String Matching

Finding all occurrences of a pattern in the text.

Applications

- Spell Checkers.
- Search Engines.
- Spam Filters.
- Intrusion Detection System.
- Plagiarism Detection.
- Bioinformatics – DNA Sequencing.
- Digital Forensics.
- Information Retrieval, etc.

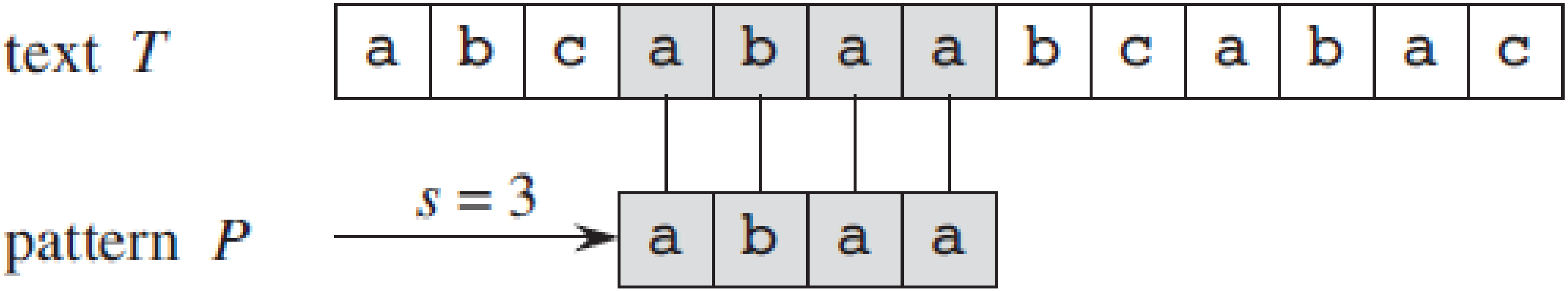
String Matching Problem

- Let,
- Text $T\,[1..n]$ is an array of length n.
- Pattern $P\,[1..m]$ is an array of length $m \leq n$.
- P and T are drawn from a finite alphabet Σ.
  - $\Sigma = \{0,1\}$ or $\Sigma = \{a, b, \ldots, z\}$.
- Example:
  - T = a b c a b a a b c a b a c
  - P = a b a a

Contd…

	1	2	3	4	5	6	7	8	9	10	11	12	13
T	a	b	c	a	b	a	a	b	c	a	b	a	c
shift = 0 ✗ P	a	b	a	a									
shift = 1 ✗ P		a	b	a	a								
shift = 2 ✗ P			a	b	a	a							
shift = 3 ✓ P				a	b	a	a						
shift = 4 ✗ P					a	b	a	a					
shift = 5 ✗ P						a	b	a	a				
shift = 6 ✗ P							a	b	a	a			
shift = 7 ✗ P								a	b	a	a		
shift = 8 ✗ P									a	b	a	a	
shift = 9 ✗ P										a	b	a	a

Contd…



- Shift s a valid shift, if P occurs with shift s in T.
  - $0 \le s \le n - m$ and $T[s + 1..s + m] = P[1..m]$.
- Otherwise, shift s is an invalid shift.
- String-matching problem means finding all valid shifts with which a given pattern P occurs in a given text T.

Naive String Matching Algorithm

NAIVE-STRING-MATCHER(T,P)

$n = T.length$

$m = P.length$

for $s = 0$ to $n - m$

    if $P[1..m] == T[s+1..s+m]$

        print “Pattern occurs with shift” s

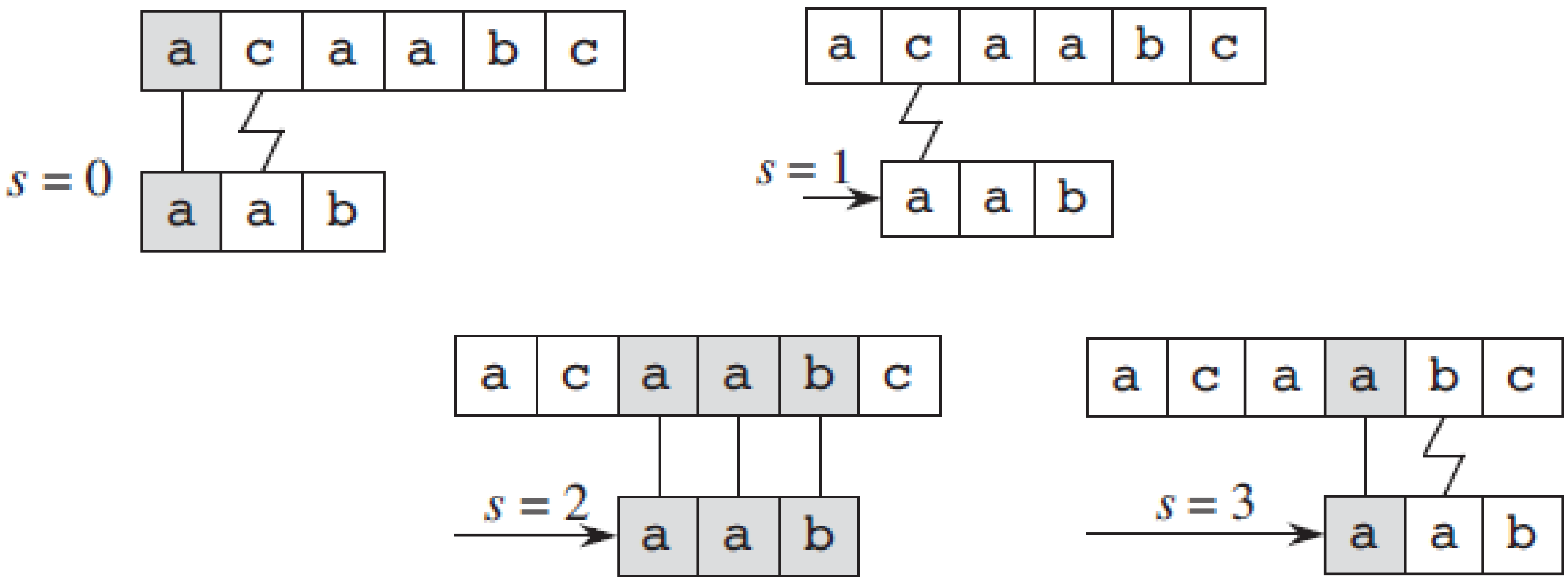
- Complexity: $O((n - m + 1)m)$

Example

NAIVE-STRING-MATCHER(T, P)

```
n = T.length
m = P.length
for s = 0 to n − m
    if P[1..m] == T[s + 1..s + m]
        print "Pattern occurs with shift" s
```

