2.2 Optimal cost spanning trees

Spanning trees have a number of applications:

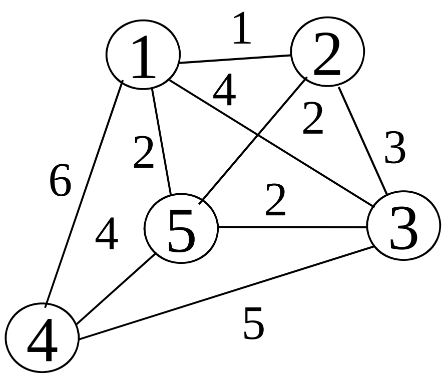
- network design (communication, electrical,...)
- IP network protocols
- compact memory storage (DNA)
- ...

2.2.1 Minimum cost spanning tree problem

Example

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

$n = 5$



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

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.

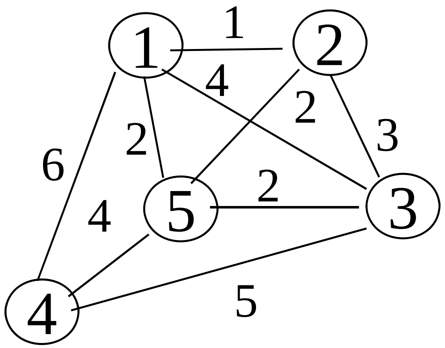
Problem

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

$$\min_{T \in X} \sum_{e \in T} c_e$$

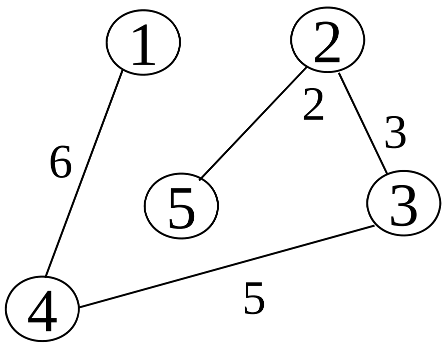
where X is the set of all spanning trees of G

Example

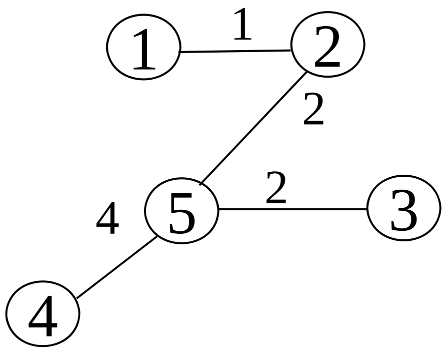


$G = (N,E)$

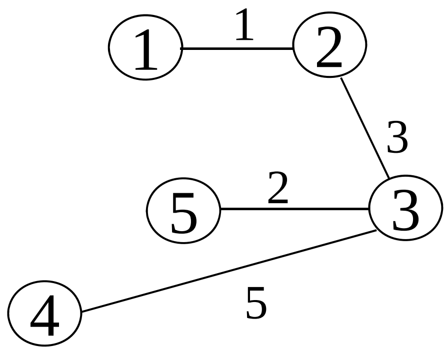
Some feasible solutions:



$c(T_1)=16$



$c(T_2)=9$

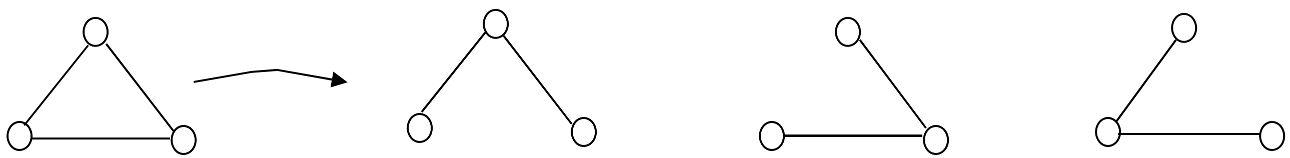


$c(T_3)=11$

Theorem (A. Cayley 1889)

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

Examples: K_3 (n=3, m=3 edges) has 3 spanning trees



K_5 (n=5, m=10) has 125 spanning trees

<u>Recall</u>: A tree with n nodes has $n - 1$ edges.

