**Порождающей грамматикой** называется четверка

$G = (\Sigma, N, P, S)$, где

$\Sigma$ – алфавит терминальных символов;

$N$ – алфавит нетерминальных символов $\Sigma \cap N = \varnothing$;

$P$ – множество правил вида $\alpha \to \beta$, $\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$, $\beta \in (N \cup \Sigma)^*$;

$S$ – начальный символ ($S \in N$).

**Пример** $G = (\{a,b,c\}, \{A,B,S\}, P, S)$,

$P = \{S \to AB, A \to a, A \to ac, B \to b, B \to cb\}$.

(1) $S \to AB \to aB \to ab$

(2) $S \to AB \to aB \to acb$

(3) $S \to AB \to acB \to acb$

(4) $S \to AB \to acB \to accb$

$L(G) = \{ab, acb, accb\}$.

## Конечные автоматы – средство распознавания

*Детерминированный конечный автомат* – это пятерка

M = ($S$, $\Sigma$, $\delta$, $s_0$, $F$), где

S – конечное множество состояний;

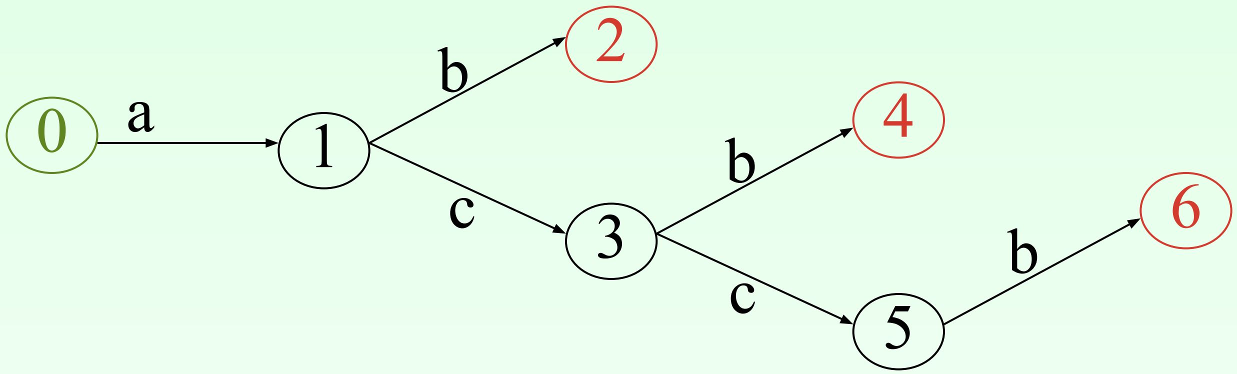
$\Sigma$ – алфавит;

$\delta : S \times \Sigma \rightarrow S$ – функция переходов;

$s_0 \in S$ – выделенное начальное состояние;

$F \subseteq S$ – множество заключительных состояний;

ДКА, допускающий {ab, acb, accb}.



# Формальные последовательности

**Последовательность Туэ - Морса**

Способы задания

1. Итерации морфизмов.
   $\Sigma = \{a_1 \ldots a_q\}$ $\phi : \Sigma^* \to \Sigma^*$ – *морфизм*, если $\phi(XY) = \phi(X)\phi(Y)$ $\forall$ слов X и Y.
   $\phi = \{0 \to 01, 1 \to 10\}$.
   $X_0 = 0, X_1 = 01, X_2 = 0110, X_3 = 01101001, X_4 = 0110100110010110 \ldots$

2. $X[i] = 0$, если число единиц в двоичной записи числа $i$ чётно,
   $X[i] = 1$, в противном случае.

3. Итеративный способ: $X[0] = 0$, $X[2i] = X[i]$, $X[2i+1] = ((X[i] + 1) \bmod 2$
4. Индуктивной схемой : $X_0 = 0$, $X_n = X_{n-1} \overline{X_{n-1}}$
   где $\overline{X_{n-1}}$ – *отрицание слова* $X_{n-1}$, которое получается из $X_{n-1}$ заменой всех нулей на единицы, а всех единиц на нули.

Свойства последовательности Туэ-Морса:

1. Отсутствуют подслова вида VVV.
2. $X_{2n} = X_n X_n^R$ : слово, полученное на чётном шаге является палиндромом.

