## Data Structures and Algorithms

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

Department of Computer Science and Engineering IIT Bombay

Session: Knuth-Morris-Pratt Algorithm





## Summary of Pre-Processing and Matching time

. 6	49 <sup>50</sup>		
June esso.	Algorithm	Pre-Processing Time	Matching Time
The	Naive	0	O((n-m+1)m)
240	Rabin-Karp	O(m)	O((n-m+1)m)
Solenine	Finite Automaton	$O(m^3 \Sigma )$	O(n)
	Knuth-Morris-Pratt	O(m)	O(n)
8(.	) Table: Summ	ary Time of String Matching	g Algorithms

## Introduction<sup>1</sup>

- lacksquare linear-time string-matching algorithm for finding all occurrences of pattern P in a text T
- lacksquare avoids computing the transition function  $\delta$
- computes prefix function  $\pi$  for the pattern P by comparing the pattern against itself
- Given a pattern P[1..m], the prefix function for the pattern P is the function  $\pi:\{1,2,...,m\} \rightarrow \{0,1,...,m-1\}$  such that  $\pi[q]=\max\big\{k:k< q \ \& \ P_k\subseteq P_q\big\}$  i.e.  $\pi[q]$  is the length of the longest prefix of P that is a proper suffix of  $P_q$



<sup>&</sup>lt;sup>1</sup>Chapter 32, CLRS, Third Edition

### Computing $\pi[i]$

- 1st Cell Pattern a
- $\pi[1] = 0$

i	1	2	3	4	5	6
P[i]	а	b	а	b	а	b
$\pi[i]$	0					

## Computing $\pi[i]$

- 2nd Cell Pattern a b
- Proper Prefix:
  - **>** 0
- Proper Suffix:
  - ▶ 1
- Not matching, so,  $\pi[2] = 0$

i	1	2	3	4	5	6
P[i]	а	b	а	b	а	b
$\pi[i]$	0	0				

## Computing $\pi[i]$

- $\blacksquare$  3rd Cell Pattern  $a\ b\ a$
- Proper Prefix:
  - ▶ a, ab
- Proper Suffix:
  - ▶ a. ba
- **a** is present in both. Since it is 1 character length,  $\pi[3] = 1$

i	1	2	3	4	5	6
P[i]	a	<sub>e</sub> b	а	b	а	b
$\pi[i]$	0	0	-1			

### Computing $\pi[i]$

- 4th Cell Pattern a b a b
- Proper Prefix:
  - ► a, ab, aba
- Proper Suffix:
  - **▶** *b*, *ab*, *bab*
- ab is present in both. Since it is 2 character length,  $\pi[4] = 2$

i	1	2	3	4	5	6
P[i]	<u>a</u>	b	а	b	а	b
$\pi[i]$	0	0	1	_2		

## Computing $\pi[i]$

- 5th Cell Pattern a b a b a
- Proper Prefix:
  - a, ab, aba, abab
- Proper Suffix:
  - ▶ a, ba, aba, baba
- a and aba are present in both. Since aba has 3 character length which is greater than a, so,  $\pi[5] = 3$

i	1	2	3	4	5	6
P[i]	а	b	а	b	а	b
$\pi[i]$	0	0	1	2	3	

## Computing $\pi[i]$

- 6th Cell Pattern a b a b a b
- Proper Prefix:
  - a, ab, aba, abab, ababa
- Proper Suffix:
  - ▶ b, ab, bab, abab, babab
- ab and abab are present in both. Since abab has 4 character length which is greater than ab, so,  $\pi[6] = 4$

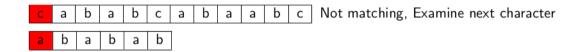
i	1	2	3	4	5	6
P[i]	а	b	а	Ь	а	b
$\pi[i]$	0	0	1	2	3	4

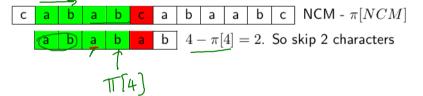
#### Knuth-Morris-Pratt Illustration

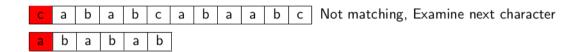
- $\blacksquare \text{ Text } T: c \ a \ b \ a \ b \ c \ a \ b \ a \ b \ c$
- $\blacksquare$  Pattern P:  $a \ b \ a \ b \ a \ b$
- We compare each character of pattern P with text T and obtain partial matching pair
- Number of characters currently processed (NCM) for pattern P in text T is used to compute the following
  - ▶ NCM  $\pi[NCM]$
- The result denotes the number of characters that needs to be skipped in text T

