



SAPIENZA
UNIVERSITÀ DI ROMA

Network Optimization

Volume 1 — Graphs

Directed and undirected graphs and their language: computer representations, the node–arc incidence matrix, paths, cycles and connectivity, trees, breadth-first and depth-first search.

Fabio Furini

fabiofurini.github.io

Department of Computer, Control and Management Engineering — Sapienza University of Rome

Online version and downloadable material:
fabiofurini.github.io/network-optimization/
github.com/fabiofurini/network-optimization

Academic year 2026–2027

Contents

1	Graphs	2
1.1	Undirected and directed graphs	3
1.2	Representations of a graph	6
1.3	Connectivity	9
1.4	Trees and forests	11
1.5	The rank of the incidence matrix	14
1.6	Graph search	16
1.7	Typical mistakes	20
1.8	In summary	21



Graphs

Prerequisites

- Linear algebra: matrices, linear combinations, linearly independent vectors and the rank of a matrix.
- Sets, relations and equivalence classes; the $O(\cdot)$ notation for the running time of an algorithm.
- No knowledge of graph theory is needed: every notion is defined in this chapter.

Objectives

At the end of this chapter you will be able to:

- use the basic notions on undirected and directed graphs: adjacency, degree, walks, paths and cycles;
- store a graph as an adjacency matrix, as adjacency lists, as an arc list and as a node–arc incidence matrix, and compare the space they need;
- find the connected components and the strongly connected components of a graph;
- recognize a tree with the equivalent characterizations, and compute the rank of the incidence matrix;
- run the breadth-first search and the depth-first search, draw the predecessor tree and prove that the search finds the reachable nodes in $O(|N| + |A|)$ time.

1.1 Undirected and directed graphs

Definition 1.1: undirected graph

An **undirected graph** $G = (V, E)$ is given by a finite set V of **vertices** and a set E of **edges**, each edge being a 2-element subset of V :

$$E \subseteq \{ \{u, v\} : u, v \in V, u \neq v \}.$$

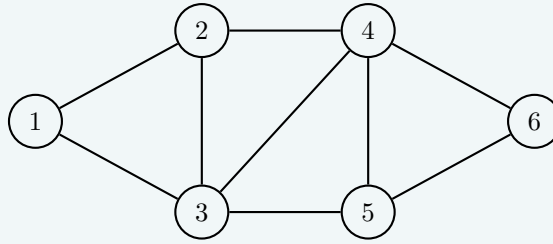
We write $n = |V|$ and $m = |E|$, and the vertices are labelled $1, 2, \dots, n$.

- The vertices u and v are the **endpoints** of the edge $\{u, v\}$; the edge is **incident** to u and to v , and u and v are **adjacent**.
- The **neighbourhood** of v is $N(v) = \{u \in V : \{u, v\} \in E\}$, the set of the incident edges is $\delta(v) = \{ \{u, v\} \in E : u \in V \}$, and the **degree** of v is $\deg(v) = |\delta(v)|$.

Example 1.1: an undirected graph

We consider the graph $G = (V, E)$ with $n = 6$ vertices and $m = 9$ edges:

$$V = \{1, 2, 3, 4, 5, 6\}, \quad E = \{ \{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{3, 5\}, \{4, 5\}, \{4, 6\}, \{5, 6\} \}.$$



The neighbourhood of vertex 3 is $N(3) = \{1, 2, 4, 5\}$, so $\deg(3) = 4$. The degrees of the six vertices are

$$\deg(1) = 2, \quad \deg(2) = 3, \quad \deg(3) = 4, \quad \deg(4) = 4, \quad \deg(5) = 3, \quad \deg(6) = 2.$$

Proposition 1.2: handshake lemma

For every undirected graph $G = (V, E)$ we have

$$\sum_{v \in V} \deg(v) = 2m.$$

Proof. The sum counts the pairs (v, e) with $v \in V$, $e \in \delta(v)$. Every edge $e = \{u, v\}$ appears in exactly two pairs, (u, e) and (v, e) , so the number of pairs is $2m$. $\square$

- In [Example 1.1](#) the degrees add up to $2 + 3 + 4 + 4 + 3 + 2 = 18 = 2 \cdot 9$.

Definition 1.3: directed graph

A **directed graph** $G = (N, A)$ is given by a finite set N of **nodes** and a set A of **arcs**, each arc being an ordered pair of distinct nodes:

$$A \subseteq \{(i, j) : i, j \in N, i \neq j\}.$$

We write $n = |N|$ and $m = |A|$, and the nodes are labelled $1, 2, \dots, n$.

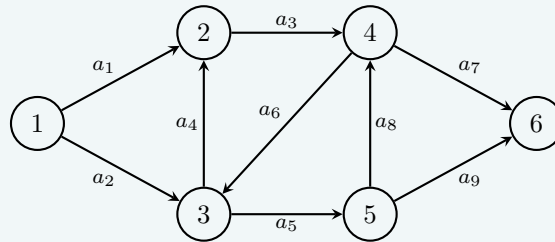
- The arc (i, j) goes from its **tail** i to its **head** j ; the node j is a **successor** of i , and i is a **predecessor** of j .
 - The **forward star** $\delta^+(i) = \{(i, j) \in A\}$ contains the arcs leaving i , and the **backward star** $\delta^-(i) = \{(j, i) \in A\}$ the arcs entering i .
 - The **out-degree** of i is $\deg^+(i) = |\delta^+(i)|$, and the **in-degree** is $\deg^-(i) = |\delta^-(i)|$.
- The arcs (i, j) and (j, i) are different: they are called **antiparallel** and can both belong to A .
 - The **underlying undirected graph** of $G = (N, A)$ has the edge $\{i, j\}$ for every arc (i, j) or (j, i) of A : it is obtained by ignoring the orientation of the arcs.

