МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«20» 09 2022г.

ПРИМЕНЕНИЕ КОНЕЧНЫХ АВТОМАТОВ ДЛЯ ПОИСКА И РАСПОЗНАВАНИЯ ПОДСТРОК

Методические рекомендации к лабораторным работам
для студентов направления 09.03.01

Курск 2022

УДК 681.3

Составители: И.Е. Чернецкая

Рецензент
Кандидат технических наук, доцент *Т.Н. Конаныхина*

Применение конечных автоматов для поиска и распознавания подстрок: методические рекомендации к лабораторной работе / Юго-Зап. гос. ун-т; сост.: И.Е. Чернецкая. – Курск, 2022. - 24 с.: - ил. 6.– Библиогр.: с. 24

Содержит сведения о способах программной реализации детерминированных конечных автоматов. Рассматриваются приложения теории автоматов к задачам поиска и распознавания подстрок: метод конечных автоматов и метод Кнута-Морриса-Пратта. Каждый из методов подкреплен примером.

Методические рекомендации соответствуют рабочей программе дисциплины «Теория автоматов».

Предназначены для студентов направления 09.03.01 очной и заочной форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60*84 1/16.
Усл. печ. л.1,4. Уч.-изд. л. 1,26. Тираж 50 экз. Заказ 1902. Бесплатно.
Юго-Западный государственный университет.
305040 Курск, ул. 50 лет Октября, 94.

Оглавление

1. ЦЕЛЬ РАБОТЫ

Освоить программную реализацию конечных автоматов. Научиться наиболее эффективным методам поиска и распознавания подстрок: методу конечных автоматов и алгоритму Кнута-Морриса-Пратта.

2. ОСНОВНЫЕ ПОНЯТИЯ

2.1. Строки и операции над строками

Строкой называется последовательность символов, взятых из некоторого алфавита. Длина строки A равна количеству символов в этой строке (обозначается $|A|$). Особым типом строки является пустая строка **ε** нулевой длины.

Строки A и B равны ($A=B$), если они имеют равную длину, один и тот же состав символов, и порядок следования символов в строках совпадает.

Префиксом строки A называется строка P, полученная удалением нуля или более последних символов строки A (обозначается $P \sqsubset A$).

Суффиксом строки A называется строка S, полученная удалением нуля или более первых символов строки A (обозначается $S \sqsupset A$).

Подстрока строки A получается удалением префикса и суффикса строки A. Пустая строка ε является префиксом, суффиксом и подстрокой любой строки. Отсюда следует, что все префиксы и суффиксы строки A являются ее подстроками. Сама строка A также является своим собственным префиксом и суффиксом.

Над строками определены следующие операции:

- ***конкатенация*** (сложение) строк A и B – дописывание символов строки B в конец строки A (обозначается AB). Например, A = **под**, B = **мост**, конкатенация AB = **подмост**.
- ***итерация*** (повторение) строки n раз, $n \geq 0$ (обозначается A^n) – это конкатенация строки самой с собой n раз. A^0=ε – пустая строка. Например, A = **ма**, A^2 = **мама**.
- ***обращение*** (инверсия) строки, обозначается A^R, – запись символов строки в обратном порядке. Например, A = **тор**, A^R = **рот**.

2.2. Постановка задачи поиска подстрок

Пусть даны "текст" $T[1..n]$ – строка символов длины n и "образец" $P[1..m]$ – строка символов длины m, причем образец короче текста ($m \le n$). Считается, что символы, входящие в T и P, берутся из некоторого конечного алфавита Σ (например, алфавит состоит из латинских букв $\Sigma=\{a,b,\ldots,z\}$).

Говорят, что образец P входит со сдвигом s в текст T, если подстрока текста T длины m, которая начинается с позиции $(s+1)$, совпадает со строкой P (то есть $T[s+1..s+m] = P[1..m]$). Естественно, сдвиг s не может быть больше $n–m$. Если P входит со сдвигом s в текст T, то говорят, что s – *допустимый сдвиг*. Если подстрока текста T начиная с позиции $s+1$ не совпадает с образцом ($T[s+1..s+m] \neq P[1..m]$), то говорят, что s – *недопустимый сдвиг*. Задача поиска подстрок состоит в нахождении всех допустимых сдвигов для заданных текста T и образца P.

Например, образец P = **aba** входит в текст T = **cababababcaba**. с позиций 2, 4 и 9. Соответственно, допустимые сдвиги: 1, 3 и 8.

2.3. Постановка задачи распознавания подстрок

Пусть задан текст T и набор образцов $P_1, P_2,\ldots P_Z$. Требуется найти все подстроки текста T, совпадающие с образцами, и установить, с каким именно образцом совпадает найденная подстрока.

Задачи поиска и распознавания подстрок встречаются:

- в трансляторах, компиляторах, командных процессорах и других программах, где нужно выделять и распознавать слова из текста;
- в криптографии – при шифровании и дешифровании данных;
- в программах сжатия информации (архиваторах и им подобных).

3. МЕТОДЫ ПОИСКА ПОДСТРОК

3.1. Простейший алгоритм

Простейший алгоритм для поиска образца P в тексте T последовательно проверяет равенство $T[s+1..s+m] = P[1..m]$ для каждого из возможных значений сдвига s от 0 до $n–m$. То есть мы двигаем образец вдоль текста и проверяем все его положения.

