Thời gian còn lại 0:08:44

Câu hỏi 12

Chính xác

Điểm 1,00 của 1,00

In this question, you have to perform add **and delete on binary search tree**. Note that:

- When deleting a node which still have 2 children, **take the inorder successor** (smallest node of the right sub tree of that node) to replace it.

- When adding a node which has the same value as parent node, add it in the **left sub tree**.

Your task is to implement two functions: add and deleteNode. You could define one or more functions to achieve this task.

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
#define SEPARATOR "#<ab@17943918#@>#"
template<class T>
class BinarySearchTree
{
public:
    class Node;
private:
    Node* root;
public:
    BinarySearchTree() : root(nullptr) {}
    ~BinarySearchTree()
    {
        // You have to delete all Nodes in BinaryTree. However in this task, you can ignore it.
    }

    //Helping function

    void add(T value){
        //TODO
    }

    void deleteNode(T value){
        //TODO
    }
    string inOrderRec(Node* root) {
        stringstream ss;
        if (root != nullptr) {
            ss << inOrderRec(root->pLeft);
            ss << root->value << " ";
            ss << inOrderRec(root->pRight);
        }
        return ss.str();
    }

    string inOrder(){
        return inOrderRec(this->root);
    }

    class Node
    {
    private:
        T value;
        Node* pLeft, * pRight;
        friend class BinarySearchTree<T>;
    public:
        Node(T value) : value(value), pLeft(NULL), pRight(NULL) {}
        ~Node() {}
    };
};
```

For example:

Test	Result

BinarySearchTree<int> bst;
bst.add(9);
bst.add(2);
bst.add(10);
bst.deleteNode(9);
cout << bst.inOrder();
	2 10

Test	Result

BinarySearchTree<int> bst;
bst.add(9);
bst.add(2);
bst.add(10);
bst.add(8);
cout << bst.inOrder()<<endl;
bst.add(11);
bst.deleteNode(9);
cout << bst.inOrder();
	2 8 9 10
	2 8 10 11

Answer: (penalty regime: 0 %)

Reset answer

```
void add(T value) {
        root = addRec(root, value);
}
Node* findMin(Node* node) {
    Node* current = node;
    while (current && current->pLeft) {
        current = current->pLeft;
    }
    return current;
}
Node* addRec(Node* root, T value) {
    if (root == nullptr) {
        return new Node(value);
    }

    if (value < root->value) {
        root->pLeft = addRec(root->pLeft, value);
    } else if (value == root->value) {
        root->pLeft = addRec(root->pLeft, value);
    } else {
        root->pRight = addRec(root->pRight, value);
    }

    return root;
}
Node* deleteNodeRec(Node* root, T value) {
    if (root == nullptr) {
        return root;
    }
    if (value < root->value) {
        root->pLeft = deleteNodeRec(root->pLeft, value);
    } else if (value > root->value) {
        root->pRight = deleteNodeRec(root->pRight, value);
    } else {
        if (root->pLeft == nullptr) {
            Node* temp = root->pRight;
            delete root;
            return temp;
        } else if (root->pRight == nullptr) {
            Node* temp = root->pLeft;
            delete root;
            return temp;
        }

        Node* temp = findMin(root->pRight);
        root->value = temp->value;
        root->pRight = deleteNodeRec(root->pRight, temp->value);
    }

    return root;
}
void deleteNode(T value) {
    root = deleteNodeRec(root, value);
}
```

Precheck | Kiểm tra

	Test	Expected	Got	

✔	`BinarySearchTree<int> bst;`
	`bst.add(9);`
	`bst.add(2);`
	`bst.add(10);`
	`bst.deleteNode(9);`
	`cout << bst.inOrder();`
		`2 10`	`2 10`	✔

✔	`BinarySearchTree<int> bst;`
	`bst.add(9);`
	`bst.add(2);`
	`bst.add(10);`
	`bst.add(8);`
	`cout << bst.inOrder()<<endl;`
	`bst.add(11);`
	`bst.deleteNode(9);`
	`cout << bst.inOrder();`
		`2 8 9 10`
		`2 8 10 11`
			`2 8 9 10`
			`2 8 10 11`
				✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 13

Chính xác

Điểm 1,00 của 1,00

Class `BSTNode` is used to store a node in binary search tree, described on the following:

