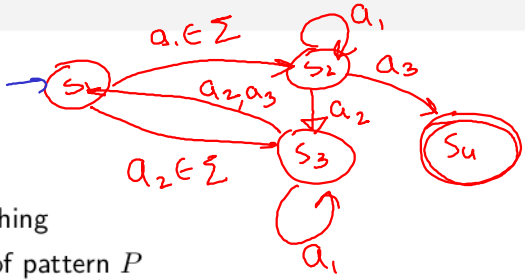
Data Structures and Algorithms

Prof. Ganesh Ramakrishnan,
Prof. Ajit Diwan,
Prof. D.B. Phatak

Department of Computer Science and Engineering
IIT Bombay

Session: Finite Automaton Algorithm

Introduction: Finite Automata[1]



- A simple machine for string matching
- Scans text T for all occurrences of pattern P

Variable

Fixed

$\{a_1 a_3, a_1 a_1 a_3, a_1 a_1 a_1 a_3, a_2 a_1 a_2 a_1 a_3 \cdots\}$

[1] Chapter 32, CLRS, Third Edition

Finite Automata[2]

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 the set of accepting states
- Σ is a finite input alphabet
- δ is the transition function of M from Q x Σ into Q

$\Sigma = \{a_1, a_2, a_3\}$

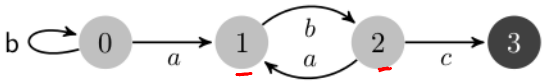
$\{\delta(s_1, a_1) = s_2, \ldots\}$

[2] Chapter 32, CLRS, Third Edition

Illustration

Text $T = ababababc$ Pattern $P = abc$

cbabacbc



$\Sigma = \{a, b, c\}$

$Q = \{0, 1, 2, 3\}$

$\delta(0, a) = 1$

$\delta(1, b) = 2$

$\delta(2, a) = 1$ . . . .

Algorithm for constructing finite automaton

Illustration on $P = ababc$ (length = m)

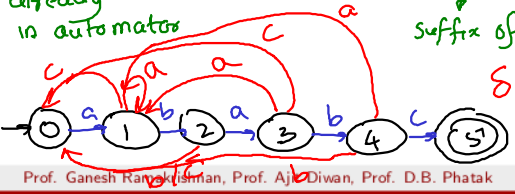
Goal: Keep track of character prefix of P that has already been matched so far

$\Sigma = \{a, b, c\}$ $Q = \{0, 1, 2, 3, 4, 5\}$ (5: accept state)

$\sigma(x)$ = length of longest prefix of P that is a suffix of x

sequence of chars already matched in automaton

$= \max\{k \mid P_k \sqsupset x\}$ (suffix of)



$\delta(q, a) = \sigma(P_q a)$

Construct_Finite_Automaton(P, Σ){
for q = 0 to m { → $m^3|\Sigma|$
for each $a \in \Sigma$ {
$k = \min(m, q+1)$
while ($P_k \not\sqsupset P_q a$)
$k = k-1$
$\delta(q, a) = k$.
} }

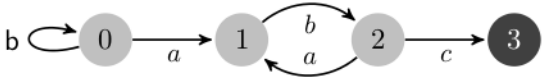
Scan backwards from $P_q a$ to find state to go to $\sigma(P_q a)$

+ Invariant

$\sigma(T[0..i]) =$ state after i

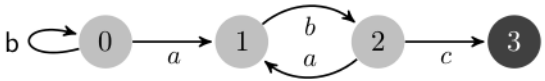
Illustration

Text $T = ababababc$ Pattern $P = abc$



Illustration

Text $T = ababababc$ Pattern $P = abc$

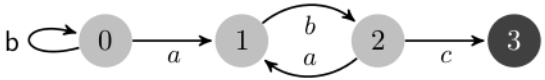


Pattern: abc

State	a	b	c
0	1	0	0
1	1	2	1
2	1	2	3
3	3	3	3

Illustration

Text $T = ababababc$ Pattern $P = abc$



Pattern: abc

State	a	b	c
0	1	0	0
1	1	2	1
2	1	2	3
3	3	3	3

i	-	1	2	3	4	5	6	7
$T[i]$	-	a	b	a	b	a	b	c
δ	0	1	2	1	2	1	2	3

Finite Automaton Algorithm (for matching T)

δ: P is offline, Onetime cost = $O(m^3|\Sigma|)$

Algorithm FiniteAutomatonStringMatchingAlgorithm(T, δ, m)
Input Text T of length n and Pattern P of length m
Define s as the shift index to T
$q = 0$
for $i \in (1...n)$ **do**
  $q = \delta(q, T[i])$
  if $q = m$ **then**
    print "Pattern occurs at shift" $i - m$
  end if
end for

Transitions also exist at accept state.

Linear scan
$O(n)$

Figure: Finite Automaton Algorithm

Thank you