4      | 6      | 1      | 3      | $key \leftarrow a[3] = 4; j \leftarrow 2$                   |
|     | 1   | 4     | 2      | 5      | 5      | 6      | 1      | 3      | $a[2] = 5 > 4$ : sposta $a[3] \leftarrow 5; j \leftarrow 1$ |
|     | –   | 4     | 2      | 4      | 5      | 6      | 1      | 3      | $a[1] = 2 \leq 4$ : esci; inserisci $a[2] \leftarrow 4$     |
| 4   | 3   | 6     | 2      | 4      | 5      | 6      | 1      | 3      | $key \leftarrow a[4] = 6; j \leftarrow 3$                   |
|     | –   | 6     | 2      | 4      | 5      | 6      | 1      | 3      | $a[3] = 5 \leq 6$ : esci; inserisci $a[4] \leftarrow 6$     |
| 5   | 4   | 1     | 2      | 4      | 5      | 6      | 1      | 3      | $key \leftarrow a[5] = 1; j \leftarrow 4$                   |
|     | 3   | 1     | 2      | 4      | 5      | 6      | 6      | 3      | $a[4] = 6 > 1$ : sposta $a[5] \leftarrow 6; j \leftarrow 3$ |
|     | 2   | 1     | 2      | 4      | 5      | 5      | 6      | 3      | $a[3] = 5 > 1$ : sposta $a[4] \leftarrow 5; j \leftarrow 2$ |
|     | 1   | 1     | 2      | 4      | 4      | 5      | 6      | 3      | $a[2] = 4 > 1$ : sposta $a[3] \leftarrow 4; j \leftarrow 1$ |
|     | 0   | 1     | 2      | 2      | 4      | 5      | 6      | 3      | $a[1] = 2 > 1$ : sposta $a[2] \leftarrow 2; j \leftarrow 0$ |
|     | –   | 1     | 1      | 2      | 4      | 5      | 6      | 3      | $j = 0$ : esci; inserisci $a[1] \leftarrow 1$               |
| 6   | 5   | 3     | 1      | 2      | 4      | 5      | 6      | 3      | $key \leftarrow a[6] = 3; j \leftarrow 5$                   |
|     | 4   | 3     | 1      | 2      | 4      | 5      | 6      | 6      | $a[5] = 6 > 3$ : sposta $a[6] \leftarrow 6; j \leftarrow 4$ |
|     | 3   | 3     | 1      | 2      | 4      | 5      | 5      | 6      | $a[4] = 5 > 3$ : sposta $a[5] \leftarrow 5; j \leftarrow 3$ |
|     | 2   | 3     | 1      | 2      | 4      | 4      | 5      | 6      | $a[3] = 4 > 3$ : sposta $a[4] \leftarrow 4; j \leftarrow 2$ |
|     | –   | 3     | 1      | 2      | 3      | 4      | 5      | 6      | $a[2] = 2 \leq 3$ : esci; inserisci $a[3] \leftarrow 3$     |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                 |
|-----|-----|--------|--------|--------|--------|--------|--------|------------------------|
| –   | –   | 5      | 2      | 4      | 6      | 1      | 3      | stato iniziale         |
| 2   | 2   | 2      | 5      | 4      | 6      | 1      | 3      | inserisci 2 all'inizio |
| 3   | 4   | 2      | 4      | 5      | 6      | 1      | 3      | inserisci 4 tra 2 e 5  |
| 4   | 6   | 2      | 4      | 5      | 6      | 1      | 3      | 6 già al suo posto     |
| 5   | 1   | 1      | 2      | 4      | 5      | 6      | 3      | inserisci 1 all'inizio |
| 6   | 3   | 1      | 2      | 3      | 4      | 5      | 6      | inserisci 3 tra 2 e 4  |

L'algoritmo esegue 12 confronti tra chiavi e 9 spostamenti, e l'array ordinato è  
 $[1, 2, 3, 4, 5, 6]$ .

2. Array  $a = [8, 3, 7, 4, 9, 2]$  ( $n = 6$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                      |
|-----|-----|-----|--------|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| –   | –   | –   | 8      | 3      | 7      | 4      | 9      | 2      | stato iniziale                                              |
| 2   | 1   | 3   | 8      | 3      | 7      | 4      | 9      | 2      | $key \leftarrow a[2] = 3; j \leftarrow 1$                   |
|     | 0   | 3   | 8      | 8      | 7      | 4      | 9      | 2      | $a[1] = 8 > 3$ : sposta $a[2] \leftarrow 8; j \leftarrow 0$ |
|     | –   | 3   | 3      | 8      | 7      | 4      | 9      | 2      | $j = 0$ : esci; inserisci $a[1] \leftarrow 3$               |
| 3   | 2   | 7   | 3      | 8      | 7      | 4      | 9      | 2      | $key \leftarrow a[3] = 7; j \leftarrow 2$                   |
|     | 1   | 7   | 3      | 8      | 8      | 4      | 9      | 2      | $a[2] = 8 > 7$ : sposta $a[3] \leftarrow 8; j \leftarrow 1$ |
|     | –   | 7   | 3      | 7      | 8      | 4      | 9      | 2      | $a[1] = 3 \leq 7$ : esci; inserisci $a[2] \leftarrow 7$     |
| 4   | 3   | 4   | 3      | 7      | 8      | 4      | 9      | 2      | $key \leftarrow a[4] = 4; j \leftarrow 3$                   |
|     | 2   | 4   | 3      | 7      | 8      | 8      | 9      | 2      | $a[3] = 8 > 4$ : sposta $a[4] \leftarrow 8; j \leftarrow 2$ |
|     | 1   | 4   | 3      | 7      | 7      | 8      | 9      | 2      | $a[2] = 7 > 4$ : sposta $a[3] \leftarrow 7; j \leftarrow 1$ |
|     | –   | 4   | 3      | 4      | 7      | 8      | 9      | 2      | $a[1] = 3 \leq 4$ : esci; inserisci $a[2] \leftarrow 4$     |
| 5   | 4   | 9   | 3      | 4      | 7      | 8      | 9      | 2      | $key \leftarrow a[5] = 9; j \leftarrow 4$                   |
|     | –   | 9   | 3      | 4      | 7      | 8      | 9      | 2      | $a[4] = 8 \leq 9$ : esci; inserisci $a[5] \leftarrow 9$     |
| 6   | 5   | 2   | 3      | 4      | 7      | 8      | 9      | 2      | $key \leftarrow a[6] = 2; j \leftarrow 5$                   |
|     | 4   | 2   | 3      | 4      | 7      | 8      | 9      | 9      | $a[5] = 9 > 2$ : sposta $a[6] \leftarrow 9; j \leftarrow 4$ |
|     | 3   | 2   | 3      | 4      | 7      | 8      | 8      | 9      | $a[4] = 8 > 2$ : sposta $a[5] \leftarrow 8; j \leftarrow 3$ |
|     | 2   | 2   | 3      | 4      | 7      | 7      | 8      | 9      | $a[3] = 7 > 2$ : sposta $a[4] \leftarrow 7; j \leftarrow 2$ |
|     | 1   | 2   | 3      | 4      | 4      | 7      | 8      | 9      | $a[2] = 4 > 2$ : sposta $a[3] \leftarrow 4; j \leftarrow 1$ |
|     | 0   | 2   | 3      | 3      | 4      | 7      | 8      | 9      | $a[1] = 3 > 2$ : sposta $a[2] \leftarrow 3; j \leftarrow 0$ |
|     | –   | 2   | 2      | 3      | 4      | 7      | 8      | 9      | $j = 0$ : esci; inserisci $a[1] \leftarrow 2$               |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                 |
|-----|-----|--------|--------|--------|--------|--------|--------|------------------------|
| –   | –   | 8      | 3      | 7      | 4      | 9      | 2      | stato iniziale         |
| 2   | 3   | 3      | 8      | 7      | 4      | 9      | 2      | inserisci 3 all'inizio |
| 3   | 7   | 3      | 7      | 8      | 4      | 9      | 2      | inserisci 7 tra 3 e 8  |
| 4   | 4   | 3      | 4      | 7      | 8      | 9      | 2      | inserisci 4 tra 3 e 7  |
| 5   | 9   | 3      | 4      | 7      | 8      | 9      | 2      | 9 già al suo posto     |
| 6   | 2   | 2      | 3      | 4      | 7      | 8      | 9      | inserisci 2 all'inizio |

L'algoritmo esegue 12 confronti tra chiavi e 9 spostamenti, e l'array ordinato è

$[2, 3, 4, 7, 8, 9]$ .

3. Array  $a = [6, 5, 4, 3, 2, 1]$  ( $n = 6$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                      |
|-----|-----|-----|--------|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | —   | 6      | 5      | 4      | 3      | 2      | 1      | stato iniziale                                              |
| 2   | 1   | 5   | 6      | 5      | 4      | 3      | 2      | 1      | $key \leftarrow a[2] = 5; j \leftarrow 1$                   |
|     | 0   | 5   | 6      | 6      | 4      | 3      | 2      | 1      | $a[1] = 6 > 5$ : sposta $a[2] \leftarrow 6; j \leftarrow 0$ |
|     | —   | 5   | 5      | 6      | 4      | 3      | 2      | 1      | $j = 0$ : esci; inserisci $a[1] \leftarrow 5$               |
| 3   | 2   | 4   | 5      | 6      | 4      | 3      | 2      | 1      | $key \leftarrow a[3] = 4; j \leftarrow 2$                   |
|     | 1   | 4   | 5      | 6      | 6      | 3      | 2      | 1      | $a[2] = 6 > 4$ : sposta $a[3] \leftarrow 6; j \leftarrow 1$ |
|     | 0   | 4   | 5      | 5      | 6      | 3      | 2      | 1      | $a[1] = 5 > 4$ : sposta $a[2] \leftarrow 5; j \leftarrow 0$ |
|     | —   | 4   | 4      | 5      | 6      | 3      | 2      | 1      | $j = 0$ : esci; inserisci $a[1] \leftarrow 4$               |
| 4   | 3   | 3   | 4      | 5      | 6      | 3      | 2      | 1      | $key \leftarrow a[4] = 3; j \leftarrow 3$                   |
|     | 2   | 3   | 4      | 5      | 6      | 6      | 2      | 1      | $a[3] = 6 > 3$ : sposta $a[4] \leftarrow 6; j \leftarrow 2$ |
|     | 1   | 3   | 4      | 5      | 5      | 6      | 2      | 1      | $a[2] = 5 > 3$ : sposta $a[3] \leftarrow 5; j \leftarrow 1$ |
|     | 0   | 3   | 4      | 4      | 5      | 6      | 2      | 1      | $a[1] = 4 > 3$ : sposta $a[2] \leftarrow 4; j \leftarrow 0$ |
|     | —   | 3   | 3      | 4      | 5      | 6      | 2      | 1      | $j = 0$ : esci; inserisci $a[1] \leftarrow 3$               |
| 5   | 4   | 2   | 3      | 4      | 5      | 6      | 2      | 1      | $key \leftarrow a[5] = 2; j \leftarrow 4$                   |
|     | 3   | 2   | 3      | 4      | 5      | 6      | 6      | 1      | $a[4] = 6 > 2$ : sposta $a[5] \leftarrow 6; j \leftarrow 3$ |
|     | 2   | 2   | 3      | 4      | 5      | 5      | 6      | 1      | $a[3] = 5 > 2$ : sposta $a[4] \leftarrow 5; j \leftarrow 2$ |
|     | 1   | 2   | 3      | 4      | 4      | 5      | 6      | 1      | $a[2] = 4 > 2$ : sposta $a[3] \leftarrow 4; j \leftarrow 1$ |
|     | 0   | 2   | 3      | 3      | 4      | 5      | 6      | 1      | $a[1] = 3 > 2$ : sposta $a[2] \leftarrow 3; j \leftarrow 0$ |
|     | —   | 2   | 2      | 3      | 4      | 5      | 6      | 1      | $j = 0$ : esci; inserisci $a[1] \leftarrow 2$               |
| 6   | 5   | 1   | 2      | 3      | 4      | 5      | 6      | 1      | $key \leftarrow a[6] = 1; j \leftarrow 5$                   |
|     | 4   | 1   | 2      | 3      | 4      | 5      | 6      | 6      | $a[5] = 6 > 1$ : sposta $a[6] \leftarrow 6; j \leftarrow 4$ |
|     | 3   | 1   | 2      | 3      | 4      | 5      | 5      | 6      | $a[4] = 5 > 1$ : sposta $a[5] \leftarrow 5; j \leftarrow 3$ |
|     | 2   | 1   | 2      | 3      | 4      | 4      | 5      | 6      | $a[3] = 4 > 1$ : sposta $a[4] \leftarrow 4; j \leftarrow 2$ |
|     | 1   | 1   | 2      | 3      | 3      | 4      | 5      | 6      | $a[2] = 3 > 1$ : sposta $a[3] \leftarrow 3; j \leftarrow 1$ |
|     | 0   | 1   | 2      | 2      | 3      | 4      | 5      | 6      | $a[1] = 2 > 1$ : sposta $a[2] \leftarrow 2; j \leftarrow 0$ |
|     | —   | 1   | 1      | 2      | 3      | 4      | 5      | 6      | $j = 0$ : esci; inserisci $a[1] \leftarrow 1$               |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                 |
|-----|-----|--------|--------|--------|--------|--------|--------|------------------------|
| —   | —   | 6      | 5      | 4      | 3      | 2      | 1      | stato iniziale         |
| 2   | 5   | 5      | 6      | 4      | 3      | 2      | 1      | inserisci 5 all'inizio |
| 3   | 4   | 4      | 5      | 6      | 3      | 2      | 1      | inserisci 4 all'inizio |
| 4   | 3   | 3      | 4      | 5      | 6      | 2      | 1      | inserisci 3 all'inizio |
| 5   | 2   | 2      | 3      | 4      | 5      | 6      | 1      | inserisci 2 all'inizio |
| 6   | 1   | 1      | 2      | 3      | 4      | 5      | 6      | inserisci 1 all'inizio |

L'algoritmo esegue 15 confronti tra chiavi e 15 spostamenti, e l'array ordinato è

$[1, 2, 3, 4, 5, 6]$ .

4. Array  $a = [1, 2, 3, 4, 5, 6]$  ( $n = 6$ ).

*Traccia dettagliata:*

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                  |
|-----|-----|-----|--------|--------|--------|--------|--------|--------|---------------------------------------------------------|
| —   | —   | —   | 1      | 2      | 3      | 4      | 5      | 6      | stato iniziale                                          |
| 2   | 1   | 2   | 1      | 2      | 3      | 4      | 5      | 6      | $key \leftarrow a[2] = 2; j \leftarrow 1$               |
| —   | —   | 2   | 1      | 2      | 3      | 4      | 5      | 6      | $a[1] = 1 \leq 2$ : esci; inserisci $a[2] \leftarrow 2$ |
| 3   | 2   | 3   | 1      | 2      | 3      | 4      | 5      | 6      | $key \leftarrow a[3] = 3; j \leftarrow 2$               |
| —   | —   | 3   | 1      | 2      | 3      | 4      | 5      | 6      | $a[2] = 2 \leq 3$ : esci; inserisci $a[3] \leftarrow 3$ |
| 4   | 3   | 4   | 1      | 2      | 3      | 4      | 5      | 6      | $key \leftarrow a[4] = 4; j \leftarrow 3$               |
| —   | —   | 4   | 1      | 2      | 3      | 4      | 5      | 6      | $a[3] = 3 \leq 4$ : esci; inserisci $a[4] \leftarrow 4$ |
| 5   | 4   | 5   | 1      | 2      | 3      | 4      | 5      | 6      | $key \leftarrow a[5] = 5; j \leftarrow 4$               |
| —   | —   | 5   | 1      | 2      | 3      | 4      | 5      | 6      | $a[4] = 4 \leq 5$ : esci; inserisci $a[5] \leftarrow 5$ |
| 6   | 5   | 6   | 1      | 2      | 3      | 4      | 5      | 6      | $key \leftarrow a[6] = 6; j \leftarrow 5$               |
| —   | —   | 6   | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = 5 \leq 6$ : esci; inserisci $a[6] \leftarrow 6$ |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione             |
|-----|-----|--------|--------|--------|--------|--------|--------|--------------------|
| —   | —   | 1      | 2      | 3      | 4      | 5      | 6      | stato iniziale     |
| 2   | 2   | 1      | 2      | 3      | 4      | 5      | 6      | 2 già al suo posto |
| 3   | 3   | 1      | 2      | 3      | 4      | 5      | 6      | 3 già al suo posto |
| 4   | 4   | 1      | 2      | 3      | 4      | 5      | 6      | 4 già al suo posto |
| 5   | 5   | 1      | 2      | 3      | 4      | 5      | 6      | 5 già al suo posto |
| 6   | 6   | 1      | 2      | 3      | 4      | 5      | 6      | 6 già al suo posto |

L'algoritmo esegue 5 confronti tra chiavi e 0 spostamenti, e l'array ordinato è

$[1, 2, 3, 4, 5, 6]$ .

5. Array  $a = [10, 5, 15, 3]$  ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $key$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                        |
|-----|-----|-------|--------|--------|--------|--------|---------------------------------------------------------------|
| —   | —   | —     | 10     | 5      | 15     | 3      | stato iniziale                                                |
| 2   | 1   | 5     | 10     | 5      | 15     | 3      | $key \leftarrow a[2] = 5; j \leftarrow 1$                     |
|     | 0   | 5     | 10     | 10     | 15     | 3      | $a[1] = 10 > 5$ : sposta $a[2] \leftarrow 10; j \leftarrow 0$ |
|     | —   | 5     | 5      | 10     | 15     | 3      | $j = 0$ : esci; inserisci $a[1] \leftarrow 5$                 |
| 3   | 2   | 15    | 5      | 10     | 15     | 3      | $key \leftarrow a[3] = 15; j \leftarrow 2$                    |
|     | —   | 15    | 5      | 10     | 15     | 3      | $a[2] = 10 \leq 15$ : esci; inserisci $a[3] \leftarrow 15$    |
| 4   | 3   | 3     | 5      | 10     | 15     | 3      | $key \leftarrow a[4] = 3; j \leftarrow 3$                     |
|     | 2   | 3     | 5      | 10     | 15     | 15     | $a[3] = 15 > 3$ : sposta $a[4] \leftarrow 15; j \leftarrow 2$ |
|     | 1   | 3     | 5      | 10     | 10     | 15     | $a[2] = 10 > 3$ : sposta $a[3] \leftarrow 10; j \leftarrow 1$ |
|     | 0   | 3     | 5      | 5      | 10     | 15     | $a[1] = 5 > 3$ : sposta $a[2] \leftarrow 5; j \leftarrow 0$   |
|     | —   | 3     | 3      | 5      | 10     | 15     | $j = 0$ : esci; inserisci $a[1] \leftarrow 3$                 |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $key$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                 |
|-----|-------|--------|--------|--------|--------|------------------------|
| —   | —     | 10     | 5      | 15     | 3      | stato iniziale         |
| 2   | 5     | 5      | 10     | 15     | 3      | inserisci 5 all'inizio |
| 3   | 15    | 5      | 10     | 15     | 3      | 15 già al suo posto    |
| 4   | 3     | 3      | 5      | 10     | 15     | inserisci 3 all'inizio |

L'algoritmo esegue 5 confronti tra chiavi e 4 spostamenti, e l'array ordinato è

$[3, 5, 10, 15]$ .

### Esercizio 3.5 — Algoritmi di ordinamento (E5)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [7, 2, 9, 4]$
2.  $a = [15, 10, 5]$
3.  $a = [3, 1, 4, 1, 5, 9, 2, 6]$
4.  $a = [20, 18, 16, 14]$
5.  $a = [11, 11, 11]$

#### Domande

1. Applicare l'algoritmo **InsertionSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni iterazione del ciclo esterno, indicare la chiave da inserire, ogni spostamento eseguito dal ciclo interno e lo stato dell'array dopo l'inserimento.
2. Per ciascun array, contare i confronti tra chiavi  $a[j] > key$  e gli spostamenti eseguiti dall'algoritmo.

## Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [7, 2, 9, 4]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                      |
|-----|-----|-----|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | —   | 7      | 2      | 9      | 4      | stato iniziale                                              |
| 2   | 1   | 2   | 7      | 2      | 9      | 4      | $key \leftarrow a[2] = 2; j \leftarrow 1$                   |
|     | 0   | 2   | 7      | 7      | 9      | 4      | $a[1] = 7 > 2$ : sposta $a[2] \leftarrow 7; j \leftarrow 0$ |
|     | —   | 2   | 2      | 7      | 9      | 4      | $j = 0$ : esci; inserisci $a[1] \leftarrow 2$               |
| 3   | 2   | 9   | 2      | 7      | 9      | 4      | $key \leftarrow a[3] = 9; j \leftarrow 2$                   |
|     | —   | 9   | 2      | 7      | 9      | 4      | $a[2] = 7 \leq 9$ : esci; inserisci $a[3] \leftarrow 9$     |
| 4   | 3   | 4   | 2      | 7      | 9      | 4      | $key \leftarrow a[4] = 4; j \leftarrow 3$                   |
|     | 2   | 4   | 2      | 7      | 9      | 9      | $a[3] = 9 > 4$ : sposta $a[4] \leftarrow 9; j \leftarrow 2$ |
|     | 1   | 4   | 2      | 7      | 7      | 9      | $a[2] = 7 > 4$ : sposta $a[3] \leftarrow 7; j \leftarrow 1$ |
|     | —   | 4   | 2      | 4      | 7      | 9      | $a[1] = 2 \leq 4$ : esci; inserisci $a[2] \leftarrow 4$     |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                 |
|-----|-----|--------|--------|--------|--------|------------------------|
| —   | —   | 7      | 2      | 9      | 4      | stato iniziale         |
| 2   | 2   | 2      | 7      | 9      | 4      | inserisci 2 all'inizio |
| 3   | 9   | 2      | 7      | 9      | 4      | 9 già al suo posto     |
| 4   | 4   | 2      | 4      | 7      | 9      | inserisci 4 tra 2 e 7  |

L'algoritmo esegue 5 confronti tra chiavi e 3 spostamenti, e l'array ordinato è

$[2, 4, 7, 9]$ .

2. Array  $a = [15, 10, 5]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                         |
|-----|-----|-----|--------|--------|--------|----------------------------------------------------------------|
| —   | —   | —   | 15     | 10     | 5      | stato iniziale                                                 |
| 2   | 1   | 10  | 15     | 10     | 5      | $key \leftarrow a[2] = 10; j \leftarrow 1$                     |
|     | 0   | 10  | 15     | 15     | 5      | $a[1] = 15 > 10$ : sposta $a[2] \leftarrow 15; j \leftarrow 0$ |
|     | —   | 10  | 10     | 15     | 5      | $j = 0$ : esci; inserisci $a[1] \leftarrow 10$                 |
| 3   | 2   | 5   | 10     | 15     | 5      | $key \leftarrow a[3] = 5; j \leftarrow 2$                      |
|     | 1   | 5   | 10     | 15     | 15     | $a[2] = 15 > 5$ : sposta $a[3] \leftarrow 15; j \leftarrow 1$  |
|     | 0   | 5   | 10     | 10     | 15     | $a[1] = 10 > 5$ : sposta $a[2] \leftarrow 10; j \leftarrow 0$  |
|     | —   | 5   | 5      | 10     | 15     | $j = 0$ : esci; inserisci $a[1] \leftarrow 5$                  |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | Azione                  |
|-----|-----|--------|--------|--------|-------------------------|
| —   | —   | 15     | 10     | 5      | stato iniziale          |
| 2   | 10  | 10     | 15     | 5      | inserisci 10 all'inizio |
| 3   | 5   | 5      | 10     | 15     | inserisci 5 all'inizio  |

L'algoritmo esegue 3 confronti tra chiavi e 3 spostamenti, e l'array ordinato è  
[5, 10, 15].

3. Array  $a = [3, 1, 4, 1, 5, 9, 2, 6]$  ( $n = 8$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | $a[7]$ | $a[8]$ | Azione                                                      |
|-----|-----|-----|--------|--------|--------|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | —   | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | stato iniziale                                              |
| 2   | 1   | 1   | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | $key \leftarrow a[2] = 1; j \leftarrow 1$                   |
|     | 0   | 1   | 3      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[1] = 3 > 1$ : sposta $a[2] \leftarrow 3; j \leftarrow 0$ |
|     | —   | 1   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $j = 0$ : esci; inserisci $a[1] \leftarrow 1$               |
| 3   | 2   | 4   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $key \leftarrow a[3] = 4; j \leftarrow 2$                   |
|     | —   | 4   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[2] = 3 \leq 4$ : esci; inserisci $a[3] \leftarrow 4$     |
| 4   | 3   | 1   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $key \leftarrow a[4] = 1; j \leftarrow 3$                   |
|     | 2   | 1   | 1      | 3      | 4      | 4      | 5      | 9      | 2      | 6      | $a[3] = 4 > 1$ : sposta $a[4] \leftarrow 4; j \leftarrow 2$ |
|     | 1   | 1   | 1      | 3      | 3      | 4      | 5      | 9      | 2      | 6      | $a[2] = 3 > 1$ : sposta $a[3] \leftarrow 3; j \leftarrow 1$ |
|     | —   | 1   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | $a[1] = 1 \leq 1$ : esci; inserisci $a[2] \leftarrow 1$     |
| 5   | 4   | 5   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | $key \leftarrow a[5] = 5; j \leftarrow 4$                   |
|     | —   | 5   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | $a[4] = 4 \leq 5$ : esci; inserisci $a[5] \leftarrow 5$     |
| 6   | 5   | 9   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | $key \leftarrow a[6] = 9; j \leftarrow 5$                   |
|     | —   | 9   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | $a[5] = 5 \leq 9$ : esci; inserisci $a[6] \leftarrow 9$     |
| 7   | 6   | 2   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | $key \leftarrow a[7] = 2; j \leftarrow 6$                   |
|     | 5   | 2   | 1      | 1      | 3      | 4      | 5      | 9      | 9      | 6      | $a[6] = 9 > 2$ : sposta $a[7] \leftarrow 9; j \leftarrow 5$ |
|     | 4   | 2   | 1      | 1      | 3      | 4      | 5      | 5      | 9      | 6      | $a[5] = 5 > 2$ : sposta $a[6] \leftarrow 5; j \leftarrow 4$ |
|     | 3   | 2   | 1      | 1      | 3      | 4      | 4      | 5      | 9      | 6      | $a[4] = 4 > 2$ : sposta $a[5] \leftarrow 4; j \leftarrow 3$ |
|     | 2   | 2   | 1      | 1      | 3      | 3      | 4      | 5      | 9      | 6      | $a[3] = 3 > 2$ : sposta $a[4] \leftarrow 3; j \leftarrow 2$ |
|     | —   | 2   | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 6      | $a[2] = 1 \leq 2$ : esci; inserisci $a[3] \leftarrow 2$     |
| 8   | 7   | 6   | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 6      | $key \leftarrow a[8] = 6; j \leftarrow 7$                   |
|     | 6   | 6   | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 9      | $a[7] = 9 > 6$ : sposta $a[8] \leftarrow 9; j \leftarrow 6$ |
|     | —   | 6   | 1      | 1      | 2      | 3      | 4      | 5      | 6      | 9      | $a[6] = 5 \leq 6$ : esci; inserisci $a[7] \leftarrow 6$     |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | $a[7]$ | $a[8]$ | Azione                 |
|-----|-----|--------|--------|--------|--------|--------|--------|--------|--------|------------------------|
| —   | —   | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | stato iniziale         |
| 2   | 1   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | inserisci 1 all'inizio |
| 3   | 4   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | 4 già al suo posto     |
| 4   | 1   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | inserisci 1 tra 1 e 3  |
| 5   | 5   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | 5 già al suo posto     |
| 6   | 9   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | 9 già al suo posto     |
| 7   | 2   | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 6      | inserisci 2 tra 1 e 3  |
| 8   | 6   | 1      | 1      | 2      | 3      | 4      | 5      | 6      | 9      | inserisci 6 tra 5 e 9  |

L'algoritmo esegue 14 confronti tra chiavi e 8 spostamenti, e l'array ordinato è

$[1, 1, 2, 3, 4, 5, 6, 9]$ .

4. Array  $a = [20, 18, 16, 14]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                         |
|-----|-----|-----|--------|--------|--------|--------|----------------------------------------------------------------|
| —   | —   | —   | 20     | 18     | 16     | 14     | stato iniziale                                                 |
| 2   | 1   | 18  | 20     | 18     | 16     | 14     | $key \leftarrow a[2] = 18; j \leftarrow 1$                     |
|     | 0   | 18  | 20     | 20     | 16     | 14     | $a[1] = 20 > 18$ : sposta $a[2] \leftarrow 20; j \leftarrow 0$ |
|     | —   | 18  | 18     | 20     | 16     | 14     | $j = 0$ : esci; inserisci $a[1] \leftarrow 18$                 |
| 3   | 2   | 16  | 18     | 20     | 16     | 14     | $key \leftarrow a[3] = 16; j \leftarrow 2$                     |
|     | 1   | 16  | 18     | 20     | 20     | 14     | $a[2] = 20 > 16$ : sposta $a[3] \leftarrow 20; j \leftarrow 1$ |
|     | 0   | 16  | 18     | 18     | 20     | 14     | $a[1] = 18 > 16$ : sposta $a[2] \leftarrow 18; j \leftarrow 0$ |
|     | —   | 16  | 16     | 18     | 20     | 14     | $j = 0$ : esci; inserisci $a[1] \leftarrow 16$                 |
| 4   | 3   | 14  | 16     | 18     | 20     | 14     | $key \leftarrow a[4] = 14; j \leftarrow 3$                     |
|     | 2   | 14  | 16     | 18     | 20     | 20     | $a[3] = 20 > 14$ : sposta $a[4] \leftarrow 20; j \leftarrow 2$ |
|     | 1   | 14  | 16     | 18     | 18     | 20     | $a[2] = 18 > 14$ : sposta $a[3] \leftarrow 18; j \leftarrow 1$ |
|     | 0   | 14  | 16     | 16     | 18     | 20     | $a[1] = 16 > 14$ : sposta $a[2] \leftarrow 16; j \leftarrow 0$ |
|     | —   | 14  | 14     | 16     | 18     | 20     | $j = 0$ : esci; inserisci $a[1] \leftarrow 14$                 |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                  |
|-----|-----|--------|--------|--------|--------|-------------------------|
| —   | —   | 20     | 18     | 16     | 14     | stato iniziale          |
| 2   | 18  | 18     | 20     | 16     | 14     | inserisci 18 all'inizio |
| 3   | 16  | 16     | 18     | 20     | 14     | inserisci 16 all'inizio |
| 4   | 14  | 14     | 16     | 18     | 20     | inserisci 14 all'inizio |

L'algoritmo esegue 6 confronti tra chiavi e 6 spostamenti, e l'array ordinato è

$[14, 16, 18, 20]$ .

5. Array  $a = [11, 11, 11]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                     |
|-----|-----|-----|--------|--------|--------|------------------------------------------------------------|
| —   | —   | —   | 11     | 11     | 11     | stato iniziale                                             |
| 2   | 1   | 11  | 11     | 11     | 11     | $key \leftarrow a[2] = 11; j \leftarrow 1$                 |
|     | —   | 11  | 11     | 11     | 11     | $a[1] = 11 \leq 11$ : esci; inserisci $a[2] \leftarrow 11$ |
| 3   | 2   | 11  | 11     | 11     | 11     | $key \leftarrow a[3] = 11; j \leftarrow 2$                 |
|     | —   | 11  | 11     | 11     | 11     | $a[2] = 11 \leq 11$ : esci; inserisci $a[3] \leftarrow 11$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | Azione              |
|-----|-----|--------|--------|--------|---------------------|
| –   | –   | 11     | 11     | 11     | stato iniziale      |
| 2   | 11  | 11     | 11     | 11     | 11 già al suo posto |
| 3   | 11  | 11     | 11     | 11     | 11 già al suo posto |

L'algoritmo esegue 2 confronti tra chiavi e 0 spostamenti, e l'array ordinato è

[11, 11, 11].

### Esercizio 3.6 — Algoritmi di ordinamento (E6)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [9, 3, 7, 2]$
2.  $a = [12, 4, 18, 6, 10]$
3.  $a = [5, 4, 3, 2, 1]$
4.  $a = [100, 50, 25]$
5.  $a = [8]$

#### Domande

1. Applicare l'algoritmo **InsertionSort** a ciascun array, per ordinarlo in ordine non crescente. Mostrare l'esecuzione passo per passo: per ogni iterazione del ciclo esterno, indicare la chiave da inserire, ogni spostamento eseguito dal ciclo interno e lo stato dell'array dopo l'inserimento.
2. Per ciascun array, contare i confronti tra chiavi  $a[j] > key$  e gli spostamenti eseguiti dall'algoritmo.

#### Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [9, 3, 7, 2]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                      |
|-----|-----|-----|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | —   | 9      | 3      | 7      | 2      | stato iniziale                                              |
| 2   | 1   | 3   | 9      | 3      | 7      | 2      | $key \leftarrow a[2] = 3; j \leftarrow 1$                   |
|     | 0   | 3   | 9      | 9      | 7      | 2      | $a[1] = 9 > 3$ : sposta $a[2] \leftarrow 9; j \leftarrow 0$ |
|     | —   | 3   | 3      | 9      | 7      | 2      | $j = 0$ : esci; inserisci $a[1] \leftarrow 3$               |
| 3   | 2   | 7   | 3      | 9      | 7      | 2      | $key \leftarrow a[3] = 7; j \leftarrow 2$                   |
|     | 1   | 7   | 3      | 9      | 9      | 2      | $a[2] = 9 > 7$ : sposta $a[3] \leftarrow 9; j \leftarrow 1$ |
|     | —   | 7   | 3      | 7      | 9      | 2      | $a[1] = 3 \leq 7$ : esci; inserisci $a[2] \leftarrow 7$     |
| 4   | 3   | 2   | 3      | 7      | 9      | 2      | $key \leftarrow a[4] = 2; j \leftarrow 3$                   |
|     | 2   | 2   | 3      | 7      | 9      | 9      | $a[3] = 9 > 2$ : sposta $a[4] \leftarrow 9; j \leftarrow 2$ |
|     | 1   | 2   | 3      | 7      | 7      | 9      | $a[2] = 7 > 2$ : sposta $a[3] \leftarrow 7; j \leftarrow 1$ |
|     | 0   | 2   | 3      | 3      | 7      | 9      | $a[1] = 3 > 2$ : sposta $a[2] \leftarrow 3; j \leftarrow 0$ |
|     | —   | 2   | 2      | 3      | 7      | 9      | $j = 0$ : esci; inserisci $a[1] \leftarrow 2$               |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                 |
|-----|-----|--------|--------|--------|--------|------------------------|
| —   | —   | 9      | 3      | 7      | 2      | stato iniziale         |
| 2   | 3   | 3      | 9      | 7      | 2      | inserisci 3 all'inizio |
| 3   | 7   | 3      | 7      | 9      | 2      | inserisci 7 tra 3 e 9  |
| 4   | 2   | 2      | 3      | 7      | 9      | inserisci 2 all'inizio |

L'algoritmo esegue 6 confronti tra chiavi e 5 spostamenti, e l'array ordinato è

$[2, 3, 7, 9]$ .

2. Array  $a = [12, 4, 18, 6, 10]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                         |
|-----|-----|-----|--------|--------|--------|--------|--------|----------------------------------------------------------------|
| —   | —   | —   | 12     | 4      | 18     | 6      | 10     | stato iniziale                                                 |
| 2   | 1   | 4   | 12     | 4      | 18     | 6      | 10     | $key \leftarrow a[2] = 4; j \leftarrow 1$                      |
|     | 0   | 4   | 12     | 12     | 18     | 6      | 10     | $a[1] = 12 > 4$ : sposta $a[2] \leftarrow 12; j \leftarrow 0$  |
|     | —   | 4   | 4      | 12     | 18     | 6      | 10     | $j = 0$ : esci; inserisci $a[1] \leftarrow 4$                  |
| 3   | 2   | 18  | 4      | 12     | 18     | 6      | 10     | $key \leftarrow a[3] = 18; j \leftarrow 2$                     |
|     | —   | 18  | 4      | 12     | 18     | 6      | 10     | $a[2] = 12 \leq 18$ : esci; inserisci $a[3] \leftarrow 18$     |
| 4   | 3   | 6   | 4      | 12     | 18     | 6      | 10     | $key \leftarrow a[4] = 6; j \leftarrow 3$                      |
|     | 2   | 6   | 4      | 12     | 18     | 18     | 10     | $a[3] = 18 > 6$ : sposta $a[4] \leftarrow 18; j \leftarrow 2$  |
|     | 1   | 6   | 4      | 12     | 12     | 18     | 10     | $a[2] = 12 > 6$ : sposta $a[3] \leftarrow 12; j \leftarrow 1$  |
|     | —   | 6   | 4      | 6      | 12     | 18     | 10     | $a[1] = 4 \leq 6$ : esci; inserisci $a[2] \leftarrow 6$        |
| 5   | 4   | 10  | 4      | 6      | 12     | 18     | 10     | $key \leftarrow a[5] = 10; j \leftarrow 4$                     |
|     | 3   | 10  | 4      | 6      | 12     | 18     | 18     | $a[4] = 18 > 10$ : sposta $a[5] \leftarrow 18; j \leftarrow 3$ |
|     | 2   | 10  | 4      | 6      | 12     | 12     | 18     | $a[3] = 12 > 10$ : sposta $a[4] \leftarrow 12; j \leftarrow 2$ |
|     | —   | 10  | 4      | 6      | 10     | 12     | 18     | $a[2] = 6 \leq 10$ : esci; inserisci $a[3] \leftarrow 10$      |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                  |
|-----|-----|--------|--------|--------|--------|--------|-------------------------|
| —   | —   | 12     | 4      | 18     | 6      | 10     | stato iniziale          |
| 2   | 4   | 4      | 12     | 18     | 6      | 10     | inserisci 4 all'inizio  |
| 3   | 18  | 4      | 12     | 18     | 6      | 10     | 18 già al suo posto     |
| 4   | 6   | 4      | 6      | 12     | 18     | 10     | inserisci 6 tra 4 e 12  |
| 5   | 10  | 4      | 6      | 10     | 12     | 18     | inserisci 10 tra 6 e 12 |

L'algoritmo esegue 8 confronti tra chiavi e 5 spostamenti, e l'array ordinato è

[4, 6, 10, 12, 18].

