МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и технологий

Высшая школа искусственного интеллекта

Дисциплина:
ТЕОРИЯ ГРАФИИ

ОТЧЕТ
Лабораторная работа №0:
«Шифратор и дешифратор»

Обучающийся гр. 3530201/10001 Нгуен Куок Дат

Руководитель _____ Востров Алексей Владимирович

Санкт-Петербург 2022

Содержание

Введение

В данной работе необходимо реализовать шифратор и дешифратор для данных алгоритм: Фано и RLE, применяющиеся на текстовый файл 10000 символов. Программно кодировать и декодировать алгоритимами Фано, RLE и также применять двухстепенчатое кодирование; одновременно необходимо определить цену кодирования и коэффициент сжатия.

1 Матиматическое описание

1.1 Кодирование

1.1.1 Алфавитное кодирование

Алфавитное (или побуквенное) кодирование задается схемой (или таблицей кодов) σ

$$\sigma \stackrel{\text{Def}}{=} \langle a_1 \to \beta_1, \dots a_n \to \beta_n \rangle, \quad a_i \in A, \beta_i \in B^*,$$

гдеβ_i - кодовое слово.

Алфавитное кодирование пригодно для любого множества сообщений S:

$$F : A^* \to B^*, \qquad a_{i_1} \dots a_{i_k} = \alpha \in A*, \qquad F(\alpha) = \beta_{i_1} \dots \beta_{i_k}$$

1.1.2 Разделимые и префиксные схемы

Схема σ называется *разделимой*, если

$$\beta_{i_1} \dots \beta_{i_k} = \beta_{j_1} \dots \beta_{j_l} \implies k = l \& \forall t \in 1..k (i_t = j_t)$$

то есть любое слово, составленное из элементарных кодов, единственным образом разлагается на элементарные коды. Алфавитное кодирование с разделимой схемой допускает декодирование.

Схема σ называется префиксной, если элементарный код одной буквы не

является префиксом элементарного кода другой буквы:

$$\neg\beta_i, \beta_j \in V, \beta \in B^*(i \neq j \quad \& \quad \beta_i = \beta_j\beta)$$

Префиксная схема является разделимой, поэтому допускает декодирование.

1.1.3 Цена кодирования

Заданы алфавит $= \{a_1, ..., a_n\}$ и вероятности появления букв в сообщении $P = \langle p_i, \ldots, p_n\rangle$ (p_i — вероятность появления буквы a_i). Не ограничивая общности, можно считать, что $p_1 + \cdots + p_n = 1$ и $p_1 \geq \cdots \geq p_n > 0$.

Для каждой разделимой схемы $\sigma = \langle a_i \to \beta_i\rangle_{i=1}^n$ алфавитного кодирования математическое ожидание коэффицеинта увеличения длины сообщения при кодировании σ называется средней *ценой кодирования* σ при распределении вероятностей P:

$$l_\sigma(P) \stackrel{\text{Def}}{=} \sum_{i=1}^{n} p_i|\beta_i|$$

1.2 Алгоритм Фано

Рекурсивный алгоритм Фано строит разделимую префиксную схему алфавитного кодирования, близкого к оптимальному.

Ввод: частото-убывающий алфавит, начало, конец, кодовая схема

Вывод: кодовая схема (словарь)

