Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Сжатие/распаковка данных на основе статистических методов**

Студент: Дрозд А. И.

ФИТ 3 курс 2 группа

Преподаватель:

Нистюк Ольга Александровна

Минск 2024

**Цель**: приобретение практических навыков использования статистических методов Шеннона − Фано и Хаффмана (ShannonFano and Huffman coding) для сжатия/распаковки данных.

Задачи:

1. Закрепить теоретические знания по алгебраическому описанию и использованию методов сжатия/распаковки (архивации/ разархивации) данных на основе методов Шеннона − Фано и Хаффмана.
2. Разработать приложение для реализации методов Шеннона-Фано и Хаффмана.
3. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

# **Теоретические сведения**

Статистические алгоритмы позволяют создавать более короткие коды для часто встречающихся и более длинные – для редко встречающихся символов алфавита или конкретного сообщения. В первом случае метод считается статическим статистическим, во втором – динамическим статистическим: вероятностные свойства символов подсчитываются для конкретного сообщения или потока данных.

За счет использования для каждого значения байта кодов ASCII (символа алфавита) кода различной длины в соответствии с частостью (вероятностью появления этого символа в сообщении) можно значительно уменьшить общий размер данных. Эта базовая идея лежит в основе алгоритмов статистических (вероятностных) методов сжатия: Шеннона − Фано и Хаффмана.

Частота или вероятность появления того или иного символа алфавита в произвольном сообщении, лежащая в основе алгоритмов, дали название этим алгоритмам и соответствующим методам. Иногда эти методы называют также префиксными. К примеру, если имеется некоторый код, который записывается как Х1 = А1А2, и другой код – Х2 = А1, то говорят, что Х2 является префиксом Х1. Или если Х1 = 1010, а Х2 = 10101100, то Х2 также является префиксом Х1. Таким образом, использование описываемых методов предусматривает создание кодовой таблицы (подобно кодам ASCII или base64). Формально процедура сжатия (прямое преобразование) состоит в подстановке соответствующего бинарного кода вместо символа исходного алфавита и наоборот – при обратном преобразовании.

Код Шеннона – Фано не является оптимальным (обеспечивает минимальную избыточность) в общем смысле, хотя и дает оптимальные результаты при некоторых распределениях вероятностей. Для одного и того же распределения вероятностей можно построить, вообще говоря, несколько кодов Шеннона – Фано, и все они могут дать различные результаты.

Итак, необходимо выполнить следующие действия:

1) подсчитать вероятностные параметры символов алфавита А = {ai} (реализуется статическая версия алгоритма);

2) отсортировать – обычно в порядке убывания (невозрастания, т. е. могут иметь место повторяющиеся значения) вероятностей р(аi); р(аi) – вероятность появления в сжимаемом сообщении на произвольной позиции символа аi алфавита, т. е. создать таблицу символов алфавита, на основе которого генерируется сжимаемое сообщение;

3) каждому символу отсортированного множества поставить в соответствие бинарный код, для чего это множество (таблица) символов делится на две группы таким образом, чтобы каждая из групп имела приблизительно одинаковую суммарную частоту (вероятность). Очевидно, на первом шаге такая суммарная вероятность в каждой из групп должна быть максимально близка к 0,5. Первому из полученных подмножеств устанавливается первый символ бинарного кода: 0, второй − 1 (или наоборот). Для вычисления следующих битов кодов данная процедура повторяется рекурсивно для каждого из полученных на текущем шаге подмножеств, в котором содержится больше одного символа. Получим таблицу, в которой длина кодовых комбинаций меняется от минимального (lmin) до максимального (lmax) значений.

Пример. Имеем алфавит А{ai}, i = [1, …, 10], N(A) = 10 – мощность алфавита. При этом получены следующие вероятности для символов алфавита представлены на рисунке 1.1.

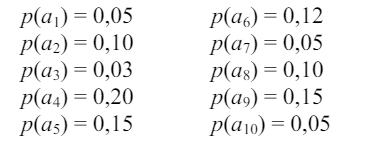


Рисунок 1.1 – Нахождение вероятностей появления алфавита

Далее рассмотрим процесс создания таблицы кодов. Отсортируем таблицу символов в порядке убывания вероятностей. Разделим ее на две части (два подмножества), как показано ниже (отделены горизонтальной линией). Видно, что сумма вероятностей в обоих подмножествах одинакова и равна 0,5. Символам верхней части общей таблицы определим старший символ кода (1), нижней части – 0. Пример изображён на рисунке 1.2.

