Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра компьютерных технологий

И. Р. Ахметов

Поиск подстрок с помощью конечных автоматов

Санкт-Петербург
2005

Содержание

Введение 3

1. Обозначения и терминология. 3

2. Конечные автоматы 4

3. Автомат для поиска подстрок 4

   3.1. Структура автомата 4

   3.2. Построение автомата по образцу 5

   3.3. Применение построенного автомата для поиска образца в тексте. . . . 6

4. Пример построения конечного автомата 6

Заключение. 8

Приложение 1. Исходные код программы 9

Приложение 2. Исходные данные 10

Приложение 3. Результат работы программы 11

Введение

Задача поиска подстроки в строке является одной из наиболее известных задач информатики. Она тщательна изучена и предложено множество алгоритмов для решения этой задачи и ее модификаций ([1], [2]).

В настоящей работе излагается алгоритм поиска подстрок с помощью конечных автоматов. Его достоинства заключаются в его наглядности и простоте и вместе с тем он является линейным от длины текста временем работы (при построенном конечном автомате).

1. Обозначения и терминология

Пусть даны «текст» — массив $T[1\,..\,n]$ длины n и «образец» — массив $P[1\,..\,m]$ длины $m \leq n$. Будем считаем, что элементы массивов P и T — символы некоторого конечного алфавита Σ (например, $\Sigma = \{0, 1\}$ или $\Sigma = \{a, b, ..., z\}$). Массивы, состоящие из символов алфавита Σ, часто называют *строками* символов или *словами* в этом алфавите.

Будем говорить, что образец P *входит со сдвигом* s, или, что то же самое, *входит с позиции* $s + 1$ в текст T, если $0 \leq s \leq n - m$ и $T[s + 1\,..\,s + m] = P[1\,..\,m]$ (иными словами, если $T[s + j] = P[j]$ при $1 \leq j \leq m$). Если P входит со сдвигом s в текст T, то говорят, что s — *допустимый сдвиг*, в противном случае s — *недопустимый сдвиг*. Задача поиска подстрок состоит в нахождении всех допустимых сдвигов для данных текста T и образца P.

Через Σ^* обозначим множество всех конечных строк в алфавите Σ, включая *пустую строку*, имеющую нулевую длину и обозначаемую ε. Длина строки x обозначается $|x|$. *Конкатенация* строк x и y образуется, если выписать строку x, а сразу за ней — строку y. Конкатенация строк x и y обозначается xy, при этом $|xy| = |x| + |y|$.

Говорят, что строка ω — *префикс* строки x, если $x = \omega y$ для некоторого $y \in \Sigma^*$, а строка ω — суффикс строки x, если $x = y\omega$ для некоторого $y \in \Sigma^*$. Будем писать $\omega \sqsubset x$, если ω — префикс x, и $\omega \sqsupset x$, если ω — суффикс x. Например, $\mathtt{ab} \sqsubset \mathtt{abcca}$ и $\mathtt{cca} \sqsupset \mathtt{abcca}$.

Пустая строка является префиксом и суффиксом любой строки; если ω — префикс или суффикс x, то $|\omega| \leq |x|$. Для любых строк x и y и для любого символа a соотношения $x \sqsupset y$ и $xa \sqsupset ya$ равносильны. Отношения $\sqsubset$ и $\sqsupset$ транзитивны.

Если $S[1\,..\,r]$ — строка длины r, то ее префикс длины $k \leq r$ будет обозначаться $S_k = S[1\,..\,k]$ (в частности, $S_0 = \varepsilon$ и $S_r = S$). В этих обозначениях задача о нахождении образца P длины m в тексте T длины n состоит в нахождении всех таких s из промежутка $0 \leq s \leq n - m$, что $P \sqsupset T_{s+m}$.

2. Конечные автоматы

По определению, *конечный автомат* представляет собой пятерку $M = (Q, q_0, A, \Sigma, \delta)$, где:

- Q — конечное множество *состояний*;
- $q_0 \in Q$ — *начальное состояние*;
- $A \subseteq Q$ — конечное множество *допускающих состояний*;
- Σ — конечный *входной алфавит*;
- δ — функция $Q \times \Sigma \to Q$, называемая *функцией переходов* автомата.

