



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 ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

по дисциплине «Защита информации»

«Алгоритм сжатия Хаффмана»

Студент группы ИУ7-72Б

(Подпись, дата)

М. Х. Фам
(И.О. Фамилия)

Руководитель

(Подпись, дата)

И. С. Чиж
(И.О. Фамилия)

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Аналитическая часть	5
1.1 Алгоритм Хаффмана	5
2 Конструкторская часть	6
2.1 Разработка алгоритмов	6
3 Технологическая часть	7
3.1 Средства реализации	7
3.2 Реализация алгоритма	7
3.3 Тестирование	10
Заключение	11

ВВЕДЕНИЕ

Сжатие данных — обратимое преобразование данных, производимое с целью уменьшения занимаемого ими объёма с целью уменьшения объёма данных, который требуется для хранения или передачи информации.

Целью данной работы является реализация в виде программы алгоритма сжатия данных Хафмана, обеспечить сжатие и разжатие произвольного файла с использованием разработанной программы, рассчитывать коэффициент сжатия. Предусмотреть работу с пустым, однобайтовым файлами.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- 1) изучить алгоритм сжатия Хафмана;
- 2) реализовать алгоритм сжатия Хафмана в виде программы, обеспечив возможности сжатия и разжатия произвольного файла и расчёт коэффициента сжатия;
- 3) протестировать разработанную программу, показать, что удаётся сжимать и разжимать файлы разных форматов;
- 4) описать и обосновать полученные результаты в отчёте о выполненной лабораторной работе.

1 Аналитическая часть

1.1 Алгоритм Хаффмана

Алгоритм Хаффмана — алгоритм сжатия данных, который формирует основную идею сжатия файлов. Кодирование Хаффмана — это тип кодирования с переменной длиной слова. Был разработан в 1952 году аспирантом Массачусетского технологического института Дэвидом Хаффманом при написании им курсовой работы. В настоящее время используется во многих программах сжатия данных.

Основные этапы алгоритма сжатия с помощью кодов Хаффмана:

Сбор статистической информации для последующего построения таблиц кодов переменной длины

Построение кодов переменной длины на основании собранной статистической информации

Кодирование (сжатие) данных с использованием построенных кодов

Алгоритм состоит из следующих шагов:

- сортировка выходных символов, не меняя местоположения символа, по вероятности их встречаемости в убывающем порядке;
- объединение двух символов с наименьшими вероятностями в композицию символов с вероятностью, равной сумме исходных вероятностей;
- повторения предыдущего шага до тех пор, пока не получится композиция с вероятностью 1, которая называется корнем. Полученная структура называется деревом Хаффмана;
- проход по дереву от корня до соответствующего символа и присвоение 0 (1) — левой и 1 (0) — правой ветви.

2 Конструкторская часть

2.1 Разработка алгоритмов



Рисунок 2.1 – Схема кодирования Хаффмана

3 Технологическая часть

3.1 Средства реализации

Для программной реализации шифровальной машины был выбран язык C++ [?]. В данном языке есть все требующиеся инструменты для данной лабораторной работы. В качестве среды разработки была выбрана среда Visual Studio Code.

3.2 Реализация алгоритма

Листинг 3.1 – Алгоритм сжатия файла

```
1  void decompress(const char* filename, const ll Filesize)
2  {
3      const std::string fl = filename;
4      FILE* iptr = fopen(std::string("../data/" + fl).c_str(),
5                          "rb");
6      FILE* optr = fopen(std::string("../data/d_" +
7                          fl).c_str(), "wb");
8
9      if (iptr == NULL)
10     {
11         perror("Error: File not found");
12         exit(-1);
13     }
14
15     auto [padding, headersize] = decode_header(iptr);
16     store_huffman_value();
17     print_tree();
18
19     byte ch;
20     char counter = 7;
```

```

19     ll size = 0;
20     const ll filesize = Filesize - headersize;
21     Node* traverse = root;
22     ch = fgetc(iptr);
23     while (size != filesize)
24     {
25         while (counter >= 0)
26         {
27             traverse = ch & (1 << counter) ? traverse->right
28                 : traverse->left;
29             --counter;
30             if (!traverse->left && !traverse->right) {
31                 fputc(traverse->character, optr);
32                 if (size == filesize - 1 && padding ==
33                     counter + 1) {
34                     break;
35                 }
36                 traverse = root;
37             }
38             ++size;
39             counter = 7;
40             ch = fgetc(iptr);
41         }
42         fclose(iptr);
43         fclose(optr);
44     }

```

Листинг 3.2 – Алгоритм разжатия файла

```

1 void decompress(const char* filename, const ll Filesize)
2 {
3     const std::string fl = filename;
4     FILE* iptr = fopen(std::string("../data/" + fl).c_str(),
5         "rb");
6     FILE* optr = fopen(std::string("../data/d_" + fl).c_str(),

```

```

        "wb");
6
7     if (iptr == NULL)
8     {
9         perror("Error: File not found");
10        exit(-1);
11    }
12
13    auto [padding, headersize] = decode_header(iptr);
14    store_huffman_value();
15    print_tree();
16
17    byte ch;
18    char counter = 7;
19    ll size = 0;
20    const ll filesize = Filesize - headersize;
21    Node* traverse = root;
22    ch = fgetc(iptr);
23    while (size != filesize)
24    {
25        while (counter >= 0)
26        {
27            traverse = ch & (1 << counter) ? traverse->right :
                traverse->left;
28            --counter;
29            if (!traverse->left && !traverse->right) {
30                fputc(traverse->character, optr);
31                if (size == filesize - 1 && padding == counter +
                    1) {
32                    break;
33                }
34                traverse = root;
35            }
36        }
37        ++size;

```



```
38         counter = 7;
39         ch = fgetc(iptr);
40     }
41     fclose(iptr);
42     fclose(optr);
43 }
```

3.3 Тестирование

Все тесты с файлами успешно пройдены.

Заключение

Целью данной работы является реализация в виде программы алгоритма сжатия данных Хафмана, обеспечить сжатие и разжатие произвольного файла с использованием разработанной программы, рассчитывать коэффициент сжатия. Предусмотреть работу с пустым, однобайтовым файлами.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- 1) изучить алгоритм сжатия Хафмана;
- 2) реализовать алгоритм сжатия Хафмана в виде программы, обеспечив возможности сжатия и разжатия произвольного файла и расчёт коэффициента сжатия;
- 3) протестировать разработанную программу, показать, что удаётся сжимать и разжимать файлы разных форматов;
- 4) описать и обосновать полученные результаты в отчёте о выполненной лабораторной работе.