Algoritmi di approssimazione per il bin packing

Indice

- Perché algoritmi per il bin packing?
- Notazione
- Next Fit
- La classe Any Fit
  - First Fit
  - Best Fit
  - Worst Fit
- Proprietà di riempimento a metà
- Garanzie di prestazione
  - Il rapporto di prestazione nel caso peggiore di NF
  - Il rapporto di prestazione nel caso peggiore di WF
  - I rapporti di prestazione nel caso peggiore di FF e BF
  - Che cosa dà e che cosa non dà la condizione Any Fit
- Errori tipici
- Implementazione in C++
- In sintesi

Obiettivi

Alla fine di questo capitolo saremo in grado di:

- eseguire a mano gli algoritmi Next Fit, First Fit, Best Fit e Worst Fit, e scrivere l'assegnamento oggetto-contenitore e il numero di contenitori;
- decidere se un algoritmo appartiene alla classe Any Fit, e se una soluzione soddisfa la proprietà di riempimento a metà;
- dimostrare che Next Fit e ogni algoritmo Any Fit hanno una garanzia di prestazione relativa pari a 2;
- descrivere le famiglie di istanze che rendono tight la garanzia 2 per Next Fit e Worst Fit, ed enunciare il rapporto nel caso peggiore $\frac{17}{10}$ di First Fit e Best Fit;
- leggere ed eseguire le implementazioni in C++ dei quattro algoritmi.

Prerequisiti

- Da questo modulo, il capitolo «Principi degli algoritmi di approssimazione»: rapporto di prestazione, garanzia di prestazione relativa, garanzia tight e rapporto di prestazione nel caso peggiore $r(\mathscr{A})$.
- Dal modulo *Problemi di ottimizzazione combinatoria*, il capitolo «Problemi di packing»: il BPP, le sue istanze e il lower bound di conteggio $\left\lceil \frac{\sum_{i \in \mathscr{I}} w_i}{c} \right\rceil$ sul valore ottimo.
- Dal modulo *Algoritmi euristici*, il capitolo «Euristiche costruttive»: algoritmi euristici che costruiscono una soluzione un oggetto alla volta.

Il problema di questo capitolo

Il problema di questo capitolo è il
problema del bin packing (BPP):

la sua definizione e le sue ipotesi sono nella sua pagina.

- Studiamo quattro algoritmi semplici che pongono gli oggetti uno alla volta: Next Fit, First Fit, Best Fit e Worst Fit
- Per ciascuno dimostriamo quanto può essere lontano dall'ottimo, nel caso peggiore

Perché algoritmi per il bin packing?

Introduzione: Perché algoritmi per il bin packing?

- Partiamo da una piccola sfida: porre cinque oggetti nel minor numero possibile di contenitori
- Su un'istanza piccola possiamo trovare l'ottimo a mano, e dimostrarlo
- Con molti oggetti servono regole semplici, e una garanzia su quanti contenitori possono sprecare

Una sfida

Porre gli oggetti di pesi $(6, 5, 4, 7, 3)$
in contenitori di capacità 10,
usando il minor numero possibile di contenitori.

Sapete dimostrare che due contenitori non bastano?

La risposta

- Il peso totale è $25 > 2 \cdot 10$, quindi servono almeno

$$\left\lceil \frac{25}{10} \right\rceil = 3$$

contenitori.

- Tre contenitori bastano, per esempio

$$\{6, 4\}, \qquad \{5, 3\}, \qquad \{7\}.$$

Regole semplici con una garanzia

- Con molti oggetti non possiamo confrontarci con l'ottimo, e vogliamo regole semplici che non usino mai troppi contenitori.
- I quattro algoritmi di questo capitolo decidono il contenitore di ogni oggetto appena arriva.
- Il capitolo dimostra quanto possono essere lontani dall'ottimo, nel caso peggiore.

Riepilogo

La sfida:

- il bound di conteggio $\lceil \frac{25}{10} \rceil = 3$ mostra che due contenitori non bastano
- tre contenitori bastano: $\{6, 4\}$, $\{5, 3\}$, $\{7\}$

L'obiettivo del capitolo:

- regole semplici che collocano ogni oggetto appena arriva
- una dimostrazione di quanto possono essere lontane dall'ottimo, nel caso peggiore

Notazione

Introduzione: Notazione

- In questa sezione fissiamo la notazione del capitolo
- Ricordiamo che cos'è una soluzione ammissibile del BPP
- Definiamo il valore $z(I)$ di una soluzione ottima e il valore $\mathscr{A}(I)$ della soluzione di un algoritmo, che confrontiamo in tutto il capitolo

Oggetti, pesi e contenitori

- Come nella pagina del problema, c è la capacità dei contenitori e $\mathscr{I} = \{1, 2, \ldots, n\}$ l'insieme degli oggetti; in questo capitolo l'oggetto $i \in \mathscr{I}$ ha peso w_i.

- Una **soluzione ammissibile del BPP** è una partizione di $\mathscr{I}$ in **schemi di carico** (di impacchettamento) non vuoti, cioè sottoinsiemi di oggetti assegnati ciascuno a un contenitore, tali che il peso totale di ogni schema di carico non superi la capacità c dei contenitori.

Valore ottimo e valore di un algoritmo

- Indichiamo con $z(I)$ il **valore ottimo** del BPP sull'istanza I, cioè il numero minimo di contenitori.
- Indichiamo con $\mathscr{A}(I)$ il numero di contenitori usati dalla soluzione ammissibile restituita da un algoritmo euristico $\mathscr{A}$.
- Poiché ogni algoritmo euristico di questo capitolo restituisce una soluzione ammissibile,

$$\mathscr{A}(I) \geq z(I) \qquad \text{e} \qquad \frac{\mathscr{A}(I)}{z(I)} \geq 1$$

per ogni istanza I del BPP.

Riepilogo

Notazione:

- capacità c, oggetti $\mathscr{I} = \{1, 2, \ldots, n\}$, pesi w_i
- una soluzione ammissibile è una partizione di $\mathscr{I}$ in schemi di carico di peso totale al più c

Valori:

- $z(I)$ è il valore ottimo, $\mathscr{A}(I)$ il numero di contenitori usati dall'algoritmo $\mathscr{A}$
- $\mathscr{A}(I) \geq z(I)$, cioè $\frac{\mathscr{A}(I)}{z(I)} \geq 1$

Next Fit

Introduzione: Next Fit

- Next Fit è l'algoritmo più semplice del capitolo: mantiene aperto un solo contenitore
- Descriviamo la struttura dati della soluzione, lo pseudocodice e un'esecuzione a mano
- Vediamo anche come leggere lo schema di carico di ogni contenitore dall'output di un algoritmo: lo usiamo in tutto il capitolo

L'algoritmo Next Fit

- Il *Next Fit Algorithm* (indicato con NF) è un'euristica costruttiva per il BPP.
- Esamina gli oggetti nell'ordine in cui sono dati nell'istanza, mantenendo in ogni momento esattamente un contenitore aperto.
- Per ogni oggetto, NF lo pone nel contenitore aperto se vi entra; altrimenti chiude il contenitore, ne apre uno nuovo, e vi pone l'oggetto.

Struttura dati per la soluzione

- Usiamo un vettore $bin = (bin_1, bin_2, \ldots, bin_n)$ di lunghezza n per memorizzare la soluzione.
- In posizione i, il vettore contiene il numero del contenitore a cui è assegnato l'oggetto i. Più formalmente:

$$bin_i = b \iff \text{l'oggetto } i \text{ è posto nel contenitore } b.$$

Per esempio, se $bin_3 = 2$, l'oggetto 3 è assegnato al contenitore 2.

- Inoltre, manteniamo una variabile nb che memorizza il numero totale di contenitori usati, cioè alla fine dell'algoritmo $NF(I) = nb$.

Ora: lo pseudocodice di NF.

Algorithm: Next Fit (1/2)

Input : istanza del BPP I con n oggetti, pesi $w[1..n]$, capacità c

Output: assegnamento oggetto-contenitore $bin[1..n]$ e numero di contenitori usati nb

```
1 function NextFit(n, w, c)
2 |   bin := nuovo vettore[1..n]        // assegnamento
  |    oggetto-contenitore
3 |   nb := 1                    // contenitore corrente
4 |   rc := c                      // capacità residua
```

Algorithm: Next Fit (2/2)

```
function NextFit(n, w, c) (continua)
5     for i := 1 to n do
6         if w[i] > rc then
7             nb ← nb + 1            // aprire un nuovo
                contenitore
8             rc ← c
9         end
10        bin[i] ← nb          // assegnare l'oggetto al
            contenitore
11        rc ← rc − w[i]
12    end
13    return bin, nb
14 end
```

Esempio: esecuzione di NF (1/3)

Consideriamo contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5\}$ di 5 oggetti ($n = 5$), con i seguenti pesi degli oggetti:

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
w_i	6	5	4	7	3

Inizialmente $nb := 1$, $rc := c$. L'algoritmo NF produce la soluzione ammissibile delle prossime slide.

Esempio: esecuzione di NF (2/3)

1. Oggetto 1 (peso $w[1] = 6$): $w[1] = 6 \leq rc = 10$, quindi non si apre un nuovo contenitore. L'oggetto 1 è posto nel contenitore $nb = 1$, e $rc \leftarrow 10 - 6 = 4$.

2. Oggetto 2 (peso $w[2] = 5$): $w[2] = 5 > rc = 4$, quindi $nb \leftarrow 2$ e $rc \leftarrow c = 10$. L'oggetto 2 è posto nel contenitore $nb = 2$, e $rc \leftarrow 10 - 5 = 5$.

3. Oggetto 3 (peso $w[3] = 4$): $w[3] = 4 \leq rc = 5$, quindi non si apre un nuovo contenitore. L'oggetto 3 è posto nel contenitore $nb = 2$, e $rc \leftarrow 5 - 4 = 1$.

Esempio: esecuzione di NF (3/3)

4. Oggetto 4 (peso $w[4] = 7$): $w[4] = 7 > rc = 1$, quindi $nb \leftarrow 3$ e $rc \leftarrow c = 10$. L'oggetto 4 è posto nel contenitore $nb = 3$, e $rc \leftarrow 10 - 7 = 3$.

5. Oggetto 5 (peso $w[5] = 3$): $w[5] = 3 \leq rc = 3$, quindi non si apre un nuovo contenitore. L'oggetto 5 è posto nel contenitore $nb = 3$, e $rc \leftarrow 3 - 3 = 0$.