# Формальные последовательности

**Чи́сла Фибона́ччи** — 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, …

**Последовательность Фибоначчи**

$X_0 = 0, X_1 = 01, \; X_n = X_{n-1}X_{n-2}$

$X_2 = 01.0, \; X_3 = 010.01, \; X_4 = 01001.010, X_5 = 01001010.01001$

Морфизм: $\phi = \{0 \rightarrow 01, 1 \rightarrow 0\}$

# Алгоритмы поиска точно заданных образцов

**Дано**: $P = p_1 p_2 \ldots p_m$ **– образец,** $T = t_1\ t_2 \ldots t_N$ **– текст,**
$t_i \in \Sigma,\ p_j \in \Sigma,\ 1 \le i \le N,\ 1 \le j \le m,\ m < N.$

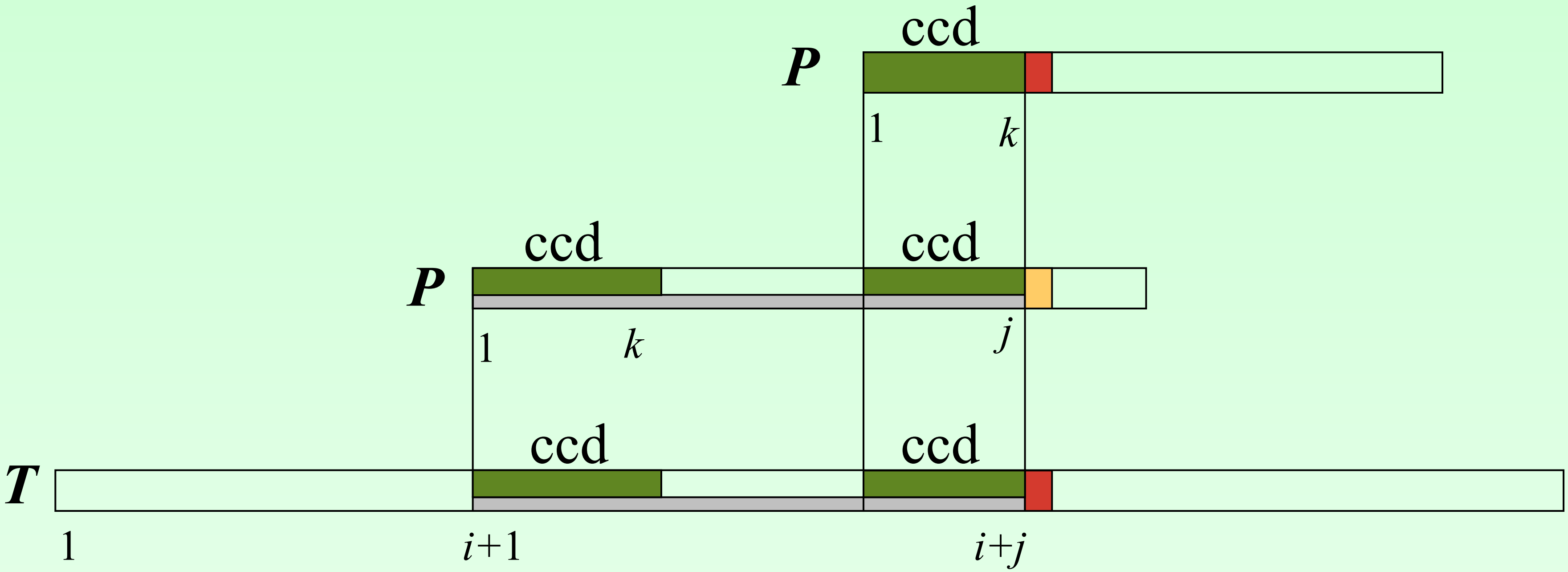
**Задача: обнаружить все вхождения образца** $P$ **в текст** $T$.

**Пример.** $\Sigma = \{a,b,c,d\}$, $P =$ **aba**, $T =$ **bbabacabcabad;** $m = 3$, $N = 13$.

Образец входит в текст в 3 и 10 позиции. **Прямой алгоритм: 18 сравнений:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | = | b | b | a | b | a | c | a | b | c | a | b | a | d | |
| | | a | b | a | | | | | | | | | | | 1 сравнение |
| | | | a | b | a | | | | | | | | | | 1 сравнение |
| | | | | **a** | **b** | **a** | | | | | | | | | 3 сравнения, успех! |
| | | | | | a | b | a | | | | | | | | 1 сравнение |
| | | | | | | **a** | b | a | | | | | | | 2 |
| | | | | | | | a | b | a | | | | | | 1 |
| | | | | | | | | **a** | **b** | a | | | | | 3 сравнения! |
| | | | | | | | | | a | b | a | | | | 1 |
| | | | | | | | | | | a | b | a | | | 1 |
| | | | | | | | | | | | **a** | **b** | **a** | | 3, успех! |
| | | | | | | | | | | | | a | b | a | 1 |

