Foundations of Operations Research

Marta Pascoal (marta.brazpascoal@polimi.it)

Dipartimento di Elettronica, Informazione e Bioingegneria – Politecnico di Milano

2024/25

2. Graph and network optimization

Outline

- Graphs
- Optimal cost spanning trees
- Optimal paths
- Network flows

2. Graph and network optimization

Many decision-making problems can be formulated in terms of graphs and networks.

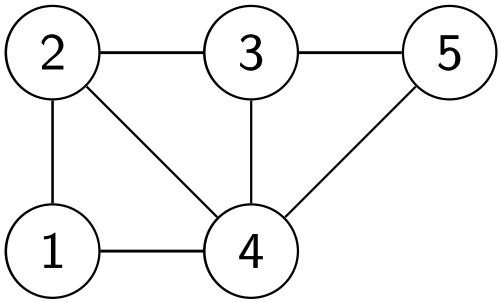
Examples

- transportation and distribution problems,
- network design (communication, electrical, ...),
- location problems (services and facilities),
- project planning, resource management,
- timetable scheduling,
- production planning, ...

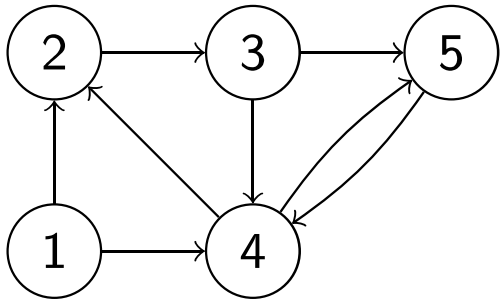
2.1. Graphs

- A graph is a pair $G = (N, E)$, with N a set of nodes or vertices and $E \subseteq N \times N$ a set of edges or arcs connecting them pairwise.
- An edge connecting the nodes i and j is represented by $\{i, j\}$ (or (i, j)) if the graph is undirected (or directed), respectively.

Example A road network which connects n cities can be modelled, by a graph where a city corresponds to a node, and a connection corresponds to an edge.



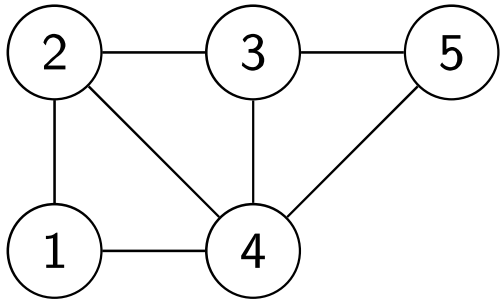
$N = \{1, 2, 3, 4, 5\}$
$E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$
Undirected graph



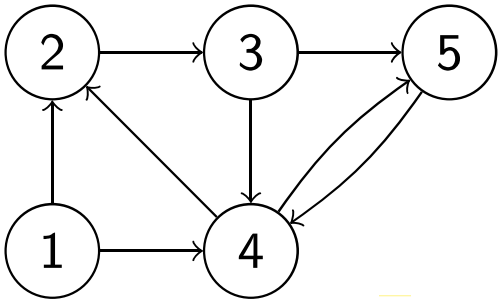
$N = \{1, 2, 3, 4, 5\}$
$E' = \{(1, 2), (1, 4), (2, 3), (2, 4), (3, 4), (3, 5), (4, 5)\}$
Directed graph

- Two nodes are adjacent if they are connected by an edge.
- An edge e is incident in a node v if v is an endpoint of e.
- Undirected graphs: The degree of a node is the number of incident edges
  Directed graphs: The in-degree (out-degree) of a node is the number of arcs that have it as successor (predecessor).