3. Array  $a = [5, 4, 3, 2, 1]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                      |
|-----|-----|-----|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | —   | 5      | 4      | 3      | 2      | 1      | stato iniziale                                              |
| 2   | 1   | 4   | 5      | 4      | 3      | 2      | 1      | $key \leftarrow a[2] = 4; j \leftarrow 1$                   |
|     | 0   | 4   | 5      | 5      | 3      | 2      | 1      | $a[1] = 5 > 4$ : sposta $a[2] \leftarrow 5; j \leftarrow 0$ |
|     | —   | 4   | 4      | 5      | 3      | 2      | 1      | $j = 0$ : esci; inserisci $a[1] \leftarrow 4$               |
| 3   | 2   | 3   | 4      | 5      | 3      | 2      | 1      | $key \leftarrow a[3] = 3; j \leftarrow 2$                   |
|     | 1   | 3   | 4      | 5      | 5      | 2      | 1      | $a[2] = 5 > 3$ : sposta $a[3] \leftarrow 5; j \leftarrow 1$ |
|     | 0   | 3   | 4      | 4      | 5      | 2      | 1      | $a[1] = 4 > 3$ : sposta $a[2] \leftarrow 4; j \leftarrow 0$ |
|     | —   | 3   | 3      | 4      | 5      | 2      | 1      | $j = 0$ : esci; inserisci $a[1] \leftarrow 3$               |
| 4   | 3   | 2   | 3      | 4      | 5      | 2      | 1      | $key \leftarrow a[4] = 2; j \leftarrow 3$                   |
|     | 2   | 2   | 3      | 4      | 5      | 5      | 1      | $a[3] = 5 > 2$ : sposta $a[4] \leftarrow 5; j \leftarrow 2$ |
|     | 1   | 2   | 3      | 4      | 4      | 5      | 1      | $a[2] = 4 > 2$ : sposta $a[3] \leftarrow 4; j \leftarrow 1$ |
|     | 0   | 2   | 3      | 3      | 4      | 5      | 1      | $a[1] = 3 > 2$ : sposta $a[2] \leftarrow 3; j \leftarrow 0$ |
|     | —   | 2   | 2      | 3      | 4      | 5      | 1      | $j = 0$ : esci; inserisci $a[1] \leftarrow 2$               |
| 5   | 4   | 1   | 2      | 3      | 4      | 5      | 1      | $key \leftarrow a[5] = 1; j \leftarrow 4$                   |
|     | 3   | 1   | 2      | 3      | 4      | 5      | 5      | $a[4] = 5 > 1$ : sposta $a[5] \leftarrow 5; j \leftarrow 3$ |
|     | 2   | 1   | 2      | 3      | 4      | 4      | 5      | $a[3] = 4 > 1$ : sposta $a[4] \leftarrow 4; j \leftarrow 2$ |
|     | 1   | 1   | 2      | 3      | 3      | 4      | 5      | $a[2] = 3 > 1$ : sposta $a[3] \leftarrow 3; j \leftarrow 1$ |
|     | 0   | 1   | 2      | 2      | 3      | 4      | 5      | $a[1] = 2 > 1$ : sposta $a[2] \leftarrow 2; j \leftarrow 0$ |
|     | —   | 1   | 1      | 2      | 3      | 4      | 5      | $j = 0$ : esci; inserisci $a[1] \leftarrow 1$               |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                 |
|-----|-----|--------|--------|--------|--------|--------|------------------------|
| —   | —   | 5      | 4      | 3      | 2      | 1      | stato iniziale         |
| 2   | 4   | 4      | 5      | 3      | 2      | 1      | inserisci 4 all'inizio |
| 3   | 3   | 3      | 4      | 5      | 2      | 1      | inserisci 3 all'inizio |
| 4   | 2   | 2      | 3      | 4      | 5      | 1      | inserisci 2 all'inizio |
| 5   | 1   | 1      | 2      | 3      | 4      | 5      | inserisci 1 all'inizio |

L'algoritmo esegue 10 confronti tra chiavi e 10 spostamenti, e l'array ordinato è

$[1, 2, 3, 4, 5]$ .

4. Array  $a = [100, 50, 25]$  ( $n = 3$ ).

*Traccia dettagliata:*

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                           |
|-----|-----|-----|--------|--------|--------|------------------------------------------------------------------|
| —   | —   | —   | 100    | 50     | 25     | stato iniziale                                                   |
| 2   | 1   | 50  | 100    | 50     | 25     | $key \leftarrow a[2] = 50; j \leftarrow 1$                       |
|     | 0   | 50  | 100    | 100    | 25     | $a[1] = 100 > 50$ : sposta $a[2] \leftarrow 100; j \leftarrow 0$ |
|     | —   | 50  | 50     | 100    | 25     | $j = 0$ : esci; inserisci $a[1] \leftarrow 50$                   |
| 3   | 2   | 25  | 50     | 100    | 25     | $key \leftarrow a[3] = 25; j \leftarrow 2$                       |
|     | 1   | 25  | 50     | 100    | 100    | $a[2] = 100 > 25$ : sposta $a[3] \leftarrow 100; j \leftarrow 1$ |
|     | 0   | 25  | 50     | 50     | 100    | $a[1] = 50 > 25$ : sposta $a[2] \leftarrow 50; j \leftarrow 0$   |
|     | —   | 25  | 25     | 50     | 100    | $j = 0$ : esci; inserisci $a[1] \leftarrow 25$                   |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | Azione                  |
|-----|-----|--------|--------|--------|-------------------------|
| —   | —   | 100    | 50     | 25     | stato iniziale          |
| 2   | 50  | 50     | 100    | 25     | inserisci 50 all'inizio |
| 3   | 25  | 25     | 50     | 100    | inserisci 25 all'inizio |

L'algoritmo esegue 3 confronti tra chiavi e 3 spostamenti, e l'array ordinato è

$[25, 50, 100]$ .

5. Array  $a = [8]$  ( $n = 1$ ).

L'array ha un solo elemento: il ciclo esterno non viene eseguito e l'array è già ordinato.

| $a[1]$ | Azione                                                        |
|--------|---------------------------------------------------------------|
| 8      | stato iniziale: $n = 1$ , il ciclo esterno non viene eseguito |

L'array ordinato è  $[8]$ , ottenuto senza alcun confronto tra chiavi e senza alcuno spostamento.

### Esercizio 3.7 — Algoritmi di ordinamento (E7)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [6, 1, 8, 2, 5]$

2.  $a = [13, 11, 9, 7]$

3.  $a = [2, 8, 4, 6]$

4.  $a = [30, 20, 10]$

5.  $a = [5, 10, 15, 20]$

**Domande**

1. Applicare l'algoritmo **InsertionSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni iterazione del ciclo esterno, indicare la chiave da inserire, ogni spostamento eseguito dal ciclo interno e lo stato dell'array dopo l'inserimento.
2. Per ciascun array, contare i confronti tra chiavi  $a[j] > key$  e gli spostamenti eseguiti dall'algoritmo.

**Soluzione**

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [6, 1, 8, 2, 5]$  ( $n = 5$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $key$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                      |
|-----|-----|-------|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | —     | 6      | 1      | 8      | 2      | 5      | stato iniziale                                              |
| 2   | 1   | 1     | 6      | 1      | 8      | 2      | 5      | $key \leftarrow a[2] = 1; j \leftarrow 1$                   |
|     | 0   | 1     | 6      | 6      | 8      | 2      | 5      | $a[1] = 6 > 1$ : sposta $a[2] \leftarrow 6; j \leftarrow 0$ |
|     | —   | 1     | 1      | 6      | 8      | 2      | 5      | $j = 0$ : esci; inserisci $a[1] \leftarrow 1$               |
| 3   | 2   | 8     | 1      | 6      | 8      | 2      | 5      | $key \leftarrow a[3] = 8; j \leftarrow 2$                   |
|     | —   | 8     | 1      | 6      | 8      | 2      | 5      | $a[2] = 6 \leq 8$ : esci; inserisci $a[3] \leftarrow 8$     |
| 4   | 3   | 2     | 1      | 6      | 8      | 2      | 5      | $key \leftarrow a[4] = 2; j \leftarrow 3$                   |
|     | 2   | 2     | 1      | 6      | 8      | 8      | 5      | $a[3] = 8 > 2$ : sposta $a[4] \leftarrow 8; j \leftarrow 2$ |
|     | 1   | 2     | 1      | 6      | 6      | 8      | 5      | $a[2] = 6 > 2$ : sposta $a[3] \leftarrow 6; j \leftarrow 1$ |
|     | —   | 2     | 1      | 2      | 6      | 8      | 5      | $a[1] = 1 \leq 2$ : esci; inserisci $a[2] \leftarrow 2$     |
| 5   | 4   | 5     | 1      | 2      | 6      | 8      | 5      | $key \leftarrow a[5] = 5; j \leftarrow 4$                   |
|     | 3   | 5     | 1      | 2      | 6      | 8      | 8      | $a[4] = 8 > 5$ : sposta $a[5] \leftarrow 8; j \leftarrow 3$ |
|     | 2   | 5     | 1      | 2      | 6      | 6      | 8      | $a[3] = 6 > 5$ : sposta $a[4] \leftarrow 6; j \leftarrow 2$ |
|     | —   | 5     | 1      | 2      | 5      | 6      | 8      | $a[2] = 2 \leq 5$ : esci; inserisci $a[3] \leftarrow 5$     |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                 |
|-----|-----|--------|--------|--------|--------|--------|------------------------|
| —   | —   | 6      | 1      | 8      | 2      | 5      | stato iniziale         |
| 2   | 1   | 1      | 6      | 8      | 2      | 5      | inserisci 1 all'inizio |
| 3   | 8   | 1      | 6      | 8      | 2      | 5      | 8 già al suo posto     |
| 4   | 2   | 1      | 2      | 6      | 8      | 5      | inserisci 2 tra 1 e 6  |
| 5   | 5   | 1      | 2      | 5      | 6      | 8      | inserisci 5 tra 2 e 6  |

L'algoritmo esegue 8 confronti tra chiavi e 5 spostamenti, e l'array ordinato è

$$[1, 2, 5, 6, 8].$$

2. Array  $a = [13, 11, 9, 7]$  ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                         |
|-----|-----|-----|--------|--------|--------|--------|----------------------------------------------------------------|
| —   | —   | —   | 13     | 11     | 9      | 7      | stato iniziale                                                 |
| 2   | 1   | 11  | 13     | 11     | 9      | 7      | $key \leftarrow a[2] = 11; j \leftarrow 1$                     |
|     | 0   | 11  | 13     | 13     | 9      | 7      | $a[1] = 13 > 11$ : sposta $a[2] \leftarrow 13; j \leftarrow 0$ |
|     | —   | 11  | 11     | 13     | 9      | 7      | $j = 0$ : esci; inserisci $a[1] \leftarrow 11$                 |
| 3   | 2   | 9   | 11     | 13     | 9      | 7      | $key \leftarrow a[3] = 9; j \leftarrow 2$                      |
|     | 1   | 9   | 11     | 13     | 13     | 7      | $a[2] = 13 > 9$ : sposta $a[3] \leftarrow 13; j \leftarrow 1$  |
|     | 0   | 9   | 11     | 11     | 13     | 7      | $a[1] = 11 > 9$ : sposta $a[2] \leftarrow 11; j \leftarrow 0$  |
|     | —   | 9   | 9      | 11     | 13     | 7      | $j = 0$ : esci; inserisci $a[1] \leftarrow 9$                  |
| 4   | 3   | 7   | 9      | 11     | 13     | 7      | $key \leftarrow a[4] = 7; j \leftarrow 3$                      |
|     | 2   | 7   | 9      | 11     | 13     | 13     | $a[3] = 13 > 7$ : sposta $a[4] \leftarrow 13; j \leftarrow 2$  |
|     | 1   | 7   | 9      | 11     | 11     | 13     | $a[2] = 11 > 7$ : sposta $a[3] \leftarrow 11; j \leftarrow 1$  |
|     | 0   | 7   | 9      | 9      | 11     | 13     | $a[1] = 9 > 7$ : sposta $a[2] \leftarrow 9; j \leftarrow 0$    |
|     | —   | 7   | 7      | 9      | 11     | 13     | $j = 0$ : esci; inserisci $a[1] \leftarrow 7$                  |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                  |
|-----|-----|--------|--------|--------|--------|-------------------------|
| —   | —   | 13     | 11     | 9      | 7      | stato iniziale          |
| 2   | 11  | 11     | 13     | 9      | 7      | inserisci 11 all'inizio |
| 3   | 9   | 9      | 11     | 13     | 7      | inserisci 9 all'inizio  |
| 4   | 7   | 7      | 9      | 11     | 13     | inserisci 7 all'inizio  |

L'algoritmo esegue 6 confronti tra chiavi e 6 spostamenti, e l'array ordinato è

$$[7, 9, 11, 13].$$

3. Array  $a = [2, 8, 4, 6]$  ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                      |
|-----|-----|-----|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | —   | 2      | 8      | 4      | 6      | stato iniziale                                              |
| 2   | 1   | 8   | 2      | 8      | 4      | 6      | $key \leftarrow a[2] = 8; j \leftarrow 1$                   |
|     | —   | 8   | 2      | 8      | 4      | 6      | $a[1] = 2 \leq 8$ : esci; inserisci $a[2] \leftarrow 8$     |
| 3   | 2   | 4   | 2      | 8      | 4      | 6      | $key \leftarrow a[3] = 4; j \leftarrow 2$                   |
|     | 1   | 4   | 2      | 8      | 8      | 6      | $a[2] = 8 > 4$ : sposta $a[3] \leftarrow 8; j \leftarrow 1$ |
|     | —   | 4   | 2      | 4      | 8      | 6      | $a[1] = 2 \leq 4$ : esci; inserisci $a[2] \leftarrow 4$     |
| 4   | 3   | 6   | 2      | 4      | 8      | 6      | $key \leftarrow a[4] = 6; j \leftarrow 3$                   |
|     | 2   | 6   | 2      | 4      | 8      | 8      | $a[3] = 8 > 6$ : sposta $a[4] \leftarrow 8; j \leftarrow 2$ |
|     | —   | 6   | 2      | 4      | 6      | 8      | $a[2] = 4 \leq 6$ : esci; inserisci $a[3] \leftarrow 6$     |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                |
|-----|-----|--------|--------|--------|--------|-----------------------|
| —   | —   | 2      | 8      | 4      | 6      | stato iniziale        |
| 2   | 8   | 2      | 8      | 4      | 6      | 8 già al suo posto    |
| 3   | 4   | 2      | 4      | 8      | 6      | inserisci 4 tra 2 e 8 |
| 4   | 6   | 2      | 4      | 6      | 8      | inserisci 6 tra 4 e 8 |

L'algoritmo esegue 5 confronti tra chiavi e 2 spostamenti, e l'array ordinato è

$[2, 4, 6, 8]$ .

4. Array  $a = [30, 20, 10]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                         |
|-----|-----|-----|--------|--------|--------|----------------------------------------------------------------|
| —   | —   | —   | 30     | 20     | 10     | stato iniziale                                                 |
| 2   | 1   | 20  | 30     | 20     | 10     | $key \leftarrow a[2] = 20; j \leftarrow 1$                     |
|     | 0   | 20  | 30     | 30     | 10     | $a[1] = 30 > 20$ : sposta $a[2] \leftarrow 30; j \leftarrow 0$ |
|     | —   | 20  | 20     | 30     | 10     | $j = 0$ : esci; inserisci $a[1] \leftarrow 20$                 |
| 3   | 2   | 10  | 20     | 30     | 10     | $key \leftarrow a[3] = 10; j \leftarrow 2$                     |
|     | 1   | 10  | 20     | 30     | 30     | $a[2] = 30 > 10$ : sposta $a[3] \leftarrow 30; j \leftarrow 1$ |
|     | 0   | 10  | 20     | 20     | 30     | $a[1] = 20 > 10$ : sposta $a[2] \leftarrow 20; j \leftarrow 0$ |
|     | —   | 10  | 10     | 20     | 30     | $j = 0$ : esci; inserisci $a[1] \leftarrow 10$                 |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | Azione                  |
|-----|-----|--------|--------|--------|-------------------------|
| —   | —   | 30     | 20     | 10     | stato iniziale          |
| 2   | 20  | 20     | 30     | 10     | inserisci 20 all'inizio |
| 3   | 10  | 10     | 20     | 30     | inserisci 10 all'inizio |

L'algoritmo esegue 3 confronti tra chiavi e 3 spostamenti, e l'array ordinato è  
 $[10, 20, 30]$ .

5. Array  $a = [5, 10, 15, 20]$  ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                     |
|-----|-----|-----|--------|--------|--------|--------|------------------------------------------------------------|
| —   | —   | —   | 5      | 10     | 15     | 20     | stato iniziale                                             |
| 2   | 1   | 10  | 5      | 10     | 15     | 20     | $key \leftarrow a[2] = 10; j \leftarrow 1$                 |
| —   | —   | 10  | 5      | 10     | 15     | 20     | $a[1] = 5 \leq 10$ : esci; inserisci $a[2] \leftarrow 10$  |
| 3   | 2   | 15  | 5      | 10     | 15     | 20     | $key \leftarrow a[3] = 15; j \leftarrow 2$                 |
| —   | —   | 15  | 5      | 10     | 15     | 20     | $a[2] = 10 \leq 15$ : esci; inserisci $a[3] \leftarrow 15$ |
| 4   | 3   | 20  | 5      | 10     | 15     | 20     | $key \leftarrow a[4] = 20; j \leftarrow 3$                 |
| —   | —   | 20  | 5      | 10     | 15     | 20     | $a[3] = 15 \leq 20$ : esci; inserisci $a[4] \leftarrow 20$ |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione              |
|-----|-----|--------|--------|--------|--------|---------------------|
| —   | —   | 5      | 10     | 15     | 20     | stato iniziale      |
| 2   | 10  | 5      | 10     | 15     | 20     | 10 già al suo posto |
| 3   | 15  | 5      | 10     | 15     | 20     | 15 già al suo posto |
| 4   | 20  | 5      | 10     | 15     | 20     | 20 già al suo posto |

L'algoritmo esegue 3 confronti tra chiavi e 0 spostamenti, e l'array ordinato è  
 $[5, 10, 15, 20]$ .

### Esercizio 3.8 — Algoritmi di ordinamento (E8)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [14, 7, 21, 3, 18]$
2.  $a = [40, 30, 20, 10]$
3.  $a = [3, 6, 1, 9, 2]$
4.  $a = [25, 12, 50]$
5.  $a = [17, 17, 17, 17]$

#### Domande

1. Applicare l'algoritmo **InsertionSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni iterazione del ciclo esterno,

indicare la chiave da inserire, ogni spostamento eseguito dal ciclo interno e lo stato dell'array dopo l'inserimento.

2. Per ciascun array, contare i confronti tra chiavi  $a[j] > key$  e gli spostamenti eseguiti dall'algoritmo.

### Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [14, 7, 21, 3, 18]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | $key$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                         |
|-----|-----|-------|--------|--------|--------|--------|--------|----------------------------------------------------------------|
| —   | —   | —     | 14     | 7      | 21     | 3      | 18     | stato iniziale                                                 |
| 2   | 1   | 7     | 14     | 7      | 21     | 3      | 18     | $key \leftarrow a[2] = 7; j \leftarrow 1$                      |
|     | 0   | 7     | 14     | 14     | 21     | 3      | 18     | $a[1] = 14 > 7$ : sposta $a[2] \leftarrow 14; j \leftarrow 0$  |
|     | —   | 7     | 7      | 14     | 21     | 3      | 18     | $j = 0$ : esci; inserisci $a[1] \leftarrow 7$                  |
| 3   | 2   | 21    | 7      | 14     | 21     | 3      | 18     | $key \leftarrow a[3] = 21; j \leftarrow 2$                     |
|     | —   | 21    | 7      | 14     | 21     | 3      | 18     | $a[2] = 14 \leq 21$ : esci; inserisci $a[3] \leftarrow 21$     |
| 4   | 3   | 3     | 7      | 14     | 21     | 3      | 18     | $key \leftarrow a[4] = 3; j \leftarrow 3$                      |
|     | 2   | 3     | 7      | 14     | 21     | 21     | 18     | $a[3] = 21 > 3$ : sposta $a[4] \leftarrow 21; j \leftarrow 2$  |
|     | 1   | 3     | 7      | 14     | 14     | 21     | 18     | $a[2] = 14 > 3$ : sposta $a[3] \leftarrow 14; j \leftarrow 1$  |
|     | 0   | 3     | 7      | 7      | 14     | 21     | 18     | $a[1] = 7 > 3$ : sposta $a[2] \leftarrow 7; j \leftarrow 0$    |
|     | —   | 3     | 3      | 7      | 14     | 21     | 18     | $j = 0$ : esci; inserisci $a[1] \leftarrow 3$                  |
| 5   | 4   | 18    | 3      | 7      | 14     | 21     | 18     | $key \leftarrow a[5] = 18; j \leftarrow 4$                     |
|     | 3   | 18    | 3      | 7      | 14     | 21     | 21     | $a[4] = 21 > 18$ : sposta $a[5] \leftarrow 21; j \leftarrow 3$ |
|     | —   | 18    | 3      | 7      | 14     | 18     | 21     | $a[3] = 14 \leq 18$ : esci; inserisci $a[4] \leftarrow 18$     |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $key$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                   |
|-----|-------|--------|--------|--------|--------|--------|--------------------------|
| —   | —     | 14     | 7      | 21     | 3      | 18     | stato iniziale           |
| 2   | 7     | 7      | 14     | 21     | 3      | 18     | inserisci 7 all'inizio   |
| 3   | 21    | 7      | 14     | 21     | 3      | 18     | 21 già al suo posto      |
| 4   | 3     | 3      | 7      | 14     | 21     | 18     | inserisci 3 all'inizio   |
| 5   | 18    | 3      | 7      | 14     | 18     | 21     | inserisci 18 tra 14 e 21 |

L'algoritmo esegue 7 confronti tra chiavi e 5 spostamenti, e l'array ordinato è

$[3, 7, 14, 18, 21]$ .

2. Array  $a = [40, 30, 20, 10]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                         |
|-----|-----|-----|--------|--------|--------|--------|----------------------------------------------------------------|
| –   | –   | –   | 40     | 30     | 20     | 10     | stato iniziale                                                 |
| 2   | 1   | 30  | 40     | 30     | 20     | 10     | $key \leftarrow a[2] = 30; j \leftarrow 1$                     |
|     | 0   | 30  | 40     | 40     | 20     | 10     | $a[1] = 40 > 30$ : sposta $a[2] \leftarrow 40; j \leftarrow 0$ |
|     | –   | 30  | 30     | 40     | 20     | 10     | $j = 0$ : esci; inserisci $a[1] \leftarrow 30$                 |
| 3   | 2   | 20  | 30     | 40     | 20     | 10     | $key \leftarrow a[3] = 20; j \leftarrow 2$                     |
|     | 1   | 20  | 30     | 40     | 40     | 10     | $a[2] = 40 > 20$ : sposta $a[3] \leftarrow 40; j \leftarrow 1$ |
|     | 0   | 20  | 30     | 30     | 40     | 10     | $a[1] = 30 > 20$ : sposta $a[2] \leftarrow 30; j \leftarrow 0$ |
|     | –   | 20  | 20     | 30     | 40     | 10     | $j = 0$ : esci; inserisci $a[1] \leftarrow 20$                 |
| 4   | 3   | 10  | 20     | 30     | 40     | 10     | $key \leftarrow a[4] = 10; j \leftarrow 3$                     |
|     | 2   | 10  | 20     | 30     | 40     | 40     | $a[3] = 40 > 10$ : sposta $a[4] \leftarrow 40; j \leftarrow 2$ |
|     | 1   | 10  | 20     | 30     | 30     | 40     | $a[2] = 30 > 10$ : sposta $a[3] \leftarrow 30; j \leftarrow 1$ |
|     | 0   | 10  | 20     | 20     | 30     | 40     | $a[1] = 20 > 10$ : sposta $a[2] \leftarrow 20; j \leftarrow 0$ |
|     | –   | 10  | 10     | 20     | 30     | 40     | $j = 0$ : esci; inserisci $a[1] \leftarrow 10$                 |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                  |
|-----|-----|--------|--------|--------|--------|-------------------------|
| –   | –   | 40     | 30     | 20     | 10     | stato iniziale          |
| 2   | 30  | 30     | 40     | 20     | 10     | inserisci 30 all'inizio |
| 3   | 20  | 20     | 30     | 40     | 10     | inserisci 20 all'inizio |
| 4   | 10  | 10     | 20     | 30     | 40     | inserisci 10 all'inizio |

L'algoritmo esegue 6 confronti tra chiavi e 6 spostamenti, e l'array ordinato è

[10, 20, 30, 40].

### 3. Array $a = [3, 6, 1, 9, 2]$ ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                      |
|-----|-----|-----|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| –   | –   | –   | 3      | 6      | 1      | 9      | 2      | stato iniziale                                              |
| 2   | 1   | 6   | 3      | 6      | 1      | 9      | 2      | $key \leftarrow a[2] = 6; j \leftarrow 1$                   |
|     | –   | 6   | 3      | 6      | 1      | 9      | 2      | $a[1] = 3 \leq 6$ : esci; inserisci $a[2] \leftarrow 6$     |
| 3   | 2   | 1   | 3      | 6      | 1      | 9      | 2      | $key \leftarrow a[3] = 1; j \leftarrow 2$                   |
|     | 1   | 1   | 3      | 6      | 6      | 9      | 2      | $a[2] = 6 > 1$ : sposta $a[3] \leftarrow 6; j \leftarrow 1$ |
|     | 0   | 1   | 3      | 3      | 6      | 9      | 2      | $a[1] = 3 > 1$ : sposta $a[2] \leftarrow 3; j \leftarrow 0$ |
|     | –   | 1   | 1      | 3      | 6      | 9      | 2      | $j = 0$ : esci; inserisci $a[1] \leftarrow 1$               |
| 4   | 3   | 9   | 1      | 3      | 6      | 9      | 2      | $key \leftarrow a[4] = 9; j \leftarrow 3$                   |
|     | –   | 9   | 1      | 3      | 6      | 9      | 2      | $a[3] = 6 \leq 9$ : esci; inserisci $a[4] \leftarrow 9$     |
| 5   | 4   | 2   | 1      | 3      | 6      | 9      | 2      | $key \leftarrow a[5] = 2; j \leftarrow 4$                   |
|     | 3   | 2   | 1      | 3      | 6      | 9      | 9      | $a[4] = 9 > 2$ : sposta $a[5] \leftarrow 9; j \leftarrow 3$ |
|     | 2   | 2   | 1      | 3      | 6      | 6      | 9      | $a[3] = 6 > 2$ : sposta $a[4] \leftarrow 6; j \leftarrow 2$ |
|     | 1   | 2   | 1      | 3      | 3      | 6      | 9      | $a[2] = 3 > 2$ : sposta $a[3] \leftarrow 3; j \leftarrow 1$ |
|     | –   | 2   | 1      | 2      | 3      | 6      | 9      | $a[1] = 1 \leq 2$ : esci; inserisci $a[2] \leftarrow 2$     |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                 |
|-----|-----|--------|--------|--------|--------|--------|------------------------|
| —   | —   | 3      | 6      | 1      | 9      | 2      | stato iniziale         |
| 2   | 6   | 3      | 6      | 1      | 9      | 2      | 6 già al suo posto     |
| 3   | 1   | 1      | 3      | 6      | 9      | 2      | inserisci 1 all'inizio |
| 4   | 9   | 1      | 3      | 6      | 9      | 2      | 9 già al suo posto     |
| 5   | 2   | 1      | 2      | 3      | 6      | 9      | inserisci 2 tra 1 e 3  |

L'algoritmo esegue 8 confronti tra chiavi e 5 spostamenti, e l'array ordinato è  
 $[1, 2, 3, 6, 9]$ .

4. Array  $a = [25, 12, 50]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                         |
|-----|-----|-----|--------|--------|--------|----------------------------------------------------------------|
| —   | —   | —   | 25     | 12     | 50     | stato iniziale                                                 |
| 2   | 1   | 12  | 25     | 12     | 50     | $key \leftarrow a[2] = 12; j \leftarrow 1$                     |
|     | 0   | 12  | 25     | 25     | 50     | $a[1] = 25 > 12$ : sposta $a[2] \leftarrow 25; j \leftarrow 0$ |
|     | —   | 12  | 12     | 25     | 50     | $j = 0$ : esci; inserisci $a[1] \leftarrow 12$                 |
| 3   | 2   | 50  | 12     | 25     | 50     | $key \leftarrow a[3] = 50; j \leftarrow 2$                     |
|     | —   | 50  | 12     | 25     | 50     | $a[2] = 25 \leq 50$ : esci; inserisci $a[3] \leftarrow 50$     |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | Azione                  |
|-----|-----|--------|--------|--------|-------------------------|
| —   | —   | 25     | 12     | 50     | stato iniziale          |
| 2   | 12  | 12     | 25     | 50     | inserisci 12 all'inizio |
| 3   | 50  | 12     | 25     | 50     | 50 già al suo posto     |

L'algoritmo esegue 2 confronti tra chiavi e 1 spostamento, e l'array ordinato è  
 $[12, 25, 50]$ .

5. Array  $a = [17, 17, 17, 17]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                     |
|-----|-----|-----|--------|--------|--------|--------|------------------------------------------------------------|
| —   | —   | —   | 17     | 17     | 17     | 17     | stato iniziale                                             |
| 2   | 1   | 17  | 17     | 17     | 17     | 17     | $key \leftarrow a[2] = 17; j \leftarrow 1$                 |
|     | —   | 17  | 17     | 17     | 17     | 17     | $a[1] = 17 \leq 17$ : esci; inserisci $a[2] \leftarrow 17$ |
| 3   | 2   | 17  | 17     | 17     | 17     | 17     | $key \leftarrow a[3] = 17; j \leftarrow 2$                 |
|     | —   | 17  | 17     | 17     | 17     | 17     | $a[2] = 17 \leq 17$ : esci; inserisci $a[3] \leftarrow 17$ |
| 4   | 3   | 17  | 17     | 17     | 17     | 17     | $key \leftarrow a[4] = 17; j \leftarrow 3$                 |
|     | —   | 17  | 17     | 17     | 17     | 17     | $a[3] = 17 \leq 17$ : esci; inserisci $a[4] \leftarrow 17$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | key | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione              |
|-----|-----|--------|--------|--------|--------|---------------------|
| –   | –   | 17     | 17     | 17     | 17     | stato iniziale      |
| 2   | 17  | 17     | 17     | 17     | 17     | 17 già al suo posto |
| 3   | 17  | 17     | 17     | 17     | 17     | 17 già al suo posto |
| 4   | 17  | 17     | 17     | 17     | 17     | 17 già al suo posto |

L'algoritmo esegue 3 confronti tra chiavi e 0 spostamenti, e l'array ordinato è

[17, 17, 17, 17].

### Esercizio 3.9 — Algoritmi di ordinamento (E9)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [5, 2, 4, 6, 1, 3]$
2.  $a = [8, 3, 7, 4, 9, 2]$
3.  $a = [6, 5, 4, 3, 2, 1]$
4.  $a = [1, 2, 3, 4, 5, 6]$
5.  $a = [10, 5, 15, 3]$

#### Domande

1. Applicare l'algoritmo **ExchangeSort** a ciascun array, per ordinarlo in ordine non crescente. Mostrare l'esecuzione passo per passo: per ogni confronto, dire se viene eseguito uno scambio e indicare lo stato dell'array dopo il confronto.
2. Per ciascun array, contare i confronti tra chiavi e gli scambi eseguiti dall'algoritmo.

#### Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [5, 2, 4, 6, 1, 3]$  ( $n = 6$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                      |
|-----|-----|--------|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | 5      | 2      | 4      | 6      | 1      | 3      | stato iniziale                                              |
| 1   | 2   | 2      | 5      | 4      | 6      | 1      | 3      | $a[1] = 5 > a[2] = 2$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 2      | 5      | 4      | 6      | 1      | 3      | $a[1] = 2 \leq a[3] = 4$ : nessuno scambio                  |
|     | 4   | 2      | 5      | 4      | 6      | 1      | 3      | $a[1] = 2 \leq a[4] = 6$ : nessuno scambio                  |
|     | 5   | 1      | 5      | 4      | 6      | 2      | 3      | $a[1] = 2 > a[5] = 1$ : scambia $a[1] \leftrightarrow a[5]$ |
|     | 6   | 1      | 5      | 4      | 6      | 2      | 3      | $a[1] = 1 \leq a[6] = 3$ : nessuno scambio                  |
| 2   | 3   | 1      | 4      | 5      | 6      | 2      | 3      | $a[2] = 5 > a[3] = 4$ : scambia $a[2] \leftrightarrow a[3]$ |
|     | 4   | 1      | 4      | 5      | 6      | 2      | 3      | $a[2] = 4 \leq a[4] = 6$ : nessuno scambio                  |
|     | 5   | 1      | 2      | 5      | 6      | 4      | 3      | $a[2] = 4 > a[5] = 2$ : scambia $a[2] \leftrightarrow a[5]$ |
|     | 6   | 1      | 2      | 5      | 6      | 4      | 3      | $a[2] = 2 \leq a[6] = 3$ : nessuno scambio                  |
| 3   | 4   | 1      | 2      | 5      | 6      | 4      | 3      | $a[3] = 5 \leq a[4] = 6$ : nessuno scambio                  |
|     | 5   | 1      | 2      | 4      | 6      | 5      | 3      | $a[3] = 5 > a[5] = 4$ : scambia $a[3] \leftrightarrow a[5]$ |
|     | 6   | 1      | 2      | 3      | 6      | 5      | 4      | $a[3] = 4 > a[6] = 3$ : scambia $a[3] \leftrightarrow a[6]$ |
| 4   | 5   | 1      | 2      | 3      | 5      | 6      | 4      | $a[4] = 6 > a[5] = 5$ : scambia $a[4] \leftrightarrow a[5]$ |
|     | 6   | 1      | 2      | 3      | 4      | 6      | 5      | $a[4] = 5 > a[6] = 4$ : scambia $a[4] \leftrightarrow a[6]$ |
| 5   | 6   | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = 6 > a[6] = 5$ : scambia $a[5] \leftrightarrow a[6]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                 |
|-----|--------|--------|--------|--------|--------|--------|----------------------------------------|
| —   | 5      | 2      | 4      | 6      | 1      | 3      | stato iniziale                         |
| 1   | 1      | 5      | 4      | 6      | 2      | 3      | $a[1] = \min(a[1..6]) = 1$ (2 scambi)  |
| 2   | 1      | 2      | 5      | 6      | 4      | 3      | $a[2] = \min(a[2..6]) = 2$ (2 scambi)  |
| 3   | 1      | 2      | 3      | 6      | 5      | 4      | $a[3] = \min(a[3..6]) = 3$ (2 scambi)  |
| 4   | 1      | 2      | 3      | 4      | 6      | 5      | $a[4] = \min(a[4..6]) = 4$ (2 scambi)  |
| 5   | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = \min(a[5..6]) = 5$ (1 scambio) |

L'algoritmo esegue 15 confronti tra chiavi e 9 scambi, e l'array ordinato è

$$[1, 2, 3, 4, 5, 6].$$

2. Array  $a = [8, 3, 7, 4, 9, 2]$  ( $n = 6$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                      |
|-----|-----|--------|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | 8      | 3      | 7      | 4      | 9      | 2      | stato iniziale                                              |
| 1   | 2   | 3      | 8      | 7      | 4      | 9      | 2      | $a[1] = 8 > a[2] = 3$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 3      | 8      | 7      | 4      | 9      | 2      | $a[1] = 3 \leq a[3] = 7$ : nessuno scambio                  |
|     | 4   | 3      | 8      | 7      | 4      | 9      | 2      | $a[1] = 3 \leq a[4] = 4$ : nessuno scambio                  |
|     | 5   | 3      | 8      | 7      | 4      | 9      | 2      | $a[1] = 3 \leq a[5] = 9$ : nessuno scambio                  |
|     | 6   | 2      | 8      | 7      | 4      | 9      | 3      | $a[1] = 3 > a[6] = 2$ : scambia $a[1] \leftrightarrow a[6]$ |
| 2   | 3   | 2      | 7      | 8      | 4      | 9      | 3      | $a[2] = 8 > a[3] = 7$ : scambia $a[2] \leftrightarrow a[3]$ |
|     | 4   | 2      | 4      | 8      | 7      | 9      | 3      | $a[2] = 7 > a[4] = 4$ : scambia $a[2] \leftrightarrow a[4]$ |
|     | 5   | 2      | 4      | 8      | 7      | 9      | 3      | $a[2] = 4 \leq a[5] = 9$ : nessuno scambio                  |
|     | 6   | 2      | 3      | 8      | 7      | 9      | 4      | $a[2] = 4 > a[6] = 3$ : scambia $a[2] \leftrightarrow a[6]$ |
| 3   | 4   | 2      | 3      | 7      | 8      | 9      | 4      | $a[3] = 8 > a[4] = 7$ : scambia $a[3] \leftrightarrow a[4]$ |
|     | 5   | 2      | 3      | 7      | 8      | 9      | 4      | $a[3] = 7 \leq a[5] = 9$ : nessuno scambio                  |
|     | 6   | 2      | 3      | 4      | 8      | 9      | 7      | $a[3] = 7 > a[6] = 4$ : scambia $a[3] \leftrightarrow a[6]$ |
| 4   | 5   | 2      | 3      | 4      | 8      | 9      | 7      | $a[4] = 8 \leq a[5] = 9$ : nessuno scambio                  |
|     | 6   | 2      | 3      | 4      | 7      | 9      | 8      | $a[4] = 8 > a[6] = 7$ : scambia $a[4] \leftrightarrow a[6]$ |
| 5   | 6   | 2      | 3      | 4      | 7      | 8      | 9      | $a[5] = 9 > a[6] = 8$ : scambia $a[5] \leftrightarrow a[6]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                 |
|-----|--------|--------|--------|--------|--------|--------|----------------------------------------|
| —   | 8      | 3      | 7      | 4      | 9      | 2      | stato iniziale                         |
| 1   | 2      | 8      | 7      | 4      | 9      | 3      | $a[1] = \min(a[1..6]) = 2$ (2 scambi)  |
| 2   | 2      | 3      | 8      | 7      | 9      | 4      | $a[2] = \min(a[2..6]) = 3$ (3 scambi)  |
| 3   | 2      | 3      | 4      | 8      | 9      | 7      | $a[3] = \min(a[3..6]) = 4$ (2 scambi)  |
| 4   | 2      | 3      | 4      | 7      | 9      | 8      | $a[4] = \min(a[4..6]) = 7$ (1 scambio) |
| 5   | 2      | 3      | 4      | 7      | 8      | 9      | $a[5] = \min(a[5..6]) = 8$ (1 scambio) |

L'algoritmo esegue 15 confronti tra chiavi e 9 scambi, e l'array ordinato è

[2, 3, 4, 7, 8, 9].

