МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность Информационные системы и технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9 НА ТЕМУ:**

**Сжатие/распаковка данных на основе статистических методов**

Ф.И.О.

Божко Денис Владимирович

Преподаватель

асс. Берников Владислав Олегович

Минск 2021

**Цель:** приобретение практических навыков использования статистических методов Шеннона − Фано и Хаффмана (ShannonFano and Huffman coding) для сжатия/распаковки данных.

**Теоретические сведения**

До появления уже упоминавшихся работ К. Шеннона кодирование символов алфавита при передаче сообщения по каналам связи осуществлялось одинаковым количеством битов, получаемым по формуле Хартли (см. формулу (2.2)). Позднее начали появляться способы, кодирующие символы разным числом битов в зависимости от вероятности появления их в тексте, подтверждение чему мы получили при выполнении лабораторной работы № 2. Таким образом, за счет использования для каждого значения байта кодов ASCII (символа алфавита) кода различной длины в соответствии с частостью (вероятностью появления этого символа в сообщении) можно значительно уменьшить общий размер данных.

Эта базовая идея лежит в основе алгоритмов статистических (вероятностных) методов сжатия: Шеннона − Фано и Хаффмана.

Статистические алгоритмы позволяют создавать более короткие коды для часто встречающихся и более длинные – для редко встречающихся символов алфавита или конкретного сообщения. В первом случае метод считается статическим статистическим, во втором – динамическим статистическим: вероятностные свойства символов подсчитываются для конкретного сообщения или потока данных.

Частота или вероятность появления того или иного символа алфавита в произвольном сообщении, лежащая в основе алгоритмов, дали название этим алгоритмам и соответствующим методам.

Иногда эти методы называют также префиксными.

Таким образом, использование описываемых методов предусматривает создание кодовой таблицы (подобно кодам ASCII или base64). Формально процедура сжатия (прямое преобразование) состоит в подстановке соответствующего бинарного кода вместо символа исходного алфавита и наоборот – при обратном.

Методы относятся к классу «сжатие без потерь». Различие между двумя рассматриваемыми методами состоит лишь в особенностях формирования таблицы бинарных кодов. При формировании этой таблицы для обоих методов можно воспользоваться статистическими свойствами алфавитов, полученными при выполнении лабораторной работы № 2. преобразовании.

**Листинг кода**

|  |
| --- |
| from collections import Counter  def divide\_table(table):  optimal\_difference = sum(table.values())  optimal\_index = 0  for i in range(len(table)):  current\_difference = abs(sum(list(table.values())[:i]) - sum(list(table.values())[i:]))  if current\_difference < optimal\_difference:  optimal\_difference = current\_difference  optimal\_index = i  return dict({item for i, item in enumerate(table.items()) if i < optimal\_index}), \  dict({item for i, item in enumerate(table.items()) if i >= optimal\_index})  def shenon\_get\_codes(table, value='', codes={}):  if len(table) != 1:  a, b = divide\_table(table)  shenon\_get\_codes(a, value + '0', codes)  shenon\_get\_codes(b, value + '1', codes)  else:  codes[table.popitem()[0]] = value  return codes  def decode\_symbol(table, code, index=0):  if len(table) != 1:  a, b = divide\_table(table)  if code[index] == '0':  return decode\_symbol(a, code, index + 1)  else:  return decode\_symbol(b, code, index + 1)  else:  return table.popitem()[0]  data = input('Шифруемая строка: ')  counter = Counter(data)  sorted\_freq = sorted(set(data), key=lambda letter: counter[letter], reverse=True)  sorted\_freq\_dict = {letter: counter[letter] for letter in sorted\_freq}  code\_table = shenon\_get\_codes(sorted\_freq\_dict)  print(sorted\_freq\_dict)  for symbol, key in sorted(code\_table.items(), key=lambda item: len(item[1])):  print(symbol, key, sep=': ')  encoded = [code\_table[letter] for letter in data]  encoded\_bits = ''.join(encoded)  encoded\_str = [chr(int(encoded\_bits[i:i + 8], 2)) for i in range(0, len(encoded\_bits), 8)]  print('Исходная строка ({} bits): '.format(len(data) \* 8), data)  print('Сжатая строка ({} bits): '.format(len(encoded\_str) \* 8), ''.join(encoded\_bits))  print('Данные: {}'.format(encoded\_bits))  index = 0  decoded\_str = ''  while index < len(encoded\_bits):  current = decode\_symbol(sorted\_freq\_dict, encoded\_bits, index)  decoded\_str += current  index += len(code\_table[current])  print('Расшифрованная строка: ', decoded\_str) |

**Вывод**: в данной работе был рассмотрен способ сжатия данных на основе статистических данных – метод Шеннона – Фано. Метод не является оптимальным так как при равном распределении вероятностей он может выдавать несколько различных вариантов результатов. Однако метод дает оптимальное распределение вероятностей и максимальное сжатие.