CS 385, Homework 3: Graph Algorithms

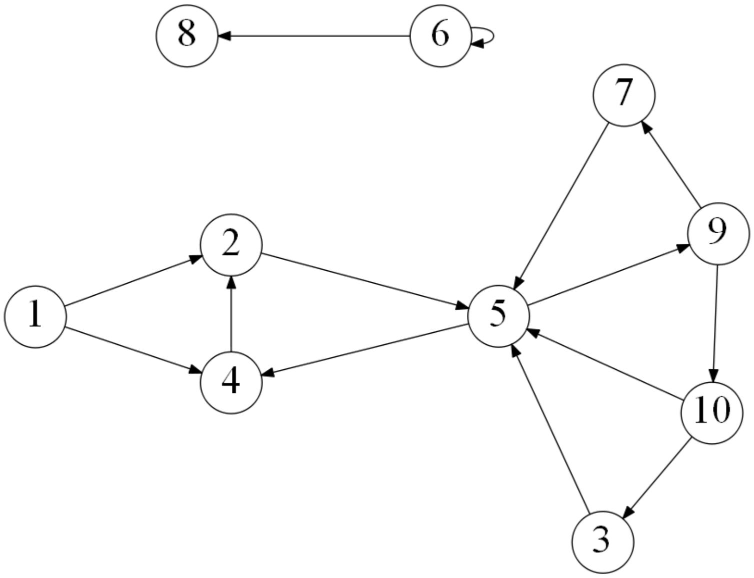
Name: Rohan Kallur

Date: 10/17/2022

Point values are assigned for each question.

Points earned: ____ / 100

Consider the following graph:



1. Draw how the graph would look if represented by an adjacency matrix. You may assume the indexes are from 1 through 10. Indicate 1 if there is an edge from vertex A -> vertex B, and 0 otherwise. (10 points) ****Blank means 0**

Node	1	2	3	4	5	6	7	8	9	10
1		1		1						
2					1					
3					1					
4		1								
5				1					1	
6						1		1		
7					1					
8										
9							1			1
10			1		1					

2. Draw how the graph would look if represented by an adjacency list. You may assume the indexes are from 1 through 10. (10 points)

   [1]→[2]→[4]
   [2]→[5]
   [3]→[5]
   [4]→[2]
   [5]→[4]→[9]
   [6]→[6]→[8]
   [7]→[5]
   [8]
   [9]→[7]→[10]
   [10]→[3]→[5]

3. List the order in which the vertices are visited with a breadth-first search. If there are multiple vertices adjacent to a given vertex, visit the adjacent vertex with the lowest value first. (10 points)

   [1], [2], [4], [5], [9], [7], [10], [3], [6], [8]

4. List the order in which the vertices are visited with a depth-first search. If there are multiple vertices adjacent to a given vertex, visit the adjacent vertex with the lowest value first. (10 points)

   [1], [2], [5], [4], [9], [7], [10], [3], [6], [8]

5. a) What is the running time of breadth-first search with an adjacency matrix? (5 points)

   θ(V^2)

   b) What is the running time of breadth-first search with an adjacency list? (5 points)

   θ(V + E)

6. a) What is the running time of depth-first search with an adjacency matrix? (5 points)

   θ(V^2)

   b) What is the running time of depth-first search with an adjacency list? (5 points)

   θ(V + E)

7. While an adjacency matrix is typically easier to code than an adjacency list, it is not always a better solution. Explain when an adjacency list is a clear winner in the efficiency of your algorithm? (5 points)

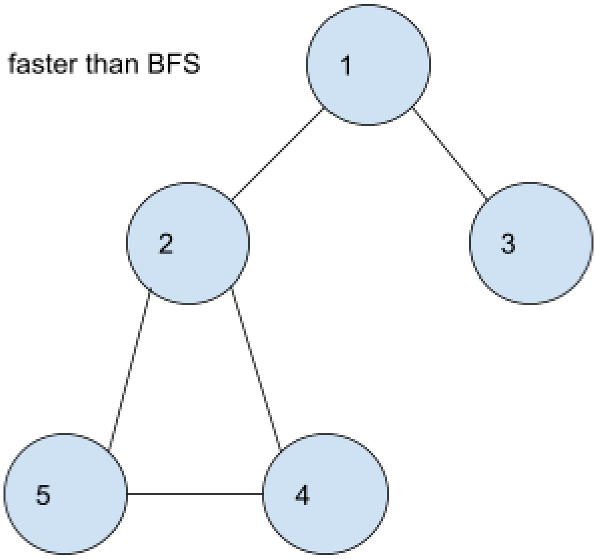
As the number of vertices get larger the adjacency list will grow more and more efficient, as the run time for the matrix will constantly be V^2, versus as the vertices get larger the run time of the list will get closer and closer to V. This applies for both BFS and DFS

