2.2. Optimal cost spanning trees

Spanning trees have a number of applications:

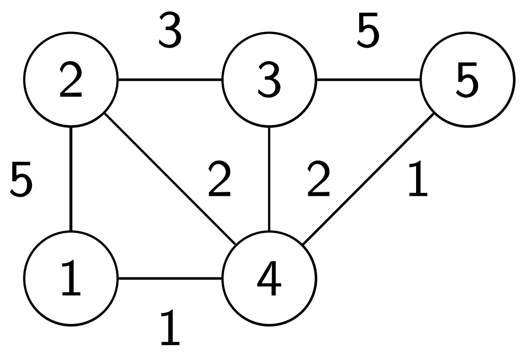
- network design (communication, electrical, ...)
- IP network protocols
- compact memory storage (DNA)
- ...

Example

Design a communication network so as to connect n cities at minimum total cost.

Model

Graph $G = (N, E)$ with $n = |N|$, $m = |E|$, and a cost function $c : E \to c_e \in \mathbb{R}$, with $e = \{u, v\} \in E$.



Required properties:

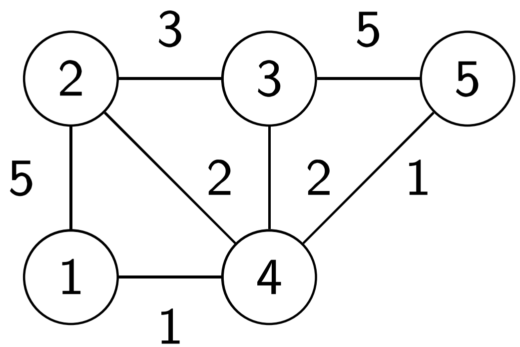
1. Each pair of cities must communicate $\Rightarrow$ connected subgraph containing all the nodes.
2. Minimum total cost $\Rightarrow$ subgraph with no cycles.

Problem

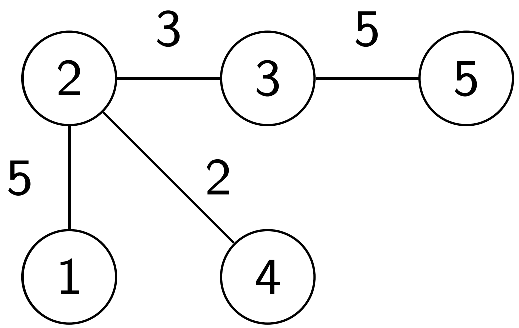
Given an undirected graph $G = (N, E)$ and a cost function, find a spanning tree $G_T = (N, T)$ of minimum total cost.

$$\min_{T \in X} \sum_{e \in T} c_e, \quad \text{where } X \text{ is the set of all spanning trees of } G.$$

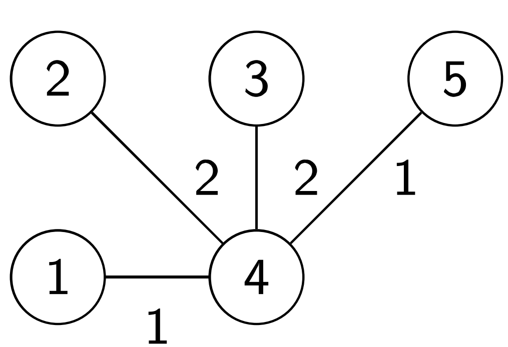
Example



G



$c(T_1) = 15$
(feasible solution)



$c(T_2) = 6$
(feasible and optimal solution)

Theorem 1 (Cayley, 1889)

A complete graph with n nodes ($n \geq 1$) has n^{n-2} spanning trees.

Example

- K_3 ($n = m = 3$) has $3^{3-2} = 3$ spanning trees.
- K_5 ($n = 5,\ m = 10$) has $5^{5-2} = 125$ spanning trees.

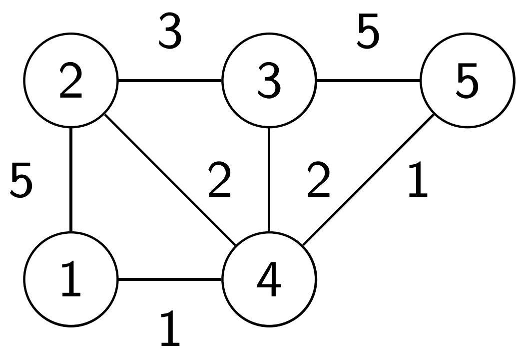
Prim's algorithm

Idea: Iteratively build a spanning tree.