3. Array  $a = [6, 5, 4, 3, 2, 1]$  ( $n = 6$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                      |
|-----|-----|--------|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | 6      | 5      | 4      | 3      | 2      | 1      | stato iniziale                                              |
| 1   | 2   | 5      | 6      | 4      | 3      | 2      | 1      | $a[1] = 6 > a[2] = 5$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 4      | 6      | 5      | 3      | 2      | 1      | $a[1] = 5 > a[3] = 4$ : scambia $a[1] \leftrightarrow a[3]$ |
|     | 4   | 3      | 6      | 5      | 4      | 2      | 1      | $a[1] = 4 > a[4] = 3$ : scambia $a[1] \leftrightarrow a[4]$ |
|     | 5   | 2      | 6      | 5      | 4      | 3      | 1      | $a[1] = 3 > a[5] = 2$ : scambia $a[1] \leftrightarrow a[5]$ |
|     | 6   | 1      | 6      | 5      | 4      | 3      | 2      | $a[1] = 2 > a[6] = 1$ : scambia $a[1] \leftrightarrow a[6]$ |
| 2   | 3   | 1      | 5      | 6      | 4      | 3      | 2      | $a[2] = 6 > a[3] = 5$ : scambia $a[2] \leftrightarrow a[3]$ |
|     | 4   | 1      | 4      | 6      | 5      | 3      | 2      | $a[2] = 5 > a[4] = 4$ : scambia $a[2] \leftrightarrow a[4]$ |
|     | 5   | 1      | 3      | 6      | 5      | 4      | 2      | $a[2] = 4 > a[5] = 3$ : scambia $a[2] \leftrightarrow a[5]$ |
|     | 6   | 1      | 2      | 6      | 5      | 4      | 3      | $a[2] = 3 > a[6] = 2$ : scambia $a[2] \leftrightarrow a[6]$ |
| 3   | 4   | 1      | 2      | 5      | 6      | 4      | 3      | $a[3] = 6 > a[4] = 5$ : scambia $a[3] \leftrightarrow a[4]$ |
|     | 5   | 1      | 2      | 4      | 6      | 5      | 3      | $a[3] = 5 > a[5] = 4$ : scambia $a[3] \leftrightarrow a[5]$ |
|     | 6   | 1      | 2      | 3      | 6      | 5      | 4      | $a[3] = 4 > a[6] = 3$ : scambia $a[3] \leftrightarrow a[6]$ |
| 4   | 5   | 1      | 2      | 3      | 5      | 6      | 4      | $a[4] = 6 > a[5] = 5$ : scambia $a[4] \leftrightarrow a[5]$ |
|     | 6   | 1      | 2      | 3      | 4      | 6      | 5      | $a[4] = 5 > a[6] = 4$ : scambia $a[4] \leftrightarrow a[6]$ |
| 5   | 6   | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = 6 > a[6] = 5$ : scambia $a[5] \leftrightarrow a[6]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                 |
|-----|--------|--------|--------|--------|--------|--------|----------------------------------------|
| —   | 6      | 5      | 4      | 3      | 2      | 1      | stato iniziale                         |
| 1   | 1      | 6      | 5      | 4      | 3      | 2      | $a[1] = \min(a[1..6]) = 1$ (5 scambi)  |
| 2   | 1      | 2      | 6      | 5      | 4      | 3      | $a[2] = \min(a[2..6]) = 2$ (4 scambi)  |
| 3   | 1      | 2      | 3      | 6      | 5      | 4      | $a[3] = \min(a[3..6]) = 3$ (3 scambi)  |
| 4   | 1      | 2      | 3      | 4      | 6      | 5      | $a[4] = \min(a[4..6]) = 4$ (2 scambi)  |
| 5   | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = \min(a[5..6]) = 5$ (1 scambio) |

L'algoritmo esegue 15 confronti tra chiavi e 15 scambi, e l'array ordinato è

$$[1, 2, 3, 4, 5, 6].$$

4. Array  $a = [1, 2, 3, 4, 5, 6]$  ( $n = 6$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                     |
|-----|-----|--------|--------|--------|--------|--------|--------|--------------------------------------------|
| —   | —   | 1      | 2      | 3      | 4      | 5      | 6      | stato iniziale                             |
| 1   | 2   | 1      | 2      | 3      | 4      | 5      | 6      | $a[1] = 1 \leq a[2] = 2$ : nessuno scambio |
|     | 3   | 1      | 2      | 3      | 4      | 5      | 6      | $a[1] = 1 \leq a[3] = 3$ : nessuno scambio |
|     | 4   | 1      | 2      | 3      | 4      | 5      | 6      | $a[1] = 1 \leq a[4] = 4$ : nessuno scambio |
|     | 5   | 1      | 2      | 3      | 4      | 5      | 6      | $a[1] = 1 \leq a[5] = 5$ : nessuno scambio |
|     | 6   | 1      | 2      | 3      | 4      | 5      | 6      | $a[1] = 1 \leq a[6] = 6$ : nessuno scambio |
| 2   | 3   | 1      | 2      | 3      | 4      | 5      | 6      | $a[2] = 2 \leq a[3] = 3$ : nessuno scambio |
|     | 4   | 1      | 2      | 3      | 4      | 5      | 6      | $a[2] = 2 \leq a[4] = 4$ : nessuno scambio |
|     | 5   | 1      | 2      | 3      | 4      | 5      | 6      | $a[2] = 2 \leq a[5] = 5$ : nessuno scambio |
|     | 6   | 1      | 2      | 3      | 4      | 5      | 6      | $a[2] = 2 \leq a[6] = 6$ : nessuno scambio |
| 3   | 4   | 1      | 2      | 3      | 4      | 5      | 6      | $a[3] = 3 \leq a[4] = 4$ : nessuno scambio |
|     | 5   | 1      | 2      | 3      | 4      | 5      | 6      | $a[3] = 3 \leq a[5] = 5$ : nessuno scambio |
|     | 6   | 1      | 2      | 3      | 4      | 5      | 6      | $a[3] = 3 \leq a[6] = 6$ : nessuno scambio |
| 4   | 5   | 1      | 2      | 3      | 4      | 5      | 6      | $a[4] = 4 \leq a[5] = 5$ : nessuno scambio |
|     | 6   | 1      | 2      | 3      | 4      | 5      | 6      | $a[4] = 4 \leq a[6] = 6$ : nessuno scambio |
| 5   | 6   | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = 5 \leq a[6] = 6$ : nessuno scambio |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                |
|-----|--------|--------|--------|--------|--------|--------|---------------------------------------|
| —   | 1      | 2      | 3      | 4      | 5      | 6      | stato iniziale                        |
| 1   | 1      | 2      | 3      | 4      | 5      | 6      | $a[1] = \min(a[1..6]) = 1$ (0 scambi) |
| 2   | 1      | 2      | 3      | 4      | 5      | 6      | $a[2] = \min(a[2..6]) = 2$ (0 scambi) |
| 3   | 1      | 2      | 3      | 4      | 5      | 6      | $a[3] = \min(a[3..6]) = 3$ (0 scambi) |
| 4   | 1      | 2      | 3      | 4      | 5      | 6      | $a[4] = \min(a[4..6]) = 4$ (0 scambi) |
| 5   | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = \min(a[5..6]) = 5$ (0 scambi) |

L'algoritmo esegue 15 confronti tra chiavi e 0 scambi, e l'array ordinato è

$$[1, 2, 3, 4, 5, 6].$$

5. Array  $a = [10, 5, 15, 3]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                        |
|-----|-----|--------|--------|--------|--------|---------------------------------------------------------------|
| —   | —   | 10     | 5      | 15     | 3      | stato iniziale                                                |
| 1   | 2   | 5      | 10     | 15     | 3      | $a[1] = 10 > a[2] = 5$ : scambia $a[1] \leftrightarrow a[2]$  |
|     | 3   | 5      | 10     | 15     | 3      | $a[1] = 5 \leq a[3] = 15$ : nessuno scambio                   |
|     | 4   | 3      | 10     | 15     | 5      | $a[1] = 5 > a[4] = 3$ : scambia $a[1] \leftrightarrow a[4]$   |
| 2   | 3   | 3      | 10     | 15     | 5      | $a[2] = 10 \leq a[3] = 15$ : nessuno scambio                  |
|     | 4   | 3      | 5      | 15     | 10     | $a[2] = 10 > a[4] = 5$ : scambia $a[2] \leftrightarrow a[4]$  |
| 3   | 4   | 3      | 5      | 10     | 15     | $a[3] = 15 > a[4] = 10$ : scambia $a[3] \leftrightarrow a[4]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                  |
|-----|--------|--------|--------|--------|-----------------------------------------|
| –   | 10     | 5      | 15     | 3      | stato iniziale                          |
| 1   | 3      | 10     | 15     | 5      | $a[1] = \min(a[1..4]) = 3$ (2 scambi)   |
| 2   | 3      | 5      | 15     | 10     | $a[2] = \min(a[2..4]) = 5$ (1 scambio)  |
| 3   | 3      | 5      | 10     | 15     | $a[3] = \min(a[3..4]) = 10$ (1 scambio) |

L'algoritmo esegue 6 confronti tra chiavi e 4 scambi, e l'array ordinato è

$[3, 5, 10, 15]$ .

### Esercizio 3.10 — Algoritmi di ordinamento (E10)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [7, 2, 9, 4]$
2.  $a = [15, 10, 5]$
3.  $a = [3, 1, 4, 1, 5, 9, 2, 6]$
4.  $a = [20, 18, 16, 14]$
5.  $a = [11, 11, 11]$

#### Domande

1. Applicare l'algoritmo **ExchangeSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni confronto, dire se viene eseguito uno scambio e indicare lo stato dell'array dopo il confronto.
2. Per ciascun array, contare i confronti tra chiavi e gli scambi eseguiti dall'algoritmo.

#### Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [7, 2, 9, 4]$  ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                      |
|-----|-----|--------|--------|--------|--------|-------------------------------------------------------------|
| –   | –   | 7      | 2      | 9      | 4      | stato iniziale                                              |
| 1   | 2   | 2      | 7      | 9      | 4      | $a[1] = 7 > a[2] = 2$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 2      | 7      | 9      | 4      | $a[1] = 2 \leq a[3] = 9$ : nessuno scambio                  |
|     | 4   | 2      | 7      | 9      | 4      | $a[1] = 2 \leq a[4] = 4$ : nessuno scambio                  |
| 2   | 3   | 2      | 7      | 9      | 4      | $a[2] = 7 \leq a[3] = 9$ : nessuno scambio                  |
|     | 4   | 2      | 4      | 9      | 7      | $a[2] = 7 > a[4] = 4$ : scambia $a[2] \leftrightarrow a[4]$ |
| 3   | 4   | 2      | 4      | 7      | 9      | $a[3] = 9 > a[4] = 7$ : scambia $a[3] \leftrightarrow a[4]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                 |
|-----|--------|--------|--------|--------|----------------------------------------|
| –   | 7      | 2      | 9      | 4      | stato iniziale                         |
| 1   | 2      | 7      | 9      | 4      | $a[1] = \min(a[1..4]) = 2$ (1 scambio) |
| 2   | 2      | 4      | 9      | 7      | $a[2] = \min(a[2..4]) = 4$ (1 scambio) |
| 3   | 2      | 4      | 7      | 9      | $a[3] = \min(a[3..4]) = 7$ (1 scambio) |

L'algoritmo esegue 6 confronti tra chiavi e 3 scambi, e l'array ordinato è

$[2, 4, 7, 9]$ .

2. Array  $a = [15, 10, 5]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                        |
|-----|-----|--------|--------|--------|---------------------------------------------------------------|
| –   | –   | 15     | 10     | 5      | stato iniziale                                                |
| 1   | 2   | 10     | 15     | 5      | $a[1] = 15 > a[2] = 10$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 5      | 15     | 10     | $a[1] = 10 > a[3] = 5$ : scambia $a[1] \leftrightarrow a[3]$  |
| 2   | 3   | 5      | 10     | 15     | $a[2] = 15 > a[3] = 10$ : scambia $a[2] \leftrightarrow a[3]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                  |
|-----|--------|--------|--------|-----------------------------------------|
| –   | 15     | 10     | 5      | stato iniziale                          |
| 1   | 5      | 15     | 10     | $a[1] = \min(a[1..3]) = 5$ (2 scambi)   |
| 2   | 5      | 10     | 15     | $a[2] = \min(a[2..3]) = 10$ (1 scambio) |

L'algoritmo esegue 3 confronti tra chiavi e 3 scambi, e l'array ordinato è

$[5, 10, 15]$ .

3. Array  $a = [3, 1, 4, 1, 5, 9, 2, 6]$  ( $n = 8$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | $a[7]$ | $a[8]$ | Azione                                                      |
|-----|-----|--------|--------|--------|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| –   | –   | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | stato iniziale                                              |
| 1   | 2   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[1] = 3 > a[2] = 1$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[1] = 1 \leq a[3] = 4$ : nessuno scambio                  |
|     | 4   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[1] = 1 \leq a[4] = 1$ : nessuno scambio                  |
|     | 5   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[1] = 1 \leq a[5] = 5$ : nessuno scambio                  |
|     | 6   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[1] = 1 \leq a[6] = 9$ : nessuno scambio                  |
|     | 7   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[1] = 1 \leq a[7] = 2$ : nessuno scambio                  |
|     | 8   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[1] = 1 \leq a[8] = 6$ : nessuno scambio                  |
| 2   | 3   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[2] = 3 \leq a[3] = 4$ : nessuno scambio                  |
|     | 4   | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[2] = 3 > a[4] = 1$ : scambia $a[2] \leftrightarrow a[4]$ |
|     | 5   | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[2] = 1 \leq a[5] = 5$ : nessuno scambio                  |
|     | 6   | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[2] = 1 \leq a[6] = 9$ : nessuno scambio                  |
|     | 7   | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[2] = 1 \leq a[7] = 2$ : nessuno scambio                  |
|     | 8   | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[2] = 1 \leq a[8] = 6$ : nessuno scambio                  |
| 3   | 4   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | $a[3] = 4 > a[4] = 3$ : scambia $a[3] \leftrightarrow a[4]$ |
|     | 5   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | $a[3] = 3 \leq a[5] = 5$ : nessuno scambio                  |
|     | 6   | 1      | 1      | 3      | 4      | 5      | 9      | 2      | 6      | $a[3] = 3 \leq a[6] = 9$ : nessuno scambio                  |
|     | 7   | 1      | 1      | 2      | 4      | 5      | 9      | 3      | 6      | $a[3] = 3 > a[7] = 2$ : scambia $a[3] \leftrightarrow a[7]$ |
|     | 8   | 1      | 1      | 2      | 4      | 5      | 9      | 3      | 6      | $a[3] = 2 \leq a[8] = 6$ : nessuno scambio                  |
| 4   | 5   | 1      | 1      | 2      | 4      | 5      | 9      | 3      | 6      | $a[4] = 4 \leq a[5] = 5$ : nessuno scambio                  |
|     | 6   | 1      | 1      | 2      | 4      | 5      | 9      | 3      | 6      | $a[4] = 4 \leq a[6] = 9$ : nessuno scambio                  |
|     | 7   | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[4] = 4 > a[7] = 3$ : scambia $a[4] \leftrightarrow a[7]$ |
|     | 8   | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[4] = 3 \leq a[8] = 6$ : nessuno scambio                  |
| 5   | 6   | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[5] = 5 \leq a[6] = 9$ : nessuno scambio                  |
|     | 7   | 1      | 1      | 2      | 3      | 4      | 9      | 5      | 6      | $a[5] = 5 > a[7] = 4$ : scambia $a[5] \leftrightarrow a[7]$ |
|     | 8   | 1      | 1      | 2      | 3      | 4      | 9      | 5      | 6      | $a[5] = 4 \leq a[8] = 6$ : nessuno scambio                  |
| 6   | 7   | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 6      | $a[6] = 9 > a[7] = 5$ : scambia $a[6] \leftrightarrow a[7]$ |
|     | 8   | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 6      | $a[6] = 5 \leq a[8] = 6$ : nessuno scambio                  |
| 7   | 8   | 1      | 1      | 2      | 3      | 4      | 5      | 6      | 9      | $a[7] = 9 > a[8] = 6$ : scambia $a[7] \leftrightarrow a[8]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | $a[7]$ | $a[8]$ | Azione                                 |
|-----|--------|--------|--------|--------|--------|--------|--------|--------|----------------------------------------|
| –   | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | stato iniziale                         |
| 1   | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[1] = \min(a[1..8]) = 1$ (1 scambio) |
| 2   | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[2] = \min(a[2..8]) = 1$ (1 scambio) |
| 3   | 1      | 1      | 2      | 4      | 5      | 9      | 3      | 6      | $a[3] = \min(a[3..8]) = 2$ (2 scambi)  |
| 4   | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[4] = \min(a[4..8]) = 3$ (1 scambio) |
| 5   | 1      | 1      | 2      | 3      | 4      | 9      | 5      | 6      | $a[5] = \min(a[5..8]) = 4$ (1 scambio) |
| 6   | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 6      | $a[6] = \min(a[6..8]) = 5$ (1 scambio) |
| 7   | 1      | 1      | 2      | 3      | 4      | 5      | 6      | 9      | $a[7] = \min(a[7..8]) = 6$ (1 scambio) |

L'algoritmo esegue 28 confronti tra chiavi e 8 scambi, e l'array ordinato è

$[1, 1, 2, 3, 4, 5, 6, 9]$ .

4. Array  $a = [20, 18, 16, 14]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                        |
|-----|-----|--------|--------|--------|--------|---------------------------------------------------------------|
| —   | —   | 20     | 18     | 16     | 14     | stato iniziale                                                |
| 1   | 2   | 18     | 20     | 16     | 14     | $a[1] = 20 > a[2] = 18$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 16     | 20     | 18     | 14     | $a[1] = 18 > a[3] = 16$ : scambia $a[1] \leftrightarrow a[3]$ |
|     | 4   | 14     | 20     | 18     | 16     | $a[1] = 16 > a[4] = 14$ : scambia $a[1] \leftrightarrow a[4]$ |
| 2   | 3   | 14     | 18     | 20     | 16     | $a[2] = 20 > a[3] = 18$ : scambia $a[2] \leftrightarrow a[3]$ |
|     | 4   | 14     | 16     | 20     | 18     | $a[2] = 18 > a[4] = 16$ : scambia $a[2] \leftrightarrow a[4]$ |
| 3   | 4   | 14     | 16     | 18     | 20     | $a[3] = 20 > a[4] = 18$ : scambia $a[3] \leftrightarrow a[4]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                  |
|-----|--------|--------|--------|--------|-----------------------------------------|
| —   | 20     | 18     | 16     | 14     | stato iniziale                          |
| 1   | 14     | 20     | 18     | 16     | $a[1] = \min(a[1..4]) = 14$ (3 scambi)  |
| 2   | 14     | 16     | 20     | 18     | $a[2] = \min(a[2..4]) = 16$ (2 scambi)  |
| 3   | 14     | 16     | 18     | 20     | $a[3] = \min(a[3..4]) = 18$ (1 scambio) |

L'algoritmo esegue 6 confronti tra chiavi e 6 scambi, e l'array ordinato è

[14, 16, 18, 20].

5. Array  $a = [11, 11, 11]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                       |
|-----|-----|--------|--------|--------|----------------------------------------------|
| —   | —   | 11     | 11     | 11     | stato iniziale                               |
| 1   | 2   | 11     | 11     | 11     | $a[1] = 11 \leq a[2] = 11$ : nessuno scambio |
|     | 3   | 11     | 11     | 11     | $a[1] = 11 \leq a[3] = 11$ : nessuno scambio |
| 2   | 3   | 11     | 11     | 11     | $a[2] = 11 \leq a[3] = 11$ : nessuno scambio |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                 |
|-----|--------|--------|--------|----------------------------------------|
| —   | 11     | 11     | 11     | stato iniziale                         |
| 1   | 11     | 11     | 11     | $a[1] = \min(a[1..3]) = 11$ (0 scambi) |
| 2   | 11     | 11     | 11     | $a[2] = \min(a[2..3]) = 11$ (0 scambi) |

L'algoritmo esegue 3 confronti tra chiavi e 0 scambi, e l'array ordinato è

[11, 11, 11].

**Esercizio 3.11 — Algoritmi di ordinamento (E11)**

Consideriamo i seguenti array di numeri naturali:

1.  $a = [9, 3, 7, 2]$
2.  $a = [12, 4, 18, 6, 10]$
3.  $a = [5, 4, 3, 2, 1]$
4.  $a = [100, 50, 25]$
5.  $a = [8]$

**Domande**

1. Applicare l'algoritmo **ExchangeSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni confronto, dire se viene eseguito uno scambio e indicare lo stato dell'array dopo il confronto.
2. Per ciascun array, contare i confronti tra chiavi e gli scambi eseguiti dall'algoritmo.

**Soluzione**

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [9, 3, 7, 2]$  ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                      |
|-----|-----|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | 9      | 3      | 7      | 2      | stato iniziale                                              |
| 1   | 2   | 3      | 9      | 7      | 2      | $a[1] = 9 > a[2] = 3$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 3      | 9      | 7      | 2      | $a[1] = 3 \leq a[3] = 7$ : nessuno scambio                  |
|     | 4   | 2      | 9      | 7      | 3      | $a[1] = 3 > a[4] = 2$ : scambia $a[1] \leftrightarrow a[4]$ |
| 2   | 3   | 2      | 7      | 9      | 3      | $a[2] = 9 > a[3] = 7$ : scambia $a[2] \leftrightarrow a[3]$ |
|     | 4   | 2      | 3      | 9      | 7      | $a[2] = 7 > a[4] = 3$ : scambia $a[2] \leftrightarrow a[4]$ |
| 3   | 4   | 2      | 3      | 7      | 9      | $a[3] = 9 > a[4] = 7$ : scambia $a[3] \leftrightarrow a[4]$ |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                 |
|-----|--------|--------|--------|--------|----------------------------------------|
| —   | 9      | 3      | 7      | 2      | stato iniziale                         |
| 1   | 2      | 9      | 7      | 3      | $a[1] = \min(a[1..4]) = 2$ (2 scambi)  |
| 2   | 2      | 3      | 9      | 7      | $a[2] = \min(a[2..4]) = 3$ (2 scambi)  |
| 3   | 2      | 3      | 7      | 9      | $a[3] = \min(a[3..4]) = 7$ (1 scambio) |

L'algoritmo esegue 6 confronti tra chiavi e 5 scambi, e l'array ordinato è

$[2, 3, 7, 9]$ .

2. Array  $a = [12, 4, 18, 6, 10]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                        |
|-----|-----|--------|--------|--------|--------|--------|---------------------------------------------------------------|
| —   | —   | 12     | 4      | 18     | 6      | 10     | stato iniziale                                                |
| 1   | 2   | 4      | 12     | 18     | 6      | 10     | $a[1] = 12 > a[2] = 4$ : scambia $a[1] \leftrightarrow a[2]$  |
|     | 3   | 4      | 12     | 18     | 6      | 10     | $a[1] = 4 \leq a[3] = 18$ : nessuno scambio                   |
|     | 4   | 4      | 12     | 18     | 6      | 10     | $a[1] = 4 \leq a[4] = 6$ : nessuno scambio                    |
|     | 5   | 4      | 12     | 18     | 6      | 10     | $a[1] = 4 \leq a[5] = 10$ : nessuno scambio                   |
| 2   | 3   | 4      | 12     | 18     | 6      | 10     | $a[2] = 12 \leq a[3] = 18$ : nessuno scambio                  |
|     | 4   | 4      | 6      | 18     | 12     | 10     | $a[2] = 12 > a[4] = 6$ : scambia $a[2] \leftrightarrow a[4]$  |
|     | 5   | 4      | 6      | 18     | 12     | 10     | $a[2] = 6 \leq a[5] = 10$ : nessuno scambio                   |
| 3   | 4   | 4      | 6      | 12     | 18     | 10     | $a[3] = 18 > a[4] = 12$ : scambia $a[3] \leftrightarrow a[4]$ |
|     | 5   | 4      | 6      | 10     | 18     | 12     | $a[3] = 12 > a[5] = 10$ : scambia $a[3] \leftrightarrow a[5]$ |
| 4   | 5   | 4      | 6      | 10     | 12     | 18     | $a[4] = 18 > a[5] = 12$ : scambia $a[4] \leftrightarrow a[5]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                  |
|-----|--------|--------|--------|--------|--------|-----------------------------------------|
| —   | 12     | 4      | 18     | 6      | 10     | stato iniziale                          |
| 1   | 4      | 12     | 18     | 6      | 10     | $a[1] = \min(a[1..5]) = 4$ (1 scambio)  |
| 2   | 4      | 6      | 18     | 12     | 10     | $a[2] = \min(a[2..5]) = 6$ (1 scambio)  |
| 3   | 4      | 6      | 10     | 18     | 12     | $a[3] = \min(a[3..5]) = 10$ (2 scambi)  |
| 4   | 4      | 6      | 10     | 12     | 18     | $a[4] = \min(a[4..5]) = 12$ (1 scambio) |

L'algoritmo esegue 10 confronti tra chiavi e 5 scambi, e l'array ordinato è

$[4, 6, 10, 12, 18]$ .

3. Array  $a = [5, 4, 3, 2, 1]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                      |
|-----|-----|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | 5      | 4      | 3      | 2      | 1      | stato iniziale                                              |
| 1   | 2   | 4      | 5      | 3      | 2      | 1      | $a[1] = 5 > a[2] = 4$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 3      | 5      | 4      | 2      | 1      | $a[1] = 4 > a[3] = 3$ : scambia $a[1] \leftrightarrow a[3]$ |
|     | 4   | 2      | 5      | 4      | 3      | 1      | $a[1] = 3 > a[4] = 2$ : scambia $a[1] \leftrightarrow a[4]$ |
|     | 5   | 1      | 5      | 4      | 3      | 2      | $a[1] = 2 > a[5] = 1$ : scambia $a[1] \leftrightarrow a[5]$ |
| 2   | 3   | 1      | 4      | 5      | 3      | 2      | $a[2] = 5 > a[3] = 4$ : scambia $a[2] \leftrightarrow a[3]$ |
|     | 4   | 1      | 3      | 5      | 4      | 2      | $a[2] = 4 > a[4] = 3$ : scambia $a[2] \leftrightarrow a[4]$ |
|     | 5   | 1      | 2      | 5      | 4      | 3      | $a[2] = 3 > a[5] = 2$ : scambia $a[2] \leftrightarrow a[5]$ |
| 3   | 4   | 1      | 2      | 4      | 5      | 3      | $a[3] = 5 > a[4] = 4$ : scambia $a[3] \leftrightarrow a[4]$ |
|     | 5   | 1      | 2      | 3      | 5      | 4      | $a[3] = 4 > a[5] = 3$ : scambia $a[3] \leftrightarrow a[5]$ |
| 4   | 5   | 1      | 2      | 3      | 4      | 5      | $a[4] = 5 > a[5] = 4$ : scambia $a[4] \leftrightarrow a[5]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                 |
|-----|--------|--------|--------|--------|--------|----------------------------------------|
| –   | 5      | 4      | 3      | 2      | 1      | stato iniziale                         |
| 1   | 1      | 5      | 4      | 3      | 2      | $a[1] = \min(a[1..5]) = 1$ (4 scambi)  |
| 2   | 1      | 2      | 5      | 4      | 3      | $a[2] = \min(a[2..5]) = 2$ (3 scambi)  |
| 3   | 1      | 2      | 3      | 5      | 4      | $a[3] = \min(a[3..5]) = 3$ (2 scambi)  |
| 4   | 1      | 2      | 3      | 4      | 5      | $a[4] = \min(a[4..5]) = 4$ (1 scambio) |

L'algoritmo esegue 10 confronti tra chiavi e 10 scambi, e l'array ordinato è

[1, 2, 3, 4, 5].

4. Array  $a = [100, 50, 25]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                         |
|-----|-----|--------|--------|--------|----------------------------------------------------------------|
| –   | –   | 100    | 50     | 25     | stato iniziale                                                 |
| 1   | 2   | 50     | 100    | 25     | $a[1] = 100 > a[2] = 50$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 25     | 100    | 50     | $a[1] = 50 > a[3] = 25$ : scambia $a[1] \leftrightarrow a[3]$  |
| 2   | 3   | 25     | 50     | 100    | $a[2] = 100 > a[3] = 50$ : scambia $a[2] \leftrightarrow a[3]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                  |
|-----|--------|--------|--------|-----------------------------------------|
| –   | 100    | 50     | 25     | stato iniziale                          |
| 1   | 25     | 100    | 50     | $a[1] = \min(a[1..3]) = 25$ (2 scambi)  |
| 2   | 25     | 50     | 100    | $a[2] = \min(a[2..3]) = 50$ (1 scambio) |

L'algoritmo esegue 3 confronti tra chiavi e 3 scambi, e l'array ordinato è

[25, 50, 100].

5. Array  $a = [8]$  ( $n = 1$ ).

L'array ha un solo elemento: il ciclo esterno non viene eseguito e l'array è già ordinato.

| $a[1]$ | Azione                                                        |
|--------|---------------------------------------------------------------|
| 8      | stato iniziale: $n = 1$ , il ciclo esterno non viene eseguito |

L'array ordinato è [8], ottenuto senza alcun confronto tra chiavi e senza alcuno scambio.

**Esercizio 3.12 — Algoritmi di ordinamento (E12)**

Consideriamo i seguenti array di numeri naturali:

1.  $a = [6, 1, 8, 2, 5]$
2.  $a = [13, 11, 9, 7]$
3.  $a = [2, 8, 4, 6]$
4.  $a = [30, 20, 10]$
5.  $a = [5, 10, 15, 20]$

**Domande**

1. Applicare l'algoritmo **ExchangeSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni confronto, dire se viene eseguito uno scambio e indicare lo stato dell'array dopo il confronto.
2. Per ciascun array, contare i confronti tra chiavi e gli scambi eseguiti dall'algoritmo.

**Soluzione**

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [6, 1, 8, 2, 5]$  ( $n = 5$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                      |
|-----|-----|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| —   | —   | 6      | 1      | 8      | 2      | 5      | stato iniziale                                              |
| 1   | 2   | 1      | 6      | 8      | 2      | 5      | $a[1] = 6 > a[2] = 1$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 1      | 6      | 8      | 2      | 5      | $a[1] = 1 \leq a[3] = 8$ : nessuno scambio                  |
|     | 4   | 1      | 6      | 8      | 2      | 5      | $a[1] = 1 \leq a[4] = 2$ : nessuno scambio                  |
|     | 5   | 1      | 6      | 8      | 2      | 5      | $a[1] = 1 \leq a[5] = 5$ : nessuno scambio                  |
| 2   | 3   | 1      | 6      | 8      | 2      | 5      | $a[2] = 6 \leq a[3] = 8$ : nessuno scambio                  |
|     | 4   | 1      | 2      | 8      | 6      | 5      | $a[2] = 6 > a[4] = 2$ : scambia $a[2] \leftrightarrow a[4]$ |
|     | 5   | 1      | 2      | 8      | 6      | 5      | $a[2] = 2 \leq a[5] = 5$ : nessuno scambio                  |
| 3   | 4   | 1      | 2      | 6      | 8      | 5      | $a[3] = 8 > a[4] = 6$ : scambia $a[3] \leftrightarrow a[4]$ |
|     | 5   | 1      | 2      | 5      | 8      | 6      | $a[3] = 6 > a[5] = 5$ : scambia $a[3] \leftrightarrow a[5]$ |
| 4   | 5   | 1      | 2      | 5      | 6      | 8      | $a[4] = 8 > a[5] = 6$ : scambia $a[4] \leftrightarrow a[5]$ |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                 |
|-----|--------|--------|--------|--------|--------|----------------------------------------|
| –   | 6      | 1      | 8      | 2      | 5      | stato iniziale                         |
| 1   | 1      | 6      | 8      | 2      | 5      | $a[1] = \min(a[1..5]) = 1$ (1 scambio) |
| 2   | 1      | 2      | 8      | 6      | 5      | $a[2] = \min(a[2..5]) = 2$ (1 scambio) |
| 3   | 1      | 2      | 5      | 8      | 6      | $a[3] = \min(a[3..5]) = 5$ (2 scambi)  |
| 4   | 1      | 2      | 5      | 6      | 8      | $a[4] = \min(a[4..5]) = 6$ (1 scambio) |

L'algoritmo esegue 10 confronti tra chiavi e 5 scambi, e l'array ordinato è  
 $[1, 2, 5, 6, 8]$ .

2. Array  $a = [13, 11, 9, 7]$  ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                        |
|-----|-----|--------|--------|--------|--------|---------------------------------------------------------------|
| –   | –   | 13     | 11     | 9      | 7      | stato iniziale                                                |
| 1   | 2   | 11     | 13     | 9      | 7      | $a[1] = 13 > a[2] = 11$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 9      | 13     | 11     | 7      | $a[1] = 11 > a[3] = 9$ : scambia $a[1] \leftrightarrow a[3]$  |
|     | 4   | 7      | 13     | 11     | 9      | $a[1] = 9 > a[4] = 7$ : scambia $a[1] \leftrightarrow a[4]$   |
| 2   | 3   | 7      | 11     | 13     | 9      | $a[2] = 13 > a[3] = 11$ : scambia $a[2] \leftrightarrow a[3]$ |
|     | 4   | 7      | 9      | 13     | 11     | $a[2] = 11 > a[4] = 9$ : scambia $a[2] \leftrightarrow a[4]$  |
| 3   | 4   | 7      | 9      | 11     | 13     | $a[3] = 13 > a[4] = 11$ : scambia $a[3] \leftrightarrow a[4]$ |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                  |
|-----|--------|--------|--------|--------|-----------------------------------------|
| –   | 13     | 11     | 9      | 7      | stato iniziale                          |
| 1   | 7      | 13     | 11     | 9      | $a[1] = \min(a[1..4]) = 7$ (3 scambi)   |
| 2   | 7      | 9      | 13     | 11     | $a[2] = \min(a[2..4]) = 9$ (2 scambi)   |
| 3   | 7      | 9      | 11     | 13     | $a[3] = \min(a[3..4]) = 11$ (1 scambio) |

L'algoritmo esegue 6 confronti tra chiavi e 6 scambi, e l'array ordinato è  
 $[7, 9, 11, 13]$ .

3. Array  $a = [2, 8, 4, 6]$  ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                      |
|-----|-----|--------|--------|--------|--------|-------------------------------------------------------------|
| –   | –   | 2      | 8      | 4      | 6      | stato iniziale                                              |
| 1   | 2   | 2      | 8      | 4      | 6      | $a[1] = 2 \leq a[2] = 8$ : nessuno scambio                  |
|     | 3   | 2      | 8      | 4      | 6      | $a[1] = 2 \leq a[3] = 4$ : nessuno scambio                  |
|     | 4   | 2      | 8      | 4      | 6      | $a[1] = 2 \leq a[4] = 6$ : nessuno scambio                  |
| 2   | 3   | 2      | 4      | 8      | 6      | $a[2] = 8 > a[3] = 4$ : scambia $a[2] \leftrightarrow a[3]$ |
|     | 4   | 2      | 4      | 8      | 6      | $a[2] = 4 \leq a[4] = 6$ : nessuno scambio                  |
| 3   | 4   | 2      | 4      | 6      | 8      | $a[3] = 8 > a[4] = 6$ : scambia $a[3] \leftrightarrow a[4]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                 |
|-----|--------|--------|--------|--------|----------------------------------------|
| —   | 2      | 8      | 4      | 6      | stato iniziale                         |
| 1   | 2      | 8      | 4      | 6      | $a[1] = \min(a[1..4]) = 2$ (0 scambi)  |
| 2   | 2      | 4      | 8      | 6      | $a[2] = \min(a[2..4]) = 4$ (1 scambio) |
| 3   | 2      | 4      | 6      | 8      | $a[3] = \min(a[3..4]) = 6$ (1 scambio) |

L'algoritmo esegue 6 confronti tra chiavi e 2 scambi, e l'array ordinato è  
 $[2, 4, 6, 8]$ .

4. Array  $a = [30, 20, 10]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                        |
|-----|-----|--------|--------|--------|---------------------------------------------------------------|
| —   | —   | 30     | 20     | 10     | stato iniziale                                                |
| 1   | 2   | 20     | 30     | 10     | $a[1] = 30 > a[2] = 20$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 10     | 30     | 20     | $a[1] = 20 > a[3] = 10$ : scambia $a[1] \leftrightarrow a[3]$ |
| 2   | 3   | 10     | 20     | 30     | $a[2] = 30 > a[3] = 20$ : scambia $a[2] \leftrightarrow a[3]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                  |
|-----|--------|--------|--------|-----------------------------------------|
| —   | 30     | 20     | 10     | stato iniziale                          |
| 1   | 10     | 30     | 20     | $a[1] = \min(a[1..3]) = 10$ (2 scambi)  |
| 2   | 10     | 20     | 30     | $a[2] = \min(a[2..3]) = 20$ (1 scambio) |

L'algoritmo esegue 3 confronti tra chiavi e 3 scambi, e l'array ordinato è  
 $[10, 20, 30]$ .

5. Array  $a = [5, 10, 15, 20]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                       |
|-----|-----|--------|--------|--------|--------|----------------------------------------------|
| —   | —   | 5      | 10     | 15     | 20     | stato iniziale                               |
| 1   | 2   | 5      | 10     | 15     | 20     | $a[1] = 5 \leq a[2] = 10$ : nessuno scambio  |
|     | 3   | 5      | 10     | 15     | 20     | $a[1] = 5 \leq a[3] = 15$ : nessuno scambio  |
|     | 4   | 5      | 10     | 15     | 20     | $a[1] = 5 \leq a[4] = 20$ : nessuno scambio  |
| 2   | 3   | 5      | 10     | 15     | 20     | $a[2] = 10 \leq a[3] = 15$ : nessuno scambio |
|     | 4   | 5      | 10     | 15     | 20     | $a[2] = 10 \leq a[4] = 20$ : nessuno scambio |
| 3   | 4   | 5      | 10     | 15     | 20     | $a[3] = 15 \leq a[4] = 20$ : nessuno scambio |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                 |
|-----|--------|--------|--------|--------|----------------------------------------|
| –   | 5      | 10     | 15     | 20     | stato iniziale                         |
| 1   | 5      | 10     | 15     | 20     | $a[1] = \min(a[1..4]) = 5$ (0 scambi)  |
| 2   | 5      | 10     | 15     | 20     | $a[2] = \min(a[2..4]) = 10$ (0 scambi) |
| 3   | 5      | 10     | 15     | 20     | $a[3] = \min(a[3..4]) = 15$ (0 scambi) |

L'algoritmo esegue 6 confronti tra chiavi e 0 scambi, e l'array ordinato è

[5, 10, 15, 20].

### Esercizio 3.13 — Algoritmi di ordinamento (E13)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [14, 7, 21, 3, 18]$
2.  $a = [40, 30, 20, 10]$
3.  $a = [3, 6, 1, 9, 2]$
4.  $a = [25, 12, 50]$
5.  $a = [17, 17, 17, 17]$

#### Domande

1. Applicare l'algoritmo **ExchangeSort** a ciascun array, per ordinarlo in ordine non crescente. Mostrare l'esecuzione passo per passo: per ogni confronto, dire se viene eseguito uno scambio e indicare lo stato dell'array dopo il confronto.
2. Per ciascun array, contare i confronti tra chiavi e gli scambi eseguiti dall'algoritmo.

#### Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [14, 7, 21, 3, 18]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                        |
|-----|-----|--------|--------|--------|--------|--------|---------------------------------------------------------------|
| –   | –   | 14     | 7      | 21     | 3      | 18     | stato iniziale                                                |
| 1   | 2   | 7      | 14     | 21     | 3      | 18     | $a[1] = 14 > a[2] = 7$ : scambia $a[1] \leftrightarrow a[2]$  |
|     | 3   | 7      | 14     | 21     | 3      | 18     | $a[1] = 7 \leq a[3] = 21$ : nessuno scambio                   |
|     | 4   | 3      | 14     | 21     | 7      | 18     | $a[1] = 7 > a[4] = 3$ : scambia $a[1] \leftrightarrow a[4]$   |
|     | 5   | 3      | 14     | 21     | 7      | 18     | $a[1] = 3 \leq a[5] = 18$ : nessuno scambio                   |
| 2   | 3   | 3      | 14     | 21     | 7      | 18     | $a[2] = 14 \leq a[3] = 21$ : nessuno scambio                  |
|     | 4   | 3      | 7      | 21     | 14     | 18     | $a[2] = 14 > a[4] = 7$ : scambia $a[2] \leftrightarrow a[4]$  |
|     | 5   | 3      | 7      | 21     | 14     | 18     | $a[2] = 7 \leq a[5] = 18$ : nessuno scambio                   |
| 3   | 4   | 3      | 7      | 14     | 21     | 18     | $a[3] = 21 > a[4] = 14$ : scambia $a[3] \leftrightarrow a[4]$ |
|     | 5   | 3      | 7      | 14     | 21     | 18     | $a[3] = 14 \leq a[5] = 18$ : nessuno scambio                  |
| 4   | 5   | 3      | 7      | 14     | 18     | 21     | $a[4] = 21 > a[5] = 18$ : scambia $a[4] \leftrightarrow a[5]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                  |
|-----|--------|--------|--------|--------|--------|-----------------------------------------|
| –   | 14     | 7      | 21     | 3      | 18     | stato iniziale                          |
| 1   | 3      | 14     | 21     | 7      | 18     | $a[1] = \min(a[1..5]) = 3$ (2 scambi)   |
| 2   | 3      | 7      | 21     | 14     | 18     | $a[2] = \min(a[2..5]) = 7$ (1 scambio)  |
| 3   | 3      | 7      | 14     | 21     | 18     | $a[3] = \min(a[3..5]) = 14$ (1 scambio) |
| 4   | 3      | 7      | 14     | 18     | 21     | $a[4] = \min(a[4..5]) = 18$ (1 scambio) |

L'algoritmo esegue 10 confronti tra chiavi e 5 scambi, e l'array ordinato è

$[3, 7, 14, 18, 21]$ .

2. Array  $a = [40, 30, 20, 10]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                        |
|-----|-----|--------|--------|--------|--------|---------------------------------------------------------------|
| –   | –   | 40     | 30     | 20     | 10     | stato iniziale                                                |
| 1   | 2   | 30     | 40     | 20     | 10     | $a[1] = 40 > a[2] = 30$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 20     | 40     | 30     | 10     | $a[1] = 30 > a[3] = 20$ : scambia $a[1] \leftrightarrow a[3]$ |
|     | 4   | 10     | 40     | 30     | 20     | $a[1] = 20 > a[4] = 10$ : scambia $a[1] \leftrightarrow a[4]$ |
| 2   | 3   | 10     | 30     | 40     | 20     | $a[2] = 40 > a[3] = 30$ : scambia $a[2] \leftrightarrow a[3]$ |
|     | 4   | 10     | 20     | 40     | 30     | $a[2] = 30 > a[4] = 20$ : scambia $a[2] \leftrightarrow a[4]$ |
| 3   | 4   | 10     | 20     | 30     | 40     | $a[3] = 40 > a[4] = 30$ : scambia $a[3] \leftrightarrow a[4]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                  |
|-----|--------|--------|--------|--------|-----------------------------------------|
| –   | 40     | 30     | 20     | 10     | stato iniziale                          |
| 1   | 10     | 40     | 30     | 20     | $a[1] = \min(a[1..4]) = 10$ (3 scambi)  |
| 2   | 10     | 20     | 40     | 30     | $a[2] = \min(a[2..4]) = 20$ (2 scambi)  |
| 3   | 10     | 20     | 30     | 40     | $a[3] = \min(a[3..4]) = 30$ (1 scambio) |

L'algoritmo esegue 6 confronti tra chiavi e 6 scambi, e l'array ordinato è

[10, 20, 30, 40].

