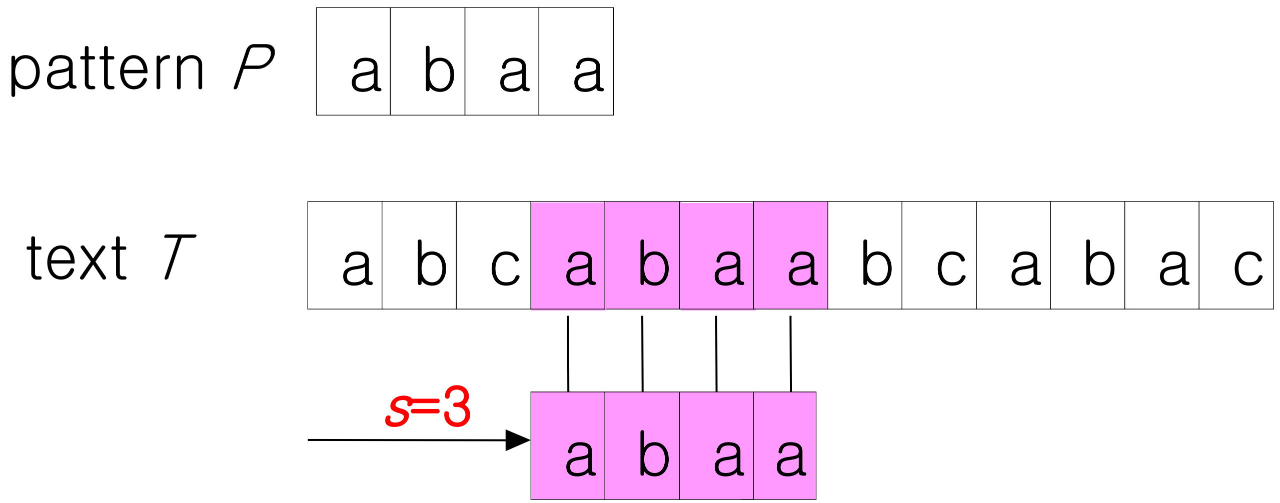
Chapter 32
String Matching

Algorithm Analysis

School of CSEE

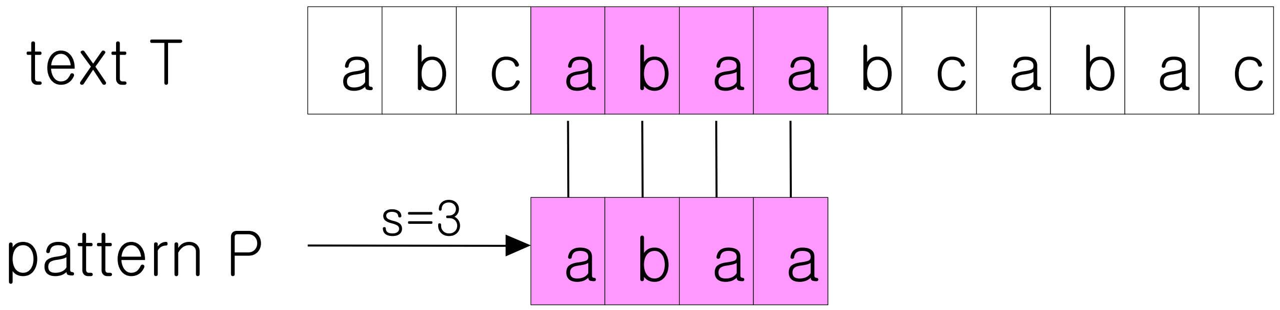
The string matching problem

Find all valid shifts with which a given pattern P occurs in a given text T.

