Homework 8 - Graphs

Problem 1:

What is the big-Oh space complexity of an adjacency list? Justify your answer.

ANS

Assume that the number of vertices of a graph is ***n***, and there are ***e*** edges in it. For an adjacency list, the big-Oh space complexity is $O(n + e)$.

We use the adjacency list to represent a graph, which means each node in the graph are stored in a Double Linked List, and for each node storing a list of its neighbors
First for each node of graph, the big-Oh space complexity is $O(n)$. Then,
for undirected graph every edge is stored in DLL for twice, while for directed graph every edge is stored in DLL for once.
The largest number of edges is *2E, which* $\Rightarrow E$. Thus, for a graph representing by an adjacency list, the big-Oh space complexity is $O(n + e)$.

Problem 2:

What is the big-Oh space complexity of an adjacency matrix? Justify your answer.

ANS

For each vertex, it takes n blocks of space, and its adjacency matrix takes n^2 blocks of space to represent all of the relationships given.
Therefore, given n nodes, the total space required by the adjacency matrix is $n^2 + n$.
The space complexity in Big-O is $O(n^2)$.

Notice that undirected adjacency matrix is symmetrical, one of its diagonals from left top to right bottom is always zero, so it takes $\frac{n(n-1)}{2}$ blocks of space to represent all of the relationships given.

The Big-O is also $O(n^2)$.

Problem 3:

What is the big-Oh time complexity for searching an entire graph using *depth-first search* (DFS)? Does the representation of the graph make a difference? Justify your answer.

.

ANS

Assume that the number of vertices of a graph is ***n,*** and there are ***e*** edges in it.

When traversing the graph using DFS, for each node we only invoke the DFS function once. It is because once a node has already been visited and marked, we won't search from it again. For each node *x,* we consider all other nodes *y* which *x* has directed-edges in to.

If the representation of the graph is *an adjacency list,* the big-Oh time complexity for searching an entire graph is $O(n+e)$

If the representation of the graph is *an adjacency matrix,* the big-Oh time complexity for searching an entire graph is $O(n^2)$

Problem 4:

What is the big-Oh time complexity for searching an entire graph using *breadth-first search* (BFS)? Does the representation of the graph make a difference? Justify your answer.

ANS

Assume that the number of vertices of a graph is ***n,*** and there are ***e*** edges in it.

BFS is a procedure that borrows queues for storage, looking up hierarchically. When traversing the graph using BFS, we will only push each node in the queue for once. For each node *x,* we consider all other nodes *y* which *x* has directed-edges in to, once a node has already been visited and marked, we won't search from it again.

The BFS have a different order for visiting nodes with DFS, it will access to points close to the starting point first. So the big-Oh time complexity is same as DFS.

If the representation of the graph is *an adjacency list,* the big-Oh time complexity for searching an entire graph is $O(n+e)$

If the representation of the graph is *an adjacency matrix,* the big-Oh time complexity for searching an entire graph is $O(n^2)$