



Алгоритмы и структуры данных

Меня хорошо видно && слышно?



Защита проекта

Тема: Алгоритмы сжатия данных на примере алгоритмов Хаффмана и RLE



Семенов Вадим

Инженер разработчик в компании
“Лаборатория комфорта”



План защиты



Цели проекта

Используемые
технологии

Алгоритм Хаффмана

Алгоритм RLE

Сравнение
эффективности

Выводы

Цели проекта

1. Реализовать программу сжатия данных на основе алгоритма Хаффмана
2. Реализовать программу сжатия данных на основе алгоритма RLE
3. Произвести оценку эффективности сжатия алгоритмов

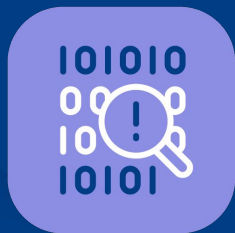
Используемые технологии

1. C# (Visual Studio + Resharper)

2. .Net

3. Linq

4. WPF





Алгоритм Хаффмана

Алгоритм Хаффмана

Алгоритм Хаффмана — жадный алгоритм префиксного кодирования алфавита. Был разработан в 1952 году аспирантом Массачусетского технологического института Дэвидом Хаффманом при написании им курсовой работы.

Метод кодирования состоит из двух основных этапов:

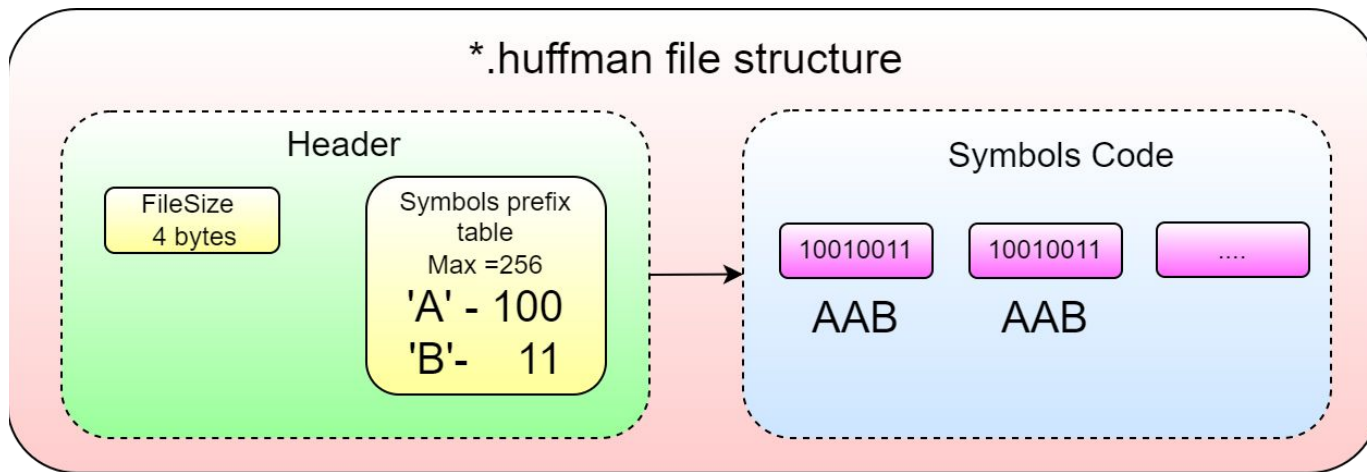
-  Построение оптимального кодового дерева.
-  Построение отображения код-символ на основе построенного дерева.

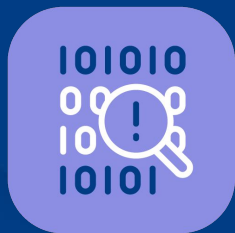
Алгоритм Хаффмана

- Алгоритм Хаффмана на входе получает таблицу частотностей символов(байтов) в файле.
- На основании этой таблицы строится дерево кодирования Хаффмана. Т.е. каждому символу (байту) назначается префикс например - 10011
- Далее в сжатый файл записывается последовательность префиксов заменяющая символы(байты)



