|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Кодирование и сжатие данных методами без потерь»** | |
|  | |
| Выполнил студент группы ИКБО-07-21 | Дамарад Д.В. |
| Принял преподаватель | Скворцова Л.А. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

**СОДЕРЖАНИЕ**

[**ЦЕЛЬ РАБОТЫ** 3](#_Toc122429427)

[**ПОСТАНОВКА ЗАДАЧИ** 4](#_Toc122429428)

[**РЕШЕНИЕ** 7](#_Toc122429429)

[**Задание 1** 8](#_Toc122429430)

[**Задание 2** 9](#_Toc122429431)

[**Задание 3** 13](#_Toc122429432)

[**ВЫВОД** 19](#_Toc122429433)

# **ЦЕЛЬ РАБОТЫ**

Получение практических навыков и знаний по выполнению сжатия данных рассматриваемыми методами.

# **ПОСТАНОВКА ЗАДАЧИ**

Задание 1. Применение алгоритма группового сжатия текста RLE. Сжать текст, используя метод RLE (run length ecoding/кодирование длин серий/групповое кодирование). Требования к выполнению заданию:

1. Описать процесс сжатия алгоритмом RLE.
2. Придумать текст, в котором есть длинные (в разумных пределах) серии из повторяющихся символов. Выполнить сжатие текста. Рассчитать коэффициент сжатия.
3. Придумать текст, в котором много неповторяющихся символов и между ними могут быть серии. Выполнить групповое сжатие, показать коэффициент сжатия. Применить алгоритм разделения текста при групповом кодировании, позволяющий повысить эффективность сжатия этого текста. Рассчитать коэффициент сжатия после применения алгоритма.
4. Оформить отчет. В отчете представьте ответы на вопросы пунктов этого задания с 1 по 3.

Задание 2. Исследование алгоритмов группового сжатия (методы Лемпеля –Зива: LZ77, LZ78) на примерах. Тексты для сжатия по вариантам в табл. 1 в столбце 1 (для LZ77) и в столбце 2 (для LZ78). Требования к выполнению заданию:

1. Выполнить каждую задачу варианта задания, представив алгоритм решения в виде таблицы и указав результат сжатия.
2. Описать процесс восстановления сжатого текста.
3. Сформировать отчет, включив задание, вариант задания, результаты выполнения задания варианта (таблица).

Задание 3. Разработать программы (или только алгоритмы на псевдокоде или описать словесно) сжатия и восстановления текста методами Шеннона-Фано и Хаффмана. Представить процесс выполнения алгоритма для задачи варианта. Все результаты представить в отчете.

* 1. Сформировать отчет по разработке каждого алгоритма в соответствии с требованиями.
  2. По методу Шеннона-Фано.

1. Данными для выполнения задания является текст, представленный в табл. 1 в столбце 3.
2. Привести постановку задачи, описать алгоритм формирования префиксного дерева и алгоритм кодирования, декодирования.
3. Представить таблицу формирования кода для текста варианта задания.
4. Изобразить префиксное дерево формирования кода.
5. Рассчитать коэффициент сжатия.
   1. По методу Хаффмана.
6. Данные для выполнения задания: ваша фамилия имя отчество.
7. Привести постановку задачи, описать алгоритм формирования префиксного дерева и алгоритм кодирования, декодирования.
8. Построить таблицу частот встречаемости символов в исходной строке для чего сформировать алфавит исходной строки и посчитать количество вхождений (частот) символов и их вероятности появления.
9. Изобразить префиксное дерево Хаффмана.
10. Упорядочить построенное дерево слева-направо (при необходимости) и изобразить его.
11. Провести кодирование исходной строки по аналогии с примером:



Рисунок 1 – Пример

1. Рассчитать коэффициенты сжатия относительно кодировки ASCII и относительно равномерного кода.
2. Рассчитать среднюю длину полученного кода и его дисперсию.
3. По результатам выполненной работы сделать выводы и сформировать отчет. Отобразить результаты выполнения всех требований (с 1 по 8), предъявленных в задании и оформить разработку программы: постановка, подход к решению, код, результаты тестирования.
   1. По желанию!!! Реализовать и отладить программу. Применить алгоритм Хаффмана для архивации данных текстового файла. Выполнить практическую оценку сложности алгоритма Хаффмана. Провести архивацию этого же файла любым архиватором. Сравнить коэффициенты сжатия разработанного алгоритма и архиватора.

