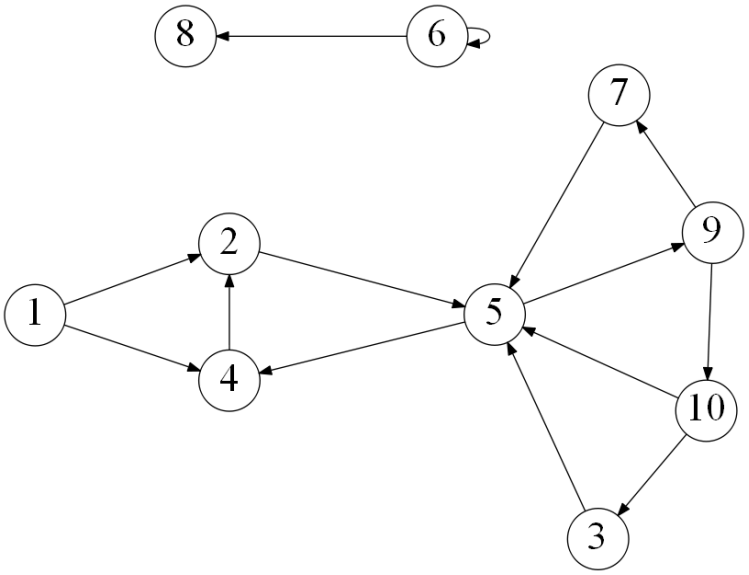
Name: Brian Sampson Date: 10/17/2022

Point values are assigned for each question. Points earned: ____ / 100

I pledge my honor that I have abided by the Stevens Honor System.

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)

0	1	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	1	0	1	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	0	0	0

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)

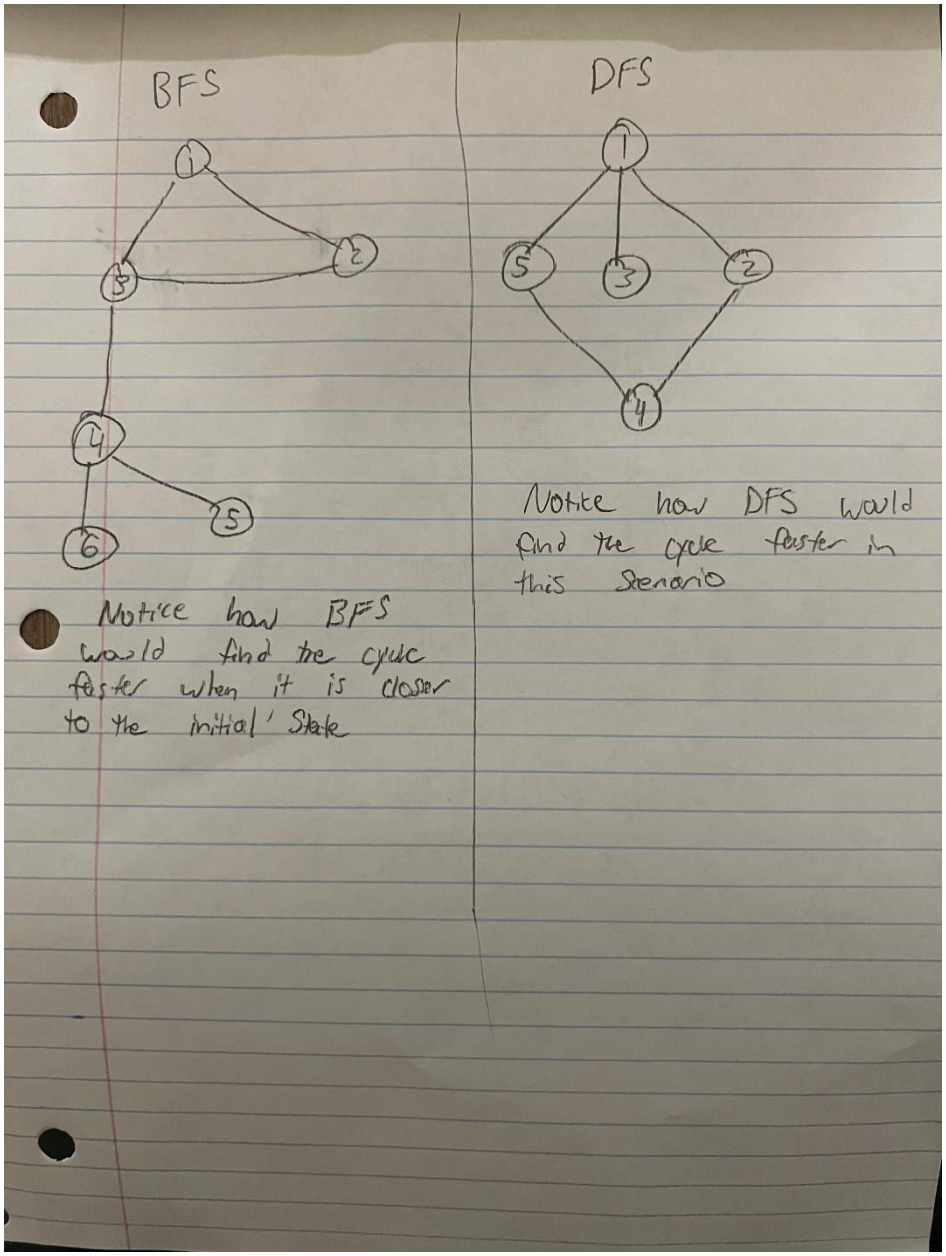
7. While an adjacency matrix is typically easier to code than an adjacency list, it is not always a better solution. Explain when an adjacency list is a clear winner in the efficiency of your algorithm? (5 points)
   An adjacency list is significantly more efficient with memory usage when there are a larger number of vertices and less edges.

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

If we visit a vertex that has been already visited and that node is not the parent node, there is a cycle in the undirected graph. (2 exceptions to the parent node rule below)

The exceptions are when there is a single node that is directed into itself or when there are two nodes cycles that are directed back into eachother on the entire graph.

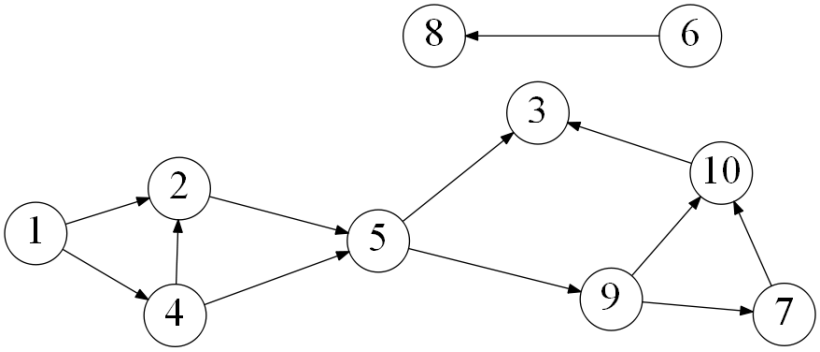
9. On undirected graphs, does either of the two traversals, DFS or BFS, always find a cycle faster than the other? If yes, indicate which of them is better and explain why it is the case; if not, draw two graphs supporting your answer and explain the graphs. (10 points)
   No, both DFS and BFS traverse at the same speed on undirected graphs.



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

The graph contains a cycle. 6 points to itself and once the first vertex is removed, there's no vertices with indegree of 0.

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