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: Структуры. Хэш-таблицы.

Студент гр. 3342 _____ Песчатский С. Д.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2024

Цель работы

Целью данной работы является изучение принципов хэширования и его применения в алгоритмах, а также реализация двух алгоритмов с использованием полученных знаний:

1. **Алгоритм для нахождения всех вхождений подстроки в строку:**
    - Используя метод хэширования, разработать эффективный алгоритм, который будет находить все вхождения заданной подстроки в строку. Этот алгоритм должен быть более быстрым, чем простой поиск подстроки, и должен использовать хэш-функции для сравнения подстрок.
2. **Алгоритм для получения выпуклого многоугольника и нахождения его площади:**
    - Реализовать алгоритм, который будет строить выпуклую оболочку (выпуклый многоугольник) для заданного набора точек на плоскости. После построения выпуклой оболочки, алгоритм должен вычислять площадь этого многоугольника.

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 10^8. Обе строки содержат только буквы латинского алфавита.

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.

Выполнение работы

1) Поиск подстроки в строке. Алгоритм Рабина-Карпа.

Алгоритм Рабина-Карпа работает следующим образом:

1. Вычисляется хэш-значение для образца (подстроки) и для первой части текста, длина которой равна длине образца.
2. Затем вычисляется хэш-значение для следующей подстроки, сдвинутой на один символ. Для оптимизации, хэш-значение не вычисляется с нуля, а корректируется на основе предыдущего хэша и изменения символов.
3. При совпадении хэш-значений происходит сравнение самих строк. Если они совпадают, индекс начала подстроки записывается в массив.
4. Хэш-значения вычисляются по модулю 101, что может привести к коллизиям.

Функция Rabin_Karp(text, pattern, d=101, q=256) реализует алгоритм Рабина-Карпа для поиска всех вхождений подстроки в строку. Параметры:

- text — строка, в которой производится поиск.
- pattern — подстрока, которую нужно найти. Возвращаемое значение — список индексов начала вхождений подстроки в строку.
- d – модуль, для вычисления хэша
- q – количество возможных символов

2) Алгоритм Грэхема

Алгоритм Грэхема реализован в функции graham_scan. Сначала массив точек сортируется по координате x.

Затем оставшиеся точки сортируются по полярному углу относительно первой точки. Для этого используется функция rotate, основанная на векторном произведении. После сортировки точки добавляются в стек, и проверяется, чтобы полученная фигура была выпуклой.

Функция calculate_area вычисляет площадь фигуры по формуле, используя координаты вершин.

Функция visualize(points, convex_hull) визуализирует точки и выпуклую оболочку на графике. Параметры:

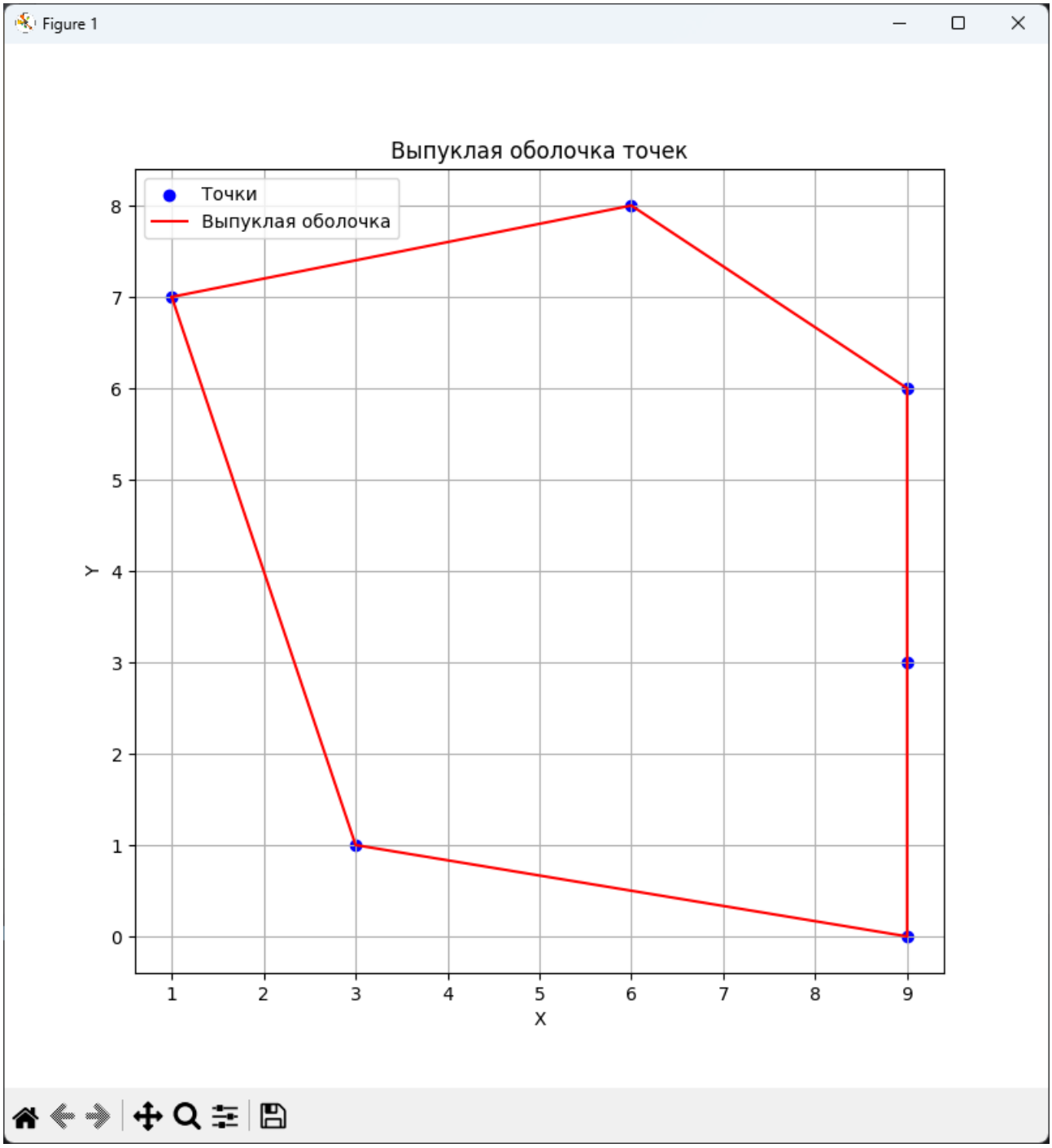
- points — список всех точек.
- convex_hull — список точек, образующих выпуклую оболочку.