Example



Nodes 1 and 2 are adjacent (unlike nodes 1 and 3)
Edge $\{1, 2\}$ is incident in nodes 1 and 2
Node 1 has degree 2, node 4 has degree 4



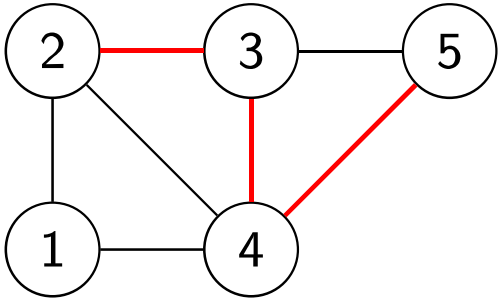
Node 1 has in-degree 0, and out-degree 2

- A (directed) path from $i \in N$ to $j \in N$ is a sequence of (arcs) edges

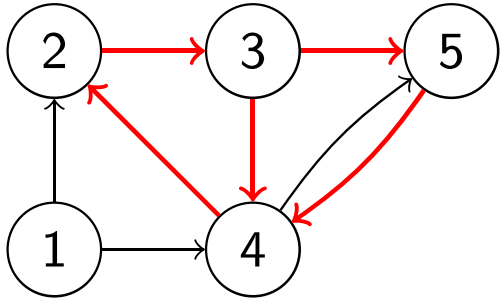
$$p = \langle \{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\} \rangle,$$

  connecting nodes v_1 and v_k, with $((v_l, v_{l+1}) \in E)$ $\{v_l, v_{l+1}\} \in E$, for $l = 0, \ldots, k-1$.
- Nodes u and v are connected if there is a path connecting them.
- A graph (N, E) is connected if u, v are connected, for any $u, v \in N$.
- A graph (N, E) is strongly connected if u, v are connected by a directed path, for any $u, v \in N$.

Example



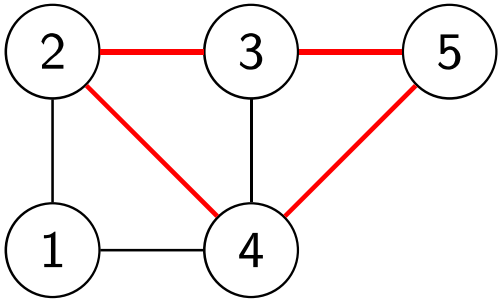
$\langle \{2,3\}, \{3,4\}, \{4,5\} \rangle$ is a path from node 2 to node 5
Nodes 2 and 5 are connected
Connected graph



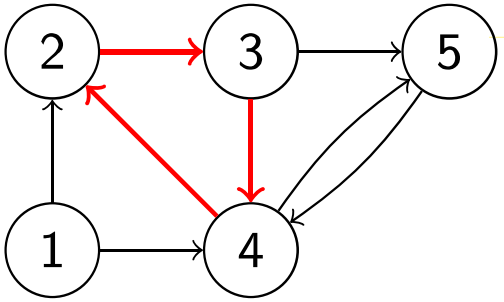
$\langle (3,5), (5,4), (4,2), (2,3), (3,4) \rangle$ is a directed path from node 3 to node 4

Not strongly connected graph

- A cycle (circuit) is a (directed) path with $v_1 = v_k$.

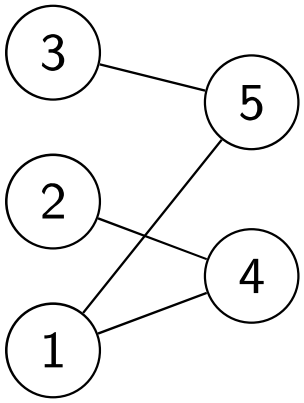


$\langle\{2,3\},\{3,5\},\{5,4\},\{4,2\}\rangle$ is a cycle

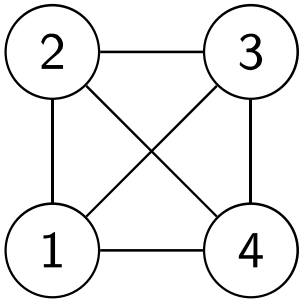


$\langle(2,3),(3,4),(4,2)\rangle$ is a circuit

- A graph is bipartite if there is a partition $N = N_1 \cup N_2$ $(N_1 \cap N_2 = \emptyset)$ such that no edge connects nodes in the same subset.
- A graph is complete if $E = \{\{v_j, v_j\} : v_i, v_j \in N \ \wedge i \leq j\}$.



