CSE 373 Section Handout #7: Graphs & Union-Find

1. Given a graph with |V| vertices and |E| edges, what is the space requirement (in big-O) for representing the graph.

   As an adjacency list?

   As an adjacency matrix?

2. So far we've implemented Depth First Search (DFS) and Breadth First Search (BFS) using data structures to keep track of pending vertices (Stacks and Queues).

   Write pseudocode for DFS using recursion rather than a Stack.

3. Suppose you have social network data for some people (including yourself and a famous person). Write pseudocode to find the answers to the following questions:

    a. How would you represent this social network with a graph?

    b. Given two people, how would you determine if they were friends?

    c. Find the person with the most friends in the data?

    d. Find the length of the shortest path from yourself to the famous person.

    e. Find the number of people who do not know anyone you know (or anyone they know, etc).

4. Suppose you are given a graph G. Explain how you would figure out if it has a cycle.

    a. If the graph is undirected:

    b. If the graph is directed:

Disjoint Sets

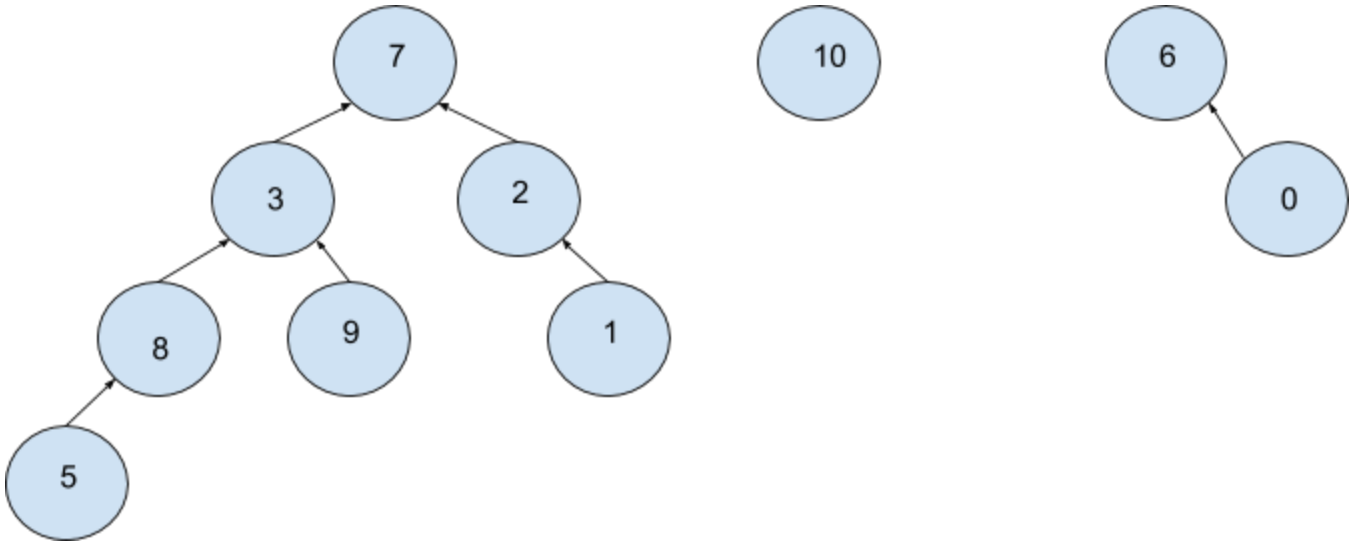
Reminder: A disjoint sets data structure keeps track of multiple sets which do not share any elements.

UnionFind ADT:

- **find(x)** Returns a number representing the set that x is in.
- **union(x, y)** Updates the sets so whatever sets x and y were in are now considered the same sets.

5. How can the Union-Find ADT be used to check whether an undirected graph contains cycle or not? Assume that the graph does not contain any self-loops.

6. Given the following trees from our implementation of disjoint sets seen in lecture, answer the following questions (assuming both weighted-unions and path compression):



   a. Draw the resulting tree(s) after calling find(5) on the above disjoint sets. Also indicate what value will be returned by this call.

   b. Draw the final result of calling union(2,6) on the result of part a.