

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

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Поиск образца в тексте: алгоритм Рабина-Карпа. Построение
выпуклой оболочки: алгоритм Грэхема

Студент гр. 3388

Снигирев А.А.

Преподаватель

Шалагинов И.В.

Санкт-Петербург

2024

Цель работы

Изучить и реализовать алгоритмы Рабина-Карпа и Грэхема. Представить визуализацию работы для алгоритма Грэхема. Провести исследование.

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

$$1 \leq |\text{Pattern}| \leq |\text{Text}| \leq 5 \cdot 10^5.$$

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

Исходный код программы см. в приложении А.

Для алгоритма Рабина-Карпа разработаны следующие функции:

`rabin_karp` - реализует алгоритм Рабина-Карпа для поиска всех вхождений подстроки `pattern` в строку `text` с использованием полиномиального хеширования. Сначала вычисляется хэш искомой строки. После этого цикл проходит по всем «окнам» длины `len(pattern)` и сравнивает их хэш с искомым. Это быстрее простого поиска.

Для алгоритма Грэхема разработаны следующие функции :

`check_positions(p1, p2, p3)` - Вычисляет векторное произведение для векторов, состоящих из входных точек. Это используется для определения ориентации поворота, чтобы понять, является ли поворот между векторами `p1p2` и `p2p3` левым или правым. Возвращает положительное значение, если поворот левый, отрицательное — если правый, ноль — если точки лежат на одной прямой.

`graham(points)` - применяет алгоритм Грэхема для построения выпуклой оболочки множества точек. Сортирует точки, затем совершает обход точек, начиная с самой правой, проходя через все точки и добавляя их в стек, если они создают левый поворот (на основе функции `check_positions`). Возвращает список точек, образующих выпуклую оболочку.

`calculate_area(points)` - Вычисляет меру фигуры, заданной последовательностью точек. Использует метод Гаусса , где площадь вычисляется как сумма площадей треугольников, площадь каждого из которых можно вычислить как векторное произведение векторов.

Вывод

Были реализованы алгоритм Рабина-Карпа и Грэхема, а также визуализация для алгоритма Грэхема.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: `graham.py` – основная логика алгоритма Грэхема

```
def check_positions(p1: list[int], p2: list[int], p3: list[int]):
    return (p2[0]-p1[0])*(p3[1]-p2[1])-(p2[1]-p1[1])*(p3[0]-
p2[0])

def calculate_area(points: list[list[int]]):
    area = 0
    for i in range(len(points)):
        j = (i + 1) % len(points)
        area += points[i][0] * points[j][1] - points[i][1] *
points[j][0]
    return abs(area)/2.0

def graham_algorithm(points: list[list[int]]):
    length = len(points)
    indexes = [i for i in range(length)]

    for i in range(1,length):
        if points[indexes[i]][0]<points[indexes[0]][0]:
            indexes[i], indexes[0] = indexes[0], indexes[i]
    for i in range(2,length):
        j = i
        while j>1 and
(check_positions(points[indexes[0]],points[indexes[j-
1]],points[indexes[j]])<0):
            indexes[j], indexes[j-1] = indexes[j-1], indexes[j]
            j -= 1
    stack = [indexes[0],indexes[1]]
    for i in range(2,length):
        while check_positions(points[stack[-2]],points[stack[-
1]],points[indexes[i]])<0:
            stack.pop()
        stack.append(indexes[i])
    ans = [points[i] for i in stack]
    return ans
```

Название файла — `karp.py` – основная логика алгоритма Рабина-Карпа

```
def rabin_karp(pattern: str, text: str):
    p = 10**9 + 7
    x = 263

    m = len(pattern)
    n = len(text)

    pattern_hash = 0
    for i in range(m):
        pattern_hash = (pattern_hash * x + ord(pattern[i])) % p

    x_pow_m = 1
```

```

for _ in range(m - 1):
    x_pow_m = (x_pow_m * x) % p

current_hash = 0
for i in range(m):
    current_hash = (current_hash * x + ord(text[i])) % p

occurrences = []
for i in range(n - m + 1):
    if current_hash == pattern_hash:
        if text[i:i + m] == pattern:
            occurrences.append(i)

    if i < n - m:
        current_hash = (current_hash - ord(text[i]) *
x_pow_m) % p
        current_hash = (current_hash * x + ord(text[i + m]))
% p
        current_hash = (current_hash + p) % p
return occurrences

```