Bipartite graph
$N_1 = \{1,2,3\}, \quad N_2 = \{4,5\}$



Complete graph

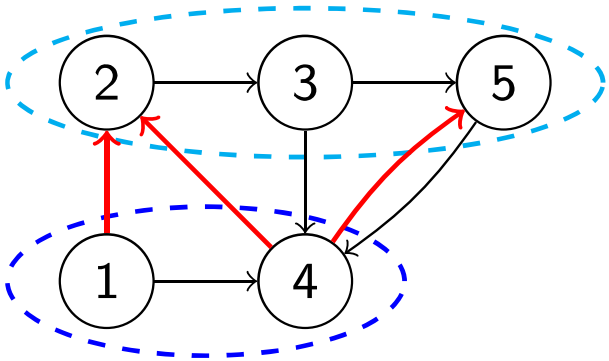
- Given a directed graph $G = (N, A)$ and $S \subset N$, the outgoing cut induced by S is

$$\delta^+(S) = \{(u, v) \in A : u \in S \ \wedge v \in N \backslash S\},$$

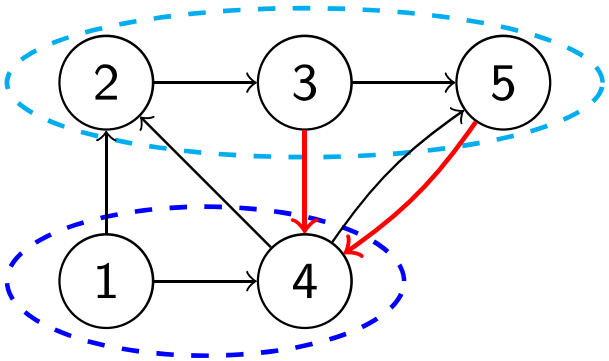
the incoming cut induced by S is

$$\delta^-(S) = \{(u, v) \in A : v \in S \ \wedge u \in N \backslash S\}.$$

Example



$\delta^+(\{1,4\}) = \{(1,2),(4,2),(4,5)\}$
$S = \{1,4\}$
$N \backslash S = \{2,3,5\}$



$\delta^-(\{1,4\}) = \{(3,4),(5,4)\}$
$S = \{1,4\}$
$N \backslash S = \{2,3,5\}$

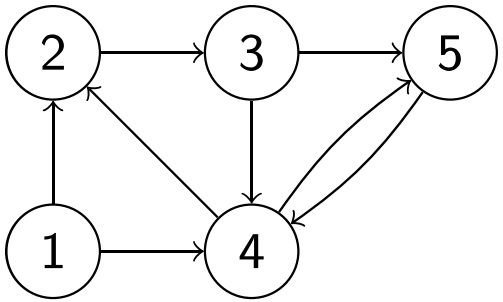
Graph representation

- A (directed) graph with n nodes has at most ($m = n(n-1)$ arcs) $m = \frac{n(n-1)}{2}$) edges.
- The graph is dense if $m \approx n^2$ and sparse if $m \ll n^2$.
- For dense directed graphs, $n \times n$ adjacency matrix:

$$a_{ij} = 1 \text{ if } (i,j) \in A \quad \text{and} \quad a_{ij} = 0 \text{ otherwise.}$$

- For sparse directed graphs, list of successors (or predecessors) for each node.
- Similar for undirected graphs.

Example



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$S(1) = \{2,4\}$$
$$S(2) = \{3\}$$
$$S(3) = \{4,5\}$$
$$S(4) = \{2,5\}$$
$$S(5) = \{4\}$$

Graph reachability problem

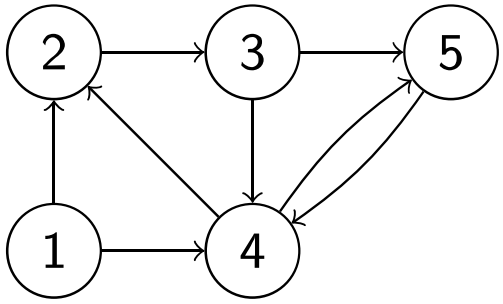
- Given a directed graph $G = (N, A)$ and a node s, determine all the nodes that are reachable from s.