Quindi NF restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 2 & 3 & 3 \end{pmatrix}, \qquad nb = 3,$$

e quindi

$$NF(I) = nb = 3.$$

Schemi di carico dal vettore bin

- Dato l'output di un algoritmo, cioè il vettore $bin = (bin_1, bin_2, \ldots, bin_n)$ e il numero di contenitori nb, possiamo ottenere lo **schema di carico** P_b di ogni contenitore b, cioè l'insieme degli oggetti assegnati a esso, come segue:

$$P_b = \left\{ \; i \in \{1, 2, \ldots, n\} : bin_i = b \; \right\},$$

  per $b \in \{1, 2, \ldots, nb\}$.

- In altre parole, P_b è l'insieme di tutti gli oggetti i tali che $bin_i = b$; P_b è lo schema di carico del contenitore b, non il contenitore stesso.

Esempio: schemi di carico dal vettore bin (1/2)

Consideriamo l'istanza dell'esempio precedente, su cui NF restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 2 & 3 & 3 \end{pmatrix}, \qquad nb = 3.$$

Gli schemi di carico sono:

$$P_1 = \{i \in \{1,2,3,4,5\} : bin_i = 1\} = \{1\},$$
$$P_2 = \{i \in \{1,2,3,4,5\} : bin_i = 2\} = \{2,3\},$$
$$P_3 = \{i \in \{1,2,3,4,5\} : bin_i = 3\} = \{4,5\}.$$

Esempio: schemi di carico dal vettore bin (2/2)

Sono esattamente gli insiemi di oggetti posti nei contenitori costruiti dall'algoritmo:

- il contenitore 1 contiene l'oggetto 1 con peso $w_1 = 6$;
- il contenitore 2 contiene gli oggetti 2 e 3 con peso totale $w_2 + w_3 = 5 + 4 = 9$;
- il contenitore 3 contiene gli oggetti 4 e 5 con peso totale $w_4 + w_5 = 7 + 3 = 10$.

Riepilogo

Next Fit:

- esamina gli oggetti nell'ordine dato, con esattamente un contenitore aperto
- se l'oggetto non entra, chiude il contenitore e ne apre uno nuovo

L'output:

- il vettore bin, con $bin_i = b$ se l'oggetto i è posto nel contenitore b, e il numero di contenitori nb
- lo schema di carico del contenitore b è $P_b = \{i : bin_i = b\}$
- sull'istanza $w = (6, 5, 4, 7, 3)$, $c = 10$: $bin = (1, 2, 2, 3, 3)$ e $NF(I) = 3$

La classe Any Fit

Introduzione: La classe Any Fit

- Gli algoritmi Any Fit mantengono tutti i contenitori aperti finora, e aprono un nuovo contenitore solo quando l'oggetto non entra da nessuna parte
- Dimostriamo la proprietà chiave della classe: gli oggetti di due contenitori non entrano mai insieme
- Poi studiamo tre membri della classe: First Fit, Best Fit e Worst Fit, con i loro pseudocodici e un'esecuzione a mano

Algoritmi Any Fit

- Un'euristica costruttiva per il BPP che mantiene tutti i contenitori aperti finora, e apre un nuovo contenitore *solo quando* l'oggetto corrente non entra in nessuno di essi, si chiama *Any Fit Algorithm* (indicato con AF).
- Any Fit non è quindi un singolo algoritmo ma una *famiglia* di algoritmi.
- Ciò che la condizione lascia libero è la regola con cui scegliere un contenitore fra quelli in cui l'oggetto entra, e ogni regola dà un membro diverso della famiglia.

Definizione 1: Algoritmo Any Fit

Un'euristica costruttiva $\mathscr{A}$ per il BPP è un ***algoritmo Any Fit*** *se, indicati con $P_1, P_2, \ldots, P_{nb}$ gli schemi di carico dei contenitori aperti finora, apre un nuovo contenitore per l'oggetto i solo quando*

$$\sum_{h \in P_b} w_h + w_i > c \qquad \text{per ogni } b \in \{1, 2, \ldots, nb\},$$

mentre la scelta del contenitore, quando più di uno può accogliere l'oggetto, non è specificata.

Gli oggetti di due contenitori non entrano mai insieme (1/2)

- Siano b e b' due contenitori aperti da un algoritmo Any Fit, con $b < b'$.
- Sia f il primo oggetto posto nel contenitore b', e sia W_b il peso totale degli oggetti contenuti nel contenitore b nel momento in cui f è stato esaminato.
- Il contenitore b' è stato aperto perché f non entrava in nessuno dei contenitori aperti fino ad allora, e in particolare non nel contenitore b, quindi

$$W_b + w_f > c.$$

Gli oggetti di due contenitori non entrano mai insieme (2/2)

- Il peso totale di un contenitore non diminuisce mai durante l'esecuzione, quindi $\sum_{i \in P_b} w_i \geq W_b$, e $f \in P_{b'}$ dà $\sum_{i \in P_{b'}} w_i \geq w_f$.
- Di conseguenza

$$\sum_{i \in P_b} w_i + \sum_{i \in P_{b'}} w_i \;\geq\; W_b + w_f \;>\; c$$

  per ogni coppia di contenitori distinti b e b': di conseguenza, due contenitori di una soluzione Any Fit non si possono fondere.
- Qui si usa solo la condizione Any Fit, la regola di scelta non ha alcun ruolo.

NF non è un algoritmo Any Fit

- NF mantiene un solo contenitore aperto e lo chiude appena un oggetto non vi entra.
- Lo fa anche quando quell'oggetto entrerebbe in un contenitore chiuso in precedenza.

Ora: un'istanza su cui NF viola la condizione Any Fit.

Esempio: NF non è un algoritmo Any Fit (1/4)

Consideriamo contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4\}$ di 4 oggetti ($n = 4$), con i seguenti pesi degli oggetti:

	$i = 1$	$i = 2$	$i = 3$	$i = 4$
w_i	3	8	3	8

Inizialmente $nb := 1$, $rc := c$. L'algoritmo NF produce la soluzione ammissibile delle prossime slide.

Esempio: NF non è un algoritmo Any Fit (2/4)

1. Oggetto 1 (peso $w[1] = 3$): $w[1] = 3 \leq rc = 10$, quindi non si apre un nuovo contenitore. L'oggetto 1 è posto nel contenitore $nb = 1$, e $rc \leftarrow 10 - 3 = 7$.

2. Oggetto 2 (peso $w[2] = 8$): $w[2] = 8 > rc = 7$, quindi $nb \leftarrow 2$ e $rc \leftarrow c = 10$. L'oggetto 2 è posto nel contenitore $nb = 2$, e $rc \leftarrow 10 - 8 = 2$.

3. Oggetto 3 (peso $w[3] = 3$): $w[3] = 3 > rc = 2$, quindi $nb \leftarrow 3$ e $rc \leftarrow c = 10$. L'oggetto 3 è posto nel contenitore $nb = 3$, e $rc \leftarrow 10 - 3 = 7$.

4. Oggetto 4 (peso $w[4] = 8$): $w[4] = 8 > rc = 7$, quindi $nb \leftarrow 4$ e $rc \leftarrow c = 10$. L'oggetto 4 è posto nel contenitore $nb = 4$, e $rc \leftarrow 10 - 8 = 2$.

Esempio: NF non è un algoritmo Any Fit (3/4)

Quindi NF restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}, \qquad nb = 4.$$

Alla terza iterazione l'oggetto 3 entrerebbe nel contenitore 1, la cui capacità residua è 7, e NF apre comunque un nuovo contenitore: NF viola la condizione Any Fit.

Esempio: NF non è un algoritmo Any Fit (4/4)

- Un algoritmo Any Fit, al contrario, non può aprire un terzo contenitore per l'oggetto 3, perché l'oggetto entra nel contenitore 1.
- Vi pone l'oggetto 3 e poi apre un nuovo contenitore per l'oggetto 4, che non entra in nessuno dei due, usando 3 contenitori invece di 4.
- Questo vale per ogni algoritmo della classe, qualunque sia la regola di scelta.

Studiamo ora tre membri della famiglia Any Fit.

Differiscono solo nella regola con cui scelgono il contenitore:

il **primo** contenitore, il **più pieno**, il **più vuoto**.

L'algoritmo First Fit

- Il *First Fit Algorithm* (indicato con FF) è un'euristica costruttiva per il BPP.
- Esamina gli oggetti nell'ordine in cui sono dati nell'istanza, mantenendo tutti i contenitori aperti finora.
- Per ogni oggetto, FF scorre i contenitori nell'ordine di apertura e pone l'oggetto nel *primo* contenitore in cui entra; se nessun contenitore corrente può accogliere l'oggetto, FF apre un nuovo contenitore e vi pone l'oggetto.

First Fit e la classe Any Fit

- FF è un algoritmo Any Fit: apre un nuovo contenitore solo quando l'oggetto non entra in nessuno dei contenitori aperti finora.
- La regola lasciata libera dalla definizione di algoritmo Any Fit è qui fissata prendendo il primo contenitore in cui l'oggetto entra.

Ora: lo pseudocodice di FF.

Algorithm: First Fit (1/3)

Input : istanza del BPP I con n oggetti, pesi $w[1..n]$, capacità c

Output: assegnamento oggetto-contenitore $bin[1..n]$ e numero di contenitori usati nb

```
1 function FirstFit(n, w, c)
2     bin := nuovo vettore[1..n]          // assegnamento
       oggetto-contenitore
3     rc := nuovo vettore[1..n]      // capacità residue
4     nb := 1                      // numero di contenitori
5     rc[1] ← c                   // inizializzare il primo
       contenitore
```

Algorithm: First Fit (2/3)

```
function FirstFit(n, w, c) (continua)
6     for i := 1 to n do
7         s := −1                          // contenitore scelto
8         for b := 1 to nb do
9             if w[i] ≤ rc[b] then
10                s ← b
11                break
12            end
13        end
```

Algorithm: First Fit (3/3)

```
function FirstFit(n, w, c) (continua)
    for i := 1 to n (continua) do
14      if s = −1 then
15          nb ← nb + 1            // aprire un nuovo
              contenitore
16          rc[nb] ← c
17          s ← nb
18      end
19      bin[i] ← s                 // assegnare l'oggetto
20      rc[s] ← rc[s] − w[i]
21  end
22  return bin, nb
23 end
```

Esempio: esecuzione di FF (1/3)

Consideriamo contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5\}$ di 5 oggetti ($n = 5$), con i seguenti pesi degli oggetti:

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
w_i	6	5	4	7	3

L'algoritmo FF produce la soluzione ammissibile delle prossime slide.

