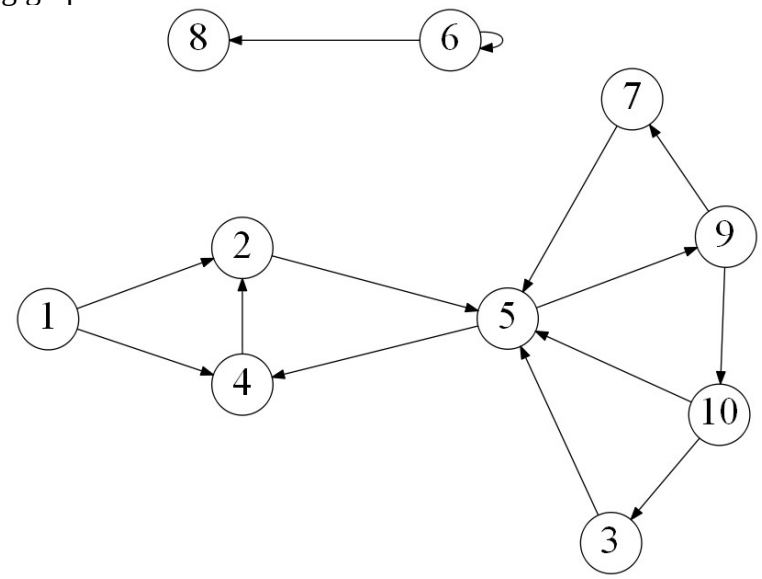
CS 385, Homework 3: Graph Algorithms

Name: ____Joshua Schmidt_____ Date: ___10/12/19_____

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

*I pledge my honor that I have abided by the Stevens Honor System. - Joshua Schmidt*

Consider the following graph:

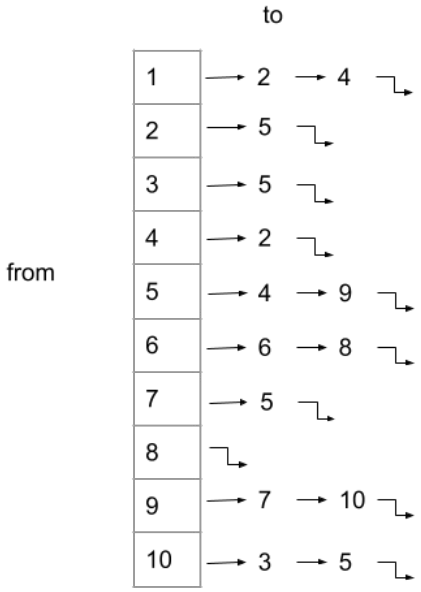



1. Draw how the graph would look if represented by an adjacency matrix. You may assume the indexes are from 1 through 10. Indicate 1 if there is an edge from vertex A -> vertex B, and 0 otherwise. (10 points)

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 = $\Theta(|V|)^2$

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

   adjacency list bfs: runtime = $\Theta(|V|+|E|)$

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

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

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

   adjacency list dfs: runtime = $\Theta(|V|+|E|)$

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)