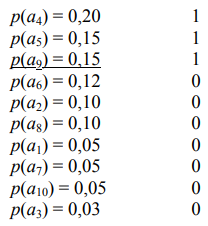
****

Рисунок 1.2 – Создание таблицы кодов

Далее в качестве исходного рассматриваем каждое из двух подмножеств (на текущем шаге). Выполняем рекурсивно одну и ту же процедуру. В частности, для первых трех символов таблицы (для первого подмножества) имеем после следующей итерации (ее деления на два меньших подмножества и определения следующих бинарных символов кода). Пример представлен на рисунке 1.3.

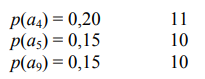
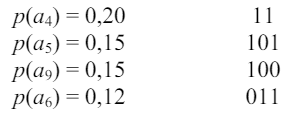


Рисунок 1.3 – Первое подмножество

Далее переходим к кодированию символов после первого деления исходной таблицы (начиная с символа р(а6)). Выполнив стандартные операции, получим таблицу бинарных кодов, как показано ниже на рисунке 1.4.



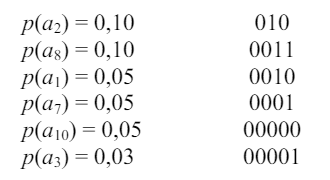


Рисунок 1.4 – Кодирование символов

Как видим, для рассмотренного примера получена минимальная длина кода, равная 2 битам (lmin = 2), и максимальная длина, равная 5 битам (lmax = 5). Действительно, символам с большими вероятностями соответствуют коды меньшей длины (lmin = 2) и наоборот (lmax = 5). Замечаем, что выполнено основное требование: все кодовые комбинации разные. И соблюдено «требование префикса»: ни одна из кодовых комбинаций меньшей длины не является началом кодовой комбинации большей длины.

 Именно последняя таблица используется в неизменном виде (речь о кодах) в процессах прямого и обратного преобразований. Алгоритм прямого преобразования: необходимо выполнить одну операцию: заменить символы входного сообщения соответствующими бинарными кодами. Алгоритм обратного преобразования: на входе – сообщение в виде бинарной последовательности.

Шаг 1. Анализируются lmin начальных бинарных символов: осуществляется поиск в таблице соответствующего совпадения. Если такое будет найдено, то на выходе будет символ исходного алфавита с совпадающим кодом. После этого процедура повторяется, т. е. анализируются очередные lmin символов. Если не найдено в таблице совпадения, переходим к шагу 2.

Шаг 2. Длина анализируемой последовательности увеличивается на 1 бит: lmin + 1. Осуществляется поиск совпадающей бинарной комбинации такой же длины в таблице. Если такая комбинация существует, на выходе распаковщика формируется соответствующий символ исходного алфавита, если нет – длина анализируемой последовательности увеличивается еще на один бит, и т. д.

По различным причинам при анализе очередной последовательности длиной lmax совпадение в таблице может быть не найдено. Для нейтрализации подобных коллизий архиваторы содержат средства контроля ошибок с помощью корректирующих кодов.

Пример. Прямое преобразование. Итак, требуется сжать сообщение Х = «а1а1а9а9а9а5а1а5а5а1» (10 символов). Заменяя символы соответствующими им кодами из таблицы, на выходе получим: Хn = 0010001010010010010100101011010010.

Как видим, общая длина (объем после сжатия Vпс) сообщения составляет 34 бита. Если бы каждый символ сообщения Х заменялся некоторым кодом, подобным ASCII, то длина его составила бы 80 битов (10 · 8).

Обратное преобразование. На входе имеем бинарную последовательность Yn = 0010001010010010010100101011010010.

Шаг 1: анализируются начальные 2 (lmin = 2) символа этой последовательности: 00. Совпадающих комбинаций в таблице нет.

 Шаг 2: длину анализируемой последовательности увеличиваем на один бит: 001. Совпадение не найдено.

Шаг 3: анализируем 4-битовую комбинацию: 0010. Этой комбинации в таблице соответствует символ исходного алфавита «а1». На выходе распаковщика будет именно этот символ.

Анализируется очередные 2 символа (lмин = 2): 00 (шаг 1), и т. д.

