**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР.**

**отчет**

**по лабораторной работе №3**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: «Алгоритмы сжатия без потерь».**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1301 |  | Ищенко Д.О. |
| Преподаватель |  | Родионова Е. А. |

Санкт-Петербург

2023

Оглавление

[Формулировка задания. 3](#_Toc25931)

[Теоретические сведения. 3](#_Toc13167)

[Оценка временной сложности. 6](#_Toc5434)

[Пример работы программы. 9](#_Toc19369)

[Текст программы. 9](#_Toc5020)

# Формулировка задания.

Реализовать следующие алгоритмы сжатия символьных данных:

1. Алгоритм Хаффмана (HA)
2. Кодирование длин серий (RKE)
3. Алгоритм Лемпеля-Зива (LZ78)
4. *Преобразование Барроуза-Уиллера (BWT)*
5. *MTF*
6. Арифметическое кодирование (AC)
7. PPM

Проанализировать эффективность и время сжатия на датасете enwik8 для процедур сжатия на основе:

1. Алгоритма Хаффмана
2. Арифметического кодирования
3. LZ78
4. BWT -> MTF -> HA
5. BWT -> MTF -> AC
6. RLE -> BWT -> MTF -> RLE -> HA
7. RLE -> BWT -> MTF -> RLE -> AC
8. PPM

Результаты представить в виде или таблиц. При слишком больших временных затратах на сжатие взять первые 10% enwiki8.

# Теоретические сведения.

Построение кода Хаффмана сводится к построению соответствующего [бинарного дерева](https://neerc.ifmo.ru/wiki/index.php?title=%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BA%D1%83%D1%87%D0%B0" \o "Двоичная куча) по следующему алгоритму:

1. Составим [список](https://neerc.ifmo.ru/wiki/index.php?title=%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA" \o "Список) кодируемых символов, при этом будем рассматривать один символ как дерево, состоящее из одного элемента c весом, равным частоте появления символа в строке.
2. Из списка выберем два узла с наименьшим весом.
3. Сформируем новый узел с весом, равным сумме весов выбранных узлов, и присоединим к нему два выбранных узла в качестве детей.
4. Добавим к списку только что сформированный узел вместо двух объединенных узлов.
5. Если в списке больше одного узла, то повторим пункты со второго по пятый.

Алгоритм RLE (англ. Run-Length Encoding) — алгоритм сжатия, заменяющий идущие подряд одинаковые символы парой (повторяющийся символ, количество повторений). Например, строчку aaababbcbbb он переводит в (a, 3) (b, 1) (a, 1) (b, 2) (c, 1) (b, 3). Этот алгоритм эффективен для строк, содержащих много цепочек повторяющихся символов, например, результата

Алгоритм LZ78 имеет идею: этот алгоритм в явном виде использует словарный подход, генерируя временный словарь во время кодирования и декодирования.

Изначально словарь пуст, а алгоритм пытается закодировать первый символ. На каждой итерации мы пытаемся увеличить кодируемый префикс, пока такой префикс есть в словаре. Кодовые слова такого алгоритма будут состоять из двух частей — номера в словаре самого длинного найденного префикса (pos) и символа, который идет за этим префиксом (next). При этом после кодирования такой пары префикс с приписанным символом добавляется в словарь, а алгоритм продолжает кодирование со следующего символа.

Преобразование Барроуза — Уилера (англ. Burrows-Wheeler transform) — алгоритм, используемый для предварительной обработки данных перед сжатием, разработанный для улучшения эффективности последующего кодирования. Преобразование Барроуза — Уилера меняет порядок символов во входной строке таким образом, что повторяющиеся подстроки образуют на выходе идущие подряд последовательности одинаковых символов.

