

Forum:[Hack Forums](#)

Thread: [find the majority using binary tree search](#)

Chaos1994 - April 18, 2018, 6:45 am

I have created the all the methods and the nods for the binary search tree but I don't know how to use it to find the majority if any can help that would be great because I'm stuck and search the internet for hints but I could find anything related too mine

```
Code using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace DSA_Homa_Assignment_Structure
{
    /// <summary>
    /// CLASS REPRESENTING A BINARY SEARCH TREE STRUCTURE REQUIRED TO FIND
    THE MAJORITY ELEMENT.
    /// YOU MUST IMPLEMENT THE FOLLOWING METHODS TO COMPLETE SECTION 1 :
    /// - InsertNewNode(Node start, int v) WHERE INDICATED
    /// - getNodeByValue(Node start, int v, int f)
    /// - getMajorityElement(Node start)
    /// </summary>
    class BinarySearchTree
    {
        //The top most node in a binary search tree
        private Node root = null;
        public Node Root { get => root; set => root = value; }

        //The number of nodes in the tree
        private int size = 0;

        /// <summary>
        /// Empty constructor, creates an instance of binary search tree with no nodes
        /// </summary>
        public BinarySearchTree()
        {
            //empty
        }

        /// <summary>
        /// Returns a pointer to a node for a specific value,
        /// if the node is not found, null is returned.
        /// HINT : LOOK FOR HOW TO PERFORM SEARCH ON BINARY TREE!
        /// </summary>
        /// <param name="start">starting piint of tree traversal</param>
        /// <param name="v">value of node to look for</param>
        /// <returns>Node a pointer to the node with the given value</returns>
```

```

public Node getNodeByValue(Node start, int v)
{
    if (start == null || start.Value == v)
    {
        return start;
    }

    if(start.Value < v)
    {
        return getNodeByValue(start.LeftChild, v);
    }

    return getNodeByValue(start.RightChild, v);
    //Replace this with the code for looking for a node in the binary search tree
    throw new NotImplementedException();
}

/// <summary>
/// Calls the Insert new Node method to add a new node in the tree
/// or increase the frequency of an existing node
/// NOTE : NO FURTHER MODIFICATIONS ARE REQUIRED FOR THIS METHOD
/// </summary>
/// <param name="start">Node from which to start traversing</param>
/// <param name="v">value of new node</param>
public void AddNode(Node start, int v)
{
    //Insert the new node in the tree
    InsertNewNode(start, v);
}

/// <summary>
/// Creates a new node using v for value
/// If v is not already in the tree, the node is added with frequency 0
/// If v is already in the tree, the frequency of the node with value v is incremented
/// </summary>
/// <param name="start">Node from which to start traversing to add</param>
/// <param name="v">value of new node</param>
/// <returns>Node parent where new node was added</returns>
public Node InsertNewNode(Node start, int v)
{
    //Add the new node to the tree
    Node newNode = new Node(v);

    //If the tree is empty, at the new node at root
    if (root == null)
    {
        root = new Node(v);
        return root;
    }

    if (start == null)

```

```

{
    start = newNode;
    size++;
    return start;
}

//Otherwise add the new node to the left or right depending on its value
if (v < start.Value)
{
    root.LeftChild = InsertNewNode(root.LeftChild, v);
}
else if (v > start.Value)
{
    root.RightChild = InsertNewNode(root.RightChild, v);
}
else
{
    size++;
    return root;
}

return root;
}

/// <summary>
/// Increases the frequency property of the given node n by 1
/// NOTE : NO FURTHER MODIFICATIONS ARE REQUIRED FOR THIS METHOD
/// </summary>
/// <param name="n">Node to increase frequency</param>
public void IncrementNodeFrequency(Node n)
{
    //If the node is not null
    if (n != null)
    {
        //Increase the frequency property for that node by 1
        n.Frequency = n.Frequency + 1;
    }
}

/// <summary>
/// Traverses the Binary Search Tree and
/// returns the value of the node if the frequency is greater than n/2
/// </summary>
/// <param name="start">node from which to start looking for majority element</param>
/// <returns>int value of node which represents majority element</returns>
public int getMajorityElement(Node start)
{
    if(start == null)
    {

```

```

    }

    //Replace this with the code for looking for the majority element in the binary search tree
    throw new NotImplementedException();

    //HINT ALGORITHM IS AS FOLLOWS
    /*
        IF the start is not empty
            check if the value of the start node occurs enough times to qualify as the majority
element
            if yes,
                return the start value
            if no,
                return the majority element when starting from the left side of the tree (Hint :
recursive call)
                return the majority element when starting from the right side of the tree (Hint :
recursive call)
            END IF

        ELSE
            return -1
        END ELSE

    */
}

}
}
Thanks advanced

```

alexconnolly - April 24, 2018, 3:51 am

Something says you didn't.

Looks like it's pre-made code which you used and are now asking us to do your assignments for you.

Anyway, look into using a Dictionary to represent the object and it's count. Then, at the end, sort your dictionary descending on occurrences and there you go.

Google wont fix your problem because it requires thought rather than a Google search. IT requires some kind of understanding of programming, how your code is working, how to write code etc.

Chaos1994 - May 4, 2018, 8:06 am

(Apr 24, 2018 03:51 AM)alexconnolly Wrote: [\[CITE\]](#) Something says you didn't.

Looks like it's pre-made code which you used and are now asking us to do your assignments for you.

Anyway, look into using a Dictionary to represent the object and it's count. Then, at the end, sort your dictionary descending on occurrences and there you go.

Google wont fix your problem because it requires thought rather than a Google search. IT requires some kind of understanding of programming, how your code is working, how to write code etc.

yes but the methods where empty and we needed to filled them up which I did

Dalvik - *May 4, 2018, 9:00 am*

What is the goal of this tree? Are you using it for a specific algorithm (ie. Huffman) or something?