МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
по теме «Алгоритмы сжатия»

Студент гр. 1321 _____ Оршак И.А.

Преподаватель _____ Пестерев Д.О.

Санкт-Петербург
2024

1. Цель лабораторной работы:

Знакомство с алгоритмами сжатия данных и их реализация.

2. Задача на лабораторную работу:

2.1. Реализовать следующие алгоритмы сжатия и преобразования
а) Алгоритм Хаффмана (HA);
б) Алгоритм RLE;
в) Алгоритм LZ77;
г) Преобразование BWT;
д) Преобразование MTF.

2.2. И их комбинации:
а) BWT+RLE;
б) BWT+MTF+HA;
в) BWT+MTF+RLE+HA;
г) LZ77+HA.

2.3. Исследовать эффективность алгоритмов на различных наборах данных.

2.4. Рассчитать коэффициенты сжатия для различных наборов данных.

2.5. Рассчитать энтропию данных для алгоритмов энтропийного кодирования.

3. Ход лабораторной работы

3.1. Реализация программы

3.1.1. Исходный код

Программа для исследования алгоритмов сжатия разработана на языке Golang v. 1.22.2 на операционной системе Linux Fedora 39.

Исходный код программы размещен в репозитории GitHub: https://github.com/stensh1/zipper.

Алгоритмы сжатия/преобразования имплементированы в виде отдельных пакетов в папке “./pkg/”. В каталоге src расположено несколько тестовых файлов разных форматов для сжатия. В каталоге “./tests” собраны функции тестирования имплементированных алгоритмов. Файлом для сборки служит файл “./main.go” в корневом каталоге проекта.

Каждый алгоритм реализован в виде методов структуры, описанной в файле “models.go” для каждого алгоритма. Каждая структура удовлетворяет типу интерфейса Zipper. Объявление интерфейса показано на рисунке 1.

Интерфейс имеет два метода: Encode и Decode, которые принимают на вход слайс байт и возвращают слайс байт и ошибку.

```
// Zipper is an interface for all compression/decompression algs
type Zipper interface {  2 usages  6 implementations  stensh1
    Encode([]byte) ([]byte, error)  6 implementations
    Decode([]byte) ([]byte, error)  6 implementations
}
```

Рисунок 1. Интерфейс Zipper

3.1.2 Использование

Выбор режима работы программы (компрессия/декомпрессия), исходного файла и используемых алгоритмов реализовано с помощью консольных флагов запуска бинарного файла:

- -zip / -unzip - режим работы алгоритмов;
- -with “...” - флаг с последующей строкой, указывающий, какие алгоритмы и в какой последовательности следует использовать:
    - “ha” - алгоритм Хаффмана;
    - “rle” - алгоритм кодирования повторов RLE;
    - “bwt” - преобразование Барроуза-Уилера;
    - “mtf” - преобразование move-to-front;
    - “lz77” - алгоритм сжатия LZ77.

Для реализации последовательности преобразований следует указывать алгоритмы в необходимой последовательности через дефис: “bwt-mtf-ha”.

- -f “...” - флаг с последующей строкой, указывающий на расположение файла, который нужно сжать.

3.1.3 Тестирование

Для проверки корректности работы реализованных алгоритмов и правильной компрессии/декомпрессии данных были написаны функции тестирования, лежащие в каталоге “./tests/”.

В качестве тестовых данных были выбраны следующие варианты: пустая байта, строка из двух различных байт, строка из двух одинаковых байт, строка из трех различных байт, строка из трех одинаковых байт и рандомная строка, содержащая повторения. Данный набор показан на рисунке 2.

```
{ name: "Empty file",  in: []byte(""),  expected: []byte("")},
{ name: "Single byte",  in: []byte("0"),  expected: []byte("0")},
{ name: "Two different bytes",  in: []byte("ab"),  expected: []byte("ab")},
{ name: "Two equal bytes",  in: []byte("aa"),  expected: []byte("aa")},
{ name: "Three different bytes",  in: []byte("abc"),  expected: []byte("abc")},
{ name: "Three equal bytes",  in: []byte("aaa"),  expected: []byte("aaa")},
{ name: "Random string",  in: []byte("aaaaaaaaaaaaaaaaaaaaaaaabbbbbdweewfewqqqqqqqqqqqqqqqqqqqq111234gff"),
    expected: []byte("aaaaaaaaaaaaaaaaaaaaaaaabbbbbdweewfewqqqqqqqqqqqqqqqqqqqq111234gff")},
```

Рисунок 2. Набор тестовых данных

Последовательный запуск всех тестировочных функций для каждого алгоритма реализован в виде команды в “./makefile”: test_all. Проведем тестирование алгоритмов. Результаты тестирования приведены на рисунке 3.

