НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ» Дисциплина: «Алгоритмы и структуры данных»

**Контрольное домашнее задание**

**«Анализ алгоритмов архивации и разархивации»**

Выполнила: Казанцева Анастасия,   
студент группы БПИ162.

Оглавление

[Постановка задачи 3](#_Toc509820126)

[Описание алгоритмов и использованных структур данных 3](#_Toc509820127)

[Алгоритм Хаффмана 3](#_Toc509820128)

[Алгоритм Шеннона-Фано 3](#_Toc509820129)

[Алгоритм Лемпела-Зива 1977 3](#_Toc509820130)

[Используемые структуры данных 3](#_Toc509820131)

[Описание плана эксперимента 4](#_Toc509820132)

[Результаты экспериментов 4](#_Toc509820133)

[Сравнительный анализ алгоритмов 5](#_Toc509820134)

[Заключение 5](#_Toc509820135)

# Постановка задачи

1. Реализовать с использованием языка C++ программы для упаковки и распаковки файлов. При этом использовать три известных алгоритма кодирования информации:
   1. Алгоритм Хаффмана
   2. Алгоритм Шеннона-Фано
   3. Алгоритм Лемпела-Зива 1977
2. Провести вычислительный эксперимент с целью оценки реализационных алгоритмов упаковки/распаковки. Оценить время работы каждого алгоритма и коэффициенты сжатия, используя стандартный набор из 36 файлов. Рассчитать энтропию каждого файла.
3. Подготовить отчет по итогам работы, содержащий постановку задачи, описание алгоритмов и задействованных структур данных, описание реализации, обобщенные результаты измерения эффективности алгоритмов, описание использованных инструментов , оценку соответствия результатов экспериментальной проверки теоретическим оценкам эффективности исследуемых алгоритмов.

# Описание алгоритмов и использованных структур данных

## Алгоритм Хаффмана

Алгоритм Хаффмана - алгоритм оптимального префиксного кодирования алфавита с минимальной избыточностью. Был разработан в 1952 году аспирантом Массачусетского технологического института Дэвидом Хаффманом при написании им курсовой работы. В настоящее время используется во многих программах сжатия данных.

Этот метод кодирования состоит из двух основных этапов:

1. Построение оптимального кодового дерева.
2. Построение отображения код-символ на основе построенного дерева.

## Алгоритм Шеннона-Фано

Алгоритм Шеннона — Фано — один из первых алгоритмов сжатия, который впервые сформулировали американские учёные Шеннон и Роберт Фано. Данный метод сжатия имеет большое сходство с алгоритмом Хаффмана, который появился на несколько лет позже и является логическим продолжением алгоритма Шеннона. Алгоритм использует коды переменной длины: часто встречающийся символ кодируется кодом меньшей длины, редко встречающийся — кодом большей длины. Коды Шеннона — Фано — префиксные, то есть никакое кодовое слово не является префиксом любого другого. Это свойство позволяет однозначно декодировать любую последовательность кодовых слов.

## Алгоритм Лемпела-Зива 1977

## Используемые структуры данных

Std::map<> - для удобного поиска и подсчета количества элементов

Std::vector<> - для сортировки и итерации по элементам

Std::pair<> - для удобного хранения пар код-символ

# Описание плана эксперимента

1. Реализовать необходимые алгоритмы упаковки и распаковки
2. Применить каждый из них к тестовому набору
3. На основе полученных результатов сделать вывод об изученных алгоритмах

# Результаты экспериментов

H - энтропия исходного файла

K - коэффициент сжатия

tp - время упаковки (10-12 секунд)