Простейший алгоритм самый медленный. Время его работы в худшем случае есть $\Theta((n–m+1)m)$ [1]. Его неэффективность связана с тем, что информация о тексте T, получаемая при проверке очередного сдвига s, никак не используется при проверке последующих сдвигов. Например, образец P = **aaab,** и мы выяснили, что сдвиг $s = 0$ допустим. Тогда сдвиги 1, 2 и 3 заведомо недопустимы, поскольку $T[4]$ = **b**. Описанные далее методы поиска подстрок используют эту идею для сокращения числа проверок.

3.2. Поиск подстрок с помощью конечных автоматов

Поиск подстроки с помощью конечного автомата Мура весьма эффективен: каждый символ поступает на вход автомата только единожды, так что общее время работы $\Theta(n)$. Однако алгоритм требует предварительной подготовки – построения конечного автомата. Время, затраченное на построение автомата, может быть весьма значительным, особенно если велик алфавит Σ.

Дадим формальное определение конечного автомата. Конечный автомат представляет собой пятерку объектов $M = (\Sigma, Q, q_0, F, \delta)$, где:

- Σ – конечный входной алфавит;
- Q – конечное множество состояний;
- q_0 – начальное состояние автомата;
- F – подмножество выходных (допускающих) состояний, $F \subset Q$;
- δ – функция переходов.

---

[1] случай, когда образец и текст состоят из повторений одного символа, например $P = \mathbf{a}^m$, $T = \mathbf{a}^n$. Тогда для каждого из $n–m+1$ значений сдвига ($s = 0…n–m$) будет выполнено m сравнений символов.

Первоначально автомат находится в начальном состоянии q_0; затем он по очереди читает символы входной строки. Находясь в состоянии q_i и читая символ α, автомат переходит в состояние q_j. В какое именно состояние перейдет автомат под действием прочитанного символа, определяется функцией переходов δ.

Состояния $q_p,..q_{p+z}$, входящие в подмножество выходных состояний F, соответствуют окончанию распознавания образцов $P_1 \ldots P_Z$. Когда автомат находится в выходном состоянии q_{p+i}, это означает, что найдено вхождение образца P_i в текст со сдвигом $s = k–m_i$, где k – номер последнего считанного символа входной строки, m_i – длина i-го образца.

Конечный автомат для распознавания единственного образца P [1..m], имеет m+1 состояние Q = {0,1,…m}. Состояние с номером 0 является начальным, с номером m – выходным (допускающим).

Прежде чем перейти к созданию функции переходов, необходимо дать определение суффикс-функции.

Суффикс-функция $\sigma(x)$ ставит в соответствие строке x длину максимального суффикса x, являющегося префиксом образца P. Суффикс-функция принимает значения целых чисел от 0 до m. Например, для образца P = **ab** $\sigma(\varepsilon)=0$, $\sigma(\mathbf{a})=1$, $\sigma(\mathbf{cda})=1$, $\sigma(\mathbf{ccdab})=2$, $\sigma(\mathbf{ccdabc})=0$.

Функция переходов $\delta(q_i,\alpha)$ показывает номер состояния, в которое перейдет конечный автомат из состояния q_i под действием символа α. Она равна

$$\delta(q_i, \alpha) = \sigma(P_i \alpha)$$

здесь $P_i \alpha$ обозначает конкатенацию префикса P длины i с символом α.

Пример 1

Построить конечный автомат для поиска образца P = **ababaca**. Входной алфавит конечного автомата ограничен тремя символами: Σ={**a**,**b**,**c**}. Длина образца |P|=7, поэтому конечный автомат будет иметь восемь состояний с номерами от 0 до 7. Значения суффикс-функции σ приведены в таблице:

i	Префикс P_i	Значение $\sigma(P_i\,\alpha)$ для входных символов		
		a	**b**	**c**
0	$P_0 = \varepsilon$	1	0	0
1	P_1= **a**	1	2	0
2	P_2= **ab**	3	0	0
3	P_3= **aba**	1	4	0
4	P_4= **abab**	5	0	0
5	P_5= **ababa**	1	4	6
6	P_6= **ababac**	7	0	0
7	P_7 = **abababa**	1	2	0

Рассмотрим, как вычисляется суффикс-функция на примере 3-й строки таблицы. Префикс образца *P* длиной 3 равен P_3 =**aba**. Для каждого символа из алфавита Σ={**a**,**b**,**c**} получаем конкатенацию подстроки P_3 с этим символом:

с символом **a**: P_3**a** =**abaa**
с символом **b**: P_3**b** =**abab**
с символом **c**: P_3**c** =**abac**

Затем, для каждой конкатенации $P_3\alpha$ (здесь α обозначает символ из алфавита) ищем суффикс $P_3\alpha$ максимальной длины, одновременно являющийся префиксом образца *P*. Его длина и есть значение суффикс-функции $\sigma(P_3\,\alpha)$.

У строки P_3**a** =**aba**__**a**__ и образца *P* = __**a**__**bababa** общая часть – это один символ **a** (здесь и далее общие части подчеркнуты), следовательно, значение суффикс-функции $\sigma(P_3$**a**$)=1$.

