МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №3

З предмету «Алгоритми та структури даних»

Виконав Студент групи КН-36а

Рубан Ю.Д.

Перевірила:

ас. Бородіна І. О.

Харків 2017

БАЗОВІ СТРУКТУРИ ДАНИХ: ЧЕРВОНО-ЧОРНІ ДЕРЕВА

Мета роботи: познайомитися з червоно-чорними деревами та отримати навички програмування алгоритмів, що їх обробляють.

Завдання

Розробити програму, яка читає з клавіатури числа N, M (1 < N, M < 256); послідовність N ключів (цілих, дійсних чисел або рядків (до 255 символів) в залежності від варіанту завдання); послідовність M ключів. Програма берігає

першу послідовність до червоно-чорного дерева. Кожного разу, коли до дерева додається новий елемент, потрібно вивести статистику

Максимальний елемент та його колір.

Після побудови дерева для кожного елементу x другої послідовності потрібно вивести результати наступних операцій над деревом

Successor(x) та його колір.

Використовувати готові реалізації структур даних (наприклад, STL) заборонено, але можна використати реалізацію рядків (наприклад, std::string у C++).

Хід виконання роботи

Розроблено програму, яка читає з клавіатури числа N, M (1 < N, M < 256); послідовність N ключів (цілих, дійсних чисел або рядків (до 255 символів) в залежності від варіанту завдання); послідовність M ключів. Програма зберігає першу послідовність до червоно-чорного дерева. Кожного разу, коли до дерева додається новий елемент, потрібно вивести статистику

Максимальний елемент та його колір.

Після побудови дерева для кожного елементу x другої послідовності виводяться результати наступних операцій над деревом

Successor(x) та його колір.

Використовувати готові реалізації структур даних (наприклад, STL) заборонено, але можна використати реалізацію рядків (наприклад, std::string у C++).

```
#include<iostream>
using namespace std;
class Tree
{
private:
	Tree *right = 0;
	Tree *left = 0;
	char color = 'b';
	int data = 0;
	Tree *parent = 0;
	int count = 0;
	Tree* getMin(Tree* root)
	{
		Tree *temp = root;
		while (temp->left != NULL)
		{
			temp = temp->left;
		}
		return temp;
	}
public:
	int getCount() { return count; }
	void add(int val)
	{
		Tree *temp = new Tree;
		temp->data = val;
		Tree *head = parent;
		//temp->parent = head;
		if (this->parent == 0)
		{
			temp->color = 'b';
			parent = temp;
		}
		else if (head->data < val)//для добавления элементов в правую часть дерева
		{
			while (head->data < val && head->right != NULL)
			{
				head = head->right;
			}
			while (head->data > val && head->left != NULL)
			{
				head = head->left;
			}
			if (head->data < val)
			{
				temp->parent = head;
				head->right = temp;
				if (head->color == 'b')
```

```
                        {
                                head->right->color = 'r';
                        }
                        else
                        {
                                head->right->color = 'b';
                        }
                }
                else
                {
                        temp->parent = head;
                        head->left = temp;
                        if (head->color == 'b')
                        {
                                head->left->color = 'r';
                        }
                        else
                        {
                                head->left->color = 'b';
                        }
                }

        }
        else//для добавления элементов в левую часть дерева
        {
                while (head->data > val && head->left != NULL)
                {
                        head = head->left;
                }
                while (head->data < val && head->right != NULL)
                {
                        head = head->right;
                }
                if (head->data < val)
                {
                        temp->parent = head;
                        head->right = temp;
                        if (head->color == 'b')
                        {
                                head->right->color = 'r';
                        }
                        else
                        {
                                head->right->color = 'b';
                        }
                }
                else
                {
                        temp->parent = head;
                        head->left = temp;
                        if (head->color == 'b')
                        {
                                head->left->color = 'r';
                        }
                        else
                        {
                                head->left->color = 'b';
                        }
                }
        }
        count++;
}
int getMax()
{
        Tree *temp = parent;
        while (temp->right != NULL)
        {
                temp = temp->right;
        }
        return temp->data;
}
char getColorOfMax()
{
        Tree *temp = parent;
        while (temp->right != NULL)
        {
                temp = temp->right;
```

```
                }
                return temp->color;
        }
        Tree* search(int i)
        {
                Tree*temp = parent;
                while (temp != NULL && i != temp->data)
                {
                        if (i < temp->data)
                        {
                                temp = temp->left;
                        }
                        else
                        {
                                temp = temp->right;
                        }
                }
                return temp;
        }
        Tree* suc(int x)
        {
                if (x >= this->getMax()) { throw; }
                Tree*root = search(x);
                if (root->right != NULL) { return getMin(root->right); }
                Tree* y = root->parent;
                while (y != NULL && root == y->right)
                {
                        root = y;
                        y = y->parent;
                }
                return y;
        }
        int getData()
        {
                return this->data;
        }
        int getColor()
        {
                return this->color;
        }
};

int main()
{
        Tree t;
        int n, m;
        cout << "enter n, m" << endl;
        cin >> n >> m;
        int val;
        cout << "enter elemnts" << endl;
        for (int i = 0; i < n; i++)
        {
                cin >> val;
                t.add(val);
        }
        cout << endl;
        cout << "enter keys" << endl;
        for (int i = 0; i < m; i++)
        {
                cin >> val;
                cout << "successor(" << val << ") = ";
                cout << t.suc(val)->getData() << ", color " << t.suc(val)->getColor() << endl;
        }
        cout << "max elem = " << t.getMax() << ", color = " << t.getColorOfMax() << endl;
        system("pause");
        return 0;
        }
```

Висновки

У даній лабораторній роботі я познайомився з червоно-чорними дерева. Навчився працювати з ними, а також вивчив необхідні алгоритми для цього.