УДК 004

Применение алгоритма преобразования Барроуза-Уилера (BWT)

Романов Даниил Алексеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью данного исследования является изучение алгоритма преобразования Барроуза-Уилера и его применения в сжатии данных. Методы исследования включают в себя реализацию алгоритма на языке программирования Python в среде Google Colab и анализ полученных результатов. В результате исследования было выяснено, что алгоритм преобразования Барроуза-Уилера позволяет подготовить данные для дальнейшего сжатия без потери информации.
Ключевые слова: Google Colab, Python, алгоритм преобразования Барроуза-Уилера (BWT), подготовка к сжатию данных

Application of the Burrows-Wheeler Transformation Algorithm (BWT)

Romanov Daniil Alekseevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of this study is to study the Burrows-Wheeler transformation algorithm and its application in data compression. The research methods include the implementation of the algorithm in the Python programming language in the Google Colab environment and the analysis of the results obtained. As a result of the study, it was found out that the Burrows-Wheeler transformation algorithm allows you to prepare data for further compression without losing information.
Keywords: Google Colab, Python, Burrows-Wheeler transformation algorithm (BWT), data compression preparation

1 Введение

1.1 Актуальность

Сжатие данных - это важный аспект хранения и передачи информации, особенно в условиях ограниченной пропускной способности сети или малого объема хранилища. Алгоритм BWT позволяет упорядочить данные таким образом, что повторяющиеся последовательности данных группируются вместе, что облегчает их сжатие. Это делает алгоритм BWT важным инструментом для обработки данных во многих областях, включая компьютерную науку, биологию, генетику и другие.

1.2 Обзор исследований

М.П. Бакулина посвятила свою статью применению преобразования Барроуза-Уилера в задачах сжатия данных. В статье описываются основные принципы этого метода и предлагается алгоритм, позволяющий достичь высокой степени сжатия данных. Кроме того, авторы проводят сравнение эффективности применения преобразования Барроуза-Уилера с другими методами сжатия данных [1].

А.А. Прокопович рассматривает различные методы сжатия данных без потерь, такие как алгоритм Хаффмана, алгоритм Лемпеля-Зива и их модификации. В статье также описываются особенности каждого метода и сравниваются их эффективность и скорость работы. В заключении автор делает вывод о том, что выбор оптимального метода сжатия данных зависит от конкретной задачи и типа данных [2].

Д. Ватолин, А. Ратушняк, М. Смирнов и В. Юкин описали в своей книге различные методы сжатия данных, включая алгоритм Хаффмана, алгоритм Лемпеля-Зива, алгоритм Шеннона-Фано и другие. Авторы также рассматривают устройство архиваторов, алгоритмы сжатия изображений и видео, их принципы работы и особенности применения. Книга содержит множество примеров и иллюстраций, которые помогают понять принципы работы методов сжатия данных [3].

1.3 Цель исследования

Целью данного исследования является изучение и применение алгоритма преобразования Барроуза-Уилера (BWT) для подготовки данных к сжатию.

2 Материалы и методы

Для работы понадобится онлайн среда программирования Google Colab [4].

3 Результаты и обсуждение

Алгоритм преобразования Барроуза-Уилера (BWT) является одним из алгоритмов сжатия данных, который используется для уменьшения размера текстовых файлов и других данных. Он работает путем перестановки символов в исходном тексте таким образом, чтобы повторяющиеся последовательности данных группировались вместе, что облегчает их сжатие. Преобразование Барроуза-Уилера не уменьшает количество информации в исходном тексте, но делает его более сжимаемым, так как группировка повторяющихся последовательностей данных позволяет более эффективно сжимать данные с помощью других алгоритмов сжатия, таких как алгоритм Хаффмана или арифметическое кодирование.

Принцип работы алгоритма BWT заключается в следующих шагах:

1. Исходный текст разбивается на все возможные перестановки символов.

2. Полученные перестановки сортируются в лексикографическом порядке.

3. Из полученной матрицы символов выбирается последний столбец и объединяется в одну строку.

4. Полученная строка и индекс позиции первого символа исходного текста в отсортированной матрице символов составляют пару, которая является выходными данными алгоритма.

Пример:

Допустим, у нас есть исходный текст "banana". Мы можем создать все возможные перестановки символов и получить следующую матрицу (рис.1).

Рисунок 1 - Матрица перестановок

Сортируем эту матрицу и получаем (рис.2).

Рисунок 2 - Сортировка матрицы в лексикографическом порядке

Затем берем последний столбец и объединяем его в одну строку, получая (рис.3).

Рисунок 3 - Преобразованный текст

Таким образом получаем преобразованный текст "annb$aa", который можно далее сжать с помощью других алгоритмов сжатия данных.

Приступаем к написанию кода, который реализует алгоритм преобразования Барроуза-Уилера и позволяет преобразовать исходный текст в другой текст, который может быть более эффективно сжат без потери информации.

