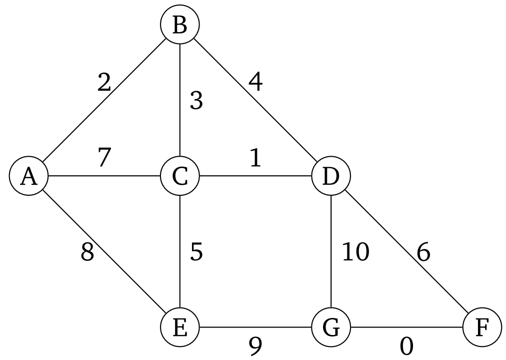
Section 07: MSTs + Disjoint Sets

Section Problems

1. MSTs: Unique Minimum Spanning Trees

Consider the following graph:



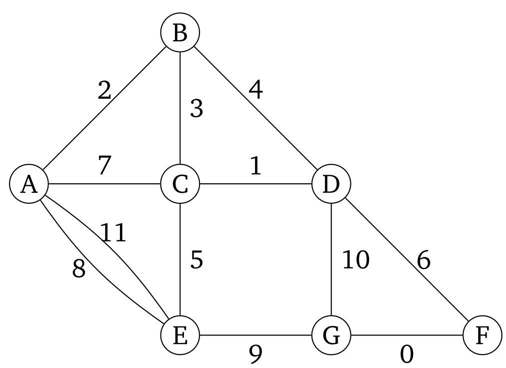
(a) What happens if we run Prim's algorithm starting on node A? What are the final costs and edges selected? Give the set of edges in the resulting MST.

(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.

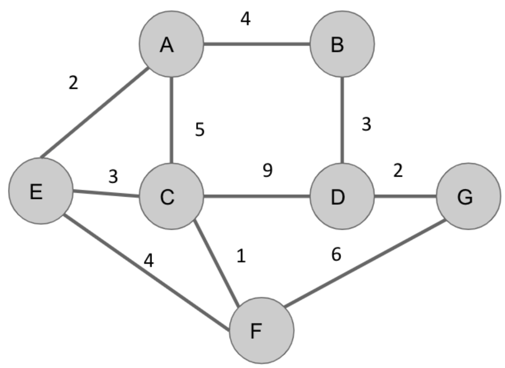
(d) What happens if we run Kruskal's algorithm? Give the set of edges in the resulting MST.

(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: Basic Kruskal's

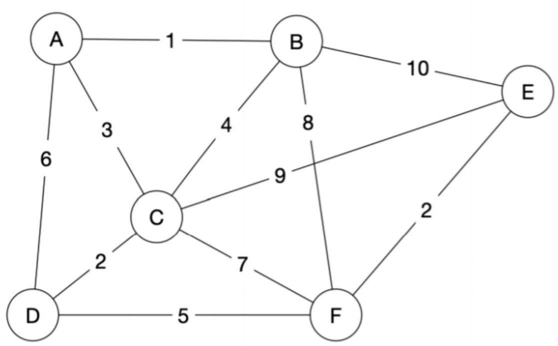
What happens if we run Kruskal's algorithm on this graph? Give the set of edges in the resulting MST.



3. MSTs: Kruskal's Algorithm

Answer these questions about Kruskal's algorithm.

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

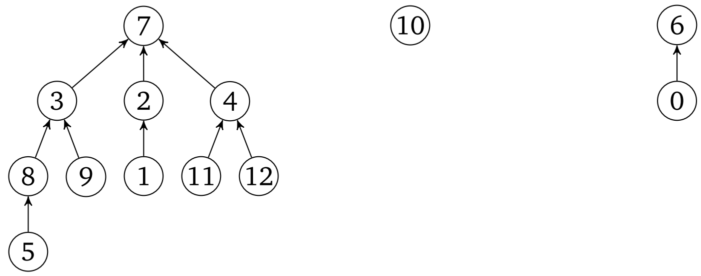


Step	Components	Edge	Include?
1	{A} {B} {C} {D} {E} {F}	A, B	Yes
2	{A, B} {C} {D} {E} {F}	D, C	
3			
4			
5			
6			
7			
8			
9			
10			
11			

