Отчет по лабораторной работе №7
по курсу "Анализ алгоритмов"
по теме "Методы поиска подстроки в строке"

Студент: Барсуков Н.М. ИУ7-56
Преподаватель: Волкова Л.Л., Строганов Ю.В.

Содержание

Введение

Поиск подстроки в строку - одна из простеших задач поиска информации. Применяется в виде встроенной функции в текстовых редакторах, СУБД, поисковых машинах, языках программирования и т. п. В задачах поиска традиционо принято обознать шаблон поиска как needle (с англ. игла), а строку, в которой ведется поиск как haystack (с англ - стог сена). Обычно через $\sum$ обозначается алфавит, на котором проводится поиск.

1 Аналитический раздел

В данном разделе описана цель и задачи необходимые для выполнения онной. Дано полное описание алгоритма Кнута - Моррса - Прата и Бойера - Мура и их математические описания.

1.1 Цель и задачи

Целью данной работы является изучение способов поиска подстроки в строке с помощью алгоритмов Кнута-Мориса-Прата и Бойера-Мура. Для достижения поставленной цели требуется решить следующие задачи:

1. изучить алгоритмы:
   (a) стандартный;
   (b) Кнута-Мориса-Прата;
   (c) Бойера-Мура;
2. математически описать решение задачи поиска подстроки на основании выше указанных алгоритмов;
3. реализовать выше указанные алгоритмы;
4. сравнить;
5. сделать выводы.

1.2 Алгоритмы

1.2.1 Обзор алгоритма Кнута-Мориса-Прата

В задачах поиска информации одной из важнейших задач является поиск точно заданной подстроки в строке. Примитивный алгоритм поиска подстроки в строке основан на переборе всех подстрок, длина которых равна длине шаблона поиска, и посимвольном сравнении таких подстрок с шаблоном поиска.

Алгоритм Кнута-Морриса-Пратта является одним из первых алгоритмов с линейной оценкой в худшем случае.

Обозначим через:

$$n = |heystack|, m = |needle| \tag{1}$$

где |word|, обозначает длину слова word

Префикс-функция строки π(S,i) – это длина наибольшего префикса строки S[1..i], который не совпадает с этой строкой и одновременно является ее суффиксом. Проще говоря, это длина наиболее длинного начала строки,

Таблица 1: Пример префикс функции для строки abcdabca

$S[i]$	a	b	c	d	a	b	c	a
$\pi(S,i)$	0	0	0	0	1	2	3	1

являющегося также и ее концом. Для строки S удобно представлять префикс функцию в виде вектора длиной |S|-1. Можно рассматривать префикс-функцию длины |S|, положив π(S,1)=0. Пример приведен на таблице 1.2.1

Наиболее полный алгоритм вычисления префикс-функции на псевдокоду показан в листинге 1

Algorithm 1: Псевдокод функции поиска префикса

```
1 алг префикс_функция (арг строка S[1 ... i]) begin
2     Преположим что π(S, i) = k;
3     if S[i + 1] = S[K + 1], mo then
4         π(S, i + 1) = k + 1;
5     else
6         if k = 0 then
7             π(S, i + 1) = 0.;
8         else
9             Положить k_i = π(S, k) и прейти к шагу 3;
```

Рассмотрим алгоритм Кнута-Морриса-Пратта. Пусть S[0. . . m–1] – образец, T[0. . . n–1] – строка, в которой ведется поиск. Рассмотрим сравнение строк на позиции i, то есть образец S[0. . . m–1] сопоставляется с частью строки T[i. . . i+m–1]. Предположим, первое несовпадение произошло между символами S[j] и T[i+j], где i < j < m. Обозначим P = S[0. . . j–1] = T[i. . . i+j–1]. При сдвиге можно ожидать, что префикс S сойдется с каким-либо суффиксом строки P. Поскольку длина наиболее длинного префикса, являющегося одновременно суффиксом, есть префикс-функция от строки S для индекса j, приходим к следующему алгоритму.

1) шаг 1: построить префикс-функцию образца S, обозначим ее F;

2) шаг 2: положить k = 0, i = 0;

3) шаг 3: сравнить символы S[k] и T[i]. Если символы равны, увеличить k на 1. Если при этом k стало равно длине образца, то вхождение образца S в строку T найдено, индекс вхождения равен i − |S| + 1. Алгоритм завершается. Пока k > 0, присвоим k = F[k–1] и переходим в начало шага 3;

4) шаг 4: Пока i < |T|, увеличиваем i на 1 и переходим к шагу 3.

