Intro To Algo HW #2 - Saaif Ahmed

Friday, February 7, 2020 3:25 PM

Problem 3.9:

The algorithm is as follows:

First run through the adjacency list and assign degree values to each node. Then we construct an array the size of the graph (n).

We then use DFS to construct to move through the graph.

For every node that is unvisited, mark it as visited, go through the neighbors and sum their degree values, and assign back to the "parent" node.

Analysis:

The first step of assigning degree values will be O(n) as it is already given in Adjacency List format.

The use of DFS is, by definition, O(n). Because we account for the neighbors of each node, the complexity becomes O(n+e).

The total summation is O(n) + O(n+e) which is still linear in Big O notation.

Intro To Algo HW #2

Friday, February 7, 2020 3:34 PM

Problem 3.15

A:

We model the mayor's problem as a graph and searching problem. The intersections can be treated as nodes and the one-way streets are directed edges.

Choose a random node and run BFS. If all edges are reached then move on to the next steps, otherwise the mayor is wrong.

Take the graph and invert it. Because this is a directed graph, this means reverse the directions of all edges.

Choose a random node and run BFS on the inverse graph. If all nodes are reached then the mayor is correct.

Analysis:

First we run BFS which is by definition linear. Then we inverse the graph, which will take all the nodes and all their edges and reverse them. This is O(n+e). Then we run BFS again which again is linear.

The total summation is $O(n+e)+O(n+e)+O(n+e)$ which is still linear

B:

We set the intersections as the nodes and the one-way streets as directed edges.

Start at the town hall. Run BFS from the town hall.
Remove any nodes that are unreachable from the graph.
Reverse the direction of all edges in the graph. Run BFS starting from the town hall. If all nodes are reachable then the mayor is correct.

Analysis:

BFS is linear. Removing the nodes is linear. Reversing the direction is linear. The final BFS is linear.

The sum of complexity is $O(n+e)+O(n+e)+O(n+e)+O(n+e)$ which is still linear.

Intro to Algo HW #2

Friday, February 7, 2020 4:00 PM

Problem 3.22

The algorithm is as follows.

Select a node from the graph. Run DFS starting from that node. Store the post DFS numbers.

Find the node with the highest DFS post number. Run DFS starting from this node. If all nodes were reached then that node is a node from which all vertices are accessible.

Analysis:

There are two operations of DFS which is a linear algorithm. This means that total sum of complexity is $O(n) + O(n)$ which is still linear.

Intro to Algo HW #2

Friday, February 7, 2020 4:12 PM

Problem 3.23

The algorithm is as follows:

Store a variable called count.
Run BFS starting from node s until you reach node t, then return in reverse marking each node as "route to t".

Then run DFS from node s. If explore finds a node that has been visited and has been marked "route to t" then increase count by 1.

At the end, count will store the number of paths.

Analysis:
We run one instance of BFS and one instance of DFS which are linear algorithms.
Thus the complexity is $O(n + e) + O(n + e)$ which is linear

Intro to Algo HW #2

Friday, February 7, 2020 4:39 PM

Problem 3.24

The algorithm is as follows.

First we topologically sort the graph and create an array the size of n where the indexes store a count for a respective node in the same topological sorted order.

Started from the beginning of the array sum all the nodes that have an edge to them and increase the count of that node's neighbor by 1.

Run through the array and verify that each index is greater than 1, except for the first node. If they are, then the graph has a directed path that touches each node once.

Analysis:

Topological sort is a linear algorithm. The next step moves through both the nodes and the edges which is $O(n+e)$. The final run through of the array is a $O(n)$ operation.

Thus the total complexity is $O(n+e)+O(n+e)+O(n)$ which is linear.