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: Метод Шеннона-Фано

Студент гр. 9304 _____ Атаманов С.Д.

Преподаватель _____ Филатов Ар. Ю.

Санкт-Петербург

2020

Цель работы.

Ознакомиться с алгоритмом шифрования Шеннона-Фано. Реализовать алгоритм, используя язык программирования C++.

Задание.

Вариант 1.

Метод Шеннона-Фано

Программа должна иметь следующие ключи запуска: encode (включить режим «кодирование»), decode (включить режим «декодирование»), file (указать название входного файла, иначе ожидается ввод с консоли), o (указать название выходного файла, иначе на консоль), debug (включить режим отладки). Файл, полученный кодированием, должен быть расшифрован программой обратно.

Выполнение работы.

Описание алгоритма.

Работа алгоритма основывается на Бинарном дереве. Алгоритм получает таблицу символов с частотой их появления в тексте(далее — весом). Все символы складываются в единую строку, их веса складываются. В узлы дерева записываются отдельные символы или строка состоящая из них. Затем, если это строка, она разделятеся по весу напополам, в левое поддерево записывается строка с меньшим весом, в правое с большим. Так продолжается до тех пор, пока в листьях не будут отдельные символы. После по дереву проходятся, добавляя к каждому коду левого поддерева каждого узла 0, а к правому 1. В результате у каждого символа получается уникальный код который тем короче, чем чаще встречается символ и наоборот, тем длинее, чем символ встречается реже.

Формат входных и выходных данных.

Программа имеет 4 ключа запуска: encode (включить режим «кодирование»), decode (включить режим «декодирование»), file (указать название входного файла, иначе ожидается ввод с консоли), o (указать название выходного файла, иначе на консоль). При указании только режима

«кодирование» программа считывает строку из stdin, выводит закодированное сообщение в stdout и сохраняет файл с кодом в папке с исполняемым файлом. В режиме только «декодирование» программе нужно скормить путь к файлу с кодом символов, ввести закодированное сообщение в stdin, раскодированное сообщение будет выведено в stdout. Ключи -f и -o требуют пути к файлам, из которых будет происходить чтение и запись соответственно.

Формат входных данных ограничивается символами из таблицы ASCII английского языка.

Используемые структуры данных и реализованные функции.

class BinTreeNode — класс, описывающий узлы бинарного дерева. Состоит из 3 полей: 2 std::shared_ptr<BinTreeNode> указателя на левое и правое поддерево, std::pair<std::string, int> - значение узла, где std::string — строка или символ, int — вес.

std::shared_ptr<BinTreeNode> getShannonFanoTree() - метод класса BinTreeNode, которое возвращает дерево Шеннона-Фано с узлами, заполненными строками и символами.

void getListOfElem() - функция заполняет словарь уникальными символами, параллельно считая вес каждого символа в кодируемой строке.

void getStringWithWeigh() - функция складывает все символы словаря в одну строку, вместе с их весами.

std::string encode() - функция кодирует исходную строку. Функция проходит по строке, при встрече каждого символа, ищет в словаре его код и записывает его в выходной файл или stdout, параллельно формируя файл с кодами символов.

std::string getStringFromFile() - преобразует содержимое файла в одну строку символов.

std::map<std::string, char> getCodesFromFile() - заполняет словарь для декодирования из файла с кодами символов.

void decode() - происходит декодирование файла. Происходит проход по строке, запись символов во временную строку и сравнение последней со

словарем кодов, после в выходной файл или же stdout записывается декодированный символ за символом.

Разработанный программный код смотри в приложении А.

Тестирование.

Для тестирования был написан bash-скрипт, который считывает аргументы коммандной строки и тестовые данные из текстовых документов папки Tests и передает их программе. Результаты работы программы сравниваются с заведомо правильными данными из папки Test_results. Данные о результатах тестирования выводятся в stdout.

Рисунок 1 — Пример работы программы

Результаты тестирования смотри в приложении Б.

Выводы.

В ходе выполнения работы были получены навыки работы с алгоритмом Шеннона-Фано.

Была написана программа на языке C++, которая выполняет кодирование и декодирование текста методом Шеннона-Фано.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ.