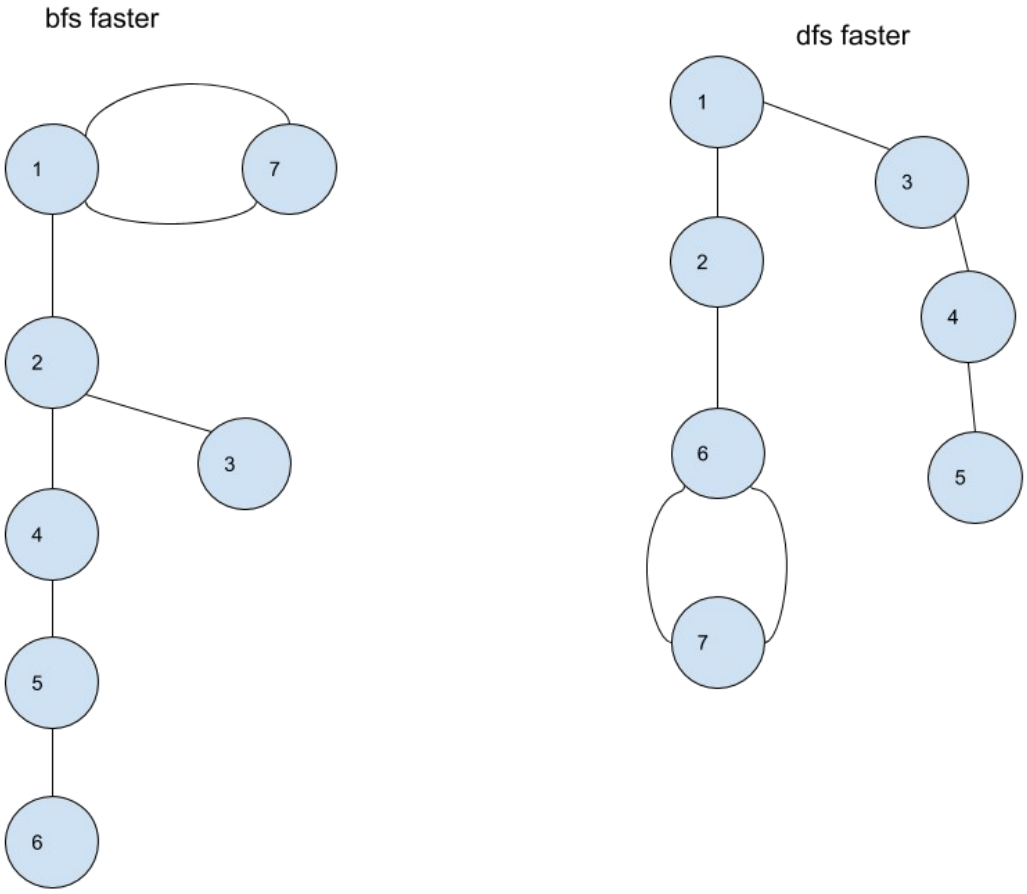
   Using an adjacency list is much better than using an adjacency matrix for large graphs or graphs with few edges. There are only several edges per given vertex in the graph, and the amount of vertexes is large at 10 total. This results in a running time of about five times higher for the adjacency matrix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

In a breadth-first search, you already keep track of the nodes that you have previously visited. When you visit a node, you add the node to an adjacency matrix or adjacency list or similar data structure. If you visit a node twhen doing breadth-first search that is already in that data structure of visited vertexes, then by definition 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)

The two traversals can each be faster than the other depending on the graph given. For example:



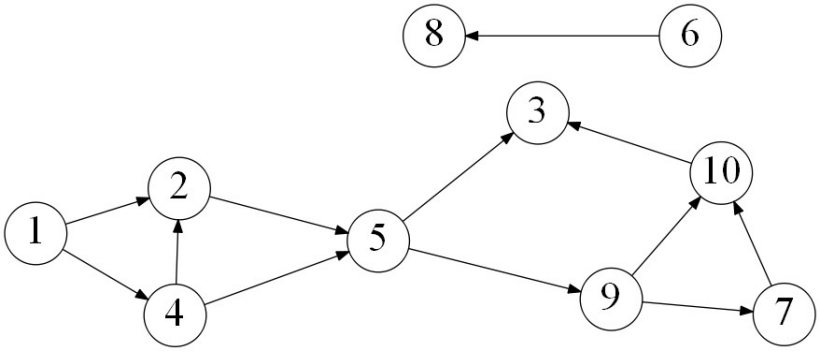

In the first graph, breadth-first search is faster because the cycle is close to the first node (node 1), but is connected to a high number node, so depth first search will go all the way down to node 6 before going back to 8. Depth-first search: (1, 2, 3, 4, 5, 6, 7, 1- cycle found). Breadth-first search: (1, 2, 7, 3, 4, 1 - cycle detected).

In the second graph, depth-first search is faster because there is a cycle far away from the starting node, and a long chain going away from the cycle chain. Therefore breadth-first search will go down both sides, while depth-first search will stay largely on the left side. Depth-first search: (1, 2, 6, 7, 6 - cycle found). Breadth-first search: (1, 2, 3, 6, 4, 7, 5, 6 - cycle found).

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. This is because when you go through the cycle, you have to visit a previously-visited node in order to continue. You are not able to do this in 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, 6, 8, 9, 7, 10, 3]