МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

ЛАБОРАТОРНАЯ РАБОТА

Алгоритм «Стопка книг»

студента 4 курса 431 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Серебрякова Алексея Владимировича

Научный руководитель

доцент, к. п. н. _____ А. С. Гераськин

подпись, дата

Саратов 2022

Метод "Стопка книг"

Метод сжатия данных "Стопка книг" был впервые предложен Б. Я. Рябко и затем переоткрыт на Западе под названием Move-To-Front (MTF).

Идея метода состоит в следующем. Буква сообщения кодируется числом, представляющим ее текущий номер в алфавите источника. Затем она ставится в начало алфавита, при этом остальные буквы сдвигаются на одну позицию вправо. То есть буквы алфавита переупорядочиваются, с тем чтобы наиболее часто встречающиеся буквы находились ближе к началу.

Рассмотрим пример. Пусть алфавит источника $A=\{a,b,c,d,e\}$ и генерируется сообщение *baadaade*... Покажем, как меняется порядок букв и какие числа формируются по мере поступления символов от источника.

Сообщение		*b*	*a*	*a*	*d*	*a*	*a*	*d*	*e*
Позиции букв в алфавите	0	*a*	*b*	*a*	*a*	*d*	*a*	*a*	*d*
	1	*b*	*a*	*b*	*b*	*a*	*d*	*d*	*a*
	2	*c*	*c*	*c*	*c*	*b*	*b*	*b*	*b*
	3	*d*	*d*	*d*	*d*	*c*	*c*	*c*	*c*
	4	*e*	*e*	*e*	*e*	*e*	*e*	*e*	*e*
Код		1	1	0	3	1	0	1	4

Последовательность 11031014 еще не является "кодом": каждое из входящих в нее чисел необходимо закодировать.

Сжатие достигается за счет того, что чаще встречающиеся буквы имеют меньшие номера позиций, которые кодируются более короткими кодами.

Способ декодирования очевиден.

Код программы

```
import time


opt = int(input('Сжать - 0, Разжать - 1\n'))

if opt == 0:
    start = time.perf_counter()
    f = open('studies\\tkisi\\Тест_8.txt')

    d = {}
    alphabet = ''
    text = ''

    while True:
        string = f.readline()
        if not string:
            break
        else:
            text += string
            for i in string:
                if i == '\n':
                    j = '*'
                elif i == ' ':
                    j = '_'
                else:
                    j = i
                if d.get(j):
                    d.update({j: d.get(j) + 1})
                else:
                    alphabet += j
                    d.update({j: 1})
    f.close()

    def dict_sort(d):
        sort_d = sorted(d.items(), key=lambda x: x[1])
        return dict(sort_d)

    d = dict_sort(d)

    f = open('res5.bin', 'wb')
    f.write((alphabet + '\n').encode())
    alphabet = sorted(alphabet)

    size = 0
    base = 1
    while base < len(alphabet):
        base *= 2
        size += 1

    def code_word(alphabet, size, i):
        code = alphabet.index(i)
        code = format(code, 'b')
        while len(code) < size:
            code = '0' + code
        return code

    code = ''
    for i in text:
        if i == ' ':
```

```
            code += code_word(alphabet, size, '_')
            j = alphabet.index('_')
            alphabet.pop(j)
            alphabet = ['_'] + alphabet
        elif i == '\n':
            code += code_word(alphabet, size, '*')
            j = alphabet.index('*')
            alphabet.pop(j)
            alphabet = ['*'] + alphabet
        else:
            code += code_word(alphabet, size, i)
            j = alphabet.index(i)
            alphabet.pop(j)
            alphabet = [i] + alphabet

    extra_zero = 0 if len(code) % 8 == 0 else 8 - len(code) % 8
    f.write((str(extra_zero) + '\n').encode())
    bts = '0' * extra_zero + code
    to_write = bytearray()
    for i in range(0, len(bts), 8):
        to_write.append(int(bts[i: i+8], 2))

    f.write(to_write)
    f.close()
    print(f'{time.perf_counter() - start} seconds')

else:
    start = time.perf_counter()
    f = open("res5.bin", "rb")

    d = {}
    tree = {}
    alphabet = {}
    text = ''
    ans = ''

    str_keys = f.readline().decode()[:-2].split()
    b = True
    for i in str_keys:
        if b:
            tmp = i
            b = False
        else:
            alphabet.update({i: tmp})
            b = True

    count_of_zero = int(f.readline().decode())
    dump = f.read()
    bitstr = ''
    for b in dump:
        bits = bin(b)[2:].rjust(8, '0')
        bitstr += bits

    text = bitstr[count_of_zero:]
    f.close()

    def true_word(pos, word, text):
        if (len(word) <= len(text) - pos):
            b = True
            for i in range(len(word)):
                if word[i] == text[pos + i]:
                    b = b and True
```

```
                else:
                    b = b and False
            else:
                b = False
            return b

        i = 0
        while i < len(text):
            for j in list(alphabet.keys()):
                if true_word(i, j, text):
                    ans += alphabet[j]
                    i += len(j)
                    break

        ans = ans.replace('_', ' ')
        ans = ans.replace('*', '\n')

        f = open("restore5.txt", "w")
        f.write(ans)
        f.close()
        print(f'{time.perf_counter() - start} seconds')




f = open("res5.bin", "rb")

alph_list = []
text = ''
ans = ''

alphabet = f.readline().decode()[:-1]
alphabet = sorted(alphabet)

count_of_zero = int(f.readline().decode())
dump = f.read()
bitstr = ''
for b in dump:
    bits = bin(b)[2:].rjust(8, '0')
    bitstr += bits

text = bitstr[count_of_zero:]
f.close()

size = 0
base = 1
while base < len(alphabet):
    base *= 2
    size += 1

pos = 0
while pos < len(text):
    tmp = ''
    for i in range(size):
        tmp += text[pos + i]
    alph_list += [tmp]
    pos += size



for i in alph_list:
```

```
    j = int(i, 2)
    ans += alphabet[j]
    tmp = alphabet.pop(j)
    alphabet = [tmp] + alphabet



ans = ans.replace('_', ' ')
ans = ans.replace('*', '\n')

f = open("restore5.txt", "w")
f.write(ans)
f.close()
```