Data Structures and Algorithms
Assignment 6

Due: 11:59 PM on 12/1/2021 in Canvas
(100 points)

Instructions

Submit your code in Canvas. Include a brief `README` file explaining what you did, especially if you implemented some of the suggestions for extra credit. Please put down your name in the comments at the beginning of all the source code files.

PROGRAM REQUIREMENTS

In this assignment you will implement three basic graph algorithms: breadth-first search, depth-first traversal, and topological sort. We are not providing any code template, but you must follow the input and output formats specified below. Read carefully what you are required to do to receive full credit and attempt the extra credit tasks only after you are done with the requirements.

First, implement a graph data structure using adjacency lists. Your code must read graphs of arbitrary size from a file (as long as there is enough memory available, your code must work: do not assume a maximum number of vertices). The input format is as follows: one line per vertex, the first element is the vertex id and the following numbers are the adjacent vertices. The input file will always be a bunch of integers and white space. For example,

```
1 3 4
2
3 1 4
4 1 3
```

is a graph with four vertices, and three undirected edges: (1, 3), (1,4) and (3,4).

Then, implement three algorithms:

- Breadth-first search. Calculate the distance from vertex 1 to all other vertices using BFS. Then, print all the vertices sorted by distance from vertex 1. Note that if the graph is unconnected, some nodes may have distance ∞.
- Depth-first traversal. Calculate discovery and finish times using DFT. Then, print all the vertices sorted by discovery time.

- Topological sort. Print the topological sort of the graph.

A few comments:

- Look carefully at the sample input and output files and follow the same format.
- When you look through vertices, visit them in increasing order.

Extra credit (20 points):

- Use BFS to determine whether a graph is connected. The input is a graph, and the output is yes or no.
- Implement an algorithm to detect whether a graph has a cycle. The input is a graph, and the output is yes or no.
- Design and implement an algorithm that takes as its input a graph $G = (V, E)$ and a permutation of V, and outputs whether the permutation of V is a topological sort of G. Your algorithm must run in $\Theta(|V| + |E|)$.