3. Array  $a = [3, 6, 1, 9, 2]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                      |
|-----|-----|--------|--------|--------|--------|--------|-------------------------------------------------------------|
| –   | –   | 3      | 6      | 1      | 9      | 2      | stato iniziale                                              |
| 1   | 2   | 3      | 6      | 1      | 9      | 2      | $a[1] = 3 \leq a[2] = 6$ : nessuno scambio                  |
|     | 3   | 1      | 6      | 3      | 9      | 2      | $a[1] = 3 > a[3] = 1$ : scambia $a[1] \leftrightarrow a[3]$ |
|     | 4   | 1      | 6      | 3      | 9      | 2      | $a[1] = 1 \leq a[4] = 9$ : nessuno scambio                  |
|     | 5   | 1      | 6      | 3      | 9      | 2      | $a[1] = 1 \leq a[5] = 2$ : nessuno scambio                  |
| 2   | 3   | 1      | 3      | 6      | 9      | 2      | $a[2] = 6 > a[3] = 3$ : scambia $a[2] \leftrightarrow a[3]$ |
|     | 4   | 1      | 3      | 6      | 9      | 2      | $a[2] = 3 \leq a[4] = 9$ : nessuno scambio                  |
|     | 5   | 1      | 2      | 6      | 9      | 3      | $a[2] = 3 > a[5] = 2$ : scambia $a[2] \leftrightarrow a[5]$ |
| 3   | 4   | 1      | 2      | 6      | 9      | 3      | $a[3] = 6 \leq a[4] = 9$ : nessuno scambio                  |
|     | 5   | 1      | 2      | 3      | 9      | 6      | $a[3] = 6 > a[5] = 3$ : scambia $a[3] \leftrightarrow a[5]$ |
| 4   | 5   | 1      | 2      | 3      | 6      | 9      | $a[4] = 9 > a[5] = 6$ : scambia $a[4] \leftrightarrow a[5]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                 |
|-----|--------|--------|--------|--------|--------|----------------------------------------|
| –   | 3      | 6      | 1      | 9      | 2      | stato iniziale                         |
| 1   | 1      | 6      | 3      | 9      | 2      | $a[1] = \min(a[1..5]) = 1$ (1 scambio) |
| 2   | 1      | 2      | 6      | 9      | 3      | $a[2] = \min(a[2..5]) = 2$ (2 scambi)  |
| 3   | 1      | 2      | 3      | 9      | 6      | $a[3] = \min(a[3..5]) = 3$ (1 scambio) |
| 4   | 1      | 2      | 3      | 6      | 9      | $a[4] = \min(a[4..5]) = 6$ (1 scambio) |

L'algoritmo esegue 10 confronti tra chiavi e 5 scambi, e l'array ordinato è

[1, 2, 3, 6, 9].

4. Array  $a = [25, 12, 50]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                        |
|-----|-----|--------|--------|--------|---------------------------------------------------------------|
| –   | –   | 25     | 12     | 50     | stato iniziale                                                |
| 1   | 2   | 12     | 25     | 50     | $a[1] = 25 > a[2] = 12$ : scambia $a[1] \leftrightarrow a[2]$ |
|     | 3   | 12     | 25     | 50     | $a[1] = 12 \leq a[3] = 50$ : nessuno scambio                  |
| 2   | 3   | 12     | 25     | 50     | $a[2] = 25 \leq a[3] = 50$ : nessuno scambio                  |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                  |
|-----|--------|--------|--------|-----------------------------------------|
| –   | 25     | 12     | 50     | stato iniziale                          |
| 1   | 12     | 25     | 50     | $a[1] = \min(a[1..3]) = 12$ (1 scambio) |
| 2   | 12     | 25     | 50     | $a[2] = \min(a[2..3]) = 25$ (0 scambi)  |

L'algoritmo esegue 3 confronti tra chiavi e 1 scambio, e l'array ordinato è

[12, 25, 50].

5. Array  $a = [17, 17, 17, 17]$  ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                       |
|-----|-----|--------|--------|--------|--------|----------------------------------------------|
| –   | –   | 17     | 17     | 17     | 17     | stato iniziale                               |
| 1   | 2   | 17     | 17     | 17     | 17     | $a[1] = 17 \leq a[2] = 17$ : nessuno scambio |
|     | 3   | 17     | 17     | 17     | 17     | $a[1] = 17 \leq a[3] = 17$ : nessuno scambio |
|     | 4   | 17     | 17     | 17     | 17     | $a[1] = 17 \leq a[4] = 17$ : nessuno scambio |
| 2   | 3   | 17     | 17     | 17     | 17     | $a[2] = 17 \leq a[3] = 17$ : nessuno scambio |
|     | 4   | 17     | 17     | 17     | 17     | $a[2] = 17 \leq a[4] = 17$ : nessuno scambio |
| 3   | 4   | 17     | 17     | 17     | 17     | $a[3] = 17 \leq a[4] = 17$ : nessuno scambio |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                 |
|-----|--------|--------|--------|--------|----------------------------------------|
| –   | 17     | 17     | 17     | 17     | stato iniziale                         |
| 1   | 17     | 17     | 17     | 17     | $a[1] = \min(a[1..4]) = 17$ (0 scambi) |
| 2   | 17     | 17     | 17     | 17     | $a[2] = \min(a[2..4]) = 17$ (0 scambi) |
| 3   | 17     | 17     | 17     | 17     | $a[3] = \min(a[3..4]) = 17$ (0 scambi) |

L'algoritmo esegue 6 confronti tra chiavi e 0 scambi, e l'array ordinato è

[17, 17, 17, 17].

### Esercizio 3.14 — Algoritmi di ordinamento (E14)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [5, 2, 4, 6, 1, 3]$
2.  $a = [8, 3, 7, 4, 9, 2]$
3.  $a = [6, 5, 4, 3, 2, 1]$
4.  $a = [1, 2, 3, 4, 5, 6]$

5.  $a = [10, 5, 15, 3]$

### Domande

1. Applicare l'algoritmo **SelectionSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni iterazione del ciclo esterno, indicare l'indice  $minIdx$  del minimo trovato dal ciclo interno e dire se viene eseguito uno scambio.
2. Per ciascun array, contare i confronti tra chiavi e gli scambi eseguiti dall'algoritmo.

### Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [5, 2, 4, 6, 1, 3]$  ( $n = 6$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 5      | 2      | 4      | 6      | 1      | 3      | stato iniziale                                            |
| 1   | —   | 1        | 5      | 2      | 4      | 6      | 1      | 3      | $minIdx := 1$ ( $a[1] = 5$ )                              |
|     | 2   | 2        | 5      | 2      | 4      | 6      | 1      | 3      | $a[2] = 2 < a[1] = 5$ : $minIdx \leftarrow 2$             |
|     | 3   | 2        | 5      | 2      | 4      | 6      | 1      | 3      | $a[3] = 4 \geq a[2] = 2$ : nessun aggiornamento           |
|     | 4   | 2        | 5      | 2      | 4      | 6      | 1      | 3      | $a[4] = 6 \geq a[2] = 2$ : nessun aggiornamento           |
|     | 5   | 5        | 5      | 2      | 4      | 6      | 1      | 3      | $a[5] = 1 < a[2] = 2$ : $minIdx \leftarrow 5$             |
|     | 6   | 5        | 5      | 2      | 4      | 6      | 1      | 3      | $a[6] = 3 \geq a[5] = 1$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 4      | 6      | 5      | 3      | $minIdx = 5 \neq 1$ : scambia $a[1] \leftrightarrow a[5]$ |
| 2   | —   | 2        | 1      | 2      | 4      | 6      | 5      | 3      | $minIdx := 2$ ( $a[2] = 2$ )                              |
|     | 3   | 2        | 1      | 2      | 4      | 6      | 5      | 3      | $a[3] = 4 \geq a[2] = 2$ : nessun aggiornamento           |
|     | 4   | 2        | 1      | 2      | 4      | 6      | 5      | 3      | $a[4] = 6 \geq a[2] = 2$ : nessun aggiornamento           |
|     | 5   | 2        | 1      | 2      | 4      | 6      | 5      | 3      | $a[5] = 5 \geq a[2] = 2$ : nessun aggiornamento           |
|     | 6   | 2        | 1      | 2      | 4      | 6      | 5      | 3      | $a[6] = 3 \geq a[2] = 2$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 4      | 6      | 5      | 3      | $minIdx = 2$ : nessuno scambio                            |
|     | —   | —        | 1      | 2      | 4      | 6      | 5      | 3      |                                                           |
| 3   | —   | 3        | 1      | 2      | 4      | 6      | 5      | 3      | $minIdx := 3$ ( $a[3] = 4$ )                              |
|     | 4   | 3        | 1      | 2      | 4      | 6      | 5      | 3      | $a[4] = 6 \geq a[3] = 4$ : nessun aggiornamento           |
|     | 5   | 3        | 1      | 2      | 4      | 6      | 5      | 3      | $a[5] = 5 \geq a[3] = 4$ : nessun aggiornamento           |
|     | 6   | 6        | 1      | 2      | 4      | 6      | 5      | 3      | $a[6] = 3 < a[3] = 4$ : $minIdx \leftarrow 6$             |
|     | —   | —        | 1      | 2      | 3      | 6      | 5      | 4      | $minIdx = 6 \neq 3$ : scambia $a[3] \leftrightarrow a[6]$ |
|     | —   | —        | 1      | 2      | 3      | 6      | 5      | 4      |                                                           |
|     | —   | —        | 1      | 2      | 3      | 6      | 5      | 4      |                                                           |
| 4   | —   | 4        | 1      | 2      | 3      | 6      | 5      | 4      | $minIdx := 4$ ( $a[4] = 6$ )                              |
|     | 5   | 5        | 1      | 2      | 3      | 6      | 5      | 4      | $a[5] = 5 < a[4] = 6$ : $minIdx \leftarrow 5$             |
|     | 6   | 6        | 1      | 2      | 3      | 6      | 5      | 4      | $a[6] = 4 < a[5] = 5$ : $minIdx \leftarrow 6$             |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx = 6 \neq 4$ : scambia $a[4] \leftrightarrow a[6]$ |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                           |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                           |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                           |
| 5   | —   | 5        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx := 5$ ( $a[5] = 5$ )                              |
|     | 6   | 5        | 1      | 2      | 3      | 4      | 5      | 6      | $a[6] = 6 \geq a[5] = 5$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx = 5$ : nessuno scambio                            |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 5      | 2      | 4      | 6      | 1      | 3      | stato iniziale                                                   |
| 1   | 5        | 1      | 2      | 4      | 6      | 5      | 3      | scambia $a[1] \leftrightarrow a[5]$ : $a[1] = \min(a[1..6]) = 1$ |
| 2   | 2        | 1      | 2      | 4      | 6      | 5      | 3      | nessuno scambio: $a[2] = \min(a[2..6]) = 2$                      |
| 3   | 6        | 1      | 2      | 3      | 6      | 5      | 4      | scambia $a[3] \leftrightarrow a[6]$ : $a[3] = \min(a[3..6]) = 3$ |
| 4   | 6        | 1      | 2      | 3      | 4      | 5      | 6      | scambia $a[4] \leftrightarrow a[6]$ : $a[4] = \min(a[4..6]) = 4$ |
| 5   | 5        | 1      | 2      | 3      | 4      | 5      | 6      | nessuno scambio: $a[5] = \min(a[5..6]) = 5$                      |

L'algoritmo esegue 15 confronti tra chiavi e 3 scambi, e l'array ordinato è

$$[1, 2, 3, 4, 5, 6].$$

2. Array  $a = [8, 3, 7, 4, 9, 2]$  ( $n = 6$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 8      | 3      | 7      | 4      | 9      | 2      | stato iniziale                                            |
| 1   | —   | 1        | 8      | 3      | 7      | 4      | 9      | 2      | $minIdx := 1$ ( $a[1] = 8$ )                              |
|     | 2   | 2        | 8      | 3      | 7      | 4      | 9      | 2      | $a[2] = 3 < a[1] = 8$ : $minIdx \leftarrow 2$             |
|     | 3   | 2        | 8      | 3      | 7      | 4      | 9      | 2      | $a[3] = 7 \geq a[2] = 3$ : nessun aggiornamento           |
|     | 4   | 2        | 8      | 3      | 7      | 4      | 9      | 2      | $a[4] = 4 \geq a[2] = 3$ : nessun aggiornamento           |
|     | 5   | 2        | 8      | 3      | 7      | 4      | 9      | 2      | $a[5] = 9 \geq a[2] = 3$ : nessun aggiornamento           |
|     | 6   | 6        | 8      | 3      | 7      | 4      | 9      | 2      | $a[6] = 2 < a[2] = 3$ : $minIdx \leftarrow 6$             |
|     | —   | —        | 2      | 3      | 7      | 4      | 9      | 8      | $minIdx = 6 \neq 1$ : scambia $a[1] \leftrightarrow a[6]$ |
| 2   | —   | 2        | 2      | 3      | 7      | 4      | 9      | 8      | $minIdx := 2$ ( $a[2] = 3$ )                              |
|     | 3   | 2        | 2      | 3      | 7      | 4      | 9      | 8      | $a[3] = 7 \geq a[2] = 3$ : nessun aggiornamento           |
|     | 4   | 2        | 2      | 3      | 7      | 4      | 9      | 8      | $a[4] = 4 \geq a[2] = 3$ : nessun aggiornamento           |
|     | 5   | 2        | 2      | 3      | 7      | 4      | 9      | 8      | $a[5] = 9 \geq a[2] = 3$ : nessun aggiornamento           |
|     | 6   | 2        | 2      | 3      | 7      | 4      | 9      | 8      | $a[6] = 8 \geq a[2] = 3$ : nessun aggiornamento           |
|     | —   | —        | 2      | 3      | 7      | 4      | 9      | 8      | $minIdx = 2$ : nessuno scambio                            |
|     | —   | —        | 2      | 3      | 7      | 4      | 9      | 8      |                                                           |
| 3   | —   | 3        | 2      | 3      | 7      | 4      | 9      | 8      | $minIdx := 3$ ( $a[3] = 7$ )                              |
|     | 4   | 4        | 2      | 3      | 7      | 4      | 9      | 8      | $a[4] = 4 < a[3] = 7$ : $minIdx \leftarrow 4$             |
|     | 5   | 4        | 2      | 3      | 7      | 4      | 9      | 8      | $a[5] = 9 \geq a[4] = 4$ : nessun aggiornamento           |
|     | 6   | 4        | 2      | 3      | 7      | 4      | 9      | 8      | $a[6] = 8 \geq a[4] = 4$ : nessun aggiornamento           |
|     | —   | —        | 2      | 3      | 4      | 7      | 9      | 8      | $minIdx = 4 \neq 3$ : scambia $a[3] \leftrightarrow a[4]$ |
|     | —   | —        | 2      | 3      | 4      | 7      | 9      | 8      |                                                           |
|     | —   | —        | 2      | 3      | 4      | 7      | 9      | 8      |                                                           |
| 4   | —   | 4        | 2      | 3      | 4      | 7      | 9      | 8      | $minIdx := 4$ ( $a[4] = 7$ )                              |
|     | 5   | 4        | 2      | 3      | 4      | 7      | 9      | 8      | $a[5] = 9 \geq a[4] = 7$ : nessun aggiornamento           |
|     | 6   | 4        | 2      | 3      | 4      | 7      | 9      | 8      | $a[6] = 8 \geq a[4] = 7$ : nessun aggiornamento           |
|     | —   | —        | 2      | 3      | 4      | 7      | 9      | 8      | $minIdx = 4$ : nessuno scambio                            |
|     | —   | —        | 2      | 3      | 4      | 7      | 9      | 8      |                                                           |
|     | —   | —        | 2      | 3      | 4      | 7      | 9      | 8      |                                                           |
|     | —   | —        | 2      | 3      | 4      | 7      | 9      | 8      |                                                           |
| 5   | —   | 5        | 2      | 3      | 4      | 7      | 9      | 8      | $minIdx := 5$ ( $a[5] = 9$ )                              |
|     | 6   | 6        | 2      | 3      | 4      | 7      | 9      | 8      | $a[6] = 8 < a[5] = 9$ : $minIdx \leftarrow 6$             |
|     | —   | —        | 2      | 3      | 4      | 7      | 8      | 9      | $minIdx = 6 \neq 5$ : scambia $a[5] \leftrightarrow a[6]$ |
|     | —   | —        | 2      | 3      | 4      | 7      | 8      | 9      |                                                           |
|     | —   | —        | 2      | 3      | 4      | 7      | 8      | 9      |                                                           |
|     | —   | —        | 2      | 3      | 4      | 7      | 8      | 9      |                                                           |
|     | —   | —        | 2      | 3      | 4      | 7      | 8      | 9      |                                                           |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 8      | 3      | 7      | 4      | 9      | 2      | stato iniziale                                                   |
| 1   | 6        | 2      | 3      | 7      | 4      | 9      | 8      | scambia $a[1] \leftrightarrow a[6]$ : $a[1] = \min(a[1..6]) = 2$ |
| 2   | 2        | 2      | 3      | 7      | 4      | 9      | 8      | nessuno scambio: $a[2] = \min(a[2..6]) = 3$                      |
| 3   | 4        | 2      | 3      | 4      | 7      | 9      | 8      | scambia $a[3] \leftrightarrow a[4]$ : $a[3] = \min(a[3..6]) = 4$ |
| 4   | 4        | 2      | 3      | 4      | 7      | 9      | 8      | nessuno scambio: $a[4] = \min(a[4..6]) = 7$                      |
| 5   | 6        | 2      | 3      | 4      | 7      | 8      | 9      | scambia $a[5] \leftrightarrow a[6]$ : $a[5] = \min(a[5..6]) = 8$ |

L'algoritmo esegue 15 confronti tra chiavi e 3 scambi, e l'array ordinato è

$$[2, 3, 4, 7, 8, 9].$$

3. Array  $a = [6, 5, 4, 3, 2, 1]$  ( $n = 6$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 6      | 5      | 4      | 3      | 2      | 1      | stato iniziale                                            |
| 1   | —   | 1        | 6      | 5      | 4      | 3      | 2      | 1      | $minIdx := 1$ ( $a[1] = 6$ )                              |
|     | 2   | 2        | 6      | 5      | 4      | 3      | 2      | 1      | $a[2] = 5 < a[1] = 6$ : $minIdx \leftarrow 2$             |
|     | 3   | 3        | 6      | 5      | 4      | 3      | 2      | 1      | $a[3] = 4 < a[2] = 5$ : $minIdx \leftarrow 3$             |
|     | 4   | 4        | 6      | 5      | 4      | 3      | 2      | 1      | $a[4] = 3 < a[3] = 4$ : $minIdx \leftarrow 4$             |
|     | 5   | 5        | 6      | 5      | 4      | 3      | 2      | 1      | $a[5] = 2 < a[4] = 3$ : $minIdx \leftarrow 5$             |
|     | 6   | 6        | 6      | 5      | 4      | 3      | 2      | 1      | $a[6] = 1 < a[5] = 2$ : $minIdx \leftarrow 6$             |
|     | —   | —        | 1      | 5      | 4      | 3      | 2      | 6      | $minIdx = 6 \neq 1$ : scambia $a[1] \leftrightarrow a[6]$ |
| 2   | —   | 2        | 1      | 5      | 4      | 3      | 2      | 6      | $minIdx := 2$ ( $a[2] = 5$ )                              |
|     | 3   | 3        | 1      | 5      | 4      | 3      | 2      | 6      | $a[3] = 4 < a[2] = 5$ : $minIdx \leftarrow 3$             |
|     | 4   | 4        | 1      | 5      | 4      | 3      | 2      | 6      | $a[4] = 3 < a[3] = 4$ : $minIdx \leftarrow 4$             |
|     | 5   | 5        | 1      | 5      | 4      | 3      | 2      | 6      | $a[5] = 2 < a[4] = 3$ : $minIdx \leftarrow 5$             |
|     | 6   | 5        | 1      | 5      | 4      | 3      | 2      | 6      | $a[6] = 6 \geq a[5] = 2$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 4      | 3      | 5      | 6      | $minIdx = 5 \neq 2$ : scambia $a[2] \leftrightarrow a[5]$ |
|     | —   | —        | 1      | 2      | 4      | 3      | 5      | 6      |                                                           |
| 3   | —   | 3        | 1      | 2      | 4      | 3      | 5      | 6      | $minIdx := 3$ ( $a[3] = 4$ )                              |
|     | 4   | 4        | 1      | 2      | 4      | 3      | 5      | 6      | $a[4] = 3 < a[3] = 4$ : $minIdx \leftarrow 4$             |
|     | 5   | 4        | 1      | 2      | 4      | 3      | 5      | 6      | $a[5] = 5 \geq a[4] = 3$ : nessun aggiornamento           |
|     | 6   | 4        | 1      | 2      | 4      | 3      | 5      | 6      | $a[6] = 6 \geq a[4] = 3$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx = 4 \neq 3$ : scambia $a[3] \leftrightarrow a[4]$ |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                           |
| 4   | —   | 4        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx := 4$ ( $a[4] = 4$ )                              |
|     | 5   | 4        | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = 5 \geq a[4] = 4$ : nessun aggiornamento           |
|     | 6   | 4        | 1      | 2      | 3      | 4      | 5      | 6      | $a[6] = 6 \geq a[4] = 4$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx = 4$ : nessuno scambio                            |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                           |
| 5   | —   | 5        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx := 5$ ( $a[5] = 5$ )                              |
|     | 6   | 5        | 1      | 2      | 3      | 4      | 5      | 6      | $a[6] = 6 \geq a[5] = 5$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx = 5$ : nessuno scambio                            |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 6      | 5      | 4      | 3      | 2      | 1      | stato iniziale                                                   |
| 1   | 6        | 1      | 5      | 4      | 3      | 2      | 6      | scambia $a[1] \leftrightarrow a[6]$ : $a[1] = \min(a[1..6]) = 1$ |
| 2   | 5        | 1      | 2      | 4      | 3      | 5      | 6      | scambia $a[2] \leftrightarrow a[5]$ : $a[2] = \min(a[2..6]) = 2$ |
| 3   | 4        | 1      | 2      | 3      | 4      | 5      | 6      | scambia $a[3] \leftrightarrow a[4]$ : $a[3] = \min(a[3..6]) = 3$ |
| 4   | 4        | 1      | 2      | 3      | 4      | 5      | 6      | nessuno scambio: $a[4] = \min(a[4..6]) = 4$                      |
| 5   | 5        | 1      | 2      | 3      | 4      | 5      | 6      | nessuno scambio: $a[5] = \min(a[5..6]) = 5$                      |

L'algoritmo esegue 15 confronti tra chiavi e 3 scambi, e l'array ordinato è

$$[1, 2, 3, 4, 5, 6].$$

4. Array  $a = [1, 2, 3, 4, 5, 6]$  ( $n = 6$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                          |
|-----|-----|----------|--------|--------|--------|--------|--------|--------|-------------------------------------------------|
| —   | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | stato iniziale                                  |
| 1   | —   | 1        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx := 1$ ( $a[1] = 1$ )                    |
|     | 2   | 1        | 1      | 2      | 3      | 4      | 5      | 6      | $a[2] = 2 \geq a[1] = 1$ : nessun aggiornamento |
|     | 3   | 1        | 1      | 2      | 3      | 4      | 5      | 6      | $a[3] = 3 \geq a[1] = 1$ : nessun aggiornamento |
|     | 4   | 1        | 1      | 2      | 3      | 4      | 5      | 6      | $a[4] = 4 \geq a[1] = 1$ : nessun aggiornamento |
|     | 5   | 1        | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = 5 \geq a[1] = 1$ : nessun aggiornamento |
|     | 6   | 1        | 1      | 2      | 3      | 4      | 5      | 6      | $a[6] = 6 \geq a[1] = 1$ : nessun aggiornamento |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx = 1$ : nessuno scambio                  |
| 2   | —   | 2        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx := 2$ ( $a[2] = 2$ )                    |
|     | 3   | 2        | 1      | 2      | 3      | 4      | 5      | 6      | $a[3] = 3 \geq a[2] = 2$ : nessun aggiornamento |
|     | 4   | 2        | 1      | 2      | 3      | 4      | 5      | 6      | $a[4] = 4 \geq a[2] = 2$ : nessun aggiornamento |
|     | 5   | 2        | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = 5 \geq a[2] = 2$ : nessun aggiornamento |
|     | 6   | 2        | 1      | 2      | 3      | 4      | 5      | 6      | $a[6] = 6 \geq a[2] = 2$ : nessun aggiornamento |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx = 2$ : nessuno scambio                  |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                 |
| 3   | —   | 3        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx := 3$ ( $a[3] = 3$ )                    |
|     | 4   | 3        | 1      | 2      | 3      | 4      | 5      | 6      | $a[4] = 4 \geq a[3] = 3$ : nessun aggiornamento |
|     | 5   | 3        | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = 5 \geq a[3] = 3$ : nessun aggiornamento |
|     | 6   | 3        | 1      | 2      | 3      | 4      | 5      | 6      | $a[6] = 6 \geq a[3] = 3$ : nessun aggiornamento |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx = 3$ : nessuno scambio                  |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                 |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                 |
| 4   | —   | 4        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx := 4$ ( $a[4] = 4$ )                    |
|     | 5   | 4        | 1      | 2      | 3      | 4      | 5      | 6      | $a[5] = 5 \geq a[4] = 4$ : nessun aggiornamento |
|     | 6   | 4        | 1      | 2      | 3      | 4      | 5      | 6      | $a[6] = 6 \geq a[4] = 4$ : nessun aggiornamento |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx = 4$ : nessuno scambio                  |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                 |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                 |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                 |
| 5   | —   | 5        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx := 5$ ( $a[5] = 5$ )                    |
|     | 6   | 5        | 1      | 2      | 3      | 4      | 5      | 6      | $a[6] = 6 \geq a[5] = 5$ : nessun aggiornamento |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | $minIdx = 5$ : nessuno scambio                  |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                 |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                 |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                 |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | 6      |                                                 |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | Azione                                      |
|-----|----------|--------|--------|--------|--------|--------|--------|---------------------------------------------|
| —   | —        | 1      | 2      | 3      | 4      | 5      | 6      | stato iniziale                              |
| 1   | 1        | 1      | 2      | 3      | 4      | 5      | 6      | nessuno scambio: $a[1] = \min(a[1..6]) = 1$ |
| 2   | 2        | 1      | 2      | 3      | 4      | 5      | 6      | nessuno scambio: $a[2] = \min(a[2..6]) = 2$ |
| 3   | 3        | 1      | 2      | 3      | 4      | 5      | 6      | nessuno scambio: $a[3] = \min(a[3..6]) = 3$ |
| 4   | 4        | 1      | 2      | 3      | 4      | 5      | 6      | nessuno scambio: $a[4] = \min(a[4..6]) = 4$ |
| 5   | 5        | 1      | 2      | 3      | 4      | 5      | 6      | nessuno scambio: $a[5] = \min(a[5..6]) = 5$ |

L'algoritmo esegue 15 confronti tra chiavi e 0 scambi, e l'array ordinato è

$$[1, 2, 3, 4, 5, 6].$$

5. Array  $a = [10, 5, 15, 3]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 10     | 5      | 15     | 3      | stato iniziale                                            |
| 1   | —   | 1        | 10     | 5      | 15     | 3      | $minIdx := 1$ ( $a[1] = 10$ )                             |
|     | 2   | 2        | 10     | 5      | 15     | 3      | $a[2] = 5 < a[1] = 10$ : $minIdx \leftarrow 2$            |
|     | 3   | 2        | 10     | 5      | 15     | 3      | $a[3] = 15 \geq a[2] = 5$ : nessun aggiornamento          |
|     | 4   | 4        | 10     | 5      | 15     | 3      | $a[4] = 3 < a[2] = 5$ : $minIdx \leftarrow 4$             |
|     | —   | —        | 3      | 5      | 15     | 10     | $minIdx = 4 \neq 1$ : scambia $a[1] \leftrightarrow a[4]$ |
| 2   | —   | 2        | 3      | 5      | 15     | 10     | $minIdx := 2$ ( $a[2] = 5$ )                              |
|     | 3   | 2        | 3      | 5      | 15     | 10     | $a[3] = 15 \geq a[2] = 5$ : nessun aggiornamento          |
|     | 4   | 2        | 3      | 5      | 15     | 10     | $a[4] = 10 \geq a[2] = 5$ : nessun aggiornamento          |
|     | —   | —        | 3      | 5      | 15     | 10     | $minIdx = 2$ : nessuno scambio                            |
| 3   | —   | 3        | 3      | 5      | 15     | 10     | $minIdx := 3$ ( $a[3] = 15$ )                             |
|     | 4   | 4        | 3      | 5      | 15     | 10     | $a[4] = 10 < a[3] = 15$ : $minIdx \leftarrow 4$           |
|     | —   | —        | 3      | 5      | 10     | 15     | $minIdx = 4 \neq 3$ : scambia $a[3] \leftrightarrow a[4]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                            |
|-----|----------|--------|--------|--------|--------|-------------------------------------------------------------------|
| —   | —        | 10     | 5      | 15     | 3      | stato iniziale                                                    |
| 1   | 4        | 3      | 5      | 15     | 10     | scambia $a[1] \leftrightarrow a[4]$ : $a[1] = \min(a[1..4]) = 3$  |
| 2   | 2        | 3      | 5      | 15     | 10     | nessuno scambio: $a[2] = \min(a[2..4]) = 5$                       |
| 3   | 4        | 3      | 5      | 10     | 15     | scambia $a[3] \leftrightarrow a[4]$ : $a[3] = \min(a[3..4]) = 10$ |

L'algoritmo esegue 6 confronti tra chiavi e 2 scambi, e l'array ordinato è

$$[3, 5, 10, 15].$$

**Esercizio 3.15 — Algoritmi di ordinamento (E15)**

Consideriamo i seguenti array di numeri naturali:

1.  $a = [7, 2, 9, 4]$
2.  $a = [15, 10, 5]$
3.  $a = [3, 1, 4, 1, 5, 9, 2, 6]$
4.  $a = [20, 18, 16, 14]$
5.  $a = [11, 11, 11]$

**Domande**

1. Applicare l'algoritmo **SelectionSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni iterazione del ciclo esterno, indicare l'indice  $minIdx$  del minimo trovato dal ciclo interno e dire se viene eseguito uno scambio.
2. Per ciascun array, contare i confronti tra chiavi e gli scambi eseguiti dall'algoritmo.

**Soluzione**

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [7, 2, 9, 4]$  ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 7      | 2      | 9      | 4      | stato iniziale                                            |
| 1   | —   | 1        | 7      | 2      | 9      | 4      | $minIdx := 1$ ( $a[1] = 7$ )                              |
|     | 2   | 2        | 7      | 2      | 9      | 4      | $a[2] = 2 < a[1] = 7$ : $minIdx \leftarrow 2$             |
|     | 3   | 2        | 7      | 2      | 9      | 4      | $a[3] = 9 \geq a[2] = 2$ : nessun aggiornamento           |
|     | 4   | 2        | 7      | 2      | 9      | 4      | $a[4] = 4 \geq a[2] = 2$ : nessun aggiornamento           |
|     | —   | —        | 2      | 7      | 9      | 4      | $minIdx = 2 \neq 1$ : scambia $a[1] \leftrightarrow a[2]$ |
| 2   | —   | 2        | 2      | 7      | 9      | 4      | $minIdx := 2$ ( $a[2] = 7$ )                              |
|     | 3   | 2        | 2      | 7      | 9      | 4      | $a[3] = 9 \geq a[2] = 7$ : nessun aggiornamento           |
|     | 4   | 4        | 2      | 7      | 9      | 4      | $a[4] = 4 < a[2] = 7$ : $minIdx \leftarrow 4$             |
|     | —   | —        | 2      | 4      | 9      | 7      | $minIdx = 4 \neq 2$ : scambia $a[2] \leftrightarrow a[4]$ |
| 3   | —   | 3        | 2      | 4      | 9      | 7      | $minIdx := 3$ ( $a[3] = 9$ )                              |
|     | 4   | 4        | 2      | 4      | 9      | 7      | $a[4] = 7 < a[3] = 9$ : $minIdx \leftarrow 4$             |
|     | —   | —        | 2      | 4      | 7      | 9      | $minIdx = 4 \neq 3$ : scambia $a[3] \leftrightarrow a[4]$ |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 7      | 2      | 9      | 4      | stato iniziale                                                   |
| 1   | 2        | 2      | 7      | 9      | 4      | scambia $a[1] \leftrightarrow a[2]$ : $a[1] = \min(a[1..4]) = 2$ |
| 2   | 4        | 2      | 4      | 9      | 7      | scambia $a[2] \leftrightarrow a[4]$ : $a[2] = \min(a[2..4]) = 4$ |
| 3   | 4        | 2      | 4      | 7      | 9      | scambia $a[3] \leftrightarrow a[4]$ : $a[3] = \min(a[3..4]) = 7$ |

L'algoritmo esegue 6 confronti tra chiavi e 3 scambi, e l'array ordinato è

$[2, 4, 7, 9]$ .

2. Array  $a = [15, 10, 5]$  ( $n = 3$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 15     | 10     | 5      | stato iniziale                                            |
| 1   | —   | 1        | 15     | 10     | 5      | $minIdx := 1$ ( $a[1] = 15$ )                             |
|     | 2   | 2        | 15     | 10     | 5      | $a[2] = 10 < a[1] = 15$ : $minIdx \leftarrow 2$           |
|     | 3   | 3        | 15     | 10     | 5      | $a[3] = 5 < a[2] = 10$ : $minIdx \leftarrow 3$            |
|     | —   | —        | 5      | 10     | 15     | $minIdx = 3 \neq 1$ : scambia $a[1] \leftrightarrow a[3]$ |
| 2   | —   | 2        | 5      | 10     | 15     | $minIdx := 2$ ( $a[2] = 10$ )                             |
|     | 3   | 2        | 5      | 10     | 15     | $a[3] = 15 \geq a[2] = 10$ : nessun aggiornamento         |
|     | —   | —        | 5      | 10     | 15     | $minIdx = 2$ : nessuno scambio                            |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                           |
|-----|----------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 15     | 10     | 5      | stato iniziale                                                   |
| 1   | 3        | 5      | 10     | 15     | scambia $a[1] \leftrightarrow a[3]$ : $a[1] = \min(a[1..3]) = 5$ |
| 2   | 2        | 5      | 10     | 15     | nessuno scambio: $a[2] = \min(a[2..3]) = 10$                     |

L'algoritmo esegue 3 confronti tra chiavi e 1 scambio, e l'array ordinato è

$[5, 10, 15]$ .

3. Array  $a = [3, 1, 4, 1, 5, 9, 2, 6]$  ( $n = 8$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | $a[7]$ | $a[8]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|--------|--------|--------|--------|-----------------------------------------------------------|
| –   | –   | –        | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | stato iniziale                                            |
| 1   | –   | 1        | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | $minIdx := 1$ ( $a[1] = 3$ )                              |
| 2   | 2   | 2        | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | $a[2] = 1 < a[1] = 3$ : $minIdx \leftarrow 2$             |
| 3   | 2   | 2        | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | $a[3] = 4 \geq a[2] = 1$ : nessun aggiornamento           |
| 4   | 2   | 2        | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | $a[4] = 1 \geq a[2] = 1$ : nessun aggiornamento           |
| 5   | 2   | 2        | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | $a[5] = 5 \geq a[2] = 1$ : nessun aggiornamento           |
| 6   | 2   | 2        | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | $a[6] = 9 \geq a[2] = 1$ : nessun aggiornamento           |
| 7   | 2   | 2        | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | $a[7] = 2 \geq a[2] = 1$ : nessun aggiornamento           |
| 8   | 2   | 2        | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | $a[8] = 6 \geq a[2] = 1$ : nessun aggiornamento           |
| –   | –   | –        | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $minIdx = 2 \neq 1$ : scambia $a[1] \leftrightarrow a[2]$ |
| 2   | –   | 2        | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $minIdx := 2$ ( $a[2] = 3$ )                              |
| 3   | 2   | 2        | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[3] = 4 \geq a[2] = 3$ : nessun aggiornamento           |
| 4   | 4   | 4        | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[4] = 1 < a[2] = 3$ : $minIdx \leftarrow 4$             |
| 5   | 4   | 4        | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[5] = 5 \geq a[4] = 1$ : nessun aggiornamento           |
| 6   | 4   | 4        | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[6] = 9 \geq a[4] = 1$ : nessun aggiornamento           |
| 7   | 4   | 4        | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[7] = 2 \geq a[4] = 1$ : nessun aggiornamento           |
| 8   | 4   | 4        | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | $a[8] = 6 \geq a[4] = 1$ : nessun aggiornamento           |
| –   | –   | –        | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $minIdx = 4 \neq 2$ : scambia $a[2] \leftrightarrow a[4]$ |
| 3   | –   | 3        | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $minIdx := 3$ ( $a[3] = 4$ )                              |
| 4   | 4   | 4        | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[4] = 3 < a[3] = 4$ : $minIdx \leftarrow 4$             |
| 5   | 4   | 4        | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[5] = 5 \geq a[4] = 3$ : nessun aggiornamento           |
| 6   | 4   | 4        | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[6] = 9 \geq a[4] = 3$ : nessun aggiornamento           |
| 7   | 7   | 7        | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[7] = 2 < a[4] = 3$ : $minIdx \leftarrow 7$             |
| 8   | 7   | 7        | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | $a[8] = 6 \geq a[7] = 2$ : nessun aggiornamento           |
| –   | –   | –        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $minIdx = 7 \neq 3$ : scambia $a[3] \leftrightarrow a[7]$ |
| 4   | –   | 4        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $minIdx := 4$ ( $a[4] = 3$ )                              |
| 5   | 4   | 4        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[5] = 5 \geq a[4] = 3$ : nessun aggiornamento           |
| 6   | 4   | 4        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[6] = 9 \geq a[4] = 3$ : nessun aggiornamento           |
| 7   | 4   | 4        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[7] = 4 \geq a[4] = 3$ : nessun aggiornamento           |
| 8   | 4   | 4        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[8] = 6 \geq a[4] = 3$ : nessun aggiornamento           |
| –   | –   | –        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $minIdx = 4$ : nessuno scambio                            |
| 5   | –   | 5        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $minIdx := 5$ ( $a[5] = 5$ )                              |
| 6   | 5   | 5        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[6] = 9 \geq a[5] = 5$ : nessun aggiornamento           |
| 7   | 7   | 7        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[7] = 4 < a[5] = 5$ : $minIdx \leftarrow 7$             |
| 8   | 7   | 7        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | $a[8] = 6 \geq a[7] = 4$ : nessun aggiornamento           |
| –   | –   | –        | 1      | 1      | 2      | 3      | 4      | 9      | 5      | 6      | $minIdx = 7 \neq 5$ : scambia $a[5] \leftrightarrow a[7]$ |
| 6   | –   | 6        | 1      | 1      | 2      | 3      | 4      | 9      | 5      | 6      | $minIdx := 6$ ( $a[6] = 9$ )                              |
| 7   | 7   | 7        | 1      | 1      | 2      | 3      | 4      | 9      | 5      | 6      | $a[7] = 5 < a[6] = 9$ : $minIdx \leftarrow 7$             |
| 8   | 7   | 7        | 1      | 1      | 2      | 3      | 4      | 9      | 5      | 6      | $a[8] = 6 \geq a[7] = 5$ : nessun aggiornamento           |
| –   | –   | –        | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 6      | $minIdx = 7 \neq 6$ : scambia $a[6] \leftrightarrow a[7]$ |
| 7   | –   | 7        | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 6      | $minIdx := 7$ ( $a[7] = 9$ )                              |
| 8   | 8   | 8        | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 6      | $a[8] = 6 < a[7] = 9$ : $minIdx \leftarrow 8$             |
| –   | –   | –        | 1      | 1      | 2      | 3      | 4      | 5      | 6      | 9      | $minIdx = 8 \neq 7$ : scambia $a[7] \leftrightarrow a[8]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | $a[6]$ | $a[7]$ | $a[8]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 3      | 1      | 4      | 1      | 5      | 9      | 2      | 6      | stato iniziale                                                   |
| 1   | 2        | 1      | 3      | 4      | 1      | 5      | 9      | 2      | 6      | scambia $a[1] \leftrightarrow a[2]$ : $a[1] = \min(a[1..8]) = 1$ |
| 2   | 4        | 1      | 1      | 4      | 3      | 5      | 9      | 2      | 6      | scambia $a[2] \leftrightarrow a[4]$ : $a[2] = \min(a[2..8]) = 1$ |
| 3   | 7        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | scambia $a[3] \leftrightarrow a[7]$ : $a[3] = \min(a[3..8]) = 2$ |
| 4   | 4        | 1      | 1      | 2      | 3      | 5      | 9      | 4      | 6      | nessuno scambio: $a[4] = \min(a[4..8]) = 3$                      |
| 5   | 7        | 1      | 1      | 2      | 3      | 4      | 9      | 5      | 6      | scambia $a[5] \leftrightarrow a[7]$ : $a[5] = \min(a[5..8]) = 4$ |
| 6   | 7        | 1      | 1      | 2      | 3      | 4      | 5      | 9      | 6      | scambia $a[6] \leftrightarrow a[7]$ : $a[6] = \min(a[6..8]) = 5$ |
| 7   | 8        | 1      | 1      | 2      | 3      | 4      | 5      | 6      | 9      | scambia $a[7] \leftrightarrow a[8]$ : $a[7] = \min(a[7..8]) = 6$ |

L'algoritmo esegue 28 confronti tra chiavi e 6 scambi, e l'array ordinato è

$[1, 1, 2, 3, 4, 5, 6, 9]$ .