У строки P_3**b** =**abab** есть два суффикса, являющиеся префиксами образца *P*. Это **ab** (P_3**b** = **ab**__**ab**__, *P* = __**ab**__**abaca**) и **abab** (P_3**b** = __**abab**__, *P* = __**abab**__**aca**). Выбираем суффикс максимальной длины (4), значение суффикс-функции $\sigma(P_3$**b**$)=4$.

У строки P_3**c** = **abac** нет суффиксов, совпадающих с префиксами образца *P*, поэтому суффикс-функция $\sigma(P_3$**c**$)=0$.

Таблица на с.8 является таблицей переходов конечного автомата: i – номер исходного состояния q_i, а значение суффикс-функции – номер целевого состояния q_j. Если из этой таблицы исключить столбец P_i, получим привычную форму записи таблицы переходов автомата:

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

Диаграмма переходов конечного автомата, построенная по таблице, показана на рис.1. Выходное состояние автомата 7 обведено двойным кружком. На диаграмме не показаны стрелки, ведущие в 0-е состояние. Если из состояния i не выходит стрелки, помеченной символом α, то подразумевается, что $\delta(i,\alpha) = 0$.

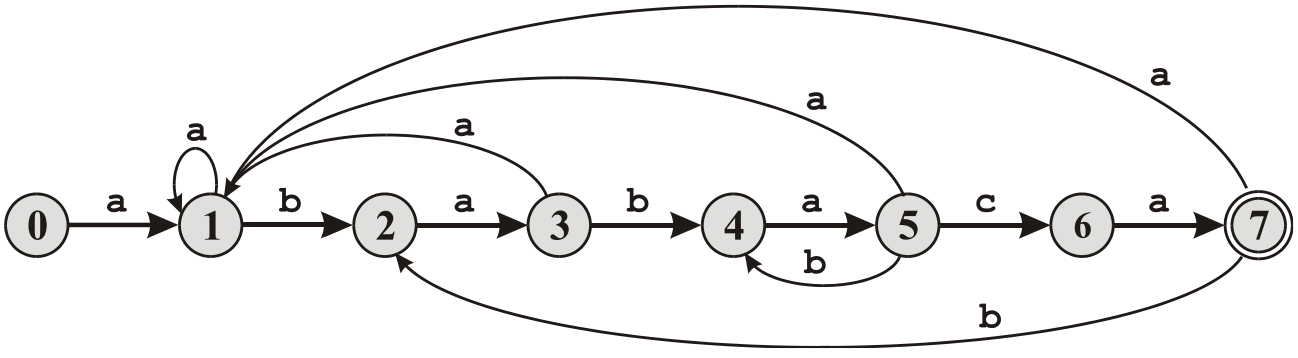

Рис. 1. Диаграмма переходов для конечного автомата, допускающего строки, оканчивающиеся на **ababaca**.

Функцию переходов конечного автомата можно находить вручную, как сделано в примере 1, а можно поручить это ЭВМ, то есть функцию переходов строить автоматически.

Простейший алгоритм автоматического построения суффикс-функции приведен в листинге 1.

```
1   m ← length[P]
2  for q ← 0  to  m
3     do  for (для) всех символов α∈Σ
4              do  k  ← min( m+1, q+2 )
5                  repeat  k ← k – 1
6                  until   Pk ⊐ Pq α
7                  δ(q,α) ← k
8  return δ
```

Листинг 1. Алгоритм автоматического построения суффикс-функции.

Он заключается в переборе всех пар (i, α), где i – номер состояния конечного автомата, i = 0, 1,…m, m – длина образца, α – символ входного алфавита Σ. Время работы этого алгоритма оценивается как Θ(m3·|Σ|), где |Σ| – количество символов в алфавите.

3.3. Распознавание подстрок с помощью конечных автоматов

Задача распознавания подстрок состоит в поиске всех вхождений в текст T заданных образцов P_1, P_2,…P_z. Длина образцов может быть различной. Простейший способ решения задачи – построить z конечных автоматов, по одному на каждый образец, и "пропустить" текст последовательно через эти автоматы. Время работы такого распознавателя пропорционально количеству образцов z и длине текста n $\Theta(z \cdot n)$. Если учесть время на автоматическое построение функций переходов, то временные затраты весьма значительны.

Если набор строк-образцов таков, что ни один из образцов не является подстрокой другого образца, то можно создать один конечный автомат для распознавания всех образцов. Текст подается на автомат

один раз, и время работы прямо пропорционально длине текста $\Theta(n)$. Рассмотрим этапы построения такого автомата.

Конечный автомат для распознавания набора образцов $P_1, P_2, \ldots P_z$ имеет одно начальное состояние и z выходных состояний $q_1, q_2 \ldots q_z$. Состояние q_i соответствует окончанию распознавания i-го образца с допустимым сдвигом $s = k - m_i$, где k – порядковый номер последнего считанного из текста символа, m_i – длина i-го образца.

Промежуточные состояния автомата создаются таким образом, что нахождение автомата в этих состояниях соответствует распознаванию какого-либо из префиксов образцов. Каждый символ образца дает новое состояние автомата. Если у нескольких образцов есть общий префикс длиной d символов, то первые d состояний автомата (по длине префикса) у них общие.

Обозначим через $P^{(j)}$ подстроку, которая распознана в состоянии автомата с номером j. Подстрока $P^{(j)}$ является префиксом одного или нескольких образцов. Функция перехода $\delta(j,\alpha)$ из состояния j под действием входного символа α равна номеру q того состояния, для которого подстрока $P^{(q)}$ является суффиксом максимальной длины для строки $P^{(j)}\alpha$.

