**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Построение и анализ алгоритмов»**

Тема: **Кнут-Моррис-Пратт**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Тимошук Е.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2025

**Цель работы:**

Изучить теоретические основы алгоритма Кнута-Морриса-Пратта.

**Задание:**

Реализуйте алгоритм КМП и с его помощью для заданных шаблона P (∣P∣≤25000) и текста *T* (∣T∣≤5000000) найдите все вхождения *P* в *T*.

**Входные данные:**  
Вход:  
- Первая строка — P   
- Вторая строка — *T*  
Выход:  
индексы начал вхождений *P*  в  *T*, разделённые запятой; если *P* не входит в *T*, то вывести -1.

**Sample Input:**

ab

abab

**Sample Output:**

0,2

**Выполнение работы**

Подробное описание алгоритма Кнута-Морриса-Пратта (KMP)

1. Назначение алгоритма

Алгоритм KMP решает задачу поиска всех вхождений заданного образца P в тексте T. Основное преимущество перед наивным алгоритмом — линейная сложность O(|P| + |T|) в худшем случае за счёт исключения избыточных сравнений символов.

2. Ключевые идеи

- Префикс-функция (LPS-массив):

Для образца P предварительно вычисляется массив LPS (Longest Proper Prefix which is also Suffix). Каждый элемент LPS[i] хранит длину максимального собственного суффикса подстроки P[0..i], совпадающего с её префиксом.

Пример:

Для P = "ABAB":

LPS = [0, 0, 1, 2]

(Для i=3: суффикс "AB" совпадает с префиксом `"AB"`).

- Оптимизация сдвига:

При несовпадении символов алгоритм использует LPS для определения нового положения указателя в образце. Это позволяет "перепрыгнуть" заведомо нерелевантные участки текста без возврата указателя.

3. Этапы работы

a) Предобработка образца (построение LPS-массива):

1. Инициализировать массив LPS длиной |P|, заполненный нулями.

2. Указатели length = 0 (длина текущего префикса-суффикса), i = 1 (текущий символ).

3. Для каждого символа в P[1..|P|-1]:

- Если P[i] == P[length]:

Увеличить length, записать LPS[i] = length, сдвинуть i вправо.

- Иначе:

Если length > 0: уменьшить length = LPS[length - 1].

Если length = 0: записать LPS[i] = 0, сдвинуть i вправо.

b) Поиск вхождений в тексте:

1. Инициализировать указатели: i = 0 (текст), j = 0 (образец).

2. Пока i < |T|:

- Если P[j] == T[i]:

Сдвинуть оба указателя вправо (`i++`, `j++`).

При полном совпадении (`j == |P|`): зафиксировать начало вхождения (i - j), сбросить j = LPS[j-1].

- Иначе:

Если j > 0: сдвинуть j = LPS[j-1] (используем префикс-функцию для "отката").

Если j = 0: сдвинуть только i++.

4. Сложность алгоритма

- Построение LPS: O(|P|)

- Поиск в тексте: O(|T|)

- Итоговая сложность: O(|P| + |T|)

Преимущество достигается за счёт отсутствия возврата указателя в тексте.

5. Входные и выходные данные

- Вход:

- Образец P (строка),

- Текст T (строка).

- Выход:

- Строка с индексами всех начал вхождений P в T (через запятую),

- -1 при отсутствии вхождений.

Пример: P = "AB", T = "CABDAB" → "1,4".

**Оценка сложности алгоритма:**

**Временная сложность**

Оценка временной сложности алгоритма Кнута–Морриса–Пратта сводится к двум этапам: сначала строится префикс-функция (pi-массив) для шаблона длины m, что делается за время O(m), поскольку каждый символ обрабатывается ровно один раз с небольшим числом откатов по ранее вычисленным значениям pi. Затем проводится проход по тексту длины n, и на каждом шаге мы либо совпадаем и двигаем указатели, либо при несовпадении одним присваиванием j = pi[j–1] быстро откатываемся, без возврата в начало шаблона. В итоге поиск занимает O(n). Суммарная временная сложность получается O(n + m). Пространственная сложность алгоритма — O(m) на хранение pi-массива, плюс константный дополнительный объём памяти на несколько индексов и счётчиков.

**Тестирование**

Таблица 1. Тестирование.

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| abab  ababab | 0,2 |
| nigger  gg | 2 |
| dthrd  jhefvjewrfverfvaverjfvjeravfbrejhvfjrecgvhbrjvxfjgfrebjgnetskbglrgkhctkjgbklhrtvsjfvrekhgkjrtnkhbjevrtbvteabkbxjhlfrsthgkbaeljvjkhrbfilr | -1 |
| ghf  ufreufuerjvhirebfjverfghjrstngkhlvejhfvrekbgklaertjhverxjhgbkhetvjhferhgvfjkestbghkvtrjhfblrtghke5jfwekhghlvftgbgiyevgvehjrfbkrtsbghtruyfgehghgiegrohgu;aelhiu5rgl | -1 |

**Вывод**

В ходе лабораторной работы мы изучили и реализовали алгоритм Кнута–Морриса–Пратта. Мы написали функцию вычисления префикс-массива π и функцию поиска вхождений, подробно отладили их работу на разных примерах и убедились, что алгоритм находит все совпадения за линейное время O(n+m). Полученная реализация демонстрирует, как с помощью π-массива удаётся жёстко избежать повторных сравнений и обеспечивает стабильную производительность даже на длинных строках.