#### 4. Array $a = [20, 18, 16, 14]$ ( $n = 4$ ).

*Traccia dettagliata:*

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 20     | 18     | 16     | 14     | stato iniziale                                            |
| 1   | —   | 1        | 20     | 18     | 16     | 14     | $minIdx := 1$ ( $a[1] = 20$ )                             |
|     | 2   | 2        | 20     | 18     | 16     | 14     | $a[2] = 18 < a[1] = 20$ : $minIdx \leftarrow 2$           |
|     | 3   | 3        | 20     | 18     | 16     | 14     | $a[3] = 16 < a[2] = 18$ : $minIdx \leftarrow 3$           |
|     | 4   | 4        | 20     | 18     | 16     | 14     | $a[4] = 14 < a[3] = 16$ : $minIdx \leftarrow 4$           |
|     | —   | —        | 14     | 18     | 16     | 20     | $minIdx = 4 \neq 1$ : scambia $a[1] \leftrightarrow a[4]$ |
| 2   | —   | 2        | 14     | 18     | 16     | 20     | $minIdx := 2$ ( $a[2] = 18$ )                             |
|     | 3   | 3        | 14     | 18     | 16     | 20     | $a[3] = 16 < a[2] = 18$ : $minIdx \leftarrow 3$           |
|     | 4   | 3        | 14     | 18     | 16     | 20     | $a[4] = 20 \geq a[3] = 16$ : nessun aggiornamento         |
|     | —   | —        | 14     | 16     | 18     | 20     | $minIdx = 3 \neq 2$ : scambia $a[2] \leftrightarrow a[3]$ |
| 3   | —   | 3        | 14     | 16     | 18     | 20     | $minIdx := 3$ ( $a[3] = 18$ )                             |
|     | 4   | 3        | 14     | 16     | 18     | 20     | $a[4] = 20 \geq a[3] = 18$ : nessun aggiornamento         |
|     | —   | —        | 14     | 16     | 18     | 20     | $minIdx = 3$ : nessuno scambio                            |

*Array dopo ogni iterazione del ciclo esterno:*

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                            |
|-----|----------|--------|--------|--------|--------|-------------------------------------------------------------------|
| —   | —        | 20     | 18     | 16     | 14     | stato iniziale                                                    |
| 1   | 4        | 14     | 18     | 16     | 20     | scambia $a[1] \leftrightarrow a[4]$ : $a[1] = \min(a[1..4]) = 14$ |
| 2   | 3        | 14     | 16     | 18     | 20     | scambia $a[2] \leftrightarrow a[3]$ : $a[2] = \min(a[2..4]) = 16$ |
| 3   | 3        | 14     | 16     | 18     | 20     | nessuno scambio: $a[3] = \min(a[3..4]) = 18$                      |

L'algoritmo esegue 6 confronti tra chiavi e 2 scambi, e l'array ordinato è

$[14, 16, 18, 20]$ .

5. Array  $a = [11, 11, 11]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                            |
|-----|-----|----------|--------|--------|--------|---------------------------------------------------|
| –   | –   | –        | 11     | 11     | 11     | stato iniziale                                    |
| 1   | –   | 1        | 11     | 11     | 11     | $minIdx := 1$ ( $a[1] = 11$ )                     |
|     | 2   | 1        | 11     | 11     | 11     | $a[2] = 11 \geq a[1] = 11$ : nessun aggiornamento |
|     | 3   | 1        | 11     | 11     | 11     | $a[3] = 11 \geq a[1] = 11$ : nessun aggiornamento |
|     | –   | –        | 11     | 11     | 11     | $minIdx = 1$ : nessuno scambio                    |
| 2   | –   | 2        | 11     | 11     | 11     | $minIdx := 2$ ( $a[2] = 11$ )                     |
|     | 3   | 2        | 11     | 11     | 11     | $a[3] = 11 \geq a[2] = 11$ : nessun aggiornamento |
|     | –   | –        | 11     | 11     | 11     | $minIdx = 2$ : nessuno scambio                    |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                       |
|-----|----------|--------|--------|--------|----------------------------------------------|
| –   | –        | 11     | 11     | 11     | stato iniziale                               |
| 1   | 1        | 11     | 11     | 11     | nessuno scambio: $a[1] = \min(a[1..3]) = 11$ |
| 2   | 2        | 11     | 11     | 11     | nessuno scambio: $a[2] = \min(a[2..3]) = 11$ |

L'algoritmo esegue 3 confronti tra chiavi e 0 scambi, e l'array ordinato è

$[11, 11, 11]$ .

### Esercizio 3.16 — Algoritmi di ordinamento (E16)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [9, 3, 7, 2]$
2.  $a = [12, 4, 18, 6, 10]$
3.  $a = [5, 4, 3, 2, 1]$
4.  $a = [100, 50, 25]$
5.  $a = [8]$

#### Domande

1. Applicare l'algoritmo **SelectionSort** a ciascun array, per ordinarlo in ordine non crescente. Mostrare l'esecuzione passo per passo: per ogni iterazione del ciclo esterno, indicare l'indice  $minIdx$  del minimo trovato dal ciclo interno e dire se viene eseguito uno scambio.
2. Per ciascun array, contare i confronti tra chiavi e gli scambi eseguiti dall'algoritmo.

## Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [9, 3, 7, 2]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 9      | 3      | 7      | 2      | stato iniziale                                            |
| 1   | —   | 1        | 9      | 3      | 7      | 2      | $minIdx := 1$ ( $a[1] = 9$ )                              |
|     | 2   | 2        | 9      | 3      | 7      | 2      | $a[2] = 3 < a[1] = 9$ : $minIdx \leftarrow 2$             |
|     | 3   | 2        | 9      | 3      | 7      | 2      | $a[3] = 7 \geq a[2] = 3$ : nessun aggiornamento           |
|     | 4   | 4        | 9      | 3      | 7      | 2      | $a[4] = 2 < a[2] = 3$ : $minIdx \leftarrow 4$             |
|     | —   | —        | 2      | 3      | 7      | 9      | $minIdx = 4 \neq 1$ : scambia $a[1] \leftrightarrow a[4]$ |
| 2   | —   | 2        | 2      | 3      | 7      | 9      | $minIdx := 2$ ( $a[2] = 3$ )                              |
|     | 3   | 2        | 2      | 3      | 7      | 9      | $a[3] = 7 \geq a[2] = 3$ : nessun aggiornamento           |
|     | 4   | 2        | 2      | 3      | 7      | 9      | $a[4] = 9 \geq a[2] = 3$ : nessun aggiornamento           |
|     | —   | —        | 2      | 3      | 7      | 9      | $minIdx = 2$ : nessuno scambio                            |
| 3   | —   | 3        | 2      | 3      | 7      | 9      | $minIdx := 3$ ( $a[3] = 7$ )                              |
|     | 4   | 3        | 2      | 3      | 7      | 9      | $a[4] = 9 \geq a[3] = 7$ : nessun aggiornamento           |
|     | —   | —        | 2      | 3      | 7      | 9      | $minIdx = 3$ : nessuno scambio                            |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 9      | 3      | 7      | 2      | stato iniziale                                                   |
| 1   | 4        | 2      | 3      | 7      | 9      | scambia $a[1] \leftrightarrow a[4]$ : $a[1] = \min(a[1..4]) = 2$ |
| 2   | 2        | 2      | 3      | 7      | 9      | nessuno scambio: $a[2] = \min(a[2..4]) = 3$                      |
| 3   | 3        | 2      | 3      | 7      | 9      | nessuno scambio: $a[3] = \min(a[3..4]) = 7$                      |

L'algoritmo esegue 6 confronti tra chiavi e 1 scambio, e l'array ordinato è

$[2, 3, 7, 9]$ .

2. Array  $a = [12, 4, 18, 6, 10]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|--------|-----------------------------------------------------------|
| –   | –   | –        | 12     | 4      | 18     | 6      | 10     | stato iniziale                                            |
| 1   | –   | 1        | 12     | 4      | 18     | 6      | 10     | $minIdx := 1$ ( $a[1] = 12$ )                             |
|     | 2   | 2        | 12     | 4      | 18     | 6      | 10     | $a[2] = 4 < a[1] = 12$ : $minIdx \leftarrow 2$            |
|     | 3   | 2        | 12     | 4      | 18     | 6      | 10     | $a[3] = 18 \geq a[2] = 4$ : nessun aggiornamento          |
|     | 4   | 2        | 12     | 4      | 18     | 6      | 10     | $a[4] = 6 \geq a[2] = 4$ : nessun aggiornamento           |
|     | 5   | 2        | 12     | 4      | 18     | 6      | 10     | $a[5] = 10 \geq a[2] = 4$ : nessun aggiornamento          |
|     | –   | –        | 4      | 12     | 18     | 6      | 10     | $minIdx = 2 \neq 1$ : scambia $a[1] \leftrightarrow a[2]$ |
| 2   | –   | 2        | 4      | 12     | 18     | 6      | 10     | $minIdx := 2$ ( $a[2] = 12$ )                             |
|     | 3   | 2        | 4      | 12     | 18     | 6      | 10     | $a[3] = 18 \geq a[2] = 12$ : nessun aggiornamento         |
|     | 4   | 4        | 4      | 12     | 18     | 6      | 10     | $a[4] = 6 < a[2] = 12$ : $minIdx \leftarrow 4$            |
|     | 5   | 4        | 4      | 12     | 18     | 6      | 10     | $a[5] = 10 \geq a[4] = 6$ : nessun aggiornamento          |
|     | –   | –        | 4      | 6      | 18     | 12     | 10     | $minIdx = 4 \neq 2$ : scambia $a[2] \leftrightarrow a[4]$ |
| 3   | –   | 3        | 4      | 6      | 18     | 12     | 10     | $minIdx := 3$ ( $a[3] = 18$ )                             |
|     | 4   | 4        | 4      | 6      | 18     | 12     | 10     | $a[4] = 12 < a[3] = 18$ : $minIdx \leftarrow 4$           |
|     | 5   | 5        | 4      | 6      | 18     | 12     | 10     | $a[5] = 10 < a[4] = 12$ : $minIdx \leftarrow 5$           |
|     | –   | –        | 4      | 6      | 10     | 12     | 18     | $minIdx = 5 \neq 3$ : scambia $a[3] \leftrightarrow a[5]$ |
| 4   | –   | 4        | 4      | 6      | 10     | 12     | 18     | $minIdx := 4$ ( $a[4] = 12$ )                             |
|     | 5   | 4        | 4      | 6      | 10     | 12     | 18     | $a[5] = 18 \geq a[4] = 12$ : nessun aggiornamento         |
|     | –   | –        | 4      | 6      | 10     | 12     | 18     | $minIdx = 4$ : nessuno scambio                            |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                            |
|-----|----------|--------|--------|--------|--------|--------|-------------------------------------------------------------------|
| –   | –        | 12     | 4      | 18     | 6      | 10     | stato iniziale                                                    |
| 1   | 2        | 4      | 12     | 18     | 6      | 10     | scambia $a[1] \leftrightarrow a[2]$ : $a[1] = \min(a[1..5]) = 4$  |
| 2   | 4        | 4      | 6      | 18     | 12     | 10     | scambia $a[2] \leftrightarrow a[4]$ : $a[2] = \min(a[2..5]) = 6$  |
| 3   | 5        | 4      | 6      | 10     | 12     | 18     | scambia $a[3] \leftrightarrow a[5]$ : $a[3] = \min(a[3..5]) = 10$ |
| 4   | 4        | 4      | 6      | 10     | 12     | 18     | nessuno scambio: $a[4] = \min(a[4..5]) = 12$                      |

L'algoritmo esegue 10 confronti tra chiavi e 3 scambi, e l'array ordinato è

[4, 6, 10, 12, 18].

3. Array  $a = [5, 4, 3, 2, 1]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 5      | 4      | 3      | 2      | 1      | stato iniziale                                            |
| 1   | —   | 1        | 5      | 4      | 3      | 2      | 1      | $minIdx := 1$ ( $a[1] = 5$ )                              |
|     | 2   | 2        | 5      | 4      | 3      | 2      | 1      | $a[2] = 4 < a[1] = 5$ : $minIdx \leftarrow 2$             |
|     | 3   | 3        | 5      | 4      | 3      | 2      | 1      | $a[3] = 3 < a[2] = 4$ : $minIdx \leftarrow 3$             |
|     | 4   | 4        | 5      | 4      | 3      | 2      | 1      | $a[4] = 2 < a[3] = 3$ : $minIdx \leftarrow 4$             |
|     | 5   | 5        | 5      | 4      | 3      | 2      | 1      | $a[5] = 1 < a[4] = 2$ : $minIdx \leftarrow 5$             |
|     | —   | —        | 1      | 4      | 3      | 2      | 5      | $minIdx = 5 \neq 1$ : scambia $a[1] \leftrightarrow a[5]$ |
| 2   | —   | 2        | 1      | 4      | 3      | 2      | 5      | $minIdx := 2$ ( $a[2] = 4$ )                              |
|     | 3   | 3        | 1      | 4      | 3      | 2      | 5      | $a[3] = 3 < a[2] = 4$ : $minIdx \leftarrow 3$             |
|     | 4   | 4        | 1      | 4      | 3      | 2      | 5      | $a[4] = 2 < a[3] = 3$ : $minIdx \leftarrow 4$             |
|     | 5   | 4        | 1      | 4      | 3      | 2      | 5      | $a[5] = 5 \geq a[4] = 2$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | $minIdx = 4 \neq 2$ : scambia $a[2] \leftrightarrow a[4]$ |
| 3   | —   | 3        | 1      | 2      | 3      | 4      | 5      | $minIdx := 3$ ( $a[3] = 3$ )                              |
|     | 4   | 3        | 1      | 2      | 3      | 4      | 5      | $a[4] = 4 \geq a[3] = 3$ : nessun aggiornamento           |
|     | 5   | 3        | 1      | 2      | 3      | 4      | 5      | $a[5] = 5 \geq a[3] = 3$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | $minIdx = 3$ : nessuno scambio                            |
| 4   | —   | 4        | 1      | 2      | 3      | 4      | 5      | $minIdx := 4$ ( $a[4] = 4$ )                              |
|     | 5   | 4        | 1      | 2      | 3      | 4      | 5      | $a[5] = 5 \geq a[4] = 4$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 3      | 4      | 5      | $minIdx = 4$ : nessuno scambio                            |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 5      | 4      | 3      | 2      | 1      | stato iniziale                                                   |
| 1   | 5        | 1      | 4      | 3      | 2      | 5      | scambia $a[1] \leftrightarrow a[5]$ : $a[1] = \min(a[1..5]) = 1$ |
| 2   | 4        | 1      | 2      | 3      | 4      | 5      | scambia $a[2] \leftrightarrow a[4]$ : $a[2] = \min(a[2..5]) = 2$ |
| 3   | 3        | 1      | 2      | 3      | 4      | 5      | nessuno scambio: $a[3] = \min(a[3..5]) = 3$                      |
| 4   | 4        | 1      | 2      | 3      | 4      | 5      | nessuno scambio: $a[4] = \min(a[4..5]) = 4$                      |

L'algoritmo esegue 10 confronti tra chiavi e 2 scambi, e l'array ordinato è

$$[1, 2, 3, 4, 5].$$

#### 4. Array $a = [100, 50, 25]$ ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 100    | 50     | 25     | stato iniziale                                            |
| 1   | —   | 1        | 100    | 50     | 25     | $minIdx := 1$ ( $a[1] = 100$ )                            |
|     | 2   | 2        | 100    | 50     | 25     | $a[2] = 50 < a[1] = 100$ : $minIdx \leftarrow 2$          |
|     | 3   | 3        | 100    | 50     | 25     | $a[3] = 25 < a[2] = 50$ : $minIdx \leftarrow 3$           |
|     | —   | —        | 25     | 50     | 100    | $minIdx = 3 \neq 1$ : scambia $a[1] \leftrightarrow a[3]$ |
| 2   | —   | 2        | 25     | 50     | 100    | $minIdx := 2$ ( $a[2] = 50$ )                             |
|     | 3   | 2        | 25     | 50     | 100    | $a[3] = 100 \geq a[2] = 50$ : nessun aggiornamento        |
|     | —   | —        | 25     | 50     | 100    | $minIdx = 2$ : nessuno scambio                            |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                            |
|-----|----------|--------|--------|--------|-------------------------------------------------------------------|
| —   | —        | 100    | 50     | 25     | stato iniziale                                                    |
| 1   | 3        | 25     | 50     | 100    | scambia $a[1] \leftrightarrow a[3]$ : $a[1] = \min(a[1..3]) = 25$ |
| 2   | 2        | 25     | 50     | 100    | nessuno scambio: $a[2] = \min(a[2..3]) = 50$                      |

L'algoritmo esegue 3 confronti tra chiavi e 1 scambio, e l'array ordinato è

[25, 50, 100].

5. Array  $a = [8]$  ( $n = 1$ ).

L'array ha un solo elemento: il ciclo esterno non viene eseguito e l'array è già ordinato.

| $a[1]$ | Azione                                                        |
|--------|---------------------------------------------------------------|
| 8      | stato iniziale: $n = 1$ , il ciclo esterno non viene eseguito |

L'array ordinato è [8], ottenuto senza alcun confronto tra chiavi e senza alcuno scambio.

### Esercizio 3.17 — Algoritmi di ordinamento (E17)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [6, 1, 8, 2, 5]$
2.  $a = [13, 11, 9, 7]$
3.  $a = [2, 8, 4, 6]$
4.  $a = [30, 20, 10]$
5.  $a = [5, 10, 15, 20]$

#### Domande

1. Applicare l'algoritmo **SelectionSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni iterazione del ciclo esterno, indicare l'indice  $minIdx$  del minimo trovato dal ciclo interno e dire se viene eseguito uno scambio.
2. Per ciascun array, contare i confronti tra chiavi e gli scambi eseguiti dall'algoritmo.

## Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [6, 1, 8, 2, 5]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 6      | 1      | 8      | 2      | 5      | stato iniziale                                            |
| 1   | —   | 1        | 6      | 1      | 8      | 2      | 5      | $minIdx := 1$ ( $a[1] = 6$ )                              |
|     | 2   | 2        | 6      | 1      | 8      | 2      | 5      | $a[2] = 1 < a[1] = 6$ : $minIdx \leftarrow 2$             |
|     | 3   | 2        | 6      | 1      | 8      | 2      | 5      | $a[3] = 8 \geq a[2] = 1$ : nessun aggiornamento           |
|     | 4   | 2        | 6      | 1      | 8      | 2      | 5      | $a[4] = 2 \geq a[2] = 1$ : nessun aggiornamento           |
|     | 5   | 2        | 6      | 1      | 8      | 2      | 5      | $a[5] = 5 \geq a[2] = 1$ : nessun aggiornamento           |
|     | —   | —        | 1      | 6      | 8      | 2      | 5      | $minIdx = 2 \neq 1$ : scambia $a[1] \leftrightarrow a[2]$ |
| 2   | —   | 2        | 1      | 6      | 8      | 2      | 5      | $minIdx := 2$ ( $a[2] = 6$ )                              |
|     | 3   | 2        | 1      | 6      | 8      | 2      | 5      | $a[3] = 8 \geq a[2] = 6$ : nessun aggiornamento           |
|     | 4   | 4        | 1      | 6      | 8      | 2      | 5      | $a[4] = 2 < a[2] = 6$ : $minIdx \leftarrow 4$             |
|     | 5   | 4        | 1      | 6      | 8      | 2      | 5      | $a[5] = 5 \geq a[4] = 2$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 8      | 6      | 5      | $minIdx = 4 \neq 2$ : scambia $a[2] \leftrightarrow a[4]$ |
| 3   | —   | 3        | 1      | 2      | 8      | 6      | 5      | $minIdx := 3$ ( $a[3] = 8$ )                              |
|     | 4   | 4        | 1      | 2      | 8      | 6      | 5      | $a[4] = 6 < a[3] = 8$ : $minIdx \leftarrow 4$             |
|     | 5   | 5        | 1      | 2      | 8      | 6      | 5      | $a[5] = 5 < a[4] = 6$ : $minIdx \leftarrow 5$             |
|     | —   | —        | 1      | 2      | 5      | 6      | 8      | $minIdx = 5 \neq 3$ : scambia $a[3] \leftrightarrow a[5]$ |
| 4   | —   | 4        | 1      | 2      | 5      | 6      | 8      | $minIdx := 4$ ( $a[4] = 6$ )                              |
|     | 5   | 4        | 1      | 2      | 5      | 6      | 8      | $a[5] = 8 \geq a[4] = 6$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 5      | 6      | 8      | $minIdx = 4$ : nessuno scambio                            |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 6      | 1      | 8      | 2      | 5      | stato iniziale                                                   |
| 1   | 2        | 1      | 6      | 8      | 2      | 5      | scambia $a[1] \leftrightarrow a[2]$ : $a[1] = \min(a[1..5]) = 1$ |
| 2   | 4        | 1      | 2      | 8      | 6      | 5      | scambia $a[2] \leftrightarrow a[4]$ : $a[2] = \min(a[2..5]) = 2$ |
| 3   | 5        | 1      | 2      | 5      | 6      | 8      | scambia $a[3] \leftrightarrow a[5]$ : $a[3] = \min(a[3..5]) = 5$ |
| 4   | 4        | 1      | 2      | 5      | 6      | 8      | nessuno scambio: $a[4] = \min(a[4..5]) = 6$                      |

L'algoritmo esegue 10 confronti tra chiavi e 3 scambi, e l'array ordinato è

$[1, 2, 5, 6, 8]$ .

2. Array  $a = [13, 11, 9, 7]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 13     | 11     | 9      | 7      | stato iniziale                                            |
| 1   | —   | 1        | 13     | 11     | 9      | 7      | $minIdx := 1$ ( $a[1] = 13$ )                             |
|     | 2   | 2        | 13     | 11     | 9      | 7      | $a[2] = 11 < a[1] = 13$ : $minIdx \leftarrow 2$           |
|     | 3   | 3        | 13     | 11     | 9      | 7      | $a[3] = 9 < a[2] = 11$ : $minIdx \leftarrow 3$            |
|     | 4   | 4        | 13     | 11     | 9      | 7      | $a[4] = 7 < a[3] = 9$ : $minIdx \leftarrow 4$             |
|     | —   | —        | 7      | 11     | 9      | 13     | $minIdx = 4 \neq 1$ : scambia $a[1] \leftrightarrow a[4]$ |
| 2   | —   | 2        | 7      | 11     | 9      | 13     | $minIdx := 2$ ( $a[2] = 11$ )                             |
|     | 3   | 3        | 7      | 11     | 9      | 13     | $a[3] = 9 < a[2] = 11$ : $minIdx \leftarrow 3$            |
|     | 4   | 3        | 7      | 11     | 9      | 13     | $a[4] = 13 \geq a[3] = 9$ : nessun aggiornamento          |
|     | —   | —        | 7      | 9      | 11     | 13     | $minIdx = 3 \neq 2$ : scambia $a[2] \leftrightarrow a[3]$ |
| 3   | —   | 3        | 7      | 9      | 11     | 13     | $minIdx := 3$ ( $a[3] = 11$ )                             |
|     | 4   | 3        | 7      | 9      | 11     | 13     | $a[4] = 13 \geq a[3] = 11$ : nessun aggiornamento         |
|     | —   | —        | 7      | 9      | 11     | 13     | $minIdx = 3$ : nessuno scambio                            |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 13     | 11     | 9      | 7      | stato iniziale                                                   |
| 1   | 4        | 7      | 11     | 9      | 13     | scambia $a[1] \leftrightarrow a[4]$ : $a[1] = \min(a[1..4]) = 7$ |
| 2   | 3        | 7      | 9      | 11     | 13     | scambia $a[2] \leftrightarrow a[3]$ : $a[2] = \min(a[2..4]) = 9$ |
| 3   | 3        | 7      | 9      | 11     | 13     | nessuno scambio: $a[3] = \min(a[3..4]) = 11$                     |

L'algoritmo esegue 6 confronti tra chiavi e 2 scambi, e l'array ordinato è

[7, 9, 11, 13].

3. Array  $a = [2, 8, 4, 6]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 2      | 8      | 4      | 6      | stato iniziale                                            |
| 1   | —   | 1        | 2      | 8      | 4      | 6      | $minIdx := 1$ ( $a[1] = 2$ )                              |
|     | 2   | 1        | 2      | 8      | 4      | 6      | $a[2] = 8 \geq a[1] = 2$ : nessun aggiornamento           |
|     | 3   | 1        | 2      | 8      | 4      | 6      | $a[3] = 4 \geq a[1] = 2$ : nessun aggiornamento           |
|     | 4   | 1        | 2      | 8      | 4      | 6      | $a[4] = 6 \geq a[1] = 2$ : nessun aggiornamento           |
|     | —   | —        | 2      | 8      | 4      | 6      | $minIdx = 1$ : nessuno scambio                            |
| 2   | —   | 2        | 2      | 8      | 4      | 6      | $minIdx := 2$ ( $a[2] = 8$ )                              |
|     | 3   | 3        | 2      | 8      | 4      | 6      | $a[3] = 4 < a[2] = 8$ : $minIdx \leftarrow 3$             |
|     | 4   | 3        | 2      | 8      | 4      | 6      | $a[4] = 6 \geq a[3] = 4$ : nessun aggiornamento           |
|     | —   | —        | 2      | 4      | 8      | 6      | $minIdx = 3 \neq 2$ : scambia $a[2] \leftrightarrow a[3]$ |
| 3   | —   | 3        | 2      | 4      | 8      | 6      | $minIdx := 3$ ( $a[3] = 8$ )                              |
|     | 4   | 4        | 2      | 4      | 8      | 6      | $a[4] = 6 < a[3] = 8$ : $minIdx \leftarrow 4$             |
|     | —   | —        | 2      | 4      | 6      | 8      | $minIdx = 4 \neq 3$ : scambia $a[3] \leftrightarrow a[4]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 2      | 8      | 4      | 6      | stato iniziale                                                   |
| 1   | 1        | 2      | 8      | 4      | 6      | nessuno scambio: $a[1] = \min(a[1..4]) = 2$                      |
| 2   | 3        | 2      | 4      | 8      | 6      | scambia $a[2] \leftrightarrow a[3]$ : $a[2] = \min(a[2..4]) = 4$ |
| 3   | 4        | 2      | 4      | 6      | 8      | scambia $a[3] \leftrightarrow a[4]$ : $a[3] = \min(a[3..4]) = 6$ |

L'algoritmo esegue 6 confronti tra chiavi e 2 scambi, e l'array ordinato è

$[2, 4, 6, 8]$ .

4. Array  $a = [30, 20, 10]$  ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 30     | 20     | 10     | stato iniziale                                            |
| 1   | —   | 1        | 30     | 20     | 10     | $minIdx := 1$ ( $a[1] = 30$ )                             |
|     | 2   | 2        | 30     | 20     | 10     | $a[2] = 20 < a[1] = 30$ : $minIdx \leftarrow 2$           |
|     | 3   | 3        | 30     | 20     | 10     | $a[3] = 10 < a[2] = 20$ : $minIdx \leftarrow 3$           |
|     | —   | —        | 10     | 20     | 30     | $minIdx = 3 \neq 1$ : scambia $a[1] \leftrightarrow a[3]$ |
| 2   | —   | 2        | 10     | 20     | 30     | $minIdx := 2$ ( $a[2] = 20$ )                             |
|     | 3   | 2        | 10     | 20     | 30     | $a[3] = 30 \geq a[2] = 20$ : nessun aggiornamento         |
|     | —   | —        | 10     | 20     | 30     | $minIdx = 2$ : nessuno scambio                            |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                            |
|-----|----------|--------|--------|--------|-------------------------------------------------------------------|
| —   | —        | 30     | 20     | 10     | stato iniziale                                                    |
| 1   | 3        | 10     | 20     | 30     | scambia $a[1] \leftrightarrow a[3]$ : $a[1] = \min(a[1..3]) = 10$ |
| 2   | 2        | 10     | 20     | 30     | nessuno scambio: $a[2] = \min(a[2..3]) = 20$                      |

L'algoritmo esegue 3 confronti tra chiavi e 1 scambio, e l'array ordinato è

$[10, 20, 30]$ .

5. Array  $a = [5, 10, 15, 20]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                            |
|-----|-----|----------|--------|--------|--------|--------|---------------------------------------------------|
| —   | —   | —        | 5      | 10     | 15     | 20     | stato iniziale                                    |
| 1   | —   | 1        | 5      | 10     | 15     | 20     | $minIdx := 1$ ( $a[1] = 5$ )                      |
|     | 2   | 1        | 5      | 10     | 15     | 20     | $a[2] = 10 \geq a[1] = 5$ : nessun aggiornamento  |
|     | 3   | 1        | 5      | 10     | 15     | 20     | $a[3] = 15 \geq a[1] = 5$ : nessun aggiornamento  |
|     | 4   | 1        | 5      | 10     | 15     | 20     | $a[4] = 20 \geq a[1] = 5$ : nessun aggiornamento  |
|     | —   | —        | 5      | 10     | 15     | 20     | $minIdx = 1$ : nessuno scambio                    |
| 2   | —   | 2        | 5      | 10     | 15     | 20     | $minIdx := 2$ ( $a[2] = 10$ )                     |
|     | 3   | 2        | 5      | 10     | 15     | 20     | $a[3] = 15 \geq a[2] = 10$ : nessun aggiornamento |
|     | 4   | 2        | 5      | 10     | 15     | 20     | $a[4] = 20 \geq a[2] = 10$ : nessun aggiornamento |
|     | —   | —        | 5      | 10     | 15     | 20     | $minIdx = 2$ : nessuno scambio                    |
| 3   | —   | 3        | 5      | 10     | 15     | 20     | $minIdx := 3$ ( $a[3] = 15$ )                     |
|     | 4   | 3        | 5      | 10     | 15     | 20     | $a[4] = 20 \geq a[3] = 15$ : nessun aggiornamento |
|     | —   | —        | 5      | 10     | 15     | 20     | $minIdx = 3$ : nessuno scambio                    |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                       |
|-----|----------|--------|--------|--------|--------|----------------------------------------------|
| —   | —        | 5      | 10     | 15     | 20     | stato iniziale                               |
| 1   | 1        | 5      | 10     | 15     | 20     | nessuno scambio: $a[1] = \min(a[1..4]) = 5$  |
| 2   | 2        | 5      | 10     | 15     | 20     | nessuno scambio: $a[2] = \min(a[2..4]) = 10$ |
| 3   | 3        | 5      | 10     | 15     | 20     | nessuno scambio: $a[3] = \min(a[3..4]) = 15$ |

L'algoritmo esegue 6 confronti tra chiavi e 0 scambi, e l'array ordinato è

[5, 10, 15, 20].

### Esercizio 3.18 — Algoritmi di ordinamento (E18)

Consideriamo i seguenti array di numeri naturali:

1.  $a = [14, 7, 21, 3, 18]$
2.  $a = [40, 30, 20, 10]$
3.  $a = [3, 6, 1, 9, 2]$
4.  $a = [25, 12, 50]$
5.  $a = [17, 17, 17, 17]$

#### Domande

1. Applicare l'algoritmo **SelectionSort** a ciascun array, per ordinarlo in ordine non decrescente. Mostrare l'esecuzione passo per passo: per ogni iterazione del ciclo esterno, indicare l'indice  $minIdx$  del minimo trovato dal ciclo interno e dire se viene eseguito uno scambio.
2. Per ciascun array, contare i confronti tra chiavi e gli scambi eseguiti dall'algoritmo.

## Soluzione

Le tabelle di traccia seguono le convenzioni enunciate nell'introduzione agli esercizi di questo capitolo. Ogni punto risponde a entrambe le domande.

1. Array  $a = [14, 7, 21, 3, 18]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|--------|-----------------------------------------------------------|
| –   | –   | –        | 14     | 7      | 21     | 3      | 18     | stato iniziale                                            |
| 1   | –   | 1        | 14     | 7      | 21     | 3      | 18     | $minIdx := 1$ ( $a[1] = 14$ )                             |
|     | 2   | 2        | 14     | 7      | 21     | 3      | 18     | $a[2] = 7 < a[1] = 14$ : $minIdx \leftarrow 2$            |
|     | 3   | 2        | 14     | 7      | 21     | 3      | 18     | $a[3] = 21 \geq a[2] = 7$ : nessun aggiornamento          |
|     | 4   | 4        | 14     | 7      | 21     | 3      | 18     | $a[4] = 3 < a[2] = 7$ : $minIdx \leftarrow 4$             |
|     | 5   | 4        | 14     | 7      | 21     | 3      | 18     | $a[5] = 18 \geq a[4] = 3$ : nessun aggiornamento          |
|     | –   | –        | 3      | 7      | 21     | 14     | 18     | $minIdx = 4 \neq 1$ : scambia $a[1] \leftrightarrow a[4]$ |
| 2   | –   | 2        | 3      | 7      | 21     | 14     | 18     | $minIdx := 2$ ( $a[2] = 7$ )                              |
|     | 3   | 2        | 3      | 7      | 21     | 14     | 18     | $a[3] = 21 \geq a[2] = 7$ : nessun aggiornamento          |
|     | 4   | 2        | 3      | 7      | 21     | 14     | 18     | $a[4] = 14 \geq a[2] = 7$ : nessun aggiornamento          |
|     | 5   | 2        | 3      | 7      | 21     | 14     | 18     | $a[5] = 18 \geq a[2] = 7$ : nessun aggiornamento          |
|     | –   | –        | 3      | 7      | 21     | 14     | 18     | $minIdx = 2$ : nessuno scambio                            |
| 3   | –   | 3        | 3      | 7      | 21     | 14     | 18     | $minIdx := 3$ ( $a[3] = 21$ )                             |
|     | 4   | 4        | 3      | 7      | 21     | 14     | 18     | $a[4] = 14 < a[3] = 21$ : $minIdx \leftarrow 4$           |
|     | 5   | 4        | 3      | 7      | 21     | 14     | 18     | $a[5] = 18 \geq a[4] = 14$ : nessun aggiornamento         |
|     | –   | –        | 3      | 7      | 14     | 21     | 18     | $minIdx = 4 \neq 3$ : scambia $a[3] \leftrightarrow a[4]$ |
| 4   | –   | 4        | 3      | 7      | 14     | 21     | 18     | $minIdx := 4$ ( $a[4] = 21$ )                             |
|     | 5   | 5        | 3      | 7      | 14     | 21     | 18     | $a[5] = 18 < a[4] = 21$ : $minIdx \leftarrow 5$           |
|     | –   | –        | 3      | 7      | 14     | 18     | 21     | $minIdx = 5 \neq 4$ : scambia $a[4] \leftrightarrow a[5]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                            |
|-----|----------|--------|--------|--------|--------|--------|-------------------------------------------------------------------|
| –   | –        | 14     | 7      | 21     | 3      | 18     | stato iniziale                                                    |
| 1   | 4        | 3      | 7      | 21     | 14     | 18     | scambia $a[1] \leftrightarrow a[4]$ : $a[1] = \min(a[1..5]) = 3$  |
| 2   | 2        | 3      | 7      | 21     | 14     | 18     | nessuno scambio: $a[2] = \min(a[2..5]) = 7$                       |
| 3   | 4        | 3      | 7      | 14     | 21     | 18     | scambia $a[3] \leftrightarrow a[4]$ : $a[3] = \min(a[3..5]) = 14$ |
| 4   | 5        | 3      | 7      | 14     | 18     | 21     | scambia $a[4] \leftrightarrow a[5]$ : $a[4] = \min(a[4..5]) = 18$ |

L'algoritmo esegue 10 confronti tra chiavi e 3 scambi, e l'array ordinato è

$[3, 7, 14, 18, 21]$ .

2. Array  $a = [40, 30, 20, 10]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 40     | 30     | 20     | 10     | stato iniziale                                            |
| 1   | —   | 1        | 40     | 30     | 20     | 10     | $minIdx := 1$ ( $a[1] = 40$ )                             |
|     | 2   | 2        | 40     | 30     | 20     | 10     | $a[2] = 30 < a[1] = 40$ : $minIdx \leftarrow 2$           |
|     | 3   | 3        | 40     | 30     | 20     | 10     | $a[3] = 20 < a[2] = 30$ : $minIdx \leftarrow 3$           |
|     | 4   | 4        | 40     | 30     | 20     | 10     | $a[4] = 10 < a[3] = 20$ : $minIdx \leftarrow 4$           |
|     | —   | —        | 10     | 30     | 20     | 40     | $minIdx = 4 \neq 1$ : scambia $a[1] \leftrightarrow a[4]$ |
| 2   | —   | 2        | 10     | 30     | 20     | 40     | $minIdx := 2$ ( $a[2] = 30$ )                             |
|     | 3   | 3        | 10     | 30     | 20     | 40     | $a[3] = 20 < a[2] = 30$ : $minIdx \leftarrow 3$           |
|     | 4   | 3        | 10     | 30     | 20     | 40     | $a[4] = 40 \geq a[3] = 20$ : nessun aggiornamento         |
|     | —   | —        | 10     | 20     | 30     | 40     | $minIdx = 3 \neq 2$ : scambia $a[2] \leftrightarrow a[3]$ |
| 3   | —   | 3        | 10     | 20     | 30     | 40     | $minIdx := 3$ ( $a[3] = 30$ )                             |
|     | 4   | 3        | 10     | 20     | 30     | 40     | $a[4] = 40 \geq a[3] = 30$ : nessun aggiornamento         |
|     | —   | —        | 10     | 20     | 30     | 40     | $minIdx = 3$ : nessuno scambio                            |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                                            |
|-----|----------|--------|--------|--------|--------|-------------------------------------------------------------------|
| —   | —        | 40     | 30     | 20     | 10     | stato iniziale                                                    |
| 1   | 4        | 10     | 30     | 20     | 40     | scambia $a[1] \leftrightarrow a[4]$ : $a[1] = \min(a[1..4]) = 10$ |
| 2   | 3        | 10     | 20     | 30     | 40     | scambia $a[2] \leftrightarrow a[3]$ : $a[2] = \min(a[2..4]) = 20$ |
| 3   | 3        | 10     | 20     | 30     | 40     | nessuno scambio: $a[3] = \min(a[3..4]) = 30$                      |

L'algoritmo esegue 6 confronti tra chiavi e 2 scambi, e l'array ordinato è

[10, 20, 30, 40].

