Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Сжатие/распаковка данных арифметическим методом**

Студент: Дрозд А. И.

ФИТ 3 курс 2 группа

Преподаватель:

Нистюк Ольга Александровна

Минск 2024

**Цель**: приобретение практических навыков использования арифметических методов сжатия/распаковки данных.

Задачи:

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

**Теоретические сведения**

При арифметическом сжатии (кодировании) текст представляется вещественными числами в интервале от 0 до 1. По мере анализа текста отображающий его интервал уменьшается, а количество битов для его представления возрастает. Очередные символы текста сокращают величину интервала, исходя из значений соответствующих вероятностей.

Основная идея арифметического метода сжатия заключается в том, чтобы присваивать коды не отдельным символам, а их последовательностям.

Таким образом, как и во всех энтропийных алгоритмах, исходной является информация о частоте встречаемости каждого символа алфавита.

Алгоритмы прямого и обратного преобразований базируются на операциях с «рабочим отрезком».

Рабочим отрезком называется интервал [a; b] с расположенными на нем точками. Причем точки расположены таким образом, что длины образованных ими отрезков пропорциональны (или равны) частоте (вероятности) появления соответствующих символов.

Дана последовательность Xk = «молоко» представленная на рисунке 1.1. Рассчитываем статистику.

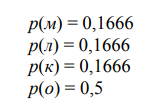


Рисунок 1.1 – Вероятности для каждого символа

Шаг 0: строим основной рабочий отрезок, как показано на рис. 1.2.



Рисунок 1.2 – Первый отрезок с учётом вероятностей

Как показано на рис. 1.2, при инициализации алгоритма a = 0, b = 1 (a0 = 0, b0 = 1). Прямое преобразование (сжатие). Один шаг сжатия (кодирования) заключается в простой операции: берется кодируемый символ, для него ищется соответствующий участок на рабочем отрезке. Найденный участок становится новым рабочим отрезком. Его тоже необходимо разбить с помощью точек.

Это и последующие разбиения отрезка (на шаге i) подразумевают определение новых значений верхней (Hi) и нижней (Li) границ для всего участка и осуществляются по следующим правилам, представленным на рисунке 1.3.



Рисунок 1.3 – Формулы определения верхней и нижней границы

Где αj – j-й символ сжимаемой последовательности, Li − 1 и Hi − 1 – соответственно нижняя и верхняя границы рабочего отрезка на (i − 1)-м шаге, L(αj)0 и H(αj)0 – соответственно исходные нижняя и верхняя границы символа αj.

Шаг 1: в нашем примере на первом шаге берется первый символ последовательности: «м» (α1 = «м») и ищется соответствующий участок на рабочем отрезке. Легко понять по рис. 1.4, что этот участок соответствует интервалу [0; 0,1666]. Сам рисунок представлен ниже. Он становится новым рабочим отрезком и опять разбивается согласно статистике и соотношениям.



Рисунок 1.4 - Разметка и точки рабочего отрезка после анализа первого символа (м) последовательности «молоко»

Шаг 2: анализируем следующий символ входной последовательности (о). На новом рабочем отрезке этому символу соответствует интервал [a2 = L2 = 0,083333; b2 = H2 = 0,1666]. Для следующего шага он станет рабочим отрезком (рис. 1.5).



Рисунок 1.5 - Рабочий отрезок после анализа первых двух символов

Строго говоря, на первом шаге не было необходимости рассчитывать новые границы для каждого символа. Как видим, это только увеличивает время анализа. Достаточно было определить новые верхнюю и нижнюю границы лишь для очередного (на шаге 2) символа (о). Мы здесь привели расчеты только для того, чтобы лучше понять процесс вычислений.

Шаг 3: поскольку очередным из анализируемых символов будет буква «л», достаточно в соответствии с выражениями определить новые границы. Отрезок представлен на рисунке 1.6.



Рисунок 1.1.6 - Рабочий отрезок с разметками после анализа последовательности «мол»

Продолжаем анализировать символы сообщения. Находим на рабочем отрезке интервал [0,104166; 0,1111], который соответствует очередному символу: «о». Производим разметку нового рабочего отрезка, как показано на рис. 1.7.



Рисунок 1.1.7 - Разметка рабочего отрезка на четвертом шаге.

У нас остался последний символ: «о». Результатом кодирования цепочки символов является любое число с итогового рабочего отрезка [0,107060185; 0,10763888]. Обычно таким числом является нижняя граница указанного отрезка. Поступим и мы в соответствии с данным принципом. Возьмем число с меньшим количеством знаков после запятой.

Таким образом, итогом сжатия входной последовательности «молоко» будет число 0,107060185 (для упрощения дальнейших вычислительных операций округлим его до 0,1071).

Обратное преобразование (декомпрессия). Для восстановления исходного сообщения необходима информация:

• о значении числа, являющегося итогом сжатия сообщения (в нашем случае 0,1071);

• количестве символов в сжатом сообщении;

• вероятностных параметрах всех символов исходного сообщения (таблица вероятностей).