Рассмотрим пример построения конечного автомата для распознавания нескольких подстрок.

Пример 2

Построить конечный автомат для распознавания трех образцов: P_1 = **aab**, P_2 = **abc**, P_3 = **ccba**. Условие о том, что ни один образец не является подстрокой другого образца, выполняется. Входной алфавит автомата включает три символа Σ={**a**,**b**,**c**}.

Первым этапом будет создание каркаса диаграммы переходов конечного автомата (рис.2). Состояние 0 является начальным; три состояния (по числу образцов) являются выходными: состояние 3 соответствует обнаружению подстроки **aab**, состояние 5 – подстроки **abc**, и состояние 9 – подстроки **ccba**.

Второй этап – построение таблицы переходов конечного автомата. Сначала в таблице переходов заполняются ячейки, образующие каркас

диаграммы переходов (эти ячейки в таблице на с.14 выделены серым). Для заполнения остальных ячеек нужно искать суффикс-функцию.

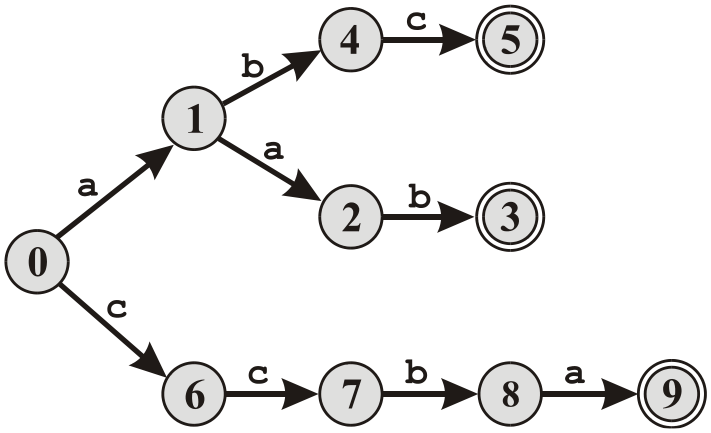

Рис.2. Каркас диаграммы переходов для распознавания подстрок P_1=**aab**, P_2=**abc**, P_3=**ccba**. Каждый переход приближает автомат к выходному состоянию. Образцы P_1 и P_2 имеют общий префикс **a**, поэтому переход $0 \rightarrow 1$ для них общий.

Функция переходов $\delta(q_i,\alpha)$ из состояния q под действием символа α равна значению суффикс-функции

$$\delta(q_i, \alpha) = \sigma(\mathrm{P}^{(j)}\alpha).$$

Строка $P^{(j)}$ определяется для каждого состояния j автомата. Она получается конкатенацией символов, записанных над стрелками переходов, при движении из начального состояния автомата в состояние с номером j по каркасу диаграммы переходов:

$\mathrm{P}^{(0)} = \varepsilon$
$\mathrm{P}^{(1)} =$ **a**
$\mathrm{P}^{(2)} =$ **aa**
$\mathrm{P}^{(3)} =$ **aab**
$\mathrm{P}^{(4)} =$ **ab**
$\mathrm{P}^{(5)} =$ **abc**
$\mathrm{P}^{(6)} =$ **c**
$\mathrm{P}^{(7)} =$ **cc**
$\mathrm{P}^{(8)} =$ **ccb**
$\mathrm{P}^{(9)} =$ **ccba**

Суффикс-функция $\sigma(\mathrm{P}^{(j)}\alpha)$ равна номеру того состояния q, у которого строка $P^{(q)}$ является суффиксом максимальной длины для

строки $P^{(j)}\alpha$ среди всех подстрок $P^{(i)}$, $i = 0,1,2,\ldots$ Поясним это на примере заполнения 3-й строки таблицы ($j = 3$). Строка $P^{(3)}$= **aab**.

Найдем $\sigma(P^{(3)}\mathbf{a})$. Конкатенация символа **a** и $P^{(3)}$ дает $P^{(3)}\mathbf{a}$=**aaba**. Ищем строки $P^{(i)}$, у которых первые символы совпадают с последними символами $P^{(3)}\mathbf{a}$. Такая строка одна – это $P^{(1)}$=**a** общая часть с $P^{(3)}\mathbf{a}$=**aab**__**a**__ подчеркнута, следовательно $\delta(3, \mathbf{a})=1$.

Найдем $\sigma(P^{(3)}\mathbf{b})$. Конкатенация символа **b** и $P^{(3)}$ дает $P^{(3)}\mathbf{b}$=**aabb**. Ищем строки $P^{(i)}$, суффикс которых совпадает с префиксом $P^{(3)}\mathbf{b}$. Таких строк нет, следовательно $\delta(3, \mathbf{b})=0$.

Найдем $\sigma(P^{(3)}\mathbf{c})$. Конкатенация символа **c** и $P^{(3)}$ дает $P^{(3)}\mathbf{c}$=**aabc**. Есть две строки, у которых первые символы совпадают с последними символами $P^{(3)}\mathbf{c}$ – это $P^{(5)}$ и $P^{(6)}$:

$P^{(5)}$=**abc** – общая часть длиной 3 ($P^{(3)}\mathbf{c}$=**aabc**);

$P^{(6)}$=**c** – общая часть длиной 1 ($P^{(3)}\mathbf{c}$=**aab**__**c**__).

