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 и алгоритмы сжатия»

Студентка гр. 7381 _____ Алясова А. Н.

Преподаватель _____ Фирсов М. А.

Санкт-Петербург

2018

Задание.

Вариант 4. На вход подаётся файл с закодированным содержимым. Требуется раскодировать содержимое файла статическим алгоритмом Хаффмана.

Описание алгоритма.

На входе нам дано бинарное дерево Хаффмана для алфавита, по нему определяем коды символов. Следует начать с корня и прочитать первый бит сжатого файла. Если это нуль, следует двигаться по левой ветке дерева; если это единица, то двигаться надо по правой ветке дерева. Далее читается второй бит и происходит движение по следующей ветке по направлению к листьям. Когда декодер достигнет листа дерева, он узнает код первого несжатого символа (обычно это символ ASCII). Процедура повторяется для следующего бита, начиная опять из корня дерева.

Далее используем фразу (набор символов) как двоичный код, т е переводим символы в набор единиц и нулей. По этому набору декодируем наше сообщение при помощи уже известных нам кодов.

Описание функций и структур данных.

Для реализации декодирования было принято создать класс HuffmanTree, которыйсодержит деревос алфавитом.

```
template <classT>
class HuffmanTree{
private: HuffmanTree*left; //0
HuffmanTree*right; //1
boolflag =false; //true
ifleaf Tvalue;
```

```
public: HuffmanTree(ifstream& s);
boolisLeaf() {return flag;
}
HuffmanTree *get_left(){
return left;
}
HuffmanTree *get_right(){
return right;
}
Tval(){
return value;
}
~HuffmanTree();
};
```

Конструктор класса инициализирует дерево строкой, содержащейся в начале декодируемого файла.

Метод bool isLeaf() возвращает true, если элемент является листом дерева.

Методы HuffmanTree *get_left() и HuffmanTree *get_right() возвращают указатель на левый и правый элемент.

В код символа добавляется 0 при спуске по левой ветви и1приспуске поправой.Метод T val() возвращает кодируемый элемент.

Тестирование.

Содержимое входного файла	Содержимое выходного файла
(((o)((r)(w)))((l)9((n)(d))((e)(n)))))lol	wwdln

Продолжение таблицы

fdysauhilak	txt is broken
(()gsgk	txt is broken
(a)khnljs	txt is broken
((a)((b)(c)))X	abcaa
(((b)(c))(a))	1.строим бинарное дерево и по нему определяем коды символов a -1 b – 00 c – 01 2.Рассматриваем X – это наш код, т е двоичный код числа X есть наше закодированное сообщение В таблице ANCII у X код 88, в двоичной системе это 01011000 3.Декодируем: 01011000 – c 01011000 - c 01011000 – a 01011000 - b Наше сообщение: ccab

Выводы.

В ходе выполнения данной работы были изучены алгоритмы кодирования и декодирования. Была написана программа, декодирующая файл с помощью алгоритма Хаффмана.

ПРИЛОЖЕНИЕ А

ТЕКСТ ОСНОВНОЙ ПРОГРАММЫ

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <cctype>
#include <string>

using namespace std;



template <class T>
class HuffmanTree{
private:
    HuffmanTree* left;   // 0
    HuffmanTree* right;  // 1
    bool flag = false; //true if leaf
    T value;

public:
    HuffmanTree(ifstream& s);
    bool isLeaf() {return flag;}
    HuffmanTree *get_left() {return left;}
    HuffmanTree *get_right() {return right;}
    T val() {return value;}
    ~HuffmanTree();
};

template <class T>
```

```
HuffmanTree<T>::HuffmanTree(ifstream& s)
{
    flag = false;
    left = NULL;
    right = NULL;
    char ch;
    if (s.peek() == '(')
        s >> ch; // remove '('
    else
        return;
    if (s.peek() == '(')
        {
            left = new HuffmanTree(s);
            if (s.peek() == ')')
                s >> ch; // remove ')'
        }
    else
        {
            flag = true;
            s >> value;
            return;
        }
    if (s.peek() == '(')
    {
        right = new HuffmanTree(s);
        s >> ch; // remove ')'
    }
    if (s.peek() == ')')
        s >> ch; // remove ')'
}
```

```
template <class T>
HuffmanTree<T>::~HuffmanTree()
{
    if (left)
        delete left;
    if (right)
        delete right;
}

int main()
{
    int n, c;
    int *el;
    string str_i, str_o;
    while(true)
    {
        cout << "Press 1 to decode a file\n" <<
                "Press 2 to exit." << endl;
        cin >> str_i;
        if (!isdigit(str_i[0]))
            continue;
        c = stoi(str_i);
        str_i.clear();
        switch (c)
        {
            case 1:
                break;
            case 2:
                return 0;
```

```
            default:
                cout << "Something went wrong. Try again!" << endl;
                continue;
        }
        cout << "Enter input file name: ";
        cin >> str_i;
        cout << "Enter output file name: ";
        cin >> str_o;
        if (str_i == str_o)
        {
            cout << "Input and output files can't be the same." <<
endl;
            continue;
        }
        ifstream f;
        ofstream o;
        char b;
        f.open(str_i);
        o.open(str_o);
        if (!f)
        {
            cout << "Unable to open the input file!" << endl;
            continue;
        }
        if (!o)
        {
            cout << "Unable to open the output file!" << endl;
            continue;
        }
        HuffmanTree<char> *Dictionary = new HuffmanTree<char>(f);
```

```
        HuffmanTree<char> *tmp = Dictionary;
        while (!f.eof())
        {
            b = f.get();
            if (b == '\n')
                if (f.peek() == EOF)
                    break;
            for (char i = 7; i >= 0; i--)
            {
                if (tmp)
                    tmp = ((b & (1 << i)) == 0 ? tmp->get_left() :
tmp->get_right());
                if (tmp && tmp->isLeaf())
                {
                    o << tmp->val();
                    tmp = Dictionary;
                }

            }
        }
        f.close();
        o.close();
        if (tmp)
            cout << str_i << " was decoded to " << str_o << "
succesfully." << endl;
        else
            cout << str_i <<" is broken." << endl;
        delete Dictionary;
    }
}
```