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: Декодирование Шеннона-Фано

Студент гр. 7383 _____ Бергалиев М.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

ОГЛАВЛЕНИЕ

1. Цель работы

Цель работы: познакомиться с алгоритмом декодирования Шеннона-Фано и его реализацией на языке программирования C++.

Формулировка задачи: На вход подаётся файл с закодированным содержимым. Требуется раскодировать содержимое алгоритмом Шеннона-Фано.

2. Реализация задачи

В функции main выводится предложение выбрать способ ввода данных либо выйти из программы. В случае выбора файла, программа предлагает ввести имя файла. Создается объект типа std::istream, из которого считывается таблица частот символов и закодированное сообщение. В случае ввода информации с консоли, то она считывается из std::cin. После считывания данных создается дерево кодов символов. Далее вызывается метод, декодирующий сообщение. Если в ходе процесса были введены некорректные данные, то выводится сообщение о неправильных данных и процесс продолжается сначала.

Переменные, используемые в функции main:

- command — строка, содержащая номер команды, отвечающий за выбор способа ввода данных или выхода из программы.
- file — буффер потока файла, участвующий в создании объекта fin типа istream, передаваемого в функцию input_parser.
- filename — имя файла, из которого берутся входные данные.
- freq — множество символов в алфавите вместе с их частотами.
- encoded — закодированное сообщение.
- n — число символов в алфавите.
- c — символ алфавита.
- num — частота встречаемости символа.
- code — дерево кодов символов.

Функция table_parser принимает на вход поток, из которого считывается таблица частот символов. Сначала считывается число символов в алфавите. Далее из каждой строки считывается символ и его частота.

Переменные, используемые в функции table_parser:

- freq — множество символов в алфавите вместе с их частотами.
- n — число символов в алфавите.
- c — символ алфавита.
- num — частота встречаемости символа.

Класс BinTree представляет из себя дерево кодов Шеннона-Фано. Класс содержит следующие поля:

- symbols — символы в данном дереве.
- left — указатель на левое поддерево.
- right — указатель на правое поддерево.

Конструктор класса BinTree принимает на вход множество символов с их частотами встречаемости. Все символы из данного множества записываются в данный узел. Создается два подмножества с примерно одинаковыми общими весами. Левое поддерево создается из подмножества с большим весом, а правое — с меньшим. Используемые переменные:

- weight — общий вес символов.
- sum — вес символов подмножества.

Метод decode принимает закодированную строку. Проходит строку, параллельно проходя по кодовому дереву, идя в левое поддерево при встрече с 1 и в правое поддерево, если с 0. Дойдя до листа, записывает в результирующую строку символ из листа. Используемые переменные:

- cur — текущее дерево в обходе.
- res — декодированное сообщение.

3. Тестирование

Программа была собрана в компиляторе G++ с ключом -std=c++14 в OS

Linux Ubuntu 16.04 LTS.

В ходе тестирования ошибки не были найдены.

Некорректный случай представлен в табл. 1, в котором пропущено число.

Таблица 1 - Некорректный случай с пропущенным числом

Входные данные	Результат
2 a a b 2 1011010	Invalid table

Корректные тестовые случаи представлены в приложении А.

4. Вывод

В ходе работы был реализован класс, представляющий из себя дерево кодов Шеннона-Фано, с методом, декодирующим сообщение. Дерево является удобным способом представления кодов символов.

ПРИЛОЖЕНИЕ А.

ТЕСТОВЫЕ СЛУЧАИ

Таблица 1 — Тестовые случаи

Входные данные	Результат
3 a 1 b 4 c 2 000111	acbb
4 e 2 o 3 l 6 h 1 0000011101	hello
6 p 3 r 7 o 1 g 2 a 2 m 3 011110001011010001	program
5 a 3 b 3 c 3 d 3 e 3 110001011101	adcbe

ПРИЛОЖЕНИЕ Б.

