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

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Построение и Анализ Алгоритмов»
Тема: Кнут-Моррис-Пратт.

Студент гр. 3343 _____ Атоян М.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2025

Цель работы.

Изучить принцип работы алгоритма Кнута-Морриса-Пратта для нахождения входов подстроки в строку.

Задание.

1. Реализуйте алгоритм КМП и с его помощью для заданных шаблона P (|P|≤15000$|P|\leq 15000$) и текста T (|T|≤5000000$|T|\leq 5000000$) найдите все вхождения P в T.
Вход:
Первая строка - P
Вторая строка - T
Выход:
индексы начал вхождений P P в TT, разделенных запятой, если PP не входит в T, то вывести −1−1

Sample Input:

ab
abab

Sample Output:

0,2

2. Заданы две строки A (|A|≤5000000$|A|\leq 5000000$) и B (|B|≤5000000$|B|\leq 5000000$).
Определить, является ли A циклическим сдвигом B (это значит, что A и B имеют одинаковую длину и A состоит из суффикса B, склеенного с префиксом BB). Например, defabc является циклическим сдвигом abcdef.
Вход:
Первая строка - A
Вторая строка - B
Выход:
Если A вляется циклическим сдвигом B, индекс начала строки B в A, иначе вывести −1−1. Если возможно несколько сдвигов вывести первый индекс.

Sample Input:

defabc
abcdef

Sample Output:

3

Описание алгоритма Кнута-Морриса-Пратта.

Алгоритм Кнута-Морриса-Пратта (KMP) — это эффективный алгоритм поиска подстроки в строке, который позволяет избежать повторных сравнений символов, что делает его быстрее наивного алгоритма поиска подстроки. Основная идея KMP заключается в использовании предварительно вычисленной таблицы префиксов, которая позволяет "перепрыгивать" через символы, уже успешно сравненные, при обнаружении несовпадения. Таблица префиксов для строки P (искомой подстроки) — это массив, где каждый элемент хранит длину наибольшего префикса строки P, который также является суффиксом текущей подстроки. Для построения таблицы используется два указателя: один идёт по строке, а другой отслеживает длину текущего префикса. Если символы совпадают, то значение в таблице увеличивается, и оба указателя сдвигаются вперёд. Если символы не совпадают, то указатель префикса сдвигается назад с использованием значений из таблицы, пока не найдётся совпадение или указатель не достигнет начала строки. После построения таблицы префиксов алгоритм проходит по тексту T, в котором ищется подстрока P. Используются два указателя: один идёт по тексту, а другой по подстроке. Если символы совпадают, оба указателя сдвигаются вперёд. Если символы не совпадают, то указатель подстроки сдвигается назад с использованием таблицы префиксов, пока не найдётся совпадение или указатель не достигнет начала подстроки. Если указатель подстроки достигает её конца, это означает, что подстрока найдена, и алгоритм фиксирует начало вхождения.

Оценка сложности по времени:

Построение таблицы префиксов для подстроки выполняется за O(m). Алгоритм итерируется по всем элементам строки 1 раз за O(n). Переходы между состояниями выполняются за константное время благодаря таблицы префиксов. Отсюда следует, что сложность по времени составляет O(m + n)

Оценка сложности по памяти:

Алгоритм требует инициализации таблицы префиксов длины m, где m – длина подстроки. Отсюда сложность по памяти составляет O(m).

Код программы смотреть в приложении А.

Тестирование.

Тестируем Алгоритм Кнута-Морриса-Прутта.

Ввод	Вывод	Ожидаемый результат
asdajlgadddjgdadd add	[7, 14]	Результат верный
efefeffffefc efef	[0, 2]	Результат верный
jdgfhklshq geg	[]	Результат верный

Исследование.