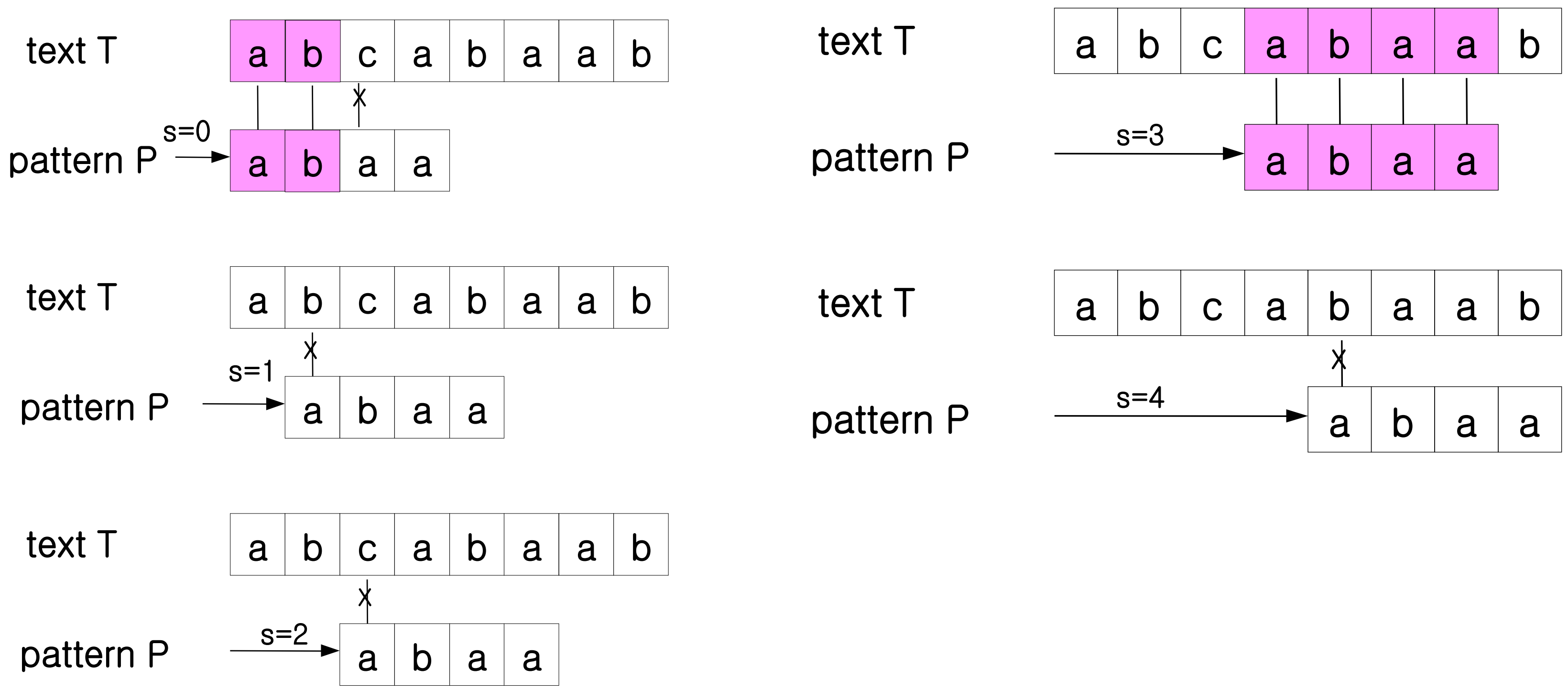


Notation and terminology

- *T*[1..*n*]: the text
- *P*[1..*m*]: the pattern
- *P* occurs with shift *s* in *T* if *T*[*s*+*j*]=p[*j*] for 1≤*j*≤*m*.
- If *P* occurs with shift s in *T*, then we call *s* a valid shift; otherwise, an invalid shift.



[1] The naïve string-matching algorithm



The naïve string-matching algorithm

```
Naïve-String-Matching(T,P)
n←|T|
m←|P|
for s←0 to n-m
   do if P[1..m]=T[s+1..s+m]
         then print "Pattern occurs with shift" s
```

Problem in the naïve algorithm

- O((*n*-*m*+1)*m*) time

- The naïve string matching is inefficient because information gained about the text for one value of *s* is entirely ignored in considering other values of *s*.
  - If *P*=aaab and we find *s*=0 is valid, then none of the shifts 1, 2, or 3 are valid.

[2] Rabin-Karp algorithm

Main Idea

: length *m* string is regarded as *m* digits radix-*d* number.

