Data Structures (CS201)

Lab Assignment 8 (Graded)

October 23, 2021

Instructor: Anil Shukla
Due: October 24, 10 am   Total Marks: 20

Note: Graded means the marks will be counted for the final grading. Place proper comments in your source code. Write in C only. C++ is not allowed.
Note: Plagiarism is strictly prohibited. An appropriate disciplinary action will be taken if you are found to be involved in plagiarism.
Note: The instructions for submitting the assignment is mentioned in the google classroom. Carefully read the same and follow the instructions.
Note: At the end, find some test cases.

(1) Write a single C-program for the following graph problems:

(1a) Graph representations: For this problem you need to represent an input graph $G = (V, E)$ as adjacency lists. Your program should ask user to input a simple graph and then represents the same using adjacency lists representation. To be precise, your program should first ask for the number of vertices n of the graph. Assume that the vertices are labelled as $0, 1, \ldots, n-1$. Then your program should ask to enter D (for directed graphs) or U (for undirected graphs). Note that for the case where the user had entered D treat the edges as directed and treat the edges as undirected for the case U. Then, your program should ask user to enter the set of edges E of the graph one by one. Once the user types 'Q' your program should stop taking the edges. The order of edges within an adjacency lists is irrelevant. Feel free to insert as you wish.
Print the entered graph nicely for both the cases D or U. See the test cases below. (10 marks)
Note: Assume that the user will enter a correct simple graph. That is, without loops and multiple edges.

(1b) Breadth-first search: For this problem you need to write the function BFS(G, s) in your C-program. Refer Chapter 22, of the CLRS book (3rd Edition) for the pseudocode. In particular, you need to run the breadth-first search algorithm on the graph $G = (V, E)$ entered by the user on $1a$. To be precise, your program should ask user to enter a (source) vertex s belonging to the graph entered in $1a$ part. Then your program runs the BFS algorithm (function) on $G = (V, E)$ from vertex s. Your program should then print the length of the shortest path (distance) of every vertex v from s. If v is not reachable from s, print 'not reachable from s' for such vertices v. Refer test cases below. (10 marks)

Test Cases

Test Cases for problem 1(a)

Enter the number of vertices: 8
Your vertices are numbered from 0, 1, . . . , 7
Enter *D* for directed or *U* for undirected graphs: *U*
Enters edges one by one of your Undirected graph. Once done, press '*Q*':
$(0, 1)$
$(0, 4)$
$(1, 5)$
$(2, 3)$
$(2, 5)$
$(2, 6)$
$(3, 6)$
$(3, 7)$
$(5, 6)$
$(6, 7)$
Q
Your undirected graph is as follows:
Vertex 0: 1 -> 4
Vertex 1: 0 -> 5
Vertex 2: 5 -> 6 -> 3
Vertex 3: 7 -> 2 -> 6
Vertex 4: 0
Vertex 5: 2 -> 1 -> 6
Vertex 6: 7 -> 5 -> 2 -> 3
Vertex 7: 6 -> 3

Test cases for problem 1b:

Enter a source vertex *s* from where you wants to run the BFS algorithm on the graph you entered for problem 1a
Your options are $0, 1, \ldots, 7$: 1
BFS computed the following distances:
The distance of the vertex 0 from source vertex 1 is: 1
The distance of the vertex 1 from source vertex 1 is: 0
The distance of the vertex 2 from source vertex 1 is: 2
The distance of the vertex 3 from source vertex 1 is: 3
The distance of the vertex 4 from source vertex 1 is: 2
The distance of the vertex 5 from source vertex 1 is: 1
The distance of the vertex 6 from source vertex 1 is: 2
The distance of the vertex 7 from source vertex 1 is: 3