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <memory>
#include <map>
#include <algorithm>
#include <getopt.h>

class BinTreeNode{
public:
   std::shared_ptr<BinTreeNode> left {nullptr};
   std::shared_ptr<BinTreeNode> right {nullptr};
   std::pair<std::string, int> data;

   std::shared_ptr<BinTreeNode> getShannonFanoTree(std::pair<std::string, int>
stringWithWeight, std::map<char, int> map, std::map<char, std::string>& codes, std::string
code) {
      std::shared_ptr<BinTreeNode> tree = std::make_shared<BinTreeNode>();
      std::pair<std::string, int> left;
      std::pair<std::string, int> right;
      tree->data = stringWithWeight;
      if(stringWithWeight.first.length() == 1) {
         codes.insert({tree->data.first[0], code});
         return tree;
      }
      while (true) {
         if ((left.second + map[stringWithWeight.first[0]]) > (stringWithWeight.second / 2) &&
left.second == 0) {
            right.first += stringWithWeight.first[0];
            right.second += map[stringWithWeight.first[0]];
            stringWithWeight.second -= map[stringWithWeight.first[0]];
            stringWithWeight.first.erase(0, 1);
            left.first = stringWithWeight.first;
            left.second = stringWithWeight.second;
            break;
         }
         else if((left.second + map[stringWithWeight.first[0]]) > (stringWithWeight.second / 2))
            break;
         else {
            left.first += stringWithWeight.first[0];
            left.second += map[stringWithWeight.first[0]];
            stringWithWeight.second -= map[stringWithWeight.first[0]];
            stringWithWeight.first.erase(0, 1);
         }
      }
      if(right.second == 0) {
         right.first = stringWithWeight.first;
         right.second = stringWithWeight.second;
      }

      if (left.second != 0) {
         tree->left = std::make_shared<BinTreeNode>();
         tree->left = getShannonFanoTree(left, map, codes, code + '0');
      }
```

```
        if (right.second != 0) {
            tree->right = std::make_shared<BinTreeNode>();
            tree->right = getShannonFanoTree(right, map, codes, code + '1');
        }
        return tree;
    }
};

void getListOfElem(std::map<char, int>& map, std::string::iterator iterator){
    if(*iterator == '\0')
        return;
    while(*iterator != '\0') {
        if (map.find(*iterator) != map.end()) {
            map[*iterator]++;
        } else {
            map.insert({*iterator, 1});
        }
        iterator++;
    }
}

void getStringWithWeigh(std::map<char, int> stringMap, std::pair<std::string, int>&
weightString){
    auto iterBeg = stringMap.begin();
    auto iterEnd = stringMap.end();
    for(iterBeg;iterBeg != iterEnd;iterBeg++){
        weightString.first += iterBeg->first;
        weightString.second += iterBeg->second;
    }
}

std::string encode(std::string::iterator encodeString){
    std::map<char, int> map;
    std::map<char, std::string> codes;
    std::pair<std::string, int> symbols;
    std::shared_ptr<BinTreeNode> shannonTree;
    std::string code;
    std::string output;

    getListOfElem(map, encodeString);
    getStringWithWeigh(map, symbols);
    shannonTree = shannonTree->getShannonFanoTree(symbols, map, codes, code);

    for(;*encodeString != '\0';encodeString++)
        output += codes[*encodeString];

    std::ofstream outputFile;
    outputFile.open("./Codes.txt");
    auto iterBeg = codes.begin();
    for(iterBeg;iterBeg != codes.end();iterBeg++)
        outputFile << iterBeg->first << ":" << iterBeg->second << ";\n";
    outputFile.close();

    return output;
}

std::string getStringFromFile(const std::string& fileName) {
    std::string stringFile;
    std::ifstream encodeFile;
    encodeFile.open(fileName);
    if (!encodeFile.is_open()){
```