Выбираем подстроку с максимальной длиной (3), номер соответствующего состояния 5, следовательно $\delta(3,\mathbf{c}) = 5$

j	Строка $P^{(j)}$	Значение $\sigma(P^{(j)}\alpha)$ для входных символов		
		a	**b**	**c**
0	$P^{(0)} = \varepsilon$	**1**	**0**	**6**
1	$P^{(1)}$= **a**	**2**	**4**	**6**
2	$P^{(2)}$= **aa**	**2**	**3**	**6**
***3**	$P^{(3)}$= **aab**	**1**	**0**	**5**
4	$P^{(4)}$= **ab**	**1**	**0**	**5**
***5**	$P^{(5)}$= **abc**	**1**	**0**	**7**
6	$P^{(6)}$= **c**	**1**	**0**	**7**
7	$P^{(7)}$= **cc**	**1**	**8**	**7**
8	$P^{(8)}$= **ccb**	**9**	**0**	**6**
***9**	$P^{(9)}$= **ccba**	**2**	**4**	**6**

Эта таблица является таблицей переходов конечного автомата: i – номер исходного состояния q_i, а значение суффикс-функции – номер целевого состояния q_j. Выходные состояния помечены звездочками.

Диаграмма переходов, построенная по таблице переходов конечного автомата, изображена на рис. 3.

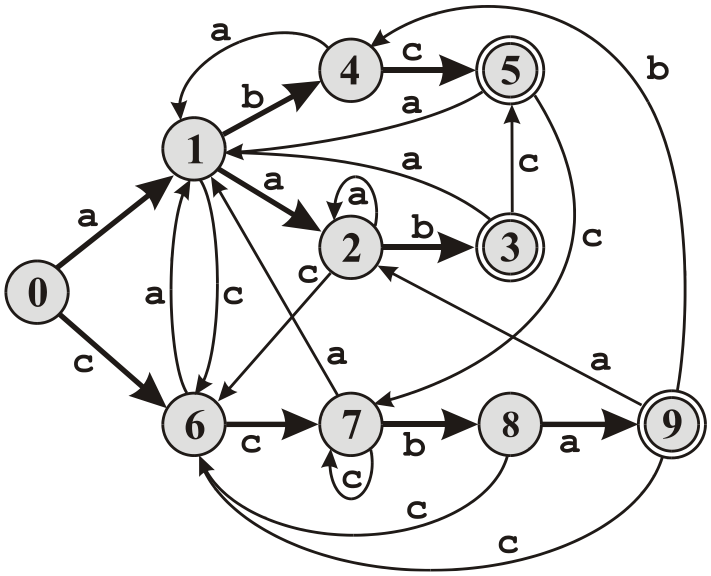

Рис. 3. Диаграмма переходов конечного автомата для распознавания подстрок P_1=**aab**, P_2=**abc**, P_3=**ccba**. На диаграмме не показаны стрелки, ведущие в 0-е состояние.

Результат применения автомата к тексту T = **aaabccbabc** иллюстрирует рис. 4. Под каждым символом $T[k]$ записано состояние автомата после прочтения этого символа. Найдено вхождение образца P_1 со сдвигом 1, образца P_2 со сдвигами 2 и 7, и образца P_3 со сдвигом 4.

Номер символа *k*:	1	2	3	4	5	6	7	8	9	10
T[*k*]:	a	a	a	b	c	c	b	a	b	c
Состояние автомата *j*:	0	1	2	2	**3**	**5**	7	8	**9**	4	**5**

вхождение P_2= **abc** со сдвигом s=7

вхождение P_3= **ccba** со сдвигом s=4

вхождение P_2= **abc** со сдвигом s=2

вхождение P_1= **aab** со сдвигом s=k-|P_1|=4-3=1

Рис. 4. Поиск образцов P_1=**aab**, P_2=**abc**, P_3=**ccba** в тексте T = **aaabccbabc** с помощью конечного автомата.

3.4. Программная реализация конечных автоматов

Структурный синтез при программной реализации конечных автоматов заключается в кодировании входных символов и состояний.

Представление входных символов

В большинстве случаев текст, который будут обрабатывать на конечном автомате, находится в одной из стандартных кодировок: ASCII, ANSI или Unicode. Перед подачей на автомат текст необходимо перекодировать по следующим причинам.

Причина первая – чувствительность автомата к регистру букв. Коды заглавных и строчных букв различны, поэтому конечный автомат будет воспринимать большие и маленькие буквы как разные символы. Чтобы автомат не делал различий между заглавными и строчными буквами, можно:

1) перекодировать текст перед подачей на вход автомата, приводя буквы к одному регистру (например, к заглавным). Тогда входной алфавит автомата будет включать только заглавные буквы.

2) другой вариант – буквы не менять. Входной алфавит будет включать и заглавные, и строчные буквы. Таблица переходов составляется таким образом, чтобы переход по большой и маленькой одноименной букве был одинаков. При этом варианте таблица переходов будет занимать больше места в памяти по сравнению с вариантом 1.

