1. In class we stated the following result: Suppose $G$ is a connected, undirected graph with $n$ vertices. If the number of edges in $G$ is $n - 1$, then $G$ is a free tree.

First note that the result can be proved by drawing the different cases if $G$ has 1 or 2 vertices. So we may as well assume that $n \geq 3$.

Since a free tree is a connected graph with no simple cycles, one way to prove this is to look at the contrapositive: Suppose $G$ is a connected, undirected graph with $n \geq 3$ vertices. If $G$ contains a simple cycle, then the number of edges in $G$ is at least $n$.

In order to prove this, we take a connected graph $G$ containing a simple cycle $C$ on $k$ vertices. As noted in the text, the smallest simple cycle has 3 vertices. So $k \geq 3$.

Now we can use induction on the number $m$ of vertices in the complement of $C$, and $k$, the number of vertices in $C$, is fixed.

(a) Prove the base case. That is, show that if $G$ is a connected graph with $k \geq 3$ vertices and $G$ contains a simple cycle on all $k$ vertices, then $G$ contains at least $k$ edges. Hint: Count the edges in a simple cycle.

Suppose the simple cycle is $(v_0, v_1, \ldots, v_{k-1}, v_0)$. Then the edges in the cycle are $e_1 = \{v_0, v_1\}, e_2 = \{v_1, v_2\}, \ldots, e_{k-1} = \{v_{k-2}, v_{k-1}\}, e_k = \{v_{k-1}, v_0\}$. Since the cycle is simple, $e_i \neq e_j$ whenever $i \neq j$. Thus, there are at least $n = k$ distinct edges in $G$.

(b) State the induction hypothesis.

Let $G$ be a connected, undirected graph subject to the following hypotheses.

i. $G$ contains $n = m + k$ vertices, where $m = m_0 \geq 0$ and $k \geq 3$.

ii. $G$ contains a simple cycle $C$ on $k$ vertices.

Then $G$ has at least $n = m + k$ edges.

(c) What do you need to show in the induction step? What are you assuming? What do you need to prove?

Suppose $G$ is a connected, undirected graph subject to the following hypotheses.

i. $G$ contains $n = m_0 + 1 + k$ vertices, where $m_0 \geq 0$ and $k \geq 3$.

ii. $G$ contains a simple cycle $C$ on $k$ vertices.

Then we want to show that $G$ contains at least $n = m_0 + 1 + k$ edges.

(d) Extra Credit. Complete the induction step. Hint: in going from the case $m = m_0 + 1$ vertices in the complement of $C$ to the case $m = m_0$ vertices in the complement of $C$, use a DFS of $G$. The DFS should visit the vertices in $C$ first. Then the vertex that needs to be deleted can be chosen to be a leaf in the depth-first spanning tree.
The idea is similar to one we’ve used many times before: delete a vertex and its incident edges; use the IH to conclude that the result holds for the new, smaller graph; add back the vertex and the incident edges to get the result for the original graph.

The tricky part is that we might disconnect the graph if we delete just any vertex in the complement of $C$. So we use DFS to make sure that we delete a vertex that won’t disconnect.

Carry out a DFS of $G$, starting at a vertex of $C$, and always visiting vertices of $C$ before other vertices, and visiting the vertices of $C$ in order: first $v_0$, then $v_1$, etc. Since $G$ is connected, this will create a spanning tree (as opposed to a forest). The vertices of $C$ will lie on the path consisting of the first $k$ left branches of the tree.

Since there is a vertex not in $C$, there must be a vertex in the spanning tree that’s not in $C$. Since $G$ is connected, there’s a tree edge in the spanning tree joining a vertex $w$ not in $C$ to a vertex of $C$. Consider the subtree rooted at $w$. By the usual argument this subtree has a leaf $u$.

Now consider the graph $G'$ obtained from $G$ by removing $u$ and all its incident edges. Then $C$ is a subgraph of $G'$ and $G'$ is connected: the depth-first spanning forest for $G'$ is simply the spanning tree of $G$ with $u$ and its incident edge deleted.