- *P*[1..*m*] : Convert it into *m*-digit number *p*
- Substring *T*[*s*+1..*s*+*m*] : Convert it into *m*-digit number t_s

Ex) ∑={0,1,2,…,9}, *P*[1..*m*]=31425,

➔ *p*= 31,425

- If $p = t_s$, then string matching!
- String matching problem

➔ is converted into number comparison problem.

Example

text T

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

ts

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
23590	35902	59023	90231	02314	23141	31415	14152	41526	15267	52673	26739	67399	73992	39921

Pattern P

3	1	4	1	5

➔ Then p is 31415

How to convert string into number?

- Use Horner's rule (from Section 30.1, page 824)

$$p = P[m] + 10(P[m-1] + 10(P[m-2] + \ldots + 10(P[2] + 10P[1])\ldots))$$

Ex) When $P[1..m]$=31425,

$$p = 5+10*(2+10*(4+10*(1+10*3))) = 31{,}425$$

- Is there faster way to calculate *ts*?

How to convert string into number?

- Calculate t_0 similarly.

  Then, we can calculate t_{s+1} from t_s
  - $t_{s+1}= 10(t_s-10^{m-1}T[s+1])+T[s+m+1]$
  - i.e., remove high order digit $T[s+1]$ and bring low order digit.

Ex) When $t_s = 31415$ and $T[s+5+1]= 2$,

$t_{s+1} = 10(31415 – 10000*3) +2 = 14152$

Computing p & t_0 : $\Theta(m)$

Computing $t_1, \ldots t_{n-m}$: $\Theta(n-m)$ or $\Theta(n)$

Comparing *p* with t_s

- How long will it take to compare *p* with t_s?

  - Constant time if *m* is very small.

  - Otherwise …

- Cure for the problem : Use 'modulo' operation.

  When comparing two numbers, we do not compare the numbers directly. Instead, take 'modulo q' operation and compare.

Comparing p with t_s

- However, the solution of working 'modulo q' is not perfect, since $ts \equiv p \pmod{q}$ does not imply $t_s = p$.
  - Valid : $t_s \equiv p \pmod{q}$ and $t_s = p$
  - Spurious hit : $t_s \equiv p \pmod{q}$ but $t_s \neq p$
  - However, if $t_s \neq p \pmod{q}$, there is no chance that $t_s = p$.

Ex) 67399!= 31415 but, 67399≡31415 (mod 13)

The Rabin-Karp algorithm

text T

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

pattern P

3	1	4	1	5

$\xrightarrow{\text{mod } 13}$ 7

***m*=5 and *q*=13**

***p*=7**

Step 1: Construct the array t_s.

text T

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

ts

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	9	3	11	0	1	7	8	4	5	10	11	7	9	11

$$t_s = \text{the decimal value of } T[s+1..s+m] \text{ mod } q$$

The Rabin-Karp algorithm

Step 2: Find s such that t_s=p.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
text T	2	3	5	9	0	2	3	1	4	1	5	2	6	7	3	9	9	2	1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ts	8	9	3	11	0	1	7	8	4	5	10	11	7	9	11

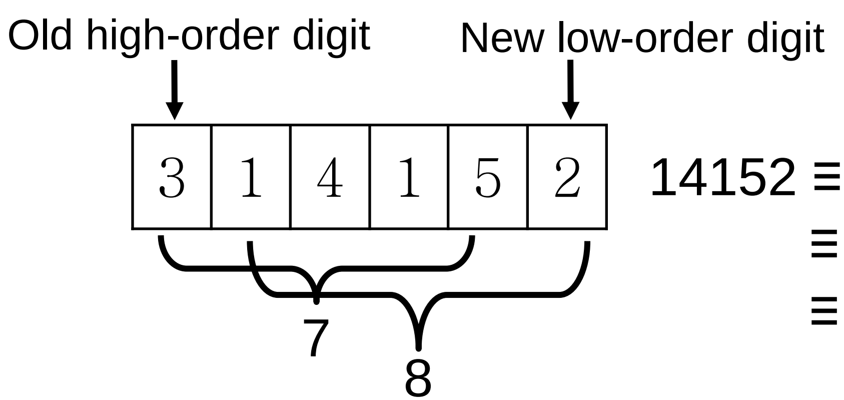
Step 3: Check if s is really valid.