Goal

- Input: Graph $G = (N, A)$, described by the successor lists and node $s \in N$.
- Output: Subset $M \subseteq N$ of nodes of G reachable from s.
- Devise an (efficient) algorithm that allows to find all nodes reachable from s.

We use a "queue" Q containing the nodes reachable from s and not yet processed (First-In First-Out policy).

Example



$Q = \{2\}$ $\quad M = \emptyset$

$Q = \{\not{2}\}$ $\quad M = \{2\}$

$Q = \{\not{3}\}$ $\quad M = M \cup \{3\}$

$Q = \{\not{4}, 5\}$ $\quad M = M \cup \{4\}$

$Q = \{\not{5}\}$ $\quad M = M \cup \{5\}$

$Q = \emptyset$

$M = \{2, 3, 4, 5\}$ is the set of nodes labeled, ie, the set of nodes reachable from $s = 2$.

Algorithm for the graph reachability problem

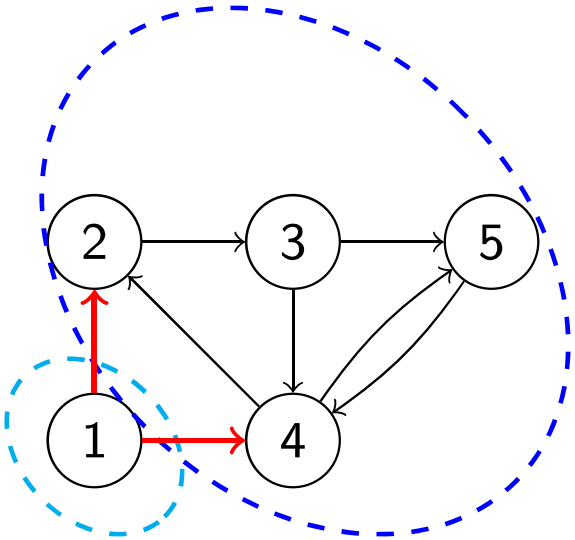
- Input: Graph $G = (N, A)$, described by the successor lists and node $s \in N$.
- Output: Subset $M \subseteq N$ of nodes of G reachable from s.

Algorithm 1: Graph reachability problem

```
1 Q ← {s}; M ← ∅
2 while Q ≠ ∅ do
3 |   u ← node in Q; Q ← Q\{u}
4 |   M ← M ∪ {u}/* label u                                   */
5 |   for (u, v) ∈ δ+(u) do      look all the arcs that start in that node
6 |   |   if v ∉ M and v ∉ Q then  Q ← Q ∪ {v}
```

If Q is managed as a FIFO queue, the nodes are explored by breadth-first search.

Graph reachability problem



- The algorithm stops when $\delta^+(M) = \emptyset$.
- $\delta^-(M)$ is the set of arcs with head node in M and tail node in $N \setminus M$.

Complexity of algorithms

- An algorithm for a problem is a sequence of instructions that allows to solve any of its instances.
- The execution time of an algorithm depends on:
  - the instance,
  - the computer.
- We want to evaluate the complexity of the algorithm as a function of the size of the instance independently from the hardware.
- Thus, we consider the number of elementary operations (e.g., arithmetic operations, comparisons, memory accesses, ...) and assume all have the same cost.