Первоначально конечный автомат находится в состоянии q_0. Затем он по очереди читает символы из входной строки. Находясь в состоянии q и читая символ a, автомат переходит в состояние $\delta(q, a)$. Если автомат находится в состоянии $q \in A$, говорят, что он *допускает* прочитанную часть входной строки. Если $q \notin A$, то прочитанная часть строки *отвергнута*.

С конечным состоянием M связана функция $\varphi\colon \Sigma^* \to Q$, называемая *функцией конечного состояния*, определяемая следующим образом: $\varphi(\omega)$ есть состояние, в которое придет автомат (из начального состояния), прочитав строку ω. Автомат допускает строку ω тогда и только тогда, когда $\varphi(\omega) \in A$. Функцию φ можно определить реккурентно:

$$\begin{aligned} \varphi(\varepsilon) &= q_0; \\ \varphi(\omega a) &= \delta(\varphi(\omega), a) \qquad \text{для любых } \omega \in \Sigma^* \text{ и } a \in \Sigma. \end{aligned}$$

3. Автомат для поиска подстрок

3.1. Структура автомата

Для каждого образца P можно построить конечный автомат, ищущий этот образец в тексте. Первым шагом в построении автомата, соответствующего строке-образцу $P[1\,..\,m]$, является построение по P вспомогательной функции, называемой *суффикс-функцией*. По определению, суффикс-функция $\sigma\colon \Sigma^* \to \{0, 1, \ldots, m\}$ ставит в соответствие строке x длину максимального суффикса x, являющегося префиксом P:

$$\sigma(x) = \max\{k : P_k \sqsupset x\}.$$

Поскольку $P_0 = \varepsilon$ является суффиксом любой строки, σ определена на всем Σ^*.

Теперь определим конечный автомат, соотвествующий образцу $P[1\,..\,m]$, следующим образом:

- его множество состояний $Q = \{0, 1, \ldots, m\}$. Начальное состояние $q_0 = 0$. Единственное допускающее состояние m;
- функция переходов δ определена соотношением (q — состояние, $a \in \Sigma$ — символ):

$$\delta(q, a) = \sigma(P_q a). \tag{1}$$

Поясним это соотношение. Требуется сконструировать автомат таким образом, чтобы при его действии на строку T соотношение

$$\varphi(T_i) = \sigma(T_i)$$

являлось инвариантом (тогда равенство $\varphi(T_i) = m$ будет равносильно тому, что P входит в T со сдвигом $i - m$, и автомат тем самым найдет все допустимые сдвиги). Однако в этом случае вычисление перехода по формуле (1) необходимо для поддержания истинности инварианта, что следует из теорем, приведенных ниже.

Теорема. *Пусть $q = \sigma(x)$, где x — строка. Тогда для любого символа a имеет место $\sigma(xa) = \sigma(P_q a)$.*

Теорема. *Пусть φ — функция конечного состояния автомата для поиска подстроки $P[1\,..\,m]$. Если $T[1\,..\,n]$ — произвольный текст, то*

$$\varphi(T_i) = \sigma(T_i)$$

для $i = 0, 1, \ldots, n$.

Из изложенного следует, что задача поиска подстроки состоит из двух частей:

- построение автомата по образцу (определение функции переходов для заданного образца);
- применение этого автомата для поиска вхождений образца в заданный текст.

3.2. Построение автомата по образцу

Функцию переходов δ, соотвествующую образцу $P[1\,..\,m]$, можно вычислить так:

```
COMPUTE-TRANSITION-FUNCTION(P, Σ)
1  m ← lenth[P]
2  for q ← 0 to m do
3       for (для) всех символов a ∈ Σ do
4            k ← min(m + 1, q + 2)
5            repeat k ← k − 1
6            until P_k ⊐ P_q a
```

7 $\quad\quad\quad\quad \delta(q,a) \leftarrow k$

8 **return** δ

Эта процедура вычисляет функцию δ «в лоб»: циклы, начинающиеся в строках 2 и 3, перебирают все пары (q,a), а в строках 4–7 наибольшее значение k, при котором для данной пары (q,a) выполнено соотношение $P_k \sqsupset P_q a$, находится прямым перебором, начиная с наибольшего возможного значения $k = \min(m, q+1)$.

