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 задание по дисциплине АИСД
«Алгоритм сжатия информации арифметическое кодирование»

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

Выполнил студент
гр. Б9121-09.03.03пикд
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

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, исключающий 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	¼=0,25	0	0,25
B	¼=0,25	0,25	0,5
C	¼=0,25	0,5	0,75
E	¼=0,25	0,75	1

Таблица 1 - таблица интервалов и вероятностей символов

Процесс кодирования

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

Для символа «С»:

L1= Low + L[C] (High- Low) = 0+0,5(1-0) = 0,5 *(1)*

H1= Low + H[C] (High-Low) = 0+0,75(1-0) = 0,75

Для символа «Е»:

L2= Low + L[E] (H1- L1) = 0,5+0, 5(0,75-0,5) = 0,6875

H2= Low + H[E] (H1- L1) = 0,5+1(0,75-0,5) = 0,75

Для символа «В»:

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) = 0,7185

Для символа «А»:

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
C	0,5	0,75
E	0,6875	0,75
B	0,703125	0,7185
A	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 4 функций (***get_number (), get_code (), coding (), decoding ()***), а также главной функции ***main (),*** где, собственно, и вызываются функции.

Для начала через директиву ***#define*** объявляются две глобальные переменные ***M*** и ***N***, где переменная ***M*** отвечает за длину сообщения (в нашем случае длина 100), и ***N*** отвечает за количество символов в используемом словаре (в нашем случае 4).

Далее определяем приватные переменные, присваиваем их классу ***suanshu*** и определяем публичные функции для данного класса. Публичные функции нужны для того, чтобы иметь доступ к ним из любого места кода.

~suanshu() – это функция деструктора, для очистки памяти.

Функция ***get_number ()*** возвращает символы (***number [i]***) и их вероятности (***chance[i]***), которые программа получает на ввод с клавиатуры.

Функция ***get_code ()*** выполняет ввод длины кодируемого сообщения, чтобы программа знала, когда ей остановиться. Также эта функция получает на ввод сам код сообщения, которое нам нужно закодировать.

Функция ***coding ()*** выполняет само кодирование сообщения, которое мы ввели. Сначала она обрабатывает первый введённый нами символ, оценивает его подстрочный индекс и находит его верхнюю и нижнюю границы.

Затем рассматривается второй символ. Если он равен предыдущему, то нижняя граница символа остаётся прежней и меняется только верхняя. Так же пересчитывается вероятность. Иначе если второй символ другой, то меняется и нижняя и верхняя границы, так же меняется и вероятность. В этой же функции выводится результат кодирования, в нашем случае это нижняя граница.

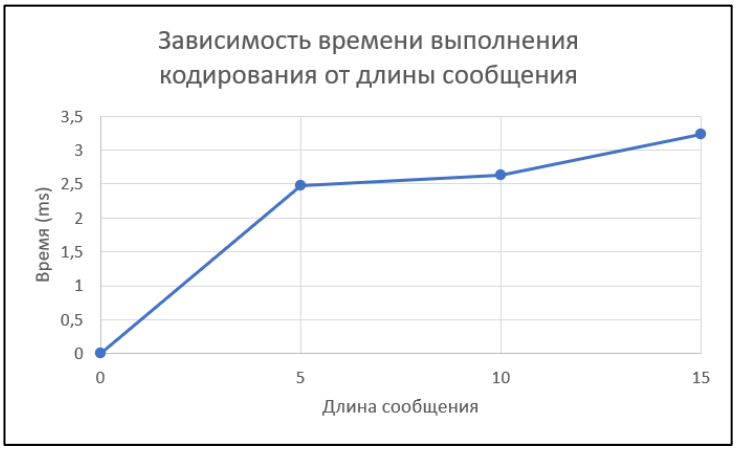
Функция ***decoding ()*** отвечает за декодирование нашего сообщения. В начале объявляем переменную типа char, которая отвечает за количество декодируемых символов.

Далее расшифровываем символы по одному, постоянно меняя интервал кодирования, исходя из полученного результата в предыдущей функции (нижней границы ***Low***).

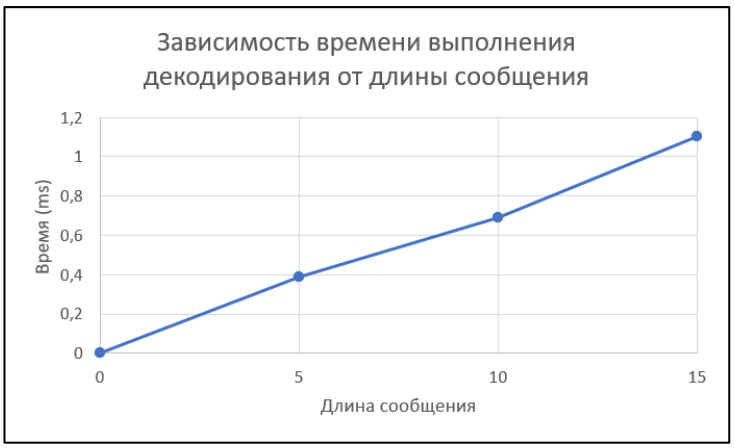
Главная функция ***main ()*** отвечает за вызов всех функций.

Тестирование

Функция кодирования (***coding()***):



Функция декодирования (***decoding()***):



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 (bezzzna/Arithmetic_Coding (github.com)).

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)