Разработанный программный код см. в приложении А.

Тестирование

Нужно проверить работоспособность алгоритма. Тестировались различные функции через pytest(см. test.py в приложении).

Для анализа алгоритма Грэхема написана Функция visualize, для визуализации получившейся фигуры.



Выводы

Былы изучены принципы работы алгоритмов Рабина-Карпа и Грэхема. Их реализация была написана на языке программирования Python. Работоспособность была проверена через pytest на некоторых входных данных.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import matplotlib.pyplot as plt
def Rabin_Karp(text, pattern, d=101, q=256):
    """Реализует алгоритм поиска подстроки в строке с использованием метода Рабина-Карпа."""
    text_len = len(text)
    pat_len = len(pattern)

    pat_hash = ord(pattern[0])
    text_hash = ord(text[0])
    for i in range(pat_len-1):
        pat_hash = pat_hash*q+ord(pattern[i+1])
        text_hash = text_hash * q + ord(text[i + 1])
    pat_hash = pat_hash%d
    text_hash = text_hash%d

    result = []
    for i in range(text_len - pat_len + 1):
        if pat_hash == text_hash:
            if pattern == text[i:i+pat_len]:
                result.append(i)
        if i < text_len - pat_len:
            text_hash = ((text_hash - ord(text[i]) * q ** (pat_len-1)) * q + ord(text[i + pat_len])) % d
    return result

'''pattern = input()
text = input()
result = Rabin_Karp(text, pattern)
print(*result)'''

def calculate_area(polygon):
    s1, s2 = 0, 0
    for i in range(1, len(polygon)):
        s1 += polygon[i - 1][0] * polygon[i][1]
    s1 += polygon[-1][0] * polygon[0][1]
```

```
    for i in range(1, len(polygon)):
        s2 += polygon[i - 1][1] * polygon[i][0]
    s2 += polygon[-1][1] * polygon[0][0]
    return abs(s1 - s2) / 2
def rotate(p1, p2, p3):
    return (p2[0] - p1[0]) * (p3[1] - p1[1]) - (p2[1] - p1[1]) * (p3[0] -
p1[0]) >= 0
def graham_scan(points):
    n = len(points)
    indices = list(range(n))

    for i in range(1, n):
        if points[indices[i]][0] < points[indices[0]][0]:
            indices[i], indices[0] = indices[0], indices[i]

    for i in range(2, n):
        j = i
        while j > 1 and not rotate(points[indices[0]], points[indices[j -
1]], points[indices[j]]):
            indices[j], indices[j - 1] = indices[j - 1], indices[j]
            j -= 1

    hull = [indices[0], indices[1]]
    for i in range(2, n):
        while not rotate(points[hull[-2]], points[hull[-1]],
points[indices[i]]):
            hull.pop()
        hull.append(indices[i])
    return [points[i] for i in hull]
def visualize(points, convex_hull):
    plt.figure(figsize=(8, 8))

    x, y = zip(*points)
    plt.scatter(x, y, color='blue', label='Точки')

    hull_x, hull_y = zip(*convex_hull)
    hull_x += (hull_x[0],)
    hull_y += (hull_y[0],)
    plt.plot(hull_x, hull_y, color='red', label='Выпуклая оболочка')
```

```
    plt.title('Выпуклая оболочка точек')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.legend()
    plt.grid(True)
    plt.show()
n = int(input())
points = [list(map(int, input().split(', '))) for _ in range(n)]
convex_hull = graham_scan(points)
area = calculate_area(convex_hull)
print((convex_hull, area))
visualize(points, convex_hull)
```

Название файла: tests.py

```
from main import *
def test_rabin_karp():
    result = Rabin_Karp('abacaba', 'aba')
    assert result == [0, 4]


def test_graham():
    result = graham_scan([[3, 1], [6, 8], [1, 7], [9, 3], [9, 6], [9,
0]])
    assert result == [[1, 7], [3, 1], [9, 0], [9, 3], [9, 6], [6, 8]]


def test_graham_square():
    result = graham_scan([[3, 1], [6, 8], [1, 7], [9, 3], [9, 6], [9,
0]])
    assert calculate_area(result) == 47.5
```