```
class BSTNode {
public:
    int val;
    BSTNode *left;
    BSTNode *right;
    BSTNode() {
        this->left = this->right = nullptr;
    }
    BSTNode(int val) {
        this->val = val;
        this->left = this->right = nullptr;
    }
    BSTNode(int val, BSTNode*& left, BSTNode*& right) {
        this->val = val;
        this->left = left;
        this->right = right;
    }
};
```

Where `val` is the value of node, `left` and `right` are the pointers to the left node and right node of it, respectively. If a repeated value is inserted to the tree, it will be inserted to the left subtree.

Also, a static method named `createBSTree` is used to create the binary search tree, by iterating the argument array left-to-right and repeatedly calling `addNode` method on the root node to insert the value into the correct position. For example:

```
int arr[] = {0, 10, 20, 30};
auto root = BSTNode::createBSTree(arr, arr + 4);
```

is equivalent to

```
auto root = new BSTNode(0);
root->addNode(10);
root->addNode(20);
root->addNode(30);
```

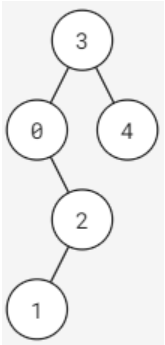
Request: Implement function:

```
vector<int> levelAlterTraverse(BSTNode* root);
```

Where `root` is the root node of given binary search tree (this tree has between 0 and 100000 elements). This function returns the values of the nodes in each level, alternating from going left-to-right and right-to-left..

Example:

Given a binary search tree in the following:



In the first level, we should traverse from left to right (order: `3`) and in the second level, we traverse from right to left (order: `4, 0`). After traversing all the nodes, the result should be `[3, 4, 0, 2, 1]`.

Note: In this exercise, the libraries `iostream`, `vector`, `stack`, `queue`, `algorithm` and `using namespace std` are used. You can write helper functions; however, you are not allowed to use other libraries.

For example:

Test	Result

`int arr[] = {0, 3, 5, 1, 2, 4};`
`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
`printVector(levelAlterTraverse(root));`
`BSTNode::deleteTree(root);`
	`[0, 3, 1, 5, 4, 2]`

Answer: (penalty regime: 0 %)

Reset answer

```
vector<int> levelAlterTraverse(BSTNode* root) {
    // STUDENT ANSWER
    vector<int> result;
    if (root == nullptr) {
        return result;
    }
    queue<BSTNode*> q;
    q.push(root);
    bool leftToRight = true;
    while (!q.empty()) {
        int levelSize = q.size();
        vector<int> levelNodes;
        for (int i = 0; i < levelSize; i++) {
            BSTNode* current = q.front();
            q.pop();
            if (leftToRight) {
                levelNodes.push_back(current->val);
            } else {
                levelNodes.insert(levelNodes.begin(), current->val);
            }

            if (current->left) {
```

Precheck Kiểm tra

	Test	Expected	Got	

✔	`int arr[] = {0, 3, 5, 1, 2, 4};`
	`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
	`printVector(levelAlterTraverse(root));`
	`BSTNode::deleteTree(root);`
		`[0, 3, 1, 5, 4, 2]`	`[0, 3, 1, 5, 4, 2]`	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 14

Chính xác

Điểm 1,00 của 1,00

Class `BSTNode` is used to store a node in binary search tree, described on the following:

```
class BSTNode {
public:
    int val;
    BSTNode *left;
    BSTNode *right;
    BSTNode() {
        this->left = this->right = nullptr;
    }
    BSTNode(int val) {
        this->val = val;
        this->left = this->right = nullptr;
    }
    BSTNode(int val, BSTNode*& left, BSTNode*& right) {
        this->val = val;
        this->left = left;
        this->right = right;
    }
};
```

Where `val` is the value of node, `left` and `right` are the pointers to the left node and right node of it, respectively. If a repeated value is inserted to the tree, it will be inserted to the left subtree.

Also, a static method named `createBSTree` is used to create the binary search tree, by iterating the argument array left-to-right and repeatedly calling `addNode` method on the root node to insert the value into the correct position. For example:

