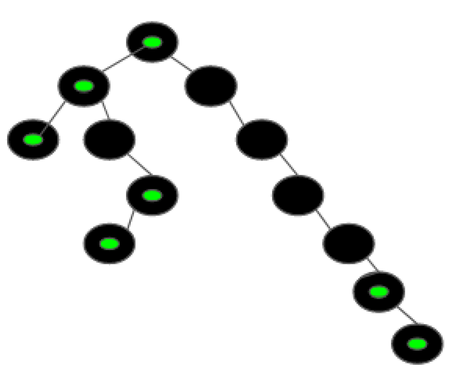
Профил на дърво

locked

Problem | Submissions | Leaderboard | Discussions

Напишете функция, която намира как изглежда едно двоично дърво за търсене погледнато отляво. При гледане на дървото отляво виждаме за всяко ниво на дървото най-левият възел. т.е се вижда корена, най-лявата част от най-левият клон и след това на всяко следващо ниво най-левият възел на дървото (това може да са възли от различни клонове на дървото). На примера отдолу са отблязани с зелена точка възлите които се виждат като ляв профил на дървото.



Input Format

Реализирайте функцията, която е в темплейта.

Constraints

възлите на дървото ще са по-малко от 100,000

Output Format

принтирайте числата във възлите започвайки последователно от корена и стигайки до най-дълбокия възел виждан от ляво.

Sample Input 0

```
5
2 0 5 8 3
```

Sample Output 0

```
2 0 3
```

Current Buffer (saved locally, editable) C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <map>
using namespace std;

struct Node {
    Node *left;
    Node *right;
    int value;
    Node(int value) {
        this->value = value;
        this->left = NULL;
        this->right = NULL;
    }
};

class BST {

public:
    BST() {
        root = NULL;
    }

    void insert(int value) {
        root = insert(root, value);
    }

    void printLeftProfile() {
      map<int,int> length;

      preorder(root,length,0);
      for(auto it:length){
        cout<<it.second<<" ";
      }
    }

private:
    void preorder(Node* current,map<int,int> &length,int a){

      if(current!=nullptr){
        if(length.find(a)==length.end()){
          length.insert({a,current->value});
        }


        preorder(current->left,length,a+1);
        preorder(current->right,length,a+1);
      }

    }
    Node* root;

    Node* insert(Node *curNode, int value) {
        if (curNode == NULL) {
            curNode = new Node(value);
        } else if (curNode->value < value) {
            curNode->right = insert(curNode->right, value);
        } else if (curNode->value > value) {
            curNode->left = insert(curNode->left, value);
        } else {
            //if we already have this value at the tree - do nothing
        }
        return curNode;
    }
};

int main() {
    int n;
    cin >> n;
    int value;
    BST tree;
    for(int i = 0 ; i < n; i++) {
        cin >> value;
        tree.insert(value);
    }
    tree.printLeftProfile();
    return 0;
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code