Вторая причина – недопустимые входные символы. Кодировки ASCII и ANSI состоят из 256 различных кодов, Unicode – из 65536. Входной алфавит конечного автомата обычно гораздо меньше. Если на вход автомата может попасть символ, который не известен автомату, для предотвращения неопределенной реакции автомата, можно:

1) перед подачей на автомат проверять на "допустимость" каждый символ текста; если найден символ, не входящий в автоматный алфавит, работа программы прекращается;

2) добавить в автоматный алфавит новый символ, означающий "прочее", и ввести новое состояние автомата "ошибка", в которое автомат будет переходить под действием этого символа. Перед подачей символов на автомат следует заменять все недопустимые символы текста на код "прочее".

Третья причина связана с программной реализацией выбора переходов автомата. Если переходы делаются методом векторов переходов или таблиц переходов (см. ниже), то код входного символа используется как индекс в массиве. А поскольку коды символов в стандартной кодировке обычно не следуют подряд друг за другом, а "разбросаны" по всей кодовой странице, то использовать их в качестве индекса нельзя, требуется перекодировка.

Представление состояний

Есть два способа, с помощью которых программа контролирует текущее состояние конечного автомата:

- номер текущего состояния запоминается в специальной переменной (*явный способ*);
- номер состояния не запоминается, а каждому состоянию соответствует свой участок программы (*неявный способ*). "Переключение" между состояниями выполняется путем перехода на соответствующий участок программы.

Выбор переходов

При *неявном* способе выбор переходов выполняется командами условного перехода, или *switch*-подобными операторами после анализа кода входного символа.

При *явном* способе наиболее эффективны два метода:

1. ***Метод вектора переходов***. Этот метод выгодно применять там, где в большинстве состояний автомат выполняет какие-либо действия, причем различные[2]. Номер текущего состояния запоминается в специальной переменной. Каждому состоянию соответствует участок программы. Таблица переходов конечного автомата оформляется в программе в виде двумерного массива. Каждая строка массива соответствует номеру состояния, а столбец – входному символу алфавита. В элементах массива записываются адреса меток программы, соответствующих нужному состоянию.

[2] конечные автоматы из примеров 1 и 2 действия выполняют только в выходных состояниях: выводят сообщения об успешном распознавании образца или печатают допустимый сдвиг. Поэтому использовать данный метод для примеров 1 и 2 не эффективно (см. листинг 2).

```
1  Procedure encode( in A, out B ) // Подпрограмма перекодировки
2   switch( A ) of
3        'a' :   B ← 0
4        'b' :   B ← 1
5        'c' :   B ← 2
6        else: B ← 3
7   end switch
8   return B
//  Подпрограммы, представляющие состояния автомата:…
// …содержат действия, выполняемые автоматом в этом состоянии
9   Procedure state_0   // состояние 0 (действий нет)
10     s ← 0              // s – номер текущего состояния автомата
11     return
12  Procedure state_1   // состояние 1 (действий нет)
13     s ← 1
14     return
15  Procedure state_3   // состояние 3 (выходное)
16     s ← 3
17     print "Вхождение образца P1 со сдвигом" k–length[P1]
18     return
… // остальные процедуры state_2, sate_4 … state_10 аналогично
главная программа:
// таблица векторов перехода, символ & означает взятие адреса
19  jump_vectors : array[0..9, 0..3]= (( &state_1, &state_0, &state_6),
                                       ( &state_2, &state_4, &state_6), и т.д. )
20  n ← length[T]        // T – входной текст,  n – его длина
21  call state_0 ;
22  for k ← 1  to  n do
23       call encode( T[k], c )   // c – код символа после перекодировки
24       if c≠3
25         then  call  jump_vectors[ s, c ] // вызов процедуры по адресу
26         else   call error
27  end for
```

Листинг 2. Алгоритм программной реализации конечного автомата из примера 2 методом векторов перехода.

В листинге 2 номер состояния автомата хранится в глобальной переменной *s*. Процедура *encode* перекодирует символы текста *T* перед подачей на автомат: **a**→0, **b**→1, **c**→2, прочие символы в код 3. При обнаружении “недопустимого” входного символа с кодом 3 автомат вызывает процедуру *error*.

2. ***Метод таблицы переходов***. Этот метод применяется в автоматах, выполняющих однотипные действия в нескольких состояниях. От предыдущего метода отличается тем, что хранит в массиве номера́ состояний. Каждая строка массива представляет состояние *j*, а столбец – входной символ алфавита. Номер следующего состояния выбирается по координатам (*Номер_Текущего_Состояния, Код_Символа*).

В листинге 3 приведен алгоритм работы конечного автомата из примера 2 методом таблицы переходов. Номер текущего состояния хранится в переменной *s*. Процедура *encode* перекодирует символы текста *T* перед подачей на автомат: **a**→0, **b**→1, **c**→2, прочие в код 3 (реализацию процедуры *encode* см. в листинге 2, строки 1–8) . Вводится новое состояние автомата с номером 10, в которое переходит автомат при поступлении “недопустимого” входного символа с кодом 3. Номера состояний переходов хранятся в таблице *jump_table*.

```
 1  jump_table : array[0..10,0..3] = ((1,0,6,10), (2,4,6,10), (2,3,6,10), (1,0,5,10), (1,0,5,10),
                                      (1,0,7,10), (1,0,7,10), (1,8,7,10),(9,0,6,10), (2,4,6,10), (1,0,6,10))
 2  n ← length[T]          // T – входной текст
 3  s ← 0                  // s – номер текущего состояния автомата
 4  for k ← 1 to n do
 5      call encode( T[k], c )   // c – код символа после перекодировки
 6      s ← jump_table[ s, c ]
// действия выполняются только в выходных состояниях 3,5,9,10
 7      switch( s ) of
 8          3: print "Вхождение образца P1 со сдвигом" k–length[P1]
 9          5: print "Вхождение образца P2 со сдвигом" k–length[P2]
10          9: print "Вхождение образца P3 со сдвигом" k–length[P3]
11         10: print "Недопустимый символ" T[k]
12        end switch
13  end for
```