Таблица 1. Задание по варианту

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Сжатие данных по методу Лемпеля-Зива LZ77 | Закодировать следующую фразу, используя код LZ78 | Закодировать фразу методами Шеннона-Фано |
| 6 | 000101110110100111 | менменаменаменатеп | По-турецки говорили. Чяби, чяряби Чяряби, чяби-чяби. Мы набрали в рот воды. |

# **РЕШЕНИЕ**

Цель алгоритмов сжатия данных – уменьшить размер данных. Основной принцип алгоритмов сжатия базируется на том, что в любом файле, содержащем неслучайные данные, информация частично повторяется. Используя статистические математические модели можно определить вероятность повторения определённой комбинации символов. После этого можно создать коды, обозначающие выбранные фразы, и назначить самым часто повторяющимся фразам самые короткие коды.

Для этого используются разные техники, например, энтропийное кодирование, кодирование повторов, сжатие при помощи словаря. С их помощью 8-битный символ, или целая строка, могут быть заменены всего лишь несколькими битами, устраняя таким образом излишнюю информацию.

Существующие алгоритмы сжатия данных можно разделить на два больших класса – с потерями, и без. Алгоритмы с потерями обычно применяются для сжатия изображений и аудио. Эти алгоритмы позволяют достичь больших степеней сжатия благодаря избирательной потере качества. Однако, по определению, восстановить первоначальные данные из сжатого результата невозможно.

**Сжатие с потерями** означает, что в процессе сжатия была потеряна часть информации. Алгоритмы сжатия с потерями стараются сделать так, чтобы потерялись только те данные, которые не слишком важны. Представьте, что сжатие с потерями — это краткий пересказ произведения из школьной программы: школьнику не так важны описания природы и авторский стиль, ему главное сюжет. Краткий пересказ сохранил только важное, но передал это намного быстрее. Алгоритмы с потерями обычно применяются для сжатия изображений и аудио. Эти алгоритмы позволяют достичь больших степеней сжатия благодаря избирательной потере качества. Однако, по определению, восстановить первоначальные данные из сжатого результата невозможно.

**Сжатие без потерь** — это когда уменьшается размер файла, при этом, не теряя в качестве. Для этого используются интересные математические приёмы и кодирование. Главное — чтобы при раскодировании все данные остались на месте. Алгоритмы сжатия без потерь применяются для уменьшения размера данных, и работают таким образом, что возможно восстановить данные в точности такими, какие они были до сжатия. Они применяются в коммуникациях, архиваторах и некоторых алгоритмах сжатии аудио и графической информации.

# **Задание 1**

Алгоритм RLE (Run-Length Encoding) – это алгоритм сжатия с потерями. Суть алгоритма: группы подряд идущих одинаковых байтов (символов) в поступающей последовательности символов, заменяют на пары вида, где счетчик определяет количество одинаковых символов (значений) в группе. Этот подход к сжатию уменьшает избыточность данных.

Группы одинаковых символов большой длины присутствуют при кодировании изображений, когда большое количество пикселей имеют один и тот же цвет. В памяти коды цвета следуют последовательно, образуя группы.

Выполним сжатие текста qqqqqqqqqqqwwwwwwwwwwwweeeeeeeeee и расчитаем коэффициент сжатия:

Исходная строка: qqqqqqqqqqqwwwwwwwwwwwweeeeeeeeee

Текущий размер: 33 byte

Результат кодирования: 11q12w10e

Полученный размер: 9 byte

Коэффициент сжатия: 0.272727

Выполним сжатие текста qwertyasdzxcffffffff и расчитаем коэффициент сжатия:

Исходная строка: qwertyasdzxcffffffff

Текущий размер: 20 byte

Результат кодирования: 1q1w1e1r1t1y1a1s1d1z1x1c8f

Полученный размер: 26 byte

Коэффициент сжатия: 1.3

Применить алгоритм разделения текста при групповом кодировании, позволяющий повысить эффективность сжатия этого текста и расчитаем коэффициент сжатия:

Исходная строка: qwertyasdzxcffffffff

Текущий размер: 20 byte

Результат кодирования: -12qwertyasdzxc8f

Полученный размер: 17 byte

Коэффициент сжатия: 0.85

# **Задание 2**

Метод Шифровки LZ77 – это простой, чёткий и легко реализуемый алгоритм сжатия данных без потерь, использующий повторяемость подстрок.

