МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования  
«Белорусский государственный технологический университет»  
  
  
  
  
  
  
Отчёт по лабораторной работе №8

СЖАТИЕ/РАСПАКОВКА ДАННЫХ МЕТОДОМ БАРРОУЗА − УИЛЕРА

Выполнил: студент 3 курса специальности ИСиТ Калоша И.В.  
Проверила: Ржеутская Н. В.

Минск 2020

**Сжатие/распаковка данных методом Барроуза − Уилера**

**Цель:** приобретение практических навыков использования метода Барроуза − Уилера для сжатия/распаковки данных.

**Теоретические сведения**

Основная цель сжатия – обеспечить более компактное представление данных, вырабатываемых источником, т. е. уменьшить физический объем сообщений, генерируемых источником, и сократить время его передачи (читай – стоимость) по каналам связи. Фундаментальная теорема К. Шеннона о кодировании информации утверждает, что «стоимость кодирования всегда не меньше энтропии источника, хотя может быть сколь угодно близка к ней» [26−27]. Поэтому для любого алгоритма сжатия всегдаимеется некоторый предел степени (или эффективности) сжатия, определяемый энтропией входного потока (или сжимаемого сообщения).

Все алгоритмы сжатия преобразуют входной поток данных, минимальной единицей которых является бит, а максимальной – байт или несколько байт. Основными техническими характеристиками процессов сжатия и результатов их работы являются:

• степень сжатия (англ. compress rating), или отношение R (англ. ratio) объемов исходного (до сжатия, Vдс) и результирующего (после сжатия, Vпс) потоков данных (сообщений);

• скорость сжатия − время, затрачиваемое на сжатие некоторого объема информации входного потока до получения из него

эквивалентного выходного потока;

• качество сжатия − величина, показывающая, насколько сильно сжат выходной поток при помощи применения к нему повторного сжатия по этому же или иному алгоритму.

Существуют различные подходы к реализации сжатия информации. Они отличаются математической базой, уровнем сложности (простоты) практической реализации, форматом кодируемого потока данных, степенью соответствия сжимаемых и распакованных данных.

По критерию, связанному с характером или форматом данных или степенью соответствия сжимаемых данных распакованным, все методы сжатия разделяют на два класса: обратимое и необратимое сжатие, или иначе: сжатие без потерь и сжатие с частичной потерей информации (англ. lossy compression).

Понятно, что недопустимы никакие потери при упаковке текстовых документов, кодов компьютерных программ, файлов баз данных.

Сжатие с потерей информации реализуется на основе таких известные форматов данных и алгоритмов сжатия, как JPEG и MPEG. Алгоритм JPEG используется при сжатии фотоизображений. Алгоритмы MPEG используют при сжатии видео и музыки.

**Листинг кода**

def Burrows\_Wheeler\_Transform(string):

time1 = time.time()

###VAR###

string\_index = 0;

str\_list = []

###SOLUTION###

print()

print('---------прямое преобразование---------')

print()

print('Исходное сообщение : ',string)

for i in range(len(string)):

str\_list.append(Shift(string,i))

#print(str\_list)

str\_list.sort()

for i in range(len(str\_list)):

if str\_list[i].\_\_eq\_\_(string):

string\_index = i

arr = [list(i) for i in str\_list]

arr = np.array(arr)

res = Get\_Last\_Col(str\_list)

print(arr)

print('Номер нашего сообщения : ',string\_index+1)

print('Выходное сообщение : ',res)

A=[string\_index,res]

time2 = time.time()

print('Время выполнения прямого прербразования :', (time2 - time1)\*1000)

return A

def Burrows\_Wheeler\_Transform\_Back(string, position):

time1 = time.time()

###VAR###

string\_index = 0;

str\_list = []

###SOLUTION###

print()

print('---------обратное преобразование---------')

print()

for i in range(len(string)):

str\_list.append(string[i])

str\_list.sort()

for i in range(len(string)-1):

for j in range(len(string)):

str\_list[j]=string[j]+str\_list[j]

str\_list.sort()

#print(str\_list)

arr = [list(i) for i in str\_list]

arr = np.array(arr)

print(arr)

result = str\_list[position]

print('Раскодированная строка :', result)

time2 = time.time()

print('Время выполнения прямого прербразования :', (time2 - time1)\*1000)

def Shift(seq, shift=1):

return seq[-shift:] + seq[:-shift]

def Get\_Last\_Col(Arr):

res =""

for i in Arr:

res += i[-1]

return res

**Ход работы**

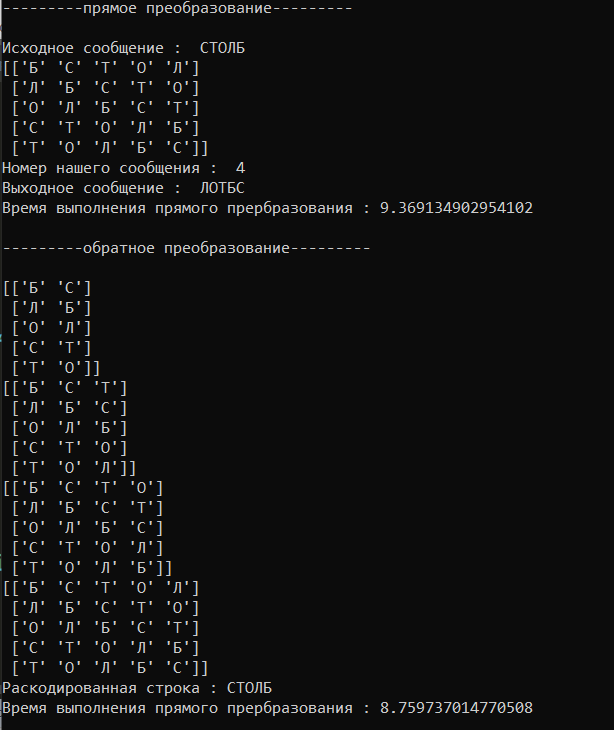
****

Рисунок 1 – Вывод консоли

**Вывод**: в данной работе был рассмотрен способ сжатия данных метод Барроуза – Уиллера. Данный метод позволяет представить данные в более удобном для сжатия формате. Особенностью алгоритма является то, что он основан на рекурсивных операциях сжатия и распаковки.