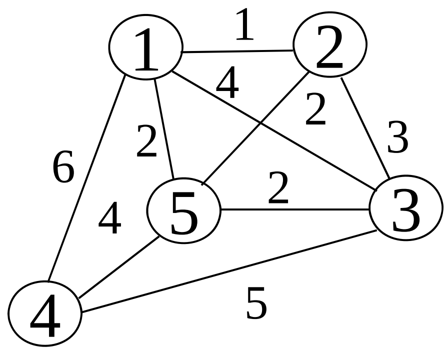
2.2.2 Prim's algorithm

Idea: Iteratively build a spanning tree.

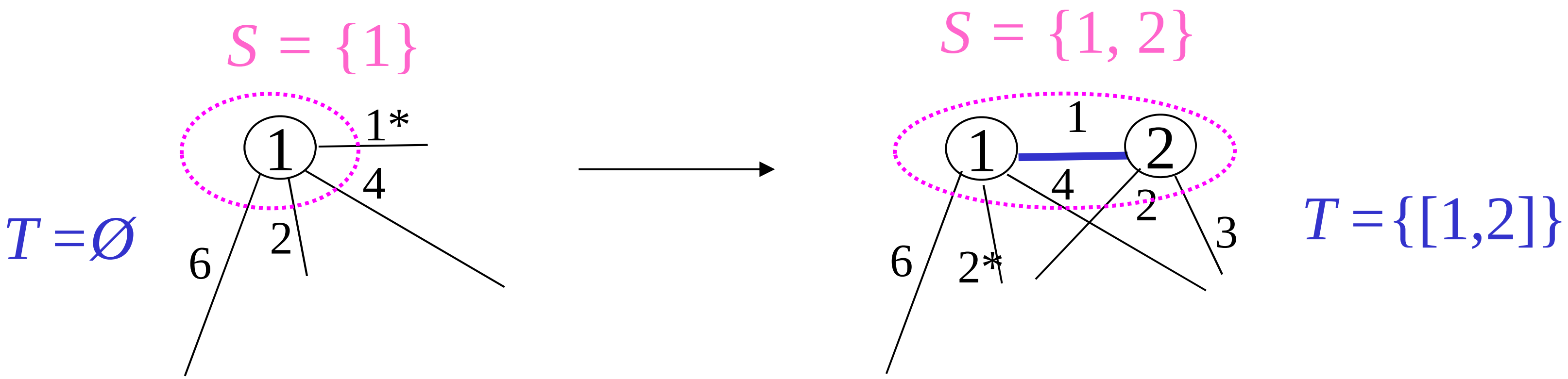
Start from tree (S, T) where S contains an arbitrary node and $T=\emptyset$.