Esempio: esecuzione di FF (2/3)

1. Oggetto 1 (peso 6): il contenitore 1 è aperto all'inizio ($nb = 1$, $rc[1] = c = 10$) e $w[1] \leq rc[1]$, quindi $s = 1$ e FF pone l'oggetto 1 nel contenitore 1, e $rc[1] \leftarrow 10 - 6 = 4$.

2. Oggetto 2 (peso 5): $b = 1$: $w[2] = 5 \leq rc[1] = 4$? No. Nessun contenitore può accoglierlo ($s = -1$), quindi FF apre il contenitore 2 ($rc[2] \leftarrow 10$). L'oggetto 2 è posto nel contenitore 2, e $rc[2] \leftarrow 10 - 5 = 5$.

3. Oggetto 3 (peso 4): $b = 1$: $w[3] = 4 \leq rc[1] = 4$? Sì, quindi $s \leftarrow 1$ e la scansione si ferma. L'oggetto 3 è posto nel contenitore 1, e $rc[1] \leftarrow 4 - 4 = 0$.

Esempio: esecuzione di FF (3/3)

4. Oggetto 4 (peso 7): $b = 1$: $w[4] = 7 \leq rc[1] = 0$? No. $b = 2$: $w[4] = 7 \leq rc[2] = 5$? No. Nessun contenitore può accoglierlo ($s = -1$), quindi FF apre il contenitore 3 ($rc[3] \leftarrow 10$). L'oggetto 4 è posto nel contenitore 3, e $rc[3] \leftarrow 10 - 7 = 3$.

5. Oggetto 5 (peso 3): $b = 1$: $w[5] = 3 \leq rc[1] = 0$? No. $b = 2$: $w[5] = 3 \leq rc[2] = 5$? Sì, quindi $s \leftarrow 2$ e la scansione si ferma. L'oggetto 5 è posto nel contenitore 2, e $rc[2] \leftarrow 5 - 3 = 2$.

Quindi FF restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 1 & 3 & 2 \end{pmatrix}, \qquad nb = 3,$$

e quindi

$$FF(I) = nb = 3.$$

Esempio: schemi di carico dal vettore bin (1/2)

Consideriamo l'istanza dell'esempio precedente, su cui FF restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 1 & 3 & 2 \end{pmatrix}, \qquad nb = 3.$$

Gli schemi di carico sono:

$$P_1 = \{i \in \{1,2,3,4,5\} : bin_i = 1\} = \{1,3\},$$
$$P_2 = \{i \in \{1,2,3,4,5\} : bin_i = 2\} = \{2,5\},$$
$$P_3 = \{i \in \{1,2,3,4,5\} : bin_i = 3\} = \{4\}.$$

Esempio: schemi di carico dal vettore bin (2/2)

Sono esattamente gli insiemi di oggetti posti nei contenitori costruiti dall'algoritmo:

- il contenitore 1 contiene gli oggetti 1 e 3 con peso totale $w_1 + w_3 = 6 + 4 = 10$;
- il contenitore 2 contiene gli oggetti 2 e 5 con peso totale $w_2 + w_5 = 5 + 3 = 8$;
- il contenitore 3 contiene l'oggetto 4 con peso $w_4 = 7$.

L'algoritmo Best Fit

- Il *Best Fit Algorithm* (indicato con BF) è un'euristica costruttiva per il BPP.
- Esamina gli oggetti nell'ordine in cui sono dati nell'istanza, mantenendo tutti i contenitori aperti finora.
- Per ogni oggetto, BF considera tutti i contenitori in cui l'oggetto entra e lo pone in quello con la *capacità residua minima* (in caso di pareggio, il contenitore aperto per primo); se nessun contenitore può accogliere l'oggetto, BF apre un nuovo contenitore e vi pone l'oggetto.

Perché Best Fit

- L'idea è riempire i contenitori il più possibile: usando il contenitore in cui l'oggetto lascia meno spazio inutilizzato, BF tiene disponibili i contenitori con capacità residua maggiore per gli oggetti più grandi che possono ancora arrivare.

- BF è un algoritmo Any Fit: apre un nuovo contenitore solo quando l'oggetto non entra in nessuno dei contenitori aperti finora, e la regola lasciata libera dalla definizione di algoritmo Any Fit è qui fissata prendendo il contenitore con la capacità residua minima.

Ora: lo pseudocodice di BF.

Algorithm: Best Fit (1/3)

Input : istanza del BPP I con n oggetti, pesi $w[1..n]$, capacità c

Output: assegnamento oggetto-contenitore $bin[1..n]$ e numero di contenitori usati nb

```
1 function BestFit(n, w, c)
2 |   bin := nuovo vettore[1..n]          // assegnamento
  |    oggetto-contenitore
3 |   rc := nuovo vettore[1..n]      // capacità residue
4 |   nb := 1                        // numero di contenitori
5 |   rc[1] ← c                      // inizializzare il primo
  |    contenitore
```

Algorithm: Best Fit (2/3)

function *BestFit*(n, w, c) (continua)

```
6   |   for i := 1 to n do
7   |   |   minrc := c + 1   // capacità residua minima
    |   |    trovata finora
8   |   |   s := −1                    // contenitore scelto
9   |   |   for b := 1 to nb do
10  |   |   |   if w[i] ≤ rc[b] e rc[b] < minrc then
11  |   |   |   |   s ← b
12  |   |   |   |   minrc ← rc[b]
13  |   |   |   end
14  |   |   end
```

Algorithm: Best Fit (3/3)

```
function BestFit(n, w, c) (continua)
    for i := 1 to n (continua) do
15      if s = −1 then
16          nb ← nb + 1              // aprire un nuovo
              contenitore
17          rc[nb] ← c
18          s ← nb
19      end
20      bin[i] ← s                   // assegnare l'oggetto
21      rc[s] ← rc[s] − w[i]
22  end
23  return bin, nb
24 end
```

Esempio: esecuzione di BF (1/4)

Consideriamo contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5\}$ di 5 oggetti ($n = 5$), con i seguenti pesi degli oggetti:

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
w_i	6	5	4	7	3

L'algoritmo BF produce la soluzione ammissibile delle prossime slide.

Esempio: esecuzione di BF (2/4)

1. Oggetto 1 (peso 6): il contenitore 1 è aperto all'inizio ($nb = 1$, $rc[1] = c = 10$) e $w[1] \leq rc[1]$, quindi $s = 1$ e BF pone l'oggetto 1 nel contenitore 1, e $rc[1] \leftarrow 10 - 6 = 4$.

2. Oggetto 2 (peso 5): $minrc := 11$, $s := -1$. $b = 1$: $w[2] = 5 \leq rc[1] = 4$? No. Nessun contenitore può accoglierlo ($s = -1$), quindi BF apre il contenitore 2 ($rc[2] \leftarrow 10$). L'oggetto 2 è posto nel contenitore 2, e $rc[2] \leftarrow 10 - 5 = 5$.

3. Oggetto 3 (peso 4): $minrc := 11$, $s := -1$. $b = 1$: $w[3] = 4 \leq rc[1] = 4$ e $rc[1] = 4 < 11$, quindi $s \leftarrow 1$, $minrc \leftarrow 4$. $b = 2$: $w[3] = 4 \leq rc[2] = 5$ e $rc[2] = 5 < minrc = 4$? No. Quindi $s = 1$: l'oggetto 3 è posto nel contenitore 1 (quello che lascia la capacità residua minima, 0, contro 1 del contenitore 2), e $rc[1] \leftarrow 4 - 4 = 0$.

Esempio: esecuzione di BF (3/4)

4. Oggetto 4 (peso 7): $b = 1$: $w[4] = 7 \leq rc[1] = 0$? No. $b = 2$: $w[4] = 7 \leq rc[2] = 5$? No. Nessun contenitore può accoglierlo ($s = -1$), quindi BF apre il contenitore 3 ($rc[3] \leftarrow 10$). L'oggetto 4 è posto nel contenitore 3, e $rc[3] \leftarrow 10 - 7 = 3$.

5. Oggetto 5 (peso 3): $minrc := 11$, $s := -1$. $b = 1$: $w[5] = 3 \leq rc[1] = 0$? No. $b = 2$: $w[5] = 3 \leq rc[2] = 5$ e $rc[2] = 5 < 11$, quindi $s \leftarrow 2$, $minrc \leftarrow 5$. $b = 3$: $w[5] = 3 \leq rc[3] = 3$ e $rc[3] = 3 < minrc = 5$, quindi $s \leftarrow 3$, $minrc \leftarrow 3$. Quindi $s = 3$: l'oggetto 5 è posto nel contenitore 3 (quello che lascia la capacità residua minima, 0, contro 2 del contenitore 2), e $rc[3] \leftarrow 3 - 3 = 0$.

Esempio: esecuzione di BF (4/4)

Quindi BF restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 1 & 3 & 3 \end{pmatrix}, \qquad nb = 3,$$

e quindi

$$BF(I) = nb = 3.$$

Esempio: schemi di carico dal vettore bin (1/2)

Consideriamo l'istanza dell'esempio precedente, su cui BF restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 1 & 3 & 3 \end{pmatrix}, \qquad nb = 3.$$

Gli schemi di carico sono:

$$P_1 = \{i \in \{1,2,3,4,5\} : bin_i = 1\} = \{1,3\},$$
$$P_2 = \{i \in \{1,2,3,4,5\} : bin_i = 2\} = \{2\},$$
$$P_3 = \{i \in \{1,2,3,4,5\} : bin_i = 3\} = \{4,5\}.$$

Esempio: schemi di carico dal vettore bin (2/2)

Sono esattamente gli insiemi di oggetti posti nei contenitori costruiti dall'algoritmo:

- il contenitore 1 contiene gli oggetti 1 e 3 con peso totale $w_1 + w_3 = 6 + 4 = 10$;
- il contenitore 2 contiene l'oggetto 2 con peso $w_2 = 5$;
- il contenitore 3 contiene gli oggetti 4 e 5 con peso totale $w_4 + w_5 = 7 + 3 = 10$.

L'algoritmo Worst Fit

- Il *Worst Fit Algorithm* (indicato con WF) è un'euristica costruttiva per il BPP.
- Esamina gli oggetti nell'ordine in cui sono dati nell'istanza, mantenendo tutti i contenitori aperti finora.
- Per ogni oggetto, WF considera tutti i contenitori in cui l'oggetto entra e lo pone in quello con la *capacità residua massima* (in caso di pareggio, il contenitore aperto per primo); se nessun contenitore può accogliere l'oggetto, WF apre un nuovo contenitore e vi pone l'oggetto.

