



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

РЕФЕРАТ

о практическом задании по дисциплине АИСД
«Алгоритм сжатия информации арифметическое кодирование»

направление подготовки 09.03.03 «Прикладная информатика»
профиль «Прикладная информатика в компьютерном дизайне»

Выполнил студент
гр. Б9121-09.03.03 пикд
Безрукова Анастасия Леонидовна

(подпись)

Руководитель практики
Доцент ИМКТ А.С. Кленин
(должность, уч. звание)

(подпись)

«_____» _____ 2022г.

Реферат защищен:

С оценкой _____

Рег. № _____

«_____» _____ 2022 г.

г. Владивосток
2023

Оглавление

Глоссарий	3
Введение	4
Постановка задачи.....	5
Авторы и история	6
Описание алгоритма.....	7
Пример кодирования и декодирования.....	8
Описание реализации.....	10
Тестирование.....	11
Анализ.....	12
Заключение.....	14
Список литературы.....	15

Глоссарий

Вероятность (встречаемость) – количество повторений одного символа делённое на общее количество символов в тексте.

Введение

Арифметическое кодирование — один из алгоритмов энтропийного сжатия.

В начале работы алгоритма исходный интервал равен $[0; 1)$. По мере кодирования текста, исходный интервал уменьшается (сокращается исходя из значения вероятности символа). При арифметическом кодировании текст представляется вещественным числом в интервале от 0 до 1.

Для того, чтобы обеспечить остановку алгоритма распаковки, надо поставить длину текста или ввести дополнительный символ-маркер конца текста.

Постановка задачи

Задача разделяется на следующие пункты:

1. Подобрать и изучить источники по теме: «Арифметическое кодирование».
2. Описать алгоритм «Арифметическое кодирование» в форме научного доклада.
3. Реализовать алгоритм «Арифметическое кодирование».
4. Выполнить анализ эффективности сжатия алгоритма «Арифметическое кодирование».

Авторы и история

Базовые алгоритмы арифметического кодирования были разработаны независимо Йорма Дж. Риссаненом из IBM Research и Ричардом К. Паско, аспирантом Стэнфордского университета; оба были опубликованы в мае 1976 года.

Описание алгоритма

Кодирование:

1. На вход программы поступает текст и его длина - вводится либо с клавиатуры, либо из файла.
2. Из полученного текста создается алфавит - массив символов, исключаяющий повторы (заданное множество символов — это, как правило, ASCII+).
3. Для каждого символа определяется его интервал, равный вероятности его появления (вероятности символов, отсутствующих в тексте, равна 0).
4. По мере просмотра текста от начала до конца вызывается функция, которая пересчитывает границы интервала каждого текущего символа по формуле [\(1\)](#).
5. В качестве закодированного текста выводится любая из границ интервала последнего символа.

Декодирование:

1. На вход поступает закодированный текст.
2. Вызывается функция декодирования, которая по итоговым интервалам символов расшифровывает текст (см. стр. [9](#)).

Пример кодирования и декодирования

Составим таблицу интервалов для символов входного сообщения «СЕВА», указав в ней вероятности символов (см. Таблица 1).

Символ алфавита	Вероятность символа	Диапазон для символа (границы)	
		Low	High
A	$\frac{1}{4}=0,25$	0	0,25
B	$\frac{1}{4}=0,25$	0,25	0,5
C	$\frac{1}{4}=0,25$	0,5	0,75
E	$\frac{1}{4}=0,25$	0,75	1

Таблица 1 - таблица интервалов и вероятностей символов

Процесс кодирования

Начальные границы интервала: Low=0, High=1.

Для символа «C»:

$$L1 = \text{Low} + L[C] (\text{High} - \text{Low}) = 0 + 0,5(1 - 0) = 0,5 \quad (1)$$

$$H1 = \text{Low} + H[C] (\text{High} - \text{Low}) = 0 + 0,75(1 - 0) = 0,75$$

Для символа «E»:

$$L2 = \text{Low} + L[E] (H1 - L1) = 0,5 + 0,5(0,75 - 0,5) = 0,6875$$

$$H2 = \text{Low} + H[E] (H1 - L1) = 0,5 + 1(0,75 - 0,5) = 0,75$$

Для символа «B»:

$$L3 = L2 + L[B] (H2 - L2) = 0,6875 + 0,25(0,75 - 0,6875) = 0,703125$$

$$H3 = L2 + H[B] (H2 - L2) = 0,6875 + 0,5(0,75 - 0,6875) = 0,71875$$

Для символа «A»:

$$L4 = L3 + L[A] (H3 - L3) = 0,703125 + 0(0,71875 - 0,703125) = 0,703125$$

$$H4 = L3 + H[A] (H3 - L3) = 0,703125 + 0,25(0,71875 - 0,703125) = 0,7070312$$

Таким образом, число 0,703125 однозначно кодирует сообщение «СЕВА» (см. Таблица 2).