So by the IH, $G'$ contains at least $m_0 + k$ edges. When we return $u$ and its incident edges to $G'$, recovering $G$, we add at least one edge, the tree edge joining $u$ to the spanning tree. Thus, $G$ contains at least $n = m_0 + 1 + k$ edges.

2. Problem 7.13, page 221. The single-destination shortest paths problem for a directed graph is to find the shortest path from every vertex to a given vertex $v_0$. Write an algorithm to solve the single-destination shortest paths problems.

Basically Dijkstra’s algorithm will do the job, with the caveat that in loops having the form

\[
\text{for each vertex } w \text{ adjacent to } v \text{ do}
\]

we’re now looking for edges from $w$ to $v$. So the algorithm should either traverse the column of the adjacency matrix headed by $v$, or it should use an adjacency list which shows edges into the vertex at the head of the list.

3. Problem 7.17, page 222. List the order in which the edges of the graph in Figure 7.25 are visited when running Prim’s MST algorithm starting at vertex 3. Show the final MST.

One spanning tree is shown in Figure 1. Tree edges are solid lines. Other edges are dashed. To obtain this tree, the edges are added in the following order: \{3, 2\}, \{2, 4\}, \{4, 6\}, \{6, 1\}, \{6, 5\}. This spanning tree is not unique. For example, a different spanning tree could be obtained by adding edges in the following order: \{3, 2\}, \{2, 4\}, \{2, 1\}, \{1, 6\}, \{6, 5\}.

4. Problem 7.23, page 222. Consider the collection of edges selected by Dijkstra’s algorithm as the shortest paths to the graph’s vertices from the start vertex. Do these edges form a spanning tree (not necessarily of minimum cost)? Do they form an MST? Explain why or why not.

First note that the graph $G$ must be connected in order for it to even have a spanning tree. So we assume the graph is connected.
Expanding on the text’s notation, make the following definitions. Suppose the vertices are marked in the order $v_0, v_1, \ldots, v_{n-1}$. Also suppose that when $v_i$ is marked, its predecessor is $p[v_i]$. Then the edges described by the text are

\[
e_1 = \{p[v_1] = v_0, v_1\}
\]

\[
e_2 = \{p[v_2], v_2\}
\]

\[
\vdots
\]

\[
e_{n-1} = \{p[v_{n-1}], v_{n-1}\}
\]

The pair $S = (\{v_0, \ldots, v_{n-1}\}, \{e_1, \ldots, e_{n-1}\})$ is a subgraph. To see this, note that each vertex is marked exactly once. So $\{v_0, \ldots, v_{n-1}\} = V(G)$, and the endpoints of every edge of $S$ belong to the set of vertices.

Furthermore $S$ is also a spanning tree. That it spans is immediate, since it contains all the vertices of $G$. It’s also connected. For if $v$ and $w$ are vertices of $S$, since $G$ is connected there’s a path from $v_0$ to $v$ and a path from $v_0$ to $w$. Reversing the path from $v_0$ to $v$ and appending to it the path from $v_0$ to $w$ gives a path from $v$ to $w$. That $S$ is a tree follows from the fact that it contains exactly $n - 1$ edges. (All the edges are distinct, since their “right” endpoints, $v_1, v_2, \ldots, v_{n-1}$ are distinct.)

If one considers how Dijkstra’s algorithm chooses vertices to mark and updates predecessors, it’s also clear that $S$ can’t contain a simple cycle.

The two homework problems, 7.7 and 7.17, show that $S$ is not in general an MST. The spanning tree in 7.7 has total cost 31, and the MST in 7.17 has total cost 23. The issue is that the shortest path to a fixed vertex $v_0$ will not, in general, coincide with a path in an MST. In our example, we see that Dijkstra’s algorithm chooses the edge of weight 11 as the shortest path from 4 to 5. However, the path in Prim’s MST goes from 4 to 6 to 5, which has total cost 13.