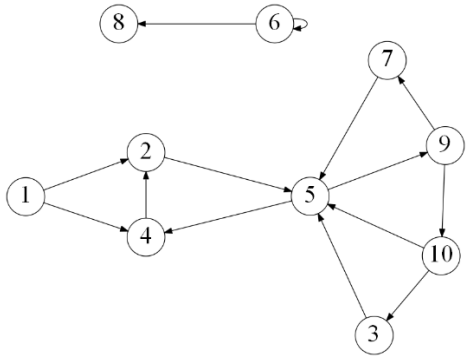
Name: Breona Pizzuta Date: 10/16/24

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)

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

   Using an adjacency matrix: $\Theta(V) + \Theta(V^2) = \Theta(V^2)$.
   Using an adjacency list: $\Theta(V) + \Theta(V + E) = \Theta(V + E)$.

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

   Using an adjacency matrix: $\Theta(V) + \Theta(V^2) = \Theta(V^2)$.
   Using an adjacency list: $\Theta(V) + \Theta(V + E) = \Theta(V + E)$.

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)

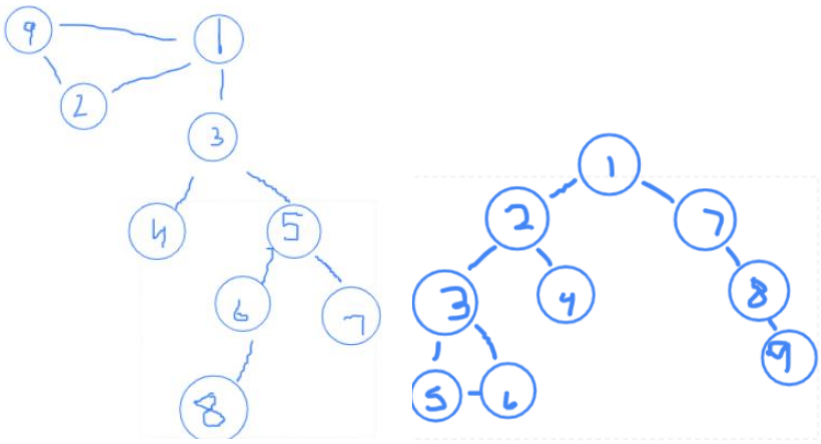
Adjacency matrices will be better for smaller and dense graphs. Adjacency lists are better for larger and sparse graphs. When there is a lot of space and less vertices/edges to traverse then the adjacency list would be better since the sparsity would cause many cells in a adjacency matrix to waste memory

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

One can use a breadth-first to determine if an undirected graph contains a cycle by keeping record of every node that it has already visited. If the algorithm tries to visit the same node again, then we will know that the graph contains 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)

In undirected graphs, neither DFS nor BFS is universally faster at detecting cycles. The efficiency of cycle detection depends on the specific structure of the graph rather than the inherent properties of the traversal method.

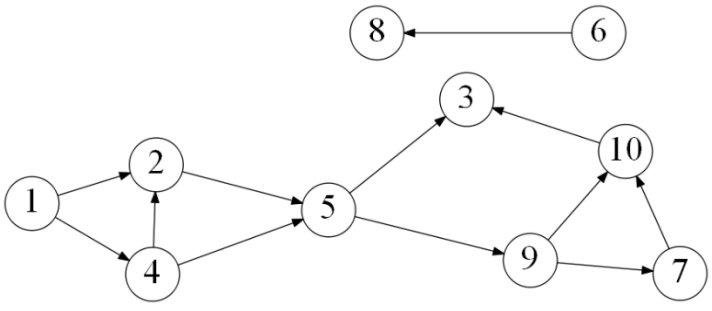


In the image on the left, BFS would be faster than DFS since the cycle is closer to the root of one. In this case DFS would be slower. In the image on the right, DFS would be faster than BFS because the cycle is further away from the root. BFS would be slower in this case.

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

The graph has a cycle at vertex 6, which means that it is not possible to find a correct order satisfying all the dependencies. Topological sort will fail.

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