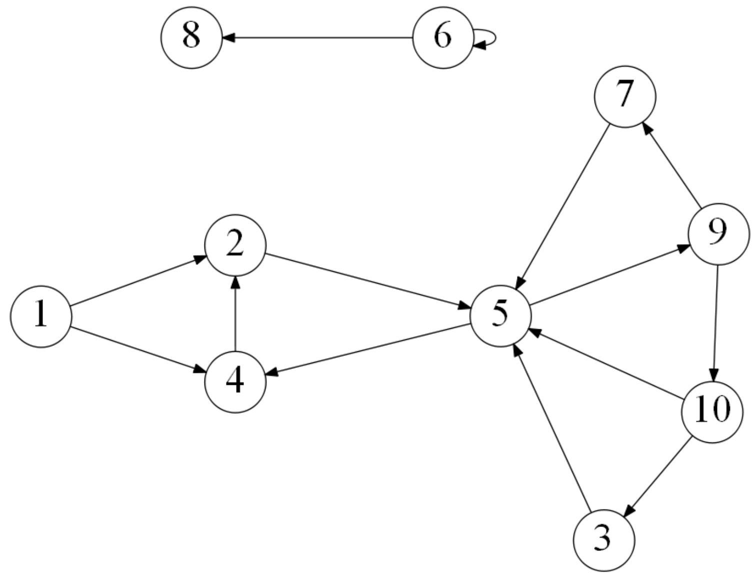
Name: Liam Brew

Date: 03.09.2020

Point values are assigned for each question.

Points earned: ____ / 100

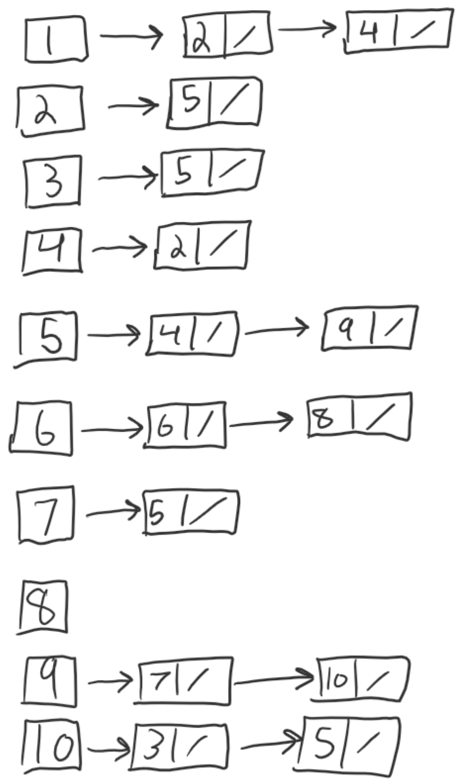
Consider the following graph:




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)




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)

   Matrix = $\theta(v^2)$
   List = $\theta(v + E)$

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)

   Matrix = $\theta(v^2)$
   List = $\theta(v + E)$

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)

   The list has a better space complexity of $\theta(v + E)$ compared to the matrix's $\theta(v^2)$.

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

   When coding the BFS algorithm, you can have it keep a record of every node that it has formerly visited. If the algorithm attempts to visit the same node twice, it is then known that the graph contains a cycle.

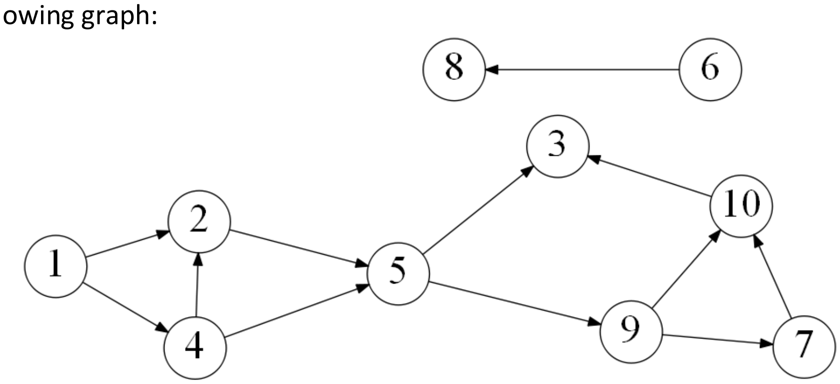
9. On undirected graphs, does either of the two traversals, DFS or BFS, always find a cycle faster than the other? If yes, indicate which of them is better and explain why it is the case; if not, draw two graphs supporting your answer and explain the graphs. (10 points)

   DFS is more efficient at finding a cycle. This is because as you move down the tree and record visited nodes, these nodes can easily be unmarked when you backtrack back up the tree.

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

   You cannot perform a topological sort because Node 6 points to itself.

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