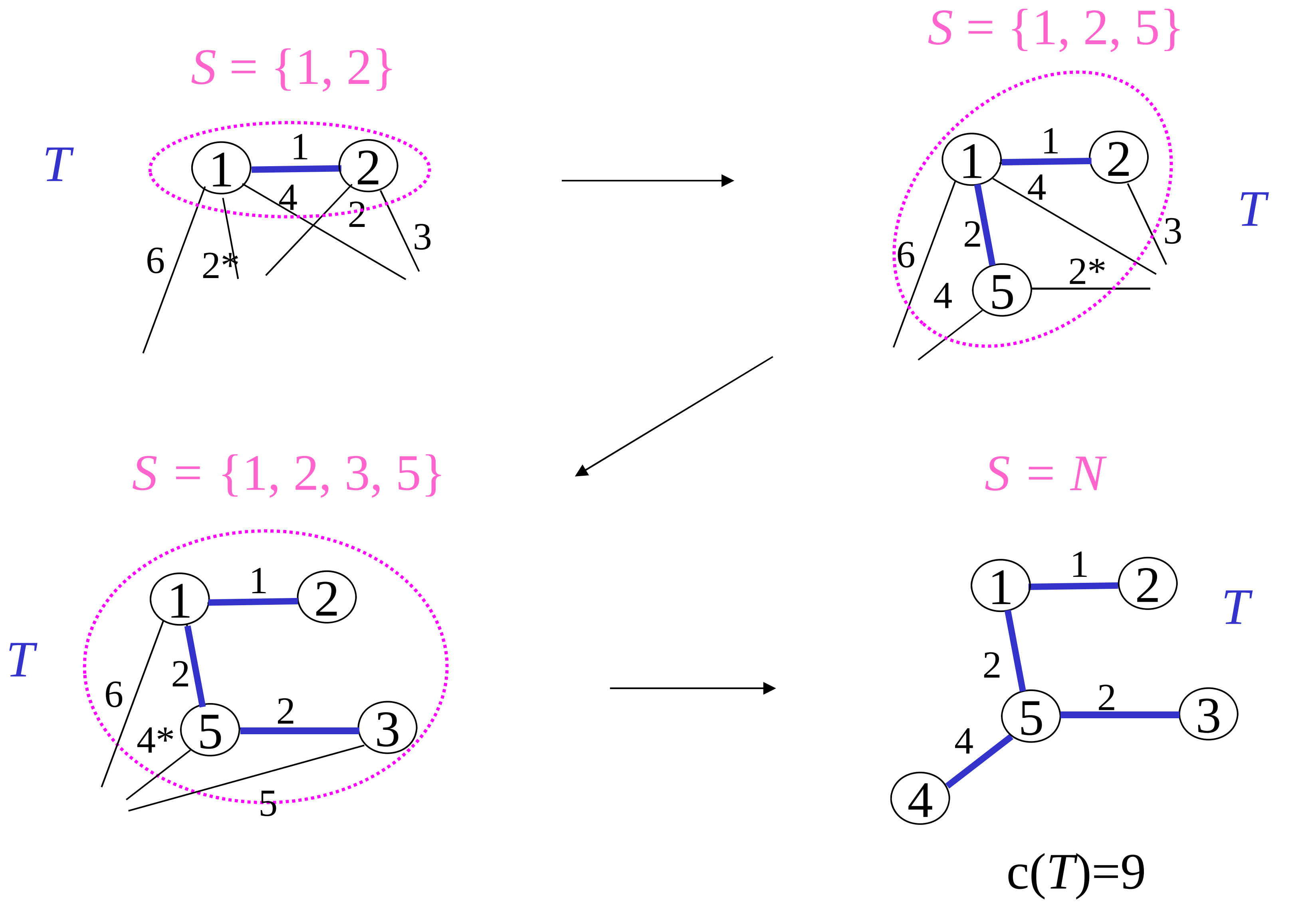
At each step, add to the current partial tree (S, T) an edge of minimum cost among those which connect a node in S to a node in $N \setminus S$.

Given $G = (N, E)$ with edge costs



Procedure:




S = {1, 2}
T
1
2
1
4
2
6
2*
3
S = {1, 2, 5}
T
1
2
5
1
4
2
6
3
2*
4
S = {1, 2, 3, 5}
T
1
2
5
3
1
2
6
2
4*
5
S = N
T
1
2
5
3
4
1
2
2
4
c(T)=9

Pseudocode of Prim's algorithm

Input: Connected $G = (N, E)$ with edge costs

Output: Subset of edges $T \subseteq E$ such that $G_T = (N, T)$ is a spanning tree of G

```
BEGIN
  S:={1}; T:= ∅;
  WHILE |T| < n-1 DO   /* a tree with n nodes has n-1 edges */
    Select an edge [v,h] ∈ δ(S) of minimum cost (v ∈ S and h ∈ N \ S);
    T := T ∪ {[v,h]};
    S := S ∪ {h};
  END-WHILE
END
```

If all edges are scanned at each iteration, complexity: O(nm)

2.2.3 Exactness of Prim's algorithm

Definition: An algorithm is ***exact*** if it provides an optimal solution for every instance, otherwise it is ***heuristic***.

Proposition: Prim's algorithm is exact.

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

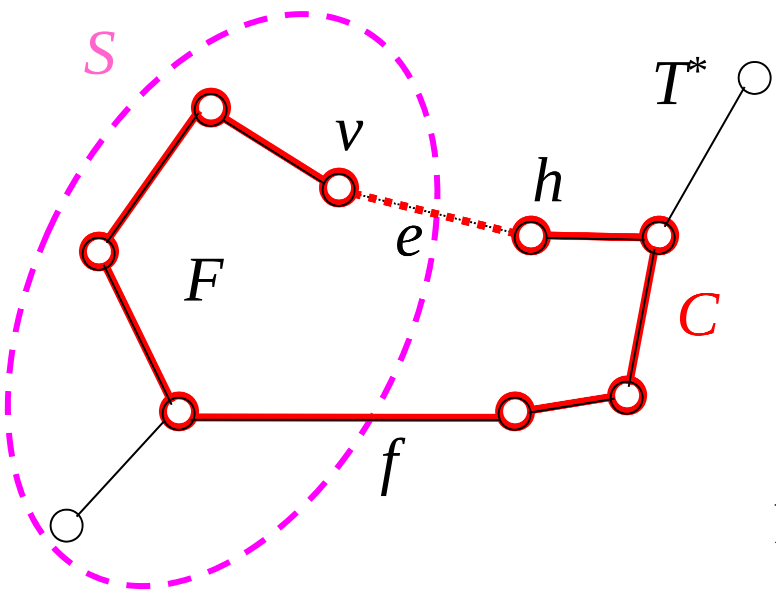
As we shall see, exactness does not depend on the choice of the first node or of the selected edge of minimum cost in $\delta(S)$.