Example

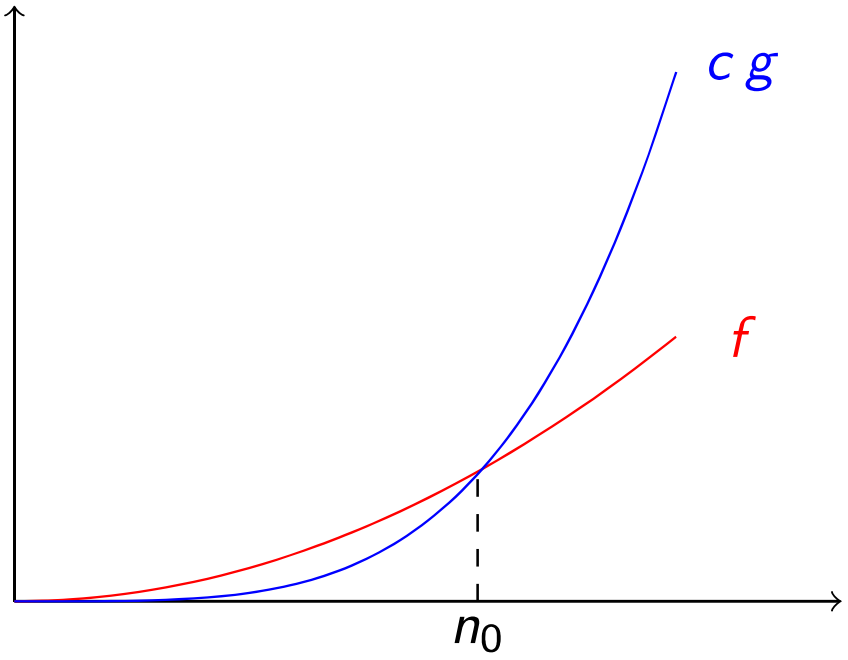
1. The dot product of $a, b \in \mathbb{R}^n$ requires n multiplications and $n-1$ additions, ie, $2n-1$ elementary operations.
2. Given two matrices $A, B \in \mathbb{R}^{n \times n}$, the product AB requires $(2n-1)n^2$ elementary operations.

Complexity of algorithms

- It is usually hard to determine the exact number of elementary operations (as a function of the instance size), so we consider the asymptotic number of elementary operations (speed of growth) in the worst case (for the worst instances).
- We look for a function $f(n)$ which is (asymptotically) an upper bound on the number of elementary operations needed to solve any instance of size at most n.

Big-O notation

- A function f is order of g, written $f(n) = O(g(n))$, if $\exists c > 0$ such that $f(n) \leq c\, g(n)$ (asymptotically), for n sufficiently large.



Example

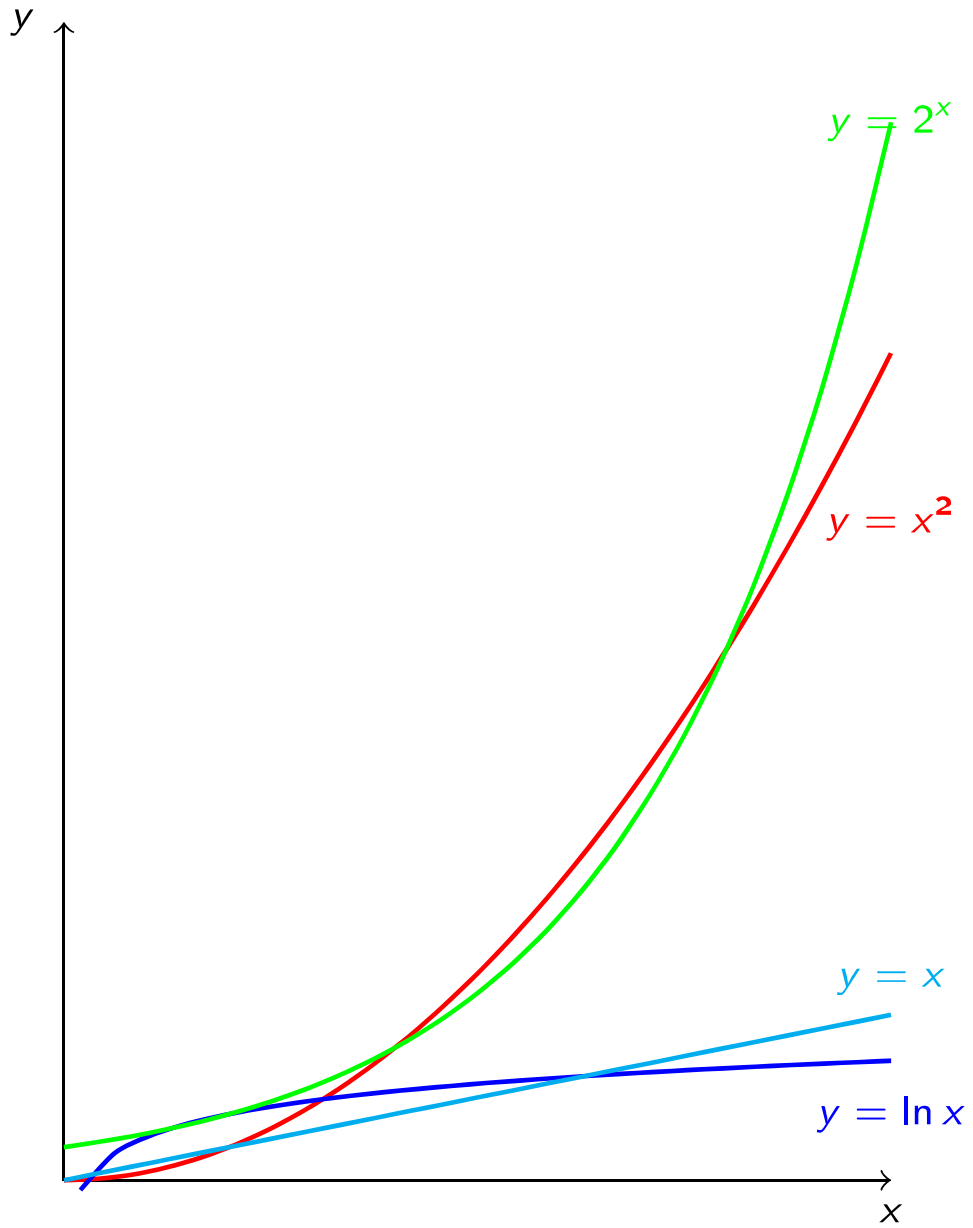
- $4n^3 - n^2 + n$ is of $O(n^3)$
- $\frac{n(n-1)}{2} + 1$ is of $O(n^2)$

