String Searching Algorithms

Robert Nazar

Institute of Mathematics

Katowice, 10 December 2015

Main Parts

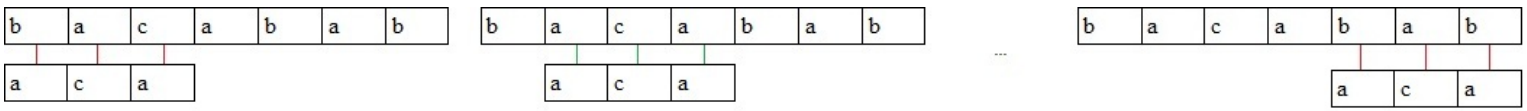
- String Matching Problem
- Naive String Search Algorithm
- Rabin-Karp String Search Algorithm
- Finite-state Automaton Based Search
- Knuth-Morris-Pratt Algorithm
- Computer Application

String Matching Problem

- Text is an array $T[1...n]$ of length n.
- String (also called pattern) is an array $P[1...m]$ of length m
- We say that pattern P occurs with shift s in text T (or, equivalently, that pattern P occurs beginning at position $s+1$ in text T) if $T[s+1, \ldots, s+m] = P[1...m]$

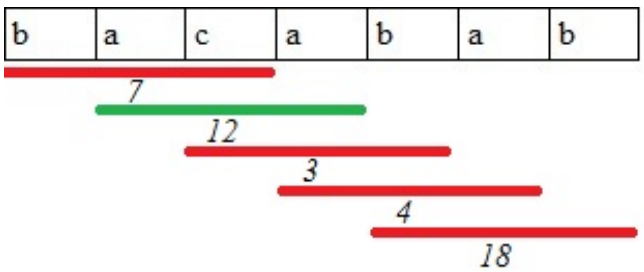
Naive String Search Algorithm

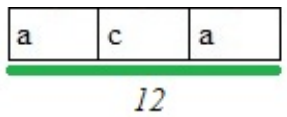
- Naive String Search Algorithm is the most simple way to find all occurrences of the pattern in the text. Algorithm compares all possible parts (of length m) of the text with the pattern.
- We have $n - m + 1$ opportunities to find a pattern.
- In every comparison we compare corresponding characters from the text and the pattern. If first two characters are matched we compare next two characters. We repeat this procedure until we don't have m matching characters or some two corresponding characters will be different.



Rabin-Karp String Search Algorithm

- Rabin-Karp String Search Algorithm uses operation called hashing to find all occurrences of the patterns in the text.
- Pattern and every part of the text of length m are converted to real number (or integer) called hashes.
- Then we compare not a text, but hashes of each part with hash of pattern. Only when the hashes are matched we compare corresponding characters from selected part of the text and the pattern.





Definition

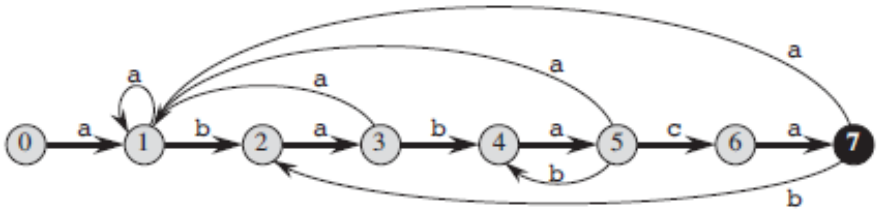
A Finite Automaton M is a 5-tuple $(Q, q_0, A, \Sigma, \delta)$, where:

- Q is a finite set of states,
- $q_0 \in Q$ is the start state,
- $A \subseteq Q$ is a distinguished set of accepting states,
- Σ is a finite input alphabet,
- δ is a function from $Q \times \Sigma \longrightarrow Q$ called the transition function of M.

Finite-state Automaton Based Search

- First we need to define function called the suffix function σ corresponding to P. The function σ maps Σ^* (words, which are constructed from alphabet Σ) to $\{0, 1, ..., m\}$ such that $\sigma(x)$ is the length of the longest prefix of P that is also a suffix of x:
  $\sigma(x) = \max\{k : P_k \sqsupset x\}$
- The transition function δ is defined by the following equation, for any state q and character a: $\delta(q, a) = \sigma(P_q a)$.
- Then we use the finite automaton for every character of the text (each symbol change the state, if value of state will be equal to m we will find a position of pattern in the text).

Example



(a)

state	input a	b	c	P
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

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
state	0	1	2	3	4	5	4	5	6	**7**	2	3

Knuth-Morris-Pratt Algorithm

- Algorithm works similar to finite-state automaton based search, but here we don't set all values of transition function. Instead of this we define auxiliary function π, which values will be save in array π (of length m). In this array we will have values of transition function only for current character from the pattern (not for all characters and every states).
- Prefix function is the function from $\{1, 2, ..., m\}$ to $\{0, 1, ..., m-1\}$ given by $\pi[q] = max\{k : k < q \wedge P_k \sqsupset P_q\}$
- Searching process is the same like in finite-state automaton based search but we uses function π instead of transition function.

Thank you for your attention.