Cut property

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

Proof

By contradiction: Let $T^* \subseteq E$ be a minimum cost spanning tree with $F \subseteq T^*$ and $e \notin T^*$.



Adding edge e 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.

Observation: 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.

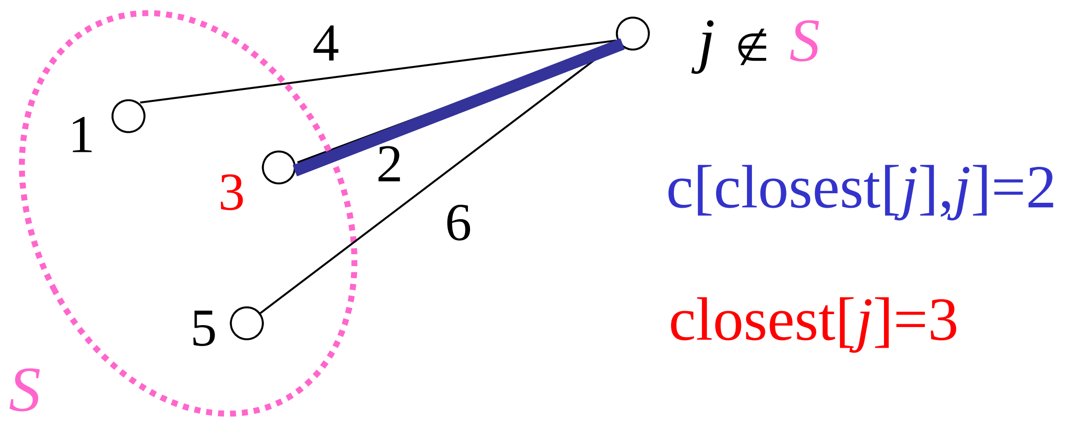
N.B. For most discrete optimization problems greedy-type algorithms yield a feasible solution with no guarantee of optimality.

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

- Boruvka (1926)
- Kruskal (1956) -- Exercise 2.2
- Prim (1957)
- …

2.2.4 $O(n^2)$ version of Prim's algorithm

- Subset $T \subseteq E$ of selected edges
- Subset $S \subseteq N$ of nodes incident to the selected edges

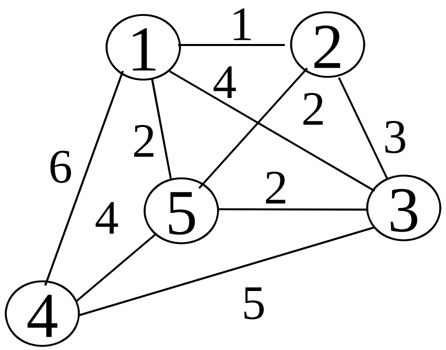


- closest[j] = $\begin{cases} \text{argmin } \{c_{ij} : i \in S\}, \text{ for } j \notin S \\ \text{"predecessor" of } j \text{ in the min spanning tree, for } j \in S \end{cases}$

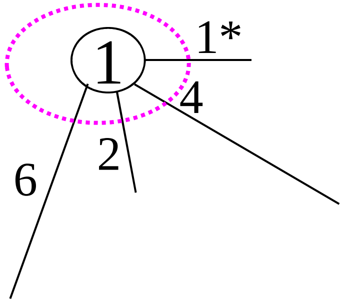
Pseudocode

```
BEGIN

 S := {1}; T := ∅;
 FOR EACH j ∉ S DO
     closest[j] := 1;
 END-FOR

 WHILE  |T| < n-1 DO     /* select n – 1 edges of the tree */
   Find h ∉ S such that [closest[h],h] ∈ δ(S) is of minimum cost;
   S := S ∪ {h}; T := T ∪ {[closest[h],h]};
   FOR EACH j ∉ S DO      /* update closest[j] */
     IF (c_hj < c[[closest[j],j]) THEN closest[j]:= h; END-IF
   END-FOR
 END-WHILE

END
```