Его главная мысль чрезвычайно проста: когда ранее прочитанная подстрока встречается повторно, алгоритм пишет вместо неё ссылку на её предыдущее вхождение.

Алгоритм LZ77 (Зив и Лемпель [1977]): на каждом шаге формируется тройка вида (d, l, s), что означает, повторяется подстрока длины l, ранее встречавшаяся - d символов назад, а потом идёт следующий символ текста. Например, текст abbbbbbaaaa, после сжатия (0,0,a)(0,0,b)(1,5,a)(1,2,a).

Допускается l>d, в этом случае кодируется подстрока с периодическим окончанием. В конкретных реализациях всё это представляется по-разному, например, разделяются (d,l) и s. Т.е. сжатый текст выглядит так: (0,0)a(0,0)b(1,5)a(1,2)a. Основная идея этого метода (его еще часто называют методом LZ1, [Ziv 77]) состоит в использовании ранее прочитанной части входного файла в качестве словаря.

Сжатие последовательности битов по алгоритму LZ77

Первоначально каждому символу алфавита присваивается определенный код (коды - порядковые номера, начиная с 0).

* 1. Выбирается первый (один) символ сообщения и заменяется на его код.
  2. Выбираются следующие два символа и заменяются своими кодами.
  3. При этом комбинации двух символов присваивается свой код. Обычно это номер, равный числу уже использованных кодов. Так, если алфавит 298 включает 8 символов, имеющих коды от 000 до 111, то первая двух символьная комбинация получит код 1000, следующая - код 1001 и т.д.
  4. Выбираются из исходного текста очередные 2, 3,...N символов до тех пор, пока не образуется еще не встречавшаяся комбинация. Тогда этой комбинации присваивается очередной код, и поскольку совокупность А из первых N-1 символов уже встречалась, то она имеет свой код, который и записывается вместо этих N-1 символов. е. можно представить формирование кода в этом случае так: хххххххх код из N(=8) символов выбран из кодируемой последовательности, тогда если для первых N-1 символов уже был сформирован код, то заменяем эти N-1 символы на их код. Каждый акт введения нового кода назовем шагом кодирования.
  5. Процесс продолжается до исчерпания исходного текста

Выполним сжатие текста 000101110110100111 и рассчитаем коэффициент сжатия:

Таблица 2. Кодирование методом LZ77

|  |  |  |
| --- | --- | --- |
| Содержимое буфера поиска (сжимаемый текст) | Содержимое упреждающего буфера | Код символов |
|  | 000101110110100111 | (0, 0, 0) |
| 0 | 00101110110100111 | (1, 2, 1) |
| 0001 | 01110110100111 | (2, 2, 1) |
| Содержимое буфера поиска (сжимаемый текст) | Содержимое упреждающего буфера | Код символов |
| 0001011 | 10110100111 | (4, 4, 0) |
| 000101110110 | 100111 | (2, 2, 0) |
| 000101110110100 | 111 | (3, 1, 1) |
| 00010111011010011 | 1 | (1, 1, ) |
| 000101110110100111 |  |  |

Результат сжатия: (0, 0, 0) (1, 2, 1) (2, 2, 1) (4, 4, 0) (2, 2, 0) (3, 1, 1) (1, 1, ).

Коэффициент сжатия: 0.388889

Распаковка сжатого текста:

Таблица 3. Дешифровка LZ77

|  |  |
| --- | --- |
| Получившийся код | Исходная строка |
| **0** | (0, 0, 0) |
| 000**1** | (1, 2, 1) |
| 0001**011** | (2, 2, 1) |
| 0001011**10110** | (4, 4, 0) |
| 000101110110**100** | (2, 2, 0) |
| 000101110110100**11** | (3, 1, 1) |
| 00010111011010011**1** | (1, 1, ) |

В отличие от LZ77, работающего с уже полученными данными, LZ78(от 1978 года) ориентируется на данные, которые только будут получены (LZ78 не использует скользящее окно), он хранит словарь из уже просмотренных фраз.