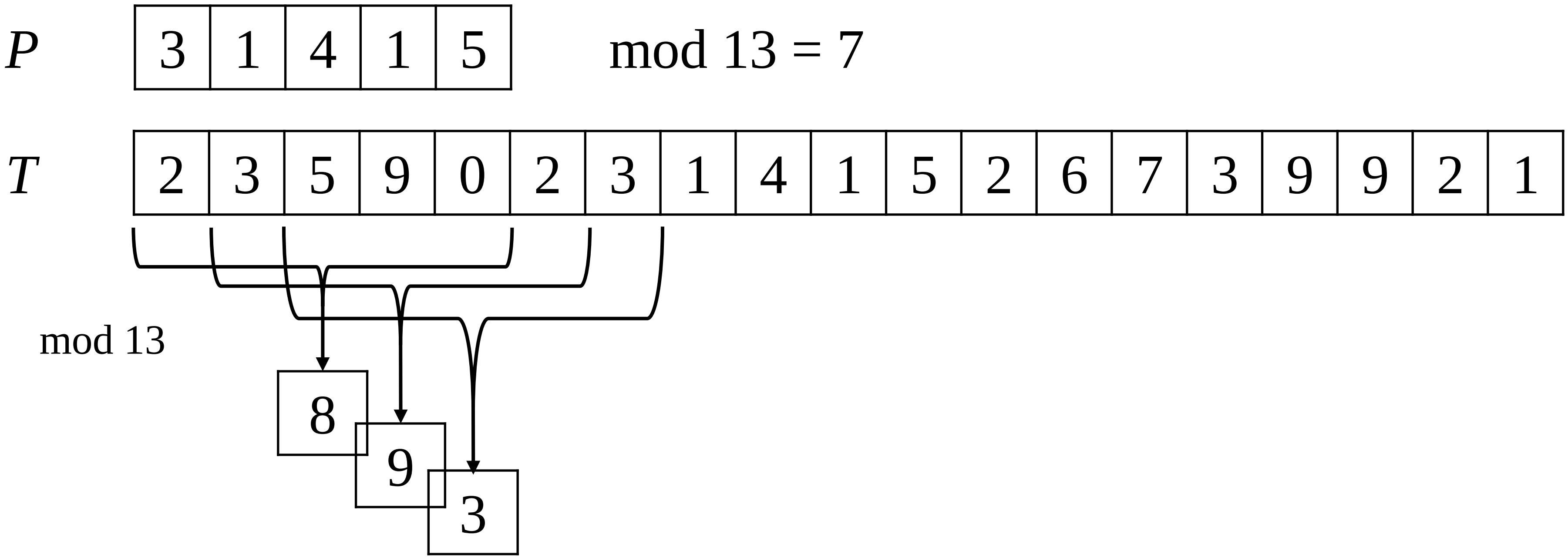
- Text $T = acaabc$, and pattern $P = aab$.

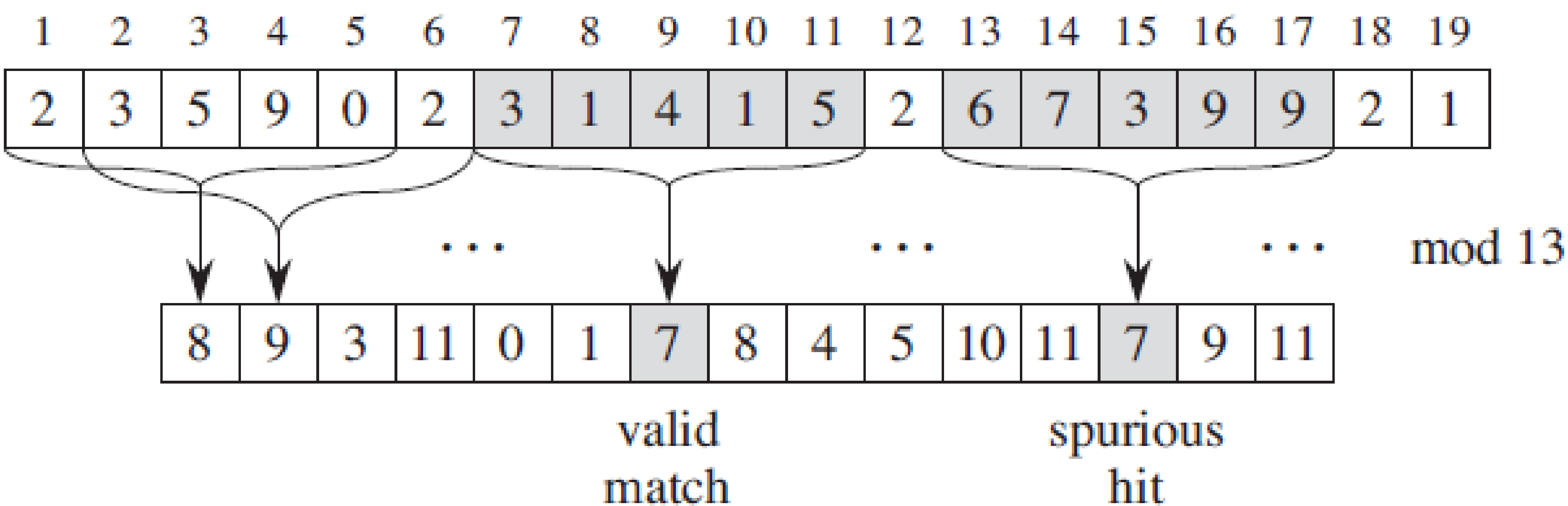


Rabin-Karp Algorithm

- Uses elementary number-theoretic notions.
  - Equivalence of two numbers modulo a third number.
- Let, $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- String of k consecutive characters represents a length-k decimal number.
  - Thus, character string 31415 corresponds to the decimal number 31,415.
- Note:
  - In the general case, each character is a digit in radix-d notation, where $d = |\Sigma|$.)