Время работы этого алгоритма $O(m^3|\Sigma|)$: в самом деле, два внешних цикла дают множитель $m|\Sigma|$, внутренний цикл **repeat** может выполняться не более $m+1$ раз, а сравнение в строке 6 требует $O(m)$ операций.

3.3. Применение построенного автомата для поиска образца в тексте

Опишем функционирование конечного автомата, ищущего подстроку P длины m в данном тексте T, в виде программы (δ обозначает функцию переходов):

FINITE-AUTOMATON-MATCHER(T, δ, m)

1 $n \leftarrow lenth[T]$

2 $q \leftarrow 0$

3 **for** $i \leftarrow 1$ **to** n **do**

4 $\quad\quad q \leftarrow \delta(q, T[i])$

5 $\quad\quad$ **if** $q = m$ **then**

6 $\quad\quad\quad\quad s \leftarrow i - m$

7 $\quad\quad\quad\quad$ print «Образец входит со сдвигом» s

Поскольку эта программа обрабатывает каждый символ из текста T по одному разу, время ее работы $O(n)$.

4. Пример построения конечного автомата

Построим конечный автомат, допускающий строку `ababaca`. Поскольку длина образца $m = 7$ символов, то в автомате будет $m + 1 = 8$ состояний.

Найдем функцию переходов δ. В соответствии с определением (1), $\delta(q,a) = \sigma(P_q a)$, где σ — префикс-функция, a — произвольный символ из алфавита Σ, q — номер состояния. Таким образом, необходимо для каждого префикса $P_q = P[0\,..\,q]$, $q = 0\,..\,m$ образца P и для всех символов a входного алфавита Σ найти длину максимального префикса P, который будет являться суффиксом строки $P_q a$. Длина этого префикса и будет значением функции переходов $\delta(q,a)$. Если $a = P[q+1]$ (очередной символ текста совпал со следующим символом образца), то $P_q a = P_{q+1}$ и $\delta(q,a) = q+1$. Такой

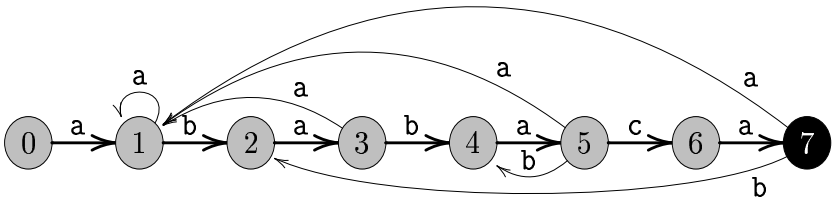
случай соответствует успешным этапам поиска. Иначе, $\delta(q, a) \leqslant q$. Например, для префикса $P[0\,..\,5] = \texttt{ababa}$ и символа `b` максимальным суффиксом строки $P[0\,..\,5]\texttt{b} = \texttt{ababab}$, который одновременно является префиксом P, будет `abab`. Его длина равна 4, поэтому значение функции переходов $\delta(5, \texttt{b}) = 4$.

Запишем построенную таким образом функцию переходов в виде таблицы:

	0	1	2	3	4	5	6	7
a	1	1	3	1	5	1	7	1
b	0	2	0	4	0	4	0	2
c	0	0	0	0	0	6	0	0

Строки соответствуют входным символам, столбцы — состояниям автомата. Ячейки, соответствующие успешным этапам поиска (входной символ совпадает со следующим символом образца), выделены серым цветом.

Построим по таблице граф переходов автомата, распознающего образец `ababaca`. Находясь в состоянии q и прочитав очередной символ a, автомат переходит в состояние $\delta(q, a)$. Обратим внимание, что его остов помечен символами образца (эти переходы выделены жирными стрелками).



