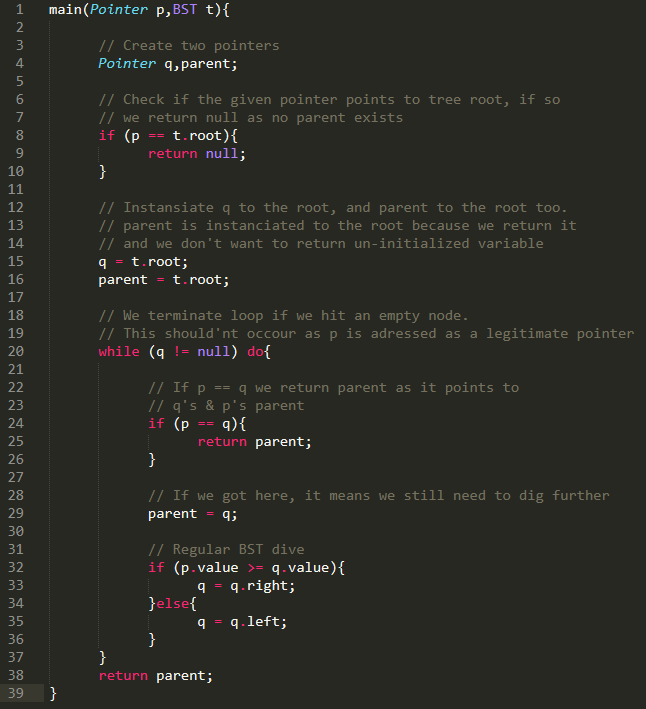
|  |
| --- |
| DATA STRUCTURS, Ex06 |
| Aviad Hahami302188347 |

# Q1.

\*note – This algorithm is correct assuming 2 nodes with equal value will be treated as “bigger than” (i.e. if we have two “8”, then one will be the sub-tree root, and the other will be the right child)



# Q2.

# 

# Q3.

# Q7.

### A.

Let person X be as follows:

Description:

We will use an augmented interval tree as described by *Cormen*.

The tree will be constructed from special nodes.

Each node will contain a person X, the person’s lifespan as the interval and the maximum high value among the tree. Below there’s an example of a node in the tree:

X (Person data)

Life span = [X.death, X.birth]

Max high value = Maximum value of birth in sub-tree

|  |  |  |
| --- | --- | --- |
| Construct tree | -- | O(*n* log *n*) |
| Insert a person X into the data structure | 1. Create a new tree node as formatted above. 2. Insert to tree according to regular interval tree properties. | O(log *n*) |
| Remove a person X from the data structure. | 1. Find person X in tree 2. Remove from tree according to regular interval tree properties | O(log *n*) |
| Given a new person, X, find at least one name of a person Y that lived in the same period. | 1. Search the tree for a corresponding interval according to regular interval tree properties. | O(log *n*) |
| Given a person, X, return another person, Y, that was born in the same month and year as X, if one exits. | Search the tree for a corresponding interval according to regular interval tree properties. | O(log *n*) |
| Give the name of the person that was born first | Since the tree is sorted by lowest key (according to *Cormen*’s specification) we should return the name of the left most child. | O(log *n*) |
| Give the name of the person that passed away last | 1. Return name of the person in the most-right node. | O(log *n*) |

### B.

In order to insert into the tree, we will do the following:

Add a field to each node in the following format:

(4 digits for year, 2 for day and month)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Y | Y | Y | Y | M | M | D | D |
| 2 | 0 | 1 | 5 | 0 | 5 | 2 | 6 |

Insert the nodes into temporary array, and sort them via *Radix* ( O(n) + O(8n) O(n) )

Now, since we know we are dealing with a tree graph, we can use post-order traversal (DFS) in order to insert the nodes to the tree.

The DFS is blocked by O(m) where m is the edges in the graph.

Edges in tree graph are known to be n-1, hence we have

So we’ve concluded this in worst-case time complexity of O(n).