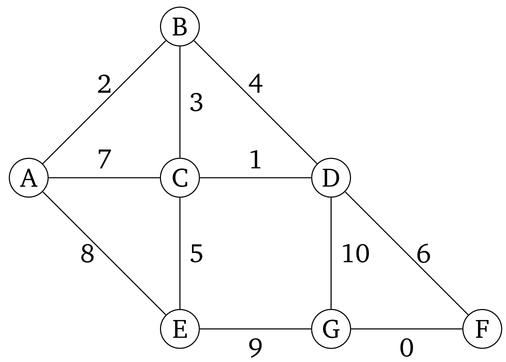
Section 09: MSTs, Disjoint Sets, and Graph Modeling

1. MSTs: Unique Minimum Spanning Trees

Consider the following graph:



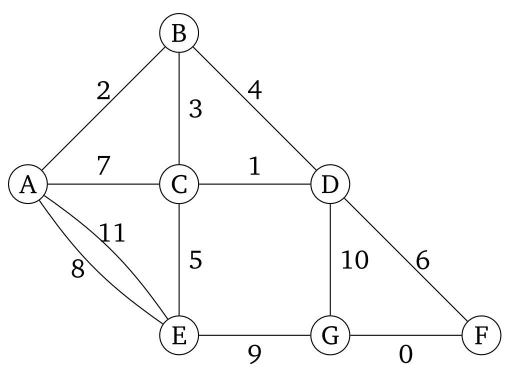
(a) What happens if we run Prim's algorithm starting on node A? What are the final costs and edges selected? Give the set of edges in the resulting MST.

(b) What happens if we run Prim's algorithm starting on node E? What are the final cost and edges selected? Give the set of edges in the resulting MST.

(c) What happens if we run Prim's algorithm starting on *any* node? What are the final costs and edges selected? Give the set of edges in the resulting MST.

(d) What happens if we run Kruskal's algorithm? Give the set of edges in the resulting MST.

(e) Suppose we modify the graph above and add a heavier parallel edge between A and E, which would result in the graph shown below. Would your answers for above subparts (a, b, c, and d) be the same for this following graph as well?



2. MSTs: True or False

Answer each of these true/false questions about minimum spanning trees.

(a) A MST contains a cycle.

(b) If we remove an edge from a MST, the resulting subgraph is still a MST.

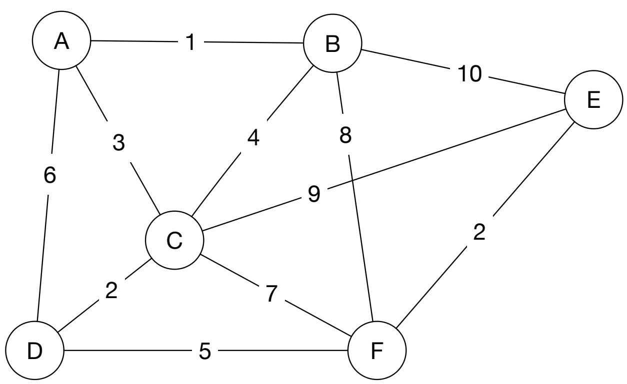
(c) If we add an edge to a MST, the resulting subgraph is still a MST.

(d) If there are V vertices in a given graph, a MST of that graph contains $|V| - 1$ edges.

3. MSTs: Kruskal's Algorithm

Answer these questions about Kruskal's algorithm.

(a) Execute Kruskal's algorithm on the following graph. Fill the table.

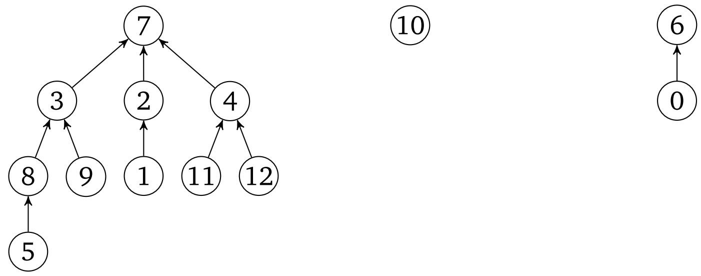


Step	Components	Edge	Include?
1	{A} {B} {C} {D} {E} {F}	A, B	Yes
2	{A, B} {C} {D} {E} {F}	B, C	
3			
4			
5			
6			
7			
8			
9			
10			
11			

(b) In this graph there are 6 vertices and 11 edges, and the for loop in the code for Kruskal's runs 11 times, a few more times after the MST is found. How would you optimize the pseudocode so the for loop terminates early, as soon as a valid MST is found.