Теперь предположим, что во входном сообщении изменился только один символ (первый): Yn = 1010001010010010010100101011010010. А распаковщик не содержит средств для ее обнаружения.

В таком случае первой совпадающей комбинацией будет 101: «а5». Следующей – «а7», за ней – «а2», и т. д. Следуя логике рассуждений, нетрудно подсчитать коэффициент компрессии для нашего примера: R1 = 34 / 80 либо R2 = (80 − 34) / 80 = 46 / 80.

Метод основан на алгоритме оптимального префиксного кодирования алфавита: исходный алгоритм Хаффмана является оптимальным для посимвольного кодирования с известным входным распределением вероятностей, т. е. для отдельного кодирования несвязанных символов в таком потоке данных. Отличается от метода Шеннона – Фано лишь в части кодирования символов исходного алфавита.

В данном случае бинарные коды создаются на основе дерева, ветви которого обозначаются бинарными символами.

Бинарным кодом символа исходного алфавита будет последовательность обозначений ветвей дерева от корня до листа, соответствующего этому символу. В основе бинарного кода лежит следующее положение.

Лемма. Для любого заданного алфавита (источника) с N > 2 символами существует оптимальный двоичный код, в котором два наименее вероятных символа (слова) имеют одну и ту же длину и отличаются лишь последним битом [31]. Построение дерева начинается с сортирования символов исходного алфавита в порядке убывания (невозрастания). Далее выбираются два символа (ai, aj) с наименьшими вероятностями (р(ai), р(aj)) и объединяются в узел. Ветви этого узла обозначаются «1» и «0». Этот узел рассматривается далее как новый, виртуальный символ (aij), которому будет соответствовать вероятность р(aij) = р(ai) + р(aj). Такой виртуальный символ будет рассматриваться далее наравне с остальными символами исходного алфавита. Два его потомка из дальнейшего рассмотрения исключаются. Создаются новые узлы дерева по тому же принципу. Корень дерева образуют два символа с наибольшими вероятностями.

**Практическое задание**

Задание: разработать авторское приложение в соответствии с целью лабораторной работы.

С помощью приложения выполнить прямое и обратное преобразования сообщения, состоящего из собственных имени и фамилии. Можно использовать любой из известных методов сортировки символов массива.

 Метод кодировки (Шеннона − Фано, Хаффмана) использовать по указанию преподавателя. При этом таблица отсортированных символов строится:

а) на основе данных, полученных в лабораторной работе № 2;

б) динамически, на основе анализа сжимаемого сообщения.

Определить эффективность (в сравнении с кодами ASCII) сжатия сообщения.

Исходя из основного задания, первоначально нам необходимо реализовать сжатие с помощью метода Шеннона-Фано. Для этого мы передаем нашей программе строку, состоящую из имени и фамилии: «Дрозд Алексей». Результат показан на рисунке 2.1.



Рисунок 2.1 – Исходное сообщение для сжатия

После этого нам необходимо высчитать частоту и вероятности появления каждого символа в нашем сообщении. Таблица вероятностей продемонстрирована на рисунке 2.2.

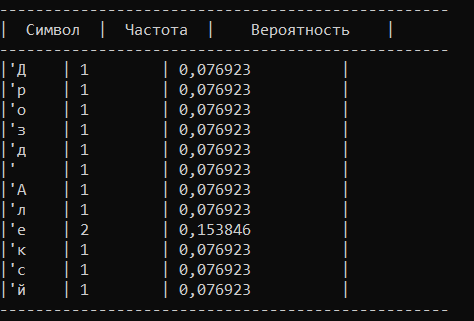


Рисунок 2.2 – Таблица вероятностей исходного сообщения

Следующим шагом необходимо отсортировать символы по убыванию значения вероятности их появления. Данное действие показано на рисунке 2.3.

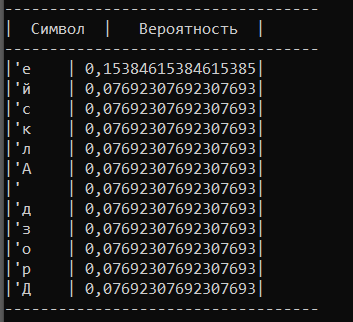


Рисунок 2.3 – Сортировка таблицы вероятностей

Далее необходимо применить алгоритм сжатия Шеннона-Фано для вычисления таблицы соответствий символов и их сжатых кодов. Таблица кодов показана на рисунке 2.4.

