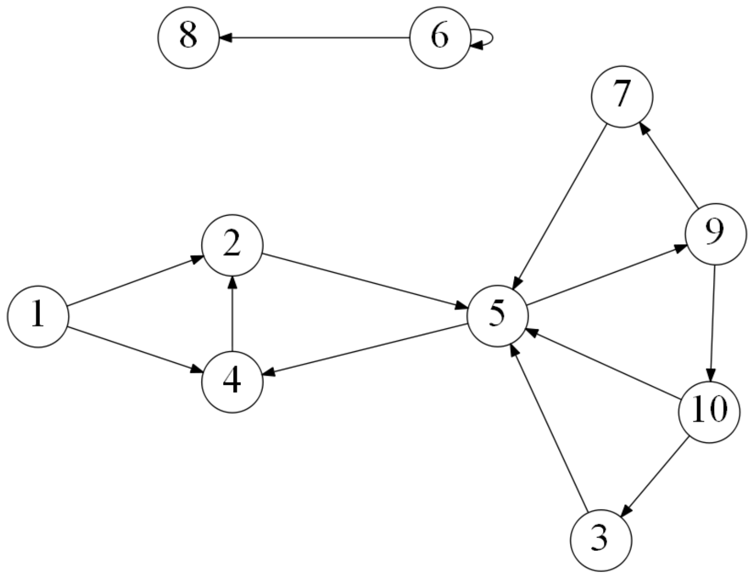
Name: Matthew McCaughan Date: October 19^{th}, 2022

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

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)

Vertex A/ Vertex B	1	2	3	4	5	6	7	8	9	10
1	0	1	0	1	0	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	1	0
6	0	0	0	0	0	1	0	1	0	0
7	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	1	0	0	1
10	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)

Index	
1	2 -> 4\
2	5 \

3	5\
4	2\
5	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)

Counter: 10

Array: 1 2 3 4 5 6 7 8 9 10

1 2 8 3 4 9 6 10 5 7

Queue:

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)

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

5. a) What is the running time of breadth-first search with an adjacency matrix? (5 points)
   $\Theta(V^2)$
   b) What is the running time of breadth-first search with an adjacency list? (5 points)
   $\Theta(V + E)$

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

   $\Theta(V^2)$

   b) What is the running time of depth-first search with an adjacency list? (5 points)
   $\Theta(V + E)$

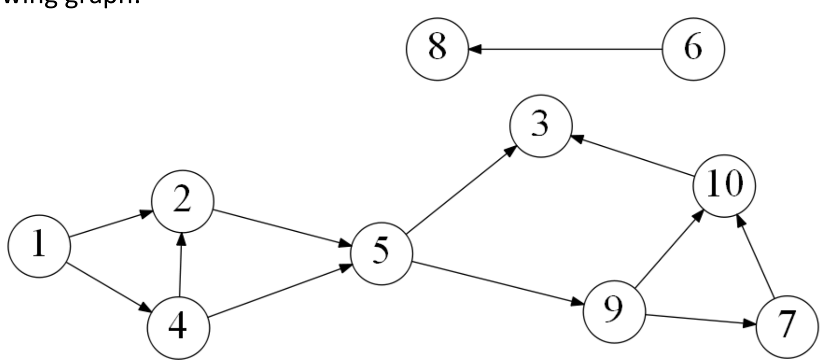
7. While an adjacency matrix is typically easier to code than an adjacency list, it is not always a better solution. Explain when an adjacency list is a clear winner in the efficiency of your algorithm? (5 points)
   An adjacency list is the clear winner for efficiency in an algorithm when the graph is large, containing many vertices, or when the graph is very sparce, containing little edges.

8. Explain how one can use a breadth-first to determine if an undirected graph contains a cycle. (10 points)
   A breadth-first search contains a cycle when a vertex contains edges that point to vertices that already have been visited in the graph. A cycle can be traced from the vertices pointed to, to the current vertex.

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)
   Depth first search will find cycles faster than the other because the vertex will occur twice on the stack of recursive calls, which is a faster way of knowing rather than seeing where each vertex points to.

10. Explain why a topological sort is not possible on the graph at the very top of this document. (5 points)
    Topological sort is not possible on the top graph because this graph contains at least 1 cycle, so it is not possible to find the correct order which satisfies all dependencies.

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)

Array: 1 2 3 4 5 6 7 8 9 10

Array: 0 0 0 0 0 0 0 0 0 0

Set: 3

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

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