Example


P
3 1 4 1 5
mod 13 = 7
T
2 3 5 9 0 2 3 1 4 1 5 2 6 7 3 9 9 2 1
mod 13
8
9
3


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
…
…
…
mod 13
8 9 3 11 0 1 7 8 4 5 10 11 7 9 11
valid match
spurious hit

A few calculations…

- For a pattern $P\ [1..m]$, let p denote its corresponding value in radix-d notation.

- Using Horner's rule, p can be computed in time $\Theta(m)$.

$$p = P\ [m] + d(P\ [m-1] + d(P\ [m-2] + \ldots + d(P\ [2] + dP[1])\ \ldots)).$$

- Similarly, for a text $T\ [1..n]$, let t_s denotes the radix-d notation value of the length-m substring $T\ [s+1..s+m]$, for $s = 0, 1, \ldots, n-m$.

- Again, t_0 can be computed from $T\ [1..m]$ in $\Theta(m)$.

Contd…

- Each of the remaining values $t_1, t_2, \ldots, t_{n-m}$ can be computed in constant time.
  - Subtracting $d^{m-1}T[s+1]$ removes the high-order digit from t_s, multiplying the result by d shifts the number left by one digit position, and adding $T[s+m+1]$ brings in the appropriate low-order digit.

$$t_{s+1} = d\,(t_s - d^{m-1}T[s+1]) + T[s+m+1].$$

- Let $h = d^{m-1}$, then

$$t_{s+1} = d\,(t_s - hT[s+1]) + T[s+m+1].$$

- Modulus:
  - p modulo q takes $\Theta(m)$ time.
  - For all t_s, t_s modulo q takes $\Theta(n - m + 1)$ time.

Algorithm

RABIN-KARP-MATCHER(T, P, d, q)

```
n = T.length
m = P.length
h = d^(m-1) mod q
p = 0
t_0 = 0
for i = 1 to m                          // preprocessing
    p = (dp + P[i]) mod q
    t_0 = (dt_0 + T[i]) mod q
for s = 0 to n - m                      // matching
    if p == t_s
        if P[1..m] == T[s+1..s+m]
            print "Pattern occurs with shift" s
    if s < n - m
        t_(s+1) = (d(t_s - T[s+1]h) + T[s+m+1]) mod q
```

Example – 1

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \|\Sigma\| = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 1: $s = 0$, $t_0 = 8$, $p = 7$.

$p == t_0 \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT[s+1]) + T[s+m+1] \pmod{13}$$

$$t_1 = 10\,(8 - 3\,(2)) + 2 \pmod{13}$$

$$t_1 = 10\,(2) + 2 \pmod{13} = 22 \pmod{13} = 9.$$

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \lvert\Sigma\rvert = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 2: $s = 1$, $t_1 = 9$, $p = 7$.

$p == t_1 \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT\,[s + 1]) + T\,[s + m + 1] \pmod{13}$$

$$t_2 = 10\,(9 - 3\,(3)) + 3 \pmod{13}$$

$$t_2 = 10\,(0) + 3 \pmod{13} = 3 \pmod{13} = 3.$$

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \|\Sigma\| = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 3: $s = 2$, $t_2 = 3$, $p = 7$.

$p == t_2 \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT[s + 1]) + T[s + m + 1] \pmod{13}$$

$$t_3 = 10\,(3 - 3\,(5)) + 1 \pmod{13} = 10\,(-12) + 1 \pmod{13}$$

$$t_3 = 10\,(1) + 1 \pmod{13} = 11 \pmod{13} = 11.$$

Because -12 mod 13 = 1

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \|\Sigma\| = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 4: $s = 3$, $t_3 = 11$, $p = 7$.

$p == t_3 \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT\,[s + 1]) + T\,[s + m + 1] \pmod{13}$$

$$t_4 = 10\,(11 - 3\,(9)) + 4 \pmod{13} = 10\,(-16) + 4 \pmod{13}$$

Because -16 mod 13 = 10

$$t_4 = 10\,(10) + 4 \pmod{13} = 104 \pmod{13} = 0.$$

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \lvert\Sigma\rvert = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 5: $s = 4$, $t_4 = 0$, $p = 7$.

$p == t_4 \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT[s+1]) + T[s+m+1] \pmod{13}$$

$$t_5 = 10\,(0 - 3\,(0)) + 1 \pmod{13}$$

$$t_5 = 1 \pmod{13} = 1.$$

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \lvert\Sigma\rvert = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 6: $s = 5$, $t_5 = 1$, $p = 7$.

$p == t_5 \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT[s+1]) + T[s+m+1] \pmod{13}$$

$$t_6 = 10\,(1 - 3\,(2)) + 5 \pmod{13} = 10\,(-5) + 5 \pmod{13}$$

$$t_6 = 10\,(8) + 5 \pmod{13} = 85 \pmod{13} = 7.$$

Because -5 mod 13 = 8

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \lvert\Sigma\rvert = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 7: $s = 6$, $t_6 = 7$, $p = 7$.

$p == t_6 \rightarrow$ Yes.

Character by character matching $p[1..5] == T[7..11]$.