```
int arr[] = {0, 10, 20, 30};
auto root = BSTNode::createBSTree(arr, arr + 4);
```

is equivalent to

```
auto root = new BSTNode(0);
root->addNode(10);
root->addNode(20);
root->addNode(30);
```

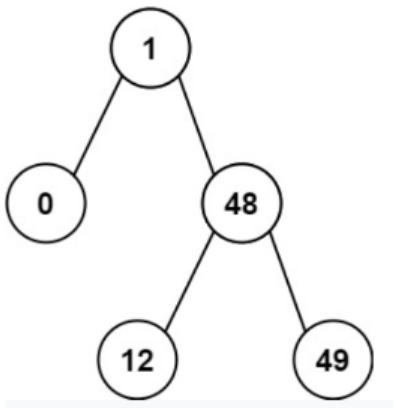
Request: Implement function:

```
int kthSmallest(BSTNode* root, int k);
```

Where `root` is the root node of given binary search tree (this tree has `n` elements) and `k` satisfy: `1 <= k <= n <= 100000`. This function returns the `k`-th smallest value in the tree.

Example:

Given a binary search tree in the following:



With `k = 2`, the result should be `1`.

Note: In this exercise, the libraries `iostream`, `vector`, `stack`, `queue`, `algorithm`, `climits` and `using namespace std` are used. You can write helper functions; however, you are not allowed to use other libraries.

For example:

Test	Result

`int arr[] = {6, 9, 2, 13, 0, 20};`
`int k = 2;`
`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
`cout << kthSmallest(root, k);`
`BSTNode::deleteTree(root);`
	2

Answer: (penalty regime: 0 %)

Reset answer

```
void kthSmallestUtil(BSTNode* root, int& k, int& result) {
    if (root == nullptr || k == 0) {
        return;
    }
    kthSmallestUtil(root->left, k, result);
    k--;
    if (k == 0) {
        result = root->val;
        return;
    }
    kthSmallestUtil(root->right, k, result);
}
int kthSmallest(BSTNode* root, int k) {
    // STUDENT ANSWER
    int result = -1;
    kthSmallestUtil(root, k, result);
    return result;
}
```

Precheck Kiểm tra

	Test	Expected	Got	

✔	`int arr[] = {6, 9, 2, 13, 0, 20};`
	`int k = 2;`
	`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
	`cout << kthSmallest(root, k);`
	`BSTNode::deleteTree(root);`
		2	2	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 15

Chính xác

Điểm 1,00 của 1,00

Class **BTNode** is used to store a node in binary search tree, described on the following:

```
class BTNode {
    public:
        int val;
        BTNode *left;
        BTNode *right;
        BTNode() {
            this->left = this->right = NULL;
        }
        BTNode(int val) {
            this->val = val;
            this->left = this->right = NULL;
        }
        BTNode(int val, BTNode*& left, BTNode*& right) {
            this->val = val;
            this->left = left;
            this->right = right;
        }
};
```

Where `val` is the value of node (non-negative integer), `left` and `right` are the pointers to the left node and right node of it, respectively.

Also, a static method named `createBSTree` is used to create the binary search tree, by iterating the argument array left-to-right and repeatedly calling `addNode` method on the root node to insert the value into the correct position. For example:

```
int arr[] = {0, 10, 20, 30};
auto root = BSTNode::createBSTree(arr, arr + 4);
```

is equivalent to