Perché Worst Fit

- L'idea è opposta a quella di BF: ponendo ogni oggetto nel contenitore più vuoto, WF evita di creare contenitori con una capacità residua così piccola che nessun oggetto futuro possa usarla, e mantiene lo spazio libero distribuito fra i contenitori aperti.

- WF è un algoritmo Any Fit: apre un nuovo contenitore solo quando l'oggetto non entra in nessuno dei contenitori aperti finora, e la regola lasciata libera dalla definizione di algoritmo Any Fit è qui fissata prendendo il contenitore con la capacità residua massima.

Ora: lo pseudocodice di WF.

Algorithm: Worst Fit (1/3)

Input : istanza del BPP I con n oggetti, pesi $w[1..n]$, capacità c

Output: assegnamento oggetto-contenitore $bin[1..n]$ e numero di contenitori usati nb

```
1 function WorstFit(n, w, c)
2 |   bin := nuovo vettore[1..n]          // assegnamento
  |    oggetto-contenitore
3 |   rc := nuovo vettore[1..n]      // capacità residue
4 |   nb := 1                        // numero di contenitori
5 |   rc[1] ← c                      // inizializzare il primo
  |    contenitore
```

Algorithm: Worst Fit (2/3)

function *WorstFit*(n, w, c) (continua)

```
6   |   for i := 1 to n do
7   |   |   maxrc := −1   // capacità residua massima
    |   |    trovata finora
8   |   |   s := −1                    // contenitore scelto
9   |   |   for b := 1 to nb do
10  |   |   |   if w[i] ≤ rc[b] e rc[b] > maxrc then
11  |   |   |   |   s ← b
12  |   |   |   |   maxrc ← rc[b]
13  |   |   |   end
14  |   |   end
```

Algorithm: Worst Fit (3/3)

```
function WorstFit(n, w, c) (continua)
    for i := 1 to n (continua) do
15      if s = −1 then
16          nb ← nb + 1            // aprire un nuovo
                contenitore
17          rc[nb] ← c
18          s ← nb
19      end
20      bin[i] ← s                 // assegnare l'oggetto
21      rc[s] ← rc[s] − w[i]
22  end
23  return bin, nb
24 end
```

Esempio: esecuzione di WF (1/4)

Consideriamo contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5\}$ di 5 oggetti ($n = 5$), con i seguenti pesi degli oggetti:

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
w_i	6	5	4	7	3

L'algoritmo WF produce la soluzione ammissibile delle prossime slide.

Esempio: esecuzione di WF (2/4)

1. Oggetto 1 (peso 6): il contenitore 1 è aperto all'inizio ($nb = 1$, $rc[1] = c = 10$) e $w[1] \leq rc[1]$, quindi $s = 1$ e WF pone l'oggetto 1 nel contenitore 1, e $rc[1] \leftarrow 10 - 6 = 4$.

2. Oggetto 2 (peso 5): $maxrc := -1$, $s := -1$. $b = 1$: $w[2] = 5 \leq rc[1] = 4$? No. Nessun contenitore può accoglierlo ($s = -1$), quindi WF apre il contenitore 2 ($rc[2] \leftarrow 10$). L'oggetto 2 è posto nel contenitore 2, e $rc[2] \leftarrow 10 - 5 = 5$.

3. Oggetto 3 (peso 4): $maxrc := -1$, $s := -1$. $b = 1$: $w[3] = 4 \leq rc[1] = 4$ e $rc[1] = 4 > -1$, quindi $s \leftarrow 1$, $maxrc \leftarrow 4$. $b = 2$: $w[3] = 4 \leq rc[2] = 5$ e $rc[2] = 5 > maxrc = 4$, quindi $s \leftarrow 2$, $maxrc \leftarrow 5$. Quindi $s = 2$: l'oggetto 3 è posto nel contenitore 2 (quello con la capacità residua massima), e $rc[2] \leftarrow 5 - 4 = 1$.

Esempio: esecuzione di WF (3/4)

4. Oggetto 4 (peso 7): $b = 1$: $w[4] = 7 \leq rc[1] = 4$? No. $b = 2$: $w[4] = 7 \leq rc[2] = 1$? No. Nessun contenitore può accoglierlo ($s = -1$), quindi WF apre il contenitore 3 ($rc[3] \leftarrow 10$). L'oggetto 4 è posto nel contenitore 3, e $rc[3] \leftarrow 10 - 7 = 3$.

5. Oggetto 5 (peso 3): $maxrc := -1$, $s := -1$. $b = 1$: $w[5] = 3 \leq rc[1] = 4$ e $rc[1] = 4 > -1$, quindi $s \leftarrow 1$, $maxrc \leftarrow 4$. $b = 2$: $w[5] = 3 \leq rc[2] = 1$? No. $b = 3$: $w[5] = 3 \leq rc[3] = 3$ e $rc[3] = 3 > maxrc = 4$? No. Quindi $s = 1$: l'oggetto 5 è posto nel contenitore 1 (quello con la capacità residua massima), e $rc[1] \leftarrow 4 - 3 = 1$.

Esempio: esecuzione di WF (4/4)

Quindi WF restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 2 & 3 & 1 \end{pmatrix}, \qquad nb = 3,$$

e quindi

$$WF(I) = nb = 3.$$

Esempio: schemi di carico dal vettore bin (1/2)

Consideriamo l'istanza dell'esempio precedente, su cui WF restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 2 & 3 & 1 \end{pmatrix}, \qquad nb = 3.$$

Gli schemi di carico sono:

$$P_1 = \{i \in \{1,2,3,4,5\} : bin_i = 1\} = \{1,5\},$$
$$P_2 = \{i \in \{1,2,3,4,5\} : bin_i = 2\} = \{2,3\},$$
$$P_3 = \{i \in \{1,2,3,4,5\} : bin_i = 3\} = \{4\}.$$

Esempio: schemi di carico dal vettore bin (2/2)

Sono esattamente gli insiemi di oggetti posti nei contenitori costruiti dall'algoritmo:

- il contenitore 1 contiene gli oggetti 1 e 5 con peso totale $w_1 + w_5 = 6 + 3 = 9$;
- il contenitore 2 contiene gli oggetti 2 e 3 con peso totale $w_2 + w_3 = 5 + 4 = 9$;
- il contenitore 3 contiene l'oggetto 4 con peso $w_4 = 7$.

Riepilogo (1/2)

La classe Any Fit:

- mantiene tutti i contenitori aperti finora, e apre un nuovo contenitore solo quando l'oggetto non entra da nessuna parte
- la regola di scelta è libera: ogni regola dà un membro della famiglia
- gli oggetti di due contenitori di una soluzione Any Fit non entrano mai insieme

Next Fit:

- non è un algoritmo Any Fit: su $w = (3, 8, 3, 8)$ usa 4 contenitori, ogni algoritmo Any Fit 3

Riepilogo (2/2)

Tre membri della classe:

- First Fit: il primo contenitore in cui l'oggetto entra
- Best Fit: il contenitore con la capacità residua minima
- Worst Fit: il contenitore con la capacità residua massima
- pareggi: il contenitore aperto per primo

Sull'istanza $w = (6, 5, 4, 7, 3)$, $c = 10$:

- 3 contenitori ciascuno, con tre assegnamenti diversi: $(1, 2, 1, 3, 2)$, $(1, 2, 1, 3, 3)$, $(1, 2, 2, 3, 1)$

Proprietà di riempimento a metà

Introduzione: Proprietà di riempimento a metà

- In questa sezione definiamo una proprietà strutturale di una soluzione del BPP: tutti i contenitori tranne al più uno sono pieni per più della metà
- Dimostriamo che ogni algoritmo Any Fit produce soluzioni con questa proprietà
- Questa proprietà è esattamente ciò che serve, nella prossima sezione, per dimostrare la garanzia 2

La proprietà di riempimento a metà (1/2)

- I risultati di questa sezione non dipendono dalla regola usata per scegliere il contenitore.
- Si dimostrano usando *solo* la condizione Any Fit, e valgono quindi per ogni algoritmo della famiglia, compresi FF, BF e WF.

La proprietà di riempimento a metà (2/2)

- La **proprietà di riempimento a metà** di una soluzione ammissibile del BPP afferma che, con l'eccezione di al più un contenitore, tutti i contenitori sono riempiti per più della metà della loro capacità o, equivalentemente, che per ogni coppia di contenitori distinti almeno uno dei due è riempito per più della metà della capacità.

- La proprietà di riempimento a metà da sola *non* garantisce che due contenitori non si possano fondere: per esempio, con $c = 10$ due contenitori di peso totale 6 e 1 soddisfano la proprietà ma si possono fondere in un solo contenitore.

- Il suo ruolo è diverso: è esattamente ciò che serve per dimostrare una garanzia di prestazione relativa pari a 2.

Definizione 2: Proprietà di riempimento a metà (1/2)

Data una soluzione ammissibile del BPP con k contenitori e schemi di carico $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ (dove P_b è l'insieme degli oggetti posti nel contenitore b, per $b \in \{1, 2, \ldots, k\}$), diciamo che la soluzione soddisfa la ***proprietà di riempimento a metà*** *se vale una delle seguenti condizioni equivalenti:*

- *esiste al più un contenitore $b \in \{1, 2, \ldots, k\}$ tale che*

$$\sum_{i \in P_b} w_i \leq \frac{c}{2}.$$

Definizione: Proprietà di riempimento a metà (2/2)

- *per ogni coppia di contenitori distinti* $b, b' \in \{1, 2, \ldots, k\}$, *almeno uno dei due è riempito per più della metà della sua capacità, cioè*

$$\sum_{i \in P_b} w_i > \frac{c}{2} \qquad o \qquad \sum_{i \in P_{b'}} w_i > \frac{c}{2}.$$

Esempio: proprietà di riempimento a metà (1/3)

Consideriamo contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4\}$ di 4 oggetti ($n = 4$), con i seguenti pesi degli oggetti:

	$i = 1$	$i = 2$	$i = 3$	$i = 4$
w_i	5	5	5	5

Inizialmente $nb := 1$, $rc := c$. L'algoritmo NF produce la soluzione ammissibile della prossima slide.