$\{3\ 1\ 4\ 1\ 5\} == \{3\ 1\ 4\ 1\ 5\}$. Match, hence $s = 6$ is a valid shift.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT[s+1]) + T[s+m+1] \pmod{13}$$

$$t_7 = 10\,(7 - 3\,(3)) + 2 \pmod{13} = 10\,(-2) + 2 \pmod{13}$$

$$t_7 = 10\,(11) + 2 \pmod{13} = 112 \pmod{13} = 8.$$

Because -2 mod 13 = 11

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \|\Sigma\| = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 8: $s = 7$, $t_7 = 8$, $p = 7$.

$p == t_7 \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT\,[s + 1]) + T\,[s + m + 1]\ (\text{mod } 13)$$

$$t_8 = 10\,(8 - 3\,(1)) + 6\ (\text{mod } 13)$$

$$t_8 = 10\,(5) + 6\ (\text{mod } 13) = 56\ (\text{mod } 13) = 4.$$

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \|\Sigma\| = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 9: $s = 8$, $t_8 = 4$, $p = 7$.

$p == t_8 \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT[s+1]) + T[s+m+1] \pmod{13}$$

$$t_9 = 10\,(4 - 3\,(4)) + 7 \pmod{13} = 10\,(-8) + 7 \pmod{13}$$

$$t_9 = 10\,(5) + 7 \pmod{13} = 57 \pmod{13} = 5.$$

Because -8 mod 13 = 5

Contd...

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \lvert\Sigma\rvert = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 10: $s = 9$, $t_9 = 5$, $p = 7$.

$p == t_9 \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT[s+1]) + T[s+m+1] \pmod{13}$$

$$t_{10} = 10\,(5 - 3\,(1)) + 3 \pmod{13}$$

$$t_{10} = 10\,(2) + 3 \pmod{13} = 23 \pmod{13} = 10.$$

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \|\Sigma\| = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 11: $s = 10$, $t_{10} = 10$, $p = 7$.

$p == t_{10} \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT\,[s + 1]) + T\,[s + m + 1]\ (\text{mod } 13)$$

$$t_{11} = 10\,(10 - 3\,(5)) + 9\ (\text{mod } 13) = 10\,(-5) + 9\ (\text{mod } 13)$$

$$t_{11} = 10\,(8) + 9\ (\text{mod } 13) = 89\ (\text{mod } 13) = 11.$$

Because -5 mod 13 = 8

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \|\Sigma\| = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 12: $s = 11$, $t_{11} = 11$, $p = 7$.

$p == t_{11} \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT\,[s + 1]) + T\,[s + m + 1]\ (\text{mod } 13)$$

$$t_{12} = 10\,(11 - 3\,(2)) + 9\ (\text{mod } 13)$$

$$t_{12} = 10\,(5) + 9\ (\text{mod } 13) = 59\ (\text{mod } 13) = 7.$$

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \|\Sigma\| = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 13: $s = 12$, $t_{12} = 7$, $p = 7$.

$p == t_{12} \rightarrow$ Yes.

Character by character matching p[1..5] == T[13..17].

{3 1 4 1 5} == {6 7 3 9 9}. Mismatch occurs at first character.

$s < 14 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT\,[s + 1]) + T\,[s + m + 1] \pmod{13}$$

$$t_{13} = 10\,(7 - 3\,(6)) + 2 \pmod{13} = 10\,(-11) + 2 \pmod{13}$$

$$t_{13} = 10\,(2) + 2 \pmod{13} = 22 \pmod{13} = 9.$$

Because -11 mod 13 = 2

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \|\Sigma\| = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 14: $s = 13$, $t_{13} = 9$, $p = 7$.

$p == t_{13} \rightarrow$ No.

$s < 14 \rightarrow$ Yes.

$t_{s+1} = d\,(t_s - hT[s+1]) + T[s+m+1] \pmod{13}$

$t_{14} = 10\,(9 - 3\,(7)) + 1 \pmod{13} = 10\,(-12) + 1 \pmod{13}$

$t_{14} = 10\,(1) + 1 \pmod{13} = 11 \pmod{13} = 11.$

Because -12 mod 13 = 1

Contd…

Index	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
Pattern: 3 1 4 1 5																			
$\Sigma = \{0, 1, \ldots, 9\}$ $d = \lvert\Sigma\rvert = 10$ $q = 13$																			

- $n = 19$, $m = 5$, $n - m = 14$.
- $h = 10^{5-1} \bmod 13 = 10^4 \bmod 13 = 3$.
- $p = (3 \times 10^4 + 1 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 5 \times 10^0) \bmod 13 = 7$.
- $t_0 = (2 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 0 \times 10^0) \bmod 13 = 8$.

Step 15: $s = 14$, $t_{14} = 11$, $p = 7$.
$p == t_{14} \rightarrow$ No.
$s < 14 \rightarrow$ No.

Step 16: $s = 15$.
Loop terminates.

Example – 2

Index	1	2	3	4	5	6	7	8
Text (T)	a	a	b	b	c	a	b	a
ASCII	97	97	98	98	99	97	98	97
Pattern: c a b								
$\Sigma = \{a, b, \ldots, z\}$ $d = \|\Sigma\| = 26$ $q = 3$								

- $n = 8$, $m = 3$, $n - m = 5$.
- $h = 26^{3-1} \bmod 3$
  $= 26^2 \bmod 3 = 1.$
- $p = (99 \times 26^2 + 97 \times 26^1 + 98 \times 26^0) \bmod 3 = 1.$
- $t_0 = (97 \times 26^2 + 97 \times 26^1 + 98 \times 26^0) \bmod 3 = 2.$

Step 1: $s = 0$, $t_0 = 2$, $p = 1$.

$p == t_0 \rightarrow$ No.

$s < 5 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT[s+1]) + T[s+m+1] \pmod 3$$

$$t_1 = 26\,(2 - 1\,(97)) + 98 \pmod 3$$

$$t_1 = 26\,(2 - 1\,(1)) + 2 \pmod 3$$

(because 97 mod 3 = 1 and 98 mod 3 = 2)

$$= 26\,(1) + 2 \pmod 3$$

$$= 28 \pmod 3 \quad = 1.$$

Contd…

Index	1	2	3	4	5	6	7	8
Text (T)	a	a	b	b	c	a	b	a
ASCII	97	97	98	98	99	97	98	97
Pattern: c a b								

Step 2: $s = 1, t_1 = 1, p = 1.$

$p == t_1 \rightarrow$ Yes.