1. s=7: T[7..11]=P → **valid match**
2. s=13: T[13..17]≠P → **invalid (spurious hit)**

Modulo operation

- q : The modulus q is typically chosen as a prime number such that $d*q$ just fits within one computer word in d-ary alphabet.
- Recalculation of p and t
  - p = original p (mod q)
  - $t_{s+1}=(d(t_s-T[s+1]h)+T[s+m+1])$ (mod q)

Ex)



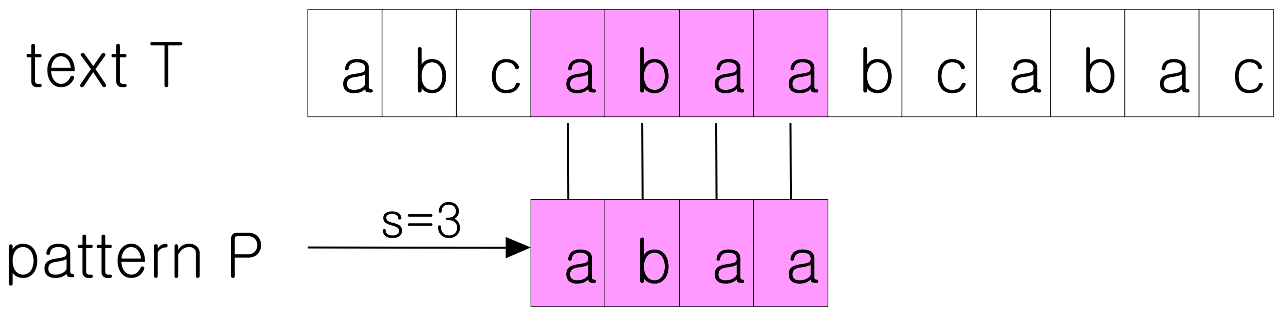
Can be precomputed

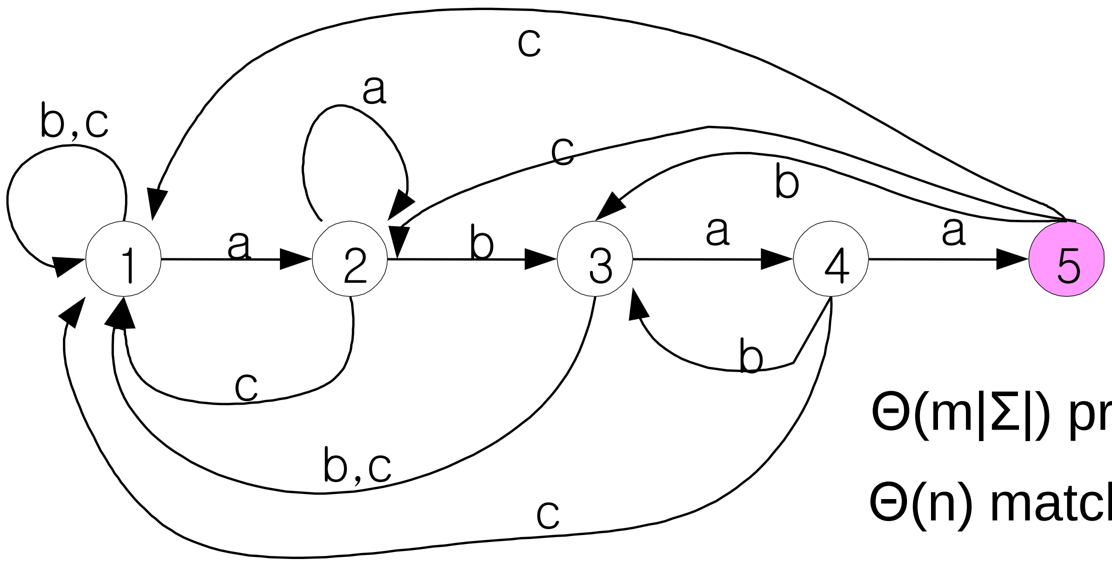
$$14152 \equiv (31415 - 3*10000) * 10 + 2 \pmod{13}$$
$$\equiv (7 - 3*3)*10 + 2 \pmod{13}$$
$$\equiv 8 \pmod{13}$$

