CSE 332: Data Abstractions

Section 8: Graphs & Connectivity

In lecture, we solved the **CONN** problem. That is,

CONN	
Input(s):	Graph G
Output:	`true` iff G is connected

We used a `WorkList` algorithm:

```
1  isConnected(G) {
2     V, E = G
3     worklist = first(V);
4     seen = {v};
5     while (worklist.hasWork()) {
6        v = worklist.next();
7        for (w : v.neighbors()) {
8           if (w ∉ seen) {
9              worklist.add(w);
10             seen.add(w);
11          }
12       }
13    }
14    return seen == V;
15 }
```

This algorithm has several distinct names based on which type of WorkList we use. If we use a `FIFOQueue`, it's called Breadth-First Search (BFS), and if we use a `Stack`, it's called Depth-First Search (DFS).

0. Recursively, Now!

Although we've implemented it here and in lecture iteratively, we could implement **DFS** recursively as well. Write Psuedo-code for a DFS that uses recursion.

1. Two-Coloring

A graph G is two-colorable if and only if we can make a function $f : V \to \{\text{red}, \text{black}\}$ such that $\{u, v\} \in G \to f(u) \neq f(v)$. That is "adjacent vertices have different colors". Write an algorithm to solve the **2-COLOR** problem.

2. A Social Networking Event

Suppose you have social network data for some people (including yourself and a famous person). Explain how to use a graph algorithm to find the answers to the following questions:

(a) Find the person with the *most friends* in the data?

(b) Find *the length of the shortest path* from yourself to the famous person.

(c) Find *the number of people* who you are not friends with.