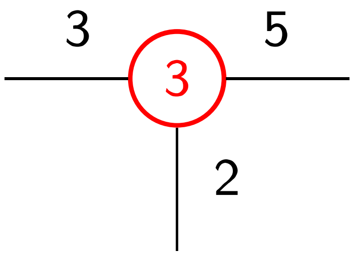
Method

- Start from initial tree (S, T) with $S = \{u\}$ and $T = \emptyset$ (u can be any node in N).
- At each step, add to the current partial tree (S, T) an edge of minimum cost among those which connect a node in S to a node in $N \backslash S$.

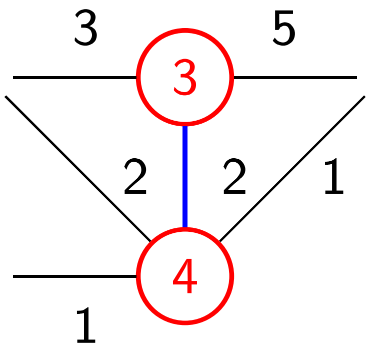
Example



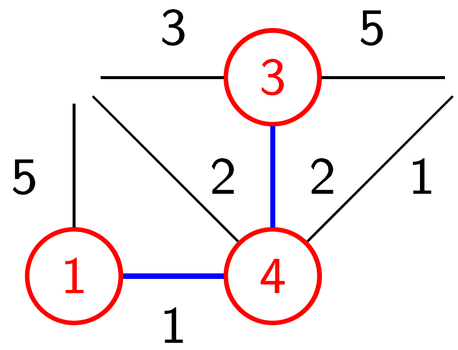
Example



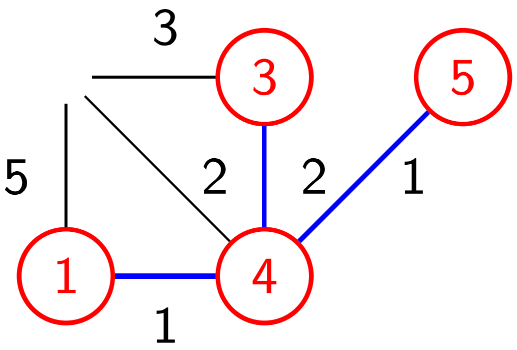
$S = \{3\}$
$T = \emptyset$



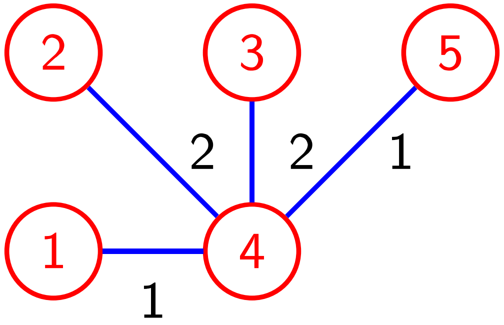
$S = \{3, 4\}$
$T = \{\{3, 4\}\}$



$S = \{1, 3, 4\}$
$T = \{\{3, 4\}, \{1, 4\}\}$



$S = \{1, 3, 4, 5\}$
$T = \{\{3, 4\}, \{1, 4\}, \{4, 5\}\}$



$S = N$
$T = \{\{3, 4\}, \{1, 4\}, \{4, 5\}, \{2, 4\}\}$

$c(T) = 6$

Prim's algorithm

- Input: Connected graph $G = (N, E)$ with edge costs
- Output: Subset $T \subseteq N$ of edges of G such that $G_T = (N, T)$ is a minimum cost spanning tree of G.

Algorithm 2: Prim's algorithm for the minimum cost spanning tree problem

---

$S \leftarrow \{u\}$ Inserisco il primo nodo

$T \leftarrow \emptyset$ Inizializzo vuoto il subset degli edges

while $|T| < n - 1$ **do**

  $\{u, v\} \leftarrow$ edge in $\delta(S)$ of minimum cost /* $u \in S$ and $v \in N \backslash S$ */ inserisco nell'edge {u,v} quello uscente da u che ha costo minore

  $S \leftarrow S \cup \{v\}$ Inserisco in S il nodo di arrivo dell'arco a costo minimo scelto al passo precedente

  $T \leftarrow T \cup \{\{u, v\}\}$ Inserisco tale edge nel subset degli edge che formeranno l'albero

---

If all edges are scanned at each iteration, the complexity order is: $O(nm)$.

In fact, for every node I need to check every edge from that node

Exactness of Prim's algorithm

Definition

An algorithm is exact if it provides an optimal solution for every instance. Otherwise, it is heuristic.

Proposition 1

Prim's algorithm is exact.