Функция create_matrix() создает матрицу, где каждая строка является сдвигом исходного текста на i символов. То есть первая строка матрицы - это исходный текст, вторая строка - исходный текст, сдвинутый на один символ вправо, и так далее (рис.4).

```
# Функция для преобразования текста в матрицу
def create_matrix(text):
    n = len(text)
    matrix = [text[i:n] + text[0:i] for i in range(n)]
    return matrix
```

Рисунок 4 - Функция для преобразования текста в матрицу

Функция sort_matrix() сортирует строки матрицы в лексикографическом порядке (рис.5).

```
# Функция для сортировки матрицы
def sort_matrix(matrix):
    sorted_matrix = sorted(matrix)
    return sorted_matrix
```

Рисунок 5 - Функция для сортировки матрицы

Функция get_last_column() возвращает последний столбец отсортированной матрицы. Этот столбец является преобразованным текстом (рис.6).

```
# Функция для получения последнего столбца матрицы
def get_last_column(matrix):
    last_column = [row[-1] for row in matrix]
    return last_column
```

Рисунок 6 - Функция для получения последнего столбца матрицы

Чтобы восстановить исходный текст из преобразованного текста, нужно знать индекс первой строки в отсортированной матрице. Этот индекс говорит какая строка исходного текста становится первой в отсортированной матрице. Функция get_first_row_index() возвращает индекс первой строки отсортированной матрицы, которая соответствует исходному тексту (рис.7).

```
# Функция для получения индекса первой строки в отсортированной матрице
def get_first_row_index(matrix, text):
    n = len(text)
    for i in range(n):
        if matrix[i] == text:
            return i
```

Рисунок 7 - Функция для получения индекса первой строки в отсортированной матрице

Функция bwt() реализует алгоритм Барроуза-Уилера. Она использует описанные выше функции для создания матрицы, ее сортировки, получения последнего столбца и индекса первой строки отсортированной матрицы. Затем она объединяет последний столбец в строку и возвращает преобразованный текст и индекс первой строки отсортированной матрицы (рис.8).

```
# Функция для преобразования текста с помощью алгоритма Барроуза-Уилера
def bwt(text):
    matrix = create_matrix(text)
    sorted_matrix = sort_matrix(matrix)
    last_column = get_last_column(sorted_matrix)
    first_row_index = get_first_row_index(sorted_matrix, text)
    bwt_text = ''.join(last_column)
    return bwt_text, first_row_index
```

Рисунок 8 - Функция для преобразования текста с помощью алгоритма Барроуза-Уилера

Выполним преобразование текста на примере "mississippi river" (рис.9).

```
# Пример использования функции
text = "mississippi river"
bwt_text, first_row_index = bwt(text)
print("Исходный текст: ", text)
print("Преобразованный текст: ", bwt_text)
print("Индекс первой строки в отсортированной матрице: ", first_row_index)
```

Рисунок 9 - Пример использования BWT

Результатом выполнения основной функции будет следующий результат (рис.10).

```
Исходный текст:  mississippi river
Преобразованный текст:  ivpssmrrpi essiii
Индекс первой строки в отсортированной матрице:  7
```

Рисунок 10 - Результаты выполнения программы

Выводы

В результате выполнения алгоритма Барроуза-Уилера получается преобразованный текст, который может быть далее сжат с помощью других алгоритмов сжатия данных. Это позволяет уменьшить размер исходного текста и сэкономить место при его хранении или передаче по сети.

Алгоритм Барроуза-Уилера также предоставляет индекс первой строки в отсортированной матрице, который является необходимой частью выходных данных и позволяет восстановить исходный текст из преобразованного текста.

Также алгоритм Барроуза-Уилера может быть использован для поиска повторяющихся фрагментов в тексте, что может быть полезно в таких областях, как сжатие данных, обработка геномных данных и анализ текстовых данных.

Использование алгоритма преобразования Барроуза-Уилера (BWT) позволяет улучшить эффективность сжатия данных. Это делает алгоритм BWT важным инструментом для обработки данных во многих областях, включая компьютерную науку, биологию, генетику и другие. Программа на Python, реализующая алгоритм BWT, позволяет легко применять этот алгоритм к любым текстовым данным.

Ознакомиться с кодом программы и проверить его работоспособность можно по данной ссылке[6].

Библиографический список

1. Бакулина М.П. Эффективное сжатие данных на основе преобразования Барроуза-Уилера // Ползуновский вестник. 2012. №2072-8921. С. 112-114.
2. Прокопович А.А. Алгоритмы сжатия данных без потерь // Высокие технологии, наука и образование: актуальные вопросы, достижения и инновации. - Пенза: МЦНС «Наука и Просвещение», 2018. С. 59-61.
3. Методы сжатия данных: Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. М.: ДИАЛОГ - МИФИ, 2003. 384 с.
4. Google Colab URL: https://colab.research.google.com
5. Код программы URL: https://colab.research.google.com/drive/1CXjmFUGsT3WqcRGj9pU72_xUakYb9EcS?usp=sharing