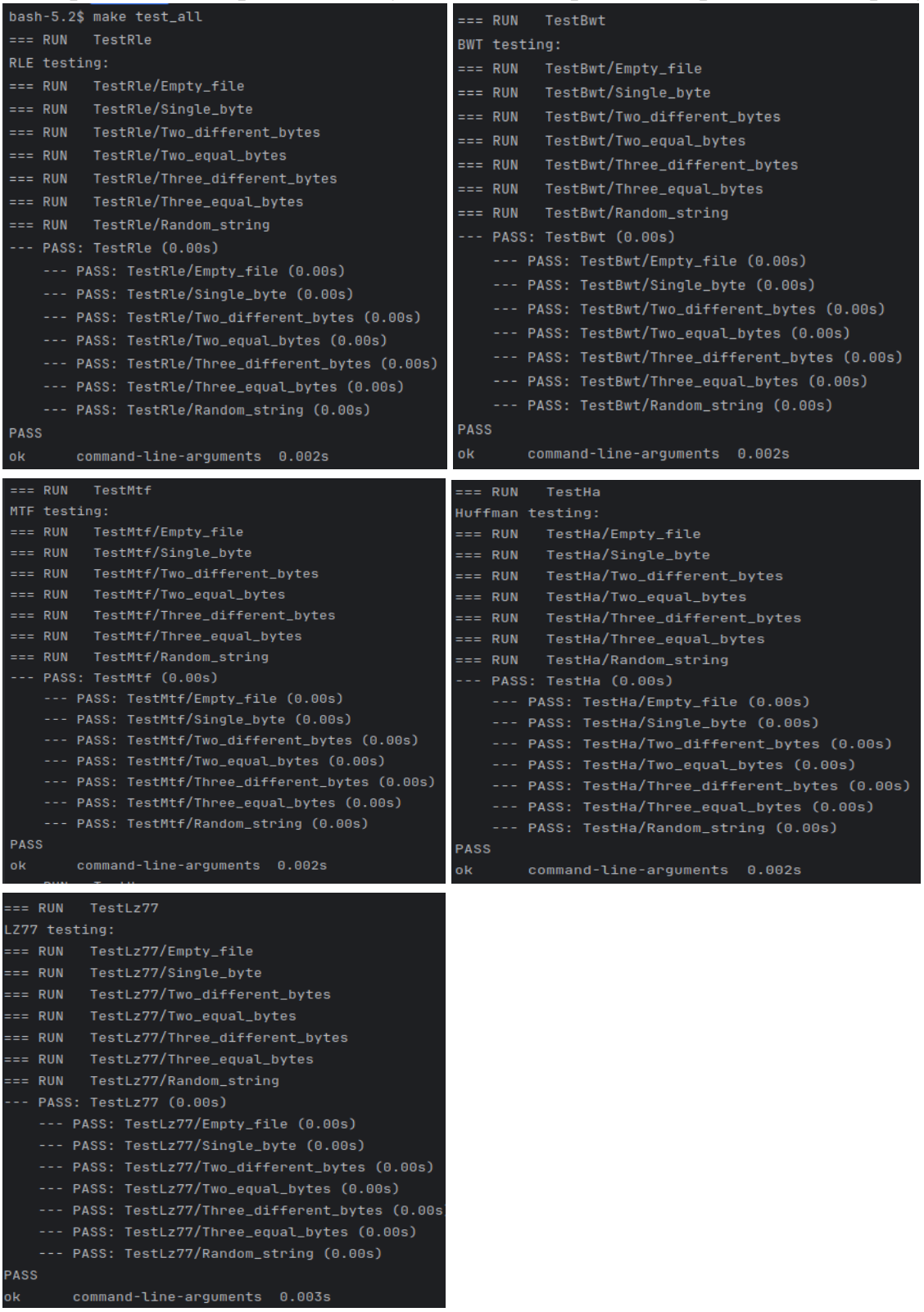

Рисунок 3. Результат тестирования алгоритмов

Вывод: все тесты пройдены, алгоритмы верно отработали на тестовом наборе данных.

3.2. Исследование эффективности компрессоров

3.2.1. Алгоритм Хаффмана:

Результаты работы алгоритма представлены в таблице 1.

Таблица 1. Результаты сжатия алгоритма Хаффмана

Входные данные	**Размер входного файла**	**Размер выходного файла**	**Коэффициент сжатия**	**Энтропия входного файла**
Текст на латинице в формате .txt	5.9 кБ	3.5 кБ	1.67	4.369
Текст на кириллице в формате .odt	62.6 кБ	63.8 кБ	0.98	7.973
Цветное изображение	1.8 МБ	1.8 МБ	0.99	7.990
Файл .mp3	10 МБ	9.9 МБ	1.001	7.955

Вывод: алгоритм Хаффмана является алгоритмом энтропийного кодирования. Для трех из 4 сжимаемых файлов энтропия почти равна 8, что означает кодирование одного байта исходных данных в среднем 8 битами (одним байтом), из этого очевидны показатели коэффициентов сжатия примерно равные 1 во всех трех последних случаях.

3.2.2. Алгоритм RLE:

Результаты работы алгоритма представлены в таблице 2.