Character by character matching p[1..3] == T[2..4].

{c a b} == {a b b}. Mismatch occurs at first character.

$s < 5 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT\,[s + 1]) + T\,[s + m + 1] \text{ (mod 3)}$$

$$t_2 = 26\,(1 - 1\,(97)) + 99 \text{ (mod 3)}$$

$$t_2 = 26\,(1 - 1\,(1)) + 0 \text{ (mod 3)}$$

(because 97 mod 3 = 1 and 99 mod 3 = 0)

$$= 26\,(0) \text{ (mod 3)}$$

$$= 0.$$

Contd…

Index	1	2	3	4	5	6	7	8
Text (T)	a	a	b	b	c	a	b	a
ASCII	97	97	98	98	99	97	98	97
Pattern: c a b								

Step 3: $s = 2$, $t_2 = 0$, $p = 1$.

$p == t_2 \rightarrow$ No.

$s < 5 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT\,[s + 1]) + T\,[s + m + 1] \text{ (mod 3)}$$

$$t_3 = 26\,(0 - 1\,(98)) + 97 \text{ (mod 3)}$$

$$t_3 = 26\,(0 - 1\,(2)) + 1 \text{ (mod 3)}$$

(because 97 mod 3 = 1 and 98 mod 3 = 2)

$$= 26\,(0 - 2) + 1 \text{ (mod 3)}$$

$$= 26\,(0 + 1) + 1 \text{ (mod 3)}$$

(because 3's complement of -2 = 1)

$$= 26\,(1) + 1 \text{ (mod 3)} \quad = 27 \text{ (mod 3)} \quad = 0.$$

Contd…

Index	1	2	3	4	5	6	7	8
Text (T)	a	a	b	b	c	a	b	a
ASCII	97	97	98	98	99	97	98	97
Pattern: c a b								

Step 4: s = 3, t_3 = 0, p = 1.

$p == t_3 \rightarrow$ No.

$s < 5 \quad \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT\,[s + 1]) + T\,[s + m + 1] \pmod 3$$

$$t_4 = 26\,(0 - 1\,(98)) + 98 \pmod 3$$

$$t_4 = 26\,(0 - 1\,(2)) + 2 \pmod 3$$

(because 98 mod 3 = 2)

$$= 26\,(0 - 2) + 2 \pmod 3$$

$$= 26\,(0 + 1) + 2 \pmod 3$$

(because 3's complement of -2 = 1)

$$= 26\,(1) + 2 \pmod 3 \quad = 28 \pmod 3 \quad = 1.$$

Contd…

Index	1	2	3	4	5	6	7	8
Text (T)	a	a	b	b	c	a	b	a
ASCII	97	97	98	98	99	97	98	97
Pattern: c a b								

Step 5: $s = 4$, $t_4 = 1$, $p = 1$.

$p == t_4 \rightarrow$ Yes.

Character by character matching p[1..3] == T[5..7].

{c a b} == {c a b}. Match, hence $s = 4$ is a valid shift.

$s < 5 \rightarrow$ Yes.

$$t_{s+1} = d\,(t_s - hT\,[s+1]) + T\,[s+m+1] \pmod 3$$

$$t_5 = 26\,(1 - 1\,(99)) + 97 \pmod 3$$

$$t_5 = 26\,(1 - 1\,(0)) + 1 \pmod 3$$

(because 97 mod 3 = 1 and 99 mod 3 = 0)

$$= 26\,(1 - 0) + 1 \pmod 3$$

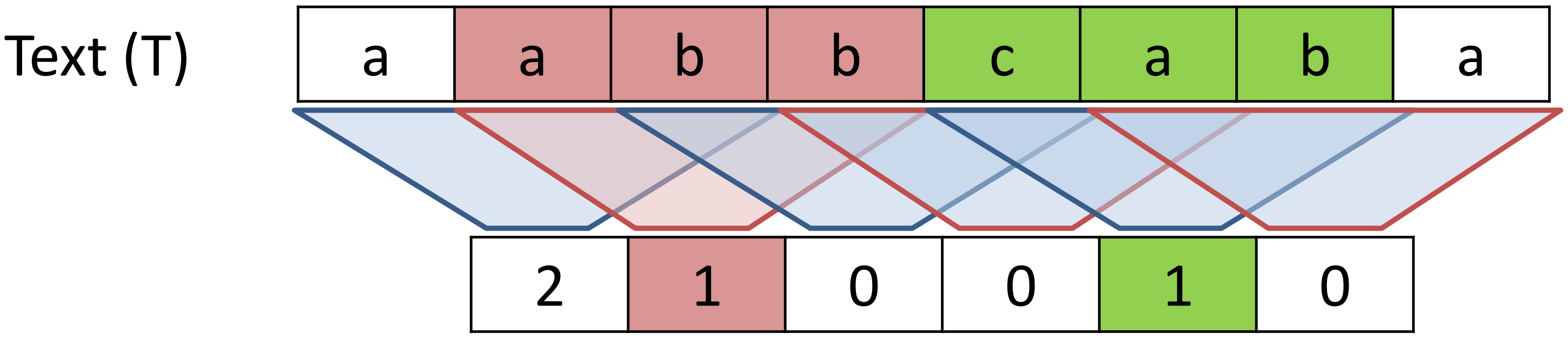
$$= 26\,(1) + 1 \pmod 3 \quad = 27 \pmod 3 \quad = 0.$$

Contd…

Index	1	2	3	4	5	6	7	8
Text (T)	a	a	b	b	c	a	b	a
ASCII	97	97	98	98	99	97	98	97
Pattern: c a b								

Step 6: $s = 5$, $t_5 = 0$, $p = 1$.
$p == t_5 \rightarrow$ No.
$s < 5 \quad \rightarrow$ No.