3. Array  $a = [3, 6, 1, 9, 2]$  ( $n = 5$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 3      | 6      | 1      | 9      | 2      | stato iniziale                                            |
| 1   | —   | 1        | 3      | 6      | 1      | 9      | 2      | $minIdx := 1$ ( $a[1] = 3$ )                              |
|     | 2   | 1        | 3      | 6      | 1      | 9      | 2      | $a[2] = 6 \geq a[1] = 3$ : nessun aggiornamento           |
|     | 3   | 3        | 3      | 6      | 1      | 9      | 2      | $a[3] = 1 < a[1] = 3$ : $minIdx \leftarrow 3$             |
|     | 4   | 3        | 3      | 6      | 1      | 9      | 2      | $a[4] = 9 \geq a[3] = 1$ : nessun aggiornamento           |
|     | 5   | 3        | 3      | 6      | 1      | 9      | 2      | $a[5] = 2 \geq a[3] = 1$ : nessun aggiornamento           |
|     | —   | —        | 1      | 6      | 3      | 9      | 2      | $minIdx = 3 \neq 1$ : scambia $a[1] \leftrightarrow a[3]$ |
| 2   | —   | 2        | 1      | 6      | 3      | 9      | 2      | $minIdx := 2$ ( $a[2] = 6$ )                              |
|     | 3   | 3        | 1      | 6      | 3      | 9      | 2      | $a[3] = 3 < a[2] = 6$ : $minIdx \leftarrow 3$             |
|     | 4   | 3        | 1      | 6      | 3      | 9      | 2      | $a[4] = 9 \geq a[3] = 3$ : nessun aggiornamento           |
|     | 5   | 5        | 1      | 6      | 3      | 9      | 2      | $a[5] = 2 < a[3] = 3$ : $minIdx \leftarrow 5$             |
|     | —   | —        | 1      | 2      | 3      | 9      | 6      | $minIdx = 5 \neq 2$ : scambia $a[2] \leftrightarrow a[5]$ |
|     | —   | —        | 1      | 2      | 3      | 9      | 6      |                                                           |
| 3   | —   | 3        | 1      | 2      | 3      | 9      | 6      | $minIdx := 3$ ( $a[3] = 3$ )                              |
|     | 4   | 3        | 1      | 2      | 3      | 9      | 6      | $a[4] = 9 \geq a[3] = 3$ : nessun aggiornamento           |
|     | 5   | 3        | 1      | 2      | 3      | 9      | 6      | $a[5] = 6 \geq a[3] = 3$ : nessun aggiornamento           |
|     | —   | —        | 1      | 2      | 3      | 9      | 6      | $minIdx = 3$ : nessuno scambio                            |
| 4   | —   | 4        | 1      | 2      | 3      | 9      | 6      | $minIdx := 4$ ( $a[4] = 9$ )                              |
|     | 5   | 5        | 1      | 2      | 3      | 9      | 6      | $a[5] = 6 < a[4] = 9$ : $minIdx \leftarrow 5$             |
|     | —   | —        | 1      | 2      | 3      | 6      | 9      | $minIdx = 5 \neq 4$ : scambia $a[4] \leftrightarrow a[5]$ |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | $a[5]$ | Azione                                                           |
|-----|----------|--------|--------|--------|--------|--------|------------------------------------------------------------------|
| —   | —        | 3      | 6      | 1      | 9      | 2      | stato iniziale                                                   |
| 1   | 3        | 1      | 6      | 3      | 9      | 2      | scambia $a[1] \leftrightarrow a[3]$ : $a[1] = \min(a[1..5]) = 1$ |
| 2   | 5        | 1      | 2      | 3      | 9      | 6      | scambia $a[2] \leftrightarrow a[5]$ : $a[2] = \min(a[2..5]) = 2$ |
| 3   | 3        | 1      | 2      | 3      | 9      | 6      | nessuno scambio: $a[3] = \min(a[3..5]) = 3$                      |
| 4   | 5        | 1      | 2      | 3      | 6      | 9      | scambia $a[4] \leftrightarrow a[5]$ : $a[4] = \min(a[4..5]) = 6$ |

L'algoritmo esegue 10 confronti tra chiavi e 3 scambi, e l'array ordinato è

$$[1, 2, 3, 6, 9].$$

#### 4. Array $a = [25, 12, 50]$ ( $n = 3$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                    |
|-----|-----|----------|--------|--------|--------|-----------------------------------------------------------|
| —   | —   | —        | 25     | 12     | 50     | stato iniziale                                            |
| 1   | —   | 1        | 25     | 12     | 50     | $minIdx := 1$ ( $a[1] = 25$ )                             |
|     | 2   | 2        | 25     | 12     | 50     | $a[2] = 12 < a[1] = 25$ : $minIdx \leftarrow 2$           |
|     | 3   | 2        | 25     | 12     | 50     | $a[3] = 50 \geq a[2] = 12$ : nessun aggiornamento         |
|     | —   | —        | 12     | 25     | 50     | $minIdx = 2 \neq 1$ : scambia $a[1] \leftrightarrow a[2]$ |
| 2   | —   | 2        | 12     | 25     | 50     | $minIdx := 2$ ( $a[2] = 25$ )                             |
|     | 3   | 2        | 12     | 25     | 50     | $a[3] = 50 \geq a[2] = 25$ : nessun aggiornamento         |
|     | —   | —        | 12     | 25     | 50     | $minIdx = 2$ : nessuno scambio                            |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | Azione                                                            |
|-----|----------|--------|--------|--------|-------------------------------------------------------------------|
| —   | —        | 25     | 12     | 50     | stato iniziale                                                    |
| 1   | 2        | 12     | 25     | 50     | scambia $a[1] \leftrightarrow a[2]$ : $a[1] = \min(a[1..3]) = 12$ |
| 2   | 2        | 12     | 25     | 50     | nessuno scambio: $a[2] = \min(a[2..3]) = 25$                      |

L'algoritmo esegue 3 confronti tra chiavi e 1 scambio, e l'array ordinato è

[12, 25, 50].

5. Array  $a = [17, 17, 17, 17]$  ( $n = 4$ ).

Traccia dettagliata:

| $i$ | $j$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                            |
|-----|-----|----------|--------|--------|--------|--------|---------------------------------------------------|
| —   | —   | —        | 17     | 17     | 17     | 17     | stato iniziale                                    |
| 1   | —   | 1        | 17     | 17     | 17     | 17     | $minIdx := 1$ ( $a[1] = 17$ )                     |
|     | 2   | 1        | 17     | 17     | 17     | 17     | $a[2] = 17 \geq a[1] = 17$ : nessun aggiornamento |
|     | 3   | 1        | 17     | 17     | 17     | 17     | $a[3] = 17 \geq a[1] = 17$ : nessun aggiornamento |
|     | 4   | 1        | 17     | 17     | 17     | 17     | $a[4] = 17 \geq a[1] = 17$ : nessun aggiornamento |
|     | —   | —        | 17     | 17     | 17     | 17     | $minIdx = 1$ : nessuno scambio                    |
| 2   | —   | 2        | 17     | 17     | 17     | 17     | $minIdx := 2$ ( $a[2] = 17$ )                     |
|     | 3   | 2        | 17     | 17     | 17     | 17     | $a[3] = 17 \geq a[2] = 17$ : nessun aggiornamento |
|     | 4   | 2        | 17     | 17     | 17     | 17     | $a[4] = 17 \geq a[2] = 17$ : nessun aggiornamento |
|     | —   | —        | 17     | 17     | 17     | 17     | $minIdx = 2$ : nessuno scambio                    |
| 3   | —   | 3        | 17     | 17     | 17     | 17     | $minIdx := 3$ ( $a[3] = 17$ )                     |
|     | 4   | 3        | 17     | 17     | 17     | 17     | $a[4] = 17 \geq a[3] = 17$ : nessun aggiornamento |
|     | —   | —        | 17     | 17     | 17     | 17     | $minIdx = 3$ : nessuno scambio                    |

Array dopo ogni iterazione del ciclo esterno:

| $i$ | $minIdx$ | $a[1]$ | $a[2]$ | $a[3]$ | $a[4]$ | Azione                                       |
|-----|----------|--------|--------|--------|--------|----------------------------------------------|
| —   | —        | 17     | 17     | 17     | 17     | stato iniziale                               |
| 1   | 1        | 17     | 17     | 17     | 17     | nessuno scambio: $a[1] = \min(a[1..4]) = 17$ |
| 2   | 2        | 17     | 17     | 17     | 17     | nessuno scambio: $a[2] = \min(a[2..4]) = 17$ |
| 3   | 3        | 17     | 17     | 17     | 17     | nessuno scambio: $a[3] = \min(a[3..4]) = 17$ |

L'algoritmo esegue 6 confronti tra chiavi e 0 scambi, e l'array ordinato è

[17, 17, 17, 17].

---

## Ricerca lineare e binaria

---

Negli esercizi di questo capitolo le ricerche sono svolte con i due algoritmi del capitolo «Ricerca lineare e ricerca binaria», richiamati di seguito. Gli indici degli array vanno da 1 a  $n$ , e viene restituito il valore  $-1$  quando il valore cercato  $m$  non è nell'array.

---

**Algorithm:** LinearSearch

---

**Input** : Un array  $a[1..n]$  di  $n$  numeri naturali e un valore  $m$  da cercare

**Output**: Indice di  $m$  in  $a$ , oppure  $-1$  se non trovato

```
1 function LinearSearch( $a, n, m$ )
2    $i := 1$ 
3   while  $i \leq n$  do
4     if  $a[i] = m$  then
5       return  $i$                                 // Elemento trovato all'indice  $i$ 
6     end
7      $i \leftarrow i + 1$ 
8   end
9   return  $-1$                                     // Elemento non trovato
10 end
```

---

**Algorithm:** BinarySearch

**Input** : Un array  $a[1..n]$  di  $n$  numeri naturali ordinato in modo non decrescente e un valore  $m$  da cercare

**Output:** Indice di  $m$  in  $a$ , oppure  $-1$  se non trovato

```

1 function BinarySearch( $a, n, m$ )
2    $left := 1$ 
3    $right := n$ 
4   while  $left \leq right$  do
5      $mid := left + \lfloor (right - left)/2 \rfloor$ 
6     if  $a[mid] = m$  then
7       return  $mid$                                 // Elemento trovato
8     end
9     if  $a[mid] < m$  then
10       $left \leftarrow mid + 1$                         // Cerca nella metà destra
11    end
12    else
13       $right \leftarrow mid - 1$                        // Cerca nella metà sinistra
14    end
15  end
16  return  $-1$                                          // Elemento non trovato
17 end

```

**Esercizio 4.1 — Ricerca lineare e ricerca binaria (E1)**

Consideriamo il seguente array:

$$a = [7, 2, 14, 3, 8, 11, 5, 9]$$

con  $n = 8$ .

**Domande**

1. Applicare l'algoritmo di ricerca lineare (Linear Search) all'array  $a$  per ciascuno dei seguenti valori cercati:

- (a)  $m = 11$
- (b)  $m = 7$
- (c)  $m = 9$
- (d)  $m = 4$
- (e)  $m = 14$

Per ogni ricerca, mostrare l'esecuzione passo per passo, indicando a ogni iterazione i valori di  $i$ ,  $a[i]$  e il risultato del confronto, e dare il valore restituito dall'algoritmo.

## Soluzione

1. Ricerca di  $m = 11$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 7      | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 2      | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 14     | $a[i] \neq m \Rightarrow$ continua |
| 4          | 4   | 3      | $a[i] \neq m \Rightarrow$ continua |
| 5          | 5   | 8      | $a[i] \neq m \Rightarrow$ continua |
| 6          | 6   | 11     | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 11 si trova all'indice 6.

2. Ricerca di  $m = 7$ :

| iterazione | $i$ | $a[i]$ | azione                          |
|------------|-----|--------|---------------------------------|
| 1          | 1   | 7      | $a[i] = m \Rightarrow$ Trovato! |

Quindi, l'elemento 7 si trova all'indice 1.

3. Ricerca di  $m = 9$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 7      | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 2      | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 14     | $a[i] \neq m \Rightarrow$ continua |
| 4          | 4   | 3      | $a[i] \neq m \Rightarrow$ continua |
| 5          | 5   | 8      | $a[i] \neq m \Rightarrow$ continua |
| 6          | 6   | 11     | $a[i] \neq m \Rightarrow$ continua |
| 7          | 7   | 5      | $a[i] \neq m \Rightarrow$ continua |
| 8          | 8   | 9      | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 9 si trova all'indice 8.

4. Ricerca di  $m = 4$ :

| iterazione | $i$ | $a[i]$ | azione                                             |
|------------|-----|--------|----------------------------------------------------|
| 1          | 1   | 7      | $a[i] \neq m \Rightarrow$ continua                 |
| 2          | 2   | 2      | $a[i] \neq m \Rightarrow$ continua                 |
| 3          | 3   | 14     | $a[i] \neq m \Rightarrow$ continua                 |
| 4          | 4   | 3      | $a[i] \neq m \Rightarrow$ continua                 |
| 5          | 5   | 8      | $a[i] \neq m \Rightarrow$ continua                 |
| 6          | 6   | 11     | $a[i] \neq m \Rightarrow$ continua                 |
| 7          | 7   | 5      | $a[i] \neq m \Rightarrow$ continua                 |
| 8          | 8   | 9      | $a[i] \neq m \Rightarrow$ continua                 |
| —          | 9   | —      | $i > n \Rightarrow$ il ciclo termina, non trovato! |

Quindi, l'elemento 4 non è presente nell'array (restituisce  $-1$ ).

5. Ricerca di  $m = 14$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 7      | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 2      | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 14     | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 14 si trova all'indice 3.

### Esercizio 4.2 — Ricerca lineare e ricerca binaria (E2)

Consideriamo il seguente array:

$$a = [15, 4, 22, 8, 1, 30, 17]$$

con  $n = 7$ .

#### Domande

1. Applicare l'algoritmo di ricerca lineare (Linear Search) all'array  $a$  per ciascuno dei seguenti valori cercati:

- (a)  $m = 22$
- (b)  $m = 15$
- (c)  $m = 17$
- (d)  $m = 10$
- (e)  $m = 1$

Per ogni ricerca, mostrare l'esecuzione passo per passo, indicando a ogni iterazione i valori di  $i$ ,  $a[i]$  e il risultato del confronto, e dare il valore restituito dall'algoritmo.

#### Soluzione

1. Ricerca di  $m = 22$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 15     | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 4      | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 22     | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 22 si trova all'indice 3.

2. Ricerca di  $m = 15$ :

| iterazione | $i$ | $a[i]$ | azione                          |
|------------|-----|--------|---------------------------------|
| 1          | 1   | 15     | $a[i] = m \Rightarrow$ Trovato! |

Quindi, l'elemento 15 si trova all'indice 1.

3. Ricerca di  $m = 17$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 15     | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 4      | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 22     | $a[i] \neq m \Rightarrow$ continua |
| 4          | 4   | 8      | $a[i] \neq m \Rightarrow$ continua |
| 5          | 5   | 1      | $a[i] \neq m \Rightarrow$ continua |
| 6          | 6   | 30     | $a[i] \neq m \Rightarrow$ continua |
| 7          | 7   | 17     | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 17 si trova all'indice 7.

4. Ricerca di  $m = 10$ :

| iterazione | $i$ | $a[i]$ | azione                                             |
|------------|-----|--------|----------------------------------------------------|
| 1          | 1   | 15     | $a[i] \neq m \Rightarrow$ continua                 |
| 2          | 2   | 4      | $a[i] \neq m \Rightarrow$ continua                 |
| 3          | 3   | 22     | $a[i] \neq m \Rightarrow$ continua                 |
| 4          | 4   | 8      | $a[i] \neq m \Rightarrow$ continua                 |
| 5          | 5   | 1      | $a[i] \neq m \Rightarrow$ continua                 |
| 6          | 6   | 30     | $a[i] \neq m \Rightarrow$ continua                 |
| 7          | 7   | 17     | $a[i] \neq m \Rightarrow$ continua                 |
| –          | 8   | –      | $i > n \Rightarrow$ il ciclo termina, non trovato! |

Quindi, l'elemento 10 non è presente nell'array (restituisce  $-1$ ).

5. Ricerca di  $m = 1$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 15     | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 4      | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 22     | $a[i] \neq m \Rightarrow$ continua |
| 4          | 4   | 8      | $a[i] \neq m \Rightarrow$ continua |
| 5          | 5   | 1      | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 1 si trova all'indice 5.

**Esercizio 4.3 — Ricerca lineare e ricerca binaria (E3)**

Consideriamo il seguente array:

$$a = [6, 19, 3, 27, 12, 5, 33]$$

con  $n = 7$ .

**Domande**

1. Applicare l'algoritmo di ricerca lineare (Linear Search) all'array  $a$  per ciascuno dei seguenti valori cercati:

- (a)  $m = 27$
- (b)  $m = 6$
- (c)  $m = 33$
- (d)  $m = 15$
- (e)  $m = 3$

Per ogni ricerca, mostrare l'esecuzione passo per passo, indicando a ogni iterazione i valori di  $i$ ,  $a[i]$  e il risultato del confronto, e dare il valore restituito dall'algoritmo.

**Soluzione**

1. Ricerca di  $m = 27$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 6      | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 19     | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 3      | $a[i] \neq m \Rightarrow$ continua |
| 4          | 4   | 27     | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 27 si trova all'indice 4.

2. Ricerca di  $m = 6$ :

| iterazione | $i$ | $a[i]$ | azione                          |
|------------|-----|--------|---------------------------------|
| 1          | 1   | 6      | $a[i] = m \Rightarrow$ Trovato! |

Quindi, l'elemento 6 si trova all'indice 1.

3. Ricerca di  $m = 33$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 6      | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 19     | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 3      | $a[i] \neq m \Rightarrow$ continua |
| 4          | 4   | 27     | $a[i] \neq m \Rightarrow$ continua |
| 5          | 5   | 12     | $a[i] \neq m \Rightarrow$ continua |
| 6          | 6   | 5      | $a[i] \neq m \Rightarrow$ continua |
| 7          | 7   | 33     | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 33 si trova all'indice 7.

4. Ricerca di  $m = 15$ :

| iterazione | $i$ | $a[i]$ | azione                                             |
|------------|-----|--------|----------------------------------------------------|
| 1          | 1   | 6      | $a[i] \neq m \Rightarrow$ continua                 |
| 2          | 2   | 19     | $a[i] \neq m \Rightarrow$ continua                 |
| 3          | 3   | 3      | $a[i] \neq m \Rightarrow$ continua                 |
| 4          | 4   | 27     | $a[i] \neq m \Rightarrow$ continua                 |
| 5          | 5   | 12     | $a[i] \neq m \Rightarrow$ continua                 |
| 6          | 6   | 5      | $a[i] \neq m \Rightarrow$ continua                 |
| 7          | 7   | 33     | $a[i] \neq m \Rightarrow$ continua                 |
| —          | 8   | —      | $i > n \Rightarrow$ il ciclo termina, non trovato! |

Quindi, l'elemento 15 non è presente nell'array (restituisce  $-1$ ).

5. Ricerca di  $m = 3$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 6      | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 19     | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 3      | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 3 si trova all'indice 3.

#### Esercizio 4.4 — Ricerca lineare e ricerca binaria (E4)

Consideriamo il seguente array:

$$a = [50, 25, 75, 10, 40, 60, 5]$$

con  $n = 7$ .

**Domande**

1. Applicare l'algoritmo di ricerca lineare (Linear Search) all'array  $a$  per ciascuno dei seguenti valori cercati:

- (a)  $m = 10$
- (b)  $m = 50$
- (c)  $m = 5$
- (d)  $m = 30$
- (e)  $m = 75$

Per ogni ricerca, mostrare l'esecuzione passo per passo, indicando a ogni iterazione i valori di  $i$ ,  $a[i]$  e il risultato del confronto, e dare il valore restituito dall'algoritmo.

### Soluzione

1. Ricerca di  $m = 10$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 50     | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 25     | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 75     | $a[i] \neq m \Rightarrow$ continua |
| 4          | 4   | 10     | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 10 si trova all'indice 4.

2. Ricerca di  $m = 50$ :

| iterazione | $i$ | $a[i]$ | azione                          |
|------------|-----|--------|---------------------------------|
| 1          | 1   | 50     | $a[i] = m \Rightarrow$ Trovato! |

Quindi, l'elemento 50 si trova all'indice 1.

3. Ricerca di  $m = 5$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 50     | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 25     | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 75     | $a[i] \neq m \Rightarrow$ continua |
| 4          | 4   | 10     | $a[i] \neq m \Rightarrow$ continua |
| 5          | 5   | 40     | $a[i] \neq m \Rightarrow$ continua |
| 6          | 6   | 60     | $a[i] \neq m \Rightarrow$ continua |
| 7          | 7   | 5      | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 5 si trova all'indice 7.

4. Ricerca di  $m = 30$ :

| iterazione | $i$ | $a[i]$ | azione                                             |
|------------|-----|--------|----------------------------------------------------|
| 1          | 1   | 50     | $a[i] \neq m \Rightarrow$ continua                 |
| 2          | 2   | 25     | $a[i] \neq m \Rightarrow$ continua                 |
| 3          | 3   | 75     | $a[i] \neq m \Rightarrow$ continua                 |
| 4          | 4   | 10     | $a[i] \neq m \Rightarrow$ continua                 |
| 5          | 5   | 40     | $a[i] \neq m \Rightarrow$ continua                 |
| 6          | 6   | 60     | $a[i] \neq m \Rightarrow$ continua                 |
| 7          | 7   | 5      | $a[i] \neq m \Rightarrow$ continua                 |
| —          | 8   | —      | $i > n \Rightarrow$ il ciclo termina, non trovato! |

Quindi, l'elemento 30 non è presente nell'array (restituisce  $-1$ ).

5. Ricerca di  $m = 75$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 50     | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 25     | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 75     | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 75 si trova all'indice 3.

#### Esercizio 4.5 — Ricerca lineare e ricerca binaria (E5)

Consideriamo il seguente array:

$$a = [11, 7, 3, 18, 24, 9, 16, 2]$$

con  $n = 8$ .

##### Domande

1. Applicare l'algoritmo di ricerca lineare (Linear Search) all'array  $a$  per ciascuno dei seguenti valori cercati:

- (a)  $m = 24$
- (b)  $m = 11$
- (c)  $m = 2$
- (d)  $m = 8$
- (e)  $m = 3$

Per ogni ricerca, mostrare l'esecuzione passo per passo, indicando a ogni iterazione i valori di  $i$ ,  $a[i]$  e il risultato del confronto, e dare il valore restituito dall'algoritmo.

## Soluzione

1. Ricerca di  $m = 24$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 11     | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 7      | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 3      | $a[i] \neq m \Rightarrow$ continua |
| 4          | 4   | 18     | $a[i] \neq m \Rightarrow$ continua |
| 5          | 5   | 24     | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 24 si trova all'indice 5.

2. Ricerca di  $m = 11$ :

| iterazione | $i$ | $a[i]$ | azione                          |
|------------|-----|--------|---------------------------------|
| 1          | 1   | 11     | $a[i] = m \Rightarrow$ Trovato! |

Quindi, l'elemento 11 si trova all'indice 1.

3. Ricerca di  $m = 2$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 11     | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 7      | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 3      | $a[i] \neq m \Rightarrow$ continua |
| 4          | 4   | 18     | $a[i] \neq m \Rightarrow$ continua |
| 5          | 5   | 24     | $a[i] \neq m \Rightarrow$ continua |
| 6          | 6   | 9      | $a[i] \neq m \Rightarrow$ continua |
| 7          | 7   | 16     | $a[i] \neq m \Rightarrow$ continua |
| 8          | 8   | 2      | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 2 si trova all'indice 8.

4. Ricerca di  $m = 8$ :

| iterazione | $i$ | $a[i]$ | azione                                             |
|------------|-----|--------|----------------------------------------------------|
| 1          | 1   | 11     | $a[i] \neq m \Rightarrow$ continua                 |
| 2          | 2   | 7      | $a[i] \neq m \Rightarrow$ continua                 |
| 3          | 3   | 3      | $a[i] \neq m \Rightarrow$ continua                 |
| 4          | 4   | 18     | $a[i] \neq m \Rightarrow$ continua                 |
| 5          | 5   | 24     | $a[i] \neq m \Rightarrow$ continua                 |
| 6          | 6   | 9      | $a[i] \neq m \Rightarrow$ continua                 |
| 7          | 7   | 16     | $a[i] \neq m \Rightarrow$ continua                 |
| 8          | 8   | 2      | $a[i] \neq m \Rightarrow$ continua                 |
| —          | 9   | —      | $i > n \Rightarrow$ il ciclo termina, non trovato! |

Quindi, l'elemento 8 non è presente nell'array (restituisce  $-1$ ).

5. Ricerca di  $m = 3$ :

| iterazione | $i$ | $a[i]$ | azione                             |
|------------|-----|--------|------------------------------------|
| 1          | 1   | 11     | $a[i] \neq m \Rightarrow$ continua |
| 2          | 2   | 7      | $a[i] \neq m \Rightarrow$ continua |
| 3          | 3   | 3      | $a[i] = m \Rightarrow$ Trovato!    |

Quindi, l'elemento 3 si trova all'indice 3.

#### Esercizio 4.6 — Ricerca lineare e ricerca binaria (E6)

Consideriamo il seguente array, ordinato in modo non decrescente:

$$a = [2, 5, 8, 12, 16, 23, 38, 45, 56, 67, 78, 89]$$

con  $n = 12$ .

##### Domande

1. Applicare l'algoritmo di ricerca binaria (Binary Search) all'array  $a$  per ciascuno dei seguenti valori cercati:

- (a)  $m = 23$
- (b)  $m = 2$
- (c)  $m = 89$
- (d)  $m = 50$
- (e)  $m = 1$

Per ogni ricerca, mostrare l'esecuzione passo per passo, indicando a ogni iterazione i valori di  $left$ ,  $right$ ,  $mid$ ,  $a[mid]$  e il risultato del confronto, e dare il valore restituito dall'algoritmo.

##### Soluzione

1. Ricerca di  $m = 23$ :

| iterazione | $left$ | $right$ | $mid$ | $a[mid]$ | azione                            |
|------------|--------|---------|-------|----------|-----------------------------------|
| 1          | 1      | 12      | 6     | 23       | $a[mid] = m \Rightarrow$ Trovato! |

Quindi, l'elemento 23 si trova all'indice 6.

2. Ricerca di  $m = 2$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                      |
|------------|-------------|--------------|------------|----------|---------------------------------------------|
| 1          | 1           | 12           | 6          | 23       | $a[mid] > m \Rightarrow right \leftarrow 5$ |
| 2          | 1           | 5            | 3          | 8        | $a[mid] > m \Rightarrow right \leftarrow 2$ |
| 3          | 1           | 2            | 1          | 2        | $a[mid] = m \Rightarrow$ Trovato!           |

Quindi, l'elemento 2 si trova all'indice 1.

3. Ricerca di  $m = 89$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                      |
|------------|-------------|--------------|------------|----------|---------------------------------------------|
| 1          | 1           | 12           | 6          | 23       | $a[mid] < m \Rightarrow left \leftarrow 7$  |
| 2          | 7           | 12           | 9          | 56       | $a[mid] < m \Rightarrow left \leftarrow 10$ |
| 3          | 10          | 12           | 11         | 78       | $a[mid] < m \Rightarrow left \leftarrow 12$ |
| 4          | 12          | 12           | 12         | 89       | $a[mid] = m \Rightarrow$ Trovato!           |

Quindi, l'elemento 89 si trova all'indice 12.

4. Ricerca di  $m = 50$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                      |
|------------|-------------|--------------|------------|----------|---------------------------------------------|
| 1          | 1           | 12           | 6          | 23       | $a[mid] < m \Rightarrow left \leftarrow 7$  |
| 2          | 7           | 12           | 9          | 56       | $a[mid] > m \Rightarrow right \leftarrow 8$ |
| 3          | 7           | 8            | 7          | 38       | $a[mid] < m \Rightarrow left \leftarrow 8$  |
| 4          | 8           | 8            | 8          | 45       | $a[mid] < m \Rightarrow left \leftarrow 9$  |
| —          | 9           | 8            | —          | —        | $left > right \Rightarrow$ Non trovato!     |

Quindi, l'elemento 50 non è presente nell'array (restituisce  $-1$ ).

5. Ricerca di  $m = 1$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                      |
|------------|-------------|--------------|------------|----------|---------------------------------------------|
| 1          | 1           | 12           | 6          | 23       | $a[mid] > m \Rightarrow right \leftarrow 5$ |
| 2          | 1           | 5            | 3          | 8        | $a[mid] > m \Rightarrow right \leftarrow 2$ |
| 3          | 1           | 2            | 1          | 2        | $a[mid] > m \Rightarrow right \leftarrow 0$ |
| —          | 1           | 0            | —          | —        | $left > right \Rightarrow$ Non trovato!     |

Quindi, l'elemento 1 non è presente nell'array (restituisce  $-1$ ).

**Esercizio 4.7 — Ricerca lineare e ricerca binaria (E7)**

Consideriamo il seguente array, ordinato in modo non decrescente:

$$a = [1, 3, 5, 7, 9, 11, 13, 15]$$

con  $n = 8$ .

**Domande**

1. Applicare l'algoritmo di ricerca binaria (Binary Search) all'array  $a$  per ciascuno dei seguenti valori cercati:

- (a)  $m = 7$
- (b)  $m = 1$
- (c)  $m = 15$
- (d)  $m = 10$
- (e)  $m = 20$

Per ogni ricerca, mostrare l'esecuzione passo per passo, indicando a ogni iterazione i valori di  $left$ ,  $right$ ,  $mid$ ,  $a[mid]$  e il risultato del confronto, e dare il valore restituito dall'algoritmo.

**Soluzione**

1. Ricerca di  $m = 7$ :

| iterazione | $left$ | $right$ | $mid$ | $a[mid]$ | azione                            |
|------------|--------|---------|-------|----------|-----------------------------------|
| 1          | 1      | 8       | 4     | 7        | $a[mid] = m \Rightarrow$ Trovato! |

Quindi, l'elemento 7 si trova all'indice 4.

2. Ricerca di  $m = 1$ :

| iterazione | $left$ | $right$ | $mid$ | $a[mid]$ | azione                                      |
|------------|--------|---------|-------|----------|---------------------------------------------|
| 1          | 1      | 8       | 4     | 7        | $a[mid] > m \Rightarrow right \leftarrow 3$ |
| 2          | 1      | 3       | 2     | 3        | $a[mid] > m \Rightarrow right \leftarrow 1$ |
| 3          | 1      | 1       | 1     | 1        | $a[mid] = m \Rightarrow$ Trovato!           |

Quindi, l'elemento 1 si trova all'indice 1.

3. Ricerca di  $m = 15$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                     |
|------------|-------------|--------------|------------|----------|--------------------------------------------|
| 1          | 1           | 8            | 4          | 7        | $a[mid] < m \Rightarrow left \leftarrow 5$ |
| 2          | 5           | 8            | 6          | 11       | $a[mid] < m \Rightarrow left \leftarrow 7$ |
| 3          | 7           | 8            | 7          | 13       | $a[mid] < m \Rightarrow left \leftarrow 8$ |
| 4          | 8           | 8            | 8          | 15       | $a[mid] = m \Rightarrow$ Trovato!          |

Quindi, l'elemento 15 si trova all'indice 8.

4. Ricerca di  $m = 10$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                      |
|------------|-------------|--------------|------------|----------|---------------------------------------------|
| 1          | 1           | 8            | 4          | 7        | $a[mid] < m \Rightarrow left \leftarrow 5$  |
| 2          | 5           | 8            | 6          | 11       | $a[mid] > m \Rightarrow right \leftarrow 5$ |
| 3          | 5           | 5            | 5          | 9        | $a[mid] < m \Rightarrow left \leftarrow 6$  |
| —          | 6           | 5            | —          | —        | $left > right \Rightarrow$ Non trovato!     |

Quindi, l'elemento 10 non è presente nell'array (restituisce  $-1$ ).

5. Ricerca di  $m = 20$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                     |
|------------|-------------|--------------|------------|----------|--------------------------------------------|
| 1          | 1           | 8            | 4          | 7        | $a[mid] < m \Rightarrow left \leftarrow 5$ |
| 2          | 5           | 8            | 6          | 11       | $a[mid] < m \Rightarrow left \leftarrow 7$ |
| 3          | 7           | 8            | 7          | 13       | $a[mid] < m \Rightarrow left \leftarrow 8$ |
| 4          | 8           | 8            | 8          | 15       | $a[mid] < m \Rightarrow left \leftarrow 9$ |
| —          | 9           | 8            | —          | —        | $left > right \Rightarrow$ Non trovato!    |

Quindi, l'elemento 20 non è presente nell'array (restituisce  $-1$ ).

#### Esercizio 4.8 — Ricerca lineare e ricerca binaria (E8)

Consideriamo il seguente array, ordinato in modo non decrescente:

$$a = [10, 20, 30, 40, 50, 60]$$

con  $n = 6$ .

##### Domande

1. Applicare l'algoritmo di ricerca binaria (Binary Search) all'array  $a$  per ciascuno dei seguenti valori cercati:

- (a)  $m = 40$
- (b)  $m = 10$
- (c)  $m = 60$
- (d)  $m = 25$
- (e)  $m = 70$

Per ogni ricerca, mostrare l'esecuzione passo per passo, indicando a ogni iterazione i valori di  $left$ ,  $right$ ,  $mid$ ,  $a[mid]$  e il risultato del confronto, e dare il valore restituito dall'algoritmo.

### Soluzione

1. Ricerca di  $m = 40$ :

| iterazione | $left$ | $right$ | $mid$ | $a[mid]$ | azione                                      |
|------------|--------|---------|-------|----------|---------------------------------------------|
| 1          | 1      | 6       | 3     | 30       | $a[mid] < m \Rightarrow left \leftarrow 4$  |
| 2          | 4      | 6       | 5     | 50       | $a[mid] > m \Rightarrow right \leftarrow 4$ |
| 3          | 4      | 4       | 4     | 40       | $a[mid] = m \Rightarrow$ Trovato!           |

Quindi, l'elemento 40 si trova all'indice 4.

2. Ricerca di  $m = 10$ :

| iterazione | $left$ | $right$ | $mid$ | $a[mid]$ | azione                                      |
|------------|--------|---------|-------|----------|---------------------------------------------|
| 1          | 1      | 6       | 3     | 30       | $a[mid] > m \Rightarrow right \leftarrow 2$ |
| 2          | 1      | 2       | 1     | 10       | $a[mid] = m \Rightarrow$ Trovato!           |

Quindi, l'elemento 10 si trova all'indice 1.

3. Ricerca di  $m = 60$ :

| iterazione | $left$ | $right$ | $mid$ | $a[mid]$ | azione                                     |
|------------|--------|---------|-------|----------|--------------------------------------------|
| 1          | 1      | 6       | 3     | 30       | $a[mid] < m \Rightarrow left \leftarrow 4$ |
| 2          | 4      | 6       | 5     | 50       | $a[mid] < m \Rightarrow left \leftarrow 6$ |
| 3          | 6      | 6       | 6     | 60       | $a[mid] = m \Rightarrow$ Trovato!          |

Quindi, l'elemento 60 si trova all'indice 6.

4. Ricerca di  $m = 25$ :

| iterazione | $left$ | $right$ | $mid$ | $a[mid]$ | azione                                      |
|------------|--------|---------|-------|----------|---------------------------------------------|
| 1          | 1      | 6       | 3     | 30       | $a[mid] > m \Rightarrow right \leftarrow 2$ |
| 2          | 1      | 2       | 1     | 10       | $a[mid] < m \Rightarrow left \leftarrow 2$  |
| 3          | 2      | 2       | 2     | 20       | $a[mid] < m \Rightarrow left \leftarrow 3$  |
| —          | 3      | 2       | —     | —        | $left > right \Rightarrow$ Non trovato!     |

Quindi, l'elemento 25 non è presente nell'array (restituisce  $-1$ ).

5. Ricerca di  $m = 70$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                     |
|------------|-------------|--------------|------------|----------|--------------------------------------------|
| 1          | 1           | 6            | 3          | 30       | $a[mid] < m \Rightarrow left \leftarrow 4$ |
| 2          | 4           | 6            | 5          | 50       | $a[mid] < m \Rightarrow left \leftarrow 6$ |
| 3          | 6           | 6            | 6          | 60       | $a[mid] < m \Rightarrow left \leftarrow 7$ |
| —          | 7           | 6            | —          | —        | $left > right \Rightarrow$ Non trovato!    |

Quindi, l'elemento 70 non è presente nell'array (restituisce  $-1$ ).

#### Esercizio 4.9 — Ricerca lineare e ricerca binaria (E9)

Consideriamo il seguente array, ordinato in modo non decrescente:

$$a = [4, 8, 15, 16, 23, 42]$$

con  $n = 6$ .

##### Domande

1. Applicare l'algoritmo di ricerca binaria (Binary Search) all'array  $a$  per ciascuno dei seguenti valori cercati:

- (a)  $m = 15$
- (b)  $m = 4$
- (c)  $m = 42$
- (d)  $m = 20$
- (e)  $m = 2$

Per ogni ricerca, mostrare l'esecuzione passo per passo, indicando a ogni iterazione i valori di  $left$ ,  $right$ ,  $mid$ ,  $a[mid]$  e il risultato del confronto, e dare il valore restituito dall'algoritmo.

##### Soluzione

1. Ricerca di  $m = 15$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                            |
|------------|-------------|--------------|------------|----------|-----------------------------------|
| 1          | 1           | 6            | 3          | 15       | $a[mid] = m \Rightarrow$ Trovato! |

Quindi, l'elemento 15 si trova all'indice 3.

2. Ricerca di  $m = 4$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                      |
|------------|-------------|--------------|------------|----------|---------------------------------------------|
| 1          | 1           | 6            | 3          | 15       | $a[mid] > m \Rightarrow right \leftarrow 2$ |
| 2          | 1           | 2            | 1          | 4        | $a[mid] = m \Rightarrow$ Trovato!           |

Quindi, l'elemento 4 si trova all'indice 1.

3. Ricerca di  $m = 42$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                     |
|------------|-------------|--------------|------------|----------|--------------------------------------------|
| 1          | 1           | 6            | 3          | 15       | $a[mid] < m \Rightarrow left \leftarrow 4$ |
| 2          | 4           | 6            | 5          | 23       | $a[mid] < m \Rightarrow left \leftarrow 6$ |
| 3          | 6           | 6            | 6          | 42       | $a[mid] = m \Rightarrow$ Trovato!          |

Quindi, l'elemento 42 si trova all'indice 6.

4. Ricerca di  $m = 20$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                      |
|------------|-------------|--------------|------------|----------|---------------------------------------------|
| 1          | 1           | 6            | 3          | 15       | $a[mid] < m \Rightarrow left \leftarrow 4$  |
| 2          | 4           | 6            | 5          | 23       | $a[mid] > m \Rightarrow right \leftarrow 4$ |
| 3          | 4           | 4            | 4          | 16       | $a[mid] < m \Rightarrow left \leftarrow 5$  |
| —          | 5           | 4            | —          | —        | $left > right \Rightarrow$ Non trovato!     |

Quindi, l'elemento 20 non è presente nell'array (restituisce  $-1$ ).

5. Ricerca di  $m = 2$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                      |
|------------|-------------|--------------|------------|----------|---------------------------------------------|
| 1          | 1           | 6            | 3          | 15       | $a[mid] > m \Rightarrow right \leftarrow 2$ |
| 2          | 1           | 2            | 1          | 4        | $a[mid] > m \Rightarrow right \leftarrow 0$ |
| —          | 1           | 0            | —          | —        | $left > right \Rightarrow$ Non trovato!     |

Quindi, l'elemento 2 non è presente nell'array (restituisce  $-1$ ).

#### Esercizio 4.10 — Ricerca lineare e ricerca binaria (E10)

Consideriamo il seguente array, ordinato in modo non decrescente:

$$a = [3, 6, 9, 12, 15, 18, 21, 24, 27, 30]$$

con  $n = 10$ .

### Domande

1. Applicare l'algoritmo di ricerca binaria (Binary Search) all'array  $a$  per ciascuno dei seguenti valori cercati:

- (a)  $m = 18$
- (b)  $m = 3$
- (c)  $m = 30$
- (d)  $m = 10$
- (e)  $m = 35$

Per ogni ricerca, mostrare l'esecuzione passo per passo, indicando a ogni iterazione i valori di  $left$ ,  $right$ ,  $mid$ ,  $a[mid]$  e il risultato del confronto, e dare il valore restituito dall'algoritmo.

### Soluzione

1. Ricerca di  $m = 18$ :

| iterazione | $left$ | $right$ | $mid$ | $a[mid]$ | azione                                      |
|------------|--------|---------|-------|----------|---------------------------------------------|
| 1          | 1      | 10      | 5     | 15       | $a[mid] < m \Rightarrow left \leftarrow 6$  |
| 2          | 6      | 10      | 8     | 24       | $a[mid] > m \Rightarrow right \leftarrow 7$ |
| 3          | 6      | 7       | 6     | 18       | $a[mid] = m \Rightarrow$ Trovato!           |

Quindi, l'elemento 18 si trova all'indice 6.