Здесь 0 — исходное состояние, 7 — единственное допускающее состояние (зачернено). Если из вершины i в вершину j ведет стрелка, помеченная буквой a, то это означает, что $\delta(i, a) = j$. Отметим, что переходы, для которых $\delta(i, a) = 0$, на графе переходов для его упрощения не обозначены. Жирные стрелки, идущие слева направо, соответствуют успешным этапам поиска подстроки P — следующий входной символ совпадает с очередным символом образца. Стрелки, идущие справа налево, соответствуют неудачам — следующий входной символ отличается от очередного символа образца.

Ниже приведен результат применения автомата к тексту $T = \texttt{abababacaba}$. Под каждым символом $T[i]$ записано состояние автомата после прочтения этого символа (иными словами, значение $\varphi(T_i)$).

i	—	1	2	3	4	5	6	7	8	9	10	11
$T[i]$	—	a	b	a	b	a	b	a	c	a	b	a
состояние $\varphi(T_i)$	0	1	2	3	4	5	4	5	6	7	2	3

Найдено одно вхождение образца (начиная с позиции 3). Найденный образец в тексте помечен серым цветом. Черным цветом помечено допускающее состояние автомата (состояние с номером 7).

Заключение

В настоящей работе было показано, что хотя сам поиск подстроки при заданном автомате работает за время $O(n)$ (где n — длина текста), предложенный алгоритм построения конечного автомата для образца длины m работает за время $O(m^3|\Sigma|)$ (где $|\Sigma|$ — количество символов в алфавите). Такие образом, рассмотренный метод поиска подстроки имеет смысл применять только при сравнительно небольших длинах образца.

В практических приложениях обычно используются более быстрые алгоритмы, такие как алгоритм Кнута-Морриса-Пратта или алгоритм Бойера-Мура ([1], [2]).

Список литературы

[1] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 2002.

[2] *Gusfield D.* Algorithms on strings, trees and sequences: computer science and computational biology // New York: Cambridge University Press, 1997. [Русск. пер.: *Гасфилд Д.* Строки, деревья и последовательности в алгоритмах. Информатика и вычислительная биология. СПб.: Невский Диалект, 2003.]