The Rabin-Karp algorithm

- $\Theta(m)$ preprocessing time --- calculation of p and t_0
- $\Theta((n\text{-}m+1)m)$ worst-case running time
  - $\Theta(n\text{-}m+1)$ times to find all s such that $p=t_s$.
  - $\Theta(m)$ time to check if each s is really valid.
  - However we expect few valid shifts.

[3] String matching with finite automata





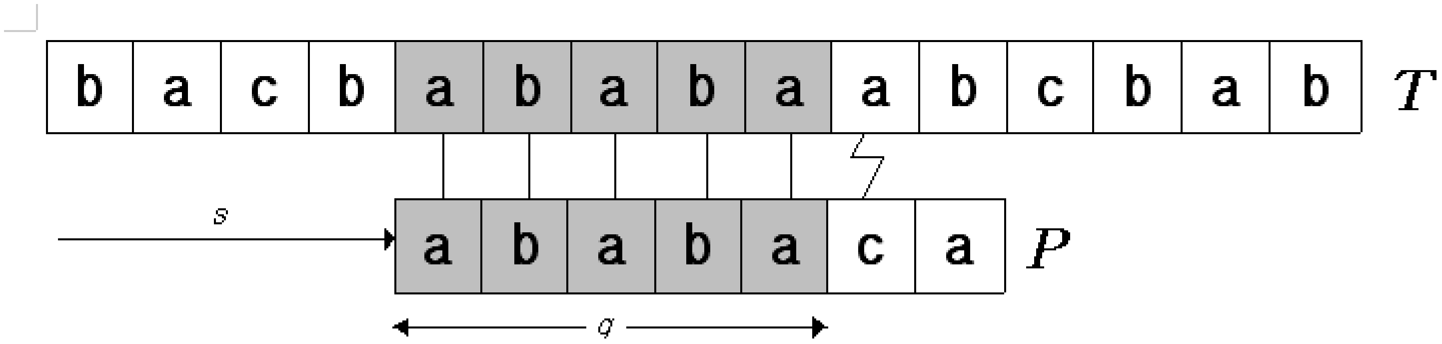
Too expensive

Θ(m|Σ|) preprocessing time

Θ(n) matching time