Example 1.2: the reference graph

We consider the directed graph $G = (N, A)$ with $n = 6$ nodes and $m = 9$ arcs:

$$N = \{1, 2, 3, 4, 5, 6\}, \quad A = \{(1, 2), (1, 3), (2, 4), (3, 2), (3, 5), (4, 3), (4, 6), (5, 4), (5, 6)\}.$$

The arcs are numbered $a_1, a_2, \dots, a_9$ in this (lexicographic) order, as in the figure.



The out-degrees and the in-degrees are:

node i	1	2	3	4	5	6
$\deg^+(i)$	2	1	2	2	2	0
$\deg^-(i)$	0	2	2	2	1	2

The underlying undirected graph is the graph of [Example 1.1](#). We call this graph the **reference graph**: all the examples of the chapter use it.

- Every arc has one tail and one head, so $\sum_{i \in N} \deg^+(i) = \sum_{i \in N} \deg^-(i) = m$. In the reference graph both sums are 9.

1.1.1 Walks, paths and cycles

Definition 1.4: walk, path and cycle

Let $G = (V, E)$ be an undirected graph.

- A **walk** from v_0 to v_k is a sequence $v_0, e_1, v_1, e_2, \dots, e_k, v_k$ of vertices and edges with $e_t = \{v_{t-1}, v_t\} \in E$ for all $t \in \{1, 2, \dots, k\}$; its **length** is k .
- The walk is **closed** if $v_0 = v_k$. It is a **path** if the vertices $v_0, v_1, \dots, v_k$ are distinct.
- A **cycle** is a closed walk with $k \geq 3$ whose vertices $v_0, v_1, \dots, v_{k-1}$ are distinct.

A graph without cycles is **acyclic**. In a simple graph a walk is identified by its vertices $v_0, v_1, \dots, v_k$.

Definition 1.5: directed walk, path and cycle

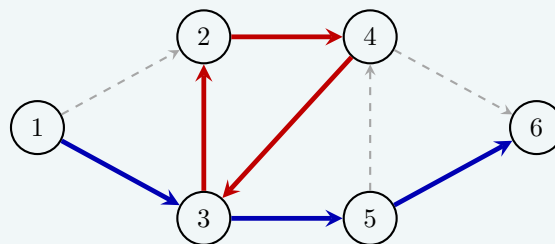
Let $G = (N, A)$ be a directed graph.

- A **directed walk** from i_0 to i_k is a sequence of nodes $i_0, i_1, \dots, i_k$ with $(i_{t-1}, i_t) \in A$ for all $t \in \{1, 2, \dots, k\}$: every arc is traversed from its tail to its head.
- It is a **directed path** if its nodes are distinct, and a **directed cycle** if $k \geq 2$, $i_0 = i_k$ and the nodes $i_0, i_1, \dots, i_{k-1}$ are distinct.
- A walk, path or cycle of the underlying undirected graph is a walk, path or cycle of G : its arcs can be traversed in both directions, and an arc is **forward** if it is traversed from tail to head, **backward** otherwise.

Example 1.3: walks, paths and cycles in the reference graph

In the reference graph:

- 1, 3, 2, 4, 3, 5 is a directed walk of length 5; it is not a path, because node 3 appears twice;
- 1, 3, 5, 6 is a directed path of length 3 (blue in the figure);
- 2, 4, 3, 2 is a directed cycle of length 3 (red in the figure);
- 1, 2, 3 is a path of G but not a directed path: the arc $(1, 2)$ is forward and the arc $(3, 2)$ is backward.



Lemma 1.6: a walk contains a path

If a graph has a walk (directed walk) from u to v , then it has a path (directed path) from u to v , whose vertices belong to the walk.

Proof. Among the walks from u to v that use only vertices of the given walk, let $v_0, v_1, \dots, v_k$ be one of minimum length. If two indices $s < t$ had $v_s = v_t$, deleting $v_{s+1}, \dots, v_t$ would give the walk $v_0, \dots, v_s, v_{t+1}, \dots, v_k$ from u to v , shorter by $t - s \geq 1$. Hence the vertices are distinct, and the walk is a path. The same argument holds for directed walks, since the deletion keeps every arc traversed from tail to head. $\square$

- In [Example 1.3](#), deleting the closed piece 3, 2, 4, 3 from the walk 1, 3, 2, 4, 3, 5 leaves the directed path 1, 3, 5.

1.1.2 Simple graphs and multigraphs

- A graph is **simple** if it has no **loops** (edges $\{v, v\}$ or arcs (i, i)) and no **parallel** edges or arcs (two copies of the same pair). The graphs of [Definitions 1.1](#) and [1.3](#) are simple, because E and A are sets of pairs of distinct elements.
- A **multigraph** allows loops and parallel edges or arcs. In this module all graphs are simple, and an arc is identified by the pair (i, j) of its tail and its head.

An undirected graph $G = (V, E)$ has edges $\{u, v\}$; a directed graph $G = (N, A)$ has arcs (i, j) from the tail i to the head j . A path has distinct vertices, and every walk contains a path with the same endpoints.

1.2 Representations of a graph

Definition 1.7: adjacency matrix

The **adjacency matrix** of a directed graph $G = (N, A)$ is the $n \times n$ matrix M with entries

$$m_{ij} = \begin{cases} 1 & \text{if } (i, j) \in A, \\ 0 & \text{otherwise,} \end{cases} \quad \forall i, j \in \{1, 2, \dots, n\}.$$

The adjacency matrix of an undirected graph has $m_{uv} = m_{vu} = 1$ for every edge $\{u, v\}$: it is symmetric.