(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: Union and Find

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

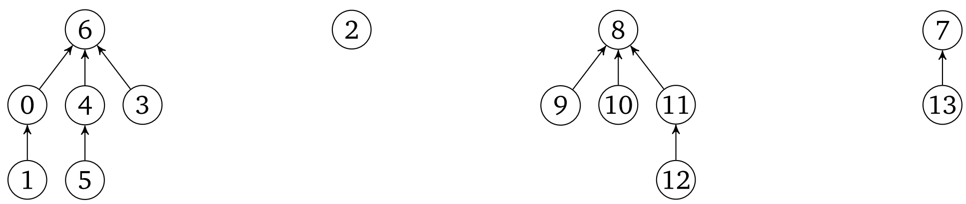


For these problems, use both the **union-by-size** and **path compression** optimizations.

(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:



What would be the result of the following calls on union if we add the **union-by-size** and **path compression** optimizations? Draw the forest at each stage.

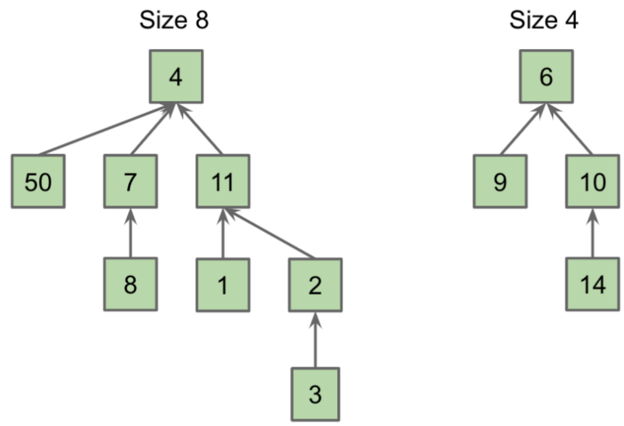
(i) `union(2, 13)`

(ii) `union(4, 12)`

(iii) `union(2, 8)`

5. Disjoint Sets: Array Representation

Fill in the correct array representation for the following disjoint sets structure given a mapping of items to their indices. Use the table below, and assume we are using the implementation from **lecture**.



Items:	50	4	7	8	9	11	1	2	3	6	10	14
Index:	0	1	2	3	4	5	6	7	8	9	10	11
Value:												

6. Disjoint Sets: Array Find Set

Using the disjoint sets **at the top of this page**, perform `findSet(2)` with path compression. Give the return value (i.e. the index of the representative), draw the updated array, and draw the resulting disjoint sets.

7. Disjoint Sets: Array Union

Using the disjoint sets **at the top of this page**, *instead* perform `union(3, 14)`. Use both the path compression and union-by-size optimizations. Draw the resulting array.

8. Graphs: 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.

9. Graph Modeling 1: Snowed In

After 4 snow days last 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 the first 8:30 AM lecture of the day.

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?

10. Graph Modeling 2: Snowden

Consider the following problems, which we can both model and solve as graph problems.

For each problem, describe (i) what your vertices and edges are and (ii) a short (2-3 sentence) description of how to solve the problem.

We will also include more detailed pseudocode in the solutions.

Your description does not need to explain how to implement any of the algorithms we discussed in lecture. However, if you *modify* any of the algorithms we discussed, you must discuss what that modification is.

(a) Suppose you have a bunch of computers networked together (haphazardly) using wires. You want to send a message to every other computer as fast as possible. Unfortunately, some wires are being monitored by some shadowy organization that wants to intercept your messages.

After doing some reconnaissance, you were able to assign each wire a "risk factor" indicating the likelihood that the wire is being monitored. For example, if a wire has a risk factor of zero, it is extremely unlikely to be monitored; if a wire has a risk factor of 10, it is more likely to be monitored. The smallest possible risk factor is 0; there is no largest possible risk factor.

Implement an algorithm that selects wires to send your message such that (a) every computer receives the message and (b) you minimize the total risk factor. The total risk factor is defined as the sum of the risks of all of the wires you use.

(b) Explain how you would implement an algorithm that finds any computers where sending a message (from a given start computer) would force you to transmit a message over a wire with a risk factor of k or higher.