Символ	Нижняя граница	Верхняя граница
С	0,5	0,75
Е	0,6875	0,75
В	0,703125	0,7185
А	0,703125	0,7070312

Таблица 2 - итоговая таблица границ символов

Декодирование слова «СЕВА».

Конечный интервал (0,703125; 0,7070312), который получился после кодирования, принадлежит символу «С». После определения первого символа интервал равен значениям границ «С» - (0,5; 0,75), в него входит интервал для буквы «Е», который получился после кодирования. Следовательно были декодированы первые два символа «С» и «Е» (см. Таблица 2).

По такой же аналогии расшифровываются остальные символы.

Описание реализации

Реализация алгоритма состоит из 3 функций (*generate()*, *coding ()*, *decoding ()*), а также главной функции *main ()*, где, собственно, и вызываются функции.

Функция *generate ()* составляет алфавит, подсчитывает вероятности символов и вычисляет их границы. С помощью флага *k* делается проверка (есть ли символ в алфавите или нет), после чего составляется алфавит. Вероятность подсчитывается с помощью длины всего текста и заносится в верхнюю границу символа ($vgran[i] = ngran[i] + ver[i]$). Нижняя граница изначально равна 0, после подсчёта вероятности она приравнивается значению верхней границы.

Функция *coding ()* выполняет само кодирование текста, которое мы ввели. Пересчитываются границы с помощью формул (см. Формула 1):

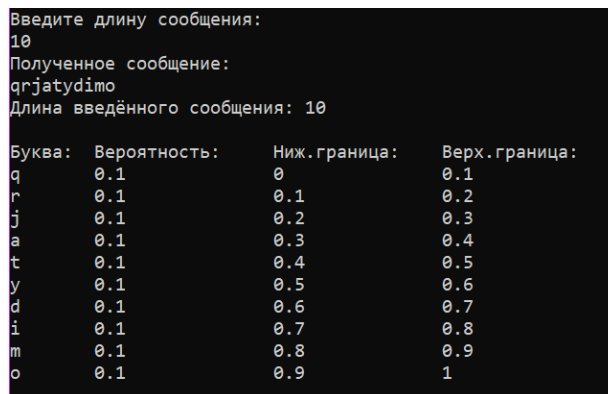
$$\begin{aligned} ngran1[i] &= low + ngran[i] * (high - low); \\ vgran1[i] &= low + vgran[i] * (high - low); \end{aligned} \quad (1)$$

Функция *decoding ()* отвечает за декодирование текста. В начале создаём пустую строку для декодирования. Запускаем цикл по алфавиту, запоминаем границу и затем пересчитываем её. Если она входит в интервал закодированного текста, то записываем найденную букву в строку и вычитаем границу из закодированного текста.

Главная функция *main ()* отвечает за вызов всех функций.

Тестирование

Для тестирования алгоритма используется случайная генерация текста (на вход поступает длина текста, после чего текст генерируется случайным способом). Чтобы проверить корректность кодирования суммируются вероятности символов алфавита. Кодирование работает корректно если сумма вероятностей равна 1. Также алгоритм реализован верно если нижняя граница первого символа равна 0, а верхняя граница последнего символа равна 1 (см. Рисунок 1).



```
Введите длину сообщения:
10
Полученное сообщение:
qrjatydim0
Длина введенного сообщения: 10
```

Буква:	Вероятность:	Ниж.граница:	Верх.граница:
q	0.1	0	0.1
r	0.1	0.1	0.2
j	0.1	0.2	0.3
a	0.1	0.3	0.4
t	0.1	0.4	0.5
y	0.1	0.5	0.6
d	0.1	0.6	0.7
i	0.1	0.7	0.8
m	0.1	0.8	0.9
o	0.1	0.9	1

Рисунок 1 – Вероятности и границы символов алфавита

В данной реализации вещественные числа типа *double* могут иметь не более 16 знаков после запятой, в следствии чего программа может работать некорректно при большом объёме текста так как границы символов округляются. Корректно работает на текстах длиной ≤ 10 символов. Всего было проведено 23 теста, 10 из которых работают корректно.

Анализ

В рамках работы была исследована производительность алгоритма «Арифметическое кодирование». Были проведены тесты на корректность, анализ по затраченному времени, а так же был определён процент сжатия текста. Исследование проводилось на случайных выборках разного объема текста – 500, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 50000 и 100000 соответственно.