Таблица 2. Результаты сжатия алгоритма RLE

Входные данные	**Размер входного файла**	**Размер выходного файла**	**Коэффициент сжатия**
Текст на латинице в формате .txt	5.9 кБ	6 кБ	0.980
Текст на кириллице в формате .odt	62.6 кБ	62.8 кБ	0.995
Черно-белое изображение	9.2 кБ	9.3 кБ	0.989
Файл .mp3	10 МБ	10 МБ	0.993

Вывод: алгоритм хорошо работает только при наличии большой серии повторений байтов. Реальные файлы, обычно, не подходят для такого

кодирования, для них, предварительно, требуется первичная обработка, например преобразованием BWT.

3.2.3. Компрессор BWT+RLE:

Результаты работы компрессора представлены в таблице 3.

Таблица 3. Результаты сжатия компрессора BWT + RLE

Входные данные	**Размер входного файла**	**Размер выходного файла**	**Коэффициент сжатия RLE**
Текст на латинице в формате .txt	5.9 кБ	4.8 кБ	1.234
Текст на кириллице в формате .odt	62.6 кБ	62.1 кБ	1.007
Черно-белое изображение	9.2 кБ	9.3 кБ	0.988

Вывод: преобразование BWT помогает более эффективной отработке алгоритма RLE, что видно из результатов в 1 и 2 строке таблицы 3.

3.2.4. Компрессор BWT+MTF+HA:

Результаты работы компрессора представлены в таблице 4.

Таблица 4. Результаты сжатия компрессора BWT + MTF + HA

Входные данные	**Размер входного файла**	**Размер выходного файла**	**Коэффициент сжатия HA**	**Энтропия входного файла**
Текст на латинице в формате .txt	5.9 кБ	3.1 кБ	1.923	4.368
Текст на кириллице в формате .odt	62.6 кБ	63.6 кБ	0.983	7.973
Черно-белое изображение	9.2 кБ	10.4 кБ	0.878	7.952

Вывод: преобразование bwt+mtf преобразовывает входные данные в числовые, удобные для применения к ним алгоритма Хаффмана, поэтому, по

сравнению с обычным запуском алгоритма Хаффмана, коэффициенты сжатия после преобразований bwt+mtf увеличились.

3.2.5. Компрессор BWT+MTF+RLE+HA:

Результаты работы компрессора представлены в таблице 5.

Таблица 5. Результаты сжатия компрессора BWT + MTF + RLE + HA

Входные данные	**Размер входного файла**	**Размер выходного файла**	**Коэфф. сжатия RLE**	**Коэфф. сжатия HA**	**Энтропия входного файла**
Текст на латинице в формате .txt	5.9 кБ	3.5 кБ	1.131	1.715	4.368
Текст на кириллице в формате .odt	62.6 кБ	63.4 кБ	1.006	0.987	7.973
Черно-белое изображение	9.2 кБ	10.5 кБ	0.988	0.870	7.952

Вывод: компрессор отработал хуже, чем компрессор в пункте 3.2.4, это связано с увеличением объема строки на входе в алгоритм Хаффмана, при энтропии почти равной 8.

3.2.6. Алгоритм LZ77:

Результаты работы алгоритма представлены в таблице 6.

Таблица 6. Результаты сжатия алгоритма LZ77

Входные данные	**Размер входного файла**	**Размер выходного файла**	**Коэфф. сжатия LZ77**
Текст на латинице в формате .txt	5.9 кБ	26 кБ	0.228
Текст на кириллице в формате .odt	62.6 кБ	489.9 кБ	0.127
Черно-белое изображение	9.2 кБ	72.9 кБ	0.125

Вывод: алгоритм плохо отрабатывает на исходных данных, это может быть связано с неправильно подобранным окном поиска или неудачными данными.

3.2.7. Компрессор LZ77+HA:

Результаты работы компрессора представлены в таблице 7.

Таблица 7. Результаты сжатия компрессора LZ77 + HA

Входные данные	**Размер входного файла**	**Размер выходного файла**	**Коэфф. сжатия LZ77**	**Коэфф. сжатия HA**	**Энтропия входного файла**
Текст на латинице в формате .txt	5.9 кБ	10.7 кБ	0.228	0.552	4.368
Текст на кириллице в формате .odt	62.6 кБ	180.3 кБ	0.127	0.346	7.973
Черно-белое изображение	9.2 кБ	27.8 кБ	0.125	0.125	7.952

Вывод: при неудачной обработке данных алгоритмом LZ77, алгоритм Хаффмана все же уменьшает выходной вес файла, поэтому результаты компрессии оказываются лучше, чем в п. 3.2.6.

4. Вывод: в ходе выполнения лабораторной работы были реализованы алгоритмы сжатия и преобразования данных: алгоритм Хаффмана, алгоритм RLE, алгоритм LZ77 и преобразования BWT и MTF. Были проведены тесты корректной компрессии/декомпрессии данных, а также исследованы эффективности компрессоров на различных наборах данных.