Esempio: proprietà di riempimento a metà (2/3)

1. Oggetto 1 (peso $w[1] = 5$): $w[1] = 5 \leq rc = 10$, quindi non si apre un nuovo contenitore. L'oggetto 1 è posto nel contenitore $nb = 1$, e $rc \leftarrow 10 - 5 = 5$.

2. Oggetto 2 (peso $w[2] = 5$): $w[2] = 5 \leq rc = 5$, quindi non si apre un nuovo contenitore. L'oggetto 2 è posto nel contenitore $nb = 1$, e $rc \leftarrow 5 - 5 = 0$.

3. Oggetto 3 (peso $w[3] = 5$): $w[3] = 5 > rc = 0$, quindi $nb \leftarrow 2$ e $rc \leftarrow c = 10$. L'oggetto 3 è posto nel contenitore $nb = 2$, e $rc \leftarrow 10 - 5 = 5$.

4. Oggetto 4 (peso $w[4] = 5$): $w[4] = 5 \leq rc = 5$, quindi non si apre un nuovo contenitore. L'oggetto 4 è posto nel contenitore $nb = 2$, e $rc \leftarrow 5 - 5 = 0$.

Esempio: proprietà di riempimento a metà (3/3)

Quindi NF restituisce:

$$bin = \begin{pmatrix} 1 & 1 & 2 & 2 \end{pmatrix}, \qquad nb = 2.$$

Gli schemi di carico sono $P_1 = \{1, 2\}$, $P_2 = \{3, 4\}$, con pesi totali:

$$\sum_{i \in P_1} w_i = 5 + 5 = 10 > \frac{c}{2} = 5, \qquad \sum_{i \in P_2} w_i = 5 + 5 = 10 > \frac{c}{2} = 5.$$

Poiché *nessun* contenitore ha peso totale al più $\frac{c}{2}$, la soluzione **soddisfa** la proprietà di riempimento a metà.

Proposizione 1

Per ogni istanza del BPP, le soluzioni euristiche prodotte da qualsiasi algoritmo Any Fit soddisfano la proprietà di riempimento a metà. In particolare questo vale per FF, BF e WF.

- La dimostrazione è per assurdo: due contenitori pieni al più per metà conterrebbero oggetti che entrano insieme in un solo contenitore.
- Questo è escluso dalla proprietà della classe Any Fit dimostrata nella sezione precedente.

Dimostrazione (1/2)

Sia $\mathscr{A}$ un algoritmo Any Fit, sia $k := \mathscr{A}(I)$, e siano $P_1, P_2, \ldots, P_k$ gli schemi di carico prodotti da $\mathscr{A}$, indicizzati nell'ordine di apertura dei contenitori. Se $k = 1$ non c'è niente da dimostrare, quindi supponiamo $k \geq 2$.

Supponiamo, per assurdo, che esistano due contenitori distinti b e b' tali che

$$\sum_{i \in P_b} w_i \leq \frac{c}{2} \qquad \text{e} \qquad \sum_{i \in P_{b'}} w_i \leq \frac{c}{2}.$$

Dimostrazione (2/2)

Sommando le due disuguaglianze si ottiene

$$\sum_{i \in P_b} w_i + \sum_{i \in P_{b'}} w_i \leq \frac{c}{2} + \frac{c}{2} = c,$$

che contraddice la proprietà della classe Any Fit dimostrata nella sezione precedente, cioè che gli oggetti di due contenitori distinti non entrano mai insieme in un solo contenitore.

Di conseguenza può esserci al più un contenitore con peso totale al più $\frac{c}{2}$.

Si usa solo la condizione Any Fit, quindi l'enunciato vale per ogni algoritmo della classe. □

Esempio: proprietà di riempimento a metà per FF (1/3)

Come illustrazione della proposizione precedente, consideriamo contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4\}$ di 4 oggetti ($n = 4$), con i seguenti pesi degli oggetti:

	$i = 1$	$i = 2$	$i = 3$	$i = 4$
w_i	7	7	3	4

Inizialmente $nb := 1$, $rc[1] := c = 10$. L'algoritmo FF produce la soluzione ammissibile della prossima slide.

Esempio: proprietà di riempimento a metà per FF (2/3)

1. Oggetto 1 (peso 7): $b = 1$: $w[1] = 7 \leq rc[1] = 10$? Sì, quindi $s \leftarrow 1$. L'oggetto 1 è posto nel contenitore 1, e $rc[1] \leftarrow 10 - 7 = 3$.

2. Oggetto 2 (peso 7): $b = 1$: $w[2] = 7 \leq rc[1] = 3$? No. Nessun contenitore può accoglierlo, quindi FF apre il contenitore 2 ($rc[2] \leftarrow 10$). L'oggetto 2 è posto nel contenitore 2, e $rc[2] \leftarrow 10 - 7 = 3$.

3. Oggetto 3 (peso 3): $b = 1$: $w[3] = 3 \leq rc[1] = 3$? Sì, quindi $s \leftarrow 1$. L'oggetto 3 è posto nel contenitore 1, e $rc[1] \leftarrow 3 - 3 = 0$.

4. Oggetto 4 (peso 4): $b = 1$: $w[4] = 4 \leq rc[1] = 0$? No. $b = 2$: $w[4] = 4 \leq rc[2] = 3$? No. Nessun contenitore può accoglierlo, quindi FF apre il contenitore 3 ($rc[3] \leftarrow 10$). L'oggetto 4 è posto nel contenitore 3, e $rc[3] \leftarrow 10 - 4 = 6$.

Esempio: proprietà di riempimento a metà per FF (3/3)

Quindi FF restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 1 & 3 \end{pmatrix}, \qquad nb = 3.$$

Gli schemi di carico sono $P_1 = \{1, 3\}$, $P_2 = \{2\}$, $P_3 = \{4\}$, con pesi totali:

$$\sum_{i \in P_1} w_i = 7 + 3 = 10 > \frac{c}{2} = 5,$$

$$\sum_{i \in P_2} w_i = 7 > \frac{c}{2} = 5,$$

$$\sum_{i \in P_3} w_i = 4 \leq \frac{c}{2} = 5.$$

Poiché c'è *esattamente un* contenitore (il contenitore 3) con peso totale al più $\frac{c}{2}$, la soluzione **soddisfa** la proprietà di riempimento a metà, come garantito dalla proposizione precedente, poiché FF è un algoritmo Any Fit.

Next Fit e la proprietà di riempimento a metà

Il Next Fit può produrre soluzioni con molti contenitori riempiti per meno della metà della capacità,

e quindi non soddisfa la proprietà di riempimento a metà.

Esempio: proprietà di riempimento a metà per NF (1/2)

Consideriamo di nuovo l'istanza dell'esempio su NF e la classe Any Fit, con capacità dei contenitori $c = 10$ e pesi degli oggetti $w = (3, 8, 3, 8)$, su cui NF apre un nuovo contenitore per ogni oggetto e restituisce:

$$bin = \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}, \qquad nb = 4.$$

Gli schemi di carico sono:

$$\begin{aligned} P_1 &= \{i \in \{1,2,3,4\} : bin_i = 1\} = \{1\}, \\ P_2 &= \{i \in \{1,2,3,4\} : bin_i = 2\} = \{2\}, \\ P_3 &= \{i \in \{1,2,3,4\} : bin_i = 3\} = \{3\}, \\ P_4 &= \{i \in \{1,2,3,4\} : bin_i = 4\} = \{4\}. \end{aligned}$$

Esempio: proprietà di riempimento a metà per NF (2/2)

I pesi totali nei contenitori sono:

$$\sum_{i \in P_1} w_i = 3 \leq \frac{c}{2} = 5, \qquad \sum_{i \in P_2} w_i = 8 > \frac{c}{2} = 5,$$

$$\sum_{i \in P_3} w_i = 3 \leq \frac{c}{2} = 5, \qquad \sum_{i \in P_4} w_i = 8 > \frac{c}{2} = 5.$$

Poiché ci sono *due* contenitori (i contenitori 1 e 3) con peso totale al più $\frac{c}{2}$, la soluzione **non** soddisfa la proprietà di riempimento a metà.

Riepilogo

La proprietà di riempimento a metà:

- tutti i contenitori tranne al più uno hanno peso totale maggiore di $\frac{c}{2}$
- equivalentemente: di due contenitori distinti, almeno uno è pieno per più della metà
- non dice che nessuna coppia di contenitori si può fondere

Chi la soddisfa:

- ogni algoritmo Any Fit (FF, BF, WF), qualunque sia la regola di scelta
- non NF: su $w = (3, 8, 3, 8)$ due contenitori hanno peso totale al più $\frac{c}{2}$

Garanzie di prestazione

Introduzione: Garanzie di prestazione

- In questa sezione determiniamo il rapporto di prestazione nel caso peggiore dei quattro algoritmi
- Dimostriamo la garanzia 2 e mostriamo famiglie di istanze su cui è tight
- Enunciamo il rapporto migliore $\frac{17}{10}$ di First Fit e Best Fit, e vediamo che cosa la sola condizione Any Fit può dare e che cosa no

Il piano della sezione

- Next Fit non è un algoritmo Any Fit, quindi richiede un argomento a sé.
- Per i tre algoritmi Any Fit il bound $\mathscr{A}(I) \leq 2\,z(I)$ segue subito dalla proprietà di riempimento a metà, senza alcuna ipotesi sulla regola usata per scegliere il contenitore.
- La domanda è quanto sia tight questo bound: per WF il fattore 2 non si può migliorare, mentre FF e BF rispettano un bound strettamente migliore.

Proposizione 2

Sia $\mathscr{A}$ un algoritmo euristico per il BPP che produce soluzioni che soddisfano la proprietà di riempimento a metà. Allora, per ogni istanza I del BPP, abbiamo

$$\frac{\mathscr{A}(I)}{z(I)} \leq 2.$$

- La dimostrazione confronta due bound sul peso totale degli oggetti: un lower bound dalla proprietà di riempimento a metà, un upper bound dalla soluzione ottima.

Dimostrazione (1/3)

Sia $k := \mathscr{A}(I)$, e sia $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ l'insieme degli schemi di carico prodotti da $\mathscr{A}$, indicizzati nell'ordine di apertura dei contenitori.

