String Matching

24 May 2025 22:42

$T[1 \ldots n]$; $m \leq n$ — like finding P in T.

$P[1 \ldots m]$

* Alphabet $\Sigma = \{0,1\}$ or $\Sigma = \{a\,b\,c \ldots z\}$.
  T & P are strings of Σ

* Pattern occurs with shift 's' in text T.

eg: → abacaabacaca. → T

P → aaba

$s = 4$ → P starts from $s+1$. {indexing '1' based}.

∴ $\boxed{0 \leq s \leq n-m}$

* Valid Shift :→ If P occurs with shift (s) → valid shift.

* String Matching → finding all valid shifts.

Prefix (⊏) and suffix (⊐)

* If w is a prefix of string x, $x = wy$ → $w \sqsubset x$

  w → suffix, $x = yw$ → $w \sqsupset x$.

eg :→ ab ⊏ abacca

ab ⊐ acab

empty string ;

ε

(in Σ^*)

Concatenation of 2 string

$x + y$.

→ $|T| = |x| + |y|$

Σ^*

all finite length string using Σ.

Naive string Matching algo

```
for (S=0 ; S <= (n-m))
    if P[1..m] = T[S+1 .... S+m]
        print "matched".
```

T.C :- $O((n-m+1)m)$.

String matching using finite automata.

Simple Machine.

process Texts in 'n' time, O(n) → Processing time = O(n)

each character exactly once

* Finite Automaton M has.

i) Q is a finite set of states.

ii) $q_0 \in Q$ is the start state

iii) $A \subseteq Q$ is accepting state

iv) Σ is finite input alphabet

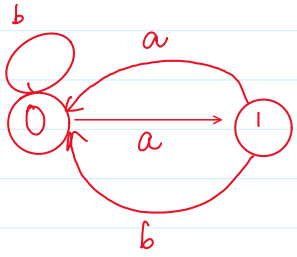
v) δ → transition function.

eg :- $Q = \{0,1\}$

$q_0 = 0$

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

State	Input a	Input b
0	1	0
1	0	0



$\varphi(w)$ → state M ends up after scanning the string w

$\delta(wa) = \delta(\varphi(w), a)$ for $w \in \Sigma^*$, $a \in \Sigma$

P = abaca

States = 0 → 4

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

Tabular representation of transition fun^n.

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

Longest Prefix of P that is a suffix of x

abaca

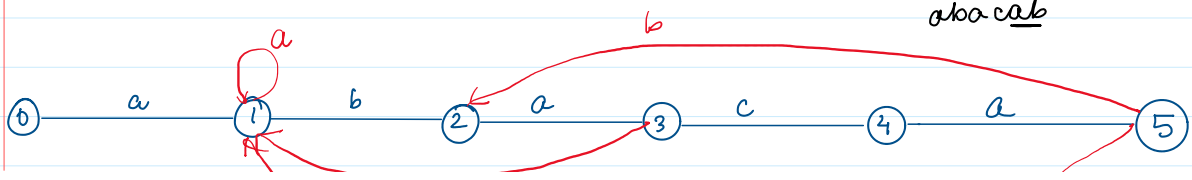
0: a, b, c

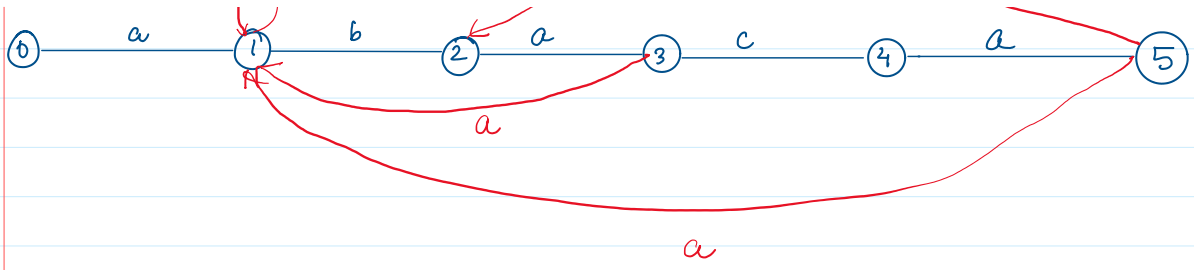
1: a/a, a/b, a/c

2: ab/a, ab/b, ab/c

3: aba/a, aba/b, aba/c

4: abac/a, b, c





$Q = \{0, 1, 2 \ldots m\}$

start q_0 , accepting state $= m$.

$\delta(q, a) = \sigma(P_q a)$ because we need to keep track of longest prefix that is matched sofar.

