Math 416

Worksheet 16. Min-Cost Spanning Trees

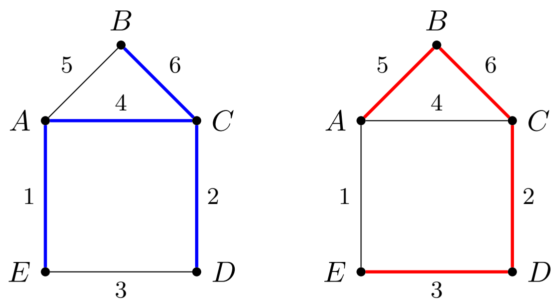
Min-cost spanning trees Suppose we want to network n computers and that connecting the i^{th} computer with the j^{th} has cost c_{ij}. We would like to connect all the computers as cheaply as possible.

Input: A weighted graph $G = (V, E, \text{wt})$ with edge weights $\text{wt}(e)$, which we will assume is *connected*.
Output: A tree $T = (V, E')$, with $E' \subseteq E$, a spanning tree of G for which the total cost:

$$\text{cost}(T) = \sum_{e \in E'} \text{wt}(e)$$

is minimal among all spanning trees of G. Such a T is called a *minimum-cost spanning tree* (MCST).

Problem 0. Find the costs of the blue and red (thickly drawn) spanning trees below. Is either one a MCST?



Idea #1: Start with *all* vertices, adding edges one at a time in increasing order of weight, as long as no cycle is introduced.

This works! It is the idea behind Kruskal's algorithm.

Idea #2: Grow the tree, adding at each stage a 'frontier' edge that minimizes an 'attachment cost,' as in Dijkstra's algorithm.

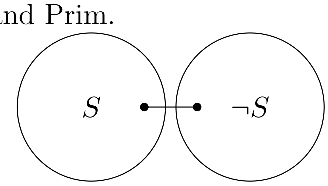
This idea works too! It is the idea behind Prim's algorithm.

Each maintains the following loop invariant.

For the tree T built so far, T is a subset of some min-cost spanning tree.

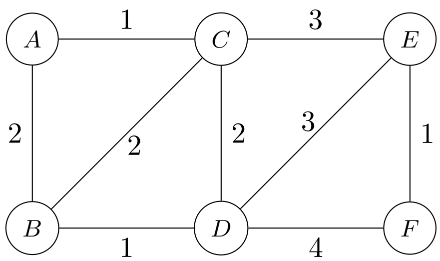
The Cut Property This is the key to the correctness of both Kruskal and Prim.

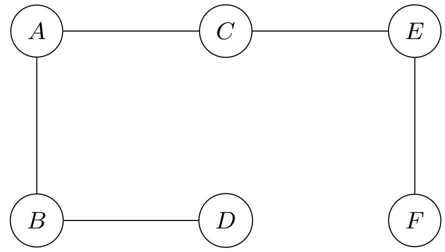
Definition. A **cut** in a graph is a partition of the vertices into two pieces, say S and $\neg S$; an edge e **crosses** the cut $(S, \neg S)$ if it's incident to one vertex in S and another in $\neg S$.



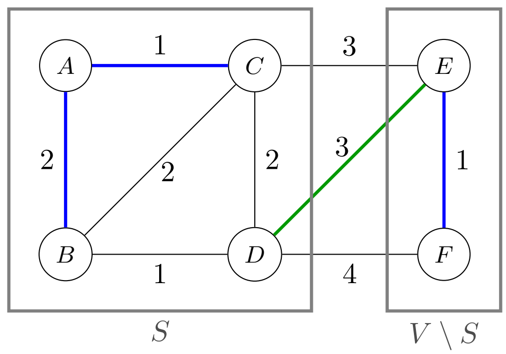
Theorem (Cut Property)**.** Let $A \subseteq E$ be included in some MCST. Let $(S, \neg S)$ be any cut of G such that A has no edges crossing $(S, \neg S)$. If e is an edge of minimal weight crossing $(S, \neg S)$, then $A \cup \{e\}$ is also a subset of a MCST.