Step 7: $s = 6$.
Loop terminates.



Complexity

- Takes Ө(m) preprocessing time.
- Worst-case running time is O(m (n – m + 1)).
  - Example: P = a^m and T = a^n, each of the [n – m + 1] possible shifts is valid.
- In many applications, there are a few valid shifts (say some constant c). In such applications, the expected matching time is only O(n – m + 1) + cm), plus the time required to process spurious hits.

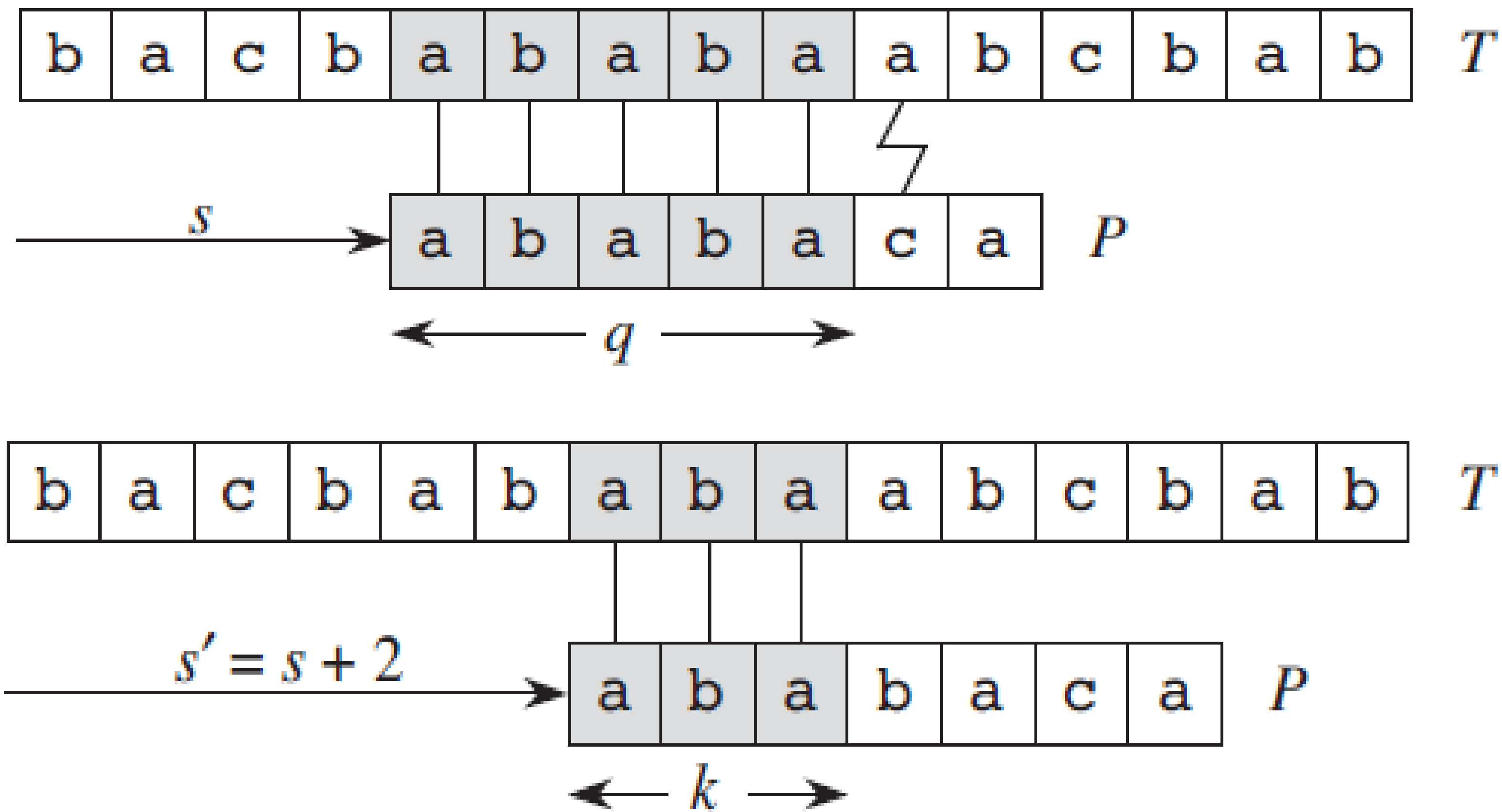
Contd…

- Probabilistic analysis
  - The probability of a false positive hit for a random input is $1/q$.
  - The expected number of false positive hits is $O(n/q)$.
  - The expected run time is $O(n) + O(m(v + n/q)))$, if v is the number of valid shifts.
- Choosing $q \geq m$ and having only a constant number of hits, then the expected matching time is $O(n + m)$.
- Since $m \leq n$, this expected matching time is $O(n)$.