Подсчёт времени производится при помощи библиотеки *chrono*. Время кодирования представлено в миллисекундах, а декодирования в наносекундах (т.к. декодирование выполняется очень быстро).

Функция кодирования (*coding()*):

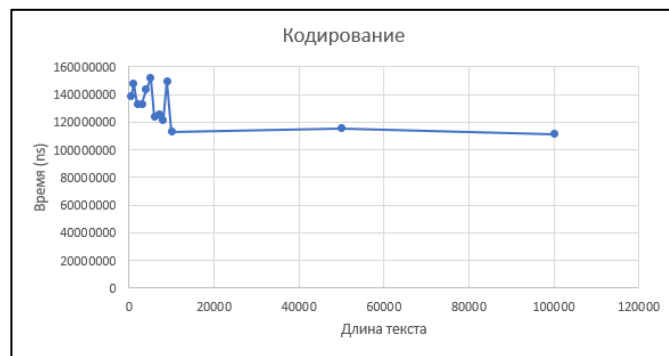


Рисунок 2 – график времени кодирования

Функция декодирования (*decoding()*):

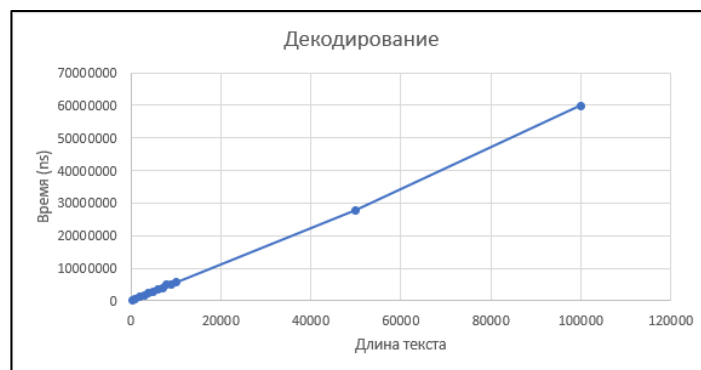


Рисунок 3 – график времени декодирования

Рассмотрев полученные графики можно сделать вывод о том, что время при кодировании текста длиной 500-10000 символов варьируется в пределах 100000000 – 160000000 наносекунд, но когда текст достигает 10000 – 100000 символов, время практически остаётся неизменным. Время выполнения декодирования, по мере увеличения длины текста, линейно увеличивается.

Процент сжатия стремится к 99,9% по мере увеличения длины текста (см. Рисунок 4).

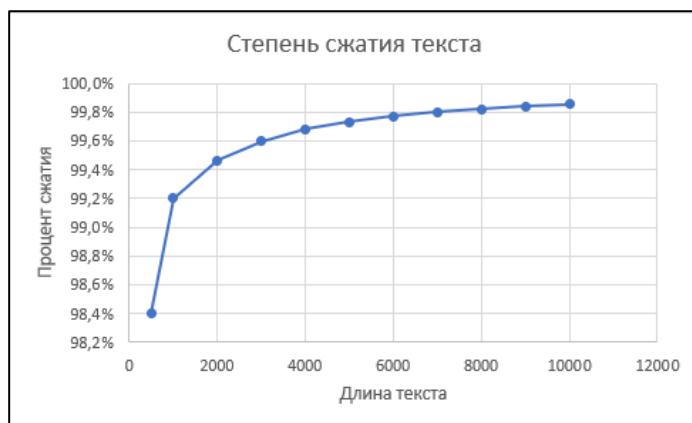


Рисунок 4 – График процента сжатия текста

Заключение

В ходе исследования достигнуты следующие результаты:

1. Изучены литературные и интернет-источники по теме «Арифметическое кодирование». Изученная информация представлена в научной форме.
2. Реализованы алгоритмы арифметического кодирования и декодирования.
3. Проведён анализ производительности алгоритма.
4. Результат работы загружен на [GitHub](https://github.com/bezzzna/Arithmetic_Coding) ([bezzzna/Arithmetic_Coding \(github.com\)](https://github.com/bezzzna/Arithmetic_Coding)).

Список литературы

- [1] Арифметическое кодирование. URL: [Arithmetic coding - Wikipedia](#)
- [2] Алгоритмы сжатия. URL: [Алгоритмы сжатия - Обзор алгоритмов сжатия без потерь \(grsu.by\)](#)
- [3] Арифметическое кодирование. URL: [Арифметическое кодирование \(helpiks.org\)](#)
- [4] Идея арифметического кодирования. URL: [Арифметическое кодирование \(studfile.net\)](#)