Алгоритм считывает символы сообщения до тех пор, пока накапливаемая подстрока входит целиком в одну из фраз словаря.

Как только эта строка перестанет соответствовать хотя бы одной фразе словаря, алгоритм генерирует код, состоящий из индекса строки в словаре, которая до последнего введенного символа содержала входную строку, и символа, нарушившего совпадение.

Затем в словарь добавляется введенная подстрока. Если словарь уже заполнен, то из него предварительно удаляют менее всех используемую в сравнениях фразу. Если в конце алгоритма мы не находим символ, нарушивший совпадения, то тогда мы выдаем код в виде (индекс строки в словаре без последнего символа, последний символ).

Выполним сжатие текста менменаменаменатеп и рассчитаем коэффициент сжатия:

Таблица 4. Кодирование методом LZ78

|  |  |  |  |
| --- | --- | --- | --- |
| Содержимое словаря | Номер фразы | Содержимое считанной строки | Коды |
|  | 1 | м | (0, м) |
| м,е | 2 | е | (0, е) |
| м,е,н | 3 | н | (0, н) |
| м,е,н, ме | 4 | ме | (1, е) |
| м,е,н, ме, на | 5 | на | (3, а) |
| м,е,н, ме, на, мен | 6 | мен | (4, н) |
| м,е,н, ме, на, мен, a | 7 | a | (0, а) |
| м,е,н, ме, на, мен, a, мена | 8 | мена | (6, а) |
| м,е,н, ме, на, мен, a, мена, т | 9 | т | (0, т) |
| м,е,н, ме, на, мен, a, мена, т, еп | 10 | еп | (2, п) |

Результат сжатия: (0, м) (0, е) (0, н) (1, е) (3, а) (4, н) (0, а) (6, а) (0, т) (2, п).

Коэффициент сжатия: 0.555556

Таблица 5. Дешифровка LZ78

|  |  |  |
| --- | --- | --- |
| Получившийся код | Исходная строка | Таблица префиксов |
| м | (0, м) | м |
| Получившийся код | Исходная строка | Таблица префиксов |
| ме | (0, е) | м е |
| мен | (0, н) | м е н |
| менме | (1, е) | м е н ме |
| менмена | (3, а) | м е н ме на |
| менменамен | (4, н) | м е н ме на мен |
| менменамена | (0, а) | м е н ме на мен а |
| менменаменамена | (6, а) | м е н ме на мен а мена |
| менменаменаменат | (0, т) | м е н ме на мен а мена т |
| менменаменаменатеп | (2, п) | еп |

# **Задание 3**

Это один из первых алгоритмов сжатия, который впервые сформулировали американские учёные Шеннон и Фано.

Исходными данными для алгоритма является текстовое сообщение, записанное символами некоторого алфавита.

Результатом алгоритма является таблица с кодами каждого символа. Код символа представляет последовательность из нулей и единиц разной длины. Код обладает префиксным свойством, т.е. код одного символа не является префиксом кода другого символа. Алгоритм основан на частоте повторения символов.

Так, часто встречающийся символ кодируется кодом меньшей длины, а редко встречающийся — кодом большей длины.

Так как коды символов, полученные при кодировании, префиксные, это и позволяет однозначно декодировать любую последовательность кодовых слов.

Алгоритм Шеннона-Фано.

На вход поступает сообщение (текст), который требуется сжать.

Шаг 1. Формируется таблица частот символов, составляющих алфавит текста. Таблица состоит из строк вида: символ, частота появления символа в тексте, упорядоченных по убыванию частот.

Шаг 2. Находится середина таблицы, которая делит алфавит примерно на две равные части. Середина определяется так: отсчитываются строки таблицы сверху вниз, так чтобы и сверху, и снизу суммы частот были равны. Получаем как бы две части таблицы: левая (верхняя часть таблицы), правая (нижняя часть таблицы). Для символов левой части присваивается «0», для правой «1».