MTF алгоритм работает по следующему принципу: изначально каждое возможное значение байта записывается в список (алфавит), в ячейку с номером, равным значению байта, т.е. (0,1,2,3,…,255)(0,1,2,3,…,255). В процессе обработки данных этот список изменяется. По мере поступления очередного символа на выход подается номер элемента, содержащего его значение. После чего этот символ перемещается в начало списка, смещая остальные элементы вправо.  
 Арифметическое кодирование (англ. Arithmetic coding) — алгоритм сжатия информации без потерь, который при кодировании ставит в соответствие тексту вещественное число из отрезка [0;1)[0;1). Данный метод, как и [алгоритм Хаффмана](https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD%D0%B0" \o "Алгоритм Хаффмана), является [энтропийным](https://neerc.ifmo.ru/wiki/index.php?title=%D0%AD%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D1%8F_%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B8%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B0" \o "Энтропия случайного источника), то есть длина кода конкретного символа зависит от частоты встречаемости этого символа в тексте. Арифметическое кодирование показывает более высокие результаты сжатия, чем алгоритм Хаффмана, для данных с неравномерными распределениями вероятностей кодируемых символов.

При арифметическом кодировании каждый символ кодируется нецелым числом бит, что эффективнее кода Хаффмана (теоретически, символу «a» с вероятностью появления p(a) допустимо ставить в соответствие код длины −log2p(a)), следовательно, при кодировании алгоритмом Хаффмана это достигается только с вероятностями, равными обратным степеням двойки).

Алгоритм PPM (prediction by partial matching) - это метод контекстно-ограниченного моделирования, позволяющий оценить вероятность символа в зависимости от предыдущих символов. Строку символов, непосредственно предшествующую текущему символу, будем называть контекстом. Модели, в которых для оценки вероятности используются контексты длиной не более чем N, принято называть моделями порядка N.

# Оценка временной сложности.

Поскольку BWT требует слишком много памяти (10Гб на 0.1% датасета) результаты представлены без него на 100% датасета. На 0.1% алгоритм отрабатывает за 50655797 микросекунд.

Исходный размер файла: 100,000,000 bytes

1. HA time: 22,106,564 ; 510,896,475 bytes
2. AC time: 4,535,171 ; 8 bytes
3. LZ78 time: 83,632,069 ; 186,289,003 bytes
4. MTF time: 85,531,969