# Алгоритм Кнута-Морриса-Пратта




Здесь $t_{i+1}\dots t_{i+j} = p_1\dots p_j$, но $t_{i+j+1} \neq p_{j+1}$.
Если максимальный суффикс цепочки $p_1\dots p_j$ , являющийся одновременно префиксом этой цепочки имеет длину $k$

($p_1\dots\ p_k = p_{j-k+1}\dots p_j$, но $p_{k+1} \neq p_{j-k}$), можно переходить к сравнению символа $t_{i+j+1}$ с $p_{k+1}$.

## Алгоритм Кнута-Морриса-Пратта

**Функция переходов**:

$g(j-1, p_j) = j$, $j = 1 \ldots m$; $g(0,a)=0$ для всех $a \in \Sigma$, $a \neq p_1$;

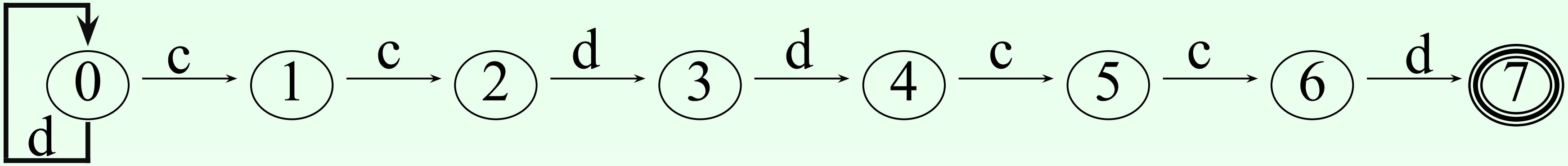
в остальных случаях $g(s,a) = fail$.

$g(s,a) = s'$ означает выходящее из $s$ ребро, помеченное символом $a$ и связывающее состояния $s$ и $s'$.

На этапе поиска функция переходов указывает, в какое состояние должен перейти автомат из текущего состояния $s$ при прочтении очередного символа текста $a$.

***p*** = ccddccd.

$g(0,\text{c}) = 1$; $g(1,\text{c}) = 2$; $g(2,\text{d}) = 3$ …



# Алгоритм Кнута-Морриса-Пратта

**Функция "отказов"** $f$

$f(j)$ – наибольшее $k < j$ для которого префикс образца P длины k ($p_1p_2\ldots p_k$) совпадает с суффиксом части образца длины $j$ ($p_1p_2\ldots p_j$),

т.е. $p_1\ldots p_{\mathrm{k}} = p_{j-k+1}\, p_{j-k+2} \ldots p_j$ . Если такого $k \geq 1$ нет, то $f(j) = 0$.

$f(1)$ **:** $= 0$;

**for** $j$ **:** $=2$ **to** $m$ **do**

**begin**

    $i$ **:**$= f(j-1)$;

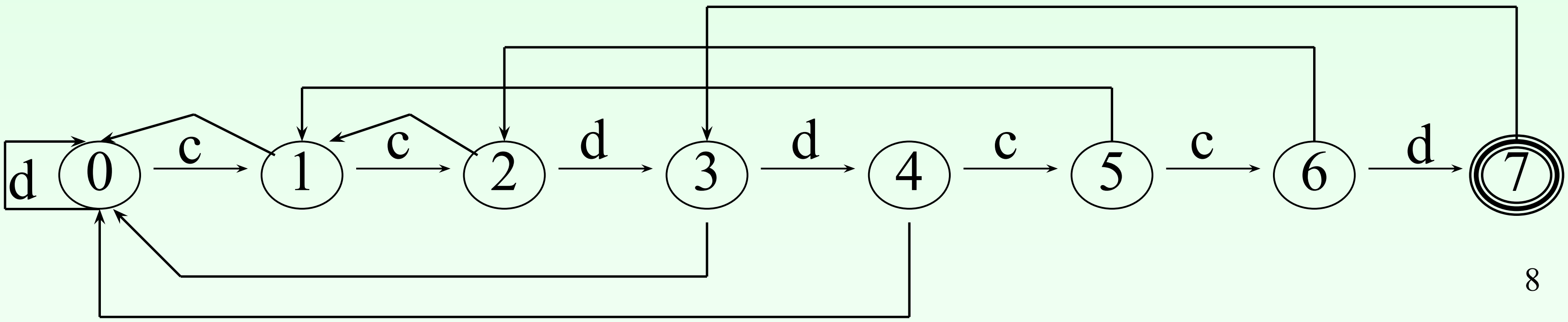
    **while** $(p_j \neq p_{i+1})$ and $(i > 0)$ **do** $i$**:**$=f(i)$;