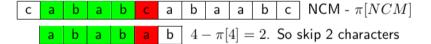


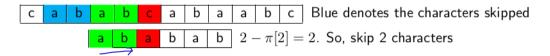
С	а	b	а	b	С	а	b	а	а	b	С	Not matching, Examine next character
а	b	а	b	а	b							





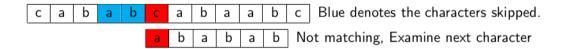






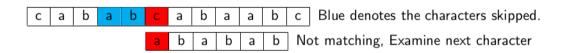


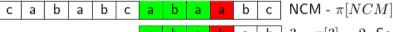
С	а	b	а	b	С	а	b	а	а	b	c Blue denotes the characters skipped
					а	b	а	b	а	b	Not matching, Examine next character





b a b a b 
$$3-\pi[3]=2$$
. So, skip 2 characters





$$oxed{a \mid b \mid a \mid b \mid} \ a \mid b \mid 3 - \pi[3] = 2.$$
 So, skip 2 characters

$$oxed{c}$$
  $oxed{a}$   $oxed{b}$   $oxed{a}$   $oxed{b}$   $oxed{a}$   $oxed{b}$   $oxed{c}$  Blue denotes the characters skipped  $oxed{a}$   $oxed{b}$   $oxed{a}$   $oxed{b}$   $oxed{b}$   $oxed{a}$   $oxed{b}$  Pattern  $P$  is longer than text  $T$ 





## Knuth-Morris-Pratt Algorithm

```
Algorithm KMPAlgorithm(T, P)
Input Text T of size n and Pattern P of size m
\pi = \mathsf{ComputePrefix}(P)
q = 0
for i \in (1...n - m) do
  while q > 0 and P[q+1] \neq T[i] do
     q = \pi[q] Shift of NCM-T(NCM)
  end while
  if P[q+1] = T[i] then
     q = q + 1
  end if
  if q = m then
     print 'Pattern occurs with shift' i-m
     q = \pi[q]
  end if
end for
```

Figure: Knuth-Morris-Pratt-Algorithm



# ComputePrefix(P) Function

```
Algorithm ComputePrefix(P)
Input Pattern P of size m
Define \pi[1...m]
\pi[1] = 0
g = 0
Note that
for i \in (2...m) do
    while q > 0 and P[q + 1] \neq P[i] do
       a = \pi[a]
    end while
   if P[q+1] = P[i] then
       q = q + 1
   end if
    \pi[i] = q
end for
return \pi
```

Figure: Compute-Prefix Function



## Analysis of ComputePrefix(P) Function

```
Algorithm ComputePrefix(P)
Input Pattern P of size m
Define \pi[1...m]
\pi[1] = 0
a = 0
for i \in (2...m) do
   while q > 0 and P[q + 1] \neq P[i] do
      q = \pi[q]
   end while
   if P[q+1] = P[i] then
      q = q + 1
   end if
   \pi[i] = q
end for \implies c_1 \times (m-1) times
return \pi
```

Figure: Compute-Prefix Function

$$T(n) = c_1(m-1) = O(m)$$



## Analysis of Knuth-Morris-Pratt Algorithm

```
Algorithm KMPAlgorithm(T, P)
Input Text T of size n and Pattern P of size m
\pi = \mathsf{ComputePrefix}(P)
a = 0
for i \in (1...n - m) do
   while q > 0 and P[q + 1] \neq T[i] do
      q = π[q] ← back tracking
   end while
  if P[q+1] = T[i] then
      q = q + 1
  end if
   if q = m then
      print 'Pattern occurs with shift' i-m
      q = \pi[q]
   end if
end for \implies c_1 \times n - m times
```

Figure: Knuth-Morris-Pratt-Algorithm

$$T(n) = c_1 n = O(n)$$



# Thank you