УДК 004

Сжатие данных с применением алгоритмов LZ77 на языке программирования python

Меркулов Андрей Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью данной работы является написания код сжатия текста, используя алгоритм LZ77. Для выполнения этой задачи был выбран язык программирования python**.** Практическая значимость работы заключается в создании скрипта способного сжать текст.
Ключевые слова: информационные технологии, сжатие данных, текст, строки.

Data compression using LZ77 algorithms in python programming language

Merkulov Andrey Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of this work is to write a text compression code using the LZ77 algorithm. The python programming language was chosen to perform this task. The practical significance of the work is to create a script capable of compressing text.
Keywords: information technology, data compression, text, strings.

1 Введение

1.1 Актуальность

В основе любого способа сжатия лежит модель источника данных, или, точнее, модель избыточности. Иными словами, для сжатия данных используются некоторые априорные сведения о том, какого рода данные сжимаются. Не обладая такими сведениями об источнике, невозможно сделать никаких предположений о преобразовании, которое позволило бы уменьшить объём сообщения. Методы, позволяющие на основе входных данных изменять модель избыточности информации, называются адаптивными. Неадаптивными являются обычно узкоспециализированные алгоритмы, применяемые для работы с данными, обладающими хорошо определёнными и неизменными характеристиками.

1.2 Обзор исследований

В.Ю. Мокрый в статье приводит описание последовательности обучения сжатию графической информации на примере алгоритма JPEG [1].

В.Г. Шишмарова рассмотрел проблемы лаконичности речи и способы сжатия текста. [2]. В.В. Исаченко описал новую модель сжатия данных. [3]. Н.Г.Шандрюк в работе представил аппаратную реализацию потокового сжатия данных на ZYNQ xc7z020clg484-1 [4]. О.О. Серый В данной статье представил метод определения характеристик текстов и их классификации с помощью архивирования. [5].

1.3 Цель исследования

Цель исследования - написать код сжатия текста используя алгоритм LZ77.

2 Материалы и методы

Для разработки алгоритма был использован язык программирования Python и среда разработки Google Colab.

3 Результаты исследования

В данной работе показана реализация алгоритма LZ77.

3.1 Добавления текста для сжатия

В среде программирования Google Colab создаётся переменная с текстом, который в дальнейшем будет сжат (рис. 1).

```
src = 'учиться учиться и еще раз учиться'
```

Рисунок 1. Текст для сжатия

Выводим веденый текст что бы после разархивации сравнить с оригиналом (рис. 2).

```
print('src: ' + src)
pack = ''
```

Рисунок 2. Вывод текста

3.2 Сжатие текста

Цикл проходит по тексту и ищет одинаковые слова размерам больше трех символов. Найдя, заменяется их * и приписывает позицию слова и его длину. Затем выводит сжатое предложения (рис.3).

```
i = 0
while i < len(src):
  ln = 7
  found = False
  while not found and ln >3:
    j =i - ln
    while j >= 0 and (i-j)<100:
      if src[i:i+ln] == src[j:j+ln]:
        pack +='*%1d%2d' % (ln,(i-j))
        i += ln
        found = True
        break
      j -=1
    ln -=1
  if not found:
     pack +=src[i]
     i += 1

print('pack: ' + pack)
```

Рисунок 3. Сжатия текста

3.4 Распаковка текста

Создаться переменная для вывода распакованного текста. Затем цикл проходит по тексту и, встретив, * извлекает позицию и длину слова копирует сжатые слова. После завершения цикла выводит получившийся текст (рис. 4).

```
uopack = ''
i = 0
while i < len(pack):
  if pack[i] !='*':
    uopack +=pack[i]
    i += 1
    continue
  ln = int(pack[i+1: i+2])
  dist = int(pack[i+2: i+4])
  uopack += uopack[-dist: -dist+ln]
  i += 4

print('uopack: ' + uopack)
if src != uopack:
  print('!!!!!')
print(len(src), len(pack))
```

Рисунок 4. Распаковка текста

Для проверки правильности выполнения программы выводим текст об ошибки, если разархивированный текст не совпадает с изначальным. После, чтобы проверить, насколько код полезный, выводим количество символов до сжатия и после него (рис.5).

```
print('uopack: ' + uopack)
if src != uopack:
  print('ошибка')
print(len(src), len(pack))
```

Рисунок 5. Вывод ошибки

3.5 Работа программы

Завершив выполнения программы, получаем три строчки кода. В строке sec находиться изначально в виденный текст, в строке pack находиться сжатый текст, в uopack храниться распакованный текст (рис.6).

```
src: учиться учиться и еще раз учиться
pack: учиться *7 8 и еще раз*718я
uopack: учиться учиться и еще раз учиться
33 27
```

Рисунок 6. Выполнения программы

Исходя из последней строки на рисунке 6, можно сделать выводы, что программа сжимает файлы очень плохо, и уже сжатый файл требуется сжимать повторно с использованием другого алгоритма.

4 Выводы

В ходе выполнения работ была реализован алгоритм LZ77, который сжимает текст. Был проведен краткий анализ, который вывел недостатки алгоритма.

Библиографический список

1. Шишмарова В.Г. Способы информационного сжатия текста без потери информации: замещение, опущение, совмещение//Сборник статей Международного научно-исследовательского конкурса. 2019. С. 91-102.
2. Исаченко В.В. Сжатие данных на основе сборки слов// сборник материалов XII Международной школы-конференции студентов, аспирантов и молодых ученых. 2016. С. 332-336.
3. Шандрюк Н.Г. Реализация на плис алгоритма сжатия данных в потоке// Фундаментальные проблемы радиоэлектронного приборостроения. 2018. Т. 18. № 4. С. 1007-1010.
4. Серый О.О. Метод кластеризации сообщений с помощью Архивирующего преобразования//ScienceRise. 2015. Т. 6. № 2 (11). С. 76-79.
5. Мокрый В.Ю. Последовательность обучения алгоритмам сжатия графической информации на примере алгоритма JPEG // Вестник Волжского университета им. В.Н. Татищева. 2011. № 18. С. 135-139.