    **if** $(p_j \neq p_{i+1})$ and $(i = 0)$

    **then** $f(j)$ **:**$= 0$

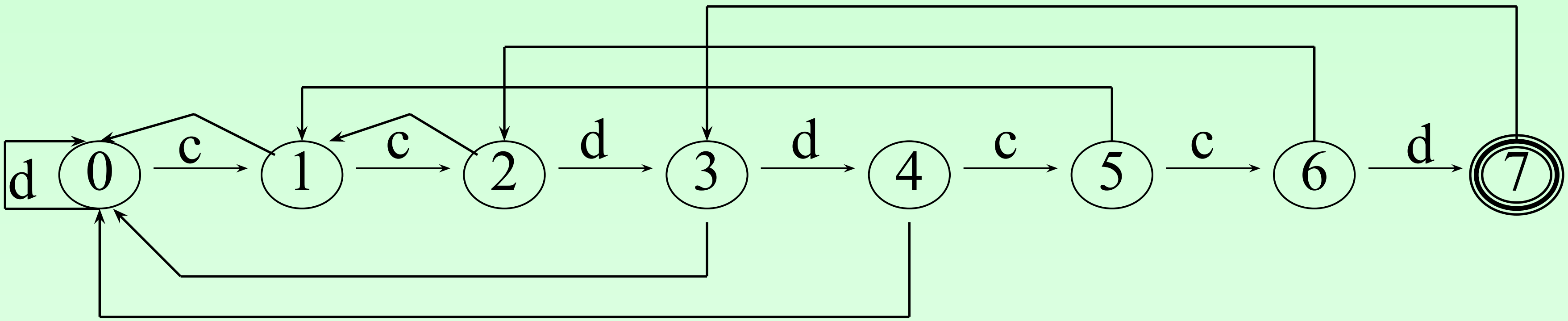
    **else** $f(j)$ **:**$= i + 1$;

**end**



# Алгоритм Кнута-Морриса-Пратта

***p*** = *ccddccd*. Машина идентификации цепочек (*Mp*):




ДКА, построенный по образцу ***p***:

| *δ* (*s*,*a*) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| *c* | 1 | 2 | 2 | 1 | 5 | 6 | 2 | 1 |
| *d* | 0 | 0 | 3 | 4 | 0 | 0 | 7 | 4 |