```
auto root = new BSTNode(0);
root->addNode(10);
root->addNode(20);
root->addNode(30);
```

Request: Implement function:

`int rangeCount(BTNode* root, int lo, int hi);`

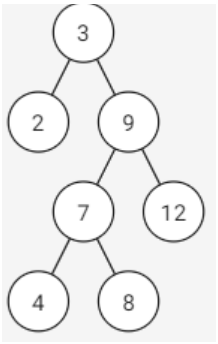
Where `root` is the root node of given binary search tree (this tree has between 0 and 100000 elements), `lo` and `hi` are 2 positives integer and `lo ≤ hi`. This function returns the number of all nodes whose values are between `[lo, hi]` in this binary search tree.

More information:

- If a node has `val` which is equal to its ancestor's, it is in the right subtree of its ancestor.

Example:

Given a binary search tree in the following:



With `lo=5, hi=10`, all the nodes satisfied are node `9, 7, 8`; there fore, the result is `3`.

Note: In this exercise, the libraries `iostream`, `stack`, `queue`, `utility` and `using namespace std` are used. You can write helper functions; however, you are not allowed to use other libraries.

For example:

Test	Result

`int value[] = {3,2,9,7,12,4,8};`
`int lo = 5, hi = 10;`
`BTNode* root = BTNode::createBSTree(value, value + sizeof(value)/sizeof(int));`
`cout << rangeCount(root, lo, hi);`
	3

`int value[] = {1167,2381,577,2568,124,1519,234,1679,2696,2359};`
`int lo = 500, hi = 2000;`
`BTNode* root = BTNode::createBSTree(value, value + sizeof(value)/sizeof(int));`
`cout << rangeCount(root, lo, hi);`
	4

Answer: (penalty regime: 0 %)

Reset answer

```
int rangeCount(BTNode* root, int lo, int hi) {
    if (!root) return 0;
    int count = 0;
    if (root->val >= lo && root->val <= hi) {
        count++;
    }
    count += rangeCount(root->right, lo, hi);
    count += rangeCount(root->left, lo, hi);
    return count;
}
```

Precheck Kiểm tra

	Test	Expected	Got	

✔	`int value[] = {3,2,9,7,12,4,8};`
	`int lo = 5, hi = 10;`
	`BTNode* root = BTNode::createBSTree(value, value + sizeof(value)/sizeof(int));`
	`cout << rangeCount(root, lo, hi);`
		3	3	✔

✔	`int value[] = {1167,2381,577,2568,124,1519,234,1679,2696,2359};`
	`int lo = 500, hi = 2000;`
	`BTNode* root = BTNode::createBSTree(value, value + sizeof(value)/sizeof(int));`
	`cout << rangeCount(root, lo, hi);`
		4	4	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 16

Chính xác

Điểm 1,00 của 1,00

Class `BSTNode` is used to store a node in binary search tree, described on the following:

```
class BSTNode {
public:
    int val;
    BSTNode *left;
    BSTNode *right;
    BSTNode() {
        this->left = this->right = nullptr;
    }
    BSTNode(int val) {
        this->val = val;
        this->left = this->right = nullptr;
    }
    BSTNode(int val, BSTNode*& left, BSTNode*& right) {
        this->val = val;
        this->left = left;
        this->right = right;
    }
};
```

Where `val` is the value of node, `left` and `right` are the pointers to the left node and right node of it, respectively. If a repeated value is inserted to the tree, it will be inserted to the left subtree.

Also, a static method named `createBSTree` is used to create the binary search tree, by iterating the argument array left-to-right and repeatedly calling `addNode` method on the root node to insert the value into the correct position. For example:

```
int arr[] = {0, 10, 20, 30};
auto root = BSTNode::createBSTree(arr, arr + 4);
```

is equivalent to

```
auto root = new BSTNode(0);
root->addNode(10);
root->addNode(20);
root->addNode(30);
```

Request: Implement function:

```
int singleChild(BSTNode* root);
```

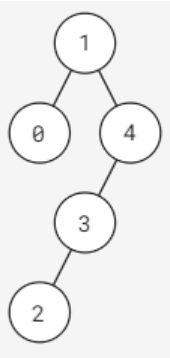
Where `root` is the root node of given binary search tree (this tree has between 0 and 100000 elements). This function returns the number of single children in the tree.

More information:

- A node is called a **single child** if its parent has only one child.

Example:

Given a binary search tree in the following:



There are 2 single children: node 2 and node 3.

Note: In this exercise, the libraries `iostream` and `using namespace std` are used. You can write helper functions; however, you are not allowed to use other libraries.

For example:

Test	Result

`int arr[] = {0, 3, 5, 1, 2, 4};`
`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
`cout << singleChild(root);`
`BSTNode::deleteTree(root);`
	3

Answer: (penalty regime: 0 %)

Reset answer

