17 June

A couple of results from graph theory

Lemma [degree-sum formula, handshaking lemma]. Let $G=(V,E)$ be a graph and let $m=|E|$. Then

$$\sum_{v\in V} \deg(v) = 2m.$$

Proof. In the sum, each edge is counted twice, once for each of its endpoints. $\square$

Corollary. The average degree of a graph $G=(V,E)$ with $|V|=n$ and $|E|=m$ is $\frac{2m}{n}$.

Notation: $\delta(G)$ = minimum degree of G.
$\Delta(G)$ = maximum degree of G.

Corollary. $\delta(G) \leq \frac{2m}{n} \leq \Delta(G)$.

17 June

The minimum spanning tree problem [P&S §12.1]

Given: A simple, undirected graph $G=(V,E)$ with edge weights, i.e., a function $d: E \to \mathbb{R}$ that associates a real number with each edge. (Weights may represent distances or costs, for example).

Assumption: G is connected. (so that it has a spanning tree)

Goal: Find a minimum-weight spanning tree, that is, a spanning tree $H=(V,T)$ of G such that $\sum_{e \in T} d(e)$ is minimized.

Applications: Build a network joining a set of nodes (cities, computers, etc.) at minimum cost.

- G represents the set of possible links that can be built, and their costs.

- Minimum spanning tree H represents the cheapest connecting network, because trees are minimally connected (Exercise 6 on Problem set 5).

Theorem. [P&S Thm 12.1]
Let $G=(V,E)$ be a connected edge-weighted graph. Let $\{(U_1,T_1),(U_2,T_2),\ldots,(U_k,T_k)\}$ be a forest spanning V. [here (U_i,T_i) are the connected components of the forest, i.e., (U_i,T_i) is a tree]. Let $\{u,v\}$ be the shortest (i.e., minimum weight) of all edges with only one endpoint in U_1. Then among all spanning trees containing all edges in $T=\bigcup_{j=1}^{k} T_j$, there is an optimal (i.e., minimum weight) one containing $\{u,v\}$.

Proof. Suppose for the sake of contradiction that there is a spanning tree (V,F) with $F\supseteq T$ and $\{u,v\}\notin F$, which has (strictly) smaller weight than all spanning trees containing all of T and also $\{u,v\}$.

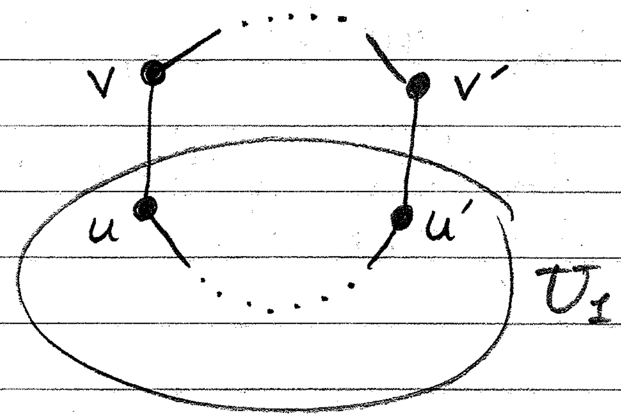
Add the edge $\{u,v\}$ to T. This creates a (unique) cycle. (The fact that a cycle is created is part of Exercise 6 on Pset 5; the fact that the cycle is unique is an additional exercise for the reader.)

By hypothesis, $\{u,v\}$ has exactly one endpoint in U_1; say $u\in U_1$ and $v\notin U_1$. The cycle includes v, so it does not consist entirely of nodes in U_1. Therefore, if we walk around the cycle we must leave U_1 at some point and then enter U_1 again at a different point.

Minimum spanning tree — 2

17 June

So the cycle must include an edge $\{u', v'\}$, different from $\{u, v\}$, with $u' \in U_1$ and $v' \notin U_1$:



Also $\{u', v'\}$ is not in T because it is not an edge in any of the (U_i, T_i)'s.

By hypothesis, the weight of $\{u,v\}$ is no greater than the weight of $\{u',v'\}$, so if we remove $\{u',v'\}$ we get a spanning tree (why?) with weight no greater than that of (V,F). But this is a contradiction, because we assumed that (V,F) has weight strictly smaller than all spanning trees containing T and $\{u,v\}$.