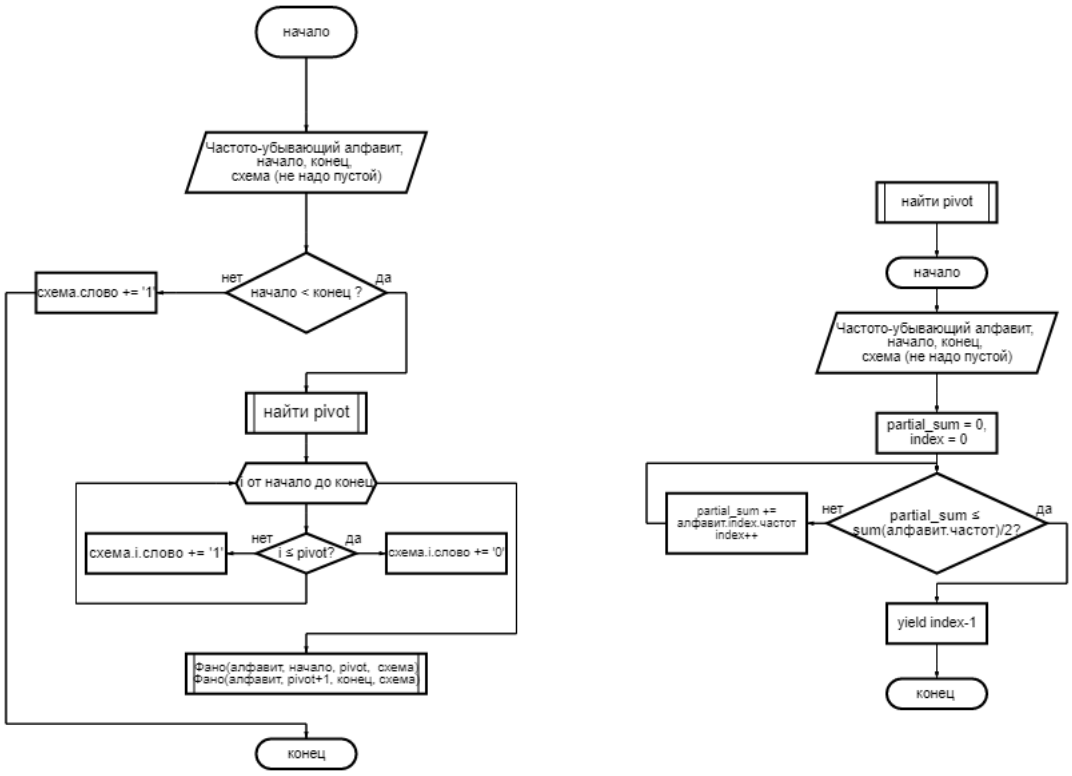


Рис. 1: Блок-схема алгоритма Фано

1.3 Алгоритм RLE

Алгоритм RLE *run-length encoding* - заменяет строку одинаковых символов строкой, содержащей сам повторяющийся символ и количество его повторов.

В этой программе, чтобы избежать использования слишком большого объема данных, если серия имеет больше 255 символов (тогда надо по крайней мере 2 char сохранящие кратность), она будет «выделена».

Ввод: частото-убывающий алфавит, начало, конец, кодовая схема

Вывод: кодовая схема (словарь)

