ММИНОБРНАУКИ 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: БДП

Студент гр. 9384 _____ Нистратов Д.Г.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Разработать БДП, основанный на алгоритме рандомизированной дерамиды поиска. А также найти заданный элемент и удалить его в БДП.

Задание.

13. БДП: Рандомизированная дерамида поиска (treap); Действия 1+2б

Выполнение работы.

При выполненнии работы был описан класс Treap, реализующий создание бинарного дерева, а также были реализованы функции: заполняющая дерево элементом, поиск и удаление заданного элемента, вывод дерева в консоль и в файл.

Оценка алгоритма:

Вставка элемента: O(logn), в худшем случае O(n)

Поиск и удаление элемента: O(logn), в худшем случае O(n)

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	2, 3, 7, 2, 8, 2,2,3,1,2,5,4 53464,6,4,754,745,67 ,5 67,5 ,756,7, 7,567,567,5, 7	1 2 2 2 2 2 3 3 4 5 5 5 6 7 7 7 7 8 67 567 567 567 745 754 453464	Массив был занесен в дерево, а также элемент 756 был удален
2.	'a', 'g', 's', 'b', 'e', 'c', 'a', 'w', 'd'	a b c d e g s w	Был занесен массив char и удален ‘a’
3.	2, 3, 7, 2, 8, 2,2,3,1,2,5,4 53464,6,4,754,745,67 ,5 67,5 ,756,7, 7,567,567,5, 7	1 2 2 2 2 2 3 3 4 5 5 5 6 7 7 7 7 8 67 567 567 567 745 754 756 453464	Элемент 333 не был найдет и удален

Выводы.

В ходе выполнения лабораторный работы была описана БДП: Рандомизированная дерамида поиска (treap);

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <chrono>
#include "treap.h"

using namespace std;

int main()
{
    srand(time(0));
    int xs[] = {2, 3, 7, 2, 8, 2,2,3,1,2,5,453464,6,4 ,754,745,67 ,567,5 ,756,7, 7,5
67,567,5,7};
    char cs[] = {'a', 'g', 's', 'b', 'e', 'c', 'a', 'w', 'd'};
    Treap<int> asd;

    std::cout << "Inserting data:" << std::endl;
    for (size_t t = 0; t < (sizeof(xs)/sizeof(xs[0])); t++)
    {
        std::cout << "Data " << t << ":\nTime spend: ";
        auto start = std::chrono::high_resolution_clock::now();
        asd.insert(asd.root, new Treap<int>(xs[t], rand()));
        auto stop = std::chrono::high_resolution_clock::now();
        auto duration = std::chrono::duration_cast<std::chrono::nanoseconds>(sto
p-start);
        std::cout << duration.count() << " nanoseconds" << std::endl;
    }
    std::cout << "Find argument and delete it:" << std::endl;
    auto start = std::chrono::high_resolution_clock::now();
```

```
    asd.erase(asd.root, 333);
    //asd.erase(asd.root, 'a');
    auto stop = std::chrono::high_resolution_clock::now();
    auto duration = std::chrono::duration_cast<std::chrono::nanoseconds>(stop-
start);
    std::cout << "Time spend: " << duration.count() << " nanoseconds" << std::e
ndl;

    asd.print(asd.root);
    std::cout << std::endl;

    std::ofstream file("output.txt");
    if (!file.is_open())
        std::cout << "File error" << std::endl;
    else
        asd.write(asd.root, file);
    file.close();
    return 0;
}
```

Название файла:treap.h

```
#ifndef TREAP_H
#define TREAP_H

#include <iostream>
#include <fstream>

template <typename T>
class Treap
```

```
{
public:
    Treap() {};
    Treap(T value, int R) : value(value), R(R), Left(nullptr), Right(nullptr) {};

    void split(Treap* t, Treap *&left, Treap *&right, T& val)
    {
        if (t == nullptr)
        {
            left = nullptr;
            right = nullptr;
            return;
        }
        else if (t->value > val)
        {
            split(t->Left, left, t->Left, val);
            right = t;
        }
        else
        {
            split(t->Right, t->Right, right, val);
            left = t;
        }
    }

    void merge(Treap*& t, Treap* left, Treap* right)
    {
        if (!left){
            t = right;
            return;
```

```
    }
    if (!right){
        t = left;
        return;
    }
    if (left->R >= right->R){
        merge(left->Right, left->Right, right);
        t = left;
    }
    else{
        merge(right->Left, left, right->Left);
        t = right;
    }
}

void insert(Treap*& t, Treap* v)
{
    if (!t)
        t = v;
    else if (v->R > t->R)
    {
        split(t, v->Left, v->Right, v->value);
        t = v;
    }
    else if (v->value < t->value)
        insert(t->Left, v);
    else
        insert(t->Right, v);
}
```

```
void erase (Treap*& t, T key)
{
   if (!t) return;
   if (t->value == key){
      std::cout << "Value: " << key << " found!" << std::endl;
      merge(t, t->Left, t->Right);
   }
   else if (t->value > key)
      erase(t->Left, key);
   else
      erase(t->Right, key);
}

void print(Treap *&t)
{
   if (!t) return;
   print(t->Left);
   std::cout << t->value <<' ';
   print(t->Right);
}

void write(Treap *&t, std::ofstream &file)
{
   if (!t) return;
   write(t->Left, file);
   file << t->value << ' ';
   write(t->Right, file);
}

T value;
```

```
    int R;

    Treap *Left, *Right;
    Treap *root = nullptr;
};

#endif // TREAP_H
```