$\square$

We can use this theorem almost directly to get an efficient algorithm to solve the minimum spanning tree problem for $G=(V,E)$:

— The theorem applies to any spanning forest, so in particular it applies to the spanning forest $(V, \emptyset)$ [all vertices, no edges].

- Choose any vertex and call it v_1.
- Apply the theorem using $U_1 = \{v_1\}$:
  - Find the minimum-weight edge $\{v_1, v_2\}$ incident upon v_1.
  - By the theorem, there is a minimum spanning tree that includes this edge.
  - So include this edge in the tree we are building.
- Now apply the theorem using $U_1 = \{v_1, v_2\}$:
  - Find the minimum-weight edge $\{v_i, v_3\}$ (for $i \in \{1, 2\}$) having exactly one endpoint in U_1.
  - By the theorem, there is a minimum spanning tree that includes the edge $\{v_1, v_2\}$ from before <u>and</u> the new edge $\{v_i, v_3\}$.
  - So include $\{v_i, v_3\}$ in the tree we are building.
- Now apply the theorem using $U_1 = \{v_1, v_2, v_3\}$...

17 June

[P&S Figure 12-2, §12.1]

Prim's algorithm (for minimum spanning tree)

Input: A graph $G=(V,E)$ and edge weights $d(e)$ for $e\in E$.

1. Initialize: $U := \{v_1\}$, $T := \emptyset$.

   arbitrary vertex

   set of vertices included in the tree so far

   set of edges included in the tree so far

2. If $U = V$, we are done. Stop.

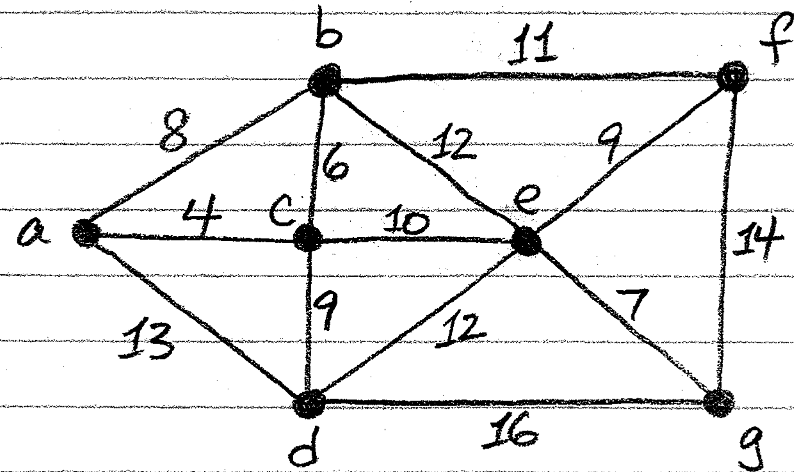
3. Out of all edges having exactly one endpoint in U, choose one with minimum weight, say $\{v, w\}$ where $v \in U$ and $w \notin U$.

4. Add w to U and add $\{v,w\}$ to T.

5. Go back to step 2.

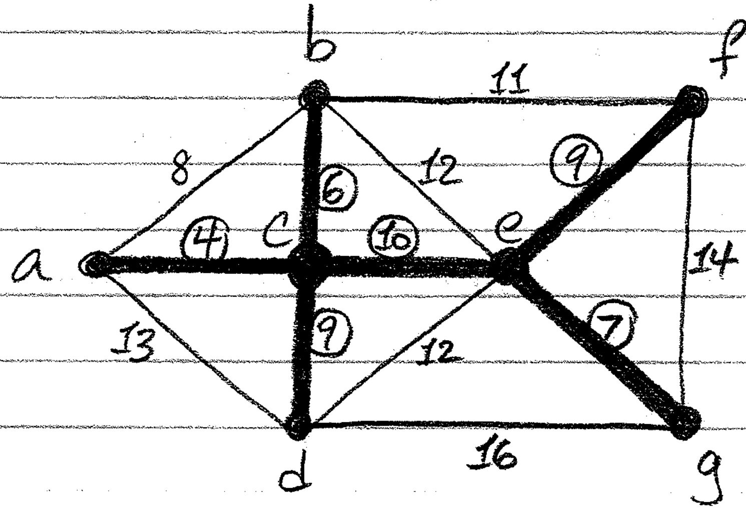
The algorithm presented in P&S is optimized a bit to speed up the search in step 3 by keeping track, for each vertex w not in U, of the closest vertex in U to w, and updating this information after each new vertex is added to U.

