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

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Алгоритмы и структуры данных»
Тема: Случайные бинарные деревья поиска

Студент гр. 9304 _____ Силкин В.А.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург
2020

Цель работы.

Изучить случайные бинарные деревья поиска и реализовать их с помощью языка программирования C++.

Задание.

Вариант 9

БДП: Случайное БДП, действие: Записать в файл элементы построенного БДП в порядке их возрастания; Вывести построенное БДП на экран в наглядном виде.

Выполнение работы.

Были реализованы 2 класса: node и tree.

В классе node хранится значение ноды - value, количество раз, сколько это число было в вводе - count, уровень, на котором находится нода в дереве (для горизонтального вывода) - level, а также левый и правый потомки - left и right. Конструктор задаёт left и right как nullptr.

В классе tree есть публичное поле - head типа node*, которое указывает на корень бинарного дерева, и приватное поле max_level, которое обозначает количество уровней в дереве. Конструктор задаёт head как nullptr, принимает на вход строку, и начинает обрабатывать каждый символ отдельно. Если текущий символ - не цифра, то итерация пропускается. Для каждого символа применяется метод searchAndInsert(), который создаёт новую ноду в случае, если такой цифры ещё не было, а в ином случае, увеличивает поле count в ноде с тем же значением. Метод horizontal() вызывает в цикле приватный метод draw_line(), который через пробел печатает каждую цифру на уровне, а horizontal() на каждой итерации выводит символ конца строки. Так на экран выводится наглядно случайное БДП. Метод LKP() принимает на вход любой поток вывода, после чего записывает в него ввод из приватного метода lkp() и символ переноса строки. lkp() - это ЛКП обход дерева, поэтому в БДП он выведет числа в порядке возрастания. В main() открывается файл с именем "tree.txt", и выводной поток в него подаётся методу LKP(). Так в файл "tree.txt" записываются все цифры из

дерева в порядке возрастания.

Тестирование.

Был написан bash-скрипт для тестирования программы, который выводит результат работы с данными, поступающими из файлов вида "test_<номер>.txt"

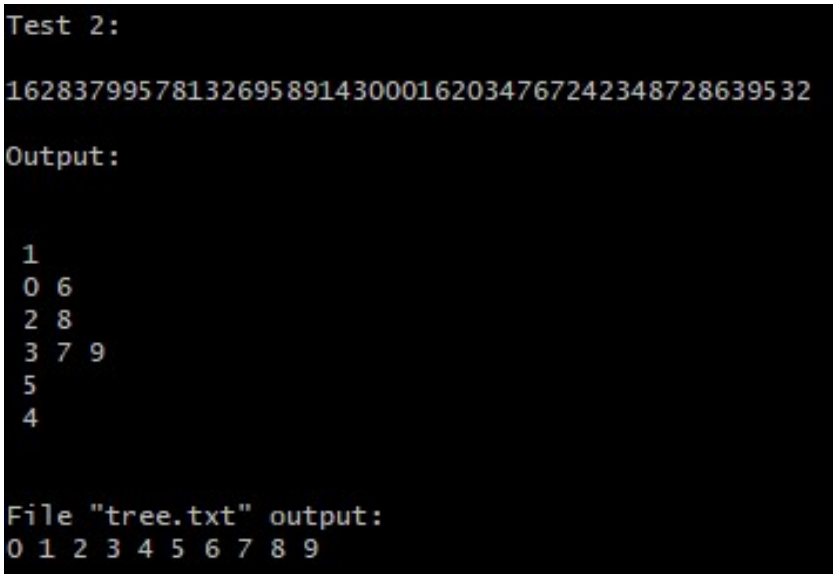

Рисунок 1 - Часть вывода bash-скрипта

Выводы.

Был реализован код для постройки и обработки случайных бинарных деревьев поиска.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <string>
#include <vector>
#include <queue>
#include <fstream>

class node{
public:
    int value;
    int count;
    int level;
    node* left;
    node* right;
    node(): left(nullptr), right(nullptr) {}
};


class tree{

    int max_lvl;

    void draw_line(int current_line, node* cur) {
        if(cur->level == current_line) {
            std::cout << " ";
            std::cout << cur->value;
        }
        if(cur->left != nullptr) draw_line(current_line, cur->left);
        if(cur->right != nullptr) draw_line(current_line, cur->right);
    }

    void lkp(node* cur, std::ostream &outstream) {
        if(cur == nullptr) return;
        lkp(cur->left, outstream);
        outstream << cur->value << " ";
        lkp(cur->right, outstream);
    }

public:

    node *head;

    tree(std::string st): head(nullptr), max_lvl(1) {
        for(auto iter = st.begin(); iter!=st.end(); iter++) {
            char a = *iter;
```

```
            if(!isdigit(a)) continue;
            searchAndInsert(atoi(&a), &head, 1);
        }
    }

    void searchAndInsert(int elem, node** ptr, int level) {
        if ((*ptr) == nullptr) {
            (*ptr) = new node;
            (*ptr)->value = elem;
            (*ptr)->count = 1;
            (*ptr)->level = level;
            if(level > max_lvl) max_lvl = level;
        } else if (elem < (*ptr)->value) {
            searchAndInsert(elem, &(*ptr)->left, level+1);
        } else if (elem > (*ptr)->value) {
            searchAndInsert(elem, &(*ptr)->right, level+1);
        } else {
            (*ptr)->count++;
        }
    }

    void horizontal() {
        for(int i = 1; i <= max_lvl; i++) {
            draw_line(i, head);
            std::cout << "\n";
        }
    }

    void LKP(std::ostream &outstream) {
        lkp(head, outstream);
        outstream << "\n";
    }
};

int main() {
    std::string st;
    std::getline(std::cin, st);
    tree sap(st);
    std::ofstream fname;
    fname.open("tree.txt");
    sap.LKP(fname);
    sap.horizontal();
    fname.close();
    return 0;
}
```