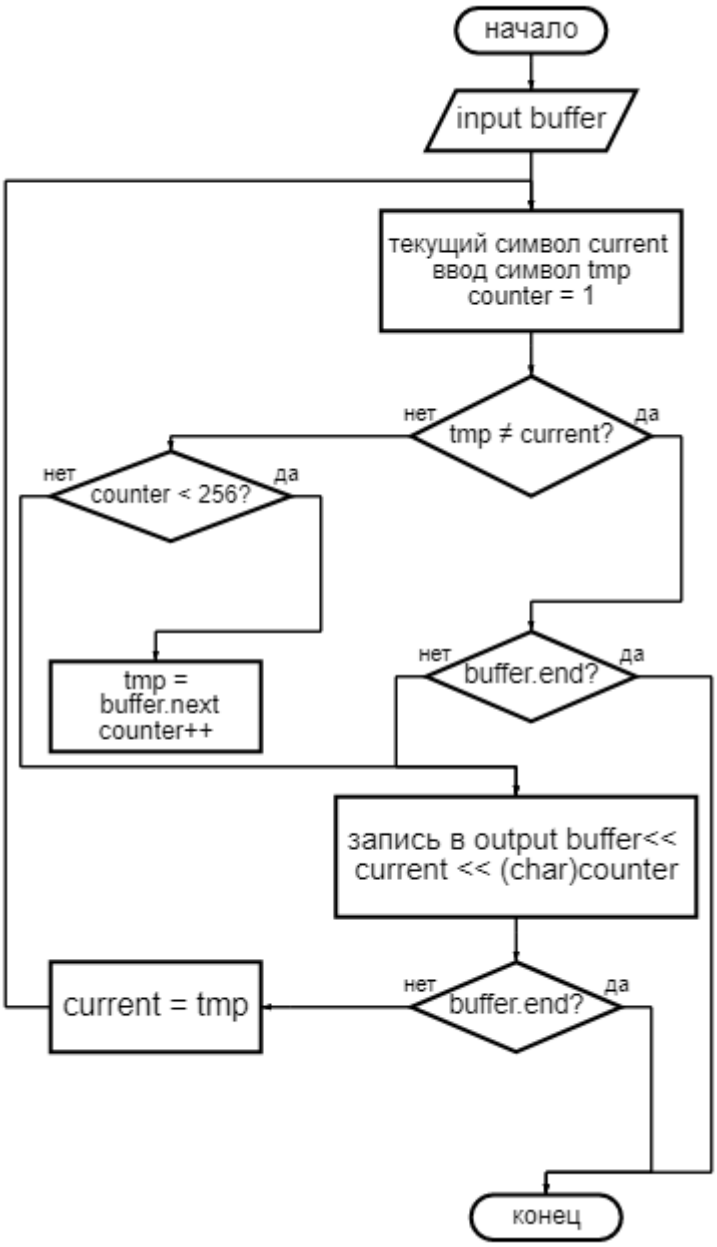


Рис. 2: Блок-схема алгоритма RLE

2 Реализация

2.1 Текстовый генератор

Используя генератор для случаных чисел (*rand()*), выберяют характер по индексу из данной списки.

Вход: Объём файл (10000 характеров), списка символов

Выход: Текстовый файл

2.2 Битовая упаковка

Один главный процедур это битовая упаковка. Поскольку система не позволяет нам напрямую работать с битами, закодированные данные нужно «упаковуют» в строке, чтобы сжать более эффективно.

При отображении с байтов на битов, данные могут быть не кратен байтам, поэтому в конце шифорванной строки нужны несколько «заполненных» битов (padding bit). Одновременно, другая проблема возникает: нулевый характер в конце строки будет пропущен, которое приводит к потере данных. Чтобы решить эту задачу, добавляют один байт в концу, которому назначен количество заполненных битов.

Так как упаковка (и распаковка) работают с битами, файл должно быть открыт в *binary mode*, чтобы избегать то, что данные меняются.

Вход: Последовательность шифруемых символов S, словарь D.

Выход: Строка упаковки.