Definition 1.8: node–arc incidence matrix

Let $G = (N, A)$ be a directed graph with arcs $a_1, a_2, \dots, a_m$. The **node–arc incidence matrix** is the $n \times m$ matrix B with one row per node and one column per arc, whose

entries are

$$b_{ik} = \begin{cases} +1 & \text{if } i \text{ is the tail of } a_k, \\ -1 & \text{if } i \text{ is the head of } a_k, \\ 0 & \text{otherwise,} \end{cases} \quad \forall i \in \{1, 2, \dots, n\}, \forall k \in \{1, 2, \dots, m\}.$$

- Every column of $\mathbf{B}$ has exactly one $+1$ (the tail) and one -1 (the head), so the sum of the rows of $\mathbf{B}$ is the zero vector.
- Row i has a $+1$ for every arc of $\delta^+(i)$ and a -1 for every arc of $\delta^-(i)$: it has $\deg^+(i) + \deg^-(i)$ nonzero entries.
- The **vertex–edge incidence matrix** of an undirected graph has a 1 in both endpoints of every edge.

Example 1.4: the matrices of the reference graph

The adjacency matrix $\mathbf{M}$ and the node–arc incidence matrix $\mathbf{B}$ of the reference graph of Example 1.2 are shown with the index of each row (a node) and of each column (a node in $\mathbf{M}$, an arc in $\mathbf{B}$):

			1	2	3	4	5	6
$\mathbf{M} :$	1	0	1	1	0	0	0	0
	2	0	0	0	1	0	0	0
	3	0	1	0	0	1	0	0
	4	0	0	1	0	0	1	0
	5	0	0	0	1	0	1	0
	6	0	0	0	0	0	0	0

		a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
$\mathbf{B} :$	1	1	1	0	0	0	0	0	0	0
	2	-1	0	1	-1	0	0	0	0	0
	3	0	-1	0	1	1	-1	0	0	0
	4	0	0	-1	0	0	1	1	-1	0
	5	0	0	0	0	-1	0	0	1	1
	6	0	0	0	0	0	0	-1	0	-1

For example, the column of $a_6 = (4, 3)$ has $+1$ in row 4 (the tail) and -1 in row 3 (the head). Row 6 of $\mathbf{M}$ is zero, because node 6 has no successors.

Definition 1.9: adjacency lists and arc list

Let $G = (N, A)$ be a directed graph.

- The **adjacency lists** store, for every node i , the list $L(i)$ of its successors, that is, the heads of the arcs of $\delta^+(i)$. In this module every list is sorted by increasing index.
- The **arc list** stores two arrays of length m , with the tail and the head of every arc.

For an undirected graph the list $L(v)$ contains the neighbourhood $N(v)$, and every edge

appears in two lists.

Example 1.5: the lists of the reference graph

The adjacency lists and the arc list of the reference graph are:

i	$L(i)$	arc	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
1	2, 3	tail	1	1	2	3	3	4	4	5	5
2	4	head	2	3	4	2	5	3	6	4	6
3	2, 5										
4	3, 6										
5	4, 6										
6	(empty)										

The lists of the underlying undirected graph are $L(1) = \{2, 3\}$, $L(2) = \{1, 3, 4\}$, $L(3) = \{1, 2, 4, 5\}$, $L(4) = \{2, 3, 5, 6\}$, $L(5) = \{3, 4, 6\}$ and $L(6) = \{4, 5\}$.

1.2.1 Space and access time

- The table compares the four representations of a directed graph. The space is the number of stored entries; the two operations are the test of an arc and the scan of the successors of a node.

representation	space	is $(i, j) \in A$?	successors of i
adjacency matrix	n^2	$O(1)$	$O(n)$
node–arc incidence matrix	nm	$O(m)$	$O(m)$
arc list	$2m$	$O(m)$	$O(m)$
adjacency lists	$n + m$	$O(\deg^+(i))$	$O(\deg^+(i))$

- For the reference graph the four representations store 36, 54, 18 and 15 entries.
- A simple directed graph has $m \leq n(n-1)$. In the networks of the applications m is much smaller than n^2 (the graph is **sparse**), and the adjacency lists are the representation of choice for the algorithms.

1.2.2 The incidence matrix and flow conservation

- Given a value x_{ij} for every arc, row i of $B\mathbf{x}$ is what leaves node i minus what enters it:

$$(B\mathbf{x})_i = \sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji}, \quad \forall i \in N.$$

- With $x_{ij} = 1$ for every arc of the reference graph, $B\mathbf{x}$ is the vector of the differences $\deg^+(i) - \deg^-(i)$:

$$B\mathbf{x} = (2, -1, 0, 0, 1, -2)'.$$

The node-arc incidence matrix has +1 in the tail and -1 in the head of every arc; the adjacency lists store a graph in $n + m$ space and give the successors of a node in $O(\deg^+(i))$ time.

1.3 Connectivity

Definition 1.10: connected graph and connected components

Let $G = (V, E)$ be an undirected graph.

- Two vertices u and v are **connected** if G has a path from u to v ; the graph G is **connected** if every two vertices are connected.
- A **connected component** of G is a maximal subset $U \subseteq V$ of pairwise connected vertices.

A directed graph is **weakly connected** if its underlying undirected graph is connected, and its connected components are those of the underlying undirected graph.

Definition 1.11: reachability and strongly connected components

Let $G = (N, A)$ be a directed graph.

- A node j is **reachable** from i if G has a directed path from i to j ; the set of the nodes reachable from i , i included, is denoted by $R(i)$.
- The nodes i and j are **strongly connected** if $j \in R(i)$ and $i \in R(j)$; the graph is **strongly connected** if every two nodes are strongly connected.
- A **strongly connected component** of G is a maximal subset $S \subseteq N$ of pairwise strongly connected nodes.