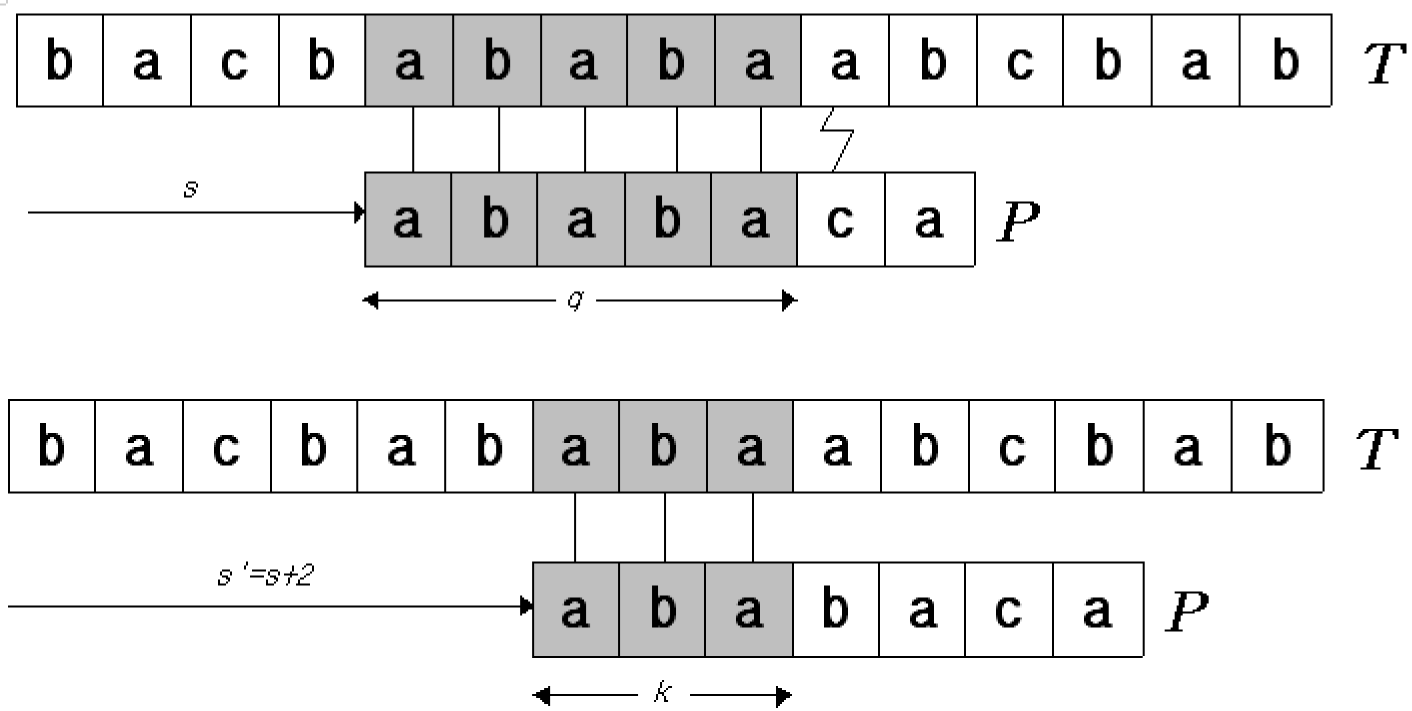
[4] Knuth-Morris-Pratt Algorithm

- Consider the operation of the naïve string matcher.



When 6th pattern character fails to match the corresponding text character, where can we resume the match again?

- We don’t have to resume the match from the character right next to it!



Knuth-Morris-Pratt Algorithm

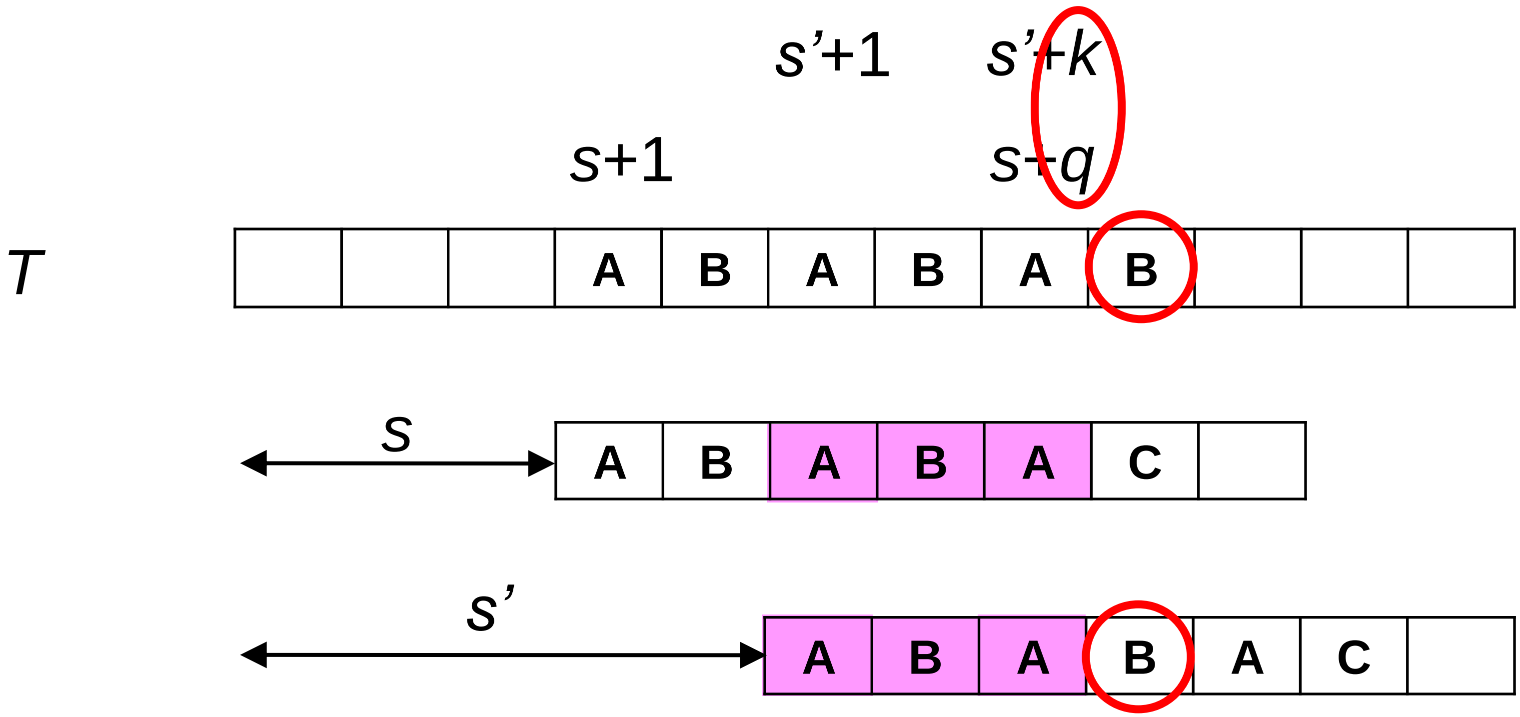
- In general, it is useful to know the answer to the following question :