Шаг 2 повторяется до тех пор, пока мы не получим единственный элемент последовательности. Полученная таблица хранит бинарное дерево, сформированное по алгоритму Фано. Содержание таблицы можно отобразить в форме бинарного дерева, представляя левые части таблицы как левое поддерево, а правые как правое поддерево. Записывая на левых ребрах код 0, а на правых ребрах код 1.

Выполним сжатие текста “По-турецки говорили. Чяби, чяряби Чяряби, чяби-чяби.” и рассчитаем коэффициент сжатия. Составим таблицу формирования кода и покажем префиксное дерево.

Таблица 6. Формирование кода

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | Кол-во | 1-ый бит | 2-ой  бит | 3-ий  бит | 4- ый  бит | 5- ый  бит | Код | Кол-во бит |
| П | 1 | 0 | 0 | 1 | 0 | 1 | 00101 | 5 |
| Ч | 2 | 0 | 1 | 0 | 1 | 1 | 01011 | 10 |
| и | 8 | 1 | 1 | 1 |  |  | 111 | 24 |
| ч | 3 | 1 | 1 | 0 | 1 |  | 1101 | 12 |
| у | 1 | 1 | 1 | 0 | 0 | 1 | 11001 | 5 |
| к | 1 | 1 | 1 | 0 | 0 | 0 | 11000 | 5 |
| т | 1 | 1 | 0 | 1 | 0 | 1 | 10101 | 5 |
| о | 3 | 1 | 0 | 0 | 1 |  | 1001 | 12 |
| е | 1 | 1 | 0 | 0 | 0 | 0 | 10000 | 5 |
| Символ | Кол-во | 1-ый бит | 2-ой  бит | 3-ий  бит | 4- ый  бит | 5- ый  бит | Код | Кол-во бит |
| ц | 1 | 0 | 1 | 0 | 1 | 0 | 01010 | 5 |
| л | 1 | 0 | 1 | 0 | 0 | 0 | 01000 | 5 |
| б | 5 | 0 | 0 | 1 | 1 |  | 0011 | 20 |
| г | 1 | 0 | 0 | 1 | 0 | 0 | 00100 | 5 |
| р | 4 | 0 | 0 | 0 | 1 |  | 0001 | 16 |
| я | 7 | 0 | 1 | 1 |  |  | 011 | 21 |
| в | 1 | 1 | 0 | 1 | 0 | 0 | 10100 | 5 |
| проб | 5 | 1 | 0 | 1 | 1 |  | 1011 | 20 |
| - | 2 | 1 | 0 | 0 | 0 | 1 | 10001 | 10 |
| , | 2 | 0 | 1 | 0 | 0 | 1 | 01001 | 10 |
| . | 2 | 0 | 0 | 0 | 0 |  | 0000 | 8 |

Объем незакодированной фразы – 52\*8 бит = 416 бит.

Объем закодированной фразы –208 бит.

Коэффициент сжатия: 208/416=0,5

Исходная строка: По-турецки говорили. Чяби, чяряби Чяряби, чяби-чяби.

Результат кодирования: 0010110011000110101110010001100000101011000111101100100100110100100100011110100011100001011010110110011111010011011110101100010110011111101101011011000101100111110100110111101011001111110001110101100111110000