Example



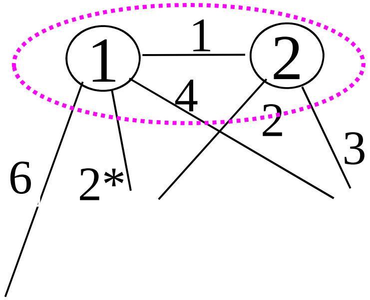
$S = \{1\}$



$T = \emptyset$

<u>closest</u> = (1, 1, 1, 1, 1)

$S = \{1, 2\}$

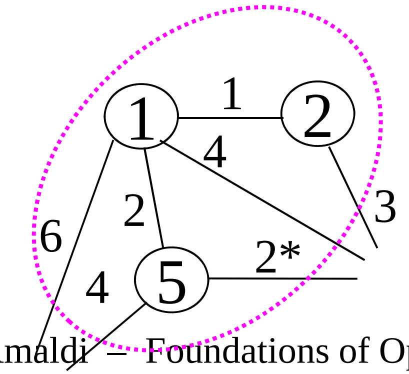


$T = \{[1,2]\}$

<u>closest</u> = (1, 1, <u>2</u>, 1, 1)

since $c_{23} < c_{13}$ closest[3]:=2

$S = \{1, 2, 5\}$



$T = \{[1,2], [1,5]\}$

<u>closest</u> = (1, 1, <u>5</u>, <u>5</u>, 1)

etc...

Complexity

```
BEGIN
  initialization
  FOR EACH j ∉ S DO
        ... END-FOR

  WHILE |T| < n-1 DO
    ...

    FOR EACH j ∉ S DO
         ... END-FOR
  END-FOR
END
```

1 initialization requires $O(n)$

3 The internal FOR cycle requires $O(n)$

2 The external cycle is executed n - 1 times

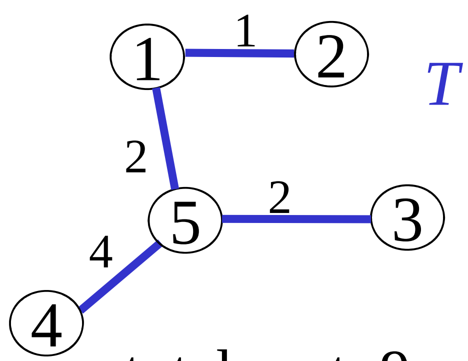
Overall complexity: $O(n^2)$

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

How to retrive the spanning tree from <u>closest</u>?

The minimum spanning tree found by Prim's algorithm consists of the n-1 edges: [closest[j], j] with j = 2,..., n.

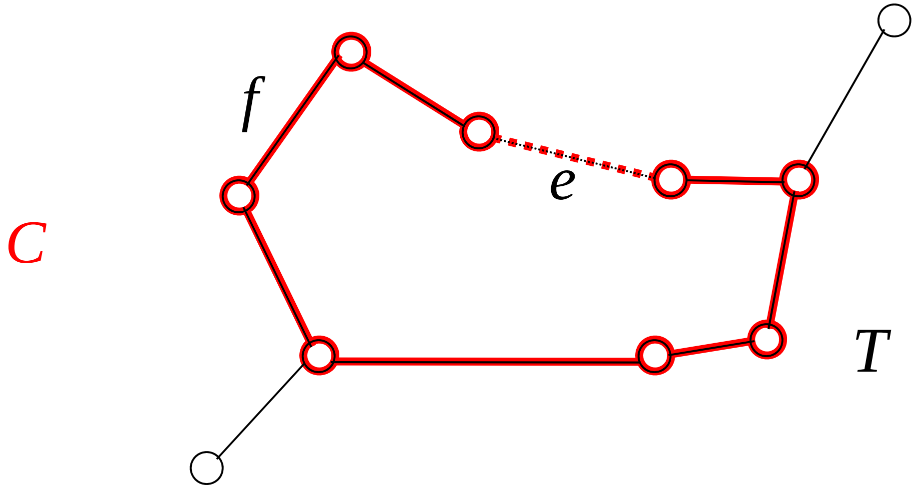
<u>Example</u>: Since <u>closest</u> =(1,1,5,5,1) a spanning tree consists of the edges:[1,2], [5,3], [5,4] and [1,5].



total cost: 9

2.2.5 Optimality condition

Definition: 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 $\exists$ an edge $f \in C \setminus \{e\}$ such that $c_e < c_f$.



