College of Engineering

CS261

Programming Assignment 7

GRAPH TRAVERSALS

This assignment is shorter than previous assignments. It consists of two parts: a short implementation and a few questions.

Part 1: Implementation

Complete the implementation of the depth-first search (DFS) and breadth-first search (BFS) algorithms in **graph.c**.

To support the implementation of these algorithms, we have provided you with a circular doubly linked list (see **cirListDeque.h** and **cirListDeque.c**), which can be used as either a stack or a queue. This gives you a chance to apply some of your data structure knowledge from earlier in the course.

There is a reference implementation of DFS implemented recursively in **graph.c**. You should write an imperative implementation using a stack. You can use the recursive definition to check your own implementation.

In the **main** function there some paramaters that you can tweak to generate different graphs and run your two search algorithms on them. For every pair of nodes *A* and *B* in graph, the **main** function will determine whether *A* is reachable from *B* using either DFS or BFS. You may want to add some new graphs of your own, or modify this file to run tests on several different graphs at once.

You may assume that all graphs are *undirected* but you should NOT assume that they are all *connected*. That is, there are not necessarily paths between all nodes.

Files: graphAssignmentFiles.zip

Part 2: Questions

Run your program with each of the five example graph and compute the reachability between each pair of nodes using both BFS and DFS on each. Then answer the following questions:

1. How is the graph stored in the provided code -- adjacency matrix or edge list?
2. Which of the graphs are connected? How can you tell?
3. Imagine that we ran each search in the other direction (from destination to source, instead of source to destination) -- would the output change at all? What if the graphs were *directed* graphs?
4. What are a few pros and cons of DFS vs. BFS?
5. What's the Big O execution time to determine if a node is reachable from another node?

Tips

Pay careful attention to the struct definitions. In particular, the **Graph** struct contains an array of **Vertex** structs (all of the vertexes in the graph), while the **Vertex** struct contains an array of *pointers* to the **Vertex** structs that are neighbors of that vertex. These pointers point to the same vertexes stored in the array in the **Graph** struct!

If a vertex named **pueblo** has at least three neighbors, to access the third neighbor we would write: **pueblo->neighbors[2]**

What to submit

- graph.c
- answers.txt (or .pdf)

Grading Rubric (Total: 75pts)

- Compiles (10pts)
- DFS (20pts)
- BFS (20pts)
- Questions (5pts each = 25pts)

]