The exactness does not depend on the choice of the first node nor on the selected edge of minimum cost in $\delta(S)$.

We show that each selected edge belongs to a minimum spanning tree.

Property 5 (Cut property)

Let F be a partial tree (spanning nodes in $S \subseteq N$) contained in an optimal tree of G. Consider $e = \{u, v\} \in \delta(S)$ of minimum cost, then there exists a minimum cost spanning tree of G containing e.

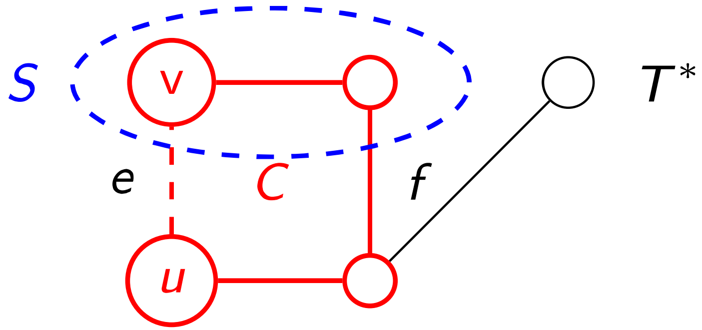
Proof.

By contradiction, assume $T^* \subseteq E$ is a minimum cost spanning tree with $F \subseteq T^*$ and $e \notin T^*$.

Adding edge e to T^* creates the cycle C. Let $f \in \delta(S) \cap C$.

- If $c_e = c_f$, then $T^* \cup \{e\} \setminus \{f\}$ is (also) optimal since it has same cost of T^*.
- If $c_e < c_f$, then $c(T^* \cup \{e\} \setminus \{f\}) < c(T^*)$, hence T^* is not optimal.

□



Definition

A greedy algorithm constructs a feasible solution iteratively by making at each step a "locally optimal" choice, without reconsidering previous choices.

Note: Prim's algorithm is a greedy algorithm.

At each step a minimum cost edge is selected among those in the cut $\delta(S)$ induced by the current set of nodes S.

Note: For most discrete optimization problems greedy-type algorithms yield a feasible solution with no guarantee of optimality.

There are various greedy algorithms for the minimum cost spanning tree problem based on the cut property:

- Boruvka (1926);
- Kruskal (1956) – Exercise 2.2;
- Prim (1957), ...

Implementation in $O(n^2)$

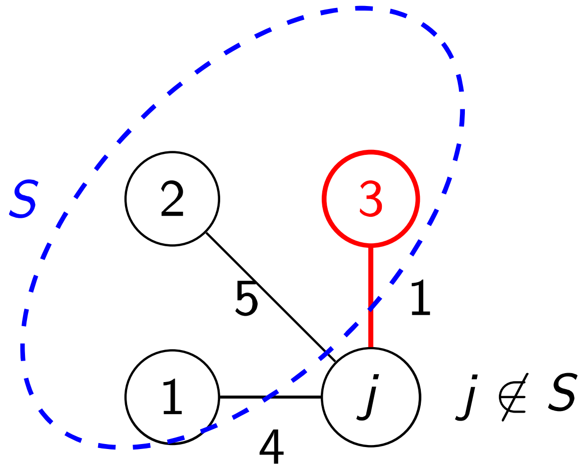
Data structure

- k: number of edges selected so far
- Subset $S \subseteq N$ of nodes incident to the selected edges
- Subset $T \subseteq E$ of selected edges
- $C_j = \begin{cases} \min\{c_{ij} : i \in S\}, & j \notin S \\ +\infty, & \text{otherwise} \end{cases}$

  minimum cost
  of an edge that links j to the set of nodes already included in the tree. That is the best edge to reach j
- $closest_j = \begin{cases} \text{argmin}\{c_{ij} : i \in S\}, & j \notin S \\ \text{"predecessor" of } j \text{ in the minimum spanning tree}, & j \in S \end{cases}$