Приложение 1. Исходные код программы

```
{$o-,q+,r+}
{$apptype console}

uses math;

const
    maxplen = 400;

type
    transfunc = array[0..maxplen, 1..255] of integer;

var
    i: integer;
    p, t: string;
    hasch: array[1..255] of boolean;
    f: transfunc;

{ Построение автомата по образцу }
procedure computefunc(const p: string; var f: transfunc);
var
    i, j, k, m, t1, t2: integer;
begin
    m := length(p);
    for i := 0 to m do begin
        for j := 1 to 255 do if hasch[j] then begin
            k := min(m + 1, i + 2);
            while k > 0 do begin
                dec(k);
                if (k > 0) and (ord(p[k]) = j) then begin
                    t1 := k - 1; t2 := i;
                    while (t1 > 0) and (p[t1] = p[t2]) do begin
                        dec(t1); dec(t2);
                    end;
                    if t1 = 0 then break;
                end;
            end;
            f[i, j] := k;
        end;
    end;
end;

{ Поиск подстроки }
procedure match(const t: string; var f: transfunc; const m: integer);
var
    n, q, i: integer;
begin
        writeln('Осуществим проход по тексту. В начале автомат находится в состоянии q
            = 0.');
        writeln('Взяв следующий символ ''a'' текста T, автомат переходит в состояние f[q
            , ''a''].');
        writeln;

    n := length(t);
    q := 0;
    for i := 1 to n do begin
            write('T[', i, '] = ', '''', t[i], ''', f[', q, ', ''', t[i], ''] = ', f[q,
                ord(t[i])]);
        q := f[q, ord(t[i])];
        if q = m then begin
            write(', мы попали в состояние ', m,
                  ', следовательно образец входит в текст с позиции ', i - m + 1, '.');
        end;
        writeln;
    end;
end;

procedure fill(const n: integer; const ch: char);
var
    i: integer;
begin
    for i := 1 to n do write(ch);
end;
```

```
procedure printfunc(var f: transfunc; const m: integer);
var
    i, j, t, digits: integer;
begin
    writeln;
    writeln('Длина образца |P| = ', m, ', поэтому количество состояний в автомате - ', m
        + 1, ' (|P|+1).');
    writeln('Автомат находит вхождение образца, если он попадает в допускающее состояние
        ', m, '.');
    writeln('Построим функцию переходов f. Для каждого префикса P[0..m], где m = 0..', m
        , ',');
    writeln('переберем все символы ''a'', встречающиеся в образце, и найдем максимальный'
        );
    writeln('суффикс строки P[0..m]a, который является префиксом P. Длину найденного
        префикса');
    writeln('сохраним в ячейке f[m, ''a''] функции переходов.');
    writeln;
    writeln('Полученная в результате функция переходов имеет вид:');
    writeln;
    t := m; digits := 0;
    while t > 0 do begin
        inc(digits); t := t div 10;
    end;

    write('  ');
    for i := 0 to m do begin
        write('| ');
        write(i:digits);
    end;
    writeln;

    for i := 1 to 255 do if hasch[i] then begin
        write('--');
        for j := 0 to m do begin
            write('+');
            fill(digits + 1, '-');
        end;
        writeln;
        write(' ', chr(i));
        for j := 0 to m do begin
            write('| ');
            write(f[j, i]:digits);
        end;
        writeln;
    end;
    writeln;
end;

begin
    reset(input, 'input.txt');
    rewrite(output, 'output.txt');

    readln(t); readln(p);
    writeln('Ищем образец "', p, '" (P) в тексте "', t, '" (T).');
    for i := 1 to length(p) do hasch[ord(p[i])] := true;

    computefunc(p, f);
    printfunc(f, length(p));
    match(t, f, length(p));
end.
```

Приложение 2. Исходные данные

Первая строка исходных данных задает текст, вторая строка — образец:

```
abababacaba
ababaca
```

Приложение 3. Результат работы программы

```
Ищем образец "abababaca" (P) в тексте "abababacaba" (T).

Длина образца |P| = 7, поэтому количество состояний в автомате - 8 (|P|+1).
Автомат находит вхождение образца, если он попадает в допускающее состояние 7.
Построим функцию переходов f. Для каждого префикса P[0..m], где m = 0..7,
переберем все символы 'a', встречающиеся в образце, и найдем максимальный
суффикс строки P[0..m]a, который является префиксом P. Длину найденного префикса
сохраним в ячейке f[m, 'a'] функции переходов.

Полученная в результате функция переходов имеет вид:

  | 0| 1| 2| 3| 4| 5| 6| 7
--+--+--+--+--+--+--+--+--
 a| 1| 1| 3| 1| 5| 1| 7| 1
--+--+--+--+--+--+--+--+--
 b| 0| 2| 0| 4| 0| 4| 0| 2
--+--+--+--+--+--+--+--+--
 c| 0| 0| 0| 0| 0| 6| 0| 0

Осуществим проход по тексту. В начале автомат находится в состоянии q = 0.
Взяв следующий символ 'a' текста T, автомат переходит в состояние f[q, 'a'].

T[1] = 'a', f[0, 'a'] = 1
T[2] = 'b', f[1, 'b'] = 2
T[3] = 'a', f[2, 'a'] = 3
T[4] = 'b', f[3, 'b'] = 4
T[5] = 'a', f[4, 'a'] = 5
T[6] = 'b', f[5, 'b'] = 4
T[7] = 'a', f[4, 'a'] = 5
T[8] = 'c', f[5, 'c'] = 6
T[9] = 'a', f[6, 'a'] = 7, мы попали в состояние 7, следовательно образец входит в текст
    с позиции 3.
T[10] = 'b', f[7, 'b'] = 2
T[11] = 'a', f[2, 'a'] = 3
```