ИСХОДНЫЙ КОД ПРОГРАММЫ

Makefile:

```
all: main.o CodeTree.o
      g++ main.o CodeTree.o -o ShannonFano
main.o: main.cpp CodeTree.hpp
      g++ main.cpp -std=c++14 -c
CodeTree.o: CodeTree.cpp CodeTree.hpp
      g++ CodeTree.cpp -std=c++14 -c
```

main.cpp:

```
#include <iostream>
#include <set>
#include <utility>
#include <fstream>
#include "CodeTree.hpp"

auto table_parser(std::istream& input) {
    std::string s;
    int n = 0;
    getline(input, s);
    n = std::stoi(s);
    char c;
    int num;
    std::set<std::pair<int, char>> freq;
    for(int i=0; i<n; ++i) {
        input >> c;
        getline(input, s);
        num = std::stoi(s);
        freq.insert(std::make_pair(num, c));
    }
    return freq;
}

int main() {
    std::string command;
    std::filebuf file;
    std::string filename;
    while(true) {
        std::cout << "Enter 0 to read input from consol or 1 to read from
file or 2 to exit: ";
        getline(std::cin, command);
        try {
            if(std::stoi(command) == 2)
                break;
        }
        catch(std::exception &e) {
            std::cout << "Invalid command, try again" << std::endl;
            continue;
        }
        std::set<std::pair<int, char>> freq;
        std::string encoded;
```

```
        try {
            switch(std::stoi(command)) {
            case 0: {
                std::cout << "Enter number of symbols: ";
            std::string s;
            int n = 0;
            getline(std::cin, s);
            n = std::stoi(s);
                std::cout << "Enter symbols and their frequency:" <<
std::endl;
                char c;
                int num;
                for(int i=0; i<n; ++i) {
                    std::cin >> c;
                    getline(std::cin, s);
                    num = std::stoi(s);
                    freq.insert(std::make_pair(num, c));
                }
                std::cout << "Enter encoded message:" << std::endl;
                std::getline(std::cin, encoded);
                break;
            }
            case 1: {
                std::cout << "Enter file name: ";
                getline(std::cin, filename);
                if(file.open(filename, std::ios::in)) {
                    std::istream fin(&file);
                    freq = table_parser(fin);
                    getline(fin, encoded);
                    file.close();
                }
                else {
                    std::cout << "Incorrect filename" << std::endl;
                    file.close();
                    continue;
                }
                break;
            }
            default: {
                std::cout << "Incorrect command, try again" << std::endl;
                continue;
            }
            }
        }
        catch(std::exception &e) {
            std::cout << "Invalid table" << std::endl;
            continue;
        }
      try{
            CodeTree code(freq);
            std::cout << "Decoded message:" << std::endl <<
code.decode(encoded) << std::endl;
      }
      catch(std::exception &e){
```

```
            std::cout << e.what() << std::endl;
        }
    }
    return 0;
}
```

CodeTree.hpp:

```
#pragma once
#include <set>
#include <utility>

class CodeTree {
public:
    CodeTree(std::set<std::pair<int, char>>);
    std::string decode(std::string&);
private:
    CodeTree* left;
    CodeTree* right;
    std::string symbols;
};
```

CodeTree.cpp:

```
#include <string>
#include <stdexcept>
#include "CodeTree.hpp"

CodeTree::CodeTree(std::set<std::pair<int, char>> freq) {
    int weight = 0, sum = 0;
    for(auto it : freq) {
        weight += std::get<0>(it);
        symbols.push_back(std::get<1>(it));
    }
    if(symbols.length() == 1) {
        left = nullptr;
        right = nullptr;
        return;
    }
    std::set<std::pair<int, char>> left_freq, right_freq;
    for(auto it = freq.rbegin(); it != freq.rend(); ++it) {
        if(std::get<0>(*it) + sum <= (weight+1)/2) {
            sum += std::get<0>(*it);
            left_freq.insert(*it);
        }
        else
            right_freq.insert(*it);
    }
    if(sum < weight - sum)
        left_freq.swap(right_freq);
    left = new CodeTree(left_freq);
    right = new CodeTree(right_freq);
}

std::string CodeTree::decode(std::string& encoded) {
    CodeTree* cur = this;
    std::string res;
    for(auto it : encoded) {
```

```
            if(it == '1')
                cur = cur->left;
            else
              if(it == '0')
                  cur = cur->right;
              else
                throw std::invalid_argument("Invalid encoded string");
            if(cur->left == nullptr) {
                res += cur->symbols;
                cur = this;
            }
        }
        return res;
    }
```