$P = abababaca.$

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

$Q = \{0 \ldots 7\}$

	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

0: a, b, c

1: a|b, a|a, a|c

2: ab|a, ab|b, ab|c

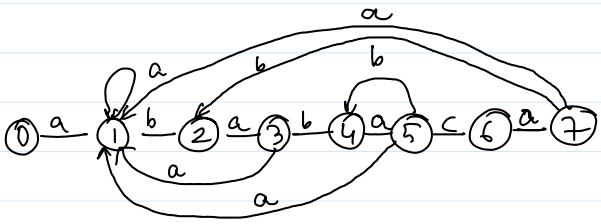
3: aba|a, aba|b, aba|c

4: abab|a, abab|b, abab|c

5: ababa|a, ababa|b, ababa|c

6: ababac|a, ababac|b, ababac|c

aca|a, aca|b, aca|c



i =	1	2	3	4	5	6	7	8	9	10	11	12
P[i] =	a	b	a	b	a	b	a	c	a	b	a	
State φ(Pi) =	0	1	2	3	4	5	4	5	6	7	2	3

Accepting state.

KNUTH - MORRIS - PRATT ALGO

* No computation of δ.
* Has a preprocessed array $\pi[1 \ldots m] \rightarrow$ order of $\Theta(m)$

$\pi[q] \rightarrow$ info to compute $\delta(q, a)$ but not dependent on a.

for automata $\delta \rightarrow \Theta(m|\Sigma|)$

$\pi \rightarrow \Theta(m)$

Prefix function (π) for a pattern: → stores info about how pattern matches with shifts.

eg:→ T = abcabaabcaac.

P = abaaba (shift s) — q = 4, Success

P = aba ba (shift s' = s+2)

Here see s+1 is not a valid shift a ≠ b

but s+2

given P[1..q] match T[s+1 ... s+q]

least shift s' for some K < q

$P[1 \dots K] \equiv T(s'+1 \dots s'+K)$ where $s'+K = s+q$ — (i)

In above case $s' = s+2$

$K = 2$

In other words given $P_q \sqsupset T_{s+q}$ find longest proper prefix P_K of P_q that is also suffix of T_{s+q}

Here K = 2

Finding smallest shift s' = finding longest prefix.

$$s' = s + q - k$$

q → no. of succes matching.

K → another longest prefix, also a suffix of T_{s+q}

Best case, K = 0, matlb ki a tak aur koi match nhi mila.

(s+1, s+2, s+3 ... s+q) → (X) Rule out.

eg^n(i) greatest K < q, $P_K \sqsupset P_q$

cabaabaabbbcc

→ abaabb (s)

→ abaabb (s+2)

FOUND

q = 4

k = 2

s' → s + (q − k)

= s + (2)

given a pattern P[1...m] the prefix function $\pi : \{1, 2, \dots m\} \to \{0 \dots m-1\}$

such that

$$\pi[q] = \max \{K : K < q \text{ and } P_K \sqsupset P_q\}$$

Prefix function.

$\pi[q] = \{K : K < q \text{ and } P_K \sqsupset P_q\}$.

eg:- for a pattern :→ ababbaca

π[q] → longest prefix of P that is a suffix of P_q

$\pi[q]$ → longest prefix of P that is a suffix of P_q

eg :→ P = ababaca.

1 → a ε

2 → a b (x)

3 → aba

4 → abab

5 → ababa

6 → ababac

7 → ababacа

i	1	2	3	4	5	6	7
P[i]	a	b	a	b	a	c	a
π[i]	0	0	1	2	3	0	1

KMP Matcher (T, P)

1.) n ← T. length

2.) m ← P. length.

3.) q = 0.

4.) π = COMPUTE PREFIX FUNCTION (P)

5.) for i = 1 to n.

6.) while (q > 0 and P[q+1] ≠ t[i])

7.) q = π[q]

8.) if P[q+1] == t[i]

9.) q = q + 1

10.) if q == m

11.) Print "Pattern occurs with shift" s - m.

12.) q = π[q]

(i)

abacababaabcbab

ababac

→ q = π[5] = 3, checks again if P[4] = t[i] (X)

so q = π[3] = 1.

if P[2] = t[i] ✓

COMPUTE PREFIX FUNCTION (P)

1.) m ← P. length

2.) π[1 ... m] = new array.

3.) π[1] = 0

4.) k = 0

5.) for q = 2 to m

6.) while (k > 0 and P[k+1] ≠ P[q])

7.) k = π[k]

Running time = Θ(m)

8.) if $P(k+1) == P(q)$

9.) $k = k + 1$

10.) $\pi[q] = k$

11.) return π

Running time analysis.

* k increments only in line 9.
  ↳ at most $(m-1)$ increments

* On entering for loop $k = 0$, $q = 2$, $\Rightarrow k < q$
  only assignment $\pi[1] = 0$
  $\pi[q] = k \Rightarrow \pi[q] < q$.
  for all $q \in \{1, 2 \dots m\}$

∴ each iteration of while loop decreases value of k

* k is always +ve

∴ The total decrease in k is bounded by the total increase in it.
which is $(m-1)$ ∴ while loop runs in $\theta(m)$.

By similar analysis, KMP matcher is $\theta(n)$

Preprocessing	$O(m\|\Sigma\|)$	$\rightarrow \theta(m)$
actual	$\theta(n)$	$\rightarrow \theta(n)$