2. Ricerca di  $m = 3$ :

| iterazione | $left$ | $right$ | $mid$ | $a[mid]$ | azione                                      |
|------------|--------|---------|-------|----------|---------------------------------------------|
| 1          | 1      | 10      | 5     | 15       | $a[mid] > m \Rightarrow right \leftarrow 4$ |
| 2          | 1      | 4       | 2     | 6        | $a[mid] > m \Rightarrow right \leftarrow 1$ |
| 3          | 1      | 1       | 1     | 3        | $a[mid] = m \Rightarrow$ Trovato!           |

Quindi, l'elemento 3 si trova all'indice 1.

3. Ricerca di  $m = 30$ :

| iterazione | $left$ | $right$ | $mid$ | $a[mid]$ | azione                                      |
|------------|--------|---------|-------|----------|---------------------------------------------|
| 1          | 1      | 10      | 5     | 15       | $a[mid] < m \Rightarrow left \leftarrow 6$  |
| 2          | 6      | 10      | 8     | 24       | $a[mid] < m \Rightarrow left \leftarrow 9$  |
| 3          | 9      | 10      | 9     | 27       | $a[mid] < m \Rightarrow left \leftarrow 10$ |
| 4          | 10     | 10      | 10    | 30       | $a[mid] = m \Rightarrow$ Trovato!           |

Quindi, l'elemento 30 si trova all'indice 10.

4. Ricerca di  $m = 10$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                      |
|------------|-------------|--------------|------------|----------|---------------------------------------------|
| 1          | 1           | 10           | 5          | 15       | $a[mid] > m \Rightarrow right \leftarrow 4$ |
| 2          | 1           | 4            | 2          | 6        | $a[mid] < m \Rightarrow left \leftarrow 3$  |
| 3          | 3           | 4            | 3          | 9        | $a[mid] < m \Rightarrow left \leftarrow 4$  |
| 4          | 4           | 4            | 4          | 12       | $a[mid] > m \Rightarrow right \leftarrow 3$ |
| –          | 4           | 3            | –          | –        | $left > right \Rightarrow$ Non trovato!     |

Quindi, l'elemento 10 non è presente nell'array (restituisce  $-1$ ).

5. Ricerca di  $m = 35$ :

| iterazione | <i>left</i> | <i>right</i> | <i>mid</i> | $a[mid]$ | azione                                      |
|------------|-------------|--------------|------------|----------|---------------------------------------------|
| 1          | 1           | 10           | 5          | 15       | $a[mid] < m \Rightarrow left \leftarrow 6$  |
| 2          | 6           | 10           | 8          | 24       | $a[mid] < m \Rightarrow left \leftarrow 9$  |
| 3          | 9           | 10           | 9          | 27       | $a[mid] < m \Rightarrow left \leftarrow 10$ |
| 4          | 10          | 10           | 10         | 30       | $a[mid] < m \Rightarrow left \leftarrow 11$ |
| –          | 11          | 10           | –          | –        | $left > right \Rightarrow$ Non trovato!     |

Quindi, l'elemento 35 non è presente nell'array (restituisce  $-1$ ).

## Algoritmi numerici

Gli esercizi di questo capitolo applicano i quattro algoritmi del capitolo «Algoritmi numerici»: `ChangeOfBase`, `BasicPrimalityTest` e `PrimalityTest`, `SieveOfEratosthenes`, `EuclideanAlgorithm`. Nelle soluzioni si usano i seguenti fatti del capitolo.

**Divisione con resto.** Per ogni numero naturale  $n$  e ogni numero naturale  $b \geq 1$ :

$$n = \lfloor n/b \rfloor b + (n \bmod b) \quad \text{con} \quad 0 \leq n \bmod b < b.$$

**Cambio di base.** Se  $n = (d_{k-1} \dots d_1 d_0)_b$ , allora  $n \bmod b = d_0$  e  $\lfloor n/b \rfloor = (d_{k-1} \dots d_1)_b$ ; il numero di cifre è  $\lfloor \log_b n \rfloor + 1$ . `ChangeOfBase` memorizza la cifra  $d_{k-1}$  nell'elemento  $d[k]$ .

### Attenzione

**Primalità.** Un numero naturale  $n \geq 2$  è primo se e solo se nessun numero naturale in  $\{2, 3, \dots, \lfloor \sqrt{n} \rfloor\}$  è un divisore di  $n$ .

**Crivello.** La marcatura dei multipli di un primo  $p$  parte da  $p^2$ , e il ciclo esterno si ferma a  $\lfloor \sqrt{n} \rfloor$ . Nelle tabelle di esecuzione le celle blu sono gli elementi appena posti a 0 e le celle rosse il valore corrente di  $p$ .

**Massimo comun divisore.** Per ogni coppia di numeri naturali  $n, m \geq 1$ :

$$\gcd(n, m) = \gcd(m, n \bmod m) \quad \text{e} \quad \gcd(n, 0) = n.$$

Due numeri naturali sono coprimi se e solo se il loro massimo comun divisore è 1.

### Esercizio 5.1 — Algoritmi numerici (E1)

Consideriamo il seguente numero naturale, scritto come sequenza di cifre in base dieci:

$$(16)_{10}$$

Domande

1. Convertire il numero in base  $b = 2$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
2. Convertire il numero in base  $b = 3$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
3. Convertire il numero in base  $b = 4$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
4. Convertire il numero in base  $b = 5$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
5. Convertire il numero in base  $b = 6$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.

### Soluzione

1. Conversione di 16 in base  $b = 2$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 16  | 16/2  | 8   | 0   | $d[1] = d_0$ (meno significativa) |
| 2              | 8   | 8/2   | 4   | 0   | $d[2] = d_1$                      |
| 3              | 4   | 4/2   | 2   | 0   | $d[3] = d_2$                      |
| 4              | 2   | 2/2   | 1   | 0   | $d[4] = d_3$                      |
| 5              | 1   | 1/2   | 0   | 1   | $d[5] = d_4$ (più significativa)  |

Quindi 16 in base 2 è  $(10000)_2$ , con  $nd = \lfloor \log_2 16 \rfloor + 1 = 5$  cifre.

2. Conversione di 16 in base  $b = 3$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 16  | 16/3  | 5   | 1   | $d[1] = d_0$ (meno significativa) |
| 2              | 5   | 5/3   | 1   | 2   | $d[2] = d_1$                      |
| 3              | 1   | 1/3   | 0   | 1   | $d[3] = d_2$ (più significativa)  |

Quindi 16 in base 3 è  $(121)_3$ , con  $nd = \lfloor \log_3 16 \rfloor + 1 = 3$  cifre.

3. Conversione di 16 in base  $b = 4$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 16  | 16/4  | 4   | 0   | $d[1] = d_0$ (meno significativa) |
| 2              | 4   | 4/4   | 1   | 0   | $d[2] = d_1$                      |
| 3              | 1   | 1/4   | 0   | 1   | $d[3] = d_2$ (più significativa)  |

Quindi 16 in base 4 è  $(100)_4$ , con  $nd = \lfloor \log_4 16 \rfloor + 1 = 3$  cifre.

4. Conversione di 16 in base  $b = 5$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 16  | 16/5  | 3   | 1   | $d[1] = d_0$ (meno significativa) |
| 2              | 3   | 3/5   | 0   | 3   | $d[2] = d_1$ (più significativa)  |

Quindi 16 in base 5 è  $(31)_5$ , con  $nd = \lfloor \log_5 16 \rfloor + 1 = 2$  cifre.

5. Conversione di 16 in base  $b = 6$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 16  | 16/6  | 2   | 4   | $d[1] = d_0$ (meno significativa) |
| 2              | 2   | 2/6   | 0   | 2   | $d[2] = d_1$ (più significativa)  |

Quindi 16 in base 6 è  $(24)_6$ , con  $nd = \lfloor \log_6 16 \rfloor + 1 = 2$  cifre.

### Esercizio 5.2 — Algoritmi numerici (E2)

Consideriamo il seguente numero naturale, scritto come sequenza di cifre in base dieci:

$$(25)_{10}$$

#### Domande

1. Convertire il numero in base  $b = 2$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
2. Convertire il numero in base  $b = 3$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
3. Convertire il numero in base  $b = 4$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
4. Convertire il numero in base  $b = 5$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
5. Convertire il numero in base  $b = 7$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.

## Soluzione

1. Conversione di 25 in base
- $b = 2$
- :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 25  | 25/2  | 12  | 1   | $d[1] = d_0$ (meno significativa) |
| 2              | 12  | 12/2  | 6   | 0   | $d[2] = d_1$                      |
| 3              | 6   | 6/2   | 3   | 0   | $d[3] = d_2$                      |
| 4              | 3   | 3/2   | 1   | 1   | $d[4] = d_3$                      |
| 5              | 1   | 1/2   | 0   | 1   | $d[5] = d_4$ (più significativa)  |

Quindi 25 in base 2 è  $(11001)_2$ , con  $nd = \lfloor \log_2 25 \rfloor + 1 = 5$  cifre.

2. Conversione di 25 in base
- $b = 3$
- :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 25  | 25/3  | 8   | 1   | $d[1] = d_0$ (meno significativa) |
| 2              | 8   | 8/3   | 2   | 2   | $d[2] = d_1$                      |
| 3              | 2   | 2/3   | 0   | 2   | $d[3] = d_2$ (più significativa)  |

Quindi 25 in base 3 è  $(221)_3$ , con  $nd = \lfloor \log_3 25 \rfloor + 1 = 3$  cifre.

3. Conversione di 25 in base
- $b = 4$
- :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 25  | 25/4  | 6   | 1   | $d[1] = d_0$ (meno significativa) |
| 2              | 6   | 6/4   | 1   | 2   | $d[2] = d_1$                      |
| 3              | 1   | 1/4   | 0   | 1   | $d[3] = d_2$ (più significativa)  |

Quindi 25 in base 4 è  $(121)_4$ , con  $nd = \lfloor \log_4 25 \rfloor + 1 = 3$  cifre.

4. Conversione di 25 in base
- $b = 5$
- :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 25  | 25/5  | 5   | 0   | $d[1] = d_0$ (meno significativa) |
| 2              | 5   | 5/5   | 1   | 0   | $d[2] = d_1$                      |
| 3              | 1   | 1/5   | 0   | 1   | $d[3] = d_2$ (più significativa)  |

Quindi 25 in base 5 è  $(100)_5$ , con  $nd = \lfloor \log_5 25 \rfloor + 1 = 3$  cifre.

5. Conversione di 25 in base
- $b = 7$
- :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 25  | 25/7  | 3   | 4   | $d[1] = d_0$ (meno significativa) |
| 2              | 3   | 3/7   | 0   | 3   | $d[2] = d_1$ (più significativa)  |

Quindi 25 in base 7 è  $(34)_7$ , con  $nd = \lfloor \log_7 25 \rfloor + 1 = 2$  cifre.

**Esercizio 5.3 — Algoritmi numerici (E3)**

Consideriamo il seguente numero naturale, scritto come sequenza di cifre in base dieci:

$$(42)_{10}$$

**Domande**

1. Convertire il numero in base  $b = 2$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
2. Convertire il numero in base  $b = 3$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
3. Convertire il numero in base  $b = 5$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
4. Convertire il numero in base  $b = 6$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
5. Convertire il numero in base  $b = 8$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.

**Soluzione**

1. Conversione di 42 in base  $b = 2$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 42  | 42/2  | 21  | 0   | $d[1] = d_0$ (meno significativa) |
| 2              | 21  | 21/2  | 10  | 1   | $d[2] = d_1$                      |
| 3              | 10  | 10/2  | 5   | 0   | $d[3] = d_2$                      |
| 4              | 5   | 5/2   | 2   | 1   | $d[4] = d_3$                      |
| 5              | 2   | 2/2   | 1   | 0   | $d[5] = d_4$                      |
| 6              | 1   | 1/2   | 0   | 1   | $d[6] = d_5$ (più significativa)  |

Quindi 42 in base 2 è  $(101010)_2$ , con  $nd = \lfloor \log_2 42 \rfloor + 1 = 6$  cifre.

2. Conversione di 42 in base  $b = 3$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 42  | 42/3  | 14  | 0   | $d[1] = d_0$ (meno significativa) |
| 2              | 14  | 14/3  | 4   | 2   | $d[2] = d_1$                      |
| 3              | 4   | 4/3   | 1   | 1   | $d[3] = d_2$                      |
| 4              | 1   | 1/3   | 0   | 1   | $d[4] = d_3$ (più significativa)  |

Quindi 42 in base 3 è  $(1120)_3$ , con  $nd = \lfloor \log_3 42 \rfloor + 1 = 4$  cifre.

3. Conversione di 42 in base  $b = 5$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 42  | 42/5  | 8   | 2   | $d[1] = d_0$ (meno significativa) |
| 2              | 8   | 8/5   | 1   | 3   | $d[2] = d_1$                      |
| 3              | 1   | 1/5   | 0   | 1   | $d[3] = d_2$ (più significativa)  |

Quindi 42 in base 5 è  $(132)_5$ , con  $nd = \lfloor \log_5 42 \rfloor + 1 = 3$  cifre.

4. Conversione di 42 in base  $b = 6$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 42  | 42/6  | 7   | 0   | $d[1] = d_0$ (meno significativa) |
| 2              | 7   | 7/6   | 1   | 1   | $d[2] = d_1$                      |
| 3              | 1   | 1/6   | 0   | 1   | $d[3] = d_2$ (più significativa)  |

Quindi 42 in base 6 è  $(110)_6$ , con  $nd = \lfloor \log_6 42 \rfloor + 1 = 3$  cifre.

5. Conversione di 42 in base  $b = 8$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 42  | 42/8  | 5   | 2   | $d[1] = d_0$ (meno significativa) |
| 2              | 5   | 5/8   | 0   | 5   | $d[2] = d_1$ (più significativa)  |

Quindi 42 in base 8 è  $(52)_8$ , con  $nd = \lfloor \log_8 42 \rfloor + 1 = 2$  cifre.

#### Esercizio 5.4 — Algoritmi numerici (E4)

Consideriamo il seguente numero naturale, scritto come sequenza di cifre in base dieci:

$$(63)_{10}$$

#### Domande

1. Convertire il numero in base  $b = 2$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
2. Convertire il numero in base  $b = 4$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.

3. Convertire il numero in base  $b = 7$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
4. Convertire il numero in base  $b = 8$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
5. Convertire il numero in base  $b = 9$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.

### Soluzione

1. Conversione di 63 in base  $b = 2$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 63  | 63/2  | 31  | 1   | $d[1] = d_0$ (meno significativa) |
| 2              | 31  | 31/2  | 15  | 1   | $d[2] = d_1$                      |
| 3              | 15  | 15/2  | 7   | 1   | $d[3] = d_2$                      |
| 4              | 7   | 7/2   | 3   | 1   | $d[4] = d_3$                      |
| 5              | 3   | 3/2   | 1   | 1   | $d[5] = d_4$                      |
| 6              | 1   | 1/2   | 0   | 1   | $d[6] = d_5$ (più significativa)  |

Quindi 63 in base 2 è  $(111111)_2$ , con  $nd = \lfloor \log_2 63 \rfloor + 1 = 6$  cifre.

2. Conversione di 63 in base  $b = 4$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 63  | 63/4  | 15  | 3   | $d[1] = d_0$ (meno significativa) |
| 2              | 15  | 15/4  | 3   | 3   | $d[2] = d_1$                      |
| 3              | 3   | 3/4   | 0   | 3   | $d[3] = d_2$ (più significativa)  |

Quindi 63 in base 4 è  $(333)_4$ , con  $nd = \lfloor \log_4 63 \rfloor + 1 = 3$  cifre.

3. Conversione di 63 in base  $b = 7$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 63  | 63/7  | 9   | 0   | $d[1] = d_0$ (meno significativa) |
| 2              | 9   | 9/7   | 1   | 2   | $d[2] = d_1$                      |
| 3              | 1   | 1/7   | 0   | 1   | $d[3] = d_2$ (più significativa)  |

Quindi 63 in base 7 è  $(120)_7$ , con  $nd = \lfloor \log_7 63 \rfloor + 1 = 3$  cifre.

4. Conversione di 63 in base  $b = 8$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 63  | 63/8  | 7   | 7   | $d[1] = d_0$ (meno significativa) |
| 2              | 7   | 7/8   | 0   | 7   | $d[2] = d_1$ (più significativa)  |

Quindi 63 in base 8 è  $(77)_8$ , con  $nd = \lfloor \log_8 63 \rfloor + 1 = 2$  cifre.

5. Conversione di 63 in base  $b = 9$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 63  | 63/9  | 7   | 0   | $d[1] = d_0$ (meno significativa) |
| 2              | 7   | 7/9   | 0   | 7   | $d[2] = d_1$ (più significativa)  |

Quindi 63 in base 9 è  $(70)_9$ , con  $nd = \lfloor \log_9 63 \rfloor + 1 = 2$  cifre.

### Esercizio 5.5 — Algoritmi numerici (E5)

Consideriamo il seguente numero naturale, scritto come sequenza di cifre in base dieci:

$$(100)_{10}$$

#### Domande

1. Convertire il numero in base  $b = 2$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
2. Convertire il numero in base  $b = 3$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
3. Convertire il numero in base  $b = 5$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
4. Convertire il numero in base  $b = 8$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.
5. Convertire il numero in base  $b = 16$  con l'algoritmo `ChangeOfBase`, mostrando il quoziente e il resto di ogni divisione.

#### Soluzione

1. Conversione di 100 in base  $b = 2$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 100 | 100/2 | 50  | 0   | $d[1] = d_0$ (meno significativa) |
| 2              | 50  | 50/2  | 25  | 0   | $d[2] = d_1$                      |
| 3              | 25  | 25/2  | 12  | 1   | $d[3] = d_2$                      |
| 4              | 12  | 12/2  | 6   | 0   | $d[4] = d_3$                      |
| 5              | 6   | 6/2   | 3   | 0   | $d[5] = d_4$                      |
| 6              | 3   | 3/2   | 1   | 1   | $d[6] = d_5$                      |
| 7              | 1   | 1/2   | 0   | 1   | $d[7] = d_6$ (più significativa)  |

Quindi 100 in base 2 è  $(1100100)_2$ , con  $nd = \lfloor \log_2 100 \rfloor + 1 = 7$  cifre.

2. Conversione di 100 in base  $b = 3$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 100 | 100/3 | 33  | 1   | $d[1] = d_0$ (meno significativa) |
| 2              | 33  | 33/3  | 11  | 0   | $d[2] = d_1$                      |
| 3              | 11  | 11/3  | 3   | 2   | $d[3] = d_2$                      |
| 4              | 3   | 3/3   | 1   | 0   | $d[4] = d_3$                      |
| 5              | 1   | 1/3   | 0   | 1   | $d[5] = d_4$ (più significativa)  |

Quindi 100 in base 3 è  $(10201)_3$ , con  $nd = \lfloor \log_3 100 \rfloor + 1 = 5$  cifre.

3. Conversione di 100 in base  $b = 5$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 100 | 100/5 | 20  | 0   | $d[1] = d_0$ (meno significativa) |
| 2              | 20  | 20/5  | 4   | 0   | $d[2] = d_1$                      |
| 3              | 4   | 4/5   | 0   | 4   | $d[3] = d_2$ (più significativa)  |

Quindi 100 in base 5 è  $(400)_5$ , con  $nd = \lfloor \log_5 100 \rfloor + 1 = 3$  cifre.

4. Conversione di 100 in base  $b = 8$ :

| iterazione $k$ | $n$ | $n/b$ | $q$ | $r$ | cifra                             |
|----------------|-----|-------|-----|-----|-----------------------------------|
| 1              | 100 | 100/8 | 12  | 4   | $d[1] = d_0$ (meno significativa) |
| 2              | 12  | 12/8  | 1   | 4   | $d[2] = d_1$                      |
| 3              | 1   | 1/8   | 0   | 1   | $d[3] = d_2$ (più significativa)  |

Quindi 100 in base 8 è  $(144)_8$ , con  $nd = \lfloor \log_8 100 \rfloor + 1 = 3$  cifre.

5. Conversione di 100 in base  $b = 16$ :

| iterazione $k$ | $n$ | $n/b$  | $q$ | $r$ | cifra                             |
|----------------|-----|--------|-----|-----|-----------------------------------|
| 1              | 100 | 100/16 | 6   | 4   | $d[1] = d_0$ (meno significativa) |
| 2              | 6   | 6/16   | 0   | 6   | $d[2] = d_1$ (più significativa)  |

Quindi 100 in base 16 è  $(64)_{16}$ , con  $nd = \lfloor \log_{16} 100 \rfloor + 1 = 2$  cifre.

### Esercizio 5.6 — Algoritmi numerici (E6)

Consideriamo i seguenti numeri naturali:

$$n \in \{2, 7, 9, 11, 25\}$$

#### Domande

1. Applicare l'algoritmo PrimalityTest a ciascun numero della lista. Per ogni  $n$  mostrare l'esecuzione passo per passo, indicando il valore di  $\lfloor \sqrt{n} \rfloor$ , i valori di  $d$  provati e il risultato di ciascun test  $n \bmod d = 0$ .

#### Soluzione

1. Esecuzione di PrimalityTest su ciascun numero.

**Verifica  $n = 2$ :**  $\lfloor \sqrt{2} \rfloor = 1$ .

|                                               |             |        |
|-----------------------------------------------|-------------|--------|
| $d$                                           | $2 \bmod d$ | azione |
| Il ciclo non viene eseguito, poiché $2 > 1$ . |             |        |
| Restituisce <b>true</b> : <b>2 è primo</b> .  |             |        |

**Verifica  $n = 7$ :**  $\lfloor \sqrt{7} \rfloor = 2$ .

|                                                                |             |          |
|----------------------------------------------------------------|-------------|----------|
| $d$                                                            | $7 \bmod d$ | azione   |
| 2                                                              | $1 \neq 0$  | continua |
| Il ciclo termina. Restituisce <b>true</b> : <b>7 è primo</b> . |             |          |

**Verifica  $n = 9$ :**  $\lfloor \sqrt{9} \rfloor = 3$ .

|                                           |             |                          |
|-------------------------------------------|-------------|--------------------------|
| $d$                                       | $9 \bmod d$ | azione                   |
| 2                                         | $1 \neq 0$  | continua                 |
| 3                                         | 0           | restituisce <b>false</b> |
| <b>9 non è primo</b> ( $9 = 3 \cdot 3$ ). |             |                          |

Verifica  $n = 11$ :  $\lfloor \sqrt{11} \rfloor = 3$ .

| $d$                                                             | $11 \bmod d$ | azione   |
|-----------------------------------------------------------------|--------------|----------|
| 2                                                               | $1 \neq 0$   | continua |
| 3                                                               | $2 \neq 0$   | continua |
| Il ciclo termina. Restituisce <b>true</b> : <b>11 è primo</b> . |              |          |

Verifica  $n = 25$ :  $\lfloor \sqrt{25} \rfloor = 5$ .

| $d$                                         | $25 \bmod d$ | azione                   |
|---------------------------------------------|--------------|--------------------------|
| 2                                           | $1 \neq 0$   | continua                 |
| 3                                           | $1 \neq 0$   | continua                 |
| 4                                           | $1 \neq 0$   | continua                 |
| 5                                           | 0            | restituisce <b>false</b> |
| <b>25 non è primo</b> ( $25 = 5 \cdot 5$ ). |              |                          |

### Esercizio 5.7 — Algoritmi numerici (E7)

Consideriamo i seguenti numeri naturali:

$$n \in \{4, 13, 17, 49, 97\}$$

#### Domande

1. Applicare l'algoritmo PrimalityTest a ciascun numero della lista. Per ogni  $n$  mostrare l'esecuzione passo per passo, indicando il valore di  $\lfloor \sqrt{n} \rfloor$ , i valori di  $d$  provati e il risultato di ciascun test  $n \bmod d = 0$ .

#### Soluzione

1. Esecuzione di PrimalityTest su ciascun numero.

Verifica  $n = 4$ :  $\lfloor \sqrt{4} \rfloor = 2$ .

| $d$                                       | $4 \bmod d$ | azione                   |
|-------------------------------------------|-------------|--------------------------|
| 2                                         | 0           | restituisce <b>false</b> |
| <b>4 non è primo</b> ( $4 = 2 \cdot 2$ ). |             |                          |

Verifica  $n = 13$ :  $\lfloor \sqrt{13} \rfloor = 3$ .

| $d$                                                             | $13 \bmod d$ | azione   |
|-----------------------------------------------------------------|--------------|----------|
| 2                                                               | $1 \neq 0$   | continua |
| 3                                                               | $1 \neq 0$   | continua |
| Il ciclo termina. Restituisce <b>true</b> : <b>13 è primo</b> . |              |          |

Verifica  $n = 17$ :  $\lfloor \sqrt{17} \rfloor = 4$ .

| $d$                                                             | $17 \bmod d$ | azione   |
|-----------------------------------------------------------------|--------------|----------|
| 2                                                               | $1 \neq 0$   | continua |
| 3                                                               | $2 \neq 0$   | continua |
| 4                                                               | $1 \neq 0$   | continua |
| Il ciclo termina. Restituisce <b>true</b> : <b>17 è primo</b> . |              |          |

Verifica  $n = 49$ :  $\lfloor \sqrt{49} \rfloor = 7$ .

| $d$                                         | $49 \bmod d$ | azione                   |
|---------------------------------------------|--------------|--------------------------|
| 2                                           | $1 \neq 0$   | continua                 |
| 3                                           | $1 \neq 0$   | continua                 |
| 4                                           | $1 \neq 0$   | continua                 |
| 5                                           | $4 \neq 0$   | continua                 |
| 6                                           | $1 \neq 0$   | continua                 |
| 7                                           | 0            | restituisce <b>false</b> |
| <b>49 non è primo</b> ( $49 = 7 \cdot 7$ ). |              |                          |

Verifica  $n = 97$ :  $\lfloor \sqrt{97} \rfloor = 9$ .

| $d$                                                             | $97 \bmod d$ | azione   |
|-----------------------------------------------------------------|--------------|----------|
| 2                                                               | $1 \neq 0$   | continua |
| 3                                                               | $1 \neq 0$   | continua |
| 4                                                               | $1 \neq 0$   | continua |
| 5                                                               | $2 \neq 0$   | continua |
| 6                                                               | $1 \neq 0$   | continua |
| 7                                                               | $6 \neq 0$   | continua |
| 8                                                               | $1 \neq 0$   | continua |
| 9                                                               | $7 \neq 0$   | continua |
| Il ciclo termina. Restituisce <b>true</b> : <b>97 è primo</b> . |              |          |

## Esercizio 5.8 — Algoritmi numerici (E8)

Consideriamo i seguenti numeri naturali:

$$n \in \{1, 3, 15, 23, 36\}$$

## Domande

1. Applicare l'algoritmo PrimalityTest a ciascun numero della lista. Per ogni  $n$  mostrare l'esecuzione passo per passo, indicando il valore di  $\lfloor \sqrt{n} \rfloor$ , i valori di  $d$  provati e il risultato di ciascun test  $n \bmod d = 0$ .

## Soluzione

1. Esecuzione di PrimalityTest su ciascun numero.

**Verifica  $n = 1$ :**  $n < 2$ .

| $d$                                                    | $1 \bmod d$ | azione |
|--------------------------------------------------------|-------------|--------|
| $n = 1 < 2$ : restituisce <b>false</b> immediatamente. |             |        |
| <b>1 non è primo.</b>                                  |             |        |

**Verifica  $n = 3$ :**  $\lfloor \sqrt{3} \rfloor = 1$ .

| $d$                                           | $3 \bmod d$ | azione |
|-----------------------------------------------|-------------|--------|
| Il ciclo non viene eseguito, poiché $2 > 1$ . |             |        |
| Restituisce <b>true</b> : <b>3 è primo.</b>   |             |        |

**Verifica  $n = 15$ :**  $\lfloor \sqrt{15} \rfloor = 3$ .

| $d$                                         | $15 \bmod d$ | azione                   |
|---------------------------------------------|--------------|--------------------------|
| 2                                           | $1 \neq 0$   | continua                 |
| 3                                           | 0            | restituisce <b>false</b> |
| <b>15 non è primo</b> ( $15 = 3 \cdot 5$ ). |              |                          |

**Verifica  $n = 23$ :**  $\lfloor \sqrt{23} \rfloor = 4$ .

| $d$                                                            | $23 \bmod d$ | azione   |
|----------------------------------------------------------------|--------------|----------|
| 2                                                              | $1 \neq 0$   | continua |
| 3                                                              | $2 \neq 0$   | continua |
| 4                                                              | $3 \neq 0$   | continua |
| Il ciclo termina. Restituisce <b>true</b> : <b>23 è primo.</b> |              |          |

Verifica  $n = 36$ :  $\lfloor \sqrt{36} \rfloor = 6$ .

| $d$                                          | $36 \bmod d$ | azione                   |
|----------------------------------------------|--------------|--------------------------|
| 2                                            | 0            | restituisce <b>false</b> |
| <b>36 non è primo</b> ( $36 = 2 \cdot 18$ ). |              |                          |

### Esercizio 5.9 — Algoritmi numerici (E9)

Consideriamo i numeri naturali  $n = 29$  e  $n = 49$ .

#### Domande

1. Per ciascun numero  $n$ , applicare sia l'algoritmo BasicPrimalityTest (ciclo fino a  $\lfloor n/2 \rfloor$ ) sia l'algoritmo PrimalityTest (ciclo fino a  $\lfloor \sqrt{n} \rfloor$ ). Mostrare l'esecuzione passo per passo di ciascun algoritmo.
2. Per ciascun numero, contare il numero di iterazioni eseguite da ciascun algoritmo e confrontarle.

#### Soluzione

1. Esecuzione passo per passo.

**Caso**  $n = 29$ .

*BasicPrimalityTest* ( $\lfloor 29/2 \rfloor = 14$ ):

| $d$ | $29 \bmod d$ | azione   |
|-----|--------------|----------|
| 2   | $1 \neq 0$   | continua |
| 3   | $2 \neq 0$   | continua |
| 4   | $1 \neq 0$   | continua |
| 5   | $4 \neq 0$   | continua |
| 6   | $5 \neq 0$   | continua |
| 7   | $1 \neq 0$   | continua |
| 8   | $5 \neq 0$   | continua |
| 9   | $2 \neq 0$   | continua |
| 10  | $9 \neq 0$   | continua |
| 11  | $7 \neq 0$   | continua |
| 12  | $5 \neq 0$   | continua |
| 13  | $3 \neq 0$   | continua |
| 14  | $1 \neq 0$   | continua |

Il ciclo termina. Restituisce **true**: **29 è primo**. (13 iterazioni)

*PrimalityTest* ( $\lfloor \sqrt{29} \rfloor = 5$ ):

| $d$                                                                            | $29 \bmod d$ | azione   |
|--------------------------------------------------------------------------------|--------------|----------|
| 2                                                                              | $1 \neq 0$   | continua |
| 3                                                                              | $2 \neq 0$   | continua |
| 4                                                                              | $1 \neq 0$   | continua |
| 5                                                                              | $4 \neq 0$   | continua |
| Il ciclo termina. Restituisce <b>true</b> : <b>29 è primo</b> . (4 iterazioni) |              |          |

**Caso**  $n = 49$ .

*BasicPrimalityTest* ( $\lfloor 49/2 \rfloor = 24$ ):

| $d$                                                        | $49 \bmod d$ | azione                   |
|------------------------------------------------------------|--------------|--------------------------|
| 2                                                          | $1 \neq 0$   | continua                 |
| 3                                                          | $1 \neq 0$   | continua                 |
| 4                                                          | $1 \neq 0$   | continua                 |
| 5                                                          | $4 \neq 0$   | continua                 |
| 6                                                          | $1 \neq 0$   | continua                 |
| 7                                                          | 0            | restituisce <b>false</b> |
| <b>49 non è primo</b> ( $49 = 7 \cdot 7$ ). (6 iterazioni) |              |                          |

*PrimalityTest* ( $\lfloor \sqrt{49} \rfloor = 7$ ):

| $d$                                                        | $49 \bmod d$ | azione                   |
|------------------------------------------------------------|--------------|--------------------------|
| 2                                                          | $1 \neq 0$   | continua                 |
| 3                                                          | $1 \neq 0$   | continua                 |
| 4                                                          | $1 \neq 0$   | continua                 |
| 5                                                          | $4 \neq 0$   | continua                 |
| 6                                                          | $1 \neq 0$   | continua                 |
| 7                                                          | 0            | restituisce <b>false</b> |
| <b>49 non è primo</b> ( $49 = 7 \cdot 7$ ). (6 iterazioni) |              |                          |

2. Confronto delle iterazioni:

| $n$ | $\lfloor n/2 \rfloor$ | $\lfloor \sqrt{n} \rfloor$ | BasicPrimalityTest | PrimalityTest |
|-----|-----------------------|----------------------------|--------------------|---------------|
| 29  | 14                    | 5                          | 13                 | 4             |
| 49  | 24                    | 7                          | 6                  | 6             |

Per  $n = 29$  (primo), *PrimalityTest* esegue solo 4 iterazioni invece di 13: entrambi gli algoritmi devono provare tutti i loro candidati, e *PrimalityTest* ne ha molti meno. Per  $n = 49$  (composto, con il più piccolo divisore  $7 = \sqrt{49}$ ), entrambi gli algoritmi si fermano allo stesso divisore  $d = 7$ , quindi il numero di iterazioni è lo stesso.

## Esercizio 5.10 — Algoritmi numerici (E10)

Consideriamo i numeri naturali  $n \in \{2, 3, 4, \dots, 30\}$ .

## Domande

1. Applicare l'algoritmo `PrimalityTest` a ciascun numero naturale  $n \in \{2, 3, \dots, 30\}$  e determinare quali numeri sono primi. Per ogni  $n$  riportare il valore di  $\lfloor \sqrt{n} \rfloor$ , se è stato trovato un divisore (e quale) e il risultato.
2. Elencare tutti i numeri primi in  $\{2, 3, \dots, 30\}$  e contarli.

## Soluzione

1. Riepilogo delle esecuzioni di `PrimalityTest` (il divisore riportato è il primo trovato, cioè il valore di  $d$  per cui la funzione restituisce `false`):

| $n$ | $\lfloor \sqrt{n} \rfloor$ | primo divisore trovato       | risultato                       |
|-----|----------------------------|------------------------------|---------------------------------|
| 2   | 1                          | nessuno (ciclo non eseguito) | <b>primo</b>                    |
| 3   | 1                          | nessuno (ciclo non eseguito) | <b>primo</b>                    |
| 4   | 2                          | $d = 2$                      | non primo ( $4 = 2 \cdot 2$ )   |
| 5   | 2                          | nessuno                      | <b>primo</b>                    |
| 6   | 2                          | $d = 2$                      | non primo ( $6 = 2 \cdot 3$ )   |
| 7   | 2                          | nessuno                      | <b>primo</b>                    |
| 8   | 2                          | $d = 2$                      | non primo ( $8 = 2 \cdot 4$ )   |
| 9   | 3                          | $d = 3$                      | non primo ( $9 = 3 \cdot 3$ )   |
| 10  | 3                          | $d = 2$                      | non primo ( $10 = 2 \cdot 5$ )  |
| 11  | 3                          | nessuno                      | <b>primo</b>                    |
| 12  | 3                          | $d = 2$                      | non primo ( $12 = 2 \cdot 6$ )  |
| 13  | 3                          | nessuno                      | <b>primo</b>                    |
| 14  | 3                          | $d = 2$                      | non primo ( $14 = 2 \cdot 7$ )  |
| 15  | 3                          | $d = 3$                      | non primo ( $15 = 3 \cdot 5$ )  |
| 16  | 4                          | $d = 2$                      | non primo ( $16 = 2 \cdot 8$ )  |
| 17  | 4                          | nessuno                      | <b>primo</b>                    |
| 18  | 4                          | $d = 2$                      | non primo ( $18 = 2 \cdot 9$ )  |
| 19  | 4                          | nessuno                      | <b>primo</b>                    |
| 20  | 4                          | $d = 2$                      | non primo ( $20 = 2 \cdot 10$ ) |
| 21  | 4                          | $d = 3$                      | non primo ( $21 = 3 \cdot 7$ )  |
| 22  | 4                          | $d = 2$                      | non primo ( $22 = 2 \cdot 11$ ) |
| 23  | 4                          | nessuno                      | <b>primo</b>                    |
| 24  | 4                          | $d = 2$                      | non primo ( $24 = 2 \cdot 12$ ) |
| 25  | 5                          | $d = 5$                      | non primo ( $25 = 5 \cdot 5$ )  |
| 26  | 5                          | $d = 2$                      | non primo ( $26 = 2 \cdot 13$ ) |
| 27  | 5                          | $d = 3$                      | non primo ( $27 = 3 \cdot 9$ )  |
| 28  | 5                          | $d = 2$                      | non primo ( $28 = 2 \cdot 14$ ) |
| 29  | 5                          | nessuno                      | <b>primo</b>                    |
| 30  | 5                          | $d = 2$                      | non primo ( $30 = 2 \cdot 15$ ) |

2. I numeri primi in  $\{2, 3, \dots, 30\}$  sono

2, 3, 5, 7, 11, 13, 17, 19, 23, 29

Ci sono 10 numeri primi in  $\{2, 3, \dots, 30\}$ .

### Esercizio 5.11 — Algoritmi numerici (E11)

Consideriamo i seguenti numeri naturali:

$$n \in \{6, 19, 49, 53, 91\}$$

#### Domande

1. Applicare l'algoritmo PrimalityTest a ciascun numero della lista. Per ogni  $n$  mostrare l'esecuzione passo per passo, indicando il valore di  $\lfloor \sqrt{n} \rfloor$ , i valori di  $d$  provati e il risultato di ciascun test  $n \bmod d = 0$ .

#### Soluzione

1. Esecuzione di PrimalityTest su ciascun numero.

**Verifica  $n = 6$ :**  $\lfloor \sqrt{6} \rfloor = 2$ .

| $d$                                       | $6 \bmod d$ | azione                   |
|-------------------------------------------|-------------|--------------------------|
| 2                                         | 0           | restituisce <b>false</b> |
| <b>6 non è primo</b> ( $6 = 2 \cdot 3$ ). |             |                          |

**Verifica  $n = 19$ :**  $\lfloor \sqrt{19} \rfloor = 4$ .

| $d$                                                             | $19 \bmod d$ | azione   |
|-----------------------------------------------------------------|--------------|----------|
| 2                                                               | $1 \neq 0$   | continua |
| 3                                                               | $1 \neq 0$   | continua |
| 4                                                               | $3 \neq 0$   | continua |
| Il ciclo termina. Restituisce <b>true</b> : <b>19 è primo</b> . |              |          |

**Verifica  $n = 49$ :**  $\lfloor \sqrt{49} \rfloor = 7$ .

| $d$                                         | $49 \bmod d$ | azione                   |
|---------------------------------------------|--------------|--------------------------|
| 2                                           | $1 \neq 0$   | continua                 |
| 3                                           | $1 \neq 0$   | continua                 |
| 4                                           | $1 \neq 0$   | continua                 |
| 5                                           | $4 \neq 0$   | continua                 |
| 6                                           | $1 \neq 0$   | continua                 |
| 7                                           | 0            | restituisce <b>false</b> |
| <b>49 non è primo</b> ( $49 = 7 \cdot 7$ ). |              |                          |

Verifica  $n = 53$ :  $\lfloor \sqrt{53} \rfloor = 7$ .

| $d$                                                             | $53 \bmod d$ | azione   |
|-----------------------------------------------------------------|--------------|----------|
| 2                                                               | $1 \neq 0$   | continua |
| 3                                                               | $2 \neq 0$   | continua |
| 4                                                               | $1 \neq 0$   | continua |
| 5                                                               | $3 \neq 0$   | continua |
| 6                                                               | $5 \neq 0$   | continua |
| 7                                                               | $4 \neq 0$   | continua |
| Il ciclo termina. Restituisce <b>true</b> : <b>53 è primo</b> . |              |          |

Verifica  $n = 91$ :  $\lfloor \sqrt{91} \rfloor = 9$ .

| $d$                                          | $91 \bmod d$ | azione                   |
|----------------------------------------------|--------------|--------------------------|
| 2                                            | $1 \neq 0$   | continua                 |
| 3                                            | $1 \neq 0$   | continua                 |
| 4                                            | $3 \neq 0$   | continua                 |
| 5                                            | $1 \neq 0$   | continua                 |
| 6                                            | $1 \neq 0$   | continua                 |
| 7                                            | 0            | restituisce <b>false</b> |
| <b>91 non è primo</b> ( $91 = 7 \cdot 13$ ). |              |                          |

### Esercizio 5.12 — Algoritmi numerici (E12)

Consideriamo il numero naturale  $n = 15$ .