8. Explain how one can use a breadth-first to determine if an undirected graph contains a cycle. (10 points)

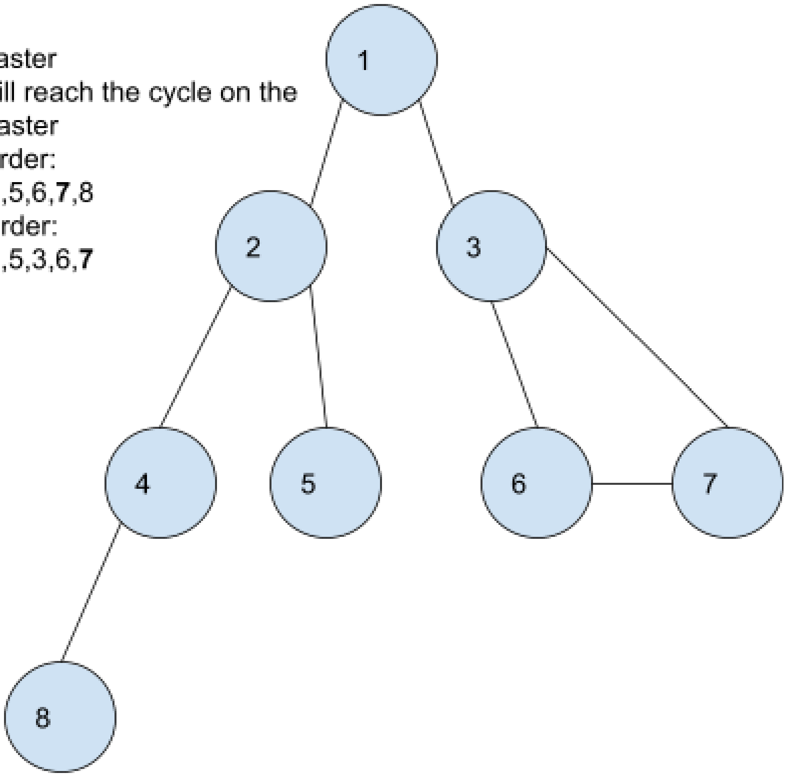
   iterate through the vertices, and if you find an adjacent vertex to the one that you are currently on that has been visited already and is not a parent of the current vertex you have a cycle

9. On undirected graphs, does either of the two traversals, DFS or BFS, always find a cycle faster than the other? If yes, indicate which of them is better and explain why it is the case; if not, draw two graphs supporting your answer and explain the graphs. (10 points)

DFS is Faster
DFS will get to 4 faster than BFS
DFS Order:
1,2,4,5,3
BFS Order
1,2,3,4,5



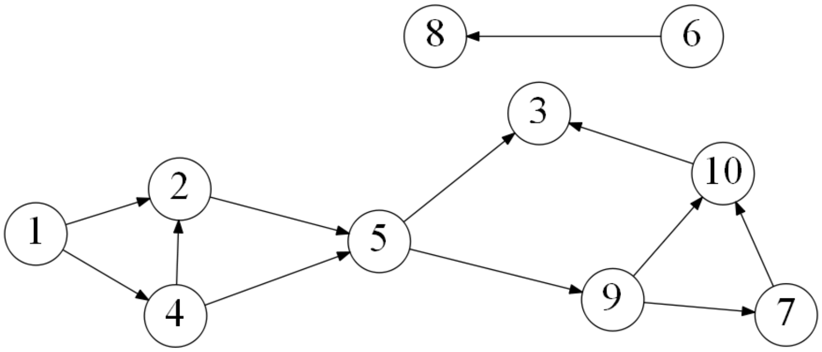
BFS Faster
BFS will reach the cycle on the Right faster
BFS Order:
1,2,3,4,5,6,**7**,8
DFS Order:
1,2,4,8,5,3,6,**7**



10. Explain why a topological sort is not possible on the graph at the very top of this document. (5 points)

There are cycles in the graph so if you were to do a topological sort you will reach a point where there are no vertices with no dependencies and you cannot continue. Also 6 doesn't have an indegree of 0 so you will never be able to access it.

Consider the following graph:



11. List the order in which the vertices are visited with a topological sort. Break ties by visiting the vertex with the lowest value first. (10 points)

[1], [6], [4], [8], [2], [5], [9], [7], [10], [3]