Как и при сжатии, вначале необходимо начальный рабочий отрезок [0; 1) разбить на интервалы, длины которых равны вероятностям появления соответствующих символов, т. е. создать рабочий отрезок.

Анализ будем проводить с использованием конкретных данных, взятых из последнего примера.

Итак, в качестве исходного участка для обратного преобразования принимается исходный участок для прямого преобразования с одинаковыми точками его разбиения. Ниже он повторяется.

На каждом шаге обратного преобразования выбираем отрезок, в который попадает текущее число (код). Символ, который соответствует данному отрезку, является очередным символом восстановленного (распакованного) сообщения.

В общем случае код символа, восстанавливаемого на шаге i, вычисляется соотношением, представленным на рисунке 1.8.



Рисунок 1.8 – Формула восстановления сообщения

Где код (i − 1) – число, анализ которого производился на предыдущем шаге – (i − 1)-м; H(αi − 1)0 и L(αi − 1)0 – соответственно верхняя и нижняя исходные границы символа сообщения, восстановленного на предыдущем шаге.

Шаг 1: определяем интервал, в который попадает начальное число: 0,1071 (код 1 = 0,1071). Это интервал [0; 0,166666] и ему соответствует символ «м». Данный интервал становится новым рабочим отрезком, а первый символ восстановленного сообщения – «м».

Шаг 2 и 3: производим вычисление в соответствии с выражением (1.8). Рисунок 1.9 и 1.10 соответственно.



Рисунок 1.9 – Вычисление кода для символа «о»



Рисунок 1.10 – Вычисления кода для символа «л»

Аналогичным образом производятся и последующие вычислительно-аналитические операции.

Другой вариант алгоритма распаковки «сжатого» сообщения основан на свойстве рекуррентности прямого преобразования. Производя на каждом шаге вычисление новых границ рабочего участка в соответствии с восстановленным на данном шаге символом (наподобие тех вычислений, которые мы выполняли при сжатии) и анализируя, на какой из отрезков этого участка попадает входное число (в нашем примере это 0,1071), восстанавливаем исходное сообщение.

Таким образом, на первом шаге это число соответствует букве «м», которая попадает в интервал [0; 0,1666]. Этот новый диапазон будет рабочим участком на шаге 2.

Так как нам известно, что символов в сообщении было 6, то на последнем шаге необходимо только определить недостающий символ. Таким символом становится символ «о». Полученное сообщение «молоко» и есть восстановленная («распакованная») последовательность.

К числу основных особенностей методов можно отнести следующие:

• проанализированный алгоритм сжатия (кодирования) ничего не передает до полного завершения анализа всего текста;

• обычно результат представляется в формате целочисленной арифметики;

• требуемая для представления интервала [Hi; Li] точность возрастает вместе с длиной анализируемого текста; постепенное выполнение алгоритма помогает преодолеть эту проблему, при этом необходимо внимательно следить за возможностью переполнения; упомянутые границы текущего интервала могут также представляться в виде целых чисел;

• как мы видим, арифметическое кодирование «работает» при помощи масштабирования или нормализации накопленных вероятностей, поставляемых моделью в интервалах [Hi; Li] для каждого передаваемого символа; если предположить, что Hi и Li настолько близки друг к другу, что операция масштабирования «приводит» разные символы сообщения к одному целому числу, входящему в [Hi; Li], то дальнейшее кодирование продолжать невозможно. Следовательно, программа-кодировщик должна следить за тем, чтобы интервал [Hi; Li] не «слипался».

**Практическое задание**

Разработать авторское приложение в соответствии с целью лабораторной работы.

С помощью приложения выполнить прямое и обратное преобразования произвольного сообщений в соответствии с таблицей.

Изменяя размеры окон, оценить скорость и эффективность выполнения операций сжатия/ распаковки.

Каждый студент выполняет задание, состоящее из двух частей. Первая часть предусматривает кодирование/декодирование сообщения, указанного в 2-м столбце, вторая часть – составного сообщения, полученного конкатенацией последовательностей из 2-го столбца, указанных в 3-м столбце. Например, для варианта № 1 такой конкатенацией будет последовательность «летоисчислениевремяпрепровождение».

Первоначально необходимо вычислить вероятности появления каждого символа в сообщении. Нахождение показано на рисунке 2.1.

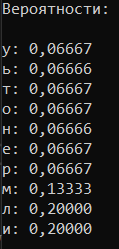


Рисунок 2.1 – Нахождение вероятностей появления символов

Далее находим интервалы, соответствующие каждому символу, опираясь на вероятности их появления. Пример нахождения интервалов показан на рисунке 2.2.

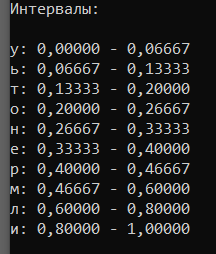


Рисунок 2.2 – Нахождение интервалов для символов

После этого мы применяем арифметический метод сжатия, принимая в расчет интервалы, которые были получены ранее. Результатом сжатия будет значение, соответствующее нижней границе символа, являющегося последним. Пример реализации арифметического метода сжатия показан на рисунке 2.3.

