**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 алгоритмы поиска и алгоритмы сжатия**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Азаревич А.Д. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2017

**Цель работы.**

Научиться применять бинарные деревья для решения задач кодирования (сжатия) и поиска.

**Постановка задачи.**

Задание:

По заданному файлу F (типа file of Elem), все элементы которого различны, построить БДП случайного типа. Для построенного БДП проверить, входит ли в него элемент е типа Elem, и если не входит, то добавить элемент е в дерево поиска.

**Основные теоретические положения.**

Бинарное дерево называется бинарным деревом поиска(БДП), если для каждого узла x выполняется следующее условие: для каждого узла l в левом поддереве справедливо key(l) < key(x), а для каждого узла r в правом поддереве – key(r) > key(x).

Можно дать другое определение БДП: бинарное дерево является БДП, если ключи узлов дерева упорядочены при ЛКП-обходе. Например (рисунок 1).

6

3

9

1

5

7

11

8

4

Рис. 1 - Пример дерева поиска

Во всех бинарных деревьях поиска процедура поиска элемента выполняется по следующему алгоритму. Если корень дерева содержит искомый ключ, то поиск завершен. Иначе, если значение в корне дерева меньше искомого, то выполняем поиск в правом поддереве, иначе – в левом поддереве. Если мы переходим в нулевое поддерево, это означает, что искомого элемента в дереве нет. Эта процедура схожа с бинарным поиском.

**Спецификация программы.**

*Назначение программы*.

Программа предназначена для создания случайного бинарного дерева поиска и добавления в него новых элементов.

*Описание программы*.

Программа написана на языке С++ с использованием компилятора ml64 9.0.30729.1. Входными данными для программы является последовательность значений типа double (одно значение встречается не более одного раза) из файла F.txt, являющаяся основой, по которой собирается случайное БДП, а также элемент, который ищется в этом дереве. Проверка входных данных из файла выражается в следующем виде: если элемент уже вставлен в дерево, то он игнорируется (так же есть проверка на наличие самого файла). Выходными данными является информация, есть ли данный элемент в списке или нет.

*Пример диалога с пользователем*.

Дерево:

\_1

\_

\_2

\_

\_5

\_3

\_15

\_6

\_

\_9

Поиск элемента: Элемент вставлен в дерево. Новое дерево:

\_1

\_-1

\_2

\_

\_5

\_3

\_15

\_6

\_

\_9

Завершить? (Y/N)

Y

*Реализация*.

Классы:

1. BTS – класс, представляющий собой случайное БДП.

Переменные класса:

1. BT \*left – указатель на левое поддерево;
2. BT \*right – указатель на правое поддерево;
3. Elem key – значение, содержащееся в данном элементе БДП (Elem = double).

Функции класса:

1. BTS( Elem x);

Конструктор класса. Присваивает key значение x, приравнивает left и right к 0.

1. ~BTS();

Деструктор класса. Вызывает деструктор для левого и правого поддерева.

1. void Display(int n =1)

Печатает n пробелов и содержимое key «действующего» элемента;

Входные параметры:

Номер «высоты» элемента;

Постусловие:

Если left!=NULL, вызывается Display для левого поддерева (аналогично для правого).

1. bool Find\_Add(Elem &x)

Проверяет равенство key==x (ищет х в БДП), если не найдёт – создаёт новый элемент, в котором выполняется данное равенство ;

Входные параметры:

Искомый/вставляемый элемент x;

Выходное значение:

True, если x является элементом бинарного дерева поиска, false – не являлся, но теперь есть;

Постусловие:

Сравнивает значение x с key. Если x< key идёт в левое поддерево, если x>key – в правое, если x==key возвращает true. В случае x<>key, а нужное поддерево отсутствует, то оно создаётся с key=x, возвращается значение false.

Сторонние функции:

void Read\_F (BTS &t, istream &in);

Пока не встретится конец файла, считывает из него значения x типа Elem и вызывает ф-цию t.Find\_Add(x);

Входные данные:

Бинарное дерево поиска t и поток вывода in.

**Тестирование.**

В файле test.txt:

1 2 5 15 6 3 9

|  |  |
| --- | --- |
| Искомое значение | Результат |
| 1 | Присутствует |
| -1 | Новое дерево:  \_1  \_-1  \_2  \_  \_5  \_3  \_15  \_6  \_  \_9 |

**Выводы.**

В ходе выполнения лабораторной работы по программированию бинарных алгоритмов поиска и сжатия мы познакомились с такой структурой данных, как случайное бинарное дерево поиска.

Так же мы освоили прямой обход ЛКП обход бинарного дерева, получили навыки по поиску элементов в БДП.

**Приложение А**

**Исходный код**

BTS.h

#include <iostream>

#include <fstream>

using namespace std;

typedef double Elem;

class BTS

{

public:

BTS( Elem x);

~BTS();

void Display(int n = 1);

bool Find\_Add (Elem &x);

private:

Elem key;

BTS \*left;

BTS \*right;

};

BTS.cpp

#include "BTS.h"

BTS::BTS( Elem x )

{

key = x;

left = right = NULL;

}

BTS::~BTS()

{

if (left != NULL)

delete left;

if (right != NULL)

delete right;

}

void Q(int n)

{

for(int i=1; i<=n; i++)

cout <<" ";

cout << "\_";

}

void BTS::Display(int n)

{

Q(n);

cout << key << endl;

if( left != NULL )

left->Display(n+1);

else

if ( right != NULL )

{

Q(n+1);

cout << endl;

}

if( right != NULL )

right->Display(n+1);

}

bool BTS::Find\_Add (Elem &x)

{

if ( key > x)

{

if ( left == NULL )

{

left = new BTS(x);

return false;

}

return left->Find\_Add(x);

}

if ( key < x)

{

if ( right == NULL )

{

right = new BTS(x);

return false;

}

return right->Find\_Add(x);

}

if ( key == x )

return true;

}

main.cpp

//

// Лабораторная работа №5

// main.cpp

// Азаревич артём, группа 6383

// 13.11.2017

//

// Программа для создания бинарного дерева

// поиска t и последующего поиска элемента х

#include <windows.h>

#include "BTS.h"

void Read\_F (BTS &t, istream &in);

bool Continous();

void main()

{

SetConsoleCP(1251); // для вывода кирилицы

SetConsoleOutputCP(1251); // для вывода кирилицы

ifstream fin("F.txt");

if (!fin)

{

cerr << "Файл не найден";

exit;

}

Elem x;

fin >> x;

BTS t(x);

Read\_F(t, fin);

do

{

system("cls");

cout << "Дерево: \n";

t.Display();

cout << "\n\nПоиск элемента: ";

cin >> x;

if ( t.Find\_Add(x) )

cout << "Элемент в дереве уже есть\n\n";

else

{

cout << "Элемент вставлен в дерево. Новое дерево:\n";

t.Display();

}

}

while ( Continous() );

}

bool Continous()

{

begin:

cout << "Завершить? (Y/N)\n";

char c;

cin >> c;

switch(c)

{

case 'Y': return false;

case 'N': return true;

default: goto begin;

}

}

void Read\_F (BTS &t, istream &in)

{

while( !in.eof() )

{

Elem x;

in >> x;

t.Find\_Add(x);

}

}