Spanning trees, II

Lecture 11

Minimum spanning trees

In many applications, it makes sense to consider an **edge-weighted graph**, which is a graph $G = (V(G), (E))$ along with a weight function $w : E(G) \to \mathbb{R}$ that associates a real number (the weight) to each edge.

An application might be if you have multiple villages you want to connect with roads, the villages are all vertices, while the edges can be weighted with the cost to build a road between those two villages. You might want to minimize the cost of road construction.

Similarly, you may have a set of computers that you want to connect into a network, and the cost of connecting computer i with computer j is $c_{i,j}$. Again you may want to economize.

In both examples, we are looking for a spanning connected subgraph of our graph with the sum of the weights of the edges as small as possible.

Of course, if we have edges with negative weights, we'd better include all of them. If the resulting graph is connected, then we are done. If not, we can shrink each component into a vertex and consider the resulting graph with modified weights.

In both examples, we are looking for a spanning connected subgraph of our graph with the sum of the weights of the edges as small as possible.

Of course, if we have edges with negative weights, we'd better include all of them. If the resulting graph is connected, then we are done. If not, we can shrink each component into a vertex and consider the resulting graph with modified weights.

In both examples, we are looking for a spanning connected subgraph of our graph with the sum of the weights of the edges as small as possible.

Of course, if we have edges with negative weights, we'd better include all of them. If the resulting graph is connected, then we are done. If not, we can shrink each component into a vertex and consider the resulting graph with modified weights.

In view of this observation, we will assume all edge weights are non-negative. In this case, among spanning subgraphs of minimum total weight there always are spanning trees.

This motivates us to study the Minimum Spanning Tree Problem in a graph. As we know, K_n has n^{n-2} distinct spanning trees, so the idea to look at all such trees and choose among them a tree of minimum weight is not a great idea.

A lemma

Lemma 2.7 : Let G be a connected loopless graph with weighted edges, where $w(e) \geq 0$ for every $e \in E(G)$.
Let $T_1, \ldots, T_k$ be vertex-disjoint trees contained in G such that $V(T_1) \cup \ldots \cup V(T_k) = V(G)$.
Let e_0 be an edge of the minimum weight among the edges of G connecting $V(T_1)$ with $V(G) - V(T_1)$.
Then among the containing $E(T_1) \cup \ldots \cup E(T_k)$ spanning trees of G of minimum weight, there is a tree containing e_0.

A lemma

Lemma 2.7 : Let G be a connected loopless graph with weighted edges, where $w(e) \geq 0$ for every $e \in E(G)$.
Let $T_1, \ldots, T_k$ be vertex-disjoint trees contained in G such that $V(T_1) \cup \ldots \cup V(T_k) = V(G)$.
Let e_0 be an edge of the minimum weight among the edges of G connecting $V(T_1)$ with $V(G) - V(T_1)$.
Then among the containing $E(T_1) \cup \ldots \cup E(T_k)$ spanning trees of G of minimum weight, there is a tree containing e_0.

Proof. Let $n = V(G)$. Let T_0 be a spanning tree of G containing $E(T_1) \cup \ldots \cup E(T_k)$ of minimum weight.

Suppose $e_0 = xy$ where $x \in V(T_1)$ and $y \in V(G) - V(T_1)$.
If $e_0 \in E(T_0)$, then we are done.

Otherwise, $T' = T_0 + e_0$ is a connected graph with n edges containing exactly one cycle, say C. By construction, $e_0 \in E(C)$.
Since $x \in V(T_1)$ and $y \in V(G) - V(T_1)$, cycle C contains another edge e_1 connecting $V(T_1)$ with $V(G) - V(T_1)$.
Then $T'' := T' - e_1$ is a connected graph with $n - 1$ edges; hence a spanning tree of G. Moreover, by the choice of e_0, $w(e_0) \leq w(e_1)$.

Otherwise, $T' = T_0 + e_0$ is a connected graph with n edges containing exactly one cycle, say C. By construction, $e_0 \in E(C)$.
Since $x \in V(T_1)$ and $y \in V(G) - V(T_1)$, cycle C contains another edge e_1 connecting $V(T_1)$ with $V(G) - V(T_1)$.
Then $T'' := T' - e_1$ is a connected graph with $n - 1$ edges; hence a spanning tree of G. Moreover, by the choice of e_0, $w(e_0) \leq w(e_1)$.

Therefore, $\sum_{e \in E(T'')} w(e) \leq \sum_{e \in E(T_0)} w(e)$. It follows that T'' also is a spanning tree of G containing $E(T_1) \cup \ldots \cup E(T_k)$ of minimum weight. □

Prim's Algorithm:

Input: A weighted connected n-vertex graph G, say, $V(G) = \{v_1, \ldots, v_n\}$.

Goal: A spanning tree with the minimum total weight of the edges.

Initialization: Let $V_0 := \{v_1\}$ and $E(T) := \emptyset$.

Step i ($i = 1, \ldots, n-1$): Let e_i be an edge of minimum weight among the edges connecting V_0 with $V(G) - V_0$. If $e_i = xy$, where $x \in V_0$ and $y \in V(G) - V_0$, then let $V_0 := V_0 \cup \{y\}$ and $E(T) := E(T) \cup \{e_i\}$.

Proof: By Lemma 2.7.

Kruskal's Algorithm:

Input: A weighted connected n-vertex graph G, say, $E(G) = \{e_1, \ldots, e_m\}$.

Goal: A spanning tree with the minimum total weight of the edges.

Initialization: Reorder the edges so that $w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m)$. Let $E(T) := \emptyset$.

Step j ($j = 1, \ldots, m$): If $E(T) \cup \{e_j\}$ does not contain cycles, then let $E(T) = E(T) \cup \{e_j\}$. Otherwise, do nothing.

Proof: By Lemma 2.7.

What if we want to find a spanning tree **of maximum total weight**?

Main theorems in Chapter 2:

1. A Characterization Theorem for trees (Theorem 2.2).

Main theorems in Chapter 2:

1. A Characterization Theorem for trees (Theorem 2.2).
2. Jordan's Theorem on centers of trees (Theorem 2.3).

Main theorems in Chapter 2:

1. A Characterization Theorem for trees (Theorem 2.2).
2. Jordan's Theorem on centers of trees (Theorem 2.3).
3. Theorem on Prüfer codes, Cayley's Formula.

Main theorems in Chapter 2:

1. A Characterization Theorem for trees (Theorem 2.2).
2. Jordan's Theorem on centers of trees (Theorem 2.3).
3. Theorem on Prüfer codes, Cayley's Formula.
4. Matrix Tree Theorem (Theorem 2.6).

Main theorems in Chapter 2:

1. A Characterization Theorem for trees (Theorem 2.2).
2. Jordan's Theorem on centers of trees (Theorem 2.3).
3. Theorem on Prüfer codes, Cayley's Formula.
4. Matrix Tree Theorem (Theorem 2.6).
5. Prim's and Kruskal's algorithms.