#### Domande

1. Applicare l'algoritmo SieveOfEratosthenes con  $n = 15$ . Mostrare l'esecuzione passo per passo: indicare il valore di  $\lfloor \sqrt{15} \rfloor$  e, per ogni primo  $p$  trovato, elencare i valori di  $j$  nel ciclo interno e i corrispondenti assegnamenti  $prime[j] \leftarrow 0$ .

2. Elencare tutti i numeri primi  $\leq 15$ .

### Soluzione

1. Si ha  $n = 15$  e  $\lfloor \sqrt{15} \rfloor = 3$ . L'array  $prime[1..15]$  è inizializzato a 1 e poi  $prime[1] \leftarrow 0$ . Ogni riga della tabella mostra lo stato dell'array dopo l'assegnamento  $prime[j] \leftarrow 0$  (blu: elemento appena posto a 0; rosso: valore corrente di  $p$ ).

|     |     | $prime[1..15]$ |   |   |   |   |   |   |   |   |    |    |    |    |    |    | azione                           |
|-----|-----|----------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----------------------------------|
| $p$ | $j$ | 1              | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |                                  |
| —   | —   | 0              | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1  | 1  | 1  | 1  | 1  | 1  | inizializzazione                 |
| 2   | 4   | 0              | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1  | 1  | 1  | 1  | 1  | 1  | $prime[4] \leftarrow 0$          |
|     | 6   | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1  | 1  | 1  | 1  | 1  | 1  | $prime[6] \leftarrow 0$          |
|     | 8   | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1  | 1  | 1  | 1  | 1  | 1  | $prime[8] \leftarrow 0$          |
|     | 10  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0  | 1  | 1  | 1  | 1  | 1  | $prime[10] \leftarrow 0$         |
|     | 12  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0  | 1  | 0  | 1  | 1  | 1  | $prime[12] \leftarrow 0$         |
|     | 14  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0  | 1  | 0  | 1  | 0  | 1  | $prime[14] \leftarrow 0$         |
|     | 16  |                |   |   |   |   |   |   |   |   |    |    |    |    |    |    | $j = 16 > 15$ : arresto          |
| 3   | 9   | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0  | 1  | 0  | 1  | 0  | 1  | $prime[9] \leftarrow 0$          |
|     | 12  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0  | 1  | 0  | 1  | 0  | 1  | $prime[12] \leftarrow 0$ (già 0) |
|     | 15  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0  | 1  | 0  | 1  | 0  | 0  | $prime[15] \leftarrow 0$         |
|     | 18  |                |   |   |   |   |   |   |   |   |    |    |    |    |    |    | $j = 18 > 15$ : arresto          |

Il ciclo esterno termina:  $p = 4 > \lfloor \sqrt{15} \rfloor = 3$ .

2. I numeri primi  $\leq 15$  sono 2, 3, 5, 7, 11, 13.

### Esercizio 5.13 — Algoritmi numerici (E13)

Consideriamo il numero naturale  $n = 30$ .

#### Domande

- Applicare l'algoritmo SieveOfEratosthenes con  $n = 30$ . Mostrare l'esecuzione passo per passo: indicare il valore di  $\lfloor \sqrt{30} \rfloor$  e, per ogni primo  $p$  trovato, elencare i valori di  $j$  nel ciclo interno e i corrispondenti assegnamenti  $prime[j] \leftarrow 0$ .
- Elencare tutti i numeri primi  $\leq 30$ .

### Soluzione

1. Si ha  $n = 30$  e  $\lfloor \sqrt{30} \rfloor = 5$ . L'array  $prime[1..30]$  è inizializzato a 1 e poi  $prime[1] \leftarrow 0$ .

- Iterazione  $p = 2$ :**  $prime[2] = 1$ , quindi 2 è primo. Ciclo interno da  $j = p^2 = 4$  con passo 2:

| $j$ | assegnamento             | nota |
|-----|--------------------------|------|
| 4   | $prime[4] \leftarrow 0$  |      |
| 6   | $prime[6] \leftarrow 0$  |      |
| 8   | $prime[8] \leftarrow 0$  |      |
| 10  | $prime[10] \leftarrow 0$ |      |
| 12  | $prime[12] \leftarrow 0$ |      |
| 14  | $prime[14] \leftarrow 0$ |      |
| 16  | $prime[16] \leftarrow 0$ |      |
| 18  | $prime[18] \leftarrow 0$ |      |
| 20  | $prime[20] \leftarrow 0$ |      |
| 22  | $prime[22] \leftarrow 0$ |      |
| 24  | $prime[24] \leftarrow 0$ |      |
| 26  | $prime[26] \leftarrow 0$ |      |
| 28  | $prime[28] \leftarrow 0$ |      |
| 30  | $prime[30] \leftarrow 0$ |      |
| 32  | $j = 32 > 30$ : arresto  |      |

- **Iterazione**  $p = 3$ :  $prime[3] = 1$ , quindi 3 è primo. Ciclo interno da  $j = p^2 = 9$  con passo 3:

| $j$ | assegnamento             | nota  |
|-----|--------------------------|-------|
| 9   | $prime[9] \leftarrow 0$  |       |
| 12  | $prime[12] \leftarrow 0$ | già 0 |
| 15  | $prime[15] \leftarrow 0$ |       |
| 18  | $prime[18] \leftarrow 0$ | già 0 |
| 21  | $prime[21] \leftarrow 0$ |       |
| 24  | $prime[24] \leftarrow 0$ | già 0 |
| 27  | $prime[27] \leftarrow 0$ |       |
| 30  | $prime[30] \leftarrow 0$ | già 0 |
| 33  | $j = 33 > 30$ : arresto  |       |

- **Iterazione**  $p = 4$ :  $prime[4] = 0$ , quindi 4 non è primo: si salta.
- **Iterazione**  $p = 5$ :  $prime[5] = 1$ , quindi 5 è primo. Ciclo interno da  $j = p^2 = 25$  con passo 5:

| $j$ | assegnamento             | nota  |
|-----|--------------------------|-------|
| 25  | $prime[25] \leftarrow 0$ |       |
| 30  | $prime[30] \leftarrow 0$ | già 0 |
| 35  | $j = 35 > 30$ : arresto  |       |

- Il ciclo esterno termina:  $p = 6 > \lfloor \sqrt{30} \rfloor = 5$ .

Stato finale dell'array  $prime[1..30]$  (gli elementi uguali a 1 sono in blu):

|            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |
|------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| $i$        | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $prime[i]$ | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0  | 1  | 0  | 1  | 0  | 0  |

  

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $i$        | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| $prime[i]$ | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  |

2. I numeri primi  $\leq 30$  sono 2, 3, 5, 7, 11, 13, 17, 19, 23, 29.

## Esercizio 5.14 — Algoritmi numerici (E14)

Consideriamo la seguente traccia di esecuzione dell'algoritmo SieveOfEratosthenes per  $n = 12$ , in cui alcuni elementi sono stati sostituiti da un punto interrogativo.

| $p$ | $j$ | $prime[1..12]$ |   |   |   |   |   |   |   |   |    |    |    | azione                           |
|-----|-----|----------------|---|---|---|---|---|---|---|---|----|----|----|----------------------------------|
|     |     | 1              | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |                                  |
| —   | —   | 0              | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1  | 1  | 1  | inizializzazione                 |
| 2   | 4   | 0              | 1 | 1 | ? | 1 | 1 | 1 | 1 | 1 | 1  | 1  | 1  | $prime[4] \leftarrow 0$          |
|     | 6   | 0              | 1 | 1 | 0 | 1 | ? | 1 | 1 | 1 | 1  | 1  | 1  | $prime[6] \leftarrow 0$          |
|     | 8   | 0              | 1 | 1 | 0 | 1 | 0 | 1 | ? | 1 | 1  | 1  | 1  | $prime[8] \leftarrow 0$          |
|     | 10  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | ?  | 1  | 1  | $prime[10] \leftarrow 0$         |
|     | 12  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0  | 1  | ?  | $prime[12] \leftarrow 0$         |
|     | 14  |                |   |   |   |   |   |   |   |   |    |    |    | $j = 14 > 12$ : arresto          |
| 3   | 9   | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | ? | 0  | 1  | 0  | $prime[9] \leftarrow 0$          |
|     | 12  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0  | 1  | ?  | $prime[12] \leftarrow 0$ (già 0) |
|     | 15  |                |   |   |   |   |   |   |   |   |    |    |    | $j = 15 > 12$ : arresto          |

Il ciclo esterno termina:  $p = 4 > \lfloor \sqrt{12} \rfloor = ?$ .

## Domande

1. Completare tutti gli elementi contrassegnati da un punto interrogativo nella tabella.
2. Qual è il valore di  $\lfloor \sqrt{12} \rfloor$ ? Perché il ciclo esterno può fermarsi quando  $p = 4$ ?
3. Elencare tutti i numeri primi  $\leq 12$ .

## Soluzione

1. Tabella completata:

| $p$ | $j$ | $prime[1..12]$ |   |   |   |   |   |   |   |   |    |    |    | azione                           |
|-----|-----|----------------|---|---|---|---|---|---|---|---|----|----|----|----------------------------------|
|     |     | 1              | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |                                  |
| —   | —   | 0              | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1  | 1  | 1  | inizializzazione                 |
| 2   | 4   | 0              | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1  | 1  | 1  | $prime[4] \leftarrow 0$          |
|     | 6   | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1  | 1  | 1  | $prime[6] \leftarrow 0$          |
|     | 8   | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1  | 1  | 1  | $prime[8] \leftarrow 0$          |
|     | 10  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0  | 1  | 1  | $prime[10] \leftarrow 0$         |
|     | 12  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0  | 1  | 0  | $prime[12] \leftarrow 0$         |
|     | 14  |                |   |   |   |   |   |   |   |   |    |    |    | $j = 14 > 12$ : arresto          |
| 3   | 9   | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0  | 1  | 0  | $prime[9] \leftarrow 0$          |
|     | 12  | 0              | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0  | 1  | 0  | $prime[12] \leftarrow 0$ (già 0) |
|     | 15  |                |   |   |   |   |   |   |   |   |    |    |    | $j = 15 > 12$ : arresto          |

Il ciclo esterno termina:  $p = 4 > \lfloor \sqrt{12} \rfloor = 3$ .

Ogni punto interrogativo è l'elemento  $prime[j]$  della riga di indice  $j$ : vale 0 dopo l'assegnamento  $prime[j] \leftarrow 0$  (nella riga  $p = 3$ ,  $j = 12$ , l'elemento era già 0).

2.  $\lfloor \sqrt{12} \rfloor = 3$ . Il ciclo esterno viene eseguito per  $p = 2, 3$  e si ferma quando  $p = 4 > 3$ . Ogni numero composto  $m \leq 12$  ha un divisore primo non maggiore di  $\lfloor \sqrt{12} \rfloor = 3$  (capitolo «Algoritmi numerici», sezione «Crivello di Eratostene»), quindi tutti i numeri composti sono già stati marcati: 4, 6, 8, 10, 12 da  $p = 2$  e 9 da  $p = 3$ .
3. I numeri primi  $\leq 12$  sono 2, 3, 5, 7, 11.

### Esercizio 5.15 — Algoritmi numerici (E15)

Consideriamo l'algoritmo SieveOfEratosthenes applicato a  $n = 25$ .

#### Domande

1. Applicare l'algoritmo SieveOfEratosthenes con  $n = 25$ . Mostrare l'esecuzione passo per passo, indicando il valore di  $\lfloor \sqrt{25} \rfloor$  e, per ogni primo  $p$ , i valori di  $j$  provati nel ciclo interno e i corrispondenti assegnamenti  $prime[j] \leftarrow 0$ .
2. Per  $p = 5$ : qual è il primo valore di  $j$  elaborato dal ciclo interno? Perché il punto di partenza è  $p^2 = 25$  e non  $2p = 10$ ?
3. Elencare tutti i numeri primi  $\leq 25$ .

#### Soluzione

1. Si ha  $n = 25$  e  $\lfloor \sqrt{25} \rfloor = 5$ . L'array  $prime[1..25]$  è inizializzato a 1 e poi  $prime[1] \leftarrow 0$ .

- **Iterazione**  $p = 2$ :  $prime[2] = 1$ , quindi 2 è primo. Ciclo interno da  $j = p^2 = 4$  con passo 2:

| $j$ | assegnamento             | nota                    |
|-----|--------------------------|-------------------------|
| 4   | $prime[4] \leftarrow 0$  |                         |
| 6   | $prime[6] \leftarrow 0$  |                         |
| 8   | $prime[8] \leftarrow 0$  |                         |
| 10  | $prime[10] \leftarrow 0$ |                         |
| 12  | $prime[12] \leftarrow 0$ |                         |
| 14  | $prime[14] \leftarrow 0$ |                         |
| 16  | $prime[16] \leftarrow 0$ |                         |
| 18  | $prime[18] \leftarrow 0$ |                         |
| 20  | $prime[20] \leftarrow 0$ |                         |
| 22  | $prime[22] \leftarrow 0$ |                         |
| 24  | $prime[24] \leftarrow 0$ |                         |
| 26  |                          | $j = 26 > 25$ : arresto |

- **Iterazione**  $p = 3$ :  $prime[3] = 1$ , quindi 3 è primo. Ciclo interno da  $j = p^2 = 9$  con passo 3:

| $j$ | assegnamento             | nota                    |
|-----|--------------------------|-------------------------|
| 9   | $prime[9] \leftarrow 0$  |                         |
| 12  | $prime[12] \leftarrow 0$ | già 0                   |
| 15  | $prime[15] \leftarrow 0$ |                         |
| 18  | $prime[18] \leftarrow 0$ | già 0                   |
| 21  | $prime[21] \leftarrow 0$ |                         |
| 24  | $prime[24] \leftarrow 0$ | già 0                   |
| 27  |                          | $j = 27 > 25$ : arresto |

- **Iterazione**  $p = 4$ :  $prime[4] = 0$ , quindi 4 non è primo: si salta.
- **Iterazione**  $p = 5$ :  $prime[5] = 1$ , quindi 5 è primo. Ciclo interno da  $j = p^2 = 25$  con passo 5:

| $j$ | assegnamento             | nota                    |
|-----|--------------------------|-------------------------|
| 25  | $prime[25] \leftarrow 0$ |                         |
| 30  |                          | $j = 30 > 25$ : arresto |

- Il ciclo esterno termina:  $p = 6 > \lfloor \sqrt{25} \rfloor = 5$ .

Stato finale dell'array  $prime[1..25]$  (gli elementi uguali a 1 sono in blu):

|            |   |   |   |   |   |   |   |   |   |    |    |    |    |
|------------|---|---|---|---|---|---|---|---|---|----|----|----|----|
| $i$        | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| $prime[i]$ | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0  | 1  | 0  | 1  |

  

|            |    |    |    |    |    |    |    |    |    |    |    |    |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|
| $i$        | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| $prime[i]$ | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  |

- Per  $p = 5$ , il ciclo interno parte da  $j = p^2 = 25$ . Tutti i multipli più piccoli di 5 maggiori di 5, cioè 10, 15, 20, sono già stati posti a 0:  $10 = 2 \cdot 5$  e  $20 = 2 \cdot 10$  quando  $p = 2$ , e  $15 = 3 \cdot 5$  quando  $p = 3$ . In generale, ogni multiplo  $m$  di  $p$  con  $p < m < p^2$  ha un divisore  $d$  con  $2 \leq d < p$ , quindi un divisore primo minore di  $p$ , che lo ha già marcato.
- I numeri primi  $\leq 25$  sono 2, 3, 5, 7, 11, 13, 17, 19, 23.

### Esercizio 5.16 — Algoritmi numerici (E16)

Consideriamo il numero naturale  $n = 50$ .

#### Domande

- Applicare l'algoritmo SieveOfEratosthenes con  $n = 50$ . Per ogni valore di  $p$  del ciclo esterno riportare il valore di  $prime[p]$  e, quando  $p$  è primo, tutti i valori di  $j$  elaborati dal ciclo interno (i multipli di  $p$  da  $p^2$  a 50), distinguendo gli elementi posti a 0 per la prima volta da quelli già uguali a 0.
- Elencare tutti i numeri primi  $\leq 50$  e contarli.

## Soluzione

1. Si ha  $n = 50$  e  $\lfloor \sqrt{50} \rfloor = 7$ . L'array  $prime[1..50]$  è inizializzato a 1 e poi  $prime[1] \leftarrow 0$ .

| $p$ | $prime[p]$ | valori di $j$   | posti a 0 ora                                                                               | già 0                      |
|-----|------------|-----------------|---------------------------------------------------------------------------------------------|----------------------------|
| 2   | 1 (primo)  | 4, 6, ..., 50   | 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50 | —                          |
| 3   | 1 (primo)  | 9, 12, ..., 48  | 9, 15, 21, 27, 33, 39, 45                                                                   | 12, 18, 24, 30, 36, 42, 48 |
| 4   | 0 (salta)  | —               | —                                                                                           | —                          |
| 5   | 1 (primo)  | 25, 30, ..., 50 | 25, 35                                                                                      | 30, 40, 45, 50             |
| 6   | 0 (salta)  | —               | —                                                                                           | —                          |
| 7   | 1 (primo)  | 49              | 49                                                                                          | —                          |

Il ciclo esterno termina:  $p = 8 > \lfloor \sqrt{50} \rfloor = 7$ .

Stato finale dell'array  $prime[1..50]$  (gli elementi uguali a 1 sono in blu):

|            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
|------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|
| $i$        | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| $prime[i]$ | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 1  |

  

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $i$        | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| $prime[i]$ | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0  |

  

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $i$        | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| $prime[i]$ | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  |

2. I numeri primi  $\leq 50$  sono

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47

Ci sono 15 numeri primi  $\leq 50$ .

## Esercizio 5.17 — Algoritmi numerici (E17)

Consideriamo le seguenti coppie di numeri naturali  $(n, m)$ :

- (1071, 462)
- (100, 75)
- (17, 5)
- (89, 55)
- (30, 84)

## Domande

1. Applicare l'algoritmo EuclideanAlgorithm per calcolare  $\gcd(n, m)$  per tutte le coppie. Per ogni coppia mostrare l'esecuzione passo per passo in una tabella, indicando a ogni passo i valori di  $n$ ,  $m$  e  $r = n \bmod m$ , e dare il numero di iterazioni del ciclo while.
2. Quali coppie sono coprima?
3. Nella coppia 5 si ha  $n < m$ . Che cosa accade nella prima iterazione?

### Soluzione

1. Esecuzioni dell'algoritmo di Euclide.

(a) Calcoliamo  $\gcd(1071, 462)$ :

| passo | $n$  | $m$ | $r = n \bmod m$        |
|-------|------|-----|------------------------|
| 1     | 1071 | 462 | $1071 \bmod 462 = 147$ |
| 2     | 462  | 147 | $462 \bmod 147 = 21$   |
| 3     | 147  | 21  | $147 \bmod 21 = 0$     |
| 4     | 21   | 0   | — (il ciclo termina)   |

**Risultato:**  $\gcd(1071, 462) = 21$ , con 3 iterazioni del ciclo while.

(b) Calcoliamo  $\gcd(100, 75)$ :

| passo | $n$ | $m$ | $r = n \bmod m$      |
|-------|-----|-----|----------------------|
| 1     | 100 | 75  | $100 \bmod 75 = 25$  |
| 2     | 75  | 25  | $75 \bmod 25 = 0$    |
| 3     | 25  | 0   | — (il ciclo termina) |

**Risultato:**  $\gcd(100, 75) = 25$ , con 2 iterazioni del ciclo while.

(c) Calcoliamo  $\gcd(17, 5)$ :

| passo | $n$ | $m$ | $r = n \bmod m$      |
|-------|-----|-----|----------------------|
| 1     | 17  | 5   | $17 \bmod 5 = 2$     |
| 2     | 5   | 2   | $5 \bmod 2 = 1$      |
| 3     | 2   | 1   | $2 \bmod 1 = 0$      |
| 4     | 1   | 0   | — (il ciclo termina) |

**Risultato:**  $\gcd(17, 5) = 1$ , con 3 iterazioni del ciclo while.

(d) Calcoliamo  $\gcd(89, 55)$ :

| passo | $n$ | $m$ | $r = n \bmod m$      |
|-------|-----|-----|----------------------|
| 1     | 89  | 55  | $89 \bmod 55 = 34$   |
| 2     | 55  | 34  | $55 \bmod 34 = 21$   |
| 3     | 34  | 21  | $34 \bmod 21 = 13$   |
| 4     | 21  | 13  | $21 \bmod 13 = 8$    |
| 5     | 13  | 8   | $13 \bmod 8 = 5$     |
| 6     | 8   | 5   | $8 \bmod 5 = 3$      |
| 7     | 5   | 3   | $5 \bmod 3 = 2$      |
| 8     | 3   | 2   | $3 \bmod 2 = 1$      |
| 9     | 2   | 1   | $2 \bmod 1 = 0$      |
| 10    | 1   | 0   | — (il ciclo termina) |

**Risultato:**  $\gcd(89, 55) = 1$ , con 9 iterazioni del ciclo while.

I numeri 89, 55, 34, 21, 13, 8, 5, 3, 2, 1 sono numeri di Fibonacci consecutivi: a ogni passo il quoziente è 1, e i resti decrescono più lentamente possibile.

(e) Calcoliamo  $\gcd(30, 84)$ :

| passo | $n$ | $m$ | $r = n \bmod m$      |
|-------|-----|-----|----------------------|
| 1     | 30  | 84  | $30 \bmod 84 = 30$   |
| 2     | 84  | 30  | $84 \bmod 30 = 24$   |
| 3     | 30  | 24  | $30 \bmod 24 = 6$    |
| 4     | 24  | 6   | $24 \bmod 6 = 0$     |
| 5     | 6   | 0   | — (il ciclo termina) |

**Risultato:**  $\gcd(30, 84) = 6$ , con 4 iterazioni del ciclo while.

- Due numeri sono coprimi se e solo se il loro massimo comun divisore è 1. Quindi le coppie coprime sono (17, 5) e (89, 55). Le coppie (1071, 462), (100, 75) e (30, 84) non sono coprime.
- Poiché  $30 < 84$ , si ha  $30 \bmod 84 = 30$ , quindi la prima iterazione sostituisce (30, 84) con (84, 30): **scambia** semplicemente i due numeri. Dalla seconda iterazione in poi l'esecuzione è la stessa che per (84, 30), che richiede 3 iterazioni. Dunque l'ipotesi  $n \geq m$  non serve per la correttezza: se  $n < m$ , l'algoritmo esegue solo un'iterazione in più.

### Esercizio 5.18 — Algoritmi numerici (E18)

Consideriamo il seguente algoritmo **naïf** per il massimo comun divisore, che prova tutti i candidati divisori da  $\min\{n, m\}$  fino a 1.

**Algorithm:** NaiveGCD**Input** : Due numeri naturali  $n, m \geq 1$ **Output**:  $\gcd(n, m)$ 

```

1 function NaiveGCD( $n, m$ )
2    $d := \min\{n, m\}$ 
3   while  $n \bmod d \neq 0$  or  $m \bmod d \neq 0$  do
4      $d \leftarrow d - 1$ 
5   end
6   return  $d$ 
7 end

```

**Domande** Le domande 2–4 usano le nozioni di invariante di ciclo, numero di passi e dimensione dell’input  $|I|$  (capitoli «Dimensione dell’input», «Tempo di esecuzione» e «Correttezza»).

1. Applicare NaiveGCD a  $(48, 18)$  e  $(35, 12)$ . Quante iterazioni del ciclo while vengono eseguite? Confrontarle con il numero di iterazioni dell’algoritmo di Euclide sulle stesse coppie.
2. Dimostrare che NaiveGCD è corretto, usando un invariante di ciclo.
3. Calcolare il numero di passi  $S$  di NaiveGCD in funzione di  $n, m$  e  $\gcd(n, m)$ , contando un passo per ogni esecuzione di una riga. Quali input danno il caso peggiore? Confrontare i due algoritmi su  $(1000, 999)$ .
4. Mostrare che NaiveGCD è esponenziale nella dimensione dell’input  $|I|$ .

**Soluzione**

1. Per  $(48, 18)$ :  $d$  parte da  $\min\{48, 18\} = 18$ , e i candidati vengono controllati nell’ordine

18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6.

Il primo divisore comune è  $d = 6$  ( $48 = 8 \cdot 6$ ,  $18 = 3 \cdot 6$ ). Il corpo del ciclo while viene eseguito  $18 - 6 = 12$  volte. L’algoritmo di Euclide richiede solo 3 iterazioni ( $48 \bmod 18 = 12$ ,  $18 \bmod 12 = 6$ ,  $12 \bmod 6 = 0$ ).

Per  $(35, 12)$ :  $d$  scende da 12 a 1, poiché 35 e 12 sono coprimi:  $12 - 1 = 11$  iterazioni. L’algoritmo di Euclide richiede 3 iterazioni ( $35 \bmod 12 = 11$ ,  $12 \bmod 11 = 1$ ,  $11 \bmod 1 = 0$ ).

2. Invariante di ciclo:

Ogni volta che la condizione alla riga 3 viene valutata,  $1 \leq \gcd(n, m) \leq d \leq \min\{n, m\}$ , cioè nessun numero  $d'$  con  $d < d' \leq \min\{n, m\}$  è un divisore comune di  $n$  e  $m$ .

- **Inizializzazione.**  $d = \min\{n, m\}$ , e ogni divisore comune è al più  $\min\{n, m\}$ .

- **Conservazione.** Se la condizione è vera,  $d$  non è un divisore comune; poiché nessun numero più grande fino a  $\min\{n, m\}$  è un divisore comune,  $\gcd(n, m) < d$ , cioè  $\gcd(n, m) \leq d-1$ . La riga 4 pone  $d \leftarrow d-1$ , e l'invariante continua a valere.
- **Terminazione.**  $d$  diminuisce di 1 a ogni iterazione e, per l'invariante, non può mai diventare minore di  $\gcd(n, m) \geq 1$ ; quando  $d = \gcd(n, m)$  la condizione è falsa. Quindi il ciclo termina, e quando termina  $d$  è un divisore comune con  $d \geq \gcd(n, m)$ , dunque  $d = \gcd(n, m)$ , che viene restituito.

3. Sia  $k = \min\{n, m\} - \gcd(n, m)$  il numero di iterazioni. Le righe 2 e 6 sono eseguite una volta, la riga 3  $k+1$  volte, la riga 4  $k$  volte:

$$S = 1 + (k+1) + k + 1 = 2k + 3 = 2(\min\{n, m\} - \gcd(n, m)) + 3.$$

Il caso peggiore si ha quando  $n$  e  $m$  sono coprimi:  $S = 2 \min\{n, m\} + 1$ . Per  $(1000, 999)$  (numeri consecutivi sono sempre coprimi) NaiveGCD esegue  $999 - 1 = 998$  iterazioni, mentre l'algoritmo di Euclide ne esegue solo 2 ( $1000 \bmod 999 = 1$ ,  $999 \bmod 1 = 0$ ).

4. Consideriamo le coppie  $(n, m) = (M, M-1)$ , con  $M \geq 3$ : sono coprimi, quindi  $S = 2(M-1) + 1 = 2M - 1 = \Theta(M)$ . L'input è formato da due numeri naturali, quindi  $|I| = \Theta(\log_2 M)$  con  $M = \max\{n, m\}$ , cioè esistono  $c_1, c_2 > 0$  tali che

$$2^{\frac{1}{c_2}|I|} \leq M \leq 2^{\frac{1}{c_1}|I|}.$$

Quindi, su questi input,  $S = 2^{\Theta(|I|)}$ , e il tempo di esecuzione nel caso peggiore soddisfa  $T(|I|) = 2^{\Theta(|I|)}$  (è anche al più  $2M + 1$  su ogni input).

NaiveGCD è esponenziale nella dimensione dell'input, mentre l'algoritmo di Euclide esegue  $O(\log_2 m)$  iterazioni (si veda l'esercizio successivo): per numeri con centinaia di cifre si può usare solo l'algoritmo di Euclide.

### Esercizio 5.19 — Algoritmi numerici (E19)

Consideriamo l'algoritmo di Euclide.

**Algorithm:** EuclideanAlgorithm**Input** : Due numeri naturali  $n, m \geq 1$ **Output:**  $\gcd(n, m)$ 

```

1 function EuclideanAlgorithm( $n, m$ )
2   while  $m \neq 0$  do
3      $r := n \bmod m$ 
4      $n \leftarrow m$ 
5      $m \leftarrow r$ 
6   end
7   return  $n$ 
8 end

```

I numeri di Fibonacci sono  $F_1 = F_2 = 1$  e  $F_k = F_{k-1} + F_{k-2}$  per  $k \geq 3$ :

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, ...

**Domande** La domanda 4 usa le nozioni di tempo di esecuzione e dimensione dell'input  $|I|$  (capitoli «Dimensione dell'input» e «Tempo di esecuzione»).

1. Contare il numero di iterazioni del ciclo while per le coppie di numeri di Fibonacci consecutivi (5, 3), (8, 5), (13, 8), (21, 13), (34, 21), (55, 34), (89, 55), (144, 89), (233, 144).
2. Dimostrare che, per ogni  $n \geq m \geq 1$ ,

$$n \bmod m < \frac{n}{2}.$$

3. Usando il punto 2, dimostrare che il numero  $t$  di iterazioni del ciclo while sull'input  $(n_0, m_0)$  soddisfa

$$t < 2 \log_2 m_0 + 2.$$

4. Assumendo che ogni operazione aritmetica sia un passo, determinare il tempo di esecuzione nel caso peggiore  $T(|I|)$  dell'algoritmo di Euclide. È polinomiale?

**Soluzione**

1. Per una coppia  $(F_{k+1}, F_k)$  con  $k \geq 3$  si ha  $F_k < F_{k+1} < 2F_k$ , quindi  $F_{k+1} \bmod F_k = F_{k+1} - F_k = F_{k-1}$ : a ogni iterazione si passa alla coppia precedente di numeri di Fibonacci consecutivi, fino a  $(2, 1) \rightarrow (1, 0)$ . Il numero di iterazioni è:

| $(n, m)$   | iterazioni $t$ | $2 \log_2 m + 2$ |
|------------|----------------|------------------|
| (5, 3)     | 3              | $\approx 5.17$   |
| (8, 5)     | 4              | $\approx 6.64$   |
| (13, 8)    | 5              | 8                |
| (21, 13)   | 6              | $\approx 9.40$   |
| (34, 21)   | 7              | $\approx 10.78$  |
| (55, 34)   | 8              | $\approx 12.17$  |
| (89, 55)   | 9              | $\approx 13.56$  |
| (144, 89)  | 10             | $\approx 14.95$  |
| (233, 144) | 11             | $\approx 16.34$  |

Per esempio,  $(89, 55) \rightarrow (55, 34) \rightarrow (34, 21) \rightarrow (21, 13) \rightarrow (13, 8) \rightarrow (8, 5) \rightarrow (5, 3) \rightarrow (3, 2) \rightarrow (2, 1) \rightarrow (1, 0)$ : 9 iterazioni. Il numero di iterazioni cresce di 1 ogni volta che  $m$  viene moltiplicato per circa 1.618: cresce in modo logaritmico in  $m$ . (I numeri di Fibonacci consecutivi sono, di fatto, il caso peggiore dell'algoritmo di Euclide.)

2. Sia  $n \geq m \geq 1$ . Due casi:

- se  $m \leq \frac{n}{2}$ , allora  $n \bmod m < m \leq \frac{n}{2}$ ;
- se  $m > \frac{n}{2}$ , allora  $m \leq n < 2m$ , quindi il quoziente di  $n$  diviso  $m$  è 1 e  $n \bmod m = n - m < n - \frac{n}{2} = \frac{n}{2}$ .

3. Indichiamo con  $m_s$  il valore di  $m$  dopo  $s$  iterazioni ( $m_0$  è l'input). Nell'iterazione  $s+1$  la coppia  $(n, m)$  diventa  $(m, n \bmod m)$ , quindi dopo l'iterazione  $s+1$  la variabile  $n$  è uguale a  $m_s$  e  $m_{s+1} < m_s$ . Dunque, se l'iterazione  $s+2$  viene eseguita,

$$m_{s+2} = m_s \bmod m_{s+1} \quad \text{con} \quad m_s > m_{s+1} \geq 1,$$

e per il punto 2,  $m_{s+2} < \frac{m_s}{2}$ : ogni due iterazioni il valore di  $m$  almeno si dimezza. Per induzione su  $k$ ,

$$m_{2k} < \frac{m_0}{2^k} \quad \text{ogni volta che l'iterazione } 2k \text{ viene eseguita, } k \geq 1.$$

Se  $t \leq 2$  la limitazione vale:  $t = 1 < 2 \leq 2 \log_2 m_0 + 2$ , e  $t = 2$  è possibile solo se  $m_0 \geq 2$  (per  $m_0 = 1$  il primo resto è 0), quindi  $2 < 2 \log_2 m_0 + 2$ . Sia  $t \geq 3$  e sia  $k = \lfloor (t-1)/2 \rfloor \geq 1$ . Poiché  $2k \leq t-1$ , l'iterazione  $2k+1$  viene eseguita, quindi  $m_{2k} \geq 1$ . Dunque

$$1 \leq m_{2k} < \frac{m_0}{2^k} \implies 2^k < m_0 \implies k < \log_2 m_0.$$

Poiché  $k \geq \frac{t-2}{2}$ , si ottiene  $\frac{t-2}{2} < \log_2 m_0$ , cioè

$$t < 2 \log_2 m_0 + 2.$$

I valori nella tabella del punto 1 rispettano questa limitazione.

4. Ogni iterazione esegue un numero costante di passi (righe 2–5), quindi il numero di passi soddisfa  $S = O(t) = O(\log_2 m_0)$ . L'input è formato da due numeri naturali, quindi  $|I| = \Theta(\log_2 M)$  con  $M = \max\{n_0, m_0\} \geq m_0$ : esiste  $c > 0$  tale che  $\log_2 m_0 \leq \log_2 M \leq \frac{1}{c}|I|$  per tutti gli input sufficientemente grandi. Quindi

$$T(|I|) = O(|I|).$$

L'algoritmo di Euclide esegue  $O(\log_2 m)$  iterazioni, e il suo tempo di esecuzione nel caso peggiore è  $T(|I|) = O(|I|)$ : è polinomiale (lineare) nella dimensione dell'input.

Osservazione: se i numeri sono molto grandi, un'operazione  $n \bmod m$  non è un singolo passo, ma può essere eseguita in un numero di passi polinomiale in  $|I|$ ; quindi l'algoritmo rimane polinomiale.

### Esercizio 5.20 — Algoritmi numerici (E20)

Consideriamo la seguente variante dell'algoritmo di Euclide, che usa solo sottrazioni (è la versione descritta originariamente da Euclide).

---

#### Algorithm: SubtractiveGCD

---

**Input** : Due numeri naturali  $n, m \geq 1$

**Output**:  $\gcd(n, m)$

```

1 function SubtractiveGCD( $n, m$ )
2   while  $n \neq m$  do
3     if  $n > m$  then
4        $n \leftarrow n - m$ 
5     end
6     else
7        $m \leftarrow m - n$ 
8     end
9   end
10  return  $n$ 
11 end
```

---

**Domande** Le domande 3 e 4 usano le nozioni di invariante di ciclo, tempo di esecuzione e dimensione dell'input  $|I|$  (capitoli «Dimensione dell'input», «Tempo di esecuzione» e «Correttezza»).

1. Applicare SubtractiveGCD a  $(48, 18)$ , mostrando i valori di  $n$  e  $m$  a ogni iterazione. Confrontare con l'algoritmo di Euclide.
2. Dimostrare che, per ogni  $n > m \geq 1$ ,  $\gcd(n, m) = \gcd(n - m, m)$ .
3. Dimostrare la correttezza di SubtractiveGCD usando un invariante di ciclo.

4. Quante iterazioni vengono eseguite sull'input  $(n, 1)$ ? SubtractiveGCD è polinomiale nella dimensione dell'input?

### Soluzione

1. Esecuzione su  $(48, 18)$ :

| iterazione | $n$ | $m$ | azione                                    |
|------------|-----|-----|-------------------------------------------|
| 1          | 48  | 18  | $n > m$ : $n \leftarrow 48 - 18 = 30$     |
| 2          | 30  | 18  | $n > m$ : $n \leftarrow 30 - 18 = 12$     |
| 3          | 12  | 18  | $n < m$ : $m \leftarrow 18 - 12 = 6$      |
| 4          | 12  | 6   | $n > m$ : $n \leftarrow 12 - 6 = 6$       |
| —          | 6   | 6   | $n = m$ : il ciclo termina, restituisce 6 |

**Risultato:**  $\gcd(48, 18) = 6$ , con 4 iterazioni. L'algoritmo di Euclide calcola  $48 \bmod 18 = 12$  in un solo passo: sottrarre 18 da 48 finché possibile (iterazioni 1–2) dà esattamente il resto 12. In generale, una sequenza di sottrazioni consecutive dello stesso numero è un solo passo dell'algoritmo di Euclide.

2. Sia  $n > m \geq 1$ . Se  $d$  è un divisore comune di  $n$  e  $m$ , allora  $n = k_1 d$  e  $m = k_2 d$ , quindi  $n - m = (k_1 - k_2)d$ , con  $k_1 - k_2 \geq 1$  poiché  $n > m$ :  $d$  è un divisore comune di  $n - m$  e  $m$ . Viceversa, se  $n - m = k_1 d$  e  $m = k_2 d$ , allora  $n = (n - m) + m = (k_1 + k_2)d$ :  $d$  è un divisore comune di  $n$  e  $m$ . Le due coppie hanno gli stessi divisori comuni, quindi lo stesso massimo comun divisore.
3. Indichiamo con  $n_0, m_0$  i valori in input. Invariante di ciclo:

Ogni volta che la condizione alla riga 2 viene valutata,  $n \geq 1$ ,  $m \geq 1$  e  $\gcd(n, m) = \gcd(n_0, m_0)$ .

- **Inizializzazione.**  $n = n_0 \geq 1$ ,  $m = m_0 \geq 1$ .
- **Conservazione.** Supponiamo che valga l'invariante e che  $n \neq m$ . Se  $n > m$ , la riga 4 sostituisce  $n$  con  $n - m \geq 1$  e, per il punto 2,  $\gcd(n - m, m) = \gcd(n, m)$ . Se  $n < m$ , la riga 7 sostituisce  $m$  con  $m - n \geq 1$  e, per il punto 2 con i ruoli scambiati,  $\gcd(n, m - n) = \gcd(n, m)$ . In entrambi i casi l'invariante continua a valere.
- **Terminazione.** Il numero naturale  $n + m$  diminuisce di almeno 1 a ogni iterazione (si sottrae un numero  $\geq 1$ ), ed è sempre almeno 2. Quindi il ciclo termina, e termina quando  $n = m$ . Allora

$$\gcd(n_0, m_0) = \gcd(n, n) = n,$$

che viene restituito alla riga 10.

4. Su  $(n, 1)$  con  $n \geq 2$ , ogni iterazione esegue  $n \leftarrow n - 1$ , fino a  $n = 1$ : il ciclo esegue  $n - 1$  iterazioni (per esempio  $(7, 1)$  ne richiede 6), mentre l'algoritmo di Euclide ne richiede

solo 1 ( $n \bmod 1 = 0$ ). Ogni iterazione esegue un numero costante di passi, quindi sugli input  $(M, 1)$  si ha  $S = \Theta(M)$ . La dimensione dell'input è  $|I| = (\lfloor \log_2 M \rfloor + 1) + 1 = \Theta(\log_2 M)$ , quindi  $M = 2^{\Theta(|I|)}$  e

$$T(|I|) = 2^{\Theta(|I|)}.$$

SubtractiveGCD è corretto, ma è esponenziale nella dimensione dell'input. Sostituendo le sottrazioni ripetute con una singola operazione mod si ottiene l'algoritmo di Euclide, che è polinomiale.