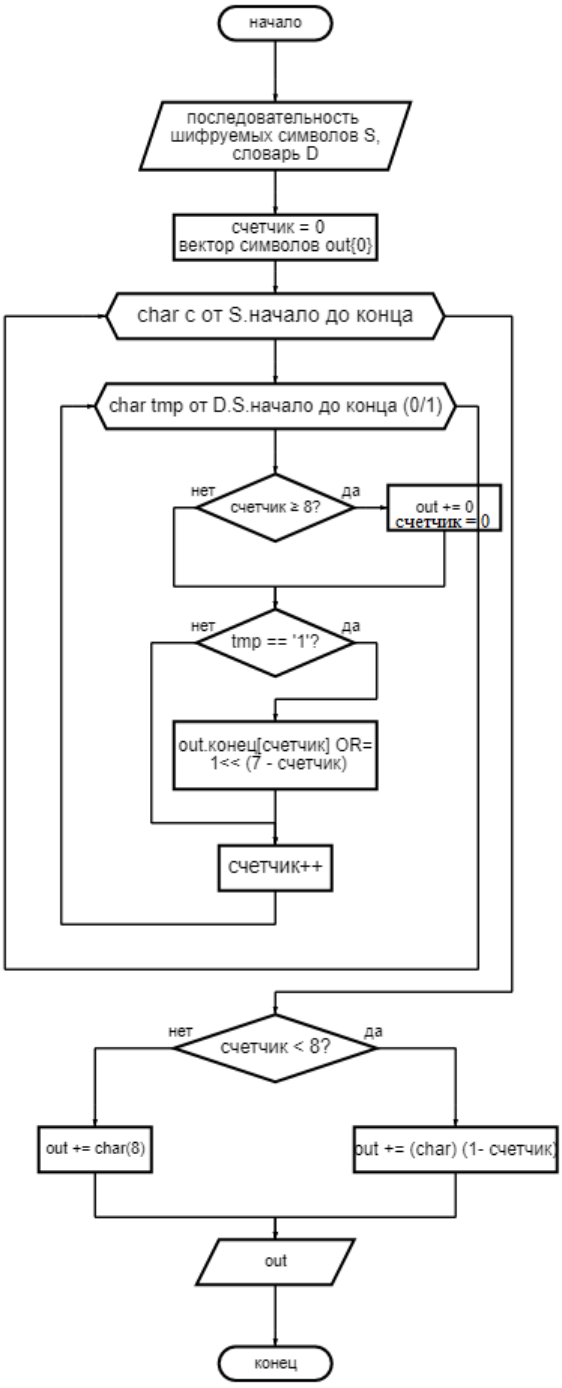


Рис. 3: Блок-схема битовой упаковки

2.3 Дешифрование

2.3.1 Для RLE

Для RLE-шифрованных данных не нужен словарь, только читать символы и переписывать данные.

Вход: Последовательность шифрованных символов S

Выход: Строка перивичных данных

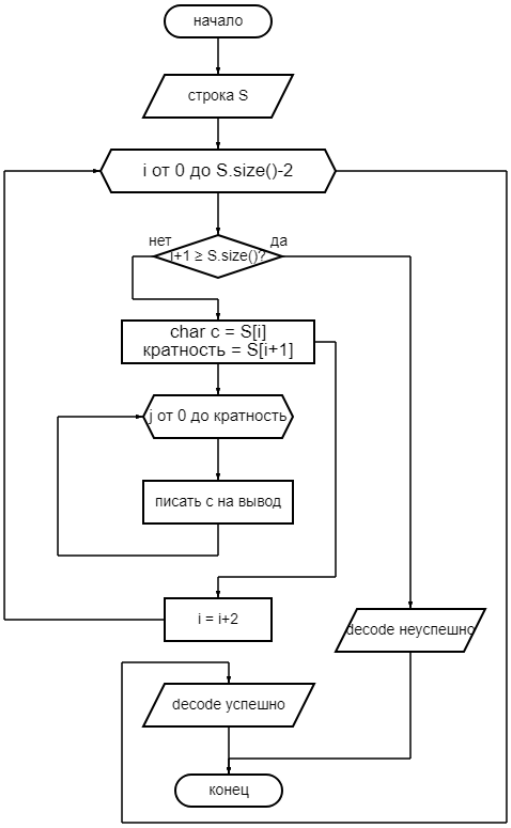


Рис. 4: Блок-схема дешифрования для RLE

2.3.2 Для Фано

Для Фано-шифорванных данных нужны внешний словарь и процедура «распаковки».