Complexity of algorithms

Two classes of algorithms are distinguished according to their worst-case order of complexity:

- polynomial: $O(n^d)$ for a given constant d
- exponential: $O(2^n)$

Algorithms with a high order polynomial complexity (such as $O(n^8)$) are not efficient in practice!



Complexity of algorithms

Two classes of algorithms are distinguished according to their worst-case order of complexity:

- polynomial: $O(n^d)$, $d \in \mathbb{R}$
- exponential: $O(2^n)$

Assuming that 1 microsecond is needed per elementary operation:

n	n^2	2^n
1	0.000001 secs	0.000002 secs
10	0.0001 secs	0.001024 secs
20	0.0004 secs	1.048576 secs
30	0.0009 secs	18 mins
40	0.0016 secs	13 days
50	0.0025 secs	36 years
60	0.0036 secs	366 centuries

Complexity of the reachability algorithm

At each iteration of the `while` loop:

- Select one node $u \in Q$, extract it from Q and insert it in M.
- For all nodes v directly reachable from u and not already in M or Q, insert v in Q.

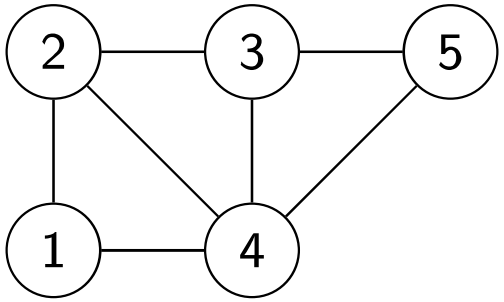
Since each node u is inserted in Q at most once and each arc (u, v) is considered at most once, the overall complexity is

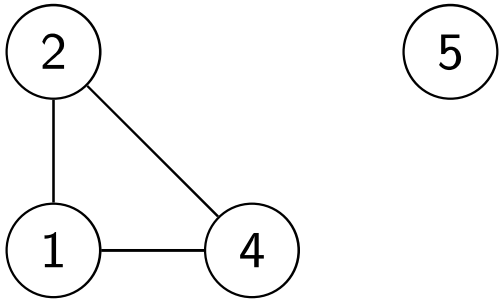
$$O(n + m), \text{ where } n = |N|, m = |A|.$$

Note: For dense graphs, $m = O(n^2)$.