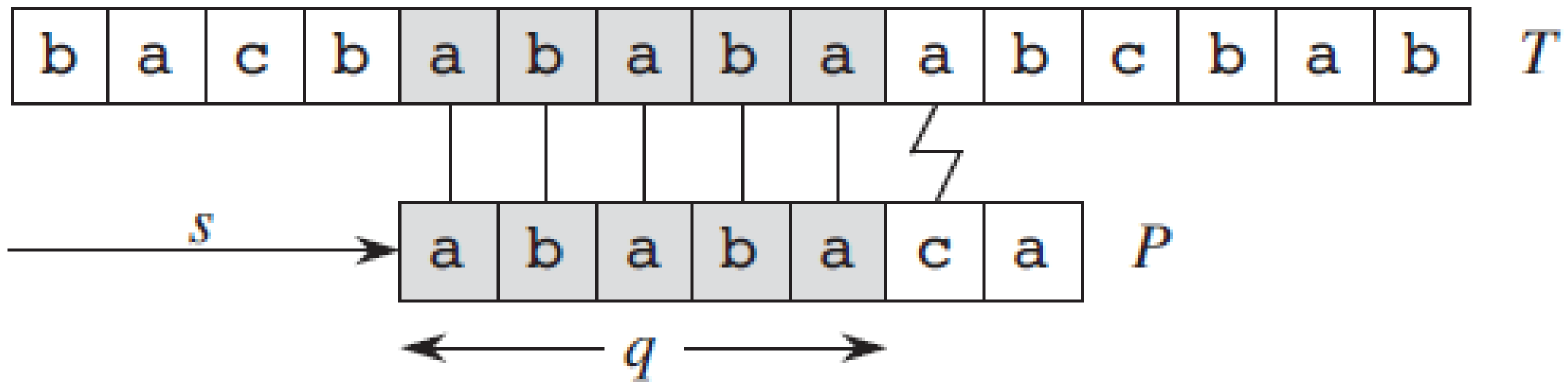
Knuth-Morris-Pratt Algorithm

- Based on the concept of prefix function for a pattern.
  - Encapsulates knowledge about how the pattern matches against shifts of itself.
  - This information can be used to avoid testing of invalid shifts.
- T: b a c b a b a b a a b c b a b
- P: a b a b a c a

Contd…


b a c b a b a b a a b c b a b
T
s
a b a b a c a
P
q
b a c b a b a b a a b c b a b
T
s′ = s + 2
a b a b a c a
P
k

Contd…

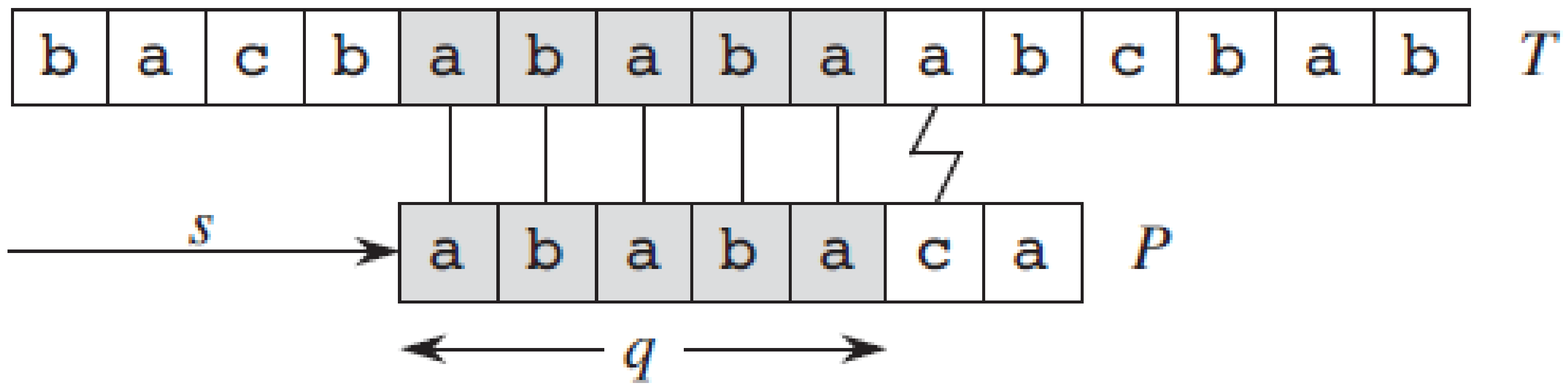


- Given that pattern characters P [1..q] match text characters T [s + 1..s + q], what is the least shift
  s' > s such that for some k < q,
  P [1..k] = T[s' + 1..s' + k], where s' + k = s + q?
  i.e. s' = s + (q – k)
- Best case k = 0. Skips (q – 1) shifts.

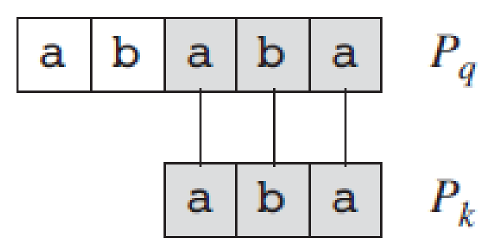
Contd…

Note: P_k = P[1..k]. Similarly, P_q = P[1..q].

- In other words, knowing that P_q is a suffix of T_{s+q}, find the longest proper prefix P_k of P_q that is also a suffix of T_{s+q}.



- Since suffix P_k of T_{s+q} should also be suffix of P_q. Thus, an equivalent statement is "determine the greatest k < q, such that P_k is a suffix of P_q".



Prefix Function

q	1	2	3	4	5	6	7
k	0	0	1	2			

- P: a b a b a c a
- P_q = P[1..q]
- P_1 = {a}. P_k = {}. No matching prefix and suffix of P_1.
- P_2 = {a b}. P_k = {}. No matching prefix and suffix of P_2.
- P_3 = {a b a}. P_k = {a}.

Prefix	**Suffix**	**k**
a	a	1
a b	b a	2

P_4 = {a b a b}. P_k = {a b}.

Prefix	**Suffix**	**k**
a	b	1
a b	a b	2
a b a	b a b	3

Contd…

q	1	2	3	4	5	6	7
k	0	0	1	2	3	0	1

- P: a b a b a c a
- P_q = P[1..q]
- P_5 = {a b a b a}. P_k = {a b a}.
  Keep the longest.

Prefix	**Suffix**	**k**
a	a	1
a b	b a	2
a b a	a b a	3
a b a b	b a b a	4

- P_7 = {a b a b a c a}. P_k = {a}.

Prefix	**Suffix**	**k**
a	a	1
a b	c a	2
a b a	a c a	3
a b a b	b a c a	4
a b a b a	a b a c a	5
a b a b a c	b a b a c a	6

- P_6 = {a b a b a c}. P_k = {}. No matching prefix and suffix of P_6 as 'c' does not appear in any of the proper prefix.

