МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационыые технологиии прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Курсовой проект
по курсу «Дискретный анализ»

Студент: Д. Д. Наумов
Преподаватель: Н. А. Зацепин
Группа: М8О-406Б-17
Дата:
Оценка:
Подпись:

Москва, 2020

Тема: Архиватор

Описание

Задача состоит в реализации архиватора, на основе алгоритма арифметическаго сжатия. Программа состоит из двух частей: кодера и декодера, которые соответственно сжимают и разжимают входные данные. В качестве входных данных декодеру принимаются текстовые файлы размеров меньше чем 2^{31} бит = 2,6 ГБ. На выходе получается сжатый файл, который подается на вход декодеру.

Описание алгоритма

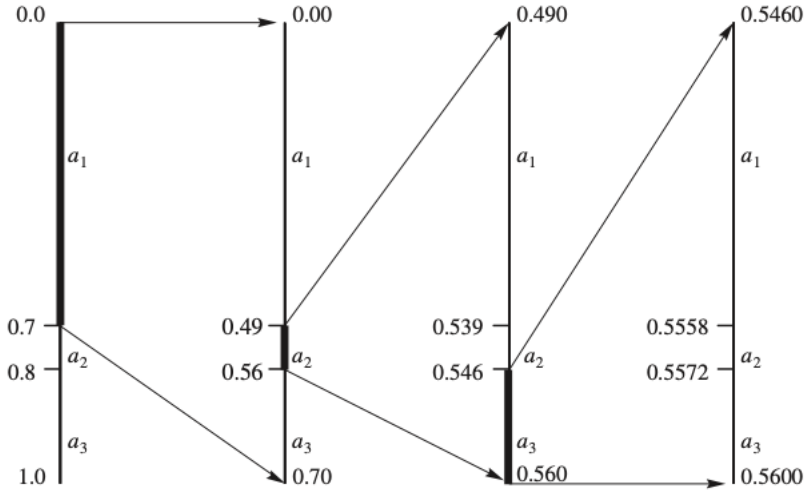
При арифметическом кодировании для кодируемой последовательности создается уникальный идентификатор или тег. Этот тег соответствует двоичной дроби, которая становится двоичным кодом последовательности. Уникальный арифметический код может быть сгенерирован для последовательности длиной m без необходимости генерировать кодовые слова для всех последовательностей длины m. Это не похоже на ситуацию с кодами Хаффмана. Чтобы сгенерировать код Хаффмана для последовательности длины m, где код не является конкатенацией кодовых слов для отдельных символов, нам необходимо получить коды Хаффмана для всех последовательностей длины m.

Чтобы отличить последовательность символов от другой последовательности символов, нам нужно пометить ее уникальным идентификатором. Один из возможных наборов тегов для представления последовательностей символов - это числа в единичном интервале $[0, 1)$. Поскольку количество чисел в единичном интервале бесконечно, должно быть возможно присвоить уникальный тег каждой отдельной последовательности символов.

Процедура создания тега работает за счет уменьшения размера интервала, в котором находится тег, по мере получения все большего количества элементов последовательности.

Пример

Рассмотрим трехбуквенный алфавит $A = \{a_1, a_2, a_3\}$ с $P(a_1) = 0{,}7$, $P(a_2) = 0{,}1$ и $P(a_3) = 0{,}2$. Используя отображение $F_X(1) = 0{,}7$, $F_X(2) = 0{,}8$ и $F_X(3) = 1$ разбиваем единичный интервал, как показано на рисунке ниже.



Для последовательности, начинающейся с $a_1, a_2, a_3, \ldots$, к тому времени, когда будет получен третий символ a_3, тег будет ограничен субынтервалом $[0,546, 0,56)$. Если бы третьим символом был a_1 вместо a_3, тег находился бы в подынтервале $[0,49, 0,539)$, который не пересекается с подынтервалом $[0,546, 0,56)$. Даже если с этого момента две последовательности идентичны (одна начинается с a_1, a_2, a_3, а другая начинается с a_1, a_2, a_1), интервал тегов для двух последовательностей всегда будет непересекающимся.

Описание структуры программы

Программа разделена на следующие фрагменты:

- **BitIoStream** - поток побитового ввода/вывода
- **FrequencyTable** - таблица с частотами символов
- **ArithmeticCoder** - функции для обновления границ тега
- **ArithmeticCompress** - основная программа для сжатия
- **ArithmeticDecompress** - программа для декодирования

Пример работы программы

```
(base) MacBook:src dandachok$ ./encoder ../tests/input/1mb.t ../tests/encode/1mb.az
(base) MacBook:src dandachok$ du -h ../tests/input/1mb.t
980K ../tests/input/1mb.t
(base) MacBook:src dandachok$ du -h ../tests/encode/1mb.az
568K ../tests/encode/1mb.az
(base) MacBook:src dandachok$ ./decoder ../tests/encode/1mb.az ../tests/output/1mb.t
(base) MacBook:src dandachok$ du -h ../tests/output/1mb.t
980K ../tests/output/1mb.t
(base) MacBook:src dandachok$ diff ../tests/input/1mb.t ../tests/output/1mb.t
(base) MacBook:src dandachok$
```

Результаты

Файла	Размер файла	Алгоритм	Время сжатия	Время декомпрессии	Размер сжатого файла	Коэффициент сжатия
world95.txt	3005020 Б	Арифметика	0m2.306s	0m2.725s	1920022 Б	1.56
world95.txt	3005020 Б	gzip	0m0.309s	0m4.359s	868440 Б	3.4
enwik8	100000000 Б	Арифметика	1m15.385s	1m29.597s	63502180 Б	1.57
enwik8	100000000 Б	gzip	0m8.411s	0m2.545s	36475811 Б	2.75

- **Процессор** 1,8 GHz 2-ядерный процессор Intel Core i5
- **Память** 4 ГБ 1600 MHz DDR3
- **Графика** Intel HD Graphics 4000 1536 МБ

Выводы

Идея арифметического сжатия довольно простая, однако при реализации возникает много нюансов. Из-за ограничений точности типов с плавающей запятой, невозможно просто пересчитывать границы интервала. Необходимо переходить от вещественных значений к целым. Так же при реализации возникают ограничения на максимальный размер входного файла.