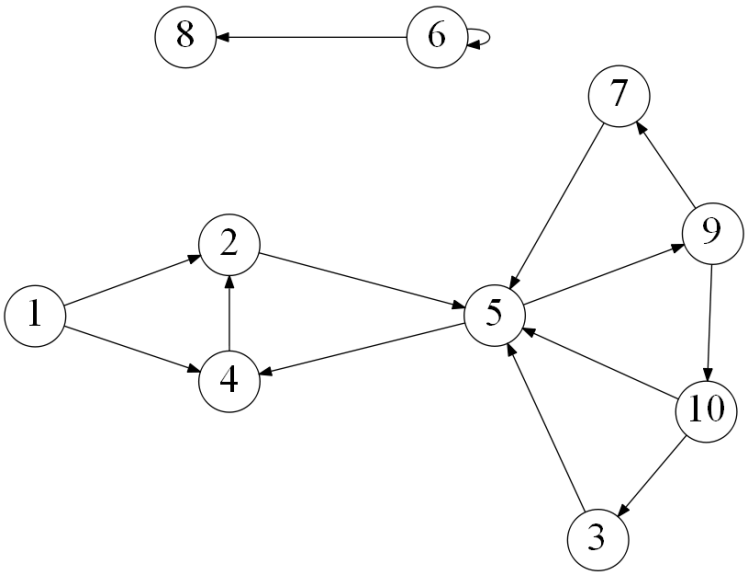
CS 385, : Graph Algorithms

Name: _____ Date: _____

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)

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)

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)

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)

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)

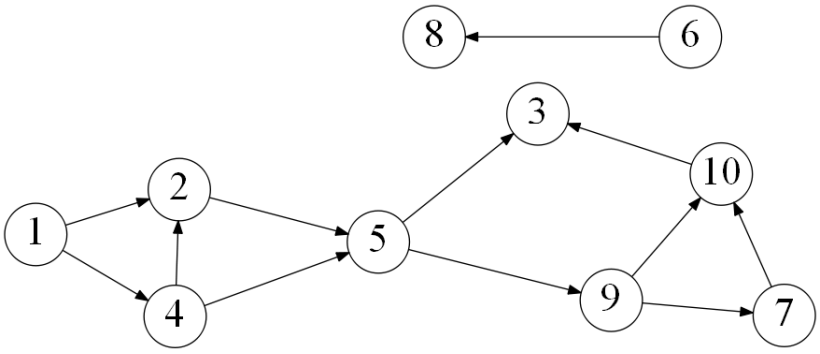
7. While an adjacency matrix is typically easier to code than an adjacency list, it is not always a better solution. Explain when an adjacency list is a clear winner in the efficiency of your algorithm? (5 points)

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

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)

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

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)