Example. Use Prim's algorithm to find a minimum spanning tree in the following graph.



Iteration	U	T	Min-weight edge with exactly 1 endpt in U
1	$\{a\}$	$\emptyset$	ac
2	$\{a,c\}$	$\{ac\}$	bc
3	$\{a,b,c\}$	$\{ac, bc\}$	cd
4	$\{a,b,c,d\}$	$\{ac, bc, cd\}$	ce
5	$\{a,b,c,d,e\}$	$\{ac, bc, cd, ce\}$	eg
6	$\{a,b,c,d,e,g\}$	$\{ac, bc, cd, ce, eg\}$	ef
7	$\{a,b,c,d,e,f,g\}$	$\{ac, bc, cd, ce, eg, ef\}$	DONE!

Minimum-weight spanning tree:



Total weight 45.

17 June

Another algorithm for minimum spanning tree.

- In Prim's algorithm, we always apply the theorem to the "same" U_1 (namely, the connected component of the spanning forest that contains v_1). But we don't have to do that.

- Instead, what if we find a minimum-weight edge anywhere in the graph that joins two vertices in different connected components of the spanning forest? Then we can choose (the vertex set of) either one of those connected components, call it U_1, and apply the theorem to see that there is a minimum spanning tree that includes that edge (in addition to the edges we've chosen so far using this same method).

Kruskal's algorithm (for minimum spanning tree)

Input: A graph $G=(V,E)$ and edge weights $d(e)$ for $e \in E$.
Let $V = \{v_1, v_2, v_3, \ldots, v_n\}$.

1. Initialize: $S_i := \{v_i\}$ for $1 \le i \le n$
(these are vertex sets of connected components of spanning forest)

$$C := \{S_1, S_2, S_3, \ldots, S_n\}$$
(set of connected components)

$$T := \emptyset$$ (set of edges included so far in spanning forest)

2. If $|C| = 1$, we are done. Stop.
(a single connected component)

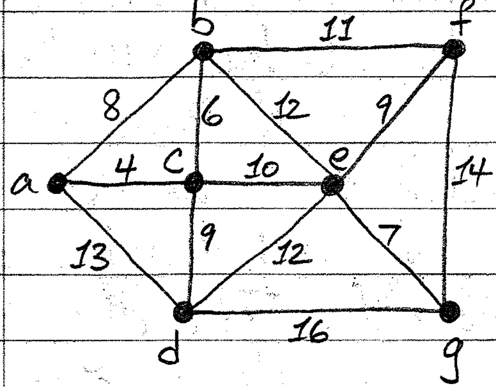
3. Out of all edges $\{v,w\}$ such that $v \in S_i$ and $w \in S_j$ with $i \neq j$, choose one with minimum weight.

4. Remove S_i and S_j from C.
Add $S_i \cup S_j$ to C.
Add $\{v,w\}$ to T.

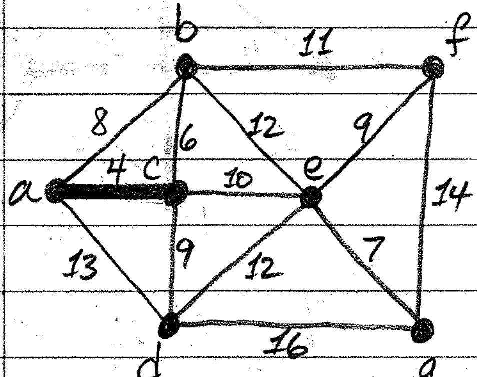
5. Go back to step 2.

17 June

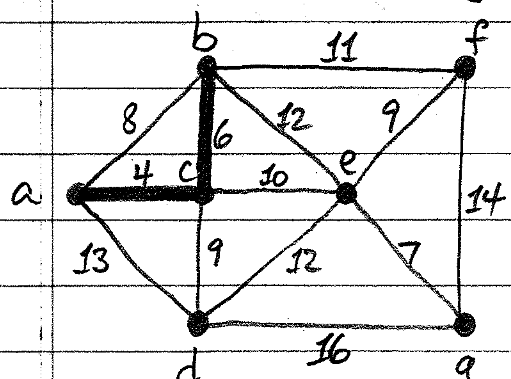
Kruskal's algorithm — Example.



