**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«Российский университет дружбы народов имени Патриса Лумумбы»**

Инженерная академия

*Кафедра механики и процессов управления*

**ОТЧЕТ**

**По Курсовой работе**

**Направление: ИУСбд-02-23**

(код направления/название направления)

**Управление информационными процессами,**

**Профиль: машинное обучение и кибербезопасность**

(название профиля)

**Тема: Алгоритмы сжатия данных: реализация кодирования Хаффмана на языке Си++**

(название лабораторной/курсовой)

**Выполнено студенткой: Самохвалова Анастасия Сергеевна**

(ФИО)

**Группа: 2**

**№студенческого: 1132233609**

Москва, 2024

**Цели работы**

1. Ознакомиться с принципами работы кодирования Хаффмана и областью его применения (ZIP, JPEG, MP3)
2. Научиться реализовывать кодирование Хаффмана на примере сжатия текста для эффективного сжатия данных без потерь
3. Изучить алгоритм построения дерева Хаффмана
4. Развить навыки работы с алгоритмами сжатия данных без потерь

**Задачи работы**

1. Реализовать кодирование Хаффмана для сжатия текста:

* определить структуру узла дерева Хаффмана, построить дерево Хаффмана на основе приоритетной очереди
* создать функцию генератора кодов (рекурсивный обход дерева), вывода кодов
* создать функцию сжатия текста с помощью кодов Хаффмана

2. Написать тестовую программу для демонстрации работы кодирования Хаффмана на примере сжатие текста (отрывка из книги)

3. определить эффективность сжатия текста алгоритмом Хаффмана.

**Теория по кодированию Хаффмана**

Кодирование Хаффмана – алгоритм сжатия данных без потерь. Основан на принципе присвоения символам кодов разной длины в зависимости от их частотности. Самым частотным символам присваивается самый короткий код, редко встречаемым – длинный. Это позволяет уменьшить общий размер данных. Сжатый код данным алгоритмом может быть декодирован однозначно и позволяет получить исходные данные без изменений. Однако для этого также должно быть соблюдено правило префиксов: код одного символа не может являться префиксом кода другого символа. Такой подход позволяет сжимать данные без потерь.

Он был разработан Дэвидом Хаффманом в 1952 году и широко применяется в различных областях, связанных с обработкой и сжатием данных.

1. Сжатие текстовых файлов: Алгоритм Хаффмана широко используется в текстовых редакторах и системах управления документами для уменьшения размера текстовых файлов, что позволяет экономить место на диске и ускорять загрузку.
2. Форматы сжатия изображений: В таких форматах, как JPEG, алгоритм Хаффмана используется для сжатия данных изображения после применения других методов, таких как дискретное косинусное преобразование (DCT).
3. Аудио и видеокодеки: Алгоритм Хаффмана применяется в различных аудио- и видеокодеках для сжатия звуковых и видеоданных. Он помогает уменьшить объем данных без значительной потери качества.
4. Передача данных по сетям: В протоколах передачи данных, таких как HTTP/2, используется кодирование Хаффмана для уменьшения объема передаваемых данных, что позволяет ускорить загрузку веб-страниц и снизить нагрузку на сеть.
5. Архиваторы: Многие архиваторы, такие как ZIP и GZIP, используют алгоритм Хаффмана в своих методах сжатия для эффективного хранения файлов

и другие области.

Пример:

В строке аабабас «а» самый частотный символ, «с» – наименее частотный;

Если не соблюдено правило префиксов:

«а»: 0

«b»: 01

«c»: 10

1. 0|0|01|0|01|0|10
2. 0|0|0|10|0|10|10

* код декодируется неоднозначно

Если дать коды одинаковой длины, запись будет длиннее:

«a»: 00

«b»: 01

«с»: 10  
«ааbаbас»: 00000000010110

Разная длина кодов и соблюдено правило префиксов:

«a»: 0

«b»: 10

«c»: 11

«aababac»: 0|0|10|0|10|0|11 – однозначно декодируется

Чтобы реализовать алгоритм Хаффмана, потребуется: проанализировать текст и подсчитать, сколько раз встречается каждый символ, символы рассортировать по частоте, объединить в бинарное дерево, начиная с самых редких. На каждом шаге два символа (или узла) с наименьшей частотой объединяются в новый узел с суммой частот.

**Вывод:**

В ходе выполнения курсовой работы было успешно реализовано кодирование Хаффмана для сжатия текста на языке C++ – алгоритма сжатия данных без потерь. Построено дерево Хаффмана: создание узла древа и приоритетной очереди, создана функция рекурсивной генерации кодов, подсчета частотности символов. Кодирование Хаффмана протестировано в функции main ( ) на примере сжатия текста

Реализация кодирования Хаффмана продемонстрировала его пользу и эффективность в решении задач, требующих сжатия данных без потерь.

**Эффективность стека**:

Алгоритм Хаффмана имеет несколько значительных преимуществ, которые делают его популярным выбором для сжатия данных:

1. Эффективность сжатия: Алгоритм Хаффмана обеспечивает хорошее сжатие данных, особенно когда некоторые символы встречаются значительно чаще других. Он использует переменную длину кодов, что позволяет более эффективно представлять часто встречающиеся символы.
2. Простота реализации: Алгоритм сравнительно прост в реализации и не требует сложных структур данных. Он может быть реализован с использованием стандартных списков и деревьев.
3. Отсутствие потерь: В отличие от некоторых методов сжатия, таких как JPEG для изображений, алгоритм Хаффмана является методом без потерь, что означает, что оригинальные данные могут быть восстановлены без каких-либо изменений.
4. . Гибкость: Алгоритм может быть применен к различным типам данных, включая текст, изображения и аудио, что делает его универсальным решением для задач сжатия.
5. Легкость декодирования: Декодирование данных, сжатых с помощью алгоритма Хаффмана, также просто и быстро, что позволяет эффективно восстанавливать оригинальные данные. Эти преимущества делают алгоритм Хаффмана одним из наиболее популярных и широко используемых методов сжатия данных в различных областях.

Несмотря на свои преимущества, имеет и ряд **недостатков**:

1. Зависимость от частот: Если частоты символов равны, выигрыша в сжатии нет.
2. Необходимость передавать таблицу кодов: Декодеру нужно знать дерево Хаффмана, иначе декодировать невозможно.
3. С помощью алгоритмов сжатия данных с потерями можно достичь более сильной степени сжатия без сильного ухудшения качества, чем с помощью кодирования Хаффмана

**Код**:











