Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №5
по курсу "Защита информации"

Тема Алгоритм сжатия Хаффмана

Студент Золотухин А.В.

Группа ИУ7-74Б

Преподаватели Чиж И.С.

Москва, 2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Алгоритм Хаффмана — алгоритм оптимального префиксного кодирования алфавита. Был разработан в 1952 году аспирантом Массачусетского технологического института Дэвидом Хаффманом при написании им курсовой работы. Используется во многих программах сжатия данных, например, PKZIP 2, LZH и др.

В рамках выполнения лабораторной работы необходимо решить следующие задачи:

— описать алгоритм Хаффмана;

— реализовать алгоритм Хаффмана;

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Алгоритм Хаффмана

Основная идея заключается в кодировании переменной длины. Мы можем использовать тот факт, что некоторые символы в тексте встречаются чаще, чем другие (см. здесь), чтобы разработать алгоритм, который будет представлять ту же последовательность символов меньшим количеством битов. При кодировании переменной длины мы присваиваем символам переменное количество битов в зависимости от частоты их появления в данном тексте. В конечном итоге некоторые символы могут занимать всего 1 бит, а другие 2 бита, 3 или больше. Проблема с кодированием переменной длины заключается лишь в последующем декодировании последовательности.

Чтобы избежать неоднозначности при декодировании, мы должны гарантировать, что наше кодирование удовлетворяет такому понятию, как префиксное правило, которое в свою очередь подразумевает, что коды можно декодировать всего одним уникальным способом. Префиксное правило гарантирует, что ни один код не будет префиксом другого.

Построение кода Хаффмана сводится к построению соответствующего бинарного дерева по следующему алгоритму:

1. Составим список кодируемых символов, при этом будем рассматривать один символ как дерево, состоящее из одного элемента с весом, равным частоте появления символа в строке.

2. Из списка выберем два узла с наименьшим весом.

3. Сформируем новый узел с весом, равным сумме весов выбранных узлов, и присоединим к нему два выбранных узла в качестве детей.

4. Добавим к списку только что сформированный узел вместо двух объединенных узлов.

5. Если в списке больше одного узла, то повторим пункты со второго по пятый.

2 КОНСТРУКТОРСКАЯ ЧАСТЬ

2.1 Схема алгоритмов работы веб-сервера

На рисунке 2.1 представлена схема алгоритма схема алгоритма Хаффмана.