  ending node of an edge that has a minimum cost and ends in j

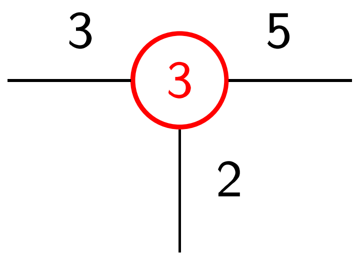
vectors: for every j there's a cell

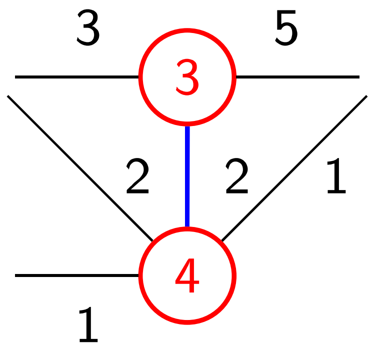


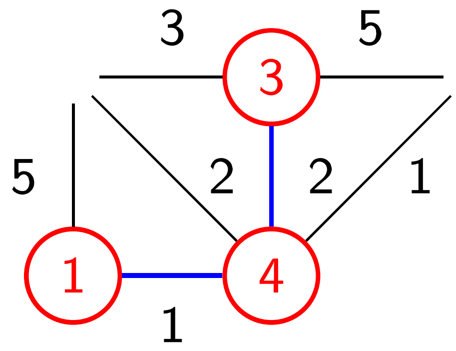
$closest_j = 3$

$c_{closest_j, j} = 1$

Example







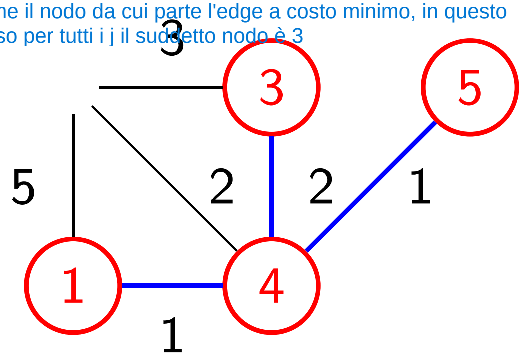
$S = \{3\}$
$T = \emptyset$
$closest = [3, 3, -, 3, 3]$

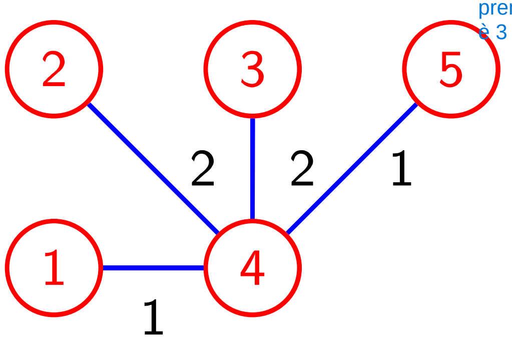
Per ogni nodo che non si trova in S, assegnamo il closest come il nodo da cui parte l'edge a costo minimo, in questo caso per tutti i j il suddetto nodo è 3

$S = \{3, 4\}$
$T = \{\{3, 4\}\}$
$closest = [4, 4, -, 3, 4]$

essendo il vertice 4 incluso in S, prendiamo come closest il predecessore nell'albero, che è 3

$S = \{1, 3, 4\}$
$T = \{\{3, 4\}, \{1, 4\}\}$
$closest = [4, 4, -, 3, 4]$



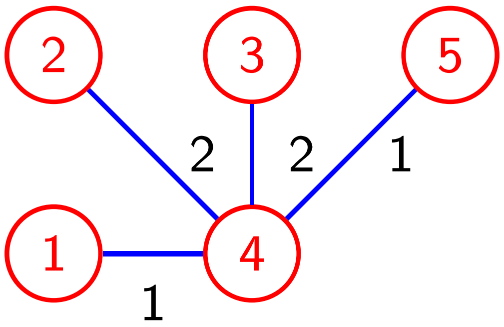


$S = \{1, 3, 4, 5\}$
$T = \{\{3, 4\}, \{1, 4\}, \{4, 5\}\}$
$closest = [4, 4, -, 3, 4]$

$S = N$
$T = \{\{3, 4\}, \{1, 4\}, \{4, 5\}, \{2, 4\}\}$
$closest = [4, 4, -, 3, 4]$

How to retrieve the spanning tree from *closest*?
The minimum spanning tree found by Prim's algorithm consists of the $n-1$ edges: $\{closest_j, j\}$, with $j = 1, 2, \ldots, n$.

Example:
Since $closest = [4, 4, -, 3, 4]$, a minimum cost spanning tree consists of the edges: $\{4, 1\}, \{4, 2\}, \{3, 4\}, \{4, 5\}$



$O(n^2)$ version of Prim's algorithm

Algorithm 3: Prim's algorithm

$T \leftarrow \emptyset$; $S \leftarrow \{u\}$ /* `Initialization` */

for $j \in N \setminus S$ **do** per tutti i nodi non in S, assegnamo il valore di C e il closest

    $C_j \leftarrow c_{uj}$ /* `Or` $+\infty$ `if` $\{u, j\} \notin E$ minimo costo di un edge che dai nodi in S (in questo caso solo u) arriva in j */

