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

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Алгоритмы и структуры данных»
Тема: Кодирование и декодирование

Студент гр. 9303 _____ Молодцев Д.А.

Преподаватель _____ Филатов Ар.Ю.

Санкт-Петербург
2020

Цель работы.

Написать программу для статического декодирования, коды символов хранятся в бинарном дереве

Задание.

Вариант 2

Декодирование: статическое, коды символов хранятся в бинарном дереве.

Основные теоретические положения.

Алгоритм Шеннона — Фано — один из первых алгоритмов сжатия, который впервые сформулировали американские учёные Клод Шеннон и Роберт Фано. Данный метод сжатия имеет большое сходство с алгоритмом Хаффмана, который появился на несколько лет позже и является логическим продолжением алгоритма Шеннона. Алгоритм использует коды переменной длины: часто встречающийся символ кодируется кодом меньшей длины, редко встречающийся — кодом большей длины. Коды Шеннона — Фано — беспрефиксные, то есть никакое кодовое слово не является префиксом любого другого. Это свойство позволяет однозначно декодировать любую последовательность кодовых слов.

Описание алгоритма

У каждого символа, встречаемого в поданной на вход строке считается частота входа в строку. Потом на основе символов и их частот встреч строится бинарное дерево, путем разбиения строки на две части по суммарной частоте этих двух частей. Слева берется более короткая строка с меньшей суммарной частотой встреч. Потом по этому дереву каждый символ кодируется путем к нему в дереве, т. е. „0“ – левое поддерево, „1“ – провае поддерево. А декодирование происходит по обратному методу: если в кодированной строке встречается „0“, то идем по дереву в левое поддерево,

если „1“, то в правое поддерево, и так пока в текущем дереве строка не будет длины 1.

Выполнение работы.

Для реализации задачи были реализованы функции: Функция DecodingProcess(pTree head, pTree fict, string& coded, string& decoded) – функция, в которой происходит декодирование закодированного сообщения. Функция принимает указатель на голову бинарного дерева кодов и на нынешний узел дерева кодов. Функция string CodingProcess(simbol* array, string& str) – функция, в которой собирается в одну строку все закодированное сообщение, принимает на вход указатель на массив структур simbol и на кодируемую строку. Функция void Sort(simbol* array, int& size) сортирует массив структур типа simbol методом слияния по возрастанию числа встреч. Принимает на вход указатель на массив структур и его размер. Функция void CountFrequency(string& str, simbol* arr) считает частоты встреч и заполняет массив типа simbol. Принимает на вход массив структур типа simbol и кодируемую строку. Функция void CreateTree(pTree tree,simbol* arr) – Функция заполняющая бинарное дерево кодов используя алгоритм Фано-Шеннона. Функция void PrintTree(pTree tree) – выводит в консоль строки, суммарные частоты и коды в текущем узле дерева, на вход принимает умный указатель на бинарное дерево кодов. Функция string Make_String(simbol* arr) – собирает строку для помещения ее в «голову» бинарного дерева, принимает на вход массив структур типа simbol.

Так же были реализован класс BinTree, хранящий указатели на левое и правое поддеревья, а так же символы, код на данном этапе и суммарную частоту данных символов. Были реализованы методы для работы с данными полями. Так же была реализована структура simbol, хранящая символ, его частоту и его код.

Исходный код программы представлен в приложении А. Результаты тестирования включены в приложение Б

Выводы.

Были реализованы несколько функций для кодирования и декодирования методом Фано-Шеннона, а так же функции для работы и заполнения бинарного дерева кодов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <memory>
#include <algorithm>

class CodesTree;
using namespace std;
typedef shared_ptr<CodesTree> pTree;

class CodesTree  {
    string data;
    int sum;
    pTree left={nullptr};
    pTree right= {nullptr};
    string code;
public:
    string& Get_Code(){
        return this->code;
    }
    void Set_Code(string& str){
        this->code=str;
    }
    string& Get_Data(){
        return this->data;
    }
    CodesTree(string& str){
        this->data=str;
    }
    void SetSum(int s){
        this->sum=s;
    }
    int GetSum(){
```

```
        return this->sum;
    }
    void Create_left(string& str) {
        this->left = make_shared<CodesTree>(str);
    }
    void Create_right(string& str){
        this->right = make_shared<CodesTree>(str);
    }
    void Set_Str(string& str){
        this->data=str;
    }
    pTree Get_left(){
        return this->left;
    }
    pTree Get_right(){
        return this->right;
    }
};

struct simbol{
    int n;
    char c;
    string code;
};

static int index=0;

void DecodingProcess(pTree head,pTree fict,string& coded,string&
decoded) {
    char c = coded[index];
    int i=index;
    if (index <= coded.length()) {
        if (fict->Get_Data().length() == 1) {
            decoded += fict->Get_Data();
```

