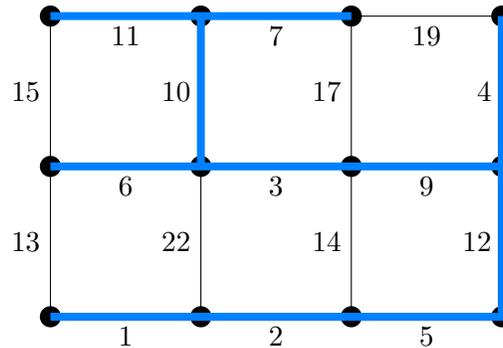


Spanning Trees

Source: Chapter 2.1 (Bills), Chapter 6.1 of Combinatorial Optimization (Korte)

Problem: Connect cities V with optic cable. For every pair of cities, it is known if the cable can be built and the cost of building it $c : V^2 \rightarrow \mathbb{R}$. Which cables should be built so that the cities are connected and the total building cost is minimized?

1: Solve the cities and cable problem for the following diagram of cities:



A *graph* $G = (V, E)$ is a pair of vertices V and edges E , where E consists of pairs of vertices.

Recall definitions of *circuit/cycle*, *tree*, *forest*, *spanning tree*, *connected components*, *path*

A **cut** for $X \subset V$ is the set of edges each with exactly one endpoint in X .

Formal definition of our problem: **Minimum spanning tree problem**

Input: Graph $G = (V, E)$ and costs $c : E \rightarrow \mathbb{R}$.

Output: Spanning tree T of minimum cost.

2: If T is a spanning, then the following are equivalent:

1. T is minimum spanning tree.
2. For every $e = \{x, y\} \in E(G) \setminus E(T)$, no edge on the x - y -path in T has higher cost than e .
3. For every $e \in E(T)$, e is a minimum cost edge of the *cut* between the connected components in $T - e$.
4. We can order $E(T) = \{e_1, \dots, e_{n-1}\}$ such that for each i there exists a set $X_i \subset V(G)$ such that e_i is the min cost edge of the cut X_i and no edge in $\{e_1, \dots, e_{i-1}\}$ is in the cut X_i .

Show $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. Try $4 \rightarrow 1$ by taking T that satisfies 4 and T^* satisfying (1) and check how they can differ.

Solution: See the book(s) for detailed solution.

(1) \rightarrow (2): If 2 violated, T was not optimal by replacing an edge

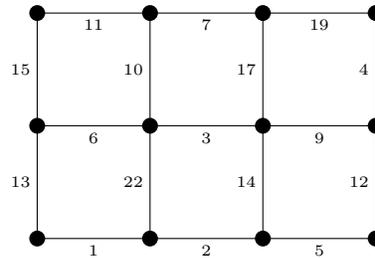
(2) \rightarrow (3): if 3 violated, so is (2)

(3) \rightarrow (4): take any ordering from (3), it gives (4). (4) \rightarrow (1): T from (4) and T^* optimum. Let e_i be the first $e_i \in T$, that is missing in T^* . Let the corresponding cut for e_i be X_i . Add e_i to T^* , it contains a circuit, one other edge of the cut X_i that is in T^* can be removed and cost of T^* decreases.

Kruskal's (greedy) algorithm [1956]

1. sort all m edges of G such that $c(e_1) \leq c(e_2) \leq \dots \leq c(e_m)$
2. set $T = (V, \emptyset)$
3. for i in 1 to m :
if $T + e_i$ does not contain a circuit, then $T := T + e$.

3: Do the steps of the algorithm on the graph with cities (note that the algorithm has 11 iterations where edge is added since the tree has 11 edges). Denote the order of edges as they enter the spanning three.



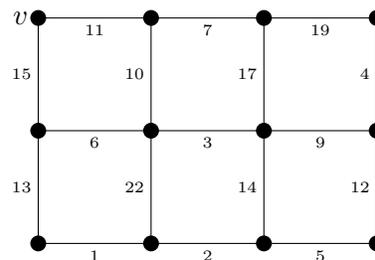
4: Why is the output of the algorithm correct?

Solution: Satisfies condition 2.

Jarník's [1930] and Prims [1957] algorithm

1. choose any $v \in V$ and $T = (\{v\}, \emptyset)$
2. while T does not contain all vertices:
pick e of minimum cost that has exactly one endpoint in T and $T := T + e$

5: Do the steps of the algorithm on the graph with cities (note that the algorithm has 11 iterations since the tree has 11 edges). Denote the order of edges as they enter the spanning three. Start with v being the left top vertex.



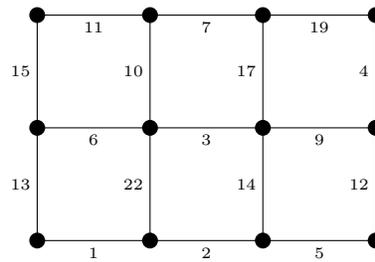
6: Why is the output of the algorithm correct?

Solution: Satisfies condition 4.

Borůvka's [1928] algorithm

1. Let $T = (V, \emptyset)$
2. while T has more than one connected component:
in parallel, for every connected component C in T , pick e of minimum cost that has exactly one endpoint in C and do $T := T + e$

7: Do the steps of the algorithm on the graph with cities. Note that one iteration always gives several edges in. The number of iterations is not clear at the beginning. Denote the order of edges as they enter the spanning tree.



8: Why is the output of the algorithm correct?

Solution: Suppose it creates a cycle. Then we get a contradiction with the choice of the edges. But we run into troubles if edges have the same weights.

Algorithmic note: Which algorithm is fastest?

The complexity of an algorithm is counted in the number of operations performed by the CPU.

Count how many times each vertex and edge is used. Ignoring the constants, we use $O(\cdot)$ notation.

Estimate the complexity of Kruskal's algorithm given a graph with m edges and n vertices.

- sorting takes $O(m \log(m))$
- for cycle is repeated up to m times
- test for a circuit can be done in $O(n)$ time

Total time: $O(m \log(m) + mn)$. Better implementation can do $O(m \log(n))$.

Jarník's algorithm can be implemented in $O(m + n \log(n))$.

More effective algorithms exist if weights are integers, graph is *planar*, ...

9: Let $G = (V, E)$ be a graph and $c : E \rightarrow \mathbb{R}$ cost function on the edges. Formulate the minimum spanning tree problem using linear programming.

(Don't be afraid that there are many constraints. Try to make constraint that graph has no cycles.)

Solution: We let every edge e_i be a variable x_i . The objective is to minimize $c_i x_i$. For every x_i we add $0 \leq x_i \leq 1$. We also can add $\sum x_i = n - 1$. But mainly we need to avoid cycles. Description of the set of feasible solutions is

$$F = \{x \in [0, 1]^E : \sum_{e \in E(G[X])} x_e \leq |X| - 1 \text{ for } \emptyset \neq X \subseteq V\},$$

where $G[X]$ is the graph induced by vertices X - contains all edges of E that have both vertices in X .

Theorem [Edmonds 1970] The set of feasible solutions is *integral* polytope - i.e. - every vertex of the polytope has all coordinates integers. The polytope is called **spanning tree polytope**.

10: Draw the spanning tree polytope for K_3 , where K_3 is the complete graph on 3 vertices.

Solution: Convex hull of points $\{1, 1, 0\}, \{0, 1, 1\}, \{1, 0, 1\}$. Notice it is two dimensional.