Листинг 3. Алгоритм программной реализации конечного автомата из примера 2 методом таблицы переходов

3.5. Алгоритм Кнута-Морриса-Пратта

Алгоритм Кнута-Морриса-Пратта (КМП), как и метод конечных автоматов, ищет подстроки за время $\Theta(n)$, где n – длина текста T. Его преимуществом является значительно меньшие затраты времени на подготовительные операции. Так, на автоматическое построение функции переходов конечного автомата необходимо $\Theta(m^3 \cdot |\Sigma|)$ единиц времени, а на построение вспомогательной префикс-функции в алгоритме КМП – только $\Theta(m)$. Здесь m – длина образца P.

Алгоритм основан на идее: предположим, что при поиске подстрок простейшим алгоритмом для некоторого сдвига s оказалось, что первые q символов образца совпадают с символами текста, а в следующем символе имеется расхождение: то есть $P[1..q] = T[s+1..s+q]$ и $P[q+1] \neq T[s+q+1]$, $q < m$ (рис.5-а). Поскольку мы знаем q символов текста, от $T[s+1]$ до $T[s+q]$, из этой информации мы можем заключить, что некоторые последующие сдвиги будут заведомо недопустимы. В примере на рис.5-а видно, что сдвиг $(s+1)$ недопустим, поскольку при этом сдвиге первый символ образца (буква **a**) окажется напротив (s+2)-го символа текста (буквы **b**). При сдвиге $(s+2)$ первые три символа образца (**aba**) совпадут с тремя последними из известных нам символов текста (рис.5-б), и есть шанс, что последующие символы текста совпадут с образцом. Дальнейшее сравнение символов образца и текста можно продолжать с $(s+q+1)$-го символа текста (в примере на рис.4 это буква **b**).

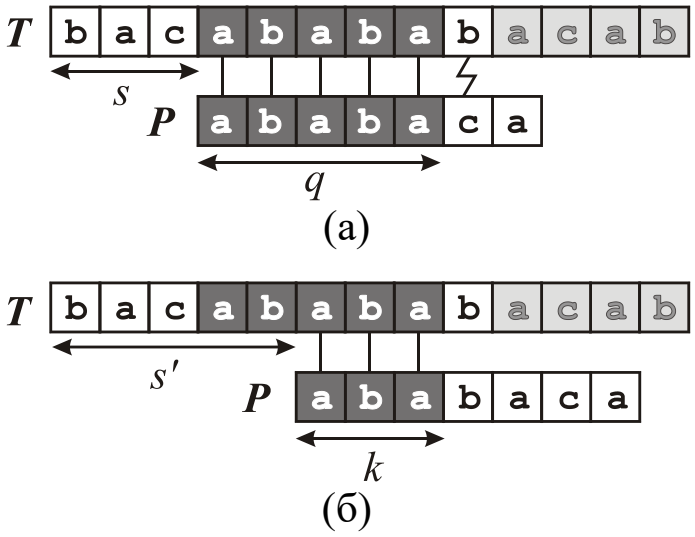

Рис. 5. К понятию префикс-функции. Светло-серым показаны еще не прочитанные символы текста.

Алгоритм КМП заключается в том, что при совпадении первых q символов образца и текста следующий проверяемый сдвиг $s' = s+(q–k)$, где k – длина максимального префикса образца P, который является суффиксом просмотренного участка текста (рис.5-б). Если такого префикса не существует, то поиск подстроки продолжается за границей просмотренного участка ($s' = s+q$) [3]. Чтобы найти число k, нам не нужно ничего знать о тексте T: достаточно знания образца P и числа q.

Обозначим через P_q первые q символов образца P. Число k – это длина наибольшего префикса P_q, являющегося (собственным) суффиксом P_q . Для примера, если P_q = **ababa**, то подстроки **a** и **aba** являются одновременно префиксом и суффиксом, максимальная из них имеет длину k = 3. Число k является значением *префикс-функции* $\pi(q)$.

Формальное определение префикс-функции звучит так: префикс-функцией, ассоциированной со строкой P [1..m], называется функция $\pi(q)$, q=1,2,…m, значением которой являются число от 0 до m–1, определенное как длина наибольшего префикса P_q, являющегося собственным суффиксом P_q:

$$\pi(q) = \max\{\ k:\ k < q \ \text{ и } \ P_k \sqsupset P_q\ \}$$

Префикс-функция для образца P = **abababaca** приведена в таблице:

q	P_q	Максимальная подстрока P_k	$\pi(q)$
1	P_1= **a**	–	0
2	P_2= **ab**	–	0
3	P_3= **aba**	**a**	1
4	P_4= **abab**	**ab**	2
5	P_5= **ababa**	**aba**	3
6	P_6= **ababac**	–	0
7	P_7 = **abababaca**	**a**	1

[3] при несовпадении первого символа образца (q = 0), сдвиг $s' = s+1$.