4. Disjoint Sets

(a) Consider the following trees, which are a part of a disjoint set:

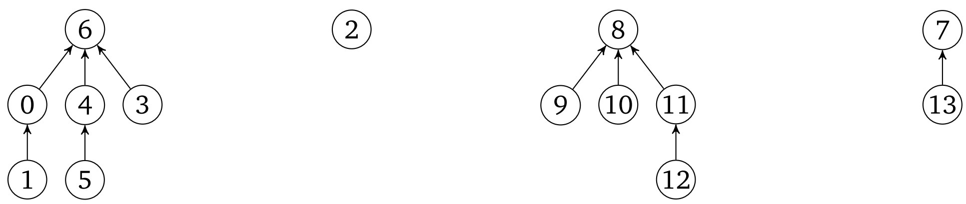


For these problems, use both the **union-by-rank** and **path compression** optimizations. Assume that the first tree has rank 3, the second has rank 0 and the last has rank 1.

(i) Draw the resulting tree(s) after calling `findSet(5)` on the above. What value does the method return?

(ii) Draw the final result of calling `union(2,6)` on the result of part (i).

(b) Consider the disjoint-set shown below:



The ranks of trees from left to right are 2, 0, 2, 1.

What would be the result of the following calls on union if we add the “union-by-rank” and “path compression optimizations. Draw the forest at each stage with corresponding ranks for each tree

(i) `union(2, 13)`

(ii) `union(4, 12)`

(iii) `union(2, 8)`

5. Graph Modeling 1: DJ Kistra

You’ve just landed your first big disk jockeying job as “DJ Kistra.”

During your show you’re playing “Shake It Off,” and decide you want to slow things down with “Wildest Dreams.” But you know that if you play two songs whose tempos differ by more than 10 beats per minute or if you play only a portion of a song, that the crowd will be very disappointed. Instead you’ll need to find a list of songs to play to gradually get you to “Wildest Dreams.” Your goal is to transition to “Wildest Dreams” as quickly as possible (in terms of seconds).

You have a list of all the songs you can play, their speeds in beats per minute, and the length of the songs in seconds.

(a) Describe a graph you could construct to help you solve the problem. At the very least you’ll want to mention what the vertices and edges are, and whether the edges are weighted or unweighted and directed or undirected.

(b) Describe an algorithm to construct your graph from the previous part. You may assume your songs are stored in whatever data structure makes this part easiest. Assume you have access to a method `makeEdge(v1, v2, w)` which creates an edge from `v1` to `v2` of weight `w`.

(c) Describe an algorithm you could run on the graph you just constructed to find the list of songs you can play to get to "Wildest Dreams" the fastest without disappointing the crowd.

(d) What is the running time of your plan to find the list of songs? You should include the time it would take to construct your graph and to find the list of songs. Give a simplified big-O running time in terms of whatever variables you need.

6. Graph Modeling 2: Snow Day

After 4 snow days this year, UW has decided to improve its snow response plan. Instead of doing "late start" days, they want an "extended passing period" plan. The goal is to clear enough sidewalks that everyone can get from every classroom to every other **eventually** but not necessarily very quickly.

Unfortunately, UW has access to only one snowplow. Your goal is to determine which sidewalks to plow and whether it can be done in time for Kasey's 8:30 AM lecture.

You have a map of campus, with each sidewalk labeled with the time it will take to plow to clear it.

(a) Describe a graph that would help you solve this problem. You will probably want to mention at least what the vertices and edges are, whether the edges are weighted or unweighted, and directed or undirected.

(b) What algorithm would you run on the graph to figure out which sidewalks to plow? Explain why the output of your algorithm will be able to produce a "extended passing period" plowing plan.

(c) How can you tell whether the plow can actually clear all the sidewalks in time?

7. Graph Modeling 3: Video Game

(a) Suppose we are trying to design a maze within a 2d top-down video-game. The world is represented as a grid, where each tile is either an impassable wall, an open space a player can pass through, or a *wormhole*. On each turn, the player may move one space on the grid to any adjacent open tile. If the player is standing on a wormhole, they can instead use their turn to teleport themselves to the other end of the wormhole, which is located somewhere else on the map.

Now, suppose the there are several coins scattered throughout the map. Your goal is to design an algorithm that finds a path between the player and some coin in the fewest number of turns possible.

Describe how you would represent this scenario as a graph (what are the vertices and edges? Is this a weighted or unwighted graph? Directed or undirected?). Then, describe how you would implement an algorithm to complete this task.

(b) **Challenge:** Suppose the map now stuffed with a huge number of players. We now want an algorithm that finds the shortest path between every single player and the closest coin.

A naive way of doing this would be to run the same algorithm from the previous part on every single player. However, this would be prohibitively expensive. Design a more efficient algorithm that does the same thing.

When designing your algorithm, you may assume that the map is relatively small/that there are only a few coins, relative to the number of players.