МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Алгоритмы и структуры данных»
Тема: Поиск образца в тексте: алгоритм Рабина-Карпа. Построение выпуклой оболочки: алгоритм Грэхема

Студент гр. 3344		Тукалкин. В.А.
Преподаватель		Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Написать алгоритм Рабина-Карпа для поиска подстроки и алгоритм Грэхема для построения выпуклой оболочки.

Задание

Поиск образца в тексте. Алгоритм Рабина-Карпа.

Напишите программу, которая ищет все вхождения строки Pattern в строку Text, используя алгоритм Карпа-Рабина.

На вход программе подается подстрока Pattern и текст Text. Необходимо вывести индексы вхождений строки Pattern в строку Text в возрастающем порядке, используя индексацию с нуля.

Примечание: в работе запрещено использовать библиотечные реализации алгоритмов и структур.

Ограничения

$1 \leq |Pattern| \leq |Text| \leq 5 \cdot 105$.

Суммарная длина всех вхождений образца в текста не превосходит 108. Обе строки содержат только буквы латинского алфавита.

Пример.

Вход:

aba

abacaba

Выход:

0 4

Подсказки:

1. Будьте осторожны с операцией взятия подстроки — она может оказаться дорогой по времени и по памяти.

2. Храните степени x ** p в списке - тогда вам не придется вычислять их каждый раз заново.

Алгоритм Грэхема

Дано множество точек, в двумерном пространстве. Необходимо построить выпуклую оболочку по заданному набору точек, используя алгоритм Грэхема.

Также необходимо посчитать площадь получившегося многоугольника.

Выпуклая оболочка - это наименьший выпуклый многоугольник, содержащий заданный набор точек.

На вход программе подается следующее:

* первая строка содержит n - число точек

* следующие n строк содержат координаты этих точек через ', '

На выходе ожидается кортеж содержащий массив точек в порядке обхода алгоритма и площадь получившегося многоугольника.

Пример входных данных

6

3, 1

6, 8

1, 7

9, 3

9, 6

9, 0

Пример выходных данных

([[1, 7], [3, 1], [9, 0], [9, 3], [9, 6], [6, 8]], 47.5)

Также к очной защите необходимо подготовить визуализацию работы алгоритма, это можно сделать выводом в консоль или с помощью сторонних библиотек (например Graphviz).

Визуализацию загружать не нужно.

Выполнение работы

Функции, написанные в ходе работы:

1) calculatingArea(array) – на вход принимает двухмерный массив с координатами точек. Эта функция рассчитывает площадь многоугольника, заданного списком точек, с помощью формулы "шнуровки". Гаусса считает площадь многоугольника.

2) Graham(array) – на вход принимает двухмерный массив с координатами точек. Основная функция для построения выпуклой оболочки с использованием алгоритма Грэхема. Сначала функция находит точку с минимальной координатой x и сортирует остальные точки по углу наклона относительно этой точки, используя функцию cmp_to_key и rotate. Затем строится выпуклая оболочка, последовательно добавляя точки и исключая те, которые не удовлетворяют условию выпуклости. Результат — список точек, образующих выпуклую оболочку множества точек.

3) Rabin_Karp(pattern, text) – на вход принимает две строки. Функция реализует алгоритм Рабина-Карпа для поиска подстроки pattern в строке text. Она вычисляет хэш подстроки и сравнивает его с хэшами подстрок текста, чтобы найти совпадения. Если хэш совпадает и строки идентичны, добавляет индекс начала совпадения в результат.

4) visualization(array, GrahamArray) – визуализация при помощи библиотеки PIL, на вход принимает массив всех точек и массив после алгоритма Грэмема. Функция проходит по всем точкам в массиве и отображает их на изображении, затем изображает линии между парами точек.

Тестирование программы

Тесты для проверки корректности работы реализованных алгоритмов находятся в файле tests.py. На рисунке 1 показан пример вывода многоугольника и точек.