Proposition 1.12: components are equivalence classes

Being connected is an equivalence relation on V , and being strongly connected is an equivalence relation on N . Hence the connected components partition V and the strongly connected components partition N .

Proof. We prove the three properties for the strong connection; the undirected case is the same with paths in place of directed paths.

1. *Reflexive*: the path i of length 0 shows that $i \in R(i)$.
2. *Symmetric*: the definition is symmetric in i and j .
3. *Transitive*: a directed path from i to j followed by one from j to h is a directed walk from i to h , which contains a directed path by [Lemma 1.6](#). The same holds from h to i .

The equivalence classes are the maximal subsets of pairwise related elements, that is, the components. $\square$

- The **reversed graph** $G' = (N, A')$ has the arc (j, i) for every arc $(i, j) \in A$. The node i is reachable from j in G if and only if j is reachable from i in G' ; we denote by $R'(i)$ the set of the nodes reachable from i in G' .
- By [Definition 1.11](#), the strongly connected component that contains i is

$$S(i) = R(i) \cap R'(i).$$

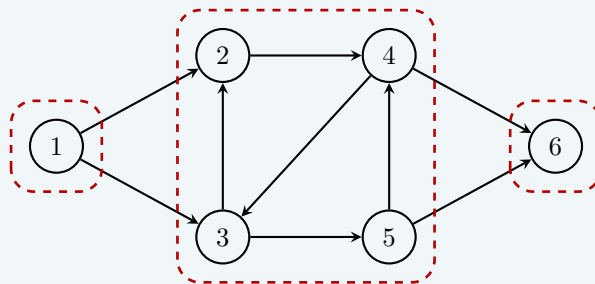
Example 1.6: components of the reference graph

In the reference graph:

$$R(1) = \{1, 2, 3, 4, 5, 6\}, \quad R(2) = \{2, 3, 4, 5, 6\}, \quad R(6) = \{6\}.$$

In the reversed graph $R'(3) = \{1, 2, 3, 4, 5\}$, and $R(3) = \{2, 3, 4, 5, 6\}$, so $S(3) = \{2, 3, 4, 5\}$. The strongly connected components are

$$\{1\}, \quad \{2, 3, 4, 5\}, \quad \{6\}.$$



The graph is weakly connected, with one connected component, but it is not strongly connected: no node is reachable from node 6.

The connected components and the strongly connected components are the classes of two equivalence relations. The strongly connected component of i is $R(i) \cap R'(i)$: the nodes reachable from i that can also reach i .

1.4 Trees and forests

Definition 1.13: tree, forest and spanning tree

Let $G = (V, E)$ be an undirected graph.

- G is a **forest** if it is acyclic, and a **tree** if it is connected and acyclic. A vertex of degree 1 is a **leaf**.
- A **spanning tree** of a connected graph G is a tree $T = (V, F)$ with $F \subseteq E$; a **spanning forest** of G is a forest (V, F) , $F \subseteq E$, with the same connected components as G .

For a directed graph $G = (N, A)$ the notions apply to the underlying undirected graph: a subset of arcs $F \subseteq A$ is a tree (a forest) if the corresponding edges form a tree (a forest).

Lemma 1.14: leaves

Every tree with $n \geq 2$ vertices has at least two leaves.

Proof. The tree is connected with $n \geq 2$ vertices, so it has an edge. Let $v_0, v_1, \dots, v_k$ be a path of maximum length, with $k \geq 1$. A neighbour $u \neq v_1$ of v_0 would either lie on the path, closing a cycle, or extend the path, against its maximality. Hence $\deg(v_0) = 1$, and in the same way $\deg(v_k) = 1$. $\square$

Lemma 1.15: adding an edge

Let $G = (V, E)$ have k connected components, and let $e = \{u, v\} \notin E$ with $u, v \in V$.

1. If u and v are in different components of G , the graph $(V, E \cup \{e\})$ has $k - 1$ components and no cycle through e .
2. If u and v are in the same component of G , the graph $(V, E \cup \{e\})$ has k components and a cycle through e .

Proof. 1. The edge e joins the components of u and v into one, and leaves the other components unchanged. A cycle through e would give, without e , a path from u to v in G : impossible.

2. The components do not change. A path $u = v_0, v_1, \dots, v_k = v$ of G has $k \geq 2$, since $\{u, v\} \notin E$; with e it closes the cycle $v_0, v_1, \dots, v_k, v_0$. $\square$

Corollary 1.16: edges and components

A graph with n vertices, m edges and k connected components satisfies $k \geq n - m$, with equality if and only if the graph is acyclic.

Proof. Start from $(V, \emptyset)$, which has n components, and insert the m edges one at a time. By [Lemma 1.15](#), each insertion either decreases the number of components by one, or leaves it unchanged and closes a cycle. Hence $k \geq n - m$.

- If the graph has a cycle C , the last edge of C to be inserted joins two vertices already connected by the rest of C : it leaves the number unchanged, and $k > n - m$.
- If the graph is acyclic, no insertion closes a cycle, because a cycle of an intermediate graph is a cycle of the final graph: $k = n - m$. $\square$

Theorem 1.17: characterizations of a tree

Let $G = (V, E)$ be an undirected graph with n vertices and m edges. The following statements are equivalent:

- G is a tree (connected and acyclic);
- G is connected and $m = n - 1$;
- G is acyclic and $m = n - 1$;
- for every two vertices u and v there is exactly one path from u to v ;
- G is connected, and removing any edge disconnects it.

Proof. Let k be the number of connected components; G is connected if and only if $k = 1$.