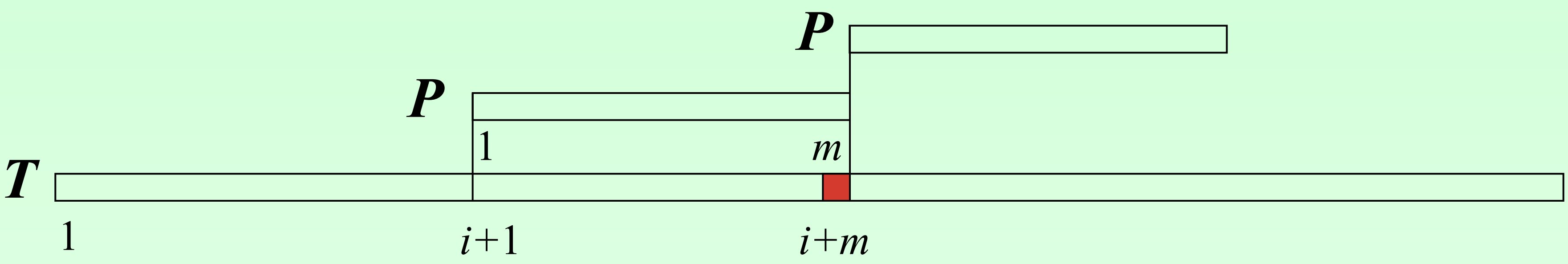
***T*** = *dddcddcdccddc**ccddccd**c*

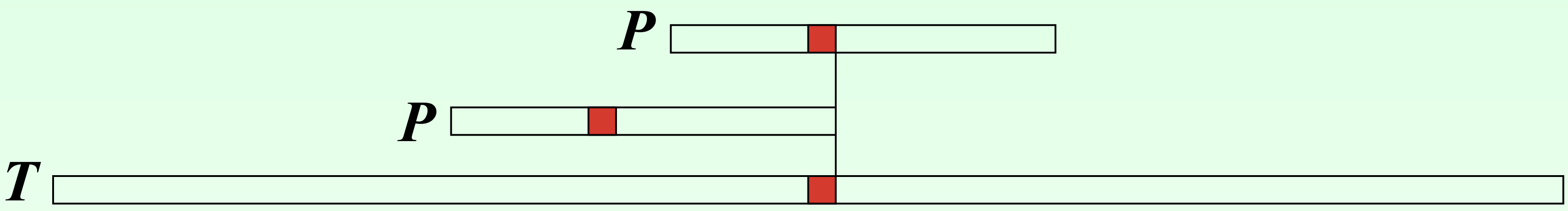
ДКА: 000100101234562345671

# Алгоритм Бойера-Мура

Сравнение символов – справа налево !!!

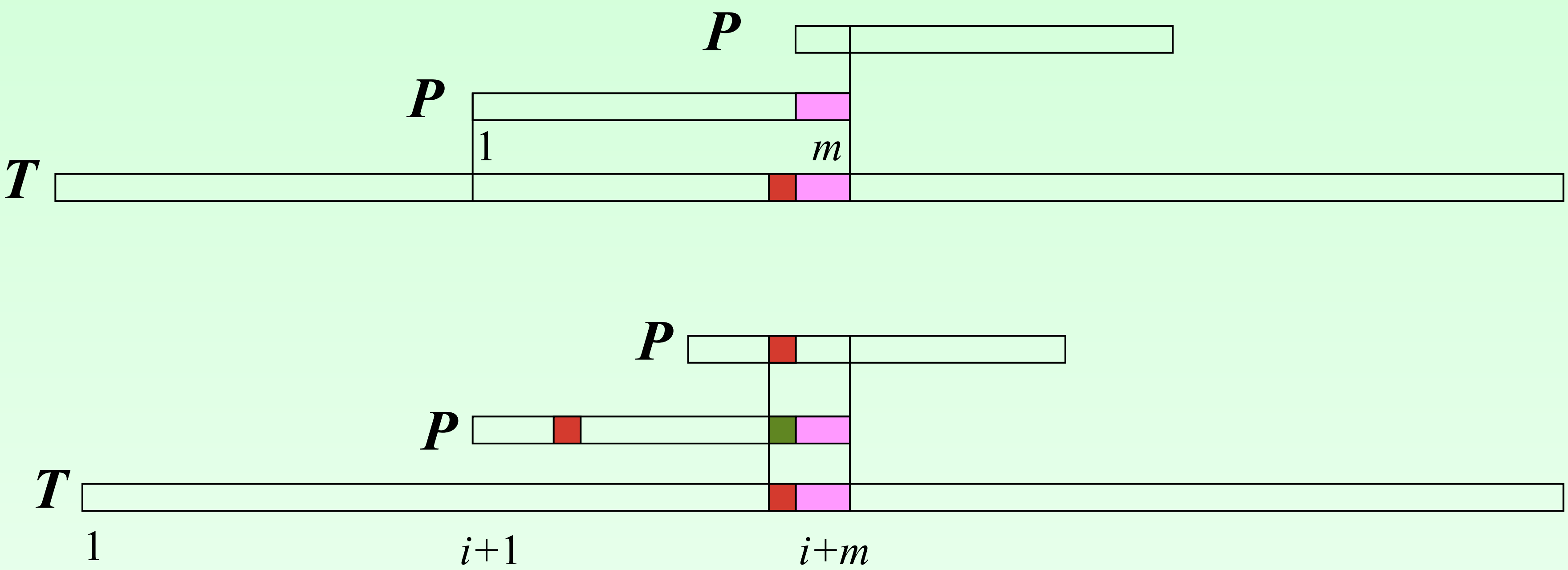
1. Правило «плохого символа».