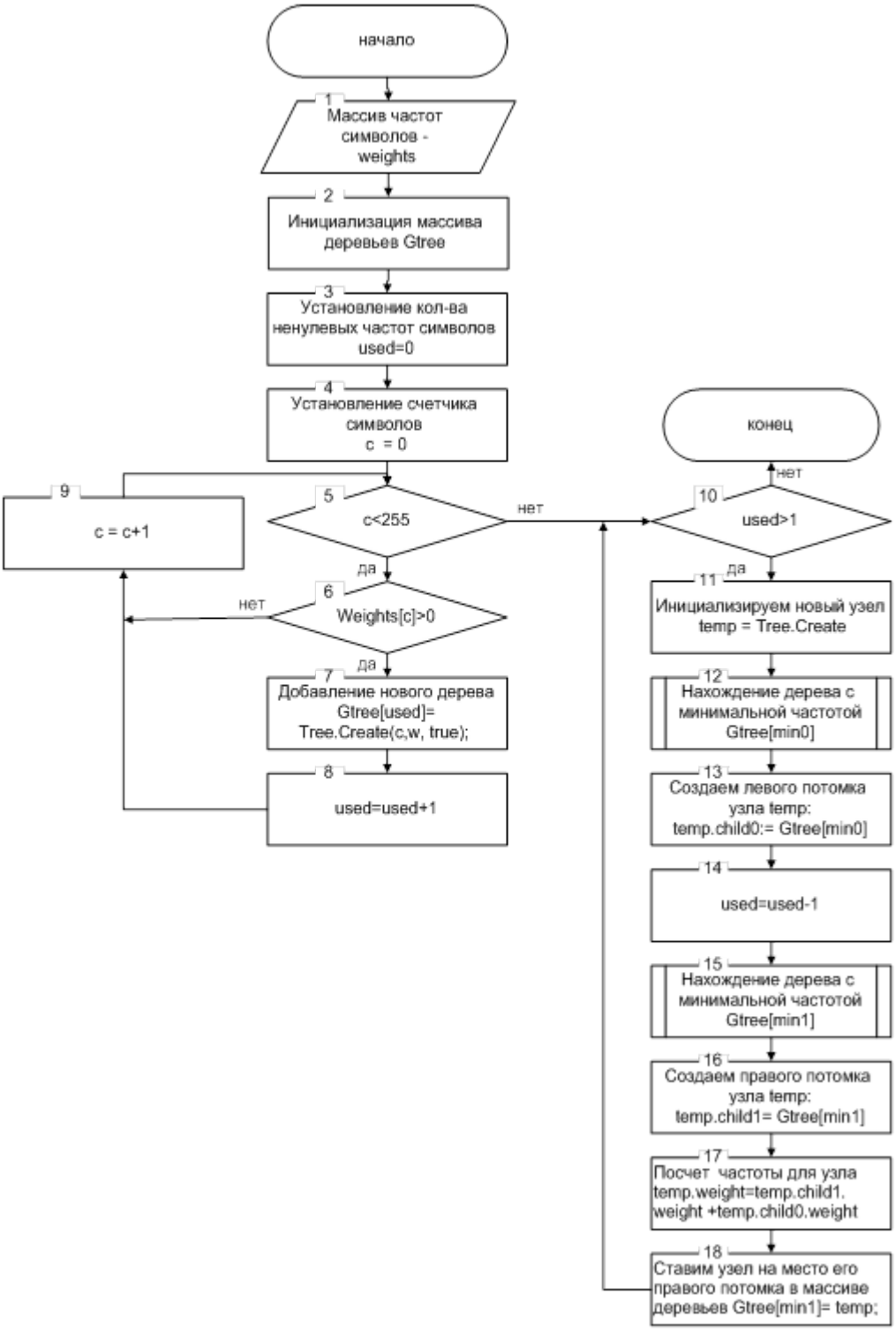

Рисунок 2.1 – Схема алгоритма Хаффмана

3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1 Выбор средств реализации

В качестве языка программирования для реализации данной лабораторной работы использовался язык программирования C, так как он позволяет работать с файлами и массивами. В качестве среды разработки использовалась Visual Studio.

3.2 Реализация алгоритма Хаффмана

В листингах 3.1 представлена реализация алгоритма Хаффмана.

Листинг 3.1 – Реализация алгоритма Хаффмана

```
Node* generate_huffman_tree(const std::map<char, ll>& value) {
    std::vector<Node*> store = sort_by_character_count(value);
    Node* one, * two, * parent;
    sort(begin(store), end(store), sortbysec);
    if (store.size() == 1) {
        return combine(store.back(), nullptr);
    }
    while (store.size() > 2) {
        one = *(store.end() - 1); two = *(store.end() - 2);
        parent = combine(one, two);
        store.pop_back(); store.pop_back();
        store.push_back(parent);

        std::vector<Node*>::iterator it1 = store.end() - 2;
        while ((*it1)->count < parent->count && it1 !=
           begin(store)) {
            --it1;
        }
        std::sort(it1, store.end(), sortbysec);
    }
    one = *(store.end() - 1); two = *(store.end() - 2);
    return combine(one, two);
}
```

3.3 Тестирование

Тест с одной буквой

Исходный размер файла = 8 байт

Размер сжатого файла = 6 байт

Исходный текст = 3131 3131 3131 3131

Сжатый текст = 6100 3101 ff00

Тест с двумя буквами и частотой 3:1

Исходный размер файла = 8 байт

Размер сжатого файла = 9 байт

Исходный текст = 3131 3132 3131 3132

Сжатый текст = 6101 3001 0162 0031 0011

Тест с тремя буквами и частотой 1:1:1

Исходный размер файла = 9 байт

Размер сжатого файла = 15 байт

Исходный текст = 3132 3331 3233 3132 3300

Сжатый текст = 6102 3101 0262 3030 0263 3130 8c01 0062

ЗАКЛЮЧЕНИЕ

В результате выполнения данной лабораторной работы была достигнута цель работы: реализована программа сжатия файла при помощи алгоритма Хаффмана. Были решены все задачи — описан и реализован алгоритм Хаффмана.