- **(a) $\Rightarrow$ (b),(c).** G is acyclic with $k = 1$, so $m = n - k = n - 1$ by [Corollary 1.16](#).
- **(b) $\Rightarrow$ (a).** Here $k = 1 = n - m$: the equality holds, and G is acyclic by the same corollary.
- **(c) $\Rightarrow$ (a).** G is acyclic, so $k = n - m = 1$ by the same corollary: G is connected.

For $(a) \Leftrightarrow (d)$, connected means at least one path between every two vertices; we show that acyclic means at most one.

- If G has a cycle $v_0, v_1, \dots, v_{k-1}, v_0$, the vertices v_0 and v_1 are joined by two paths: the edge $\{v_0, v_1\}$, and $v_0, v_{k-1}, \dots, v_2, v_1$.
- Let $P \neq Q$ be two paths from u to v . Let w be the last vertex of their common initial part, and x the first vertex after w on P that also lies on Q (v is a candidate). The pieces of P and of Q from w to x share only w and x , and they do not both consist of the single edge $\{w, x\}$, since P and Q leave w along different edges. Together they form a cycle.

For $(a) \Leftrightarrow (e)$, let G be connected and $e = \{u, v\} \in E$. Applying [Lemma 1.15](#) to $(V, E \setminus \{e\})$ and e : the edge e lies on a cycle of G if and only if u and v are connected in $(V, E \setminus \{e\})$, that is, if and only if removing e leaves the graph connected. Hence every removal disconnects G if and only if no edge lies on a cycle. $\square$

Corollary 1.18: spanning forests

Every graph with n vertices and k connected components has a spanning forest, with $n - k$ edges. In particular, every connected graph has a spanning tree, with $n - 1$ edges.

Proof. While the graph has a cycle, remove an edge $e = \{u, v\}$ of a cycle. The rest of the cycle still connects u and v , so adding e back does not change the components (Lemma 1.15, part 2): removing it does not change them either. The final graph is acyclic with the same k components, so it is a spanning forest, and it has $n - k$ edges by Corollary 1.16. $\square$

1.4.1 Fundamental cycles
Proposition 1.19: fundamental cycle

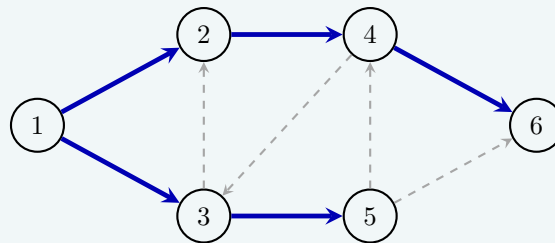
Let $T = (V, F)$ be a spanning tree of a connected graph $G = (V, E)$, and let $e = \{u, v\} \in E \setminus F$. The graph $(V, F \cup \{e\})$ contains exactly one cycle, the **fundamental cycle** $C(T, e)$, formed by e and by the unique path from u to v in T .

Proof. The vertices u and v are connected in T , so the edge e and the path from u to v in T form a cycle (Lemma 1.15, part 2). Every cycle of $(V, F \cup \{e\})$ contains e , since T is acyclic; removing e from it leaves a path from u to v in T , which is unique by Theorem 1.17(d). $\square$

- A connected graph with a spanning tree T has $m - n + 1$ fundamental cycles, one for every edge outside T .
- In a directed graph the fundamental cycle of an arc $(i, j) \notin F$ is traversed in the direction of (i, j) : the arcs of T traversed from tail to head are forward, the others backward.

Example 1.7: a spanning tree of the reference graph

The arcs $F = \{(1, 2), (1, 3), (2, 4), (3, 5), (4, 6)\}$ form a spanning tree T of the reference graph (blue in the figure): they are $5 = n - 1$ arcs, and their edges form a connected graph.



The $m - n + 1 = 4$ arcs outside T (dashed) give the fundamental cycles, written from the tail of the arc:

arc	fundamental cycle	backward arcs
(3, 2)	3, 2, 1, 3	(1, 2)
(4, 3)	4, 3, 1, 2, 4	(1, 3)
(5, 4)	5, 4, 2, 1, 3, 5	(2, 4), (1, 2)
(5, 6)	5, 6, 4, 2, 1, 3, 5	(4, 6), (2, 4), (1, 2)

For example, the cycle of (3, 2) uses (3, 2) forward, (1, 2) backward (from 2 to 1) and (1, 3) forward.

A tree is connected and acyclic; any two of the properties “connected”, “acyclic”, “ $n - 1$ edges” imply the third. Adding to a spanning tree an edge outside it creates exactly one cycle, the fundamental cycle.

1.5 The rank of the incidence matrix

- Let $G = (N, A)$ be a directed graph with node–arc incidence matrix $\mathbf{B}$. We denote by $\mathbf{B}_a$ the column of the arc a , and by $\mathbf{e}_i$ the i -th unit vector of $\mathbb{R}^n$, so that $\mathbf{B}_{(i,j)} = \mathbf{e}_i - \mathbf{e}_j$.

Lemma 1.20: columns of a forest

If the arcs $F \subseteq A$ form a forest, the columns $\mathbf{B}_a$, $a \in F$, are linearly independent.

Proof. By induction on $|F|$; for $F = \emptyset$ there is nothing to prove. Let $|F| \geq 1$ and $\sum_{a \in F} \lambda_a \mathbf{B}_a = \mathbf{0}$.

1. A component of the forest with at least one arc is a tree with at least 2 nodes, so it has a leaf ℓ (Lemma 1.14); let $a' \in F$ be the only arc of F incident to ℓ .
2. Row ℓ of the combination contains only the term of a' , which is $\pm \lambda_{a'}$; hence $\lambda_{a'} = 0$.
3. The arcs $F \setminus \{a'\}$ still form a forest, and $\sum_{a \in F \setminus \{a'\}} \lambda_a \mathbf{B}_a = \mathbf{0}$: by induction all the λ_a are zero. $\square$

Lemma 1.21: columns of a cycle

If the arcs $C \subseteq A$ form a cycle, the columns $\mathbf{B}_a$, $a \in C$, are linearly dependent.