# Алгоритм Бойера-Мура

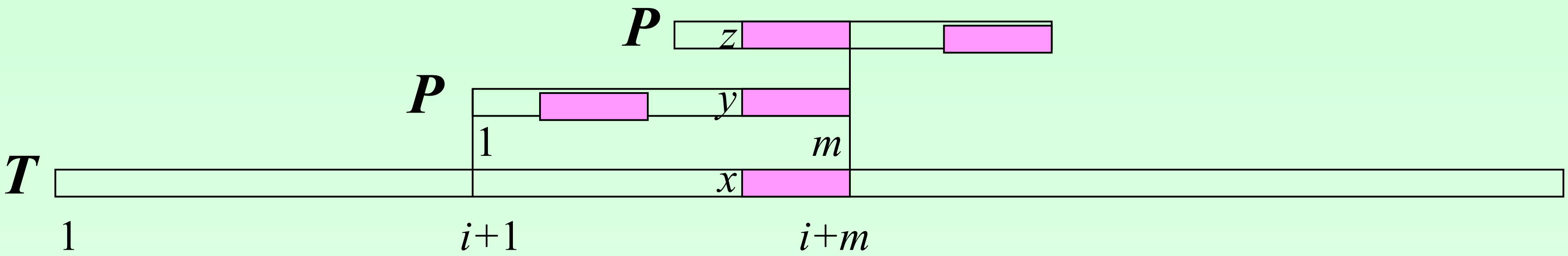
1. Правило «плохого символа».




$$\delta_1(a) = \begin{cases} m - j, \quad \text{где} \quad j = \max\{i \mid p_i = a\}, \\ m, \quad \text{если} \quad a \notin \{p_1, \ldots, p_m\}. \end{cases} \qquad a \in \Sigma$$

# Алгоритм Бойера-Мура

**2.** Правило «хорошего суффикса».




$$\delta_2(j) = j + 1 - rpr(j), \text{ где } 1 \le j \le m$$

$$rpr(j) = \max\{k \mid (P[j+1:m] = P[k:k+m-j-1])$$

$$and \quad ((k \le 1)\ or\ (p_{k-1} \ne p_j))\}$$

# Алгоритм Shift-And

$$R[i, j] = \begin{cases} 1, & \textit{если } P[1:i] = T[j-i+1:j], \\ 0, & \textit{в остальных случаях.} \end{cases}$$

$R[m, j] = 1$: $P$ в $(j - m + 1)$-й позиции $T$.

**Пример**. Пусть $\Sigma = \{a,b,c\}$, p=aabac, T=aabaacaabacab.

$R[3,3] = 1$, $R[4,4] = 1$, $R[5,5] = 0$;
$R[1,6] = 0$, $R[2,5] = 1$, $R[3,6] = 0$; $R[4,7] = 0$;

# Алгоритм Shift-And

$$R[i+1, j+1] = \begin{cases} 1, & \text{если } R[i, j] = 1 \text{ и } p_{i+1} = t_{j+1}, \\ 0, & \text{в остальных случаях.} \end{cases}$$

Схема перехода от $j$-го столбца $R$ к ($j$+1)-му состоит из:
правого сдвига $R[*, j]$
и ***And***-операции с $S[*, i + 1]$, где $s_{i+1} = t_{j+1}$.

**Пример**. Пусть Σ = {a,b,c}, p=aabac, T=aabaacaabacab.

| | a | b | c |
|---|---|---|---|
| 0 | | | |
| 1 | 1 | 0 | 0 |
| 2 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 |
| 5 | 0 | 0 | 1 |

***R***

| | | a | a | b | a | a | c | a | a | b | a | c | a | b |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| a | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| a | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| b | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| a | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 |

# Алгоритм Shift-And

$$R[i+1, j+1] = \begin{cases} 1, & \text{если } R[i,j] = 1 \text{ и } p_{i+1} = t_{j+1}, \\ 0, & \text{в остальных случаях.} \end{cases}$$

**Пример**. Пусть $\Sigma = \{a,b,c\}$, p=aabac, T=aabaacaabacab.

Схема перехода от 3-го столбца *R* к 4-му:

| $R[*,3]$ | | | | $S[a]$ | | $R[*,4]$ |
|---|---|---|---|---|---|---|
| 0 | | 1 | | 1 | | 1 |
| 0 | | 0 | | 1 | | 0 |
| 1 | → | 0 | And | 0 | = | 0 |
| 0 | | 1 | | 1 | | 1 |
| 0 | | 0 | | 0 | | 0 |

# Алгоритм Карпа-Рабина

***ns*** **: Σ → [1.. |Σ|] -** порядок символов в **Σ.**

Пусть s = |Σ|. Тогда

$H(P) = ns(p_1) \times s^{m-1} + ns(p_2) \times s^{m-2} \ldots ns(p_{m-1}) \times s + ns(p_m)$ и

$H(T[i : i + m - 1]) = ns(t_i) \times s^{m-1} + ns(t_{i+1}) \times s^{m-2} \ldots ns(t_{i+m-2}) \times s + ns(t_{i+m-1})$.

