УДК 004

Реализация алгоритма Хаффмана на языке программирования Python

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В этой статье продемонстрирован процесс реализации алгоритма Хаффмана на языке программирования Python.
Ключевые слова: Хаффман, Python, сжатие данных

Implementation of the Huffman algorithm in the Python programming language

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article demonstrates the process of implementing the Huffman algorithm in the Python programming language.
Keywords: Huffman, Python, data compression

В современном мире каждый год увеличивается количество информации, передаваемой по сети интернет. И проблема сжатия данных каждый год становится более острой.

Одним из множества решений этой проблемы это применение алгоритма Хаффмана. Основной принцип алгоритма Хаффмана это сортировка наиболее часто встречаемых элементов, сортировка их по убыванию и кодирование новыми символами меньшими по длине. За счет этого удается сжать данные примерно на 50-60%.

Задача состоит в создании автоматического алгоритма по сжатию данных методом Хаффмана на языке программирования Python.

Ранее этим вопросом интересовались М. О. Смирнова, А. П. Смирнов с темой «Программный продукт для демонстрации применения оптимального кодирования на примере алгоритмов Шеннона-Фано и Хаффмана» [1], а подробнее про программный продукт, с помощью которого можно продемонстрировать применение оптимальных кодов. К.А. Шмалева развивала тему «Код Хаффмана» [2] в которой рассказывается метод кодирования информации с помощью метода Дэвида Хаффмана, на примере передачи цифрового сообщения. В.Э.Родионов, Т.В. Чекулаева опубликовали статью «Сжатие данных с использованием кодирования Хаффмана» [3] рассказали про применение алгоритма Хаффмана.

Программа будет представлена по частям, для лучшего понимания.

```
import heapq
from collections import Counter
from collections import namedtuple
```

Рис. 1

На рисунке 1 импортированы стандартные модули для работы с «минимальной кучей» и словарь с поддержкой счетчика.

```
class Node(namedtuple("Node", ["left", "right"])):
    def walk(self, code, acc):

        self.left.walk(code, acc + "0")
        self.right.walk(code, acc + "1")
```

Рис. 2

На рисунке 2 объявление класса для хранения информации о структуре дерева.

```
class Leaf(namedtuple("Leaf", ["char"])):
    def walk(self, code, acc):

        code[self.char] = acc or "0"
```

Рис. 3

На рисунке 3 объявление класса для «листьев дерева», у них нет потомков, но есть значение символа.

```
def huffman_encode(s):
    h = []
    for ch, freq in Counter(s).items():
        h.append((freq, len(h), Leaf(ch)))
    heapq.heapify(h)
    count = len(h)
    while len(h) > 1:
        freq1, _count1, left = heapq.heappop(h)
        freq2, _count2, right = heapq.heappop(h)

        heapq.heappush(h, (freq1 + freq2, count, Node(left, right)))

        count += 1
    code = {}
    if h:
        [(_freq, _count, root)] = h
        root.walk(code, "")
    return code
```

Рис. 4

На рисунке 4 представлена главная логика программы, функция кодирования символов в коды Хаффмана.

```
def huffman_decode(encoded, code):
    sx =[]
    enc_ch = ""
    for ch in encoded:
        enc_ch += ch
        for dec_ch in code:
            if code.get(dec_ch) == enc_ch:
                sx.append(dec_ch)
                enc_ch = ""
                break
    return "".join(sx)
```

Рис. 5

На рисунке 5 представлена функция декодирования исходной строки по кодам Хаффмана.

```
def main():
    s = input()
    code = huffman_encode(s)
    encoded = "".join(code[ch] for ch in s)

    print(len(code), len(encoded))
    for ch in sorted(code):
        print("{}: {}".format(ch, code[ch]))
    print(encoded)
    print(huffman_decode(encoded, code))
```

Рис.6

На рисунке 6 представлена главная функция программы, она считывает текст пользователя, кодирует его, выводит частоту встречаемости каждого символа и его код Хаффмана. Потом выводит весь текст в виде кода Хаффмана и декодирует его обратно в читаемый тест. Это можно наблюдать на рисунке 7.

```
Приамурский государственный университет имени Шолом-Алейхема
23 258
 : 1001
-: 110110
А: 111001
П: 01100
Ш: 110100
а: 0101
в: 11000
г: 110101
д: 110111
е: 000
и: 1111
й: 0011
к: 110011
л: 01101
м: 1010
н: 1000
о: 11101
р: 1011
с: 0111
т: 0100
у: 0010
х: 110010
ы: 111000
01100101111111010110100010101101111100111111001110011101011110
Приамурский государственный университет имени Шолом-Алейхема
```

Рис. 7

Поставленная цель достигнута, программа работает правильно и её можно использовать в реальных условиях уже сейчас. На данный момент это не единственный вариант сжатия данных, но зато он является классическим для информатики.

Библиографический список

1. Смирнова М. О., Смирнов А. П. программный продукт для демонстрации применения оптимального кодирования на примере алгоритмов Шеннона-Фано и Хаффмана // Прикаспийский журнал: управление и высокие технологии. 2010. №2. С. 33-40.
2. Шмалева К.А. Код Хаффмана // Современные проблемы управления и регулирования: инновационные технологии и техника. 2016. С. 7-11.
3. Родионов В.Э., Чекулаева Т.В. Сжатие данных с использованием кодирования Хаффмена // Новая наука: теоретический и практический взгляд. 2016. №9. С. 141-143.