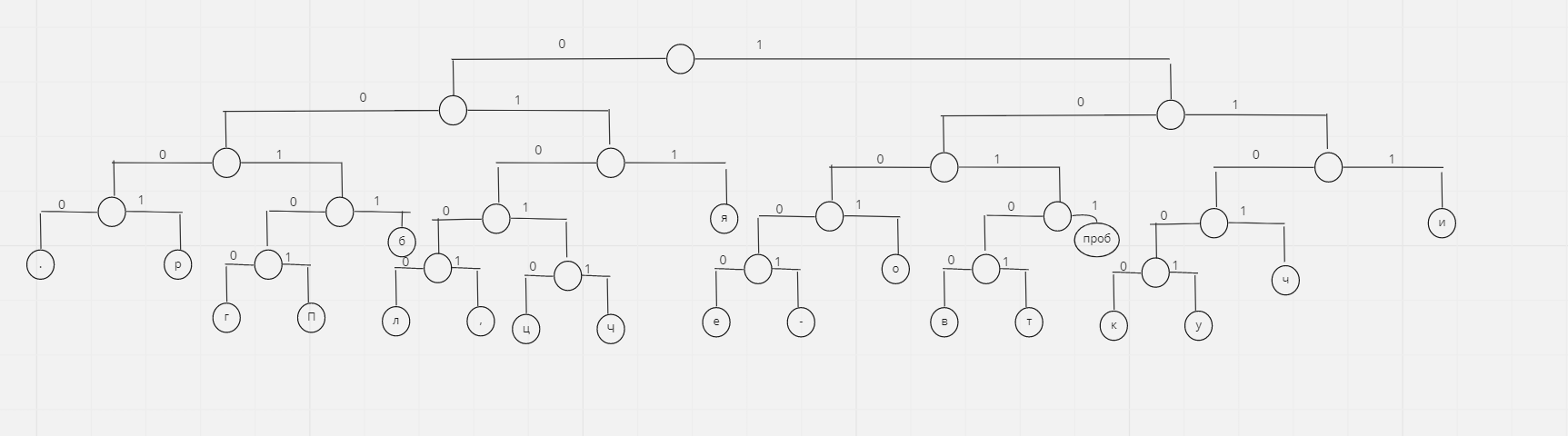


Рисунок 2 - Префиксное дерево Шеннона-Фано

Метод Хаффмана - Жадный алгоритм оптимального префиксного кодирования алфавита с минимальной избыточностью. Был разработан в 1952 году Дэвидом Хаффманом. В настоящее время используется во многих программах сжатия данных. Это алгоритм сжатия данных, который формулирует основную идею сжатия файлов.

Метод основывается на создании бинарных деревьев. В нем узел может быть либо листом, либо внутренним. Изначально все узлы считаются листьями, которые представляют сам символ и его вес (то есть частоту появления). Внутренние узлы содержат вес символа и ссылаются на два узла-наследника. По общему соглашению, бит «0» представляет следование по левой ветви, а «1» — по правой. В полном бинарном дереве N листьев и N-1 внутренних узлов. Рекомендуется, чтобы при построении дерева Хаффмана отбрасывались неиспользуемые символы для получения кодов оптимальной длины.

Кодовое дерево (дерево кодирования Хаффмана, Н-дерево) – это бинарное дерево, у которого: листья помечены символами, для которых разрабатывается кодировка; узлы (в том числе корень) помечены суммой вероятностей появления всех символов, соответствующих листьям поддерева, корнем которого является соответствующий узел.

Алгоритм построения кодового оптимального дерева.

Оптимальное дерево – так как коды символов будут минимальными по длине. Символы, частота которых большая, должны иметь код минимального размера.

На вход поступает текст, который надо сжать. Результатом должна быть таблица кодов символов. Коды символов будут представлены последовательностью битов.

Описание алгоритма.

Будем использовать очередь с приоритетами для построения дерева Хаффмана, где узлу с наименьшей частотой будет присвоен высший приоритет. Шаги построения:

* 1. Создать узел-лист для каждого символа и добавить их в очередь с приоритетами.
  2. Пока в очереди больше одного листа повторять следующее: − удалить два узла с наивысшим приоритетом (с самой низкой частотой) из очереди; − создать новый внутренний узел, где эти два узла будут наследниками, а частота появления будет равна сумме частот этих двух узлов. − Добавьте новый узел в очередь приоритетов.
  3. Единственный оставшийся узел будет корневым, на этом построение дерева закончится.

Применим метод Хаффмана на собственном ФИО. Также рассчитаем коэффициент сжатия, cоставим таблицу формирования кода и покажем префиксное дерево.

