Отчёт по лабораторной 1

Наумов Семен Дмитриевич, M4138

BWT + MTF + Huffman

Описание работы алгоритмов

Алгоритм кодирования

Burrows Wheeler transformation

Перед применением самого алгоритма кодирования применяются два алгоритма преобразования, первый из которых - алгоритм преобразования Барроуза-Уиллера.

Читаем файл, разбивая его на блоки. Для каждого блока мы сортируем все его циклические сдвиги, на выходе получая последнюю колонку матрицы и номер строки, на которой оказалась изначальная строчка. Делаем это преобразование с блоком, потому что довольно затратно было бы совершать это с большими файлами.

Сортировать строки можно, сравнивая их по индексам или при использовании суффиксного массива. Эти алгоритмы дают одинаковую производительность на практике: на 100Мб текстовых файлах и файлах из `calgarycorpus`, кроме `pic` (там разница в десяток раз). Поэтому решил использовать именно суффиксный массив.

Move To Front transformation

Далее идёт преобразование методом стопки книг. Преобразования Барроуза-Уиллера даёт положительный эффект, из-за того что получается много одинаковых символов подряд. Этим можно воспользоваться в следующем преобразовании.

Возьмём упорядоченный алфавит, будем читать по преобразованному Барроузом-Уиллером символу, возвращать индекс этого символа в алфавите и перемещать его в начало алфавита. Таким образом все повторяющиеся подряд символы, кроме первого из серии, превращаются в 0, что выгодня для следующего за этим алгоритма сжатия.

Huffman compression

Алгоритм Хаффмана основывается на вероятностях появления символа в блоке. Соответственно, за первый проход по блоку можем вычислить вероятноти, составить дерево кодовых слов по ним и за второй проход сделать последовательность кодовых слов, которая будет записана в выходной файл.

В выходном файле так же будет хэдер, содержащий количество следующего блока в битах, номер изначальной строки для преобразования Барроуза-Уиллера и дерево кодовых слов для Хаффмана. Дерево кодовых слов можно записать при помощи обхода: будем писать 0, если двигаемся по дереву, 1, если пришли в лист, как только пришли в лист, можем записать символ, соответствующий этому листу.

Алгоритм декодирования

Huffman decompression

В хэдера блока записаны кодовые слова, по которым подряд можно декодировать символы для следующего обратного преобразования.

Reverse Move To Front transformation

Всё то же самое, что и с прямым преобразованием, только теперь мы по индексу возвращаем соответствующий символ алфавита.

Reverse Burrows-Wheeler transformation

По строке, которая приходит на вход, можно восстановить вектор переходов. Отсортировав пришедшую строку, мы получим первый столбец матрицы циклических сдвигов. Можем собрать вектор переходов позиций, на которых совпадают символы в последнем и первом столбце матрицы, причём если символы повторяются, то они будут идти в том же порядке.

По вектору переходов можно восстановить изначальную строку, так как мы знаем, откуда она начинается, с какой позиции, эта информация была записана в хэдере блока.

Результаты

file name	H(X)	H(X | X)	H(X | XX)	значение средних затрат на символ (в битах)	размер сжатого файла (в байтах)	размер исходного файла (в байтах)
bib	5.20068	3.36411	2.30753	2.386604470	33192	111261
book1	4.52715	3.58452	2.81408	2.780032025	267151	768771
book2	4.79263	3.74521	2.73568	2.448773524	186981	610856

file name	H(X)	H(X | X)	H(X | XX)	значение средних затрат на символ (в битах)	размер сжатого файла (в байтах)	размер исходного файла (в байтах)
geo	5.64638	4.2642	3.45776	5.433671875	69551	102400
news	5.18963	4.09189	2.92277	2.832157281	133504	377109
obj1	5.94817	3.46356	1.40057	4.379836095	11773	21504
obj2	6.26038	3.87036	2.26544	2.875687765	88720	246814
paper1	4.98298	3.64604	2.33181	2.740505258	18211	53161
paper2	4.60143	3.52233	2.51367	2.747162253	28124	82199
pic	1.21018	0.823655	0.705199	1.582101883	101495	513216
progc	5.19902	3.60334	2.13407	2.764080685	13686	39611
progl	4.77009	3.21158	2.04359	2.091617117	18732	71646
progp	4.86877	3.18751	1.75515	2.076024221	12814	49379
trans	5.53278	3.35477	1.93052	1.911564118	22388	93695

Суммарное значение сжатого размера всех файлов в байтах: 1 006 322