Chapters 5.3–5.5 Minimum Spanning Tree

A **spanning tree** of a connected graph G is a spanning subgraph that is a tree.

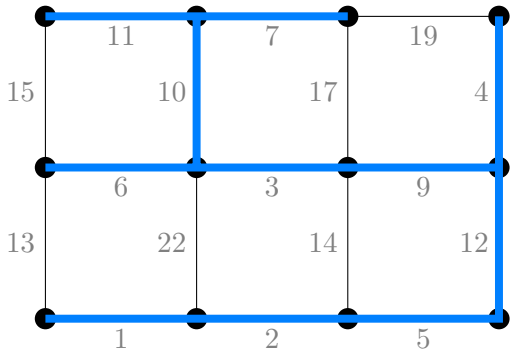
Theorem. Every connected graph has a spanning tree.

1: Prove the theorem.
Hint: Remove edges in cycles.

Solution: Proof by induction on the number of edges. Let G be a connected graph. If $|E(G)| = |V(G)| - 1$, then G is a tree. Notice that this is the minimum number of edges a connected graph on $|V(G)|$ vertices can have. It is the base case of the induction. Now assume $|E(G)| > |V(G)| - 1$. Hence G contains a cycle C. Pick any edge $e \in E(C)$. Notice that $G - e$ is still connected. By induction, it has a spanning tree T, which is also a spanning tree for G.

Problem: Connect cities V with optic cables. For every pair of cities, it is known if the cable can be built and the cost of building it $c : \binom{V}{2} \to \mathbb{R}$. Which cables should be built so that the cities are connected and the total building cost is minimized?
(The original application was with electrification.)

2: Solve the cities and cable problem for the following diagram of cities. Cost is number next to each edge. If an edge is missing, it is not possible to build the cable (think the cost is astronomical).



Formal definition of our problem: **Minimum spanning tree problem**
Input: Graph $G = (V, E)$ and costs $c : E \to \mathbb{R}$.
Output: Spanning tree T of minimum cost.

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

Theorem. Let G be a graph. If T is a spanning three of G, 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 $E(T) = \{e_1, \ldots, e_{n-1}\}$ 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 previous edge is in the cut X_i.

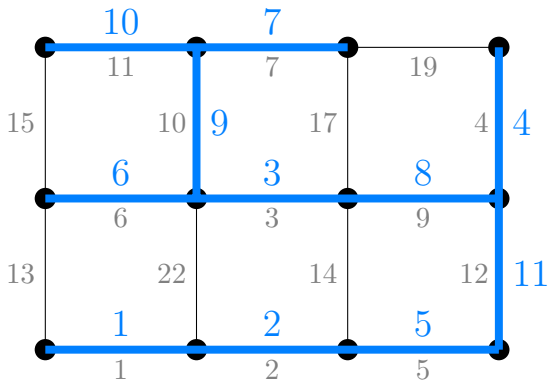
3: Show $1 \to 2 \to 3 \to 4$. Try $4 \to 1$ by taking T that satisfies 4 and $T^\star$ satisfying (1) and check how they can differ.

Solution: See the book(s) for detailed solution.
(1) $\to$ (2): If 2 violated, T was not optimal by replacing an edge
(2) $\to$ (3): if 3 violated, so is (2)
(3) $\to$ (4): take any ordering from (3), it gives (4). (4) $\to$ (1): T from (4) and $T^\star$ optimum. Let e_i be the first $e_i \in T$, that is missing in $T^\star$. Let the corresponding cut for e_i be X_i. Add e_i to $T^\star$, it contains a circuit, one other edge of the cut X_i that is in $T^\star$ can be removed and cost of $T^\star$ decreases.

Kruskal's (greedy) algorithm [1956]

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

4: 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.



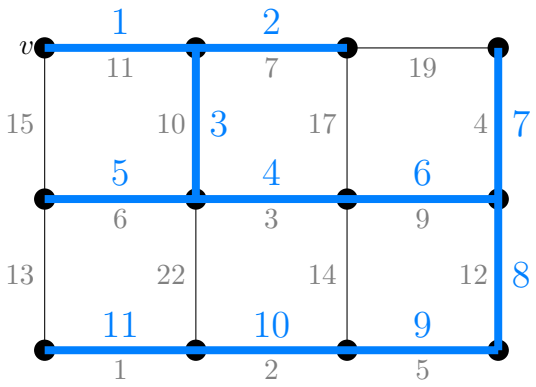
5: Why is the output of the algorithm correct?

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 $T := T + e$

6: 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.



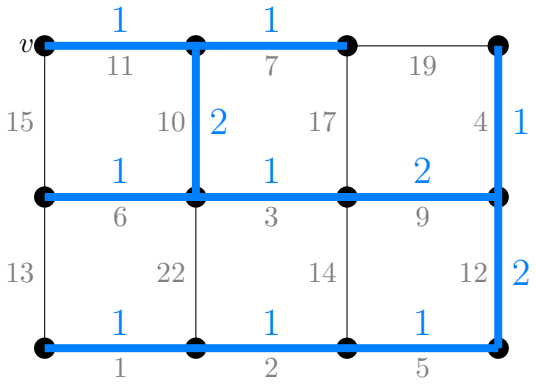
7: Why is the output of the algorithm correct?

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 $T := T + e$

8: 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 three.



9: Why is the output of the algorithm correct?

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

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 and count how many times the algorithm touches each edge or vertex.

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 exists if weights are integers, graph is *planar*, ...