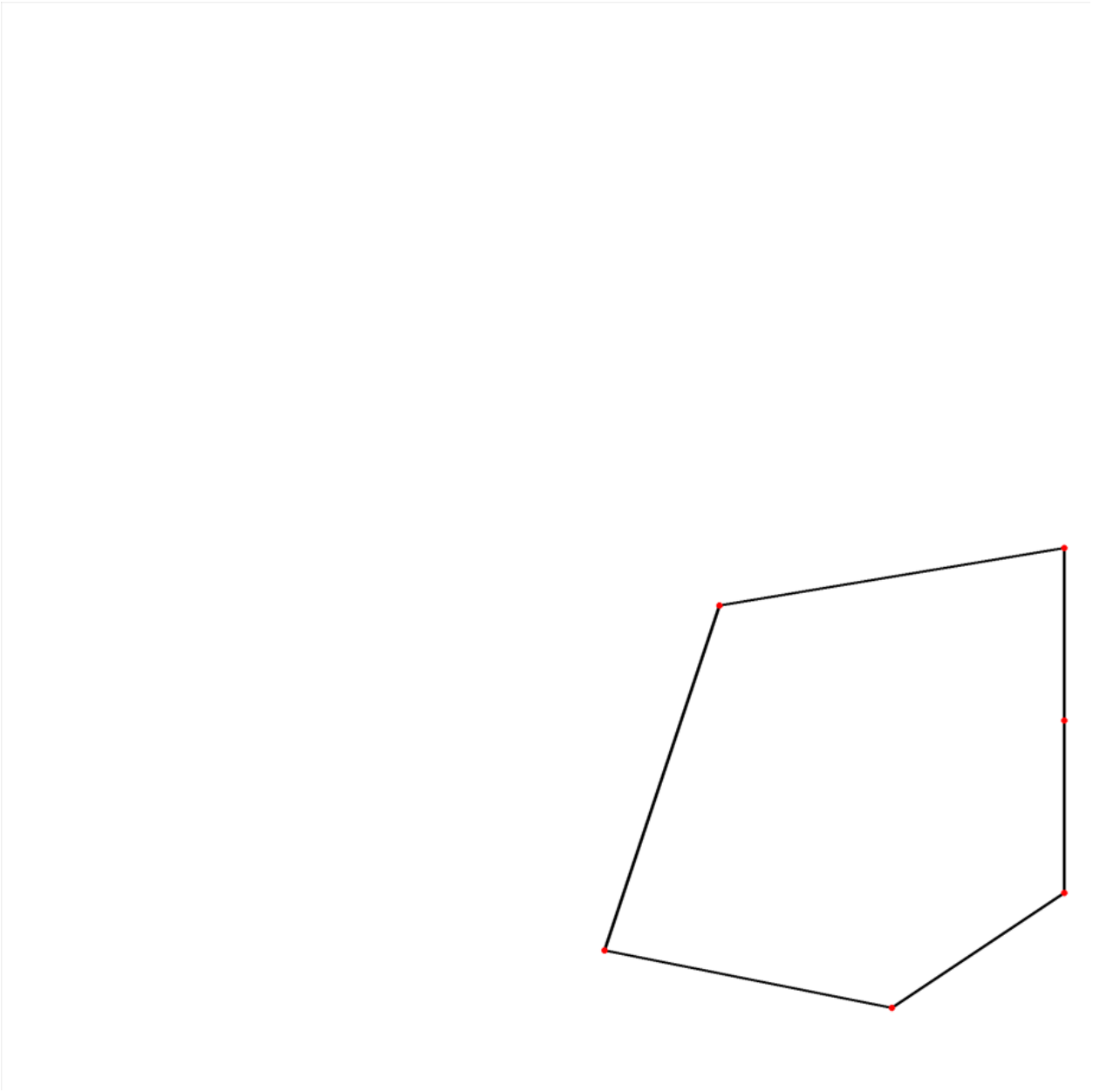

Рис.1 пример вывода многоугольника и точек

Результаты тестирования:

Тестирование и замеры времени для алгоритмов Рабина-Карпа с сгенерированными данными:

Длина текста: 85158823

Длина подстроки: 10

Время выполнения Rabin_Karp: 37.022730 секунд

Алгоритм Рабина-Карпа эффективно выполняет поиск подстроки в очень длинном тексте, демонстрируя ожидаемое время выполнения для случая с большим объёмом данных. Значительное время выполнения отражает временную сложность алгоритма, однако он успешно находит подстроку в длинном тексте.

Тестирование и замеры времени для алгоритма Грэхема с сгенерированными данными:

Количество точек: 212321

Время выполнения Graham: 2.152018 секунд

Площадь выпуклой оболочки: 3577405.0

Время выполнения calculatingArea: 0.024130 секунд

Алгоритм построения выпуклой оболочки Грэхема показывает хорошую производительность на большом количестве точек, а применение формулы «шнуровки» для расчёта площади выпуклой оболочки позволяет быстро получить результат. Время выполнения остаётся в пределах разумной сложности для больших наборов данных, демонстрируя эффективность алгоритмов для задач геометрической обработки.