Если $H(P) = H(T[i : i + m - 1])$ **-** образец встретился в $i$-й поз. текста.

**Рекуррентное хеширование:**

$H(T[i + 1 : i + m) = (H(T[i : i + m - 1]) - ns(t_i) \times s^{m-1}) \times s + ns(t_{i+m})$.

**Схема Горнера вычисления Н:**

$H(P) = (\ldots(((ns(p_1) \times s + ns(p_2)) \times s + ns(p_3)) \times s + \ldots + ns(p_{m-1})) \times s + ns(p_m)$.

Пример. Σ = {acgt}, $P$ = acat, $T$ = ggacataccagac;

$H(P) = 1 \times 4^3 + 2 \times 4^2 + 1 \times 4^1 + 4 = 104$;

$H(T[1:4]) = 3 \times 4^3 + 3 \times 4^2 + 1 \times 4^1 + 2 = 246$;

$H(T[2:5]) = 3 \times 4^3 + 1 \times 4^2 + 2 \times 4^1 + 1 = 217 = (246 - 3 \times 4^3) \times 4 + 1$;

$H(T[3:6]) = 1 \times 4^3 + 2 \times 4^2 + 1 \times 4^1 + 4 = 104 = (217 - 3 \times 4^3) \times 4 + 4$;

## Обобщения задачи поиска образца:

- Поиск образца, позиции которого заданы множествами символов A- [AG]-C-[CG]-¬T-x-A (AGCCAAA, AACCGCA…)
- Поиск образца с допустимым уровнем искажений: ACGTAC – ACTTAC – ACGTCC – ACTGTAC – ACTAC
- Поиск множества образцов
- Комбинации задач (например, поиск множества образцов, позиции которых заданы множествами символов)

# Алгоритм Ахо-Корасик

***Задача.*** Задано множество образцов $\boldsymbol{P} = \{P_1, P_2, \ldots P_z\}$. Требуется **обнаружить все вхождения в текст *T* любого образца из *P*.**

$i$-й образец $P_i = p_{i1}p_{i2}\ldots p_{i,mi}$ имеет длину $m_i$; $p_{i,j} \in \Sigma$.

Текст $\boldsymbol{T} = t_1\ t_2 \ldots t_N$, $t_k \in \Sigma$, $1 \le k \le N$.

Это обобщение называют множественной задачей точного поиска или задачей поиска по групповому запросу

Наивный алгоритм решает эту задачу путем поиска каждого образца из набора с использованием любого из рассмотренных выше линейных алгоритмов. Такой поиск имеет трудоемкость $\mathrm{O}(zN + \sum_i m_i)$.

Эффективный алгоритм решения этой задачи имеет трудоемкость $\mathrm{O}(N + \sum_i m_i)$.

# Алгоритм Ахо-Корасик

- Этап предобработки: построение ДКА по исходному множеству образцов
- Этап поиска: однократный "прогон" текста через этот автомат.

1. Этап предобработки.