Poiché la soluzione soddisfa la proprietà di riempimento a metà, esiste al più un contenitore $b \in \{1, 2, \ldots, k\}$ tale che

$$\sum_{i \in P_b} w_i \leq \frac{c}{2}.$$

Di conseguenza, almeno $k - 1$ contenitori hanno peso totale strettamente maggiore di $\frac{c}{2}$. Sommando su questi contenitori, otteniamo

$$\sum_{i \in \mathscr{I}} w_i \; > \; (k-1)\,\frac{c}{2}.$$

Dimostrazione (2/3)

D'altra parte, ogni impacchettamento ammissibile che usa $z(I)$ contenitori soddisfa

$$\sum_{i \in \mathscr{I}} w_i \;\leq\; z(I)\, c.$$

Combinando le due disuguaglianze si ottiene

$$(k-1)\,\frac{c}{2} \;<\; \sum_{i \in \mathscr{I}} w_i \;\leq\; z(I)\, c$$

$$\Longrightarrow \quad k - 1 \;<\; 2\, z(I)$$

$$\Longrightarrow \quad k \;<\; 2\, z(I) + 1.$$

Dimostrazione (3/3)

Poiché sia k sia $z(I)$ sono interi positivi, abbiamo

$$k \leq 2\, z(I).$$

Dividendo per $z(I) \geq 1$ si ottiene

$$\frac{k}{z(I)} = \frac{\mathscr{A}(I)}{z(I)} \leq 2.$$

□

La garanzia degli algoritmi Any Fit

- Poiché ogni algoritmo Any Fit produce soluzioni che soddisfano la proprietà di riempimento a metà, segue immediatamente che

$$\frac{\mathscr{A}(I)}{z(I)} \leq 2 \qquad \text{per ogni algoritmo Any Fit } \mathscr{A}$$

e per ogni istanza I del BPP.

- In particolare

$$\frac{FF(I)}{z(I)} \leq 2, \qquad \frac{BF(I)}{z(I)} \leq 2 \qquad \text{e} \qquad \frac{WF(I)}{z(I)} \leq 2,$$

e quindi FF, BF e WF sono algoritmi di 2-approssimazione.

Proposizione 3

Per ogni istanza I del BPP, abbiamo:

$$\frac{NF(I)}{z(I)} \leq 2.$$

- NF non soddisfa la proprietà di riempimento a metà, quindi la proposizione precedente non si applica.
- La dimostrazione usa invece due contenitori *consecutivi*: i loro oggetti non entrano mai insieme.

Dimostrazione (1/4)

Sia $k := NF(I)$, e siano $P_1, P_2, \ldots, P_k$ gli schemi di carico prodotti da NF, indicizzati nell'ordine di apertura dei contenitori.

Fissiamo un qualsiasi $b \in \{1, 2, \ldots, k-1\}$. Lo schema di carico P_b corrisponde a un contenitore che viene chiuso quando l'oggetto successivo (chiamiamolo f) non vi entra, quindi

$$\sum_{i \in P_b} w_i + w_f > c.$$

Lo stesso oggetto f è il primo oggetto posto nel contenitore successivo, il cui schema di carico è P_{b+1}, quindi

$$w_f \leq \sum_{i \in P_{b+1}} w_i.$$

Dimostrazione (2/4)

Di conseguenza,

$$\sum_{i \in P_b} w_i + \sum_{i \in P_{b+1}} w_i \; > \; c.$$

Consideriamo ora le coppie consecutive disgiunte $(P_1, P_2), (P_3, P_4), \ldots$. Sommando la disuguaglianza precedente sulle $\lfloor k/2 \rfloor$ coppie si ottiene

$$\underbrace{\sum_{b=1}^{k} \sum_{i \in P_b} w_i}_{= \sum_{i \in \mathscr{I}} w_i \; \leq \; z(I)\, c} \; > \left\lfloor \frac{k}{2} \right\rfloor c \quad \Longrightarrow \quad z(I) \; > \left\lfloor \frac{k}{2} \right\rfloor .$$

Dimostrazione (3/4)

Poiché $z(I)$ è intero, segue che

$$z(I) \geq \left\lfloor \frac{k}{2} \right\rfloor + 1.$$

Allora

$$\begin{cases} \text{se } k \text{ è pari, allora } \left\lfloor \frac{k}{2} \right\rfloor = \frac{k}{2} \implies z(I) \geq \frac{k}{2} + 1 \\ \qquad\qquad\qquad\qquad\qquad\qquad \implies k \leq 2z(I) - 2 < 2z(I), \\ \text{se } k \text{ è dispari, allora } \left\lfloor \frac{k}{2} \right\rfloor = \frac{k-1}{2} \implies z(I) \geq \frac{k+1}{2} \\ \qquad\qquad\qquad\qquad\qquad\qquad \implies k \leq 2z(I) - 1 < 2z(I). \end{cases}$$

Dimostrazione (4/4)

Di conseguenza $k < 2z(I)$, e dividendo per $z(I) \geq 1$ si ottiene

$$\frac{k}{z(I)} = \frac{NF(I)}{z(I)} \leq 2.$$

□

Proposizione 4

Esistono istanze I_n del BPP tali che:

$$\frac{NF(I_n)}{z(I_n)} \to 2 \qquad \textit{per} \qquad n \to +\infty.$$

- La dimostrazione costruisce una famiglia di istanze che alternano un oggetto grande e un oggetto piccolo.
- NF pone ogni oggetto grande in un contenitore a sé, mentre l'ottimo pone due oggetti grandi per contenitore.

Dimostrazione (1/4)

Fissiamo un intero $n \geq 8$ divisibile per 4 e consideriamo l'istanza I_n del BPP con capacità dei contenitori $c = n$ e insieme di oggetti $\mathscr{I} = \{1, 2, \ldots, n\}$, dove i pesi degli oggetti sono

$$w_i = \begin{cases} \frac{n}{2}, & \text{se } i \text{ è dispari,} \\ 2, & \text{se } i \text{ è pari.} \end{cases}$$

Dimostrazione (2/4)

Gli oggetti sono esaminati da NF nell'ordine naturale $1, 2, \ldots, n$. Per ogni i dispari,

$$w_i + w_{i+1} = \frac{n}{2} + 2 < n,$$

quindi NF pone gli oggetti i e $i+1$ nello stesso contenitore. Lo spazio rimanente è

$$n - \left(\frac{n}{2} + 2\right) = \frac{n}{2} - 2 < \frac{n}{2},$$

quindi l'oggetto successivo, di peso $\frac{n}{2}$, non entra e NF apre un nuovo contenitore. Di conseguenza,

$$NF(I_n) = \frac{n}{2}.$$

Dimostrazione (3/4)

Ponendo gli $\frac{n}{2}$ oggetti di peso $\frac{n}{2}$ due per contenitore si ottengono $\frac{n}{4}$ contenitori pieni, e ponendo tutti gli $\frac{n}{2}$ oggetti piccoli insieme (poiché $\frac{n}{2} \cdot 2 = n$) in un contenitore aggiuntivo si ha $z(I_n) \leq \frac{n}{4} + 1$.

D'altra parte,

$$z(I_n) \geq \left\lceil \frac{\sum_{i \in \mathscr{I}} w_i}{c} \right\rceil = \left\lceil \frac{\frac{n}{2}\ \frac{n}{2} + \frac{n}{2}\ 2}{n} \right\rceil = \left\lceil \frac{n}{4} + 1 \right\rceil = \frac{n}{4} + 1.$$

Quindi

$$z(I_n) = \frac{n}{4} + 1.$$

Dimostrazione (4/4)

Di conseguenza,

$$\lim_{n\to\infty} \frac{NF(I_n)}{z(I_n)} = \lim_{n\to\infty} \frac{\frac{n}{2}}{\frac{n}{4}+1} = \lim_{n\to\infty} \frac{\frac{n}{2}}{\frac{n+4}{4}}$$

$$= \lim_{n\to\infty} \frac{2n}{n+4} = \lim_{n\to\infty} \frac{2}{1+\frac{4}{n}} = 2.$$

□

Il rapporto di prestazione nel caso peggiore di NF

Dalle ultime due proposizioni segue che
il rapporto di prestazione nel caso peggiore di NF è due,
cioè abbiamo

$$r(NF) = 2$$

Proposizione 5

Esistono istanze I_n del BPP tali che:

$$\frac{WF(I_n)}{z(I_n)} \to 2 \qquad \textit{per} \qquad n \to +\infty.$$

- Sappiamo già che $\frac{WF(I)}{z(I)} \leq 2$ per ogni istanza, poiché WF è un algoritmo Any Fit.
- La dimostrazione usa la stessa famiglia di istanze di NF, e un'induzione sul numero di oggetti esaminati.

Dimostrazione (1/5)

Fissiamo un intero $n \geq 8$ divisibile per 4 e consideriamo l'istanza I_n del BPP con capacità dei contenitori $c = n$ e insieme di oggetti $\mathscr{I} = \{1, 2, \ldots, n\}$, dove i pesi degli oggetti sono

$$w_i = \begin{cases} \frac{n}{2}, & \text{se } i \text{ è dispari,} \\ 2, & \text{se } i \text{ è pari,} \end{cases}$$

cioè la stessa istanza usata sopra per NF. Gli oggetti sono esaminati da WF nell'ordine naturale $1, 2, \ldots, n$.

Mostriamo per induzione che, dopo che sono stati esaminati i primi $2k$ oggetti, WF ha aperto esattamente k contenitori, ciascuno con capacità residua $\frac{n}{2} - 2$.

Dimostrazione (2/5)

Per $k = 1$, l'oggetto 1 (di peso $\frac{n}{2}$) è posto nel contenitore 1, la cui capacità residua diventa $\frac{n}{2}$; poi l'oggetto 2 (di peso 2) è posto nel contenitore 1, l'unico contenitore aperto, la cui capacità residua diventa $\frac{n}{2} - 2$.

Supponiamo che l'affermazione valga per un certo k, e consideriamo gli oggetti $2k + 1$ e $2k + 2$.

L'oggetto $2k + 1$ ha peso $\frac{n}{2}$, mentre ogni contenitore aperto ha capacità residua $\frac{n}{2} - 2 < \frac{n}{2}$: l'oggetto non entra da nessuna parte, quindi WF apre il contenitore $k + 1$, la cui capacità residua diventa $\frac{n}{2}$.

Dimostrazione (3/5)

L'oggetto $2k+2$ ha peso 2 ed entra in ogni contenitore aperto; poiché il contenitore $k+1$ è quello con la capacità residua *massima*, WF vi pone l'oggetto, e la capacità residua del contenitore $k+1$ diventa $\frac{n}{2}-2$. Questo dimostra l'affermazione per $k+1$.

