**Assignment 9 Graph Traversals (2.5% - 10 points)** CSI2110 (Fall 2022)

Due date: Friday November 25 at 11:59PM.

**Late assignment policy: *1min-24hs late are accepted with 30%off; no assignments accepted after 24hs late.***

Below is an undirected graph represented using adjacency lists:

0: (0,1), (0,3), (0,4)

1: (1,0), (1,2), (1,3), (1,6), (1,7)

2: (2,1), (2,6), (2,7)

3: (3,0), (3,1), (3,5)

4: (4,0), (4,5)

5: (5,3), (5,4), (5,6), (5,8)

6: (6,1), (6,2), (6,5), (6,7), (6,9)

7: (7,1), (7,2), (7,6), (7,9)

8: (8,5), (8,9)

9: (9,6), (9,7), (9,8)

***Question 1.* [1 points]** Draw the graph corresponding to the given adjacency lists:

Chart, scatter chart

Description automatically generated

***Question 2.* [1 points]**

Change the representation of the graph from adjacency lists to **adjacency matrix,** and show the matrix.

***Question 3.* [4 points = 3+1]**

**3a)** Using the DFS algorithm in the Appendix, perform a depth-first search traversal on the given graph starting from node **0** and using the **adjacency lists** representation of the graph. The adjancency lists will influence the order in which the vertices are considered; for example G.incidentEdges(0) will return the list : (0,1), (0,3), (0,4) so that the edges will be considered in this order.

List the vertices in the order they are visited, and list the edges in the order they are labeled by the algorithm, displaying their labels.

Vertices in order of visit:

Edges and labels in order of visit:

*Please give the edges in the order they are labeled, display each edge in the direction of visit, and use the first letter of the label; Example - if a discovery edge was found coming from vertex b to a, the entry for this edge will be displayed "(b,a) D, "*

**3b)** Suppose DFS is run starting at vertex 0 using the same adjacency lists, but instead of the recursive algorithm you use the algorithm that uses a Stack. The edges incident to a vertex v are given by the same method G.incidenceEdges(v), and when inserted in the stack this order is followed.

Does the order in which vertices are visited change? If so, give the vertices in order of visit; if not explain why not.

***Question 4.* [4 points = 3+1]**

**4a)** Using the BFS algorithm in the Appendix, perform a breadth-first search traversal of the graph starting from node **0** and using the **adjacency lists** representation of the graph.

List the vertices in the order they are visited circling the groups of vertices that belong to each list L\_0, L\_1, L\_2, etc. List the edges in the order they are labeled by the algorithm, displaying their labels. Please, use similar format as suggested in question 3a.

Vertices in order of visit:

Edges and labels in order of visit:

**4b)** Suppose BFS is run starting at vertex 0 using adjacency lists where the order of the edges are completely reversed. For example the lists are: 0: (0,4), (0,3), (0,1) ... 9: (9,8), (9,7)

Does the order in which vertices are visited change? Explain why.

Do the sets of vertices contained in lists L0, L1, L2, ... change? Explain why.

**Appendix**

Depth-first Search algorithm (DFS)

**Algorithm** ***DFS***(***G, v***)

**Input** graph ***G*** and a start vertex ***v*** of ***G***

**Output** labeling of the edges of ***G*** in the connected component of ***v*** as discovery edges and back edges

***setLabel***(***v, VISITED***)

**for all**  ***e*** ∈ ***G.incidentEdges***(***v***)

**if** ***getLabel***(***e***) = ***UNEXPLORED***

***w*** ← ***opposite***(***v,e***)

**if *getLabel***(***w***) = ***UNEXPLORED***

***setLabel***(***e, DISCOVERY***)

***DFS***(***G, w***)

**else**

***setLabel***(***e, BACK***)

**Algorithm** ***DFS***(***G***)

**Input** graph ***G*** **Output** labeling of the edges of ***G*** as discovery edges and back edges

**for all**  ***u*** ∈ ***G.vertices***()

***setLabel***(***u, UNEXPLORED***)

**for all**  ***e*** ∈ ***G.edges***()

***setLabel***(***e, UNEXPLORED***)

**for all**  ***v*** ∈ ***G.vertices***()

**if** ***getLabel***(***v***) = ***UNEXPLORED***

***DFS***(***G, v***)

Breadth-first Search algorithm (BFS)

**Algorithm** ***BFS***(***G***)

**Input** graph **Output** labeling of the edges and partition of the vertices of ***G***

**for all**  ***u*** ∈ ***G.vertices***()

***setLabel***(***u, UNEXPLORED***)

**for all**  ***e*** ∈ ***G.edges***()

***setLabel***(***e, UNEXPLORED***)

**for all**  ***v*** ∈ ***G.vertices***()

**if** ***getLabel***(***v***) = ***UNEXPLORED***

***BFS***(***G, v***)

**Algorithm** ***BFS***(***G, s***)

***L***0←new empty sequence

***L***0***.insertLast***(***s***)

***setLabel***(***s, VISITED***)

***i*** ← 0

**while**  ! ***Li.isEmpty***()

***Li*** +1←new empty sequence

**for all**  ***v*** ∈ ***Li.elements***()

**for all**  ***e*** ∈ ***G.incidentEdges***(***v***)

**if** ***getLabel***(***e***) = ***UNEXPLORED***

***w*** ← ***opposite***(***v,e***)

**if *getLabel***(***w***) = ***UNEXPLORED***

***setLabel***(***e, DISCOVERY***)

***setLabel***(***w, VISITED***)

***Li*** +1***.insertLast***(***w***)

**else**

***setLabel***(***e, CROSS***)

***i*** ← ***i*** +1