$c_e < c_f$

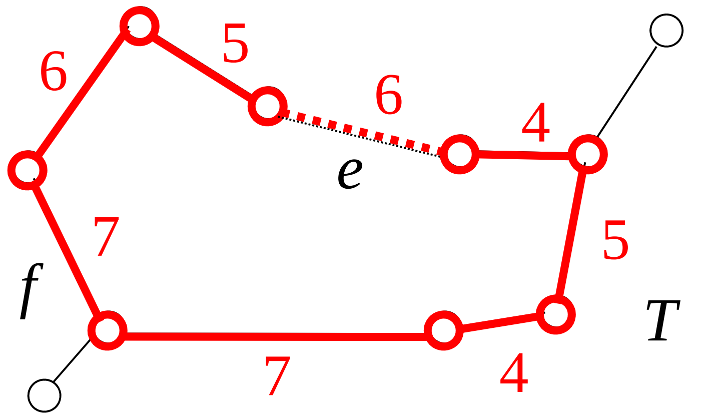
$$c\,(T \cup \{e\} \setminus \{f\}) < c(T) = \sum_{e' \in T} c_{e'}$$

Tree optimality condition

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

Proof (⇒) If a cost-decreasing edge exists, T is not of minimum total cost.

$c_e < c_f$



because the cost of T could be decreased by exchanging the cost-decreasing edge e with any f of C with $c_e < c_f$.

($\Leftarrow$)   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.

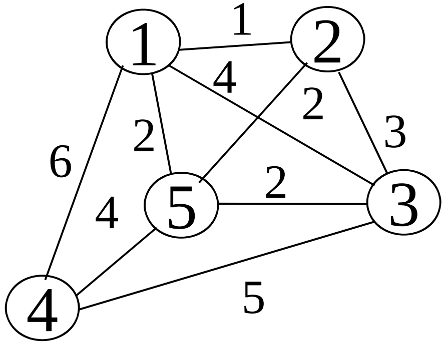
It can be verified that, by exchanging one edge at a time, T^* can be iteratively transformed into T without modifying the total cost.

Thus T is also optimal.

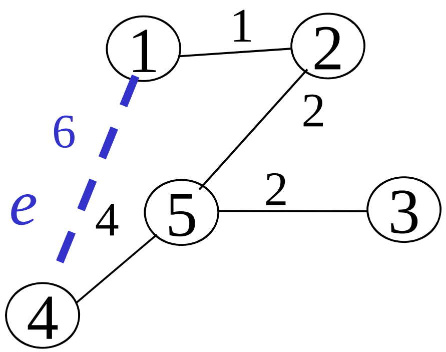
Optimality test

The optimality condition allows us to verify whether a given spanning tree G_T is optimum:

$G=(N,E)$



$G_T=(N,\mathrm{T})$



$c(T)=9$

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

2.2.6 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.

Let p_{ij}, $0 \leq p_{ij} \leq 1$, be the probability the message is intercepted along edge $[i, j] \in E$.

Problem How to broadcast the message to all the nodes of G so as to minimize the probability of interception along any edge?

Minimize the probability of interception (along any edge)

$$\Updownarrow$$

Maximize the probability of non-interception

$$\max \prod_{[i,j]\in T} (1 - p_{ij})$$

T is a <u>spanning tree</u>

- Broadcasting to all nodes $\Rightarrow$ <u>connected</u>
- <u>acyclic</u> to avoid redundancy and a higher probability of interception

By applying a montone increasing function like log(.), the optimal solutions remain unchanged (only the solution values change)

$$\Downarrow$$

$$\max \log\left(\prod_{[i,j]\in T} (1 - p_{ij}) \right) \equiv \max \sum_{[i,j]\in T} \log(1 - p_{ij})$$

Solved by a straightforward adaptation of any minimum cost spanning tree algorithm