È proprio la regola Worst Fit a mantenere ogni contenitore a capacità residua $\frac{n}{2}-2$: gli oggetti piccoli sono sempre posti nel contenitore più vuoto, quindi nessun contenitore riceve mai due oggetti di peso $\frac{n}{2}$.

Prendendo $k=\frac{n}{2}$ otteniamo

$$WF(I_n)=\frac{n}{2}.$$

Dimostrazione (4/5)

Ponendo gli $\frac{n}{2}$ oggetti di peso $\frac{n}{2}$ due per contenitore si ottengono $\frac{n}{4}$ contenitori pieni, e ponendo tutti gli $\frac{n}{2}$ oggetti piccoli insieme (poiché $\frac{n}{2} \cdot 2 = n$) in un contenitore aggiuntivo si ha $z(I_n) \leq \frac{n}{4} + 1$.

D'altra parte,

$$z(I_n) \geq \left\lceil \frac{\sum_{i \in \mathscr{J}} w_i}{c} \right\rceil = \left\lceil \frac{\frac{n}{2}\ \frac{n}{2} + \frac{n}{2}\ 2}{n} \right\rceil = \left\lceil \frac{n}{4} + 1 \right\rceil = \frac{n}{4} + 1,$$

quindi

$$z(I_n) = \frac{n}{4} + 1.$$

Dimostrazione (5/5)

Di conseguenza,

$$\lim_{n\to\infty} \frac{WF(I_n)}{z(I_n)} = \lim_{n\to\infty} \frac{\frac{n}{2}}{\frac{n}{4}+1}$$

$$= \lim_{n\to\infty} \frac{2n}{n+4} = \lim_{n\to\infty} \frac{2}{1+\frac{4}{n}} = 2.$$

□

Il rapporto di prestazione nel caso peggiore di WF

Dalla garanzia della proprietà di riempimento a metà
e dall'ultima proposizione segue che
il rapporto di prestazione nel caso peggiore di WF è due,
cioè abbiamo

$$r(WF) = 2$$

Per WF il fattore 2 non si può quindi migliorare.

WF e NF

- Abbiamo $r(WF) = r(NF) = 2$: nel caso peggiore WF non è migliore di NF.
- I due algoritmi sono però lontani dall'essere equivalenti. WF non apre mai un nuovo contenitore finché l'oggetto entra ancora in uno dei contenitori aperti finora, mentre NF chiude un contenitore appena un oggetto non vi entra e non lo riconsidera mai.
- Di conseguenza NF può lasciare un numero arbitrario di contenitori riempiti per meno della metà della loro capacità, mentre WF, come mostrato nella sezione sulla proprietà di riempimento a metà, può lasciarne al più uno.
- Il valore $r(WF) = 2$ è dovuto a

  D.S. Johnson, *Fast algorithms for bin packing*, Journal of Computer and System Sciences, 8(3):272–314, 1974.

Proposizione 6

Per ogni istanza I del BPP, abbiamo:

$$FF(I) \leq \frac{17}{10} z(I) \quad e \quad BF(I) \leq \frac{17}{10} z(I).$$

- Le regole di scelta di FF e BF danno un bound strettamente migliore del fattore 2 della classe Any Fit.
- La dimostrazione è lunga, ed è omessa: diamo gli articoli in cui si trova.

Dimostrazione (1/2)

Dimostrazione omessa. I due bound sono stati dimostrati per la prima volta nella forma più debole

$$FF(I) \leq \frac{17}{10} z(I) + 2 \qquad \text{e} \qquad BF(I) \leq \frac{17}{10} z(I) + 2$$

in

D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey e R.L. Graham, *Worst-case performance bounds for simple one-dimensional packing algorithms*, SIAM Journal on Computing, 3(4):299–325, 1974.

Dimostrazione (2/2)

La costante additiva è stata poi eliminata, nella forma enunciata sopra, rispettivamente in

G. Dósa e J. Sgall, *First Fit bin packing: a tight analysis*, STACS 2013, pp. 538–549,

G. Dósa e J. Sgall, *Optimal analysis of Best Fit bin packing*, ICALP 2014, pp. 429–441. □

Proposizione 7

Esistono istanze I_n del BPP tali che:

$$\frac{FF(I_n)}{z(I_n)} \to \frac{17}{10} \quad e \quad \frac{BF(I_n)}{z(I_n)} \to \frac{17}{10} \qquad per \qquad n \to +\infty.$$

- Insieme alla proposizione precedente, mostra che il fattore $\frac{17}{10}$ di FF e BF non si può migliorare.
- Anche questa dimostrazione è omessa: diamo i lavori in cui si trovano le famiglie di istanze.

Dimostrazione

Dimostrazione omessa. Famiglie di istanze su cui il rapporto di FF e BF tende a $\frac{17}{10}$ sono date nell'articolo di Johnson, Demers, Ullman, Garey e Graham citato sopra, e in

D.S. Johnson, *Near-optimal bin packing algorithms*, tesi di dottorato, Massachusetts Institute of Technology, 1973. □

I rapporti di prestazione nel caso peggiore di FF e BF

Dalle ultime due proposizioni segue che
il rapporto di prestazione nel caso peggiore di FF e BF
è diciassette decimi, cioè abbiamo

$$r(FF) = r(BF) = \frac{17}{10}$$

A differenza di WF, FF e BF restano strettamente sotto
il fattore 2 che la sola condizione Any Fit garantisce.

Che cosa dà la condizione Any Fit

- La sola condizione Any Fit garantisce la proprietà di riempimento a metà, e quindi il bound

$$\mathscr{A}(I) \leq 2\, z(I)$$

  per ogni algoritmo $\mathscr{A}$ della classe.

- Questo bound non si può migliorare per la classe nel suo complesso: WF è un algoritmo Any Fit e, come mostrato sopra, $r(WF) = 2$.

Che cosa lascia libero la condizione Any Fit (1/2)

- La regola usata per scegliere uno dei contenitori in cui l'oggetto entra è ciò che distingue i membri della famiglia.
- Sulle istanze della proposizione di tightness per FF e BF il rapporto di FF e BF tende a $\frac{17}{10}$, mentre su quelle della proposizione di tightness per WF il rapporto di WF tende a 2: la sola regola di scelta spiega tutto il divario fra $\frac{17}{10}$ e 2.

Che cosa lascia libero la condizione Any Fit (2/2)

- È noto che nessun algoritmo della classe può scendere sotto $\frac{17}{10}$, quindi $\frac{17}{10}$ e 2 sono esattamente il comportamento migliore e quello peggiore che la condizione Any Fit consente, ed entrambi sono raggiunti, il primo da FF e BF, il secondo da WF.

- La classe Any Fit e i bound precedenti si devono all'articolo di Johnson già citato per $r(WF) = 2$: in particolare, il lower bound $\frac{17}{10}$ per ogni algoritmo Any Fit è dovuto a quell'articolo (D.S. Johnson, 1974).

Riepilogo (1/2)

La garanzia 2:

- la proprietà di riempimento a metà dà $\mathscr{A}(I) \leq 2\,z(I)$, per ogni algoritmo Any Fit
- NF: due contenitori consecutivi non entrano mai insieme, quindi anche $NF(I) \leq 2\,z(I)$

Famiglie tight:

- $c = n$, pesi $\frac{n}{2}$ e 2 alternati: $NF(I_n) = WF(I_n) = \frac{n}{2}$ e $z(I_n) = \frac{n}{4} + 1$
- quindi $r(NF) = r(WF) = 2$

Riepilogo (2/2)

First Fit e Best Fit:

- $FF(I) \leq \frac{17}{10}\, z(I)$ e $BF(I) \leq \frac{17}{10}\, z(I)$ (dimostrazioni omesse)
- $r(FF) = r(BF) = \frac{17}{10}$

La classe Any Fit:

- $\frac{17}{10}$ e 2 sono il comportamento migliore e quello peggiore che la condizione Any Fit consente
- la sola regola di scelta spiega tutto il divario

Errori tipici

Introduzione: Errori tipici

- Raccogliamo gli errori che si fanno più spesso con gli algoritmi di questo capitolo
- Due riguardano l'esecuzione degli algoritmi, tre il significato dei risultati
- Controllateli prima di eseguire un algoritmo a mano o di leggere una garanzia

Errori tipici (1/3)

Osservazione 1: Credere che Next Fit sia un algoritmo Any Fit

NF non riconsidera mai un contenitore chiuso, quindi può aprire un nuovo contenitore per un oggetto che entrerebbe in uno più vecchio (come nell'esempio con pesi $(3, 8, 3, 8)$*).*

Osservazione 2: Scegliere il contenitore sbagliato in BF e WF

BF pone l'oggetto nel contenitore con la capacità residua minima fra i contenitori in cui entra, WF in quello con la capacità residua massima; in caso di pareggio, entrambi prendono il contenitore di indice minimo.

Errori tipici (2/3)

Osservazione 3: Leggere la proprietà di riempimento a metà come «nessuna coppia di contenitori si può fondere»

Due contenitori di peso totale 6 *e* 1 *con* $c = 10$ *soddisfano la proprietà e si possono fondere; la proprietà è ciò che dimostra la garanzia* 2*.*

Osservazione 4: Trattare $r = 2$ come un valore raggiunto

I rapporti nel caso peggiore di NF e WF sono estremi superiori: le famiglie di istanze si avvicinano a 2 *al crescere di* n*, e nessuna istanza lo raggiunge.*

Errori tipici (3/3)

Osservazione 5: Cambiare l'ordine degli oggetti

I quattro algoritmi esaminano gli oggetti nell'ordine in cui sono dati; ordinarli prima è un algoritmo diverso.

Riepilogo

Eseguire gli algoritmi:

- NF non riconsidera mai un contenitore chiuso: non è un algoritmo Any Fit
- BF: capacità residua minima, WF: massima; pareggi al contenitore di indice minimo
- mantenere l'ordine dato degli oggetti

Leggere i risultati:

- la proprietà di riempimento a metà non vieta di fondere due contenitori
- $r = 2$ è avvicinato da NF e WF, mai raggiunto

Implementazione in C++

Introduzione: Implementazione in C++

- I quattro algoritmi sono implementati in C++, una funzione ciascuno
- Guardiamo brevi estratti: Next Fit, e la scansione dei contenitori che distingue First Fit, Best Fit e Worst Fit
- Seguiamo passo per passo l'esecuzione di ogni funzione sull'istanza del capitolo

Indici e istanza (1/2)

- In questa sezione sull'implementazione in C++ usiamo indici a partire da 0 (oggetti e contenitori sono numerati a partire da 0). La logica algoritmica non cambia.