  Given that pattern characters $P[1..q]$ match text characters $T[s+1..s+q]$, what is the least shift $s' > s$ such that

  $$P[1..k] = T[s'+1..s'+k],$$

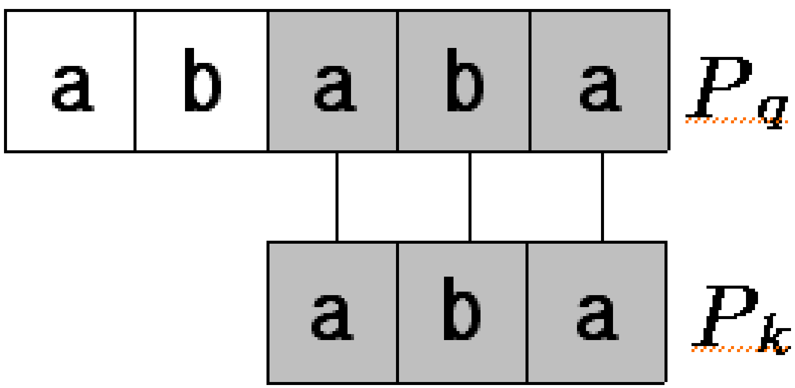
  where $s'+k = s+q$?

Knuth-Morris-Pratt Algorithm



Knuth-Morris-Pratt Algorithm

- The necessary information can be precomputed by comparing the pattern against itself.



The longest prefix of P that is also a proper suffix of P_5 is P_3

→ $\pi(5)=3$

Prefix function π

pattern P

1	2	3	4	5	6	7	8	9	10
a	b	a	b	a	b	a	b	c	a

P_{10}

1	2	3	4	5	6	7	8	9	10									
a	b	a	b	a	b	a	b	c	a									
									a	b	a	b	a	b	a	b	c	a

$\pi(10)=1$

P_9

1	2	3	4	5	6	7	8	9	10									
a	b	a	b	a	b	a	b	c	a									
									a	b	a	b	a	b	a	b	c	a