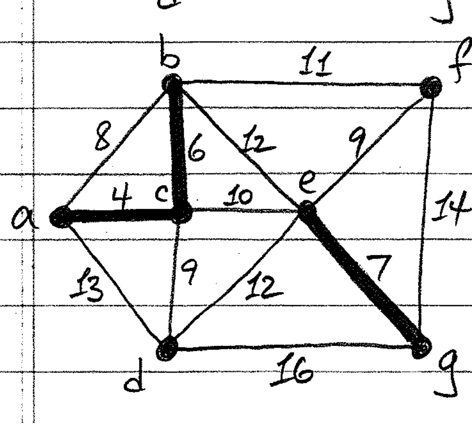
Initially each vertex is in its own connected component of the spanning forest. So the minimum-weight edge joining two vertices in different components is {a,c}. Add {a,c} to the spanning forest; now a and c are in the same connected component.

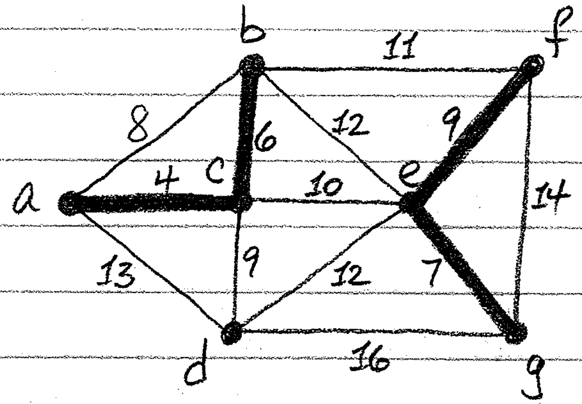


Repeat. The min-weight edge joining vertices in different components is now {b,c}. Add that edge to the spanning forest, so now a, b, and c are all in the same connected component.

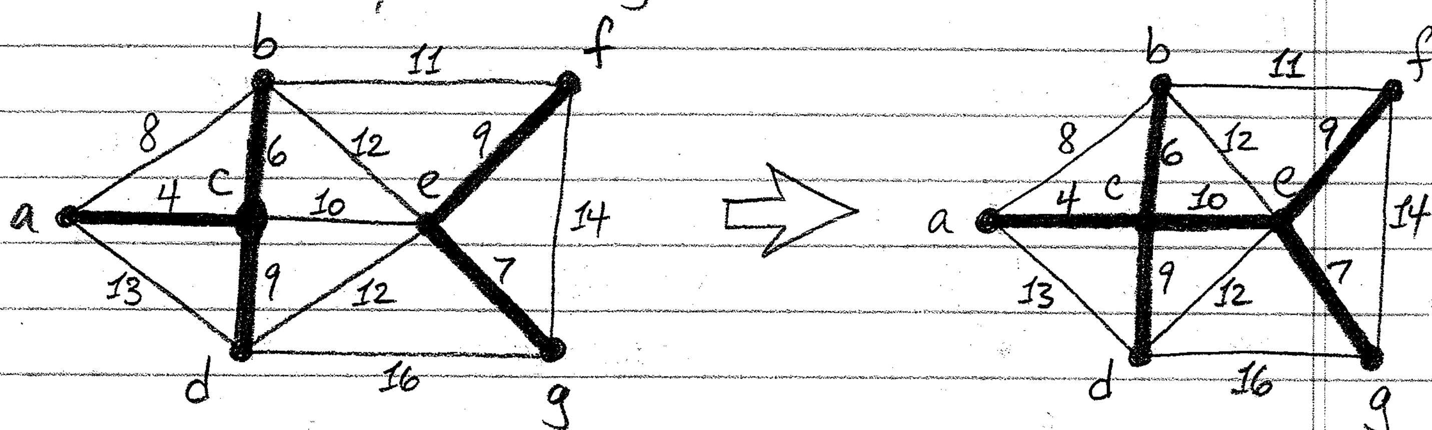


Repeat. The min-weight edge joining vertices in different components is now {e,g}. Note that we have diverged from Prim's algorithm here, because neither e nor g is in the same connected component as the rest of the edges we've added.





Now we have a tie for the min-weight edge joining vertices in different components: cd and ef. (The edge ab has smaller weight, but it joins vertices in the same component, so adding it would create a cycle.) We choose ef arbitrarily and add it to the spanning forest. The algorithm continues by adding cd and then ce:



Note that the end result is the same as the spanning tree we got with Prim's algorithm.