1.3 Обзор алгоритма Бойера - Мура

Алгоритм сравнивает символы шаблона x справа налево, начиная с самого правого, один за другим с символами исходной строки y. Если символы совпадают, производится сравнение предпоследнего символа шаблона и так до конца. Если все символы шаблона совпали с наложенными символами строки, значит, подстрока найдена, и поиск окончен. В случае несовпадения какого-либо символа (или полного совпадения всего шаблона) он использует две предварительно вычисляемых эвристических функции, чтобы сдвинуть позицию для начала сравнения вправо.

Алфавит обозначим буквой $\sum$ Пусть |y|= n,|x|= m и $|\sum|= \sigma$. Предположим, что в процессе сравнения возникает несовпадение между символом x[i]=a шаблона и символом y[i+j]=b исходного текста при проверке на позиции j. Тогда:

$$x[i+1...m-1] = y[i+j+1...j+m-1] = u \tag{2}$$

и

$$x[i] \neq y[i+j] \tag{3}$$

тогда m - i - 1 символом шаблона не совпало (оставшийся “хвост” подстроки). В целом алгоритм можно описать с помощью трех ключевых положений.

1.3.1 Сканирование слева направо, сравнение справа налево

Совмещается начало текста (строки) и шаблона, проверка начинается с последнего символа шаблона. Если символы совпадают, производится сравнение предпоследнего символа шаблона и т. д. Если все символы шаблона совпали с наложенными символами строки, значит, подстрока найдена, и выполняется поиск следующего вхождения подстроки. Если же какой-то символ шаблона не совпадает с соответствующим символом строки, шаблон сдвигается на несколько символов вправо, и проверка снова начинается с последнего символа.

1.3.2 Эвристика стоп-символа

Эвристика стоп-символа присутствует в большинстве описаний алгоритма Бойера — Мура, включая оригинальную статью Бойера и Мура, но не является необходимой для достижения оценки O(n+m) времени работы. Предположим, что мы производим поиск слова «колокол». Первая же буква не совпала — «к» (назовём эту букву стоп-символом). Тогда можно сдвинуть шаблон вправо до последней его буквы «к», что показано в таблице 2

Таблица 2: Поиск слова колокол в строке к

Строка	*	*	*	*	*	*	к	*	*	*	*	*	*
Шаблон	к	о	л	о	к	о	л						
След. шаг			к	о	л	о	к	о	л				

Если стоп-символ «к» оказался за другой буквой «к», эвристика стоп-символа не работает. В таких ситуациях может быть полезна третья идея алгоритма Бойера — Мура — эвристика совпавшего суффикса.

1.3.3 Эвристика совпавшего суффикса

Если при чтении шаблона справа налево совпал суффикс S, а символ b, стоящий перед S в шаблоне (т. е. шаблон имеет вид PbS), не совпал, то эвристика совпавшего суффикса сдвигает шаблон на наименьшее число позиций вправо так, чтобы строка S совпала с шаблоном, а символ, предшествующий в шаблоне данному совпадению S, отличался бы от b (если такой символ вообще есть). Для данного шаблона s[0... m-1] считается целочисленный массив suffshift[0... m], в котором suffshift[i] равно минимальному числу j>0, такому, что s[i-j] $\neq$ s[i-1] и s[i-j+k]=s[i-1+k] для любого k>0, для которого выполняется $0 \leq$ i-j+k $<$ m и $0 \leq$ i-1+k $<$ m. 3

Таблица 3: Поиск слова “скалкалка” в строке, рассмотрение суффикса “рка”

Строка	*	*	*	*	*	*	р	к	а	*	*	*	*	*	
Шаблон	с	к	а	л	к	а	л	к	а						
След. шаг							с	к	а	л	к	а	л	к	а

В данном случае совпал суффикс «ка», и шаблон сдвигается вправо до ближайшего «ка», перед которым нет буквы «л».

1.4 Вывод

В данном разделе было приведено общее и алгоритмическое описание алгоритмов Кнута-Морриса-Пратта и Бойера-Мура, приведены примеры.

2 Конструкторский раздел

3 Технологический раздел

4 Иследовательский раздел

5 Заключение

Список литературы