```
int singleChild(BSTNode* root) {
    // STUDENT ANSWER
    if (!root) return 0;
    int count = 0;
    if ((root->left == nullptr && root->right != nullptr) || (root->left != nullptr && root->right == null
        count++;
    count += singleChild(root->left);
    count += singleChild(root->right);
    return count;
}
```

Precheck Kiểm tra

	Test	Expected	Got	

✔	`int arr[] = {0, 3, 5, 1, 2, 4};`
	`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
	`cout << singleChild(root);`
	`BSTNode::deleteTree(root);`
		3	3	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 17

Chính xác

Điểm 1,00 của 1,00

Class `BSTNode` is used to store a node in binary search tree, described on the following:

```
class BSTNode {
public:
    int val;
    BSTNode *left;
    BSTNode *right;
    BSTNode() {
        this->left = this->right = nullptr;
    }
    BSTNode(int val) {
        this->val = val;
        this->left = this->right = nullptr;
    }
    BSTNode(int val, BSTNode*& left, BSTNode*& right) {
        this->val = val;
        this->left = left;
        this->right = right;
    }
};
```

Where `val` is the value of node, `left` and `right` are the pointers to the left node and right node of it, respectively. If a repeated value is inserted to the tree, it will be inserted to the left subtree.

Also, a static method named `createBSTree` is used to create the binary search tree, by iterating the argument array left-to-right and repeatedly calling `addNode` method on the root node to insert the value into the correct position. For example:

```
int arr[] = {0, 10, 20, 30};
auto root = BSTNode::createBSTree(arr, arr + 4);
```

is equivalent to

```
auto root = new BSTNode(0);
root->addNode(10);
root->addNode(20);
root->addNode(30);
```

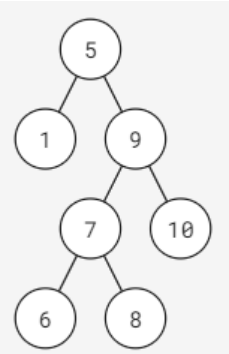
Request: Implement function:

```
BSTNode* subtreeWithRange(BSTNode* root, int lo, int hi);
```

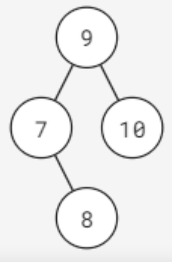
Where `root` is the root node of given binary search tree (this tree has between 0 and 100000 elements). This function returns the binary search tree after deleting all nodes whose values are outside the range `[lo, hi]` (inclusive).

Example:

Given a binary search tree in the following:



With `lo = 7` and `hi = 10`, the result should be:



Note: In this exercise, the libraries `iostream` and `using namespace std` are used. You can write helper functions; however, you are not allowed to use other libraries.

For example:

Test	Result

`int arr[] = {0, 3, 5, 1, 2, 4};`
`int lo = 1, hi = 3;`
`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
`root = subtreeWithRange(root, lo, hi);`
`BSTNode::printPreorder(root);`
`BSTNode::deleteTree(root);`
	`3 1 2`

Answer: (penalty regime: 0 %)

Reset answer

```
BSTNode* subtreeWithRange(BSTNode* root, int lo, int hi) {
    // STUDENT ANSWER
    if (!root)
        return nullptr;
    if (root->val < lo)
        return subtreeWithRange(root->right, lo, hi);
    if (root->val > hi)
        return subtreeWithRange(root->left, lo, hi);
    root->left = subtreeWithRange(root->left, lo,hi);
    root->right = subtreeWithRange(root->right, lo,hi);
    return root;
}
```

Precheck Kiểm tra

	Test	Expected	Got	

✔	`int arr[] = {0, 3, 5, 1, 2, 4};`
	`int lo = 1, hi = 3;`
	`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
	`root = subtreeWithRange(root, lo, hi);`
	`BSTNode::printPreorder(root);`
	`BSTNode::deleteTree(root);`
		`3 1 2`	`3 1 2`	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

BÁCH KHOA E-LEARNING

WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

(028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

elearning@hcmut.edu.vn