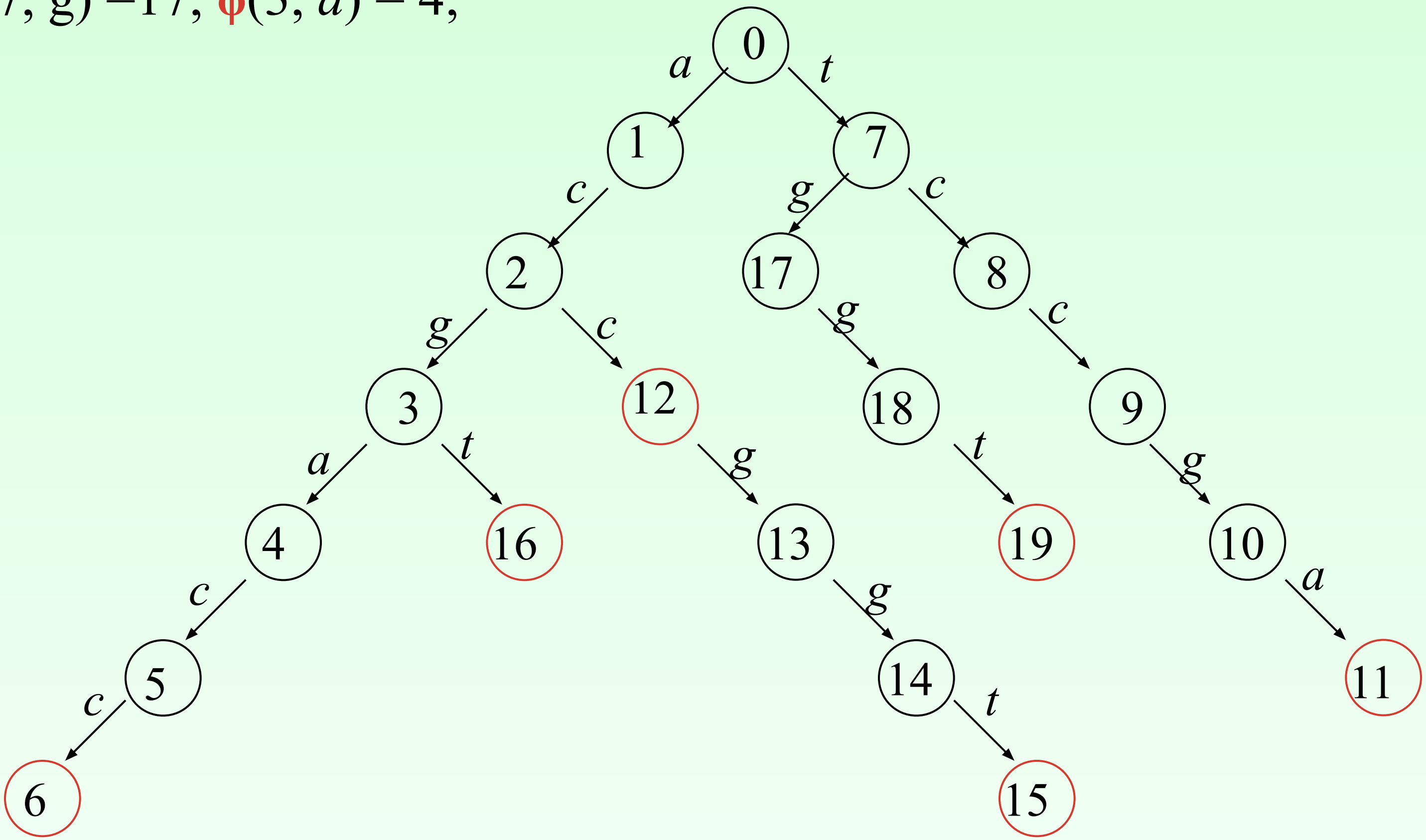
   Сначала строится "машина идентификации цепочек" ***Mp***. Работа машины ***Mp*** описывается тремя функциями: функцией переходов $\boldsymbol{\varphi(s,a)}$ ($s$ – состояние машины, $a \in \Sigma$), функцией отказов $\boldsymbol{f(s)}$
   и функцией выходов $\boldsymbol{o(s)}$.

# Алгоритм Ахо-Корасик

Функция переходов φ(s,*a*)=s', если существует выходящее из s ребро, помеченное символом "*a*" и связывающее состояния s и s'; в противном случае φ(s,a) = "fail" (ситуация, обозначаемая термином "отказ").

**Пример**. Пусть Σ = {a,c,g,t}; P = {acgacc, tccga, accggt, acgt, acc, tggt};

φ(7, g) =17; φ(3, *a*) = 4;



# Алгоритм Ахо-Корасик.

Построение *f* (*s*): пусть φ(*s_pred*,*a*) = *s*, *f*(*s_pred*) = *s"*.

Metka : Если φ(s",a)<> fail,  то  f(s)= φ(s",a); o(s) := o(s) ∪ o(f(s)) ,

иначе s" := f(s"); goto Metka.

Порядок построения: по уровням дерева  (структура «очередь»).

**Пример**. Пусть Σ = {a,c,g,t}; P = {acgatc, tccga, accggt, acgt, acc, tggt}; o(6)={1,4};

*f*(6) =12; *f*(2) = 0; *f*(16) = 7;