+ HA time: 69,250,879

Total: 154,782,848

Size: 736,934,975 bytes

1. MFT time: 85,810,845

+ AC time: 11299642

Total: 97,110,487

Size: 8 bytes

1. RLE time: 26,610,154

+ MFT time: 146,965,303

+ RLE time: 122,760,424

+ HA time: 184,775,473

Total: 481,111,354

Size: 1,581,475,260 bytes

1. RLE time: 27,294,209

+ MFT time: 153,475,788

+RLE time: 117,817,025

+AC time: 36,476,154

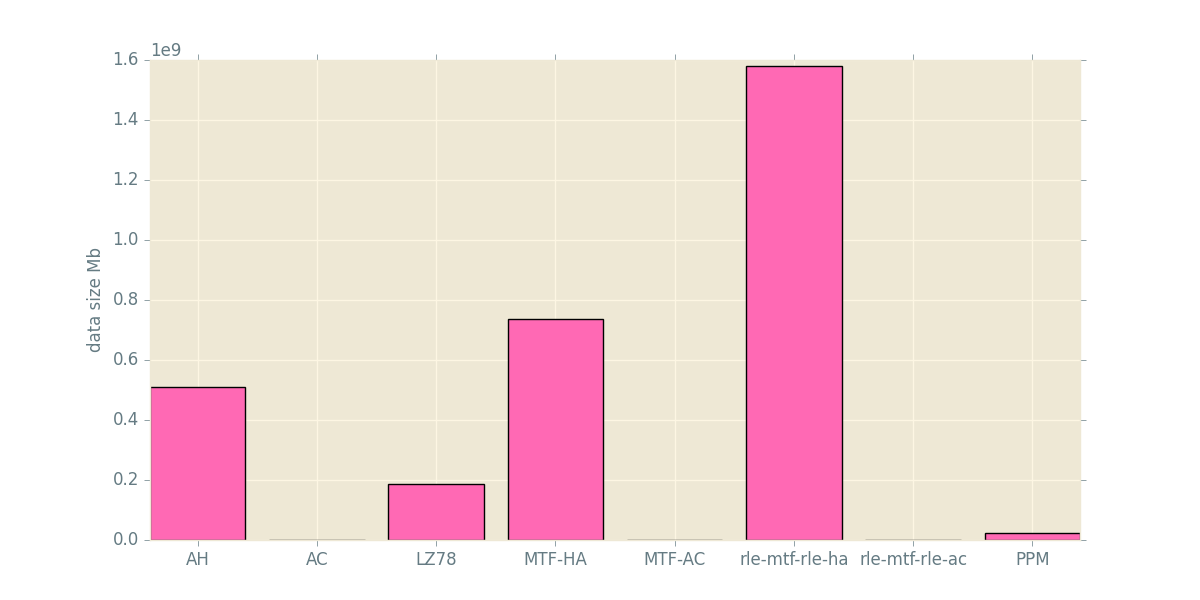
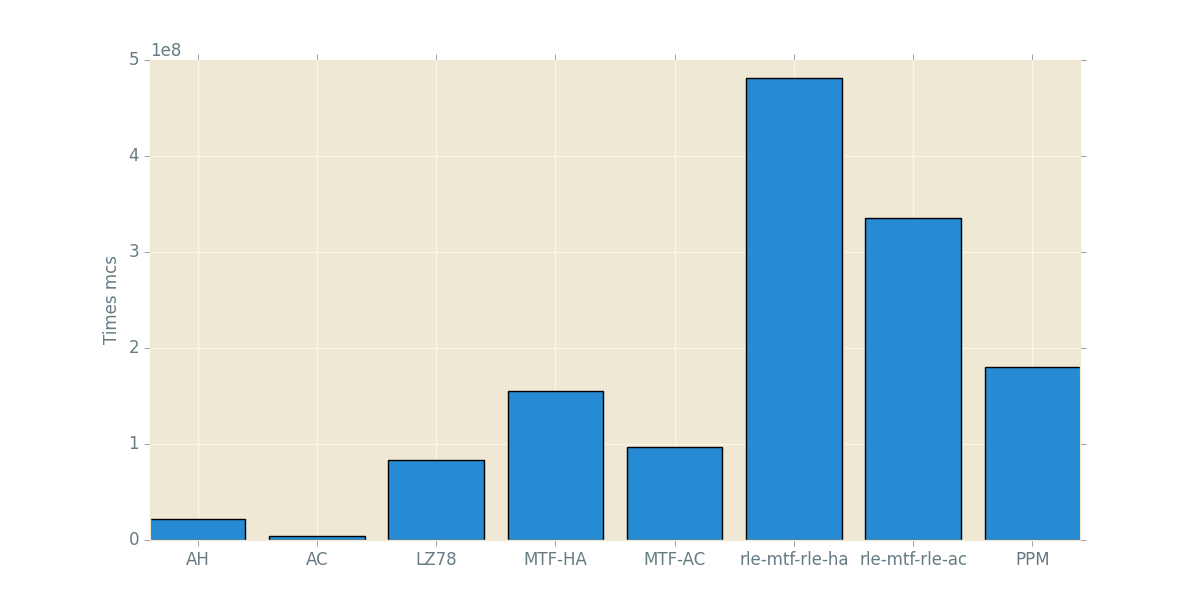
Total: 335,063,176

Size: 8 bytes

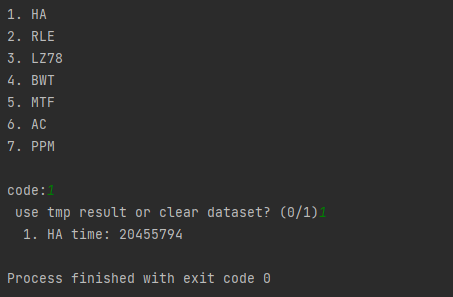
1. ppm time: 179,718,984

ppm size: 24372304 bytes

Визуализация данных:



# Пример работы программы.



# Текст программы.

[Ссылка на github](https://github.com/Nekttuman/Algosee)