Proof. Let the cycle visit $v_0, v_1, \dots, v_k = v_0$ along the arcs $c_1, c_2, \dots, c_k$, and let $\varepsilon_t = +1$ if c_t is forward and $\varepsilon_t = -1$ if it is backward. In both cases $\varepsilon_t \mathbf{B}_{c_t} = \mathbf{e}_{v_{t-1}} - \mathbf{e}_{v_t}$, and the sum telescopes:

$$\sum_{t=1}^k \varepsilon_t \mathbf{B}_{c_t} = \mathbf{e}_{v_0} - \mathbf{e}_{v_k} = \mathbf{0}.$$

The coefficients ε_t are not zero, so the columns are linearly dependent. $\square$

Theorem 1.22: rank of the incidence matrix

Let $G = (N, A)$ be a directed graph whose underlying undirected graph has k connected components. Then

$$\text{rank}(\mathbf{B}) = |N| - k.$$

Proof. Let $N_1, N_2, \dots, N_k$ be the node sets of the components, and $n = |N|$.

- $\text{rank}(\mathbf{B}) \geq n - k$. By [Corollary 1.18](#) the graph has a spanning forest with $n - k$ arcs, whose columns are linearly independent by [Lemma 1.20](#).
- $\text{rank}(\mathbf{B}) \leq n - k$. Every arc has both endpoints in the same component, so for each $h \in \{1, 2, \dots, k\}$ the rows of the nodes of N_h add up to the zero vector. The k vectors $\mathbf{y}^h$ with $y_i^h = 1$ for $i \in N_h$ and 0 otherwise satisfy $(\mathbf{y}^h)' \mathbf{B} = \mathbf{0}'$.
- These k vectors have disjoint supports, so they are linearly independent: the left null space of $\mathbf{B}$ has dimension at least k , and $\text{rank}(\mathbf{B}) \leq n - k$. $\square$

Corollary 1.23: bases of the incidence matrix

Let the underlying undirected graph of $G = (N, A)$ be connected. Then $\text{rank}(\mathbf{B}) = |N| - 1$, the matrix obtained by deleting any one row of $\mathbf{B}$ has full row rank, and a set of $|N| - 1$ columns of $\mathbf{B}$ is linearly independent if and only if its arcs form a spanning tree.

Proof. The rank is $n - 1$ by [Theorem 1.22](#) with $k = 1$.

- The rows add up to zero, so any row is minus the sum of the others: the remaining $n - 1$ rows span the row space, which has dimension $n - 1$, and they are linearly independent.
- If $n - 1$ arcs form a spanning tree, their columns are independent by [Lemma 1.20](#). If their columns are independent, the arcs contain no cycle by [Lemma 1.21](#): they are acyclic with $n - 1$ arcs, hence a spanning tree by [Theorem 1.17\(c\)](#). $\square$

Example 1.8: the rank on the reference graph

- The reference graph is weakly connected, so $\text{rank}(\mathbf{B}) = 6 - 1 = 5$; Gaussian elimination on the matrix of [Example 1.4](#) confirms it.
- The five columns of the spanning tree of [Example 1.7](#), a_1, a_2, a_3, a_5, a_7 , are linearly independent.
- The directed cycle $2, 4, 3, 2$ uses $a_3 = (2, 4)$, $a_6 = (4, 3)$, $a_4 = (3, 2)$, all forward, and $\mathbf{B}_{a_3} + \mathbf{B}_{a_6} + \mathbf{B}_{a_4} = \mathbf{0}$.
- Without the arcs $(4, 6)$ and $(5, 6)$ the graph has the two components $\{1, 2, 3, 4, 5\}$ and $\{6\}$, and the rank of its incidence matrix is $6 - 2 = 4$.

The rank of the node–arc incidence matrix is the number of nodes minus the number of connected components. In a connected graph one row is redundant, and the bases of $\mathbf{B}$ are exactly the spanning trees.

1.6 Graph search

Definition 1.24: graph search

Given a directed graph $G = (N, A)$ and a node $s \in N$, the **source**, a **graph search** finds the set $R(s)$ of the nodes reachable from s . During the search:

- a node is **labelled** when the search reaches it, and it stays labelled;
- every labelled node $j \neq s$ records its **predecessor** $pred(j)$, the node from which it was reached;
- the labelled nodes still to be processed are kept in a list, $LIST$;
- an arc (i, j) is **admissible** if i is labelled and j is not.

Algorithm

Algorithm: Generic graph search

Input: a directed graph $G = (N, A)$ given by its adjacency lists, a source $s \in N$

Output: the labelled nodes, equal to $R(s)$, and the predecessors $pred(j)$

- 1 **Step 0 (initialization).** Label s and set $LIST := \{s\}$; all the other nodes are unlabelled.
- 2 **Step 1 (selection).** If $LIST = \emptyset$, STOP. Otherwise select a node $i \in LIST$:
 - 3 the *first* node of $LIST$ in the breadth-first search (BFS),
 - 4 the *last* node of $LIST$ in the depth-first search (DFS).
- 5 **Step 2 (labelling).** If i has an admissible arc (i, j) , take the first one in $L(i)$:
 - 6 label j , set $pred(j) := i$ and append j at the end of $LIST$.
 - 7 Otherwise remove i from $LIST$. Go to Step 1.