    $closest_j \leftarrow u$ nodo in S da cui arriva l'arco a costo minimo (in questo caso c'è solo il nodo u in S)

for $k = 1, \ldots, n-1$ **do**

    $min \leftarrow +\infty$ /* `Select min edge in` $\delta(S)$ */

    for $j = 1, \ldots, n$ **do**

        if $j \notin S$ *and* $C_j < min$ **then**

            $min \leftarrow C_j$; $v \leftarrow j$

    $S \leftarrow S \cup \{v\}$; $T \leftarrow T \cup \{\{closest_v, v\}\}$ /* `Extend` S `and` T */

    for $j = 1, \ldots, n$ **do**

        if $j \notin S$ *and* $c_{vj} < C_j$ **then**

            $C_j \leftarrow c_{vj}$; $closest_j \leftarrow v$ /* `Update` C_j `and` $closest_j$, $\forall j \notin S$ */

Complexity order of Prim's algorithm

for $j = 2, \ldots, n$ **do**
  $O(1)$

for $k = 1, \ldots, n-1$ **do**
  for $j = 2, \ldots, n$ **do**
    $O(1)$
  for $j = 2, \ldots, n$ **do**
    $O(1)$

Overall complexity: $n - 1 + (n-1)(n-1+n-1)$, ie, $O(n^2)$

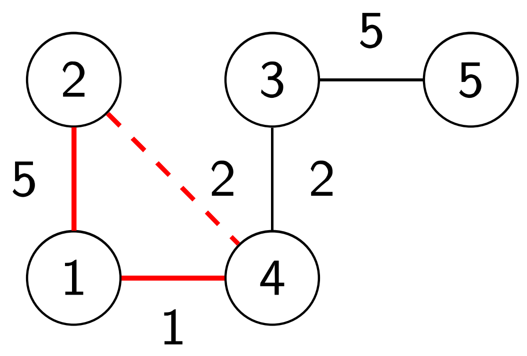
For sparse graphs, where $m \ll n(n-1)/2$, a more sophisticated data structure leads to an $O(m \log n)$ complexity.

Optimality condition

- Given a spanning tree T, an edge $e \notin T$ is cost decreasing if when e is added to T it creates a cycle C with $C \subseteq T \cup \{e\}$ and $\exists f \in C \setminus \{e\}$ such that $c_e < c_f$.

Example

Adding a new edge we create a cycle, if after that adding we can delete an edge that there were before because it costs more, then the added edge is cost decreasing: it help us decreasing the cost of the tree



$c_{24} = 2 < 5 = c_{12}$

Because $c(T \cup \{e\} \setminus \{f\}) = c(T) + c_e - c_f$, if e is cost decreasing, then

$$c(T \cup \{e\} \setminus \{f\}) < c(T).$$

Theorem 2 (Tree optimality condition)

A tree T is of minimum total cost if and only if no cost-decreasing edge exists.

Proof.

$\Rightarrow$ If a cost-decreasing edge exists, then T is not of minimum total cost.

$\Leftarrow$ If no cost-decreasing edge exists, then T is of minimum total cost.
Let T^* be a minimum cost spanning tree found by Prim's algorithm.
It can be verified that, by exchanging one edge at a time, T^* can be iteratively transformed into T without modifying the total cost.
Thus, T is also optimal.

$\square$

Optimality test

The optimality condition allows to verify whether a spanning tree T is optimal:

- It suffices to check that each $e \in E \setminus T$ is not a cost-decreasing edge.

An indirect application: optimal message passing

Given a communication network $G = (N, E)$, we want to broadcast a secret message to all the nodes so that it is not intercepted along any edge.

Problem

Let $p_{ij} \in [0, 1]$, be the probability the message is intercepted along edge $\{i, j\} \in E$. How to broadcast the message to all the nodes of G, so as to minimize the probability of interception along any edge?

- Or, equivalently, maximize the probability of non-interception along all the edges:

$$\max \Pi_{\{i,j\} \in T} (1 - p_{ij}).$$

The optimal solution is a spanning tree:

- Broadcasting to all nodes $\Rightarrow$ connected subgraph
- To avoid redundancy and a higher probability of interception $\Rightarrow$ acyclic subgraph

Indirect application: optimal message passing

Additionally, by applying a monotone increasing function (like log), the objective function values change, but the optimal solutions remain unchanged. Thus, the problem becomes:

$$\max \log \left(\Pi_{\{i,j\} \in T} (1 - p_{ij}) \right) = \max \sum_{\{i,j\} \in T} \log(1 - p_{ij}).$$

The latter problem can be solved using a straightforward adaptation of any minimum cost spanning tree algorithm.