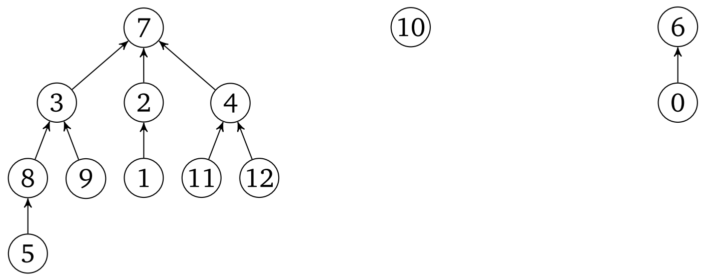
Section 07: Disjoint Sets + MSTs

1. Disjoint Sets: Union and Find

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

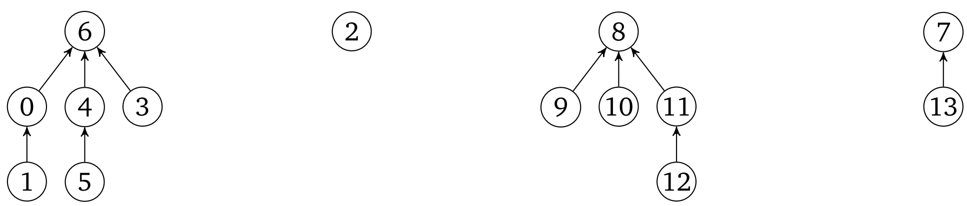


For these problems, use both the **union-by-size** and **path compression** optimizations.

(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)`

2. Graphs: True or False?

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

(a) A MST contains a cycle.

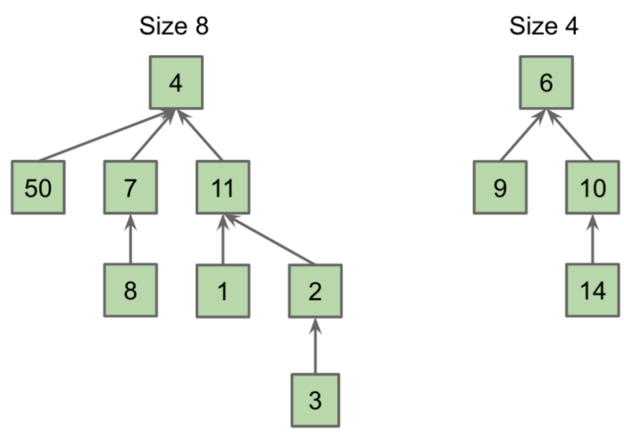
(b) If we remove an edge from a MST, the resulting subgraph is still a MST.

(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.

3. 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:												

4. Disjoint Sets: Array Find Set

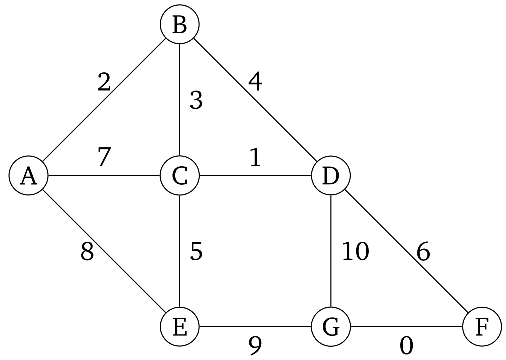
Using the disjoint sets **at the top of this page**, perform `findSet(2)` with path compression. Give the return value (i.e. the index of the representative), draw the updated array, and draw the resulting disjoint sets.

5. 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.

6. MSTs: Unique Minimum Spanning Trees

Consider the following graph:



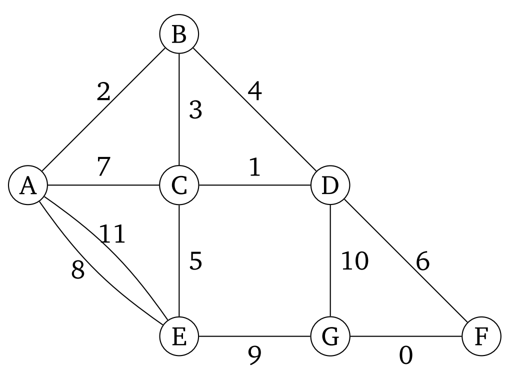
(a) What happens if we run Prim's algorithm starting on node A? What are the final costs and edges selected? Give the set of edges in the resulting MST.

(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.

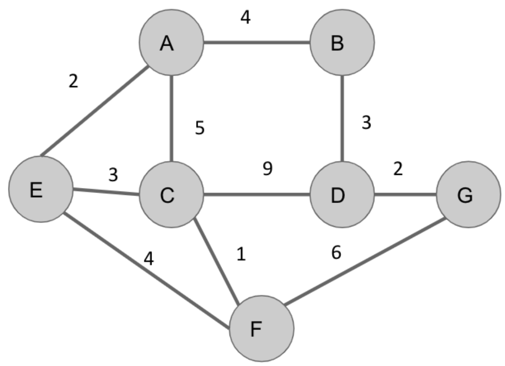
(d) What happens if we run Kruskal's algorithm? Give the set of edges in the resulting MST.

(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?



7. MSTs: Basic Kruskal's

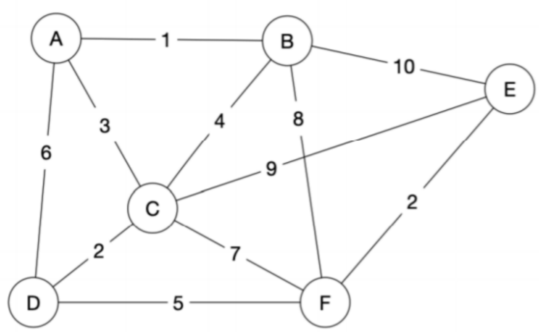
What happens if we run Kruskal's algorithm on this graph? Give the set of edges in the resulting MST.



8. 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.