tu - время распаковки (10-12 секунд).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Имя файла | H | Хаффман | | | Шеннон Фано | | | LZ77 (5) | | | LZ77 (10) | | | LZ77 (20) | | |
| K | tp | tu | K | tp | tu | K | tp | tu | K | tp | tu | K | tp | tu |
| 01 | 7.99432 | 1.00022 | 4.3 | 8.9 | 1.002 | 4.3 | 7.7 |  |  |  | 4,13266699023236 | 281.3 | 0.2 |  |  |  |
| 02 | 4.59692 | 0.579213 | 5.3 | 2.5 | 0.579268 | 0.6 | 1.3 |  |  |  | 1,32256478447104 | 82.2 | 0.1 |  |  |  |
| 03 | 7.60458 | 0.962457 | 0.2 | 0.3 | 0.968194 | 0.1 | 0.2 | 3,403564453125 | 0.7 | 0.03 | 3,21234130859375 | 0.4 | 0.01 |  |  |  |
| 04 | 7.98044 | 1.00342 | 0.2 | 1 | 1.00545 | 0.5 | 1 | 4,23018479857196 | 2.4 | 0.05 | 4,09599689078896 | 1.17 | 0.02 |  |  |  |
| 05 | 4.46063 | 0.565104 | 0.2 | 0.1 | 0.566023 | 0.7 | 0.1 | 1,67895630038184 | 0.92 | 0.02 | 1,5222740772168 | 0.85 | 0.01 |  |  |  |
| 06 | 4.59867 | 0.579276 | 2.9 | 1 | 0.581078 | 0.4 | 1 |  |  |  |  | 11.2 | 0.03 |  |  |  |
| 07 | 7.70882 | 0.969818 | 1 | 1.5 | 0.971946 | 0.7 | 1.2 |  | 14.5 | 0.08 |  | 7.2 | 0.04 |  |  |  |
| 08 | 7.99755 | 1.00022 | 5 | 10 | 1.00186 | 4.5 | 9 |  |  |  |  | 402 | 0.3 |  |  |  |
| 09 | 7.99442 | 1.00003 | 83 | 161 | 1.0028 | 38 | 86 |  |  |  |  |  |  |  |  |  |
| 10 | 7.06808 | 0.892107 | 0.5 | 0.8 | 0.894536 | 0.3 | 0.7 |  | 2.08 | 0.03 |  | 2.91 | 0.03 |  |  |  |
| 11 | 7.63922 | 0.959723 | 3 | 4.3 | 0.962578 | 2.1 | 4.1 |  |  |  |  |  |  |  |  |  |
| 12 | 4.7736 | 0.607964 | 0.1 | 0.2 | 0.633102 | 0.9 | 2.4 |  |  |  |  | 2.4 | 0.01 |  |  |  |
| 13 | 4.14819 | 0.522799 | 11 | 3 | 0.523125 | 1.1 | 2.6 |  |  |  |  | 113 | 0.4 |  |  |  |
| 14 | 4.12092 | 0.982488 | 19 | 4 | 0.519656 | 0.3 | 0.5 |  |  |  |  | 236 | 0.3 |  |  |  |
| 15 | 7.88512 | 0.995968 | 0.3 | 0.6 | 0.999823 | 2.5 | 5.9 |  |  |  |  | 1.2 | 0.02 |  |  |  |
| 16 | 6.36054 | 0.798257 | 5.6 | 7.2 | 0.799752 | 2.9 | 6 |  |  |  |  |  |  |  |  |  |
| 17 | 7.53129 | 0.945921 | 4.4 | 6.8 | 0.94846 | 3 | 6.6 |  |  |  |  |  |  |  |  |  |
| 18 | 6.35317 | 0.797354 | 5.8 | 7.4 | 0.799321 | 0.6 | 1.1 |  |  |  |  |  |  |  |  |  |
| 19 | 7.925 | 0.997829 | 0.7 | 1.3 | 1.00082 | 0.6 | 1.2 |  |  |  |  | 4 | 0.04 |  |  |  |
| 20 | 7.98104 | 1.00847 | 0.2 | 0.5 | 1.01172 | 0.2 | 0.5 |  |  |  |  | 0.9 | 0.02 |  |  |  |
| 21 | 7.98081 | 1.00841 | 0.2 | 0.5 | 1.01164 | 0.2 | 0.5 |  |  |  |  | 0.9 | 0.02 |  |  |  |
| 22 | 7.99139 | 1.00091 | 2.4 | 4.8 | 1.00359 | 2.5 | 4.7 |  |  |  |  | 61 | 0.2 |  |  |  |
| 23 | 7.52298 | 0.944369 | 4.2 | 6.7 | 0.947771 | 3.1 | 6.1 |  |  |  |  |  |  |  |  |  |
| 24 | 7.80386 | 0.519587 | 0.8 | 1.3 | 0.985006 | 0.6 | 1.1 |  |  |  |  | 4 | 0.04 |  |  |  |
| 25 | 0.011969 | 0.125179 | 0.7 | 0.2 | 0.125179 | 0.1 | 0.1 |  |  |  |  |  |  |  |  |  |
| 26 | 7.844 | 1.53222 | 0.01 | 0.01 | 1.53564 | 0.01 | 0.01 |  | 0.02 | 0.005 |  | 0.01 | 0.002 |  |  |  |
| 27 | 0.448472 | 0.197207 | 0.01 | 0.003 | 0.197207 | 0.01 | 0.003 |  |  |  |  | 0.008 | 0.002 |  |  |  |
| 28 | 3.58479 | 0.583333 | 0.01 | 0.005 | 0.608669 | 0.01 | 0.005 |  |  |  |  | 0.01 | 0.002 |  |  |  |
| 29 | 7.06614 | 0.887368 | 48.6 | 75 | 0.890281 | 30.3 | 38.3 |  |  |  |  |  |  |  |  |  |
| 30 | 7.54185 | 0.945913 | 38 | 40 | 0.948462 | 18 | 36 |  |  |  |  |  |  |  |  |  |
| 31 | 6.51992 | 0.818173 | 35 | 46 | 0.820646 | 20 | 42 |  |  |  |  |  |  |  |  |  |
| 32 | 6.1733 | 0.774598 | 156 | 179 | 0.776915 | 63 | 155 |  |  |  |  |  |  |  |  |  |
| 33 | 6.48801 | 0.813454 | 7.5 | 9.8 | 0.815691 | 4 | 9 |  |  |  |  |  |  |  |  |  |
| 34 | 6.5459 | 0.822828 | 20 | 25 | 0.82835 | 11 | 21 |  |  |  |  |  |  |  |  |  |
| 35 | 5.82342 | 0.731739 | 3.7 | 4.6 | 0.733033 | 1.5 | 3.7 |  |  |  |  |  |  |  |  |  |
| 36 | 5.93209 | 0.747421 | 1.4 | 1.6 | 0.750471 | 0.5 | 1.3 |  |  |  |  |  |  |  |  |  |

# Сравнительный анализ алгоритмов

Для подавляющего большинства файлов алгоритмы сжатия LZ77 оказались неэффективны. Это связано с малым количеством повторений в байт-кодах файлов, в то время как алгоритмы Шеннона и Хаффмана сжимали файлы лучше.

По скорости сжатия наихудший результат у алгоритмов группы LZ, но при распаковке они дают наоборот лучший временной результат.

Что касается алгоритмов Хаффмана и Фано – по эффективности сжатия они дают всегда примерно одинаковые результаты. В незначительных случаях Фано быстрее.

# Заключение

Из всех исследуемых нами алгоритмов наиболее оптимальным в чистом виде является алгоритм Фано.