МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Алгоритмы и структуры данных»

Тема: Бинарные деревья поиска

Студентка гр. 7383 _____ Ханова Ю.А.

Преподаватель _____ Размочаева Н.В

Санкт-Петербург

2018

Содержание

Цель работы

Познакомиться с бинарными деревьями поиска и научиться реализовывать их на языке программирования C++.

Вариант 20:

По заданному файлу, все элементы которого различны, построить рандомизированную пирамиду поиска;

Выполнить над пирамидой действие расщепление;

Реализация задачи

В данной лабораторной работе была реализована структура node (основная структура), nodes (структура для возврата результата расщепления) и набор функций для работы с рандомизированной пирамидой поиска:

nodes* split(node* t, int k); - выполняет расщепление бинарного дерева по заданному ключу k;

node* rotateright(node* p); - правый поворот вокруг элемента;

node* rotateleft(node* q); - левый поворот вокруг элемента;

node* insert(int key, node* root); - вставка элемента;

void printPriority(node* root); - печать случайных приоритетов, распределенных к ключам;

node* add(node* p, int el); - добавление элемента el;

void printtree(node* treenode, int l); - вывод пирамиды;

Тестирование

Программа собрана в операционной системе Ubuntu 17.04 с использованием компилятора g++. В других ОС и компиляторах тестирование не проводилось. Результаты тестирования показали, что поставленная цель выполнена. Результаты тестирования представлены в Приложении Б.

Выводы

В ходе выполнения данной лабораторной работы были освоены основные принципы работы с бинарными деревьями поиска (конкретно: рандомизированная пирамида поиска) на языке программирования C++. Также была написана программа для выполнения поставленной задачи.

ПРИЛОЖЕНИЕ А. Код программы

Treap.cpp

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <fstream>
#include <iomanip>

using namespace std;

long long int maxc = pow(2, 32);

struct node {
    int key;
    long long prior;
    node* left;
    node* right;
    node(int k) {
        key = k;
        left = right = NULL;
        prior = rand()%maxc; // рандомные числа от 0 до
2^32
    }
};
struct nodes{//Структура для возврата результата
расщепления
    node* t1;
    node* t2;
    nodes(node* tr1, node* tr2) {
        t1 = tr1;
          t2 = tr2;
    }
};

nodes* split(node* t, int k){
    nodes* ts;
    if(!t){
          return (new nodes(NULL, NULL));
    }
    else if(k > t->key){
          ts = split(t->right, k);
```

```
            t->right = ts->t1;
            return (new nodes(t, ts->t2));
        }
        else{
            ts = split(t->left, k);
            t->left = ts->t2;
            return (new nodes(ts->t1, t));
        }
    }


node* rotateright(node* p) { // правый поворот вокруг
узла p
    node* q = p->left;
    if( !q )
        return p;
    p->left = q->right;
    q->right = p;
    return q;
}

node* rotateleft(node* q) { // левый поворот вокруг узла
q
    node* p = q->right;
    if( !p )
        return q;
    q->right = p->left;
    p->left = q;
    return p;
}


node* insert(int key, node* root) { // вставка
    if(!root) {
        node* p = new node(key);
        return (p);
    }
    if(key <= root->key)
    {
        root->left = insert(key, root->left);
        if(root->left->prior < root->prior)
            root = rotateright(root);
    }
    else {
```

```
        root->right = insert(key, root->right);
        if(root->right->prior < root->prior)
            root = rotateleft(root);
    }
    return root;
}

void Delete(node* p) {
    if(p==NULL)
        return;
    Delete(p->left);
    Delete(p->right);
    delete p;
}

void printPriority(node* root) {
    if (!root)
        return;
    cout<<"Приоритет ключа ["<<setw(5)<<right<< root-
>key <<"] - "<<setw(11)<<right<<root->prior<<endl;
    printPriority(root->right);
    printPriority(root->left);
}

node* find( node* tree, int key) {
    if(!tree)
        return NULL;
    if(key == tree->key)
        return tree;
    if(key < tree->key)
        return find(tree->left, key);
    else
        return find(tree->right, key);
}

node* add(node* p, int el) {
    if (find(p,el)) {
        cout << "Ключ ["<< el <<"] повторяется"<<endl;
    return p;
    }
    else {
        p=insert(el, p);
        cout<<"приоритет нового ключа ["<< (find(p,el))-
>key <<"] - "<<(find(p,el))->prior<<endl;
```

