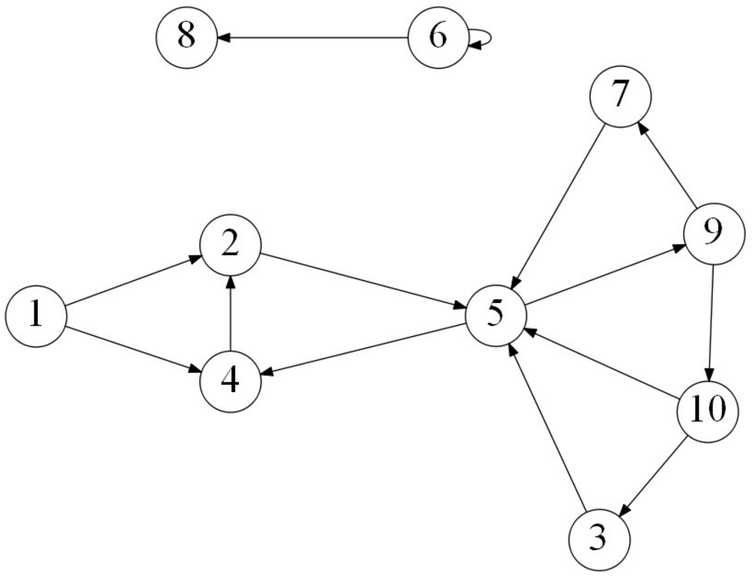
Name: ____Joshua Schmidt_______ Date: ___10/12/19______

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

Consider the following graph:



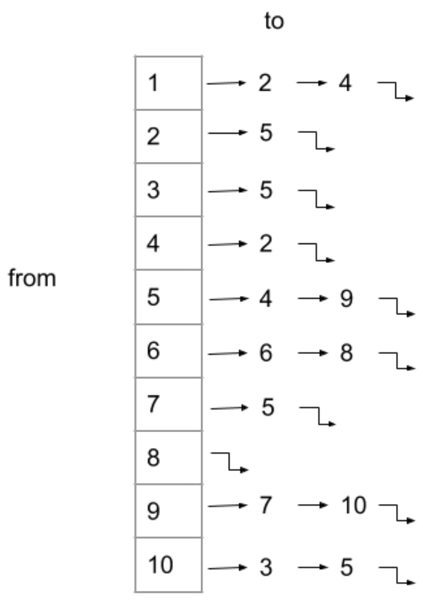
1. Draw how the graph would look if represented by an adjacency matrix. You may assume the indexes are from 1 through 10. Indicate 1 if there is an edge from vertex A -> vertex B, and 0 otherwise. (10 points)

from

to

-	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1
4	1	0	0	0	1	0	0	0	0	0
5	0	1	1	0	0	0	1	0	0	1
6	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	1	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	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)



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)

adjacency matrix bfs: runtime = $\Theta(V^2)$ = $\Theta(10^2)$ = $\Theta(100)$

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

adjacency list bfs: runtime = $\Theta(V+E)$ = $\Theta(10+14)$ = $\Theta(24)$

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

adjacency matrix dfs: runtime = $\Theta(V^2)$ = $\Theta(10^2)$ = $\Theta(100)$

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

adjacency list dfs: runtime = $\Theta(V+E)$ = $\Theta(10+14)$ = $\Theta(24)$

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)

Using an adjacency list is much better than using an adjacency matrix for large graphs or graphs with few edges. There are only several edges per given vertex in the graph, and the amount of vertexes is large at 10 total. This results in a running time of about five times higher for the adjacency matrix than for the adjacency list. This makes the adjacency list a clear winner.

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

You can start with a normal bfs algorithm. You compute the in-degree of each vertex in the graph, and add all the vertexes of in-degree of 0 to the queue. Additionally, initialize a count of the visited nodes to 0. Then while the queue is not empty, you can remove a vertex from the queue, increment the count of visited vertexes by 1, decrease the in-degree of neighboring vertexes by 1, and if the in-degree of a neighboring vertex becomes 0, add it to the queue. When the queue is empty, check if the number of visited nodes is equal to the number of nodes in the graph. If this is false, the graph has a cycle.

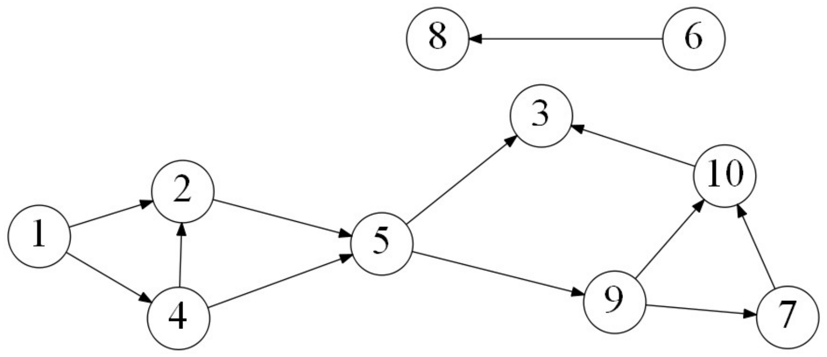
9. On undirected graphs, does either of the two traversals, DFS or BFS, always find a cycle faster than the other? If yes, indicate which of them is better and explain why it is the case; if not, draw two graphs supporting your answer and explain the graphs. (10 points)

With bfs, typically the way to find cycles is to traverse the graph and count the number of vertexes you have visited. If the number of vertexes visited is not equal to the number of vertexes in the graph at the end, then there is a cycle. With a dfs, you don't have to wait until the end to find out if there is a cycle. You are keeping track of the vertexes visited as you do the traversal, so if there is a cycle you can leave the iteration in the function immediately. If there is no cycle, they will both take the same amount of time to determine that.

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 for graphs that contain cycles. Because the graph at the top of the document contains not just one but multiple cycles (2, 5, 4; 5, 9, 7; etc.), it is not possible to use topological sorting.

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