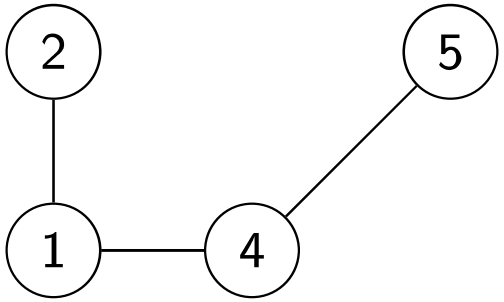
- $G' = (N', E')$ is a subgraph of $G = (N, E)$ if $N' \subseteq N$ and $E' \subseteq E$.
- A tree $G_T = (N', T)$ of G is a connected and acyclic subgraph of G.
- $G_T = (N', T)$ is a spanning tree of G if it contains all nodes in G (ie, $N = N'$). Albero contenente tutti i nodi del grafo ma solo un sottoinsieme degli archi (albero ricoprente)
- The leaves of a tree are the nodes of degree 1.

Example

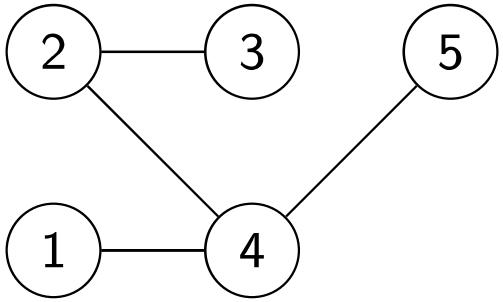




Subgraph of G



Subgraph of G and tree



Spanning tree of G

Properties of trees

Property 1

Every tree with n nodes has $n-1$ edges. Tutti i nodi sono collegati al padre, tranne la radice

Proof.

- Base case: The claim holds for $n = 1$ (tree with 1 node and 0 edges).
- Inductive step: *Show that, if this is true for trees with n nodes, then it is also true for those with $n+1$ nodes.*

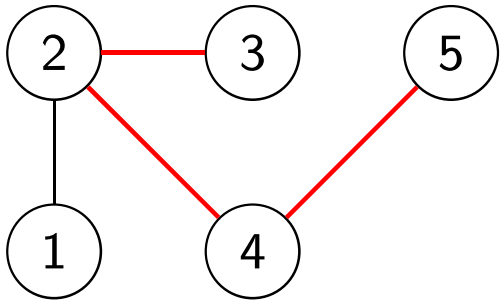
  Let T_1 be a tree with $n+1$ nodes and recall that any tree with $n \geq 2$ nodes has at least 2 leaves.
  By deleting one leaf and its incident edge, we obtain a tree T_2 with n nodes.
  By induction hypotesis, T_2 has $n-1$ edges. Therefore, the tree T_1 has $n-1+1=n$ edges.

□

Properties of trees

Property 2

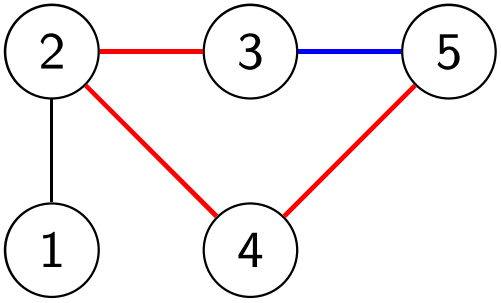
Any pair of nodes in a tree is connected via a unique path.



. . . otherwise there would be a cycle!

Property 3

By adding a new edge to a tree, we create a unique cycle.



. . . consisting of the path in Property 2 and the new edge.

Exchange property

Property 4

Let $G_T = (N, T)$ be a spanning tree of $G = (N, E)$. Consider an edge $e \notin T$ and the unique cycle C of $T \cup \{e\}$ (as in Property 3).
For each edge $f \in C \setminus \{e\}$, the subgraph $T \cup \{e\} \setminus \{f\}$ is also a spanning tree of G.