Вход: Последовательность шифрованных символов S, словарь D

Выход: Строка перивичных данных

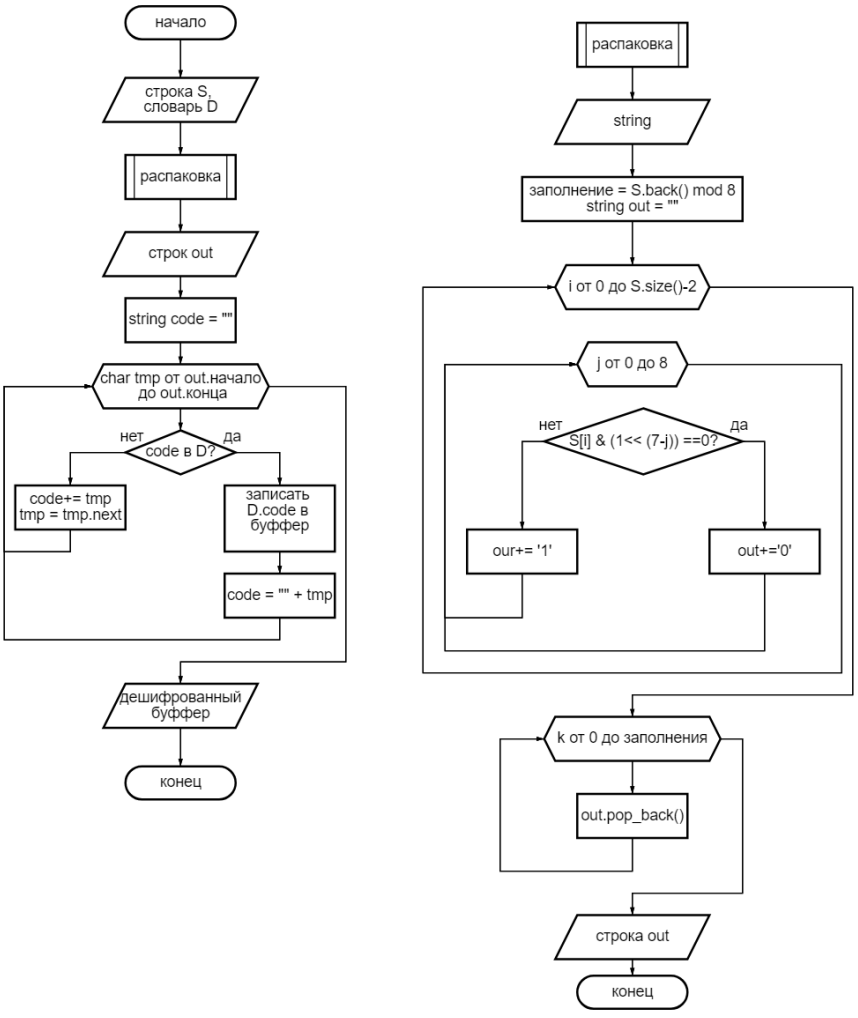


Рис. 5: Блок-схема дешифрования для Фано

2.4 Проверка дешифрованных данных

Данные проверят по символам, внутри функции сравнение 2 строки. Если они равны, то дешифрованние нормально работает.

Вход: 2 строки

Выход: 0 если равны, другие если не равны

2.5 Вычисление коэффициента сжатия

$$\text{коэффициент сжатия} = \frac{\text{Объём исходных данных}}{\text{Объём шифрованных данных}}$$

Вход: 2 строки

Выход: Вывод коэффициент сжатия на экран

3 Тестирование

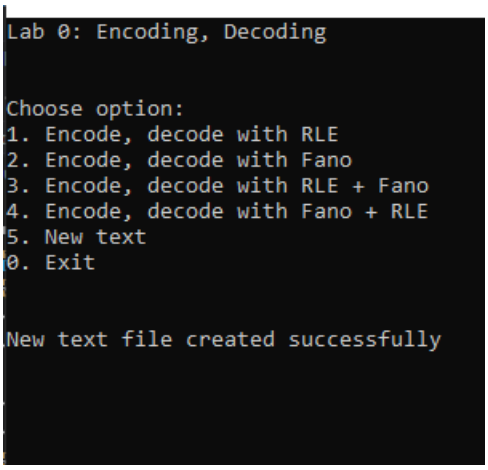
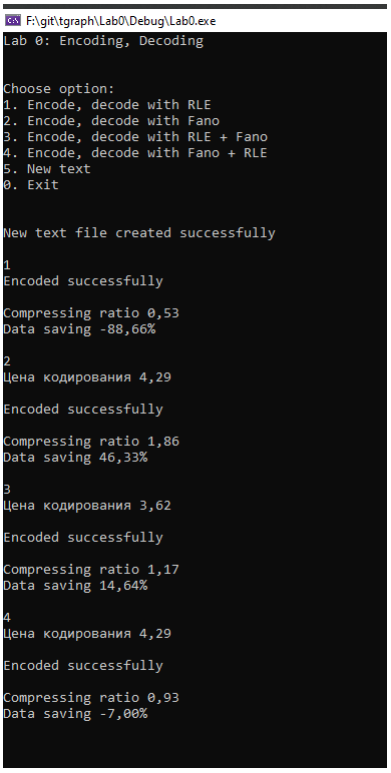