```
            DecodingProcess(head,head,coded,decoded);
        }
        if(coded[index]=='0'){
            index++;
            DecodingProcess(head,fict-
>Get_left(),coded,decoded);
        }else if(coded[index]=='1'){
            index++;
            DecodingProcess(head,fict-
>Get_right(),coded,decoded);
        }
    }
}

string CodingProcess(simbol* array, string& str){
    string res;
    for(int i=0;i<str.length();i++){
        for(int j=31;j>0;j--){
            if(array[j].c==str[i]){
                res+=array[j].code;
                continue;
            }
        }
    }
    return res;
}

void Sort(simbol* array, int& size){
    int left_index;
    int right_index;
    int left_bord;
    int mid_bord;
    int right_bord;
    for (int i = 1; i < size; i *= 2){
```

```
        for (int j = 0; j < size - i; j = j+2*i){
            left_index = 0;
            right_index = 0;
            left_bord = j;
            mid_bord = j + i;
            right_bord = j+2*i;
            if(right_bord >= size){
                right_bord = size;
            }
            simbol* sorted_array = new simbol[right_bord -
left_bord];
            while (left_bord + left_index < mid_bord && mid_bord
+ right_index < right_bord){
                if (array[left_bord + left_index].n <
array[mid_bord + right_index].n){
                    sorted_array[left_index + right_index].n =
array[left_bord + left_index].n;
                    sorted_array[left_index + right_index].c =
array[left_bord + left_index].c;
                    left_index += 1;
                }
                else{

sorted_array[left_index+right_index].n=array[mid_bord+right_inde
x].n;

sorted_array[left_index+right_index].c=array[mid_bord+right_inde
x].c;
                    right_index += 1;
                }
            }
            while (left_bord + left_index < mid_bord){
                sorted_array[left_index + right_index].n =
array[left_bord + left_index].n;
```

```
                sorted_array[left_index + right_index].c =
array[left_bord + left_index].c;
                left_index += 1;
            }
            while (mid_bord + right_index < right_bord){
                sorted_array[left_index + right_index].n =
array[mid_bord + right_index].n;
                sorted_array[left_index + right_index].c =
array[mid_bord + right_index].c;
                right_index += 1;
            }
            for (int k = 0; k < left_index + right_index; k++){
                array[left_bord + k].n = sorted_array[k].n;
                array[left_bord + k].c = sorted_array[k].c;
            }
            delete [] sorted_array;
        }
    }
}

void CountFrequency(string& str, simbol* arr){
    transform(str.begin(), str.end(), str.begin(), ::tolower);
    for(int i=0;i<str.length();i++){
        int ind=(char)(str[i])-97;
        if(isalpha(str[i])){
            if(arr[ind].n==0){
                arr[ind].c=str[i];
                arr[ind].n++;
            }else{
                arr[ind].n++;
            }
        }
            if(str[i]=='.'){
                arr[26].c=str[i];
```

```
                arr[26].n++;
            }
            if(str[i]==','){
                arr[27].c=str[i];
                arr[27].n++;
            }
            if(str[i]=='?'){
                arr[28].c=str[i];
                arr[28].n++;
            }
            if(str[i]=='!'){
                arr[29].c=str[i];
                arr[29].n++;
            }
            if(str[i]==' '){
                arr[30].c=str[i];
                arr[30].n++;
            }
    }
}

void CreateTree(pTree tree,simbol* arr){
    int curr_sum=0;
    int lsum=0;
    string ldata;
    string rdata;
    for(int i=0;i<tree->Get_Data().length();i++){
        for(int j=30;j>=0;j--){
            if(tree->Get_Data()[i]==arr[j].c){
                curr_sum+=arr[j].n;
            }
        }
    }
    tree->SetSum(curr_sum);
```

```
for(int i=0;i<tree->Get_Data().length();i++){
    for(int j=0;j<31;j++){
        if(tree->Get_Data()[i]==arr[j].c){
            if(lsum+arr[j].n<=curr_sum/2){
                lsum+=arr[j].n;
                ldata+=arr[j].c;
            }else{
                rdata+=arr[j].c;
            }
        }
    }
}
string lcode=tree->Get_Code()+'0';
string rcode=tree->Get_Code()+'1';
if(ldata.length()>1){
    tree->Create_left(ldata);
    tree->Get_left()->Set_Code(lcode);
    CreateTree(tree->Get_left(),arr);
}else if(ldata.length()==1){
    tree->Create_left(ldata);
    tree->Get_left()->Set_Code(lcode);
    tree->Get_left()->SetSum(lsum);
    for(int j=30;j>=0;j--){
        if(arr[j].c==ldata[0]){
            arr[j].code=lcode;
        }
    }
}
if(rdata.length()>1){
    tree->Create_right(rdata);
    tree->Get_right()->Set_Code(rcode);
    CreateTree(tree->Get_right(),arr);
}else if(rdata.length()==1){
    tree->Create_right(rdata);
```

