Санкт-Петербургский Политехнический Университет

Высшая школа прикладной математики и вычислительной физики, ФизМех

01.03.02 Прикладная математика и информатика

Лабораторная работа №1

Дисциплина “Дискретная математика”

Тема “Кодирование информации”

Вариант “Алгоритм Лемпеля-Зива”

Выполнил студент гр. 5030102/20201 Мелко Тимофей Андреевич

Поставленная задача

Реализовать алгоритм Лемпеля-Зива сжатия и распаковки информации с поддержкой взаимодействия с алгоритмом

Используемый язык программирования

Python 3.12.6

Описание алгоритма Лемпеля-Зива

Алгоритм LZ78 сжатие (input):

```
D = [""]  // Инициализация словаря с пустой строкой
k = 0  // Указатель на текущую позицию входных данных
output = []  // Результат сжатия

пока k < длина(input):
    // Найти самое длинное совпадение в словаре D
    p = FD(D, input, k)
    l = длина(D[p])  // Длина совпадающей строки

    // Записать индекс и следующий символ
    если k + l < длина(input):
        output.добавить(пару(p, input[k + l]))
        D.добавить(D[p] + input[k + l])  // Добавить новую строку в словарь

    k = k + l + 1  // Переместить указатель

вернуть output  // Сжатые данные

Функция FD(D, input, k):
    max_length = 0
    index = 0

    для i от 0 до длина(D):
        если D[i] == input[k:k + длина(D[i])]:
            если длина(D[i]) > max_length:
```

```
            max_length = длина(D[i])
            index = i

    вернуть index

Алгоритм LZ78 декомпрессия (compressed_input):
    D = [""]  // Инициализация словаря с пустой строкой
    output = ""  // Декодированный результат

    для каждой (p, символ) в compressed_input:
        // Восстановить строку из словаря и добавить следующий символ
        строка = D[p] + символ
        output = output + строка
        D.добавить(строка)  // Добавить новую строку в словарь

    вернуть output  // Исходные данные
```

Пример работы алгоритма

Возьмем строку abcabcabc

Сжатие

Шаг	Подстрока	Совпадение в словаре	Индекс p	Следующий исмвол q	Запись в словарь	Содержание словаря
1	a	“”	0	a	a	["", "a"]
2	b	“”	0	b	b	["", "a", "b"]
3	c	“”	0	c	c	["", "a", "b", "c"]
4	ab	“a”	1	b	ab	["", "a", "b", "c", "ab"]
5	ca	“c”	3	a	ca	["", "a", "b", "c", "ab", "ca"]
6	bc	“b”	2	c	bc	["", "a", "b", "c", "ab", "ca", "bc"]

Распаковка

Шаг	Индекс p	Символ q	Восстановленная строка	Запись в словарь	Содержание словаря
1	0	a	a	a	["", "a"]
2	0	b	b	b	["", "a", "b"]
3	0	c	c	c	["", "a", "b", "c"]
4	1	b	ab	ab	["", "a", "b", "c", "ab"]
5	3	a	ca	ca	["", "a", "b", "c", "ab", "ca"]
6	2	c	bc	bc	["", "a", "b", "c", "ab", "ca", "bc"]

Область применения алгоритма

Алгоритм LZ78 эффективен для текстов с повторяющимися последовательностями символов. Для кодирования символов будем записывать в бинарный файл индекс p с помощью 4 байт и также выделять 1 байт на запись символа q. Что может накладывать ограничение на входной размер файла, при большом входном файле будет необходимо расширение записи индекса p до большего количества байт.

Сравнения размеров файлов

Возьмем несколько файлов для примера состоящие из большого количества повторяющихся элементов и длинные случайные последовательности. Для файла, состоящего из повторяющихся элементов весом 2900 байт его сжатый файл будет весить 1505 байт. Для случайных последовательностей весом 32999 байт, сжатый файл будет иметь вес 61224 байта, что показывает о плохой сжимаемости для случайных элементов с низкой частотой повтора. Для файла весом 560 128 байт в виде обычного текста, его конечный размер получился 466 974.

Оптимизация

Для оптимизации кода можно изменить функцию FD для оптимального поиска наибольшей последовательности. Например, ввести структуру данных дерево, в котором каждый узел дерева содержит строку (или часть строки) и организовать узлы так, чтобы для каждого узла все строки в левом поддереве меньше, а в правом больше.

Вывод

Алгоритм LZ78 является эффективным методом сжатия данных, основанным на выявлении и кодировании повторяющихся последовательностей символов. Он особенно хорошо подходит для текстов с высокой степенью повторяемости, что делает его полезным для различных приложений, включая сжатие текстов, изображений и других форм данных. Были получены примерные результаты для текста с высокой повторяемостью небольшого размера в 50% и для обычного текста большого размера в 18%.