Иванов А.Б.

Алгоритмы и структуры данных

Строки

Поиск подстрок

Дано

- алфавит Σ
- образец $P[1..m] \in \Sigma^*$
- строка $T[1..n] \in \Sigma^*$

Требуется найти $0 \leq s \leq n - m$ такое, что
$T[s + i] = P[i]$, $\forall i = 1..m$ (возможно все такие s).

Наивный алгоритм

```
function find(string, pattern)
    m = len(pattern)
    n = len(string)

    for s = 0, 2,...n-m
        i = 1
        while (string[s + i] == pattern[i] && i <= m) i++

        if (i == m + 1) return s
    end
end
```

Сложность $O(nm)$

Алгоритм Рабина-Карпа

Будем рассматривать строки как числа в $|\Sigma|$-чной системе исчисления. Обозначим $d = |\Sigma|$ Тогда строка P вычисляется за $O(m)$ как

$$p = P[m] + d(P[m-1] + d(P[m-2] + \cdots + dP[1])\cdots).$$

Строку $T[s+1 \ldots s+m]$ можно вычислисть так (для всех s за $O(n)$):

$$t_{s+1} = d(t_s - d^{m-1}T[s+1]) + T[s+1+m]$$

Рассматриваем все эти числа по модулю q. Тогда

$$t_{s+1} = d(t_s - d^{m-1}T[s+1]) + T[s+1+m] \mod q$$

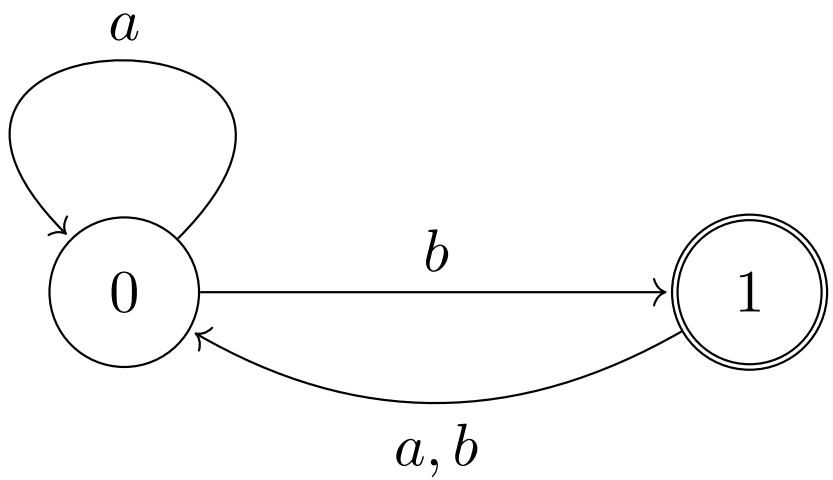
(вычислим $d^{m-1} \mod q$ заранее)

Сложность $O(nm)$, в среднем $O(n) + O(\frac{mn}{q})$

Конечный автомат

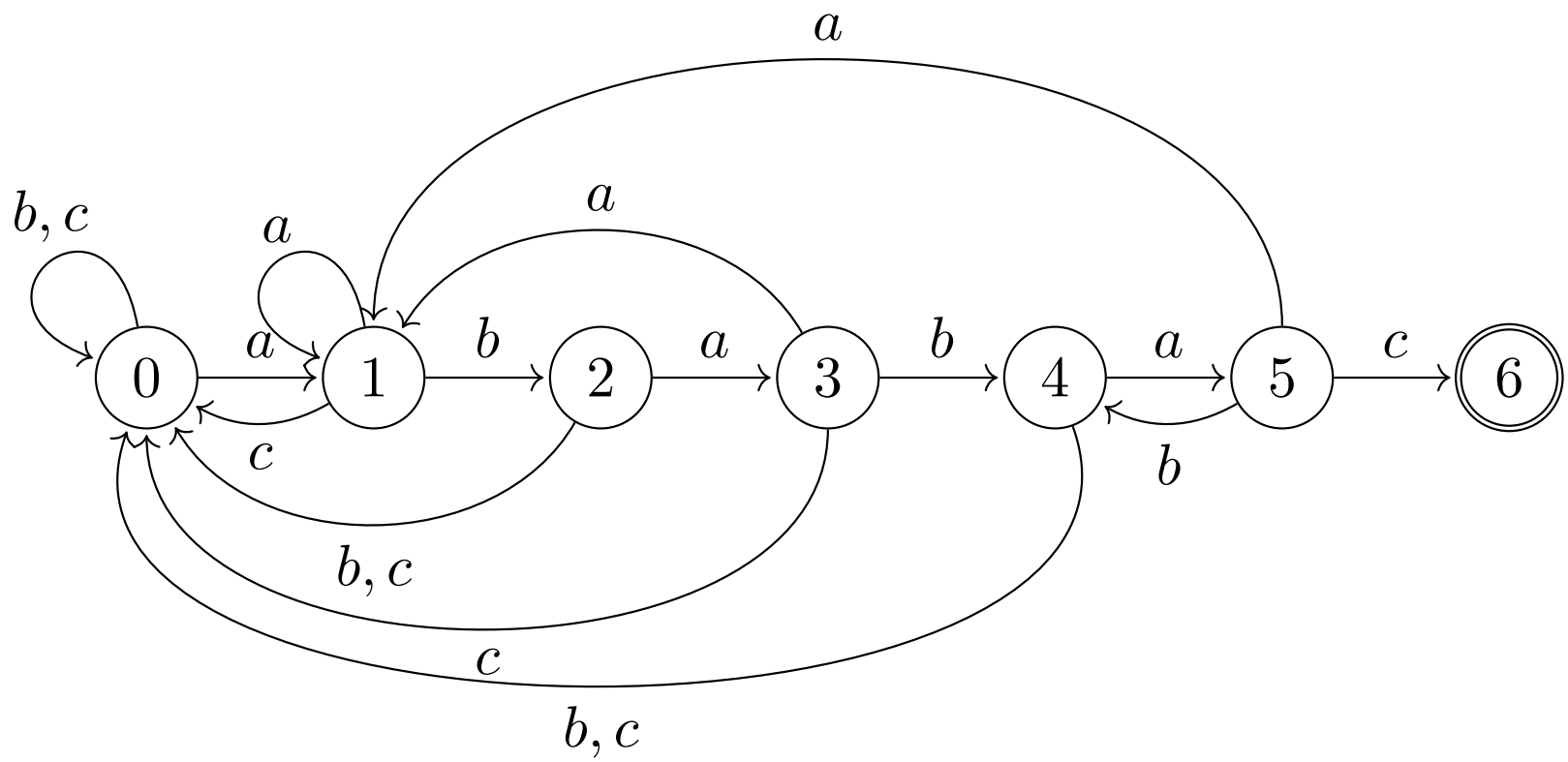
Конечный автомат - $(Q, q_0, A, \Sigma, \delta)$:

- Q - множество *состояний*
- $q_0 \in Q$ - *начальное состояние*
- $A \subseteq Q$ - множество *допустимых состояний*
- Σ - *алфавит*
- $\delta : Q \times \Sigma \to Q$ - *функция перехода*



Автомат поиска подстроки

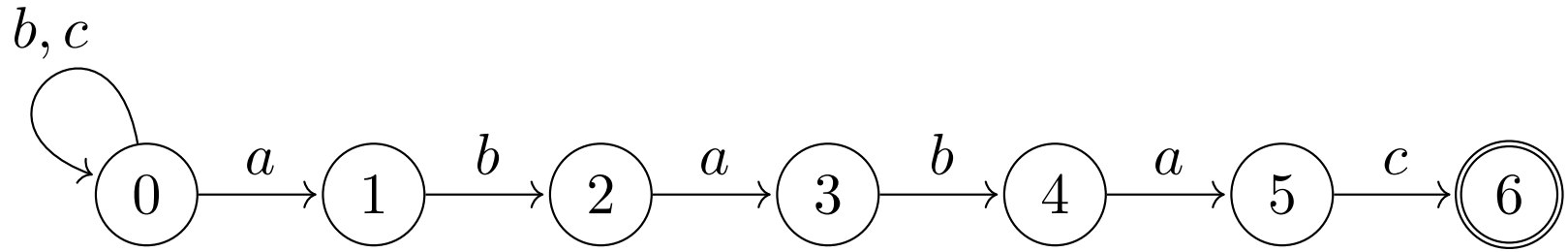
Автомат поиска $ababac$:



Сложность поиска подстроки автоматом - $O(n)$

Построение автомата

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



Сложность построения автомата - $O(|\Sigma|m)$

Подытог

Алгоритм:

- Строим автомат за $O(|\Sigma|m)$
- Поиск подстроки за $O(n)$

Проблемы (если алфавит Σ большой):

- Автомат занимает память $O(|\Sigma|m)$
- Общая сложность $O(|\Sigma|m + n)$

Можно лучше (Алгоритм Кнута-Морриса-Пратта):

- Память $O(m)$
- Общая сложность $O(m + n)$

Префикс-функция

Префикс-функция строки P:

$$\pi(i, P) = \{\text{длина наибольшего префикса } P, \text{ являющегосяго и суффиксом } P[1..i]\}$$

Построение за $O(m)$:

```
function build_prefix_function(string)
    m = len(string); pi = int[m]
    pi[1] = 0; k = 0
    for i = 0,...,m
        while (k > 0 && string[k+1] <> string[i]) k = pi[k]
        if (string[k+1] = string[i]) k = k + 1
        pi[i] = k
    end
    return pi
end
```

Алгоритм Кнута-Морриса-Пратта (КРМ)

Поиск за $O(n)$:

```
function match(string, pattern)
    m = len(pattern)
    n = len(string)
    pi = build_prefix_function(pattern)
    q = 0
    for i = 1,...,n
        while (q > 0 && string[q+1] <> pattern[i]) q = pi[q]
        if (pattern[q + 1] == string[i]) q = q + 1
        if (q == m) return i - m
    end
end
```