- Le esecuzioni passo per passo si riferiscono tutte alla stessa istanza del BPP, cioè contenitori di capacità $c = 10$ e un insieme $\mathscr{I} = \{1, 2, 3, 4, 5\}$ di 5 oggetti ($n = 5$), con i seguenti pesi degli oggetti:

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
w_i	6	5	4	7	3

Indici e istanza (2/2)

- Con gli indici a partire da 0 gli oggetti sono numerati $0, 1, 2, 3, 4$ e i loro pesi sono $w[0] = 6$, $w[1] = 5$, $w[2] = 4$, $w[3] = 7$ e $w[4] = 3$. Questa è l'istanza usata negli esempi delle sezioni precedenti, quindi le soluzioni trovate qui sono quelle già calcolate lì.

Next Fit in C++

```
1  int NextFit(int n, int c, int *w, int *bin)
2  {
3      int nb = 1;
4      int rc  = c;
5      for (int i = 0; i < n; i++)
6      {
7          if (w[i] > rc)
8          {
9              nb++;
10             rc = c;
11         }
12         bin[i] = nb - 1;
13         rc -= w[i];
14     }
15     return nb;
16 }
```

- `nb` è il contenitore corrente (numero di contenitori aperti), `rc` la sua capacità residua; l'oggetto `i` va nel contenitore `nb - 1`.

Esempio: esecuzione C++ di NF (1/2)

Inizializzazione

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	-	-	-	-	-

$nb = 1 \quad rc = 10$

Iterazione 1 – l'oggetto 0 è posto nel contenitore 0

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	-	-	-	-

$nb = 1 \quad rc = 4$

Iterazione 2 – l'oggetto 1 è posto nel contenitore 1

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	-	-	-

$nb = 2 \quad rc = 5$

Esempio: esecuzione C++ di NF (2/2)

Iterazione 3 – l'oggetto 2 è posto nel contenitore 1

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	1	-	-

$nb = 2 \quad rc = 1$

Iterazione 4 – l'oggetto 3 è posto nel contenitore 2

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	1	2	-

$nb = 3 \quad rc = 3$

Iterazione 5 – l'oggetto 4 è posto nel contenitore 2

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	1	2	2

$nb = 3 \quad rc = 0$

First Fit in C++: la scansione dei contenitori

```
int s = -1;
for (int b = 0; b < nb; b++)
{
    if (w[i] <= rc[b])
    {
        s = b;
        break;
    }
}
```

- `s` è il contenitore scelto per l'oggetto `i` (`-1` significa: non ancora trovato).
- I contenitori sono scanditi nell'ordine di apertura, e si prende il primo in cui l'oggetto entra.
- Il vettore `rc` memorizza le capacità residue dei contenitori (al più n contenitori), con `rc[0] = c` all'inizio.

First Fit in C++: aprire un nuovo contenitore

```
1 if (s == -1)
2 {
3     rc[nb] = c;
4     s = nb;
5     nb++;
6 }
7 bin[i] = s;
8 rc[s] -= w[i];
```

- Se nessun contenitore aperto può accogliere l'oggetto `i`, si apre un nuovo contenitore, con capacità residua `c`, e lo si sceglie.
- L'oggetto `i` è posto nel contenitore scelto, e la sua capacità residua è aggiornata.
- Queste righe sono le stesse in First Fit, Best Fit e Worst Fit: cambia solo la scansione dei contenitori.

Esempio: esecuzione C++ di FF (1/3)

Inizializzazione $nb = 1$

$bin[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
	-	-	-	-	-

rc$[b]$	$b = 0$	$b = 1$	$b = 2$	$b = 3$	$b = 4$
	10	-	-	-	-

Iterazione 1 – l'oggetto 0 è posto nel contenitore 0 $nb = 1$

$bin[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
	0	-	-	-	-

rc$[b]$	$b = 0$	$b = 1$	$b = 2$	$b = 3$	$b = 4$
	4	-	-	-	-

Esempio: esecuzione C++ di FF (2/3)

Iterazione 2 – l'oggetto 1 è posto nel contenitore 1 $nb = 2$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	-	-	-

rc$[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	4	5	-	-	-

Iterazione 3 – l'oggetto 2 è posto nel contenitore 0 $nb = 2$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	0	-	-

rc$[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	0	5	-	-	-

Esempio: esecuzione C++ di FF (3/3)

Iterazione 4 – l'oggetto 3 è posto nel contenitore 2 $nb = 3$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	0	2	-

rc$[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	0	5	3	-	-

Iterazione 5 – l'oggetto 4 è posto nel contenitore 1 $nb = 3$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	0	2	1

rc$[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	0	2	3	-	-

Best Fit in C++: la scansione dei contenitori

```
1  int s = -1;
2  int minrc = c + 1;
3  for (int b = 0; b < nb; b++)
4  {
5      if (w[i] <= rc[b] && rc[b] < minrc)
6      {
7          minrc = rc[b];
8          s = b;
9      }
10 }
```

- `minrc` è la capacità residua minima trovata finora.
- Fra i contenitori in cui l'oggetto `i` entra, si prende quello con la capacità residua minima.

Esempio: esecuzione C++ di BF (1/3)

Inizializzazione $nb = 1$

$bin[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
	-	-	-	-	-

rc$[b]$	$b = 0$	$b = 1$	$b = 2$	$b = 3$	$b = 4$
	10	-	-	-	-

Iterazione 1 – l'oggetto 0 è posto nel contenitore 0 $nb = 1$

$bin[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
	0	-	-	-	-

rc$[b]$	$b = 0$	$b = 1$	$b = 2$	$b = 3$	$b = 4$
	4	-	-	-	-

Esempio: esecuzione C++ di BF (2/3)

Iterazione 2 – l'oggetto 1 è posto nel contenitore 1 $nb = 2$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	-	-	-

rc$[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	4	5	-	-	-

Iterazione 3 – l'oggetto 2 è posto nel contenitore 0 $nb = 2$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	0	-	-

rc$[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	0	5	-	-	-

Esempio: esecuzione C++ di BF (3/3)

Iterazione 4 – l'oggetto 3 è posto nel contenitore 2 $nb = 3$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	0	2	-

$rc[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	0	5	3	-	-

Iterazione 5 – l'oggetto 4 è posto nel contenitore 2 $nb = 3$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	0	2	2

$rc[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	0	5	0	-	-

Worst Fit in C++: la scansione dei contenitori

```
1  int s = -1;
2  int maxrc = -1;
3  for (int b = 0; b < nb; b++)
4  {
5      if (w[i] <= rc[b] && rc[b] > maxrc)
6      {
7          maxrc = rc[b];
8          s = b;
9      }
10 }
```

- `maxrc` è la capacità residua massima trovata finora.
- Fra i contenitori in cui l'oggetto `i` entra, si prende quello con la capacità residua massima.

Esempio: esecuzione C++ di WF (1/3)

Inizializzazione $nb = 1$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	-	-	-	-	-

rc$[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	10	-	-	-	-

Iterazione 1 – l'oggetto 0 è posto nel contenitore 0 $nb = 1$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	-	-	-	-

rc$[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	4	-	-	-	-

Esempio: esecuzione C++ di WF (2/3)

Iterazione 2 – l'oggetto 1 è posto nel contenitore 1 $nb = 2$

$bin[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
	0	1	-	-	-

$\mathrm{rc}[b]$	$b = 0$	$b = 1$	$b = 2$	$b = 3$	$b = 4$
	4	5	-	-	-

Iterazione 3 – l'oggetto 2 è posto nel contenitore 1 $nb = 2$

$bin[i]$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
	0	1	1	-	-

$\mathrm{rc}[b]$	$b = 0$	$b = 1$	$b = 2$	$b = 3$	$b = 4$
	4	1	-	-	-

Esempio: esecuzione C++ di WF (3/3)

Iterazione 4 – l'oggetto 3 è posto nel contenitore 2 $nb = 3$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	1	2	-

rc$[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	4	1	3	-	-

Iterazione 5 – l'oggetto 4 è posto nel contenitore 0 $nb = 3$

$bin[i]$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
	0	1	1	2	0

rc$[b]$	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
	1	1	3	-	-

I codici del capitolo

- Tutte le funzioni di questa sezione sono nella cartella `BPP_HEUR_CODE/src`.
- Il comando `make test` le compila ed esegue ogni esempio di questo capitolo e ogni istanza degli esercizi, confrontando il numero di contenitori e l'assegnamento oggetto-contenitore con quelli attesi.

Riepilogo

Il codice:

- una funzione C++ per algoritmo, con indici a partire da 0
- FF, BF e WF differiscono solo nella scansione dei contenitori; l'apertura di un nuovo contenitore è la stessa

Le esecuzioni:

- su $w = (6, 5, 4, 7, 3)$, $c = 10$ danno le soluzioni delle sezioni precedenti, spostate di uno
- `make test` controlla ogni esempio e ogni esercizio

In sintesi

Introduzione: In sintesi

- Raccogliamo le idee chiave del capitolo sugli algoritmi di approssimazione per il BPP
- Il quadro riassuntivo raccoglie i bound sull'esempio del capitolo

Riepilogo (1/2)

- NF, FF, BF e WF collocano ogni oggetto appena arriva; NF mantiene un solo contenitore aperto, gli altri tre mantengono tutti i contenitori aperti finora e aprono un nuovo contenitore solo quando l'oggetto non entra da nessuna parte (classe Any Fit).
- In una soluzione Any Fit gli oggetti di due contenitori non entrano mai insieme, quindi tutti i contenitori tranne al più uno sono pieni per più della metà (proprietà di riempimento a metà), e $\mathscr{A}(I) \leq 2\,z(I)$.
- $r(NF) = r(WF) = 2$ e $r(FF) = r(BF) = \frac{17}{10}$: la sola regola di scelta separa il comportamento migliore e quello peggiore della classe Any Fit.

Riepilogo (2/2)

Quadro riassuntivo dell'istanza del BPP $c = 10$, $w = (6, 5, 4, 7, 3)$. I bound su $z(I)$ trovati finora:

Fonte	Lower bound	Upper bound	Da
Bound di conteggio	3	–	$\lceil \frac{25}{10} \rceil$
Questo capitolo, NF	–	3	$bin = (1, 2, 2, 3, 3)$
Questo capitolo, FF, BF, WF	–	3	3 contenitori ciascuno, con tre assegnamenti diversi
Valore ottimo	3	3	lower e upper bound coincidono