```
        std::cout << "Error!\nUndeclared <fileName>.txt\n";
                exit(EXIT_FAILURE);
        }

    std::getline(encodeFile, stringFile);
    encodeFile.close();
    if (stringFile.empty()){
        std::cout << "Error: Empty encode\n";
                exit(EXIT_FAILURE);
        }
    return stringFile;
}

std::map<std::string, char> getCodesFromFile(const std::string& fileName){
    std::map<std::string , char> codes;
    char symbol;
    std::string temp;
    std::string code;
    auto iterBeg = temp.begin();
    std::ifstream file;
    file.open(fileName);
        if(!file.is_open()){
                std::cout << "Error: Undeclared codeFile\n";
                exit(EXIT_FAILURE);
        }
    while(!file.eof()){
        std::getline(file, temp);
        iterBeg = temp.begin();
        symbol = *iterBeg;
        iterBeg += 2;
        while(*iterBeg != ';') {
            code += *iterBeg;
            iterBeg++;
        }
        codes.insert({code, symbol});
        code.clear();
        temp.clear();
    }
    file.close();
    return codes;
}

void decode(const std::string& encode, std::map<std::string, char> codes, int output){
    std::string code;
    if(output) {
        std::ofstream decodeFile;
        decodeFile.open("./Output.txt");
        auto iterBeg = encode.begin();
                if(encode.empty()){
                        std::cout << "Error!\nUndeclared <fileName>.txt\n";
                        exit(EXIT_FAILURE);
                }
        for (; iterBeg != encode.end(); iterBeg++) {
            while (true) {
                code += *iterBeg;
                if (codes.find(code) != codes.end())
                    break;
                else if((iterBeg+1) == encode.end() && codes.find(code) == codes.end()){
                                        decodeFile << "Error: Wrong codeFile\n";
                                        std::cout << "lol\n";
                                        exit(EXIT_FAILURE);
```

```
                                    }
                                    else{
                                            iterBeg++;
                        continue;
                    }
                }
                decodeFile << codes[code];
                code.clear();
            }
                        decodeFile << "\n";
                        decodeFile.close();
        }
        else{
            auto iterBeg = encode.begin();
            for (iterBeg; iterBeg != encode.end(); iterBeg++) {
                while (true) {
                    code += *iterBeg;
                    if (codes.find(code) != codes.end())
                        break;
                                    else if(iterBeg == encode.end() && codes.find(code) ==
codes.end()){
                        std::cout << "Error: Wrong codeFile\n";
                                            exit(EXIT_FAILURE);
                                    }
                    else {
                        iterBeg++;
                        continue;
                    }
                }
                std::cout << codes[code];
                code.clear();
            }
                        std::cout << "\n";
        }
}

struct Configuration{
    int encode = 0;
    int decode = 0;
            int output = 0;
    std::string codeFile;
    std::string inputFile;
};

int main(int argc, char* argv[]){
    Configuration config;
    char* opts = "e?:d:f:o?:";
    struct option longOpts[]{
            {"encode", no_argument, nullptr, 'e'},
            {"decode", required_argument, nullptr, 'd'},
            {"file", required_argument, nullptr, 'f'},
            {"output", no_argument, nullptr, 'o'},
            {nullptr, 0, nullptr, 0}
    };

    int opt;
    int longIndex;
    opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
    while(opt != -1){
        switch (opt) {
            case 'e':
```

```
            config.encode = 1;
            break;
         case 'd':
            config.decode = 1;
            config.codeFile = optarg;
            break;
         case 'f':
            config.inputFile = optarg;
            break;
         case 'o':
            config.output = 1;
            break;
      }
      opt = getopt_long(argc, argv, opts, longOpts, &longIndex);
   }

   if(config.encode == config.decode){
      std::cout << "Error: Must be or 'encode' or 'decode'\n";
                  exit(EXIT_FAILURE);
        }
   if(config.encode){
      std::string encodeString;
      std::string encoded;
      if(!config.inputFile.empty()){
         encodeString = getStringFromFile(config.inputFile);
      }
      else
         std::getline(std::cin, encodeString);

      auto iterBeg = encodeString.begin();
      encoded = encode(iterBeg);

      if(config.output){
         std::ofstream output;
         output.open("./Output.txt");
         output << encoded;
         output.close();
      }
      else
         std::cout << encoded << "\n";
   }

   if(config.decode){
      std::map<std::string, char> codes = getCodesFromFile(config.codeFile);
      std::string encode;
      if(!config.inputFile.empty()) {
         encode = getStringFromFile(config.inputFile);
      }
      else
         std::getline(std::cin, encode);