Алгоритм поиска строки методом КМП приведен в листинге 4. Поэтапная работа алгоритма КМП для образца P = **ababaca** показана на рис.6.

Рис. 6. Поиск образца в тексте алгоритмом Кнута-Морриса-Пратта

```
1  prefix_function : array[1..7] = ( 0, 0, 1, 2, 3, 0, 1 )
2  n ← length[T]                  // T [1…n] – входной текст
3  m ← length[P]                  // P [1…m] – образец
4  for s ← 0  to  n–m  do         // s – номер текущего сдвига
5     q ← 0          // q – кол-во совпавших символов образца и текста
6     while  T [s+q+1] = P [q+1]   do
8        q ← q + 1
9        if  q = m
10          then print "Вхождение образца P со сдвигом" s
11       if  s+q = n
12          then exit              // текст закончился
13    end while
14    if  q ≠ 0
15       then  s ← s + q – prefix_function[q]
16 end for
```

Листинг 4. Алгоритм Кнута-Морриса-Пратта для примера из рис.6

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить вариант задания и значения строк-образцов.
2. Изучить теоретическую часть методических указаний.
3. Выбрать метод для решения задачи (метод конечных автоматов или Кнута-Морриса-Пратта). При решении задачи методом конечных автоматов:
   3.1. Определить входной алфавит автомата.
   3.2. Составить каркас диаграммы переходов автомата. По каркасу определить количество состояний автомата.
   3.3. Вычислить значения суффикс-функции.
   3.4. Найти функцию переходов конечного автомата.
   3.5. Разработать систему кодирования входных символов.
   3.6. Разработать алгоритм решения задачи.

   При решении задачи методом КМП:
   3.1. Определить входной алфавит автомата.
   3.2. Рассчитать значения префикс-функции.
   3.3. Разработать алгоритм решения задачи.
4. Разработать программу. Оформить отчет.

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Вариант задания.

2. Диаграмма переходов конечного автомата и таблица переходов автомата (значения суффикс-функции) – если применяется метод конечных автоматов; таблица со значениями префикс-функции – если применяется алгоритм Кнута-Морриса-Пратта.

3. Алгоритм решения задачи в виде блок-схемы или словесного описания по пунктам.

4. Текст программы.

5. Тестовый пример и результаты работы программы.

6. ВАРИАНТЫ ЗАДАНИЙ

1. Найти все вхождения образца P в текст.

2. Проверить, есть перекрывающиеся вхождения образца $P[1..m]$ в текст (расстояние между соседними допустимыми сдвигами меньше m).

3. Найти минимальное расстояние между соседними допустимыми сдвигами, вывести значения сдвигов и расстояние между ними.

4. Известно, что все вхождения образца P в текст не перекрываются. Вывести значения допустимых сдвигов и нераспознанные подстроки текста.

5. Из текста исключить все вхождения образца P, в том числе перекрывающиеся.

6. Задан набор образцов $P_1 \ldots P_Z$, причем ни один из образцов не является подстрокой другого образца. Найти все вхождения образцов в текст T.

7. Задан набор образцов $P_1 \ldots P_Z$, причем ни один из образцов не является подстрокой другого образца. Известно, что все вхождения образцов в текст не перекрываются. Преобразовать текст: подстроки текста, совпадающие с образцом, заменить на пару чисел (*номер образца*; *допустимый сдвиг*), нераспознанные подстроки текста выводить без изменений.

8. Задан набор образцов $P_1 \ldots P_Z$, причем ни один из образцов не является подстрокой другого образца. Проверить, существуют ли в тексте подстроки, не совпадающие ни с одним из образцов.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение терминам: строка, суффикс, префикс, подстрока.
2. Перечислите основные операции над строками.
3. Сформулируйте задачу поиска подстрок.
4. Сформулируйте задачу распознавания подстрок.
5. Что означает фраза: "вхождение строки A в текст T со сдвигом 4".
6. В чем состоит метод поиска подстроки с помощью конечного автомата?
7. Что такое суффикс-функция? Как она вычисляется?
8. Этапы построения конечного автомата для распознавания нескольких образцов.
9. В чем состоит алгоритм Кнута-Морриса-Пратта?
10. Что такое префикс функция? Как она вычисляется?
11. Дайте сравнительную оценку трудоемкости поиска строки простейшим алгоритмом, методом конечных автоматов и алгоритмом КМП (при условии, что таблица переходов и префикс-функция построены заранее)?
12. Дайте сравнительную оценку трудоемкости поиска строки простейшим алгоритмом, методом конечных автоматов и алгоритмом КМП (при условии, что таблица переходов и префикс-функция заранее не известны и их нужно строить автоматически)?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Чернецкая, И. Е. Теория автоматов [Текст]: учебное пособие / И. Е. Чернецкая; МИНОБРНАУКИ РОССИИ, Юго-Западный государственный университет. - Курск: ЮЗГУ, 2011. - 143 с.
2. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М.: МНЦМО, 2001. – 960 с.
3. Кнут Д. Искусство программирования. В 3-х томах. Т.3. Сортировка и поиск. – М: Вильямс, 2003. – С.527, с.611
4. Вирт Н. Алгоритмы и структуры данных. – М.: Мир, 1989. – 360 с.