- In the BFS the list $LIST$ is a **queue** (first in, first out): the nodes are processed in the order in which they are labelled.
- In the DFS the list $LIST$ is a **stack** (last in, first out): the search always moves on from the most recently labelled node still in $LIST$, and goes back only when that node has no admissible arc.
- The order of the adjacency lists decides which admissible arc is taken first; with lists sorted by increasing index, as in this module, the result is unique.

1.6.1 The predecessor tree

Proposition 1.25: predecessor tree

At every iteration, the arcs $(pred(j), j)$, for the labelled nodes $j \neq s$, form a tree on the labelled nodes, the **predecessor tree**. For every labelled node j it contains a directed path from s to j .

Proof. Let r be the number of labelled nodes.

1. Every labelled node $j \neq s$ has one predecessor, labelled before j . Starting from j and moving to the predecessor, the nodes are labelled earlier and earlier, so the sequence ends in s .
2. Read backwards, it is a directed path from s to j along the arcs $(pred(h), h)$.
3. Hence the $r - 1$ arcs connect the r labelled nodes, and they form a tree by [Theorem 1.17\(b\)](#). $\square$

Theorem 1.26: correctness of the search

At the end of the generic search, BFS and DFS included, the labelled nodes are exactly the nodes of $R(s)$.

Proof. Every labelled node is reachable from s , by [Proposition 1.25](#). Conversely, let $j \in R(s)$ and let $s = i_0, i_1, \dots, i_k = j$ be a directed path; assume by contradiction that j is unlabelled.

1. Let i_t be the first unlabelled node of the path; $t \geq 1$, since s is labelled.
2. The node i_{t-1} is labelled, so it entered *LIST*, and at the end it has left it.
3. A node leaves *LIST* only when it has no admissible arc. But (i_{t-1}, i_t) is admissible until the end, because labels are never removed: a contradiction. $\square$

Theorem 1.27: running time of the search

With adjacency lists, the generic search runs in $O(|N| + |A|)$ time.

Proof. Every node keeps a pointer to the current position in its list $L(i)$; the pointer moves forward past the labelled successors and never moves back, since labels are never removed.

- Each iteration labels a node or removes a node from *LIST*. Each node is labelled at most once and removed at most once, so there are at most $2n$ iterations, each with $O(1)$ work besides the pointer.
- The pointer of node i moves at most $\deg^+(i)$ times in total, and $\sum_{i \in N} \deg^+(i) = m$.

The total time is $O(n + m) = O(|N| + |A|)$. $\square$

- With the adjacency matrix the successors of a node are found by scanning a row of length n , and the search takes $O(n^2)$ time.

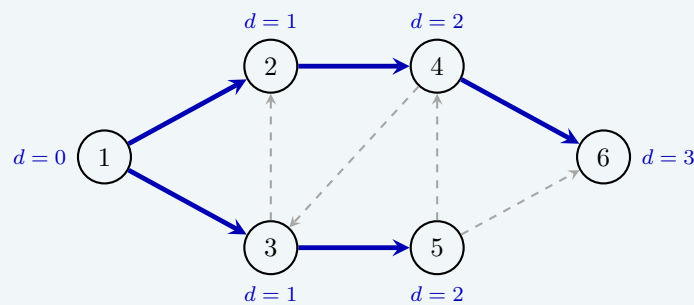
1.6.2 The breadth-first search on the reference graph

Example 1.9: BFS from node 1

We run the BFS on the reference graph from $s = 1$, with the adjacency lists of [Example 1.5](#). Each row is one iteration, and *LIST* is shown after it:

iteration	selected i	action	<i>LIST</i>
1	1	label 2, $pred(2) = 1$	1, 2
2	1	label 3, $pred(3) = 1$	1, 2, 3
3	1	remove 1	2, 3
4	2	label 4, $pred(4) = 2$	2, 3, 4
5	2	remove 2	3, 4
6	3	label 5, $pred(5) = 3$ (2 is labelled)	3, 4, 5
7	3	remove 3	4, 5
8	4	label 6, $pred(6) = 4$ (3 is labelled)	4, 5, 6
9	4	remove 4	5, 6
10	5	remove 5 (4 and 6 are labelled)	6
11	6	remove 6	(empty)

The labelling order is 1, 2, 3, 4, 5, 6: every node is reachable from 1. The predecessor tree has the arcs (1, 2), (1, 3), (2, 4), (3, 5), (4, 6); it is the spanning tree of [Example 1.7](#). Next to each node, the figure shows the number $d(j)$ of arcs of the tree path from 1 to j .



Proposition 1.28: BFS finds shortest paths in number of arcs

In the predecessor tree of the BFS, let $d(s) = 0$ and $d(j) = d(pred(j)) + 1$ for every labelled $j \neq s$. Then, for every $j \in R(s)$, $d(j)$ is the minimum number of arcs of a directed path from s to j .

Proof. We prove three facts.

1. If $LIST = u_1, u_2, \dots, u_r$, then $d(u_1) \leq d(u_2) \leq \dots \leq d(u_r) \leq d(u_1) + 1$. It holds at Step 0. A new node j is labelled from u_1 and appended with $d(j) = d(u_1) + 1$; removing u_1 keeps the property, because $d(u_r) \leq d(u_1) + 1 \leq d(u_2) + 1$.
2. The nodes are labelled in non-decreasing order of d . Each new node has $d(j) = d(u_1) + 1 \geq d(u_r)$, and u_r is the last node labelled before j .
3. For every arc (i, j) with i labelled, $d(j) \leq d(i) + 1$. If j is labelled after i , then j is labelled before i leaves $LIST$, by a node u at the front of $LIST$, labelled no later than i : $d(j) = d(u) + 1 \leq d(i) + 1$. If j is labelled before i , then $d(j) \leq d(i)$ by fact 2.

