

# CS41 lab 4

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.

The learning goals of this lab session are to gain more experience with directed and undirected graphs, and to practice algorithm design for graph problems. There are more problems on this writeup than you have time to complete. Consider the lab a success if you can make good progress on two problems.

1. Suppose we run BFS starting from a given starting vertex  $s$ . Let  $L_0 = \{s\}$  be the set containing just the starting vertex. Then, for  $j > 0$ , define  $L_j$  to be the set of vertices which are adjacent to a vertex in  $L_{j-1}$  but *not* to any vertex in  $L_0 \cup L_1 \cup \dots \cup L_{j-2}$ . (**NB:** the notation  $\cup$  is the *union* of two sets, so  $L_0 \cup L_1 \cup \dots \cup L_{j-2}$  refers to all vertices that belong to any of the sets  $L_0$  through  $L_{j-2}$ ).

Intuitively,  $L_j$  contains the vertices that are discovered by BFS for the first time while visiting a vertex in  $L_{j-1}$ .

Prove that for a vertex  $v$  in  $L_j$ , there is a path from  $s$  to  $v$  consisting of  $j$  edges, and there is no shorter path (i.e., that the *shortest* path from  $s$  to  $v$  has length  $j$ ).

2. **Cycle Detection (K & T 3.2)** Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should **output a cycle**. Otherwise, your algorithm should output NO. Your runtime should be  $O(m + n)$  for a graph with  $m$  edges and  $n$  vertices.

**Hint:** Don't forget edge cases. Don't forget to return the cycle if one is detected.

Write up pseudocode for your algorithm. Prove that it halts, runs in polynomial time, and is correct.

3. **Strongly Connected Components.** Let  $G = (V, E)$  be a directed graph. Vertices  $u$  and  $v$  are *strongly connected* if there are  $u \rightsquigarrow v$  and  $v \rightsquigarrow u$  paths in  $G$ . A *strongly connected component* is a set of vertices  $C \subseteq V$  such that  $u, v$  are strongly connected for all  $u, v \in C$  (and no other vertices are strongly connected to a vertex  $u \in C$ .)

Design and analyze an algorithm to identify all strongly connected components in  $G$ . What is the runtime of your algorithm?

4. **Bipartiteness.** A graph is *bipartite* if its vertices can be partitioned into two sets  $A$  and  $B$  such that no two vertices in  $A$  are connected by an edge, and no two vertices in  $B$  are connected by an edge.
  - (a) A graph is called *two-colorable* if its vertices can be assigned colors "red" and "blue" so that no two adjacent vertices share the same color. Prove that a graph is two-colorable if and only if it is bipartite.
  - (b) (**challenge**) Try to design an algorithm using BFS to determine if a given graph  $G$  is bipartite.