      decode(encode, codes, config.output);
   }

        return 0;
}
```

ПРИЛОЖЕНИЕ Б.

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ.

Входные данные	Выходные данные	Комментарий
-e Hello! My name is Sergey!	0110101110111101111110001000 0111111111001110110011110010 1001101011111000100010111110 1110010111111010	Кодирование из stdin в stdout
-d ./Tests/Tests/Test2/Codes.txt 0110101110111101111111000 1000011111111100111011000 1111001010011010111111000 1000101111101110010111111 11010	Hello! My name is Sergey!	Декодирование 1 текста из stdin в stdout
-e -f ./Tests/Tests/Test3/Enter.txt I show you, who boss of this gym!!!	0001011110101111011111001111 1111101111010000011111101011 1100110001101110111001110100 1001111100101110100111001100 1111111110101001001001	Кодирование из файла в stdout
-d ./Tests/Tests/Test4/Codes.txt -f ./Tests/Tests/Test4/Enter.txt 0001011110101111011111100 1111111110111101000000111 1110101111001100011011110 1110011101001001111110010 1110100111001100111111111 10101001001001	I show you, who boss of this gym!!!	Декодирование 3 текста из файла в stdout
-e -o Hey, buddy, you choose the wrong door. The leather club two box down...	0111001011111111110100001111 1111110100110011111111101000 1111111111011111100010001010 1110111011110010110011110101 0110110011111101110111011100 1101000010011101110111101011 0000111011010110110011000101 1011110111101010110111110100 1000110001111100111110011110 1111110110100011111110111111 1100010011101111111011001011 001100110	Кодирование из stdin в файл

-d ./Tests/Tests/Test6/Codes.txt -o 01110010111111111010000111111111101001100111111110100011111111111011111000100010101110111011110010110011110101011011001111110111011101110011010000100111011101110111010110000111011010110110011000101011011101111010101101111101001000110001111001111100111101111101101000111111101111111000100111011111011100101101100111001	Hey, buddy, you choose the wrong door. The leather club two box down...	Декодирование 5 текста из stdin в файл
Oh, s##t, I'm sorry! Sorry for what? Our daddy taught us not to be ashamed of our...	1001110000100111110000110001111111000100100110100010110101011111011011111011101111110000011000110111101110111111011111001111111011111010110111010111111111011111110101111110101111110110011111101111100111011100111111011100110101111011100101011010111011110101110011101110111101110001110011100111	Кодирование из файла в файл
-d ./Tests/Tests/Test8/Codes.txt -f ./Tests/Tests/Test8/Enter.txt -o 10011100001001111100001100011111110001001001101000101011010111101101111101111000001100011011110111111101111100111111	Oh, s##t, I'm sorry! Sorry for what? Our daddy taught us not to be ashamed of our...	Декодирование 7 текста из файла в файл

1011001010111110001100011001111111011100110111010101011101011101111111101111110101011111101011111011001111100111111101111001110100110111111100111111011011011011010111100110101111011001010110101101111101011100111011101111110111011111111011100011100111100111		
-e -d ./Codes.txt Proverka №9	Error: Must be or 'encode' or 'decode'	Неправильный ввод аргументов
-e -f ./HA-HA	Error! Undeclared <fileName>.txt	Не существует файла, который нужно закодировать
-d ./Tests/Tests/Test11/Codes.txt -f ./Ha-Ha	Error! Undeclared <fileName>.txt	Не существует файла, который можно декодировать
-e -f ./Tests/Tests/Test12/Enter.txt	Error: Empty encode	Файл, который нужно закодировать пустой
-d ./MustBeCodes.txt ....:	Error: Undeclared codeFile	Файл с кодом для декодирования не существует
-d ./Tests/Tests/Test15/Codes.txt 1001110000100111110000110001111111000100100110100010110101011111011011111011101111111100000110001101111101110111111110110111111110111110011111111011001010111110001100011001111111011100110111010101011101011101111111101111110101011111101011111	Error: Wrong codeFile	Файл с кодом для декодирования не от этого текста

101100111110011111110111 1001110100110111111110011 111101101101101101011110 011010111101100101011010 110111101011100111011101 1111101110111111111011100 01110011100111		