Problem 1. On the left is a weighted graph. On the right is a MCST (you can check):

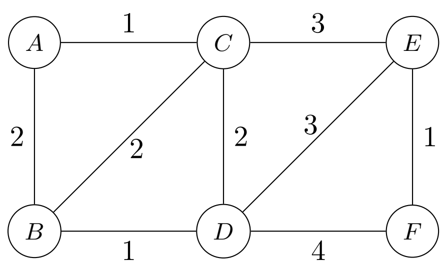




Consider the set of edges $\{AB, AC, EF\}$ and the cut $\{A, B, C, D\} \cup \{E, F\}$:



The edge DE is minimal among those crossing across the cut, so according to the Cut Property there should be a MCST including $\{AB, AC, EF, DE\}$. Find it:



Problem 2. Prove the Cut Property Theorem, as follows.

(a) The proof uses an exchange argument. Suppose that T is a MCST with $A \subseteq T$. We may assume $e \notin T$, since otherwise ...?

(b) So we have vertices $u \in S$ and $v \in \neg S$ for which $e = (u, v)$. (Draw a picture and) Explain why there is a path in T from u to v; so that this path has a first edge e' crossing the cut $(S, \neg S)$.

(c) Explain why $\mathrm{cost}(T - e' + e) \leq \mathrm{cost}(T)$.

(d) Why is the subgraph $T - e' + e$ connected?

(e) Conclude that $T - e' + e$ is a spanning tree. Conclude that it is a MCST.

(f) Verify that you have completed the proof of the Theorem.

Kruskal's algorithm. Here is a high-level description of Kruskal's algorithm:

(i) Order edges in nondecreasing order of weight:

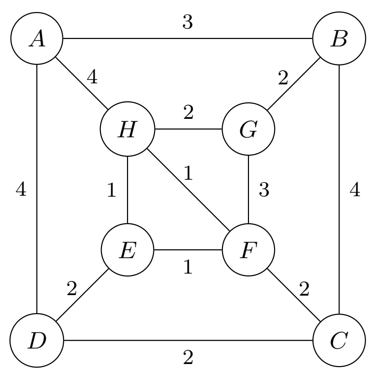
$$\mathrm{wt}(e_1) \leq \mathrm{wt}(e_2) \leq \cdots \leq \mathrm{wt}(e_m).$$

(ii) Starting with $E_0 = \emptyset$ build sets of edges $E_0 \subseteq E_1 \subseteq \cdots$ as follows.

$$E_i = \begin{cases} E_{i-1} \cup \{e_i\} & \text{if } (V, E_{i-1} \cup \{e_i\}) \text{ has no cycles} \\ E_{i-1} & \text{otherwise} \end{cases}$$

(iii) Stop when all edges have been considered (i.e., $i = m$) or $|E_i| = |V| - 1$. This final E_i is the set of edges of the output tree T.

Problem 3. Run Kruskal's algorithm, breaking ties with the alphabetical ordering:



Problem 4. Prove the correctness of Kruskal's algorithm, as follows.

(a) We will show by induction on $i \leq m$ that E_i is included in a MCST. If this induction is successful, why does the correctness of Kruskal follow?

(b) An edge e_i that we are adding to E_i connects two connected components of the graph (V, E_i). Why?

(c) Say that one of these connected components is S, so that e_i crosses the cut $(S, \neg S)$. Why is e_i a minimum-weight edge crossing the cut?

(d) Finish the proof.

Remark. Using a new data structure for unions of disjoint sets (which we won't worry about much), Kruskal's algorithm can be implemented to run in $O(m \log n)$ time. ($m = |E|$, $n = |V|$).

Algorithm 1: Prim's algorithm

Input: a weighted connected graph $G = (V, E, \mathrm{wt})$ (with nonnegative weights)
Output: a min-cost spanning tree $T = (V, E^*)$

let e be any edge of minimum weight ;
set $X = \{e\}$;
while $|X| < |V| - 1$ **do**
    let S be the set of vertices incident to at least one edge in X ;
    `// no edge in` X `crosses the cut` $(S, \neg S)$
    let e be an edge of minimum weight crossing the cut $(S, \neg S)$;
    set $X = X \cup \{e\}$;
return (V, X)

Prim's algorithm

Problem 5. Run Prim's algorithm on the graph from Problem 3, breaking ties with the alphabetical ordering:

Problem 6. Explain how the correctness of Prim's algorithm follows from the Cut Property.

Remark. Prim's algorithm can also be implemented to run in $O(m \log n)$ time. ($m = |E|$, $n = |V|$).