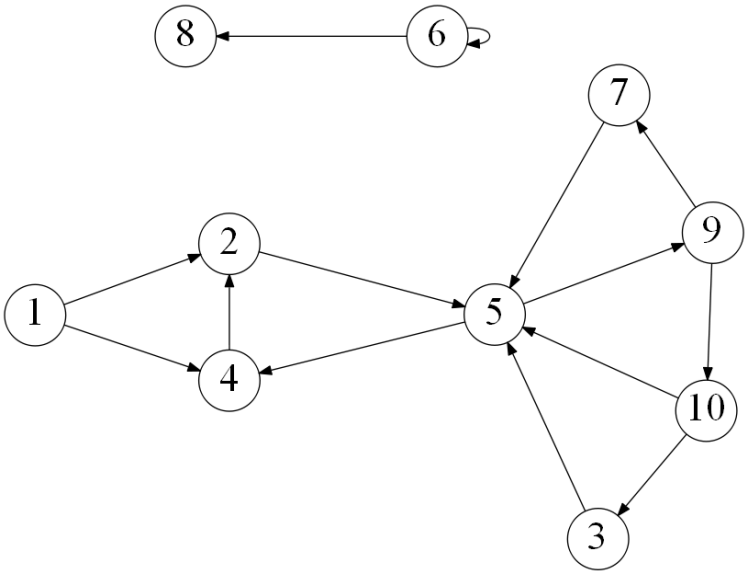
Name: __Edward Yaroslavsky_____ Date: __10/14/19_____

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)

	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	1	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)

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)

$\theta(V^2)$

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)

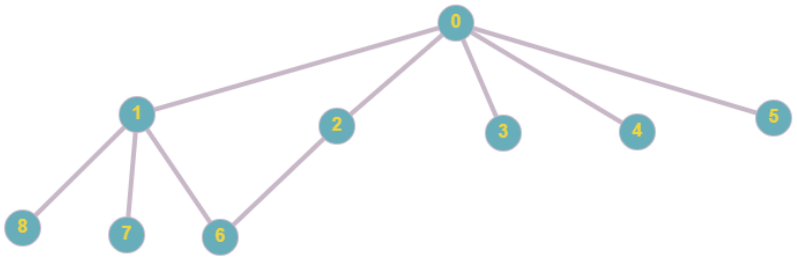
In the case with a large number of vertices being directed towards a minimal number of other vertices, resulting in minimal edges, the adjacency list is a clear winner in efficiency. This is because while an adjacency matrix takes much longer to run since it has to store V^2 elements, with V being the number of vertices, an adjacency list would be much more efficient since it only stores the vertices which a given edge is directed to. That way, the run time of an adjacency matrix in this case would resemble $\theta(n^2)$ due to it having to consider all the possible combinations of vertices whereas the run time of an adjacency list would resemble $\theta(n)$.

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

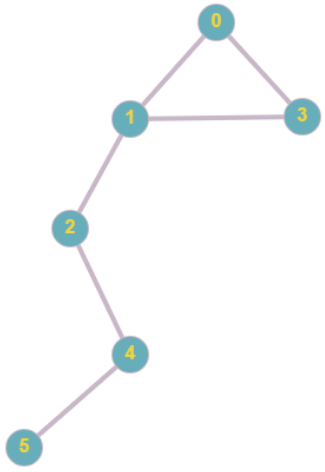
   One can use a BFS to determine if an undirected graph contains a cycle by going through the adjacent vertices of a current vertex. Then, by adding each adjacent vertex to a queue, we know what adjacent vertex we should make our new current vertex, repeating this pattern and adding to the queue. However, if a current vertex adds an adjacent vertex that has already been in the queue, then we know that the undirected 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)

   On undirected graphs, DFS and BFS do not always find a cycle faster than the other. In an undirected graph such as:



   a DFS approach would be more efficient, while in an undirected graph such as:

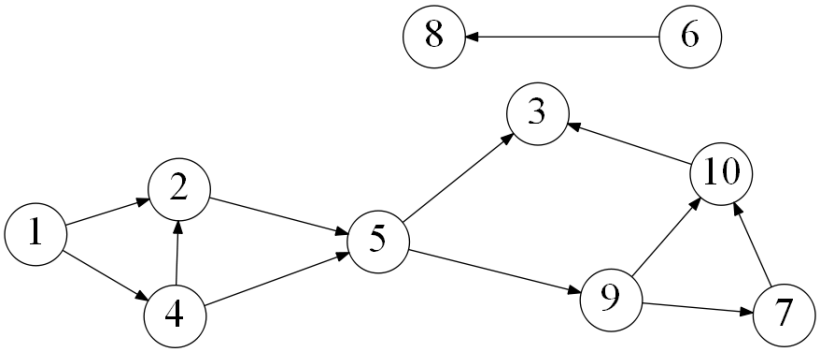


   a BFS approach would be more efficient.

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

    A topological sort is not possible on the graph at the top because the graph at the top is a directed graph with cycles. This would make the graph a Directed Acyclic Graph (DAG), where a topological sort is not possible on those types of graphs.

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)

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