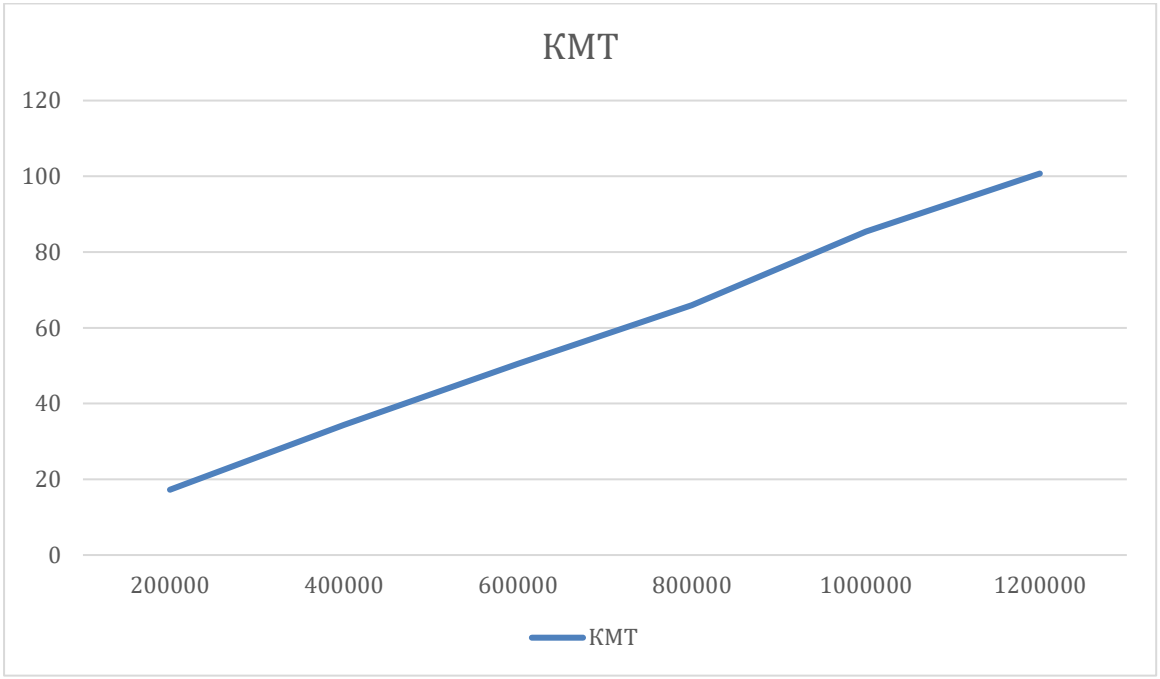

Рисунок 1 - График зависимости времени исполнения КМТ в мс. от суммарной длины строки и подстроки, где размеры строк относятся как 9:1 соответственно.

Экспериментальная сложность алгоритма КМТ составляет O(m+n).

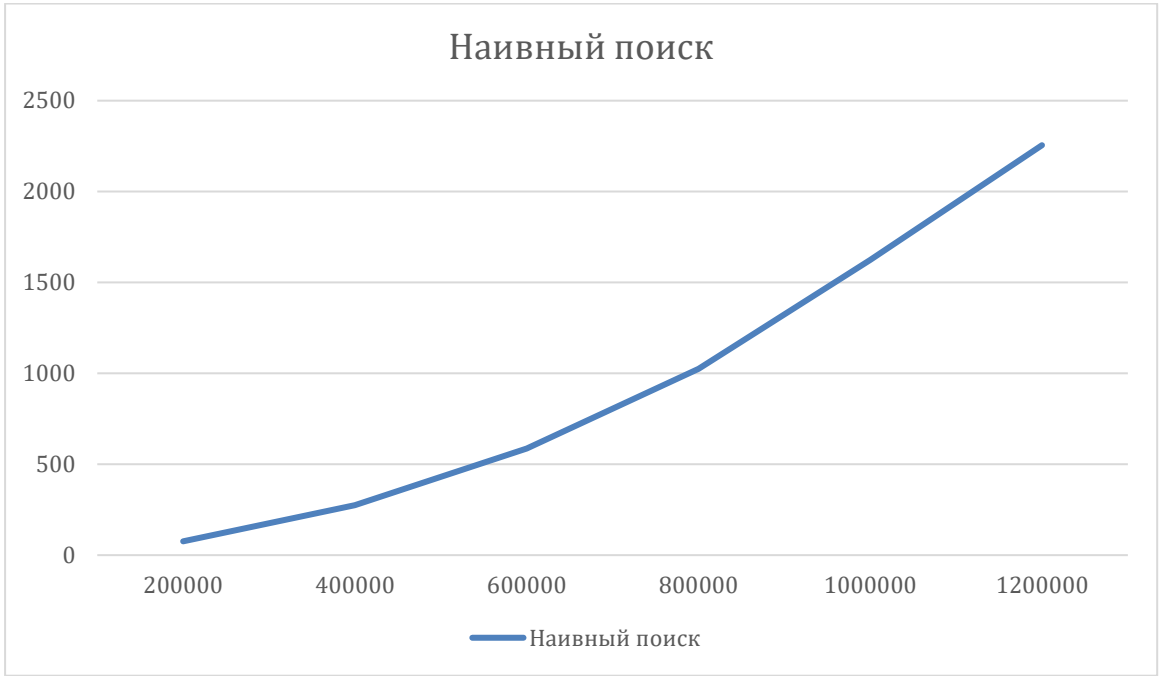

Рисунок 2 - График зависимости времени исполнения наивного алгоритма в мс. от суммарной длины строки и подстроки, где размеры строк относятся как 9:1 соответственно.

Экспериментальная сложность алгоритма наивного поиска составляет O(m*n).

Можно увидеть, что алгоритм Кнута-Морриса-Пратта работает значительно эффективнее наивного поиска благодаря использованию префиксной таблицы, исключающей проход по любому элементу основной строки больше, чем единожды.

Выводы.

В результате работы была написана программа, решающая задачу поиска всех вхождений подстроки в строку с использованием алгоритма Кнута-Морриса-Пратта. Были исследованы и сравнены скорости работы КМП и наивного поиска, где КМП показал себя в разы эффективнее наивного поиска. Программа протестирована. Результаты тестов совпадают с ожиданиями.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: kmp.py

```
def compute_prefix(pattern):
    n = len(pattern)
    pi = [0] * n
    j = 0
    for i in range(1, n):
        while j > 0 and pattern[i] != pattern[j]:
            j = pi[j - 1]
        if pattern[i] == pattern[j]:
            j += 1
            pi[i] = j
        else:
            pi[i] = 0
    return pi

def search(needle, haystack):
    if not needle:
        return []

    pi = compute_prefix(needle)

    occurrences = []
    j = 0
    for i in range(len(haystack)):
        while j > 0 and haystack[i] != needle[j]:
            j = pi[j-1]
        if haystack[i] == needle[j]:
            j += 1
            if j == len(needle):
                occurrences.append(i - j + 1)
                j = pi[j-1]

    return occurrences
```