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

(KMP)

Summary of Pre-Processing and Matching time

Remove unnecessary baggage

Algorithm	Pre-Processing Time	Matching Time
Naive	0	$O((n-m+1)m)$
Rabin-Karp	$O(m)$	$O((n-m+1)m)$
Finite Automaton	$O(m^3\|\Sigma\|)$	$O(n)$
Knuth-Morris-Pratt	$O(m)$	$O(n)$

$\delta(\,.\,)$

Table: Summary Time of String Matching Algorithms

Introduction[1]

- linear-time string-matching algorithm for finding all occurrences of pattern P in a text T
- avoids computing the transition function δ
- computes prefix function π 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
  $\pi[q] = max\,\{k : k < q \;\&\; P_k \subseteq P_q\}$
  i.e. $\pi[q]$ is the length of the longest prefix of P that is a proper suffix of P_q

[1] Chapter 32, CLRS, Third Edition

Computing prefix function π for the pattern $ababab$

Computing $\pi[i]$

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

Computing $\pi[i]$

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

Computing prefix function π for the pattern $ababab$

Computing $\pi[i]$

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

Computing $\pi[i]$

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

Computing prefix function π for the pattern *ababab*

Computing $\pi[i]$

- 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, $\pi[3] = 1$

Computing $\pi[i]$

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

Computing prefix function π for the pattern $ababab$

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, $\pi[4] = 2$

Computing $\pi[i]$

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

Computing prefix function π for the pattern $ababab$

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, $\pi[5] = 3$

Computing $\pi[i]$

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

Computing prefix function π for the pattern $ababab$

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, $\pi[6] = 4$

Computing $\pi[i]$

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

Knuth-Morris-Pratt Illustration

- Text T: $c\ a\ b\ a\ b\ c\ a\ b\ a\ a\ b\ c$
- 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

  Num Charac Matched

  - NCM - $\pi[NCM]$
- The result denotes the number of characters that needs to be skipped in text T

  $P[\ldots NCM]$

Illustration

c	a	b	a	b	c	a	b	a	a	b	c

Not matching, Examine next character

a	b	a	b	a	b

Illustration

c	a	b	a	b	c	a	b	a	a	b	c

Not matching, Examine next character

a	b	a	b	a	b

c	a	b	a	b	c	a	b	a	a	b	c

NCM - $\pi[NCM]$

a	b	a	b	a	b

$4 - \pi[4] = 2$. So skip 2 characters

$\pi[4]$

Illustration

c	a	b	a	b	c	a	b	a	a	b	c

Not matching, Examine next character

a	b	a	b	a	b

c	a	b	a	b	c	a	b	a	a	b	c

NCM - $\pi[NCM]$

a	b	a	b	a	b

$4 - \pi[4] = 2.$ So skip 2 characters

c	a	b	a	b	c	a	b	a	a	b	c

Blue denotes the characters skipped

a	b	a	b	a	b

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

Illustration

c	a	b	a	b	c	a	b	a	a	b	c

Blue denotes the characters skipped.

a	b	a	b	a	b

Not matching, Examine next character

Illustration

c	a	b	a	b	c	a	b	a	a	b	c

Blue denotes the characters skipped.

a	b	a	b	a	b

Not matching, Examine next character

c	a	b	a	b	c	a	b	a	a	b	c

NCM - $\pi[NCM]$

a	b	a	b	a	b

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

Illustration

c	a	b	a	b	c	a	b	a	a	b	c	Blue denotes the characters skipped.
					a	b	a	b	a	b		Not matching, Examine next character

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

c	a	b	a	b	c	a	b	a	a	b	c	Blue denotes the characters skipped		
								a	b	a	b	a	b	Pattern P is longer than text T

$\underline{\underline{n-m}}$

Knuth-Morris-Pratt Algorithm

Algorithm KMPAlgorithm(T, P)
Input Text T of size n and Pattern P of size m
$\pi =$ 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 - π[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 $\pi[1...m]$
$\pi[1] = 0$
$q = 0$
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
return π

Index into P but treating P as T

P_q

Figure: Compute-Prefix Function

Analysis of ComputePrefix(P) Function

Algorithm ComputePrefix(P)
Input Pattern P of size m
Define $\pi[1...m]$
$\pi[1] = 0$
$q = 0$
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 $\Longrightarrow$ c_1 $\times$ $(m-1)$ times
return π

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
π = ComputePrefix(P)
$q = 0$
for $i \in (1...n - m)$ **do**
  while $q > 0$ **and** $P[q+1] \neq T[i]$ **do**
    $q = \pi[q]$ $\leftarrow$ 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 $\Longrightarrow$ c_1 $\times$ $n - m$ times

Figure: Knuth-Morris-Pratt-Algorithm

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

Thank you