Let $s = i_0, i_1, \dots, i_k = j$ be a directed path with the minimum number k of arcs. Its nodes are labelled (Theorem 1.26), and by fact 3, $d(j) \leq d(i_{k-1}) + 1 \leq \dots \leq d(s) + k = k$. The tree path from s to j is a directed path with $d(j)$ arcs, so $d(j) \geq k$. $\square$

- In Example 1.9, the values $d = (0, 1, 1, 2, 2, 3)$ are the minimum numbers of arcs from node 1: for example, the shortest directed paths to node 6 have 3 arcs, such as 1, 2, 4, 6.

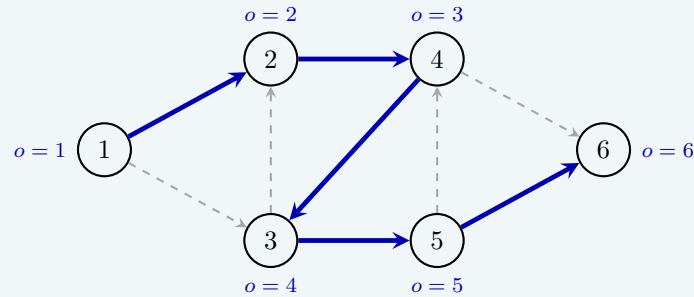
1.6.3 The depth-first search on the reference graph

Example 1.10: DFS from node 1

We run the DFS on the reference graph from $s = 1$: the selected node is always the last one of $LIST$.

iteration	selected i	action	$LIST$
1	1	label 2, $pred(2) = 1$	1, 2
2	2	label 4, $pred(4) = 2$	1, 2, 4
3	4	label 3, $pred(3) = 4$	1, 2, 4, 3
4	3	label 5, $pred(5) = 3$ (2 is labelled)	1, 2, 4, 3, 5
5	5	label 6, $pred(6) = 5$ (4 is labelled)	1, 2, 4, 3, 5, 6
6	6	remove 6	1, 2, 4, 3, 5
7	5	remove 5	1, 2, 4, 3
8	3	remove 3	1, 2, 4
9	4	remove 4 (3 and 6 are labelled)	1, 2
10	2	remove 2	1
11	1	remove 1 (2 and 3 are labelled)	(empty)

The labelling order is 1, 2, 4, 3, 5, 6, and the predecessor tree has the arcs $(1, 2)$, $(2, 4)$, $(4, 3)$, $(3, 5)$, $(5, 6)$: a single directed path. Next to each node, the figure shows its position $o(j)$ in the labelling order.



- Both searches label all the nodes, as [Theorem 1.26](#) guarantees, but their trees differ. The DFS tree reaches node 6 with 5 arcs, while the BFS tree uses 3.

1.6.4 Using the search

- Connected components.** In an undirected graph, a search from an unlabelled vertex labels its whole component; repeating it from every vertex still unlabelled finds all the components in $O(n + m)$ time.
- Strongly connected components.** A search in G and one in the reversed graph G' give $R(i)$ and $R'(i)$, hence $S(i) = R(i) \cap R'(i)$; repeating it for a node of every component still unknown takes $O(n(n + m))$ time.
- Spanning trees.** On a connected undirected graph, the predecessor tree of a search from any vertex is a spanning tree.

A graph search labels exactly the nodes reachable from the source, in $O(|N| + |A|)$ time, and its predecessors form a tree. The BFS (a queue) gives paths with the minimum number of arcs; the DFS (a stack) goes deep first.

1.7 Typical mistakes

Warning

- Swapping the signs of the incidence matrix.** The entry is $+1$ in the tail and -1 in the head: the column of $(4, 3)$ has $+1$ in row 4.
- Calling a walk a path.** A path has distinct vertices; a walk may repeat them, and it always contains a path with the same endpoints.
- Confusing weak and strong connectivity.** The reference graph is weakly connected, but it has three strongly connected components.
- Counting the edges only.** A graph with $n - 1$ edges need not be a tree: it must also be connected, or acyclic.
- Believing that the incidence matrix has full row rank.** Its rank is $|N| - k$: in a connected graph one row is always redundant.

- **Expecting shortest paths from the DFS.** Only the BFS tree gives the minimum number of arcs, and only when every arc counts one.
- **Forgetting the order of the lists.** The labelling order and the predecessor tree depend on the order in which the successors are scanned.

1.8 In summary

Summary

- An undirected graph $G = (V, E)$ has edges $\{u, v\}$, a directed graph $G = (N, A)$ has arcs (i, j) ; the degrees add up to $2m$, the out-degrees and the in-degrees to m .
- The adjacency lists need $n + m$ space; the node–arc incidence matrix has $+1$ in the tail and -1 in the head, and row i of $\mathbf{B}\mathbf{x}$ is the outflow minus the inflow of node i .
- The connected and the strongly connected components are equivalence classes; $S(i) = R(i) \cap R'(i)$.
- A tree is connected and acyclic, with $n - 1$ edges and a unique path between every two vertices; an edge outside a spanning tree closes one fundamental cycle.
- $\text{rank}(\mathbf{B}) = |N| - k$, and in a connected graph the bases of $\mathbf{B}$ are the spanning trees.
- The generic search finds $R(s)$ in $O(|N| + |A|)$ time; the BFS gives the minimum number of arcs.

The reference graph at a glance.

nodes and arcs	$n = 6, m = 9$
connected components	1
strongly connected components	$\{1\}, \{2, 3, 4, 5\}, \{6\}$
$\text{rank}(\mathbf{B})$	5
BFS from 1	order 1, 2, 3, 4, 5, 6; $d = (0, 1, 1, 2, 2, 3)$
DFS from 1	order 1, 2, 4, 3, 5, 6