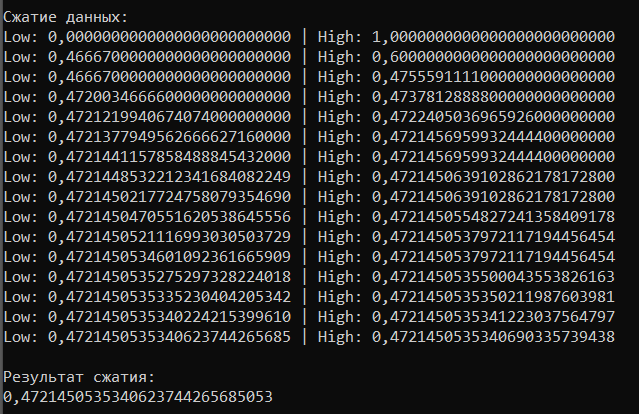


Рисунок 2.3 – Реализация метода арифметического сжатия

Далее необходимо произвести операцию распаковки сообщения. Для этого будет использоваться формула, показанная на рисунке 1.8 из теоретического введения. Пример реализации представлен на рисунке 2.4.

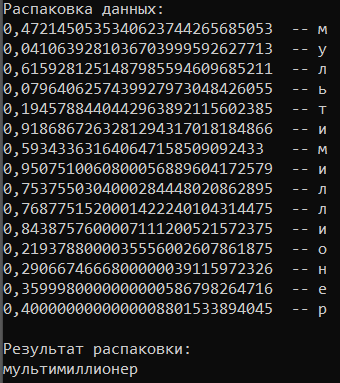


Рисунок 2.4 – Реализация метода арифметической распаковки

Последующим действием был анализ эффективности и подверженность сообщения переполнению. Помимо этого, находилось время, затраченное на сжатие и распаковку сообщения арифметическим методом. Результаты на рисунке 2.5.

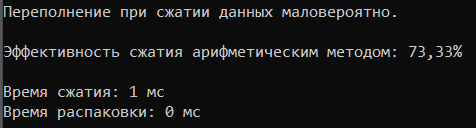


Рисунок 2.5 – Анализ произведенных действий

Во втором случае было необходимо взять сообщение, которое является более длинным. Выполнив все этапы, показанные ранее, мы получаем следующий результат, представленный на рисунке 2.6.

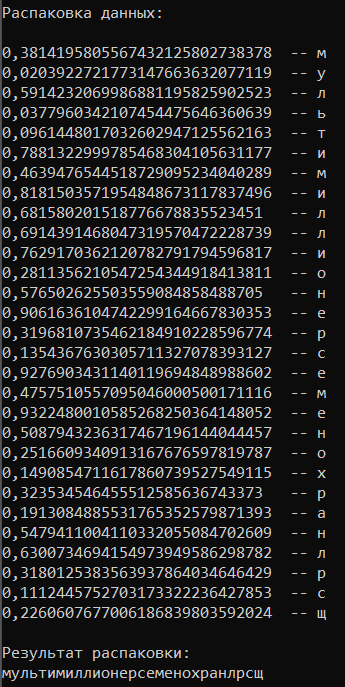


Рисунок 2.6 – Распаковка второго сообщения

Как мы можем заметить, результат является неточным. При использовании арифметического метода сжатия входное слово длинное, что приводит к малому различию между нижними и верхними границами отрезков. Это может привести к неточным вычислениям кодов из-за недостаточной точности в представлении вещественных чисел на компьютере. В результате возможны ошибки при восстановлении символов или даже потеря информации.

**Вывод:**

Арифметический алгоритм сжатия данных – это эффективный метод сжатия, основанный на использовании арифметических операций для представления данных в виде одного вещественного числа в определенном диапазоне. Этот алгоритм позволяет компактно представить последовательности символов путем вычисления диапазонов для каждого символа и их комбинаций во входных данных.

Основная идея арифметического сжатия заключается в том, что каждый символ представляется диапазоном значений в интервале от 0 до 1. При сжатии алгоритм вычисляет диапазон, соответствующий всей последовательности символов, что позволяет компактно записать эту последовательность в виде одного числа. Это число затем может быть записано или передано более эффективно, чем исходная последовательность.

Однако при использовании арифметического метода сжатия важно учитывать возможные ограничения представления вещественных чисел на компьютере. Недостаточная точность при вычислениях может привести к потере информации или неточному восстановлению данных при их распаковке. Это особенно критично при работе с длинными последовательностями или символами с близкими частотами встречаемости.

Таким образом, арифметический алгоритм сжатия данных представляет мощный инструмент для сжатия информации, но требует тщательного контроля точности вычислений и обработки данных для предотвращения потери информации и обеспечения корректного восстановления исходных данных.