Схема структуры файла *.huffman





Алгоритм RLE

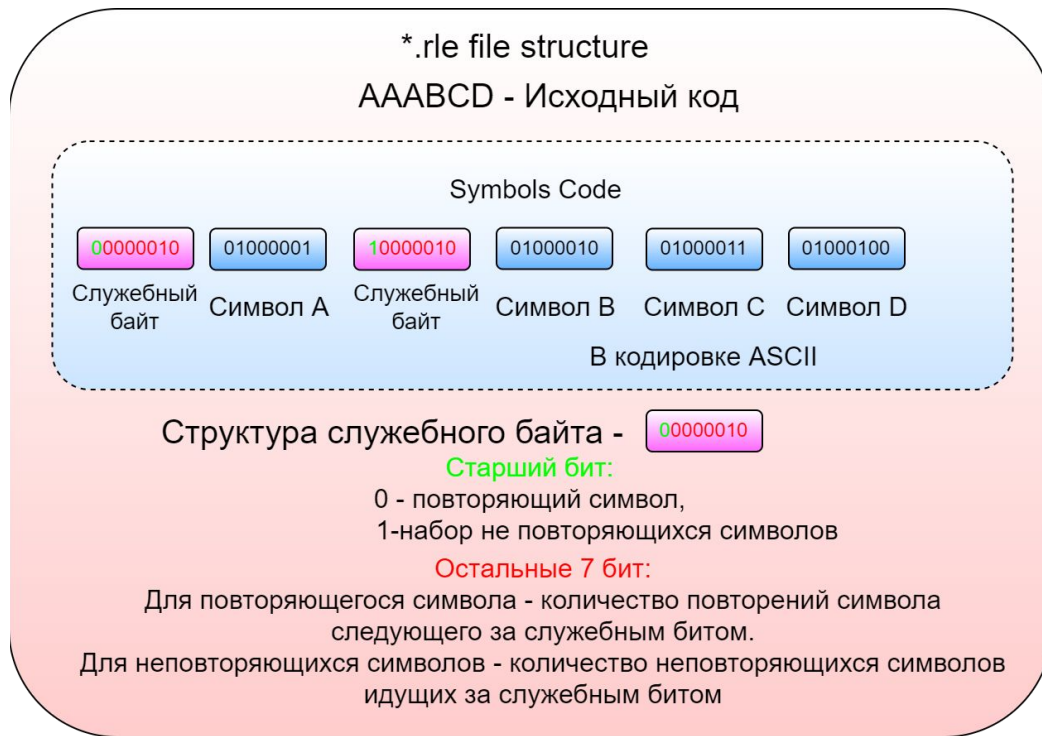
Алгоритм RLE

Алгоритм RLE — Кодирование длин серий (англ. run-length encoding, RLE) или кодирование повторов — алгоритм сжатия данных, заменяющий повторяющиеся символы (серии) на один символ и число его повторов.

Серией называется последовательность, состоящая из нескольких одинаковых символов. При кодировании (упаковке, сжатии) строка одинаковых символов, составляющих серию, заменяется строкой, содержащей сам повторяющийся символ и количество его повторов.

Один из самых простых алгоритмов сжатия.

Схема структуры файла *.rle





Оценка эффективности алгоритмов

Оценка эффективности сжатия

Таблица эффективности сжатия файлов

Название файла	Тип файла	Размер файла до сжатия, МБ	Размер файла *.huffman, МБ	Сжатие в %	Размер файла *.rle, МБ	Сжатие в %
Drawing.bmp	графический (несжатый BMP)	143.043	71.743	49%	67.032	54%
Sound.wav	звуковой (несжатый wav)	36.132	32.458	11%	36.129	1%
Text.txt	текстовый (книга на русском)	0.713	0.377	48%	0.719	-1%



Выводы сравнения сжатий

1. Наиболее эффективным для всех типов файлов показал себя алгоритм Хаффмана.
2. RLE очень эффективен для сжатия графики без градиентов (при повторяющихся последовательностях битов).
3. Алгоритм Хаффмана наиболее эффективен для файлов с маленьким диапазоном значений байтов.

Спасибо за внимание!

Исходный код программы на языке C# можно посмотреть по ссылке:

https://github.com/vadsemenov/OTUS_Algorithms_and_data_structures/tree/main/26.Compression%20algorithms/Compression