Рис. 6: Новый текст файл созданый

В фигуре 6, новый текст **INP.txt** с 10000 символов создан.



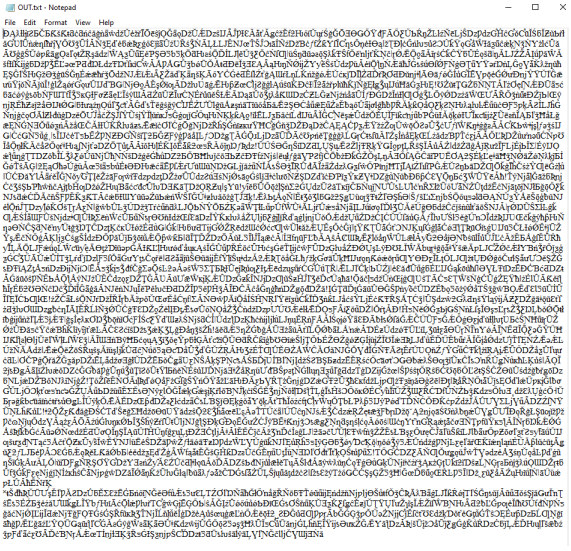


Рис. 7: Шифрование

В фигуре 7, характеры в **INP.txt** шифруются и записываются в **OUT.txt**. Цену кодирования и коэффициенты сжатия вычислены.

Видимо, что RLE-шифорвание увеличивает объём файла. Однако, Фано-шифрование сжает файл на 50%.

Сравнивая двухстепенчатые RLE-Фано и Фано-RLE шифрования, можем заключить, что с данной файлом, RLE-Фано шифрование более эффективно.

Рис. 8: Дешифрование

В фигуре 8, дешифруемые данные записываются в **check.txt**

Заключение

В данной работы успешно написана программа, реализующая шифратор и дешифратор с алгоритмами Фано, RLE и соответственными двухстепечатыми. Шифрование и дешифрование произошлы, цена кодирования и коэффициенты сжатия определены. Видимо, что Фано-шифрование самое эффективное, и двухстепенчатое RLE-Фано шифрование более эффективно, чем Фано-RLE шифрование.

Преимущества программы заключаются в том, что реализация в общем виде, для любой списки символов. С идеями о битовой упаковке и распаковке могут создать функцию в отдельную библиотеку для других целей.

Недостаток программы заключается в том, что исходный код не оптимизирован, особенно с алгоритмом RLE: шифрованные данные больше исходных. Процедуры упаковки и распаковки написаны внутри функции шифрования, значит им надо переписывать, иначе не могут повторно использовать.

Данная лабораторная работа была реализована на языке C++ в среде программирования Visual Studio 2019.

Список литературы

[1] Ф.А. Новиков «Дискретная математика для программистов: Учебник для вузов». 3-е издание.- СПБ.: Питер, 2009. -384 с.: ил. - (Серия «Учебник для вузов»)

День обращения: 10.02.2023

[2] Конспект лекций Вострова А.В по дискретной математике.

День обращения: 10.02.2023

[3] Алгоритм RLE https://en.wikipedia.org/wiki/Run-length_encoding

День обращения: 10.02.2023