Таблица 7. Таблица алфавита, частот вхождений, вероятность появления и код символа

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | Кол-во | 1-ый бит | 2-ой  бит | 3-ий  бит | 4- ый  бит | 5- ый  бит | Код | Кол-во бит |
| пробел | 2 | 1 | 0 | 0 | 0 |  | 1000 | 8 |
| Д | 2 | 1 | 1 | 1 | 0 |  | 1110 | 8 |
| В | 1 | 1 | 1 | 1 | 1 | 0 | 11110 | 10 |
| с | 1 | 1 | 1 | 1 | 1 | 1 | 11111 | 10 |
| а | 5 | 0 | 1 |  |  |  | 01 | 10 |
| и | 4 | 1 | 1 | 0 |  |  | 110 | 12 |
| ч | 1 | 1 | 0 | 1 | 1 | 1 | 10111 | 10 |
| е | 1 | 1 | 0 | 1 | 1 | 0 | 10110 | 10 |
| л | 2 | 1 | 0 | 1 | 0 |  | 1010 | 8 |
| ь | 1 | 1 | 0 | 0 | 1 | 1 | 10011 | 10 |
| д | 1 | 1 | 0 | 0 | 1 | 0 | 10010 | 10 |
| р | 1 | 0 | 0 | 1 | 1 |  | 0011 | 4 |
| в | 1 | 0 | 0 | 1 | 0 |  | 0010 | 4 |
| Символ | Кол-во | 1-ый бит | 2-ой  бит | 3-ий  бит | 4- ый  бит | 5- ый  бит | Код | Кол-во бит |
| н | 1 | 0 | 0 | 0 | 1 |  | 0001 | 4 |
| м | 1 | 0 | 0 | 0 | 0 |  | 0000 | 4 |

Объем незакодированной фразы – 25\*8 бит = 200 бит.

Объем закодированной фразы –112 бит.

Коэффициент сжатия: 112/200=0,56

Исходная строка: Дамарад Даниил Васильевич

Результат кодирования: 11100100000100110110010100011100100011101101010100011110011111111010101001110110001011010111

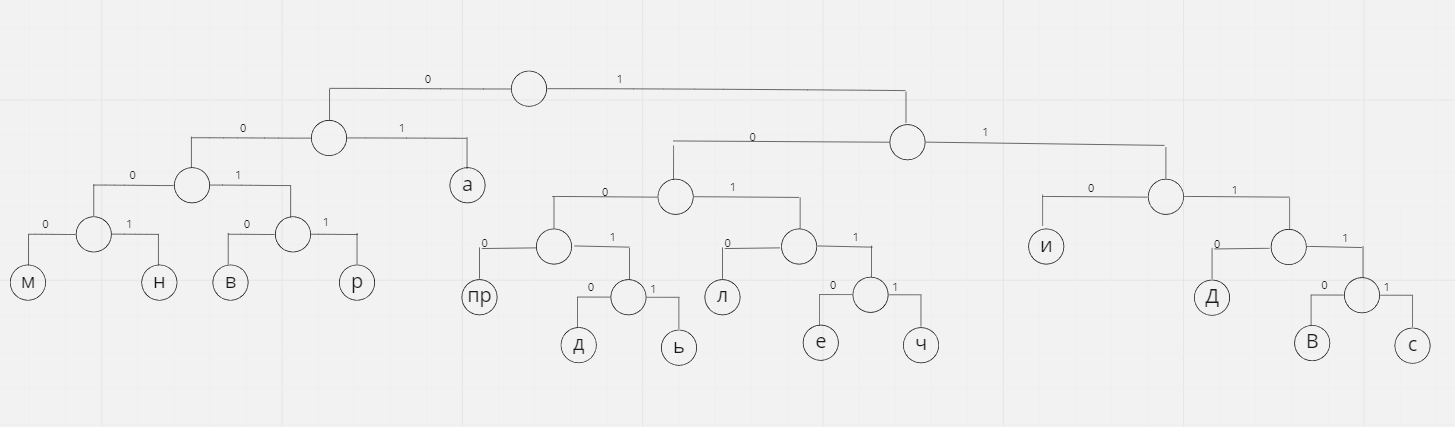


Рисунок 3 - Префиксное дерево

# **ВЫВОД**

В результате выполнения работы были получены практические навыки и знания по выполнению сжатия данных методами RLE, LZ77, LZ78, Шеннона-Фано, Хаффмана.