Выводы

Были написаны алгоритмы Рабина-Карпа и Грэхема на языке Python и визуализация многоугольников по точкам. Алгоритм Рабина-Карпа имеет среднюю теоретическую сложность O(n+m), где n — длина текста, а m — длина подстроки. В тестировании он успешно обработал строку длиной 85158823 символов за 37.022730 секунд, демонстрируя высокую эффективность поиска подстроки благодаря полиномиальному хешированию. Алгоритм Грэхема для построения выпуклой оболочки имеет теоретическую сложность O(nlogn), что обусловлено сортировкой точек. Использованный в работе алгоритм справился с 212321 точкой за 2.152018 секунд, подтверждая свою скорость в обработке геометрических данных.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from modules.Rabin_Karp import Rabin_Karp
from modules.calculatingArea import calculatingArea
from modules.Graham import Graham
from modules.visualization import visualization


if __name__ == "__main__":
    option = int(input())
    if option:
        pattern, text = input(), input()
        print(*Rabin_Karp(pattern, text))
    else:
        arr = [[int(x) for x in input().split(', ')] for i in
range(int(input()))]
        answerArray = Graham(arr)
        area = calculatingArea(answerArray)
        print(answerArray, area)
        visualization(arr, answerArray)
```

Название файла: calculatingArea.py

```
def calculatingArea(array): return abs(sum([array[i][0] * array[(i
+ 1) % len(array)][1] - array[i][1] * array[(i + 1) % len(array)][0] for
i in range(len(array))])) / 2
```

Название файла: Rabin_Karp.py

```
def Rabin_Karp(pattern, text): return list(map(str, [i for i in
range(len(text) - len(pattern) + 1) if hash(text[i:i + len(pattern)])
== hash(pattern) and text[i:i + len(pattern)] == pattern]))
```

Название файла: visualization.py

```
from PIL import Image, ImageDraw


def visualization(array, GrahamArray):
    size = 0
    for i, j in array:
        size = max(size, i, j)
    size = size * 200 + 100

    img = Image.new('RGB', (size, size), 'white')
    draw = ImageDraw.Draw(img)
    prevX, prevY = GrahamArray[-1]
    for i, j in GrahamArray:
        draw.line((size // 2 + i * 100, size // 2 + j * 100, size
// 2 + prevX * 100, size // 2 + prevY * 100), fill='black', width=5)
        prevX, prevY = i, j
    for i, j in array:
        draw.ellipse((size // 2 + i * 100 - 5, size // 2 + j * 100
- 5, size // 2 + i * 100 + 5, size // 2 + j * 100 + 5), fill='red')

    img.show()
```

Название файла: Graham.py

```
def rotate(A,B,C):
    return (B[0]-A[0])*(C[1]-B[1])-(B[1]-A[1])*(C[0]-B[0])

def Graham(array):
    n = len(array)  # число точек
    P = list(range(n))
    for i in range(1, n):
        if array[P[i]][0] < array[P[0]][0]:  # если P[i]-ая точка
лежит левее P[0]-ой точки
            P[i], P[0] = P[0], P[i]  # меняем местами номера этих
точек
    for i in range(2, n):  # сортировка вставкой
        j = i
        while j > 1 and (rotate(array[P[0]], array[P[j - 1]],
array[P[j]]) < 0):
            P[j], P[j - 1] = P[j - 1], P[j]
            j -= 1
    S = [P[0], P[1]]  # создаем стек
    for i in range(2, n):
        while rotate(array[S[-2]], array[S[-1]], array[P[i]]) < 0:
            del S[-1]  # pop(S)
        S.append(P[i])  # push(S,P[i])

    arr=[]
    for i in range(len(S)):
        arr.append(array[S[i]])
    return arr
```

Название файла: tests.py

```
from modules.Rabin_Karp import Rabin_Karp
from modules.calculatingArea import calculatingArea
from modules.Graham import Graham


def test_Rabin_Karp():
    assert Rabin_Karp('ada', 'adacdada') == ['0', '5']


def test_calculatingArea():
    assert calculatingArea([[3, 1], [6, 8], [1, 7], [9, 3], [9,
6], [9, 0]]) == 13.0


def test_Graham():
    assert Graham([[3, 1], [6, 8], [1, 7], [9, 3], [9, 6], [9, 0]])
== [[1, 7], [3, 1], [9, 0], [9, 3], [9, 6], [6, 8]]


test_Rabin_Karp()
test_calculatingArea()
test_Graham()
```