```
        tree->Get_right()->Set_Code(rcode);
        tree->Get_right()->SetSum(curr_sum-lsum);
        for(int j=30;j>=0;j--){
            if(arr[j].c==rdata[0]){
                arr[j].code=rcode;
            }
        }
    }
}

void PrintTree(pTree tree){
    cout<<tree->Get_Data()<<" "<<tree->GetSum()<<" code:
"<<tree->Get_Code()<<"\n";
    if(tree->Get_left()){
        PrintTree(tree->Get_left());
    } else{
        cout<<"#\n";
    }
    if(tree->Get_right()){
        PrintTree(tree->Get_right());
    }else{
        cout<<"#\n";
    }
}

string Make_String(simbol* arr){
    string res;
    int ind=30;
    while(ind>=0){
        if(arr[ind].n>0){
            res+=arr[ind].c;
        }else{
            break;
        }
```

```
        ind--;
    }
    return res;
}

int main() {
    string path;
    int size=31;
    cout<<"Enter decoded string:\n";
    getline(cin, path,'\n');
    simbol* arr = new simbol[size];
    CountFrequency(path,arr);
    Sort(arr,size);
    string simbols = Make_String(arr);
    pTree Bin_Tree = make_shared<CodesTree>(simbols);
    CreateTree(Bin_Tree,arr);
    cout<<"All elements of our tree:\n";
    PrintTree(Bin_Tree);
    cout<<"Simbol It'scode Frequency\n";
    for(int j=0;j<size;j++){
        if(arr[j].n>0){
            cout<<arr[j].c<<" "<<arr[j].code<<" "<<arr[j].n;
            cout<<"\n";
        }
    }
    string coded_str = CodingProcess(arr,path);
    string decoded_str;
    DecodingProcess(Bin_Tree,Bin_Tree,coded_str,decoded_str);
    cout<<"Coded message:\n"<<coded_str;
    cout<<"\nDecooded message:\n"<<decoded_str;
    delete[] arr;
    //system("pause");
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Тестовые данные генерируются случайным образом.

Таблица Б.1 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	War, war never change...	Coded message: 11010000101111110011010000101001011001111100010101001010111000010111100001011011011 Decooded message: war, war never change...	Программа работает корректно.
2.	Wolth irn totalien krieg?	Coded message: 11110101110000111100010000110110100100011011001011010000001111010010111011101000011111100111111 Decooded message: wolth irn totalien krieg?	Программа работает корректно.
3.	3d3d3d	Coded string can't have numbers!	Программа работает корректно.
4.	And nothing else matters...	Coded message: 10100010110100001111100101101011011001111001000000111001011000100111011010010010000011111101101110111011110111 Decooded message: and nothing else matters...	Программа работает корректно.
5.	Microsoft mIcro.	Coded message: 10001101010100111000010001101111110001101010100111110 Decooded message: microsoft micro.	Программа работает корректно.
6.	Ive seen things you people	Coded message: 01101110000100011100010010100000101	Программа работает корректно.

	wouldnt believe. Attack ships on fire off the shoulder of Orion. I watched C-beams glitter in the dark near the Tannhuaser Gate. All those moments will be lost in time, like tears in rain. Time to die...	1110001011010001111011100001000010 0111110110011111110010100111111110 1 0111010001111111110011111011101111 0111100001110011110100101011101100 1011100001010110001101001110111101 0111100111110001110000010110111111 0 1100001100111000001110110011011010 0100011001111011011011000101110001 0100011100000110011111011101111101 1101011010001100111101100011001111 0100110100111000101100010110001111 1111101001111111000001010110111001 1111001110100101010111001110000111 1101101110110011101110101101000101 1010000010111000101000111011110101 1010111110001100001010101101000101 1100010100010111101010001000000111 1011101011000101101000111110110100 1110101011000110101011110111001011 1000110011110001000111100110011111 0010101000011111000011111111011010 1111011100111101001000110111100111 1 0001110010110100000101110110111001 0101001000110111011011111001000101 1101010101101011000010110100000111 0101010011010001011000101110110111 0010100010111100110011101110110010 101101011010110 Decooded message: ive seen things you people wouldnt believe. attack ships on fire off the shoulder of orion. i watched cbeams glitter in the dark near the tannhuaser	

		gate. all those moments will be lost in time, like tears in rain. time to die...	