```
        return p;
    }
}

void printtree(node* treenode, int l) {
    if(treenode==NULL) {
        for(int i = 0; i<l; ++i)
            cout<<"\t";
        cout<<'#'<<endl;
        return;
    }
    printtree(treenode->right, l+1);
    for(int i = 0; i < l; i++)
        cout << "\t";
    cout << treenode->key<< endl;
    printtree(treenode->left,l+1);
}

int main() {
    node* treap = NULL;
    nodes* ts = NULL;// результат расщипления
 // пирамида поиска
    int c, el=0;
    string str;
    bool ex = 1;
    char ch;
    while(ex) {
      cout<<"1 - ввод с клавиатуры"<<endl;
      cout<<"2 - ввод из файла input.txt"<<endl;
      cout<<"0 - выход из программы"<<endl;
      cout<<"Введите номер действия:"<<endl;
        cin >> ch;
        getchar();
        switch (ch) {
        case '2': {

            ifstream infile("input.txt");
            if(!infile) {
                cout<<"Файл не открыт!"<<endl;
                continue;
            }
            getline(infile, str);
            break;
```

```
        }
        case '1': {
            cout<<"Введите ключи в строку:"<<endl;
            getline(cin, str);
            break;
        }
        case '0': {
            ex = 0;
            return 0;
        }
        default: {
            cout<<"Некорректные данные!"<<endl;
            return 0;
        }
      }

        char* arr = new char[str.size()+1];
        strcpy(arr, str.c_str());
        char* tok;
        tok = strtok(arr, " ");
        while(tok != NULL) {
            c = atoi(tok);
          if(isalpha(*tok)) {
            cout<<"Некорректные данные!"<<endl;
                return 0;
        }
        if (find(treap,c)) {
                cout << "Ключ [" << c << "]
повторяется"<<endl; // повторение ключа не допустимо, тк
должен быть уникальным
                tok = strtok(NULL, " ");
                continue;
            }
            treap = insert(c, treap);
            tok = strtok(NULL, " ");
        }
        printPriority(treap); // печать приоритетов
        cout<<endl;
        cout<<"------------------------------"<<endl;
            printtree(treap,0);
            cout << "Введите ключ для расщипления:" <<
endl;
            cin >> el;
            if(!find(treap, el)){
```

```
                cout << "Ключ для расщипления не найден!"
<< endl;
                return 0;
              }
                       cout<<"----------------------------
---"<<endl;
                       cout<<"------------SPLIT-----------
---"<<endl;
              ts = split(treap, el);
              printtree(ts->t1,0);
              cout<<"------------------------------
"<<endl;
               printtree(ts->t2,0);
              str.clear();
              delete tok;
              delete[] arr;

      }
}
```

ПРИЛОЖЕНИЕ Б. Тестовые случаи

Тест 1:

```
1 2 3 4 5 6
Приоритет ключа [    6] -   424238335
Приоритет ключа [    2] -   846930886
Приоритет ключа [    3] -  1681692777
Приоритет ключа [    4] -  1714636915
Приоритет ключа [    5] -  1957747793
Приоритет ключа [    1] -  1804289383
--------------------------------
        #
6
                                    #
                            5
                                    #
                    4
                            #
            3
                    #
        2
                    #
            1
                    #
Введите ключ для расщепления:
4
--------------------------------
------------SPLIT--------------
            #
        3
            #
2
            #
        1
            #
--------------------------------
        #
6
                    #
            5
                    #
        4
            #
```

Тест 2:

34 5 6 76 87 23 7 90

Ключ [5] повторяется

Ключ [6] повторяется

Приоритет ключа [6] - 424238335

Приоритет ключа [87] - 596516649

Приоритет ключа [90] - 1350490027

Приоритет ключа [34] - 719885386

Приоритет ключа [76] - 1649760492

Приоритет ключа [7] - 1025202362

Приоритет ключа [23] - 1189641421

Приоритет ключа [4] - 1714636915

Приоритет ключа [5] - 1957747793

```
------------------------------
                        #
                90
                        #
        87
                                #
                        76
                                #
                34
                                        #
                                23
                                        #
                        7
                                #
6
                        #
                5
                        #
        4
                #
```

Введите ключ для расщепления:

76

```
------------------------------
------------SPLIT--------------
                #
        34
                                #
                        23
                                #
                7
                        #
```

6

#

5

#

4

#

#

90

#

87

#

76

#