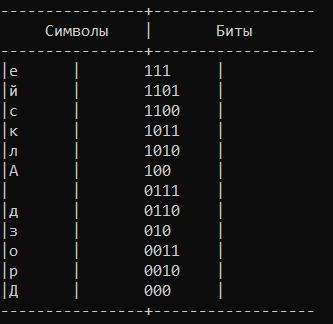


Рисунок 2.4 – Таблица кодов соответствия

В конечном счете мы получаем исходное сообщение в виде бинарного алфавита на основе построенной таблицы. Результаты прямого и обратного преобразования с использованием метода Шеннона-Фано и перевода исходного сообщения в ASCII представлен на рисунке 2.5.

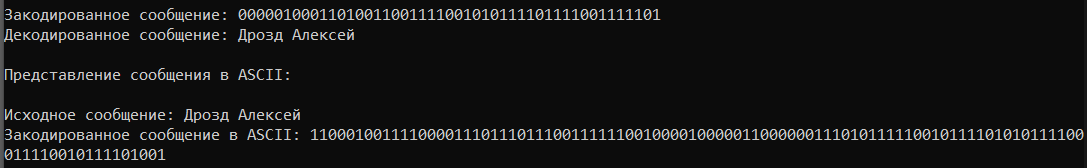


Рисунок 2.5 – Итог применения алгоритма и исходное сообщение в ASCII

В конце нам необходимо определить эффективность сжатия нашего исходного сообщения в сравнении с таблицей ASCII. Для этого необходимо подсчитать общее количество символов в итоговых сообщениях после применения метода Шеннона-Фано и перевода в ASCII. Результат на рисунке 2.6.

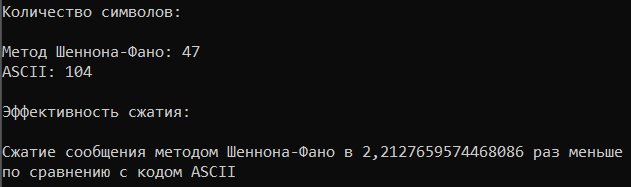


Рисунок 2.6 – Определение эффективности сжатия

Во втором случае мы реализовываем алгоритм Хаффмана, суть которого была описана в теоретических сведениях. Данный алгоритм имеет такие же два первоначальных шага, как и метод Шеннона-Фано, поэтому опустим процесс вычисления вероятностей и сортировки символов, и просто перейдем к этапу применения метода Хаффмана. Таблица символов показана на рисунке 2.7.

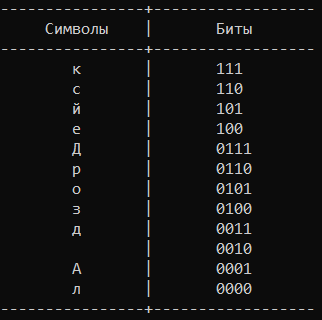


Рисунок 2.7 – Таблица соответствия символов по методу Хаффмана

Соответственно также необходимо найти эффективность применения метода сжатия по примеру, который был показан ранее. Результат нахождения на рисунке 2.8.

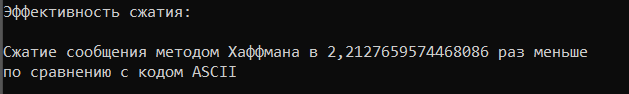


Рисунок 2.8 – Эффективность сжатия методом Хаффмана

**Вывод:**

В данной лабораторной работе сравнивались два метода сжатия данных, основанных на статистических подходах: метод Шеннона-Фано и метод Хаффмана. Оба этих метода направлены на эффективное кодирование символов алфавита с целью уменьшения объема передаваемой информации. Метод Шеннона-Фано и метод Хаффмана представляют разные стратегии построения бинарных кодов для символов алфавита. Метод Шеннона-Фано использует подход разделения алфавита на группы с близкими вероятностями и последующего создания кодов, в то время как метод Хаффмана строит оптимальное дерево кодирования путем объединения символов с наименьшими вероятностями.

Оба метода имеют свои преимущества и ограничения. Например, метод Шеннона-Фано может давать различные результаты для одинаковых вероятностей, в то время как метод Хаффмана создает оптимальные коды при известном распределении вероятностей. Более того, метод Хаффмана часто обеспечивает более компактные коды по сравнению с методом Шеннона-Фано.