$\pi(9)=0$

Prefix function π

	1	2	3	4	5	6	7	8	9	10
pattern P	a	b	a	b	a	b	a	b	c	a

P_8

1	2	3	4	5	6	7	8	9	10		
a	b	a	b	a	b	a	b	c	a		
		a	b	a	b	a	b	a	b	c	a

$\pi(8)=6$

P_7

1	2	3	4	5	6	7	8	9	10		
a	b	a	b	a	b	a	b	c	a		
		a	b	a	b	a	b	a	b	c	a

$\pi(7)=5$

Prefix function π

	1	2	3	4	5	6	7	8	9	10
pattern P	a	b	a	b	a	b	a	b	c	a

P_6

1	2	3	4	5	6	7	8	9	10		
a	b	a	b	a	b	a	b	c	a		
		a	b	a	b	a	b	a	b	c	a

$\pi(6)=4$

P_5

1	2	3	4	5	6	7	8	9	10		
a	b	a	b	a	b	a	b	c	a		
		a	b	a	b	a	b	a	b	c	a

$\pi(5)=3$

Prefix function π

pattern P

1	2	3	4	5	6	7	8	9	10
a	b	a	b	a	b	a	b	c	a

P_4

1	2	3	4	5	6	7	8	9	10		
a	b	a	b	a	b	a	b	c	a		
		a	b	a	b	a	b	a	b	c	a

$\pi(4)=2$

P_3

1	2	3	4	5	6	7	8	9	10		
a	b	a	b	a	b	a	b	c	a		
		a	b	a	b	a	b	c	a		

$\pi(3)=1$

Prefix function π

pattern P

1	2	3	4	5	6	7	8	9	10
a	b	a	b	a	b	a	b	c	a

P_2

1	2	3	4	5	6	7	8	9	10		
a	b	a	b	a	b	a	b	c	a		
		a	b	a	b	a	b	a	b	c	a

$\pi(2)=0$

P_1

1	2	3	4	5	6	7	8	9	10	
a	b	a	b	a	b	a	b	c	a	
	a	b	a	b	a	b	a	b	c	a

$\pi(1)=0$

Example

P	A	B	A	B	A	B	C	B
π	0	0	1	2	3	4	0	0

T	A	B	C	B	A	B	A	B	A	B	A	B	C	B	
	A	B	A	B	A	B	C	B							
			A	B	A	B	A	B	C	B					
				A	B	A	B	A	B	C	B				
					A	B	A	B	A	B	C	B			
							A	B	A	B	A	B	C	B	match is found!

Case 1

First character of the pattern does not match character of the text. ($P[1] \neq T[i]$)

➔ Shift pattern by 1

== increment text index by 1

(no change of pattern index)

T A B C B A B A B A B A B C B

A B A B A B C B

A B A B A B C B

Case 2

Character (other than first) of the pattern does not match character of the text. (P[q+1] ≠ T[i])

➔ Shift pattern by value π value

(Prefix of P has already been compared.)

Current text index

T	A	B	C	B	A	B	A	B	A	B	A	B	C	B
					A	B	A	B	A	B	C	B		
							A	B	A	B	A	B	C	B

π(6)=4

4

Case 3

Character of the pattern matches character of the text. (P[q+1] = T[i])

➔ Increment text index and pattern index by 1.

T	A	B	C	B	A	B	A	B	A	B	A	B	C	B
					A	B	A	B	A	B	C	B		

And if all patterns are matched

➔ match is found.

KMP Algorithm

Case 1 : When pattern does not match the text
1) first character of pattern is compared with the character of text again and fail (q==0)
2) then increase text here ➔ increase text index only

KMP-MATCHER(T, P)
  n = $length[T]$, m = $length[P]$,
  π = COMPUTE-PREFIX-FUNCTION(P)
  q = 0   % number of characters matched
  for i = 1 to n % scan the text from left to right
    while q > 0 and $P[q+1] \neq T[i]$
      do $q = \pi[q]$ % next character does not match   Case 2
    if $P[q+1] = T[i]$
      then $q = q+1$ % next character matches   Case 3
    if $q = m$ % is all of P matched?
      then   print "Pattern occurs with shift" i-m
          $q = \pi[q]$ % look for the next match

Running-time

- Using the amortized analysis (Sec 17.3)

  COMPUTE-PREFIX-FUNCTION : $\Theta(m)$,

  KMP-MATCHER : $\Theta(n)$

Exercise

Apply the 'KMP-Matcher(T, P)' for previous example.

q	*i*
0	1
1	1
1	2