Prefix Function

	i	1	2	3	4	5	6	7
$q \rightarrow$								
	$P[i]$	a	b	a	b	a	c	a
$k \rightarrow$	$\pi[i]$	0	0	1	2	3	0	1

- Given a pattern P [1..m], the prefix function for the pattern P is the function $\Pi : \{1, 2, \ldots, m\} \rightarrow \{0, 1, \ldots, m - 1\}$ such that

$$\Pi[q] = \max\{k : k < q \text{ and } P_k \text{ is a proper suffix of } P_q.$$

- It contains the length of the longest prefix of P that is a proper suffix of P_q.

Contd…

```
COMPUTE-PREFIX-FUNCTION(P)
m = P.length
let π[1..m] be a new array
π[1] = 0
k = 0
for q = 2 to m
    while k > 0 and P[k + 1] ≠ P[q]
        k = π[k]
    if P[k + 1] == P[q]
        k = k + 1
    π[q] = k
return π
```

KMP Algorithm

```
KMP-MATCHER(T, P)
n = T.length
m = P.length
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] ≠ T[i]
        q = π[q]                           // next character does not match
    if P[q + 1] == T[i]
        q = q + 1                          // next character matches
    if q == m                              // is all of P matched?
        print "Pattern occurs with shift" i − m
        q = π[q]                           // look for the next match
```

Complexity

- The COMPUTE-PREFIXFUNCTION runs in Ɵ(m) time as the while loop in lines 6–7 executes at most m – 1 times altogether.
  1. Line 4 starts k at 0, and the only way to increase k is the increment operation in line 9, which executes at most once per iteration of the for loop of lines 5–10. Thus, the total increase in k is at most m – 1.
  2. Second, since k < q upon entering the for loop and each iteration of the loop increments q and k < q always. Assignments in lines 3 and 10 ensure that Π[q] < q for all q = 1, 2, …, m, which means that each iteration of the while loop decreases k.
  3. Third, k never becomes negative.
  – Altogether, the total decrease in k from the while loop is bounded from above by the total increase in k over all iterations of the for loop, which is m – 1.
- Using similar analysis, the matching time of KMP-MATCHER is Ɵ(n).

Example (Preprocessing)

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

- m = 7, Π[1] = 0, k = 0.

Step 1: q = 2, k = 0 P[k + 1] == P[q]

(if) P[1] == P[2]. Mismatch.

Π[2] = 0.

Step 2: q = 3, k = 0. P[k + 1] == P[q]

(if) P[1] == P[3]. Match. k++ = 1

Π[3] = 1.

Step 3: q = 4, k = 1. P[k + 1] == P[q]

(if) P[2] == P[4]. Match. k++ = 2

Π[4] = 2.

Contd…

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 4: q = 5, k = 2 P[k + 1] == P[q]

(if) P[3] == P[5]. Match. k++ = 3

Π[5] = 3.

Step 5: q = 6, k = 3. P[k + 1] == P[q]

(while) P[4] == P[6]. Mismatch. k = Π[k] = 1

(while) P[2] == P[6]. Mismatch. k = Π[k] = 0

(if) P[1] == P[6]. Mismatch.

Π[6] = 0.

Step 6: q = 7, k = 0. P[k + 1] == P[q]

(if) P[1] == P[7]. Match. k++ = 1

Π[7] = 1.

Step 7: q = 8 **Loop terminates.**

Contd...
(Matching)

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

- T:

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

- P:

a	b	a	b	a	c	a

Step 1: i = 1, q = 0 P[q + 1] == T[i]

(if) P[1] == T[1]. Mismatch.

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

a	b	a	b	a	c	a

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 2: i = 2, q = 0 P[q + 1] == T[i]

(if) P[1] == T[2]. Match. q++ = 1

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

a	b	a	b	a	c	a

Contd...

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 3: i = 3, q = 1 P[q + 1] == T[i]

(while) P[2] == T[3]. Mismatch.

q = Π[q] = 0

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

a	b	a	b	a	c	a

(if) P[1] == T[3]. Mismatch.

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

a	b	a	b	a	c	a

Contd…

$n = 15$
$m = 7$

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 4: $i = 4, q = 0$ $P[q + 1] == T[i]$

(if) $P[1] == T[4]$. Mismatch.

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

a	b	a	b	a	c	a

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 5: i = 5, q = 0 P[q + 1] == T[i]

(if) P[1] == T[5]. Match. q++ = 1

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

a	b	a	b	a	c	a

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 6: i = 6, q = 1 P[q + 1] == T[i]

(if) P[2] == T[6]. Match. q++ = 2

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

a	b	a	b	a	c	a

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 7: i = 7, q = 2 P[q + 1] == T[i]

(if) P[3] == T[7]. Match. q++ = 3

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

a	b	a	b	a	c	a

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 8: i = 8, q = 3 P[q + 1] == T[i]

(if) P[4] == T[8]. Match. q++ = 4

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

a	b	a	b	a	c	a

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 9: i = 9, q = 4 P[q + 1] == T[i]

(if) P[5] == T[9]. Match. q++ = 5

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

a	b	a	b	a	c	a

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 10: i = 10, q = 5 P[q + 1] == T[i]

(while) P[6] == T[10]. Mismatch. q = Π[q] = 3

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

a	b	a	b	a	c	a

(if) P[4] == T[10]. Match. q++ = 4

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

a	b	a	b	a	c	a

Shifts skipped = 1 and first three characters are not compared. Comparison starts from P[4].

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 11: i = 11, q = 4 P[q + 1] == T[i]

(if) P[5] == T[11]. Match. q++ = 5

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

a	b	a	b	a	c	a

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 12: i = 12, q = 5 P[q + 1] == T[i]

(if) P[6] == T[12]. Match. q++ = 6

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

a	b	a	b	a	c	a

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 13: i = 13, q = 6 P[q + 1] == T[i]

(if) P[7] == T[13]. Match. q++ = 7

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

a	b	a	b	a	c	a

- q == m. Yes.
  - Pattern occurs with shift i – m =13 – 7 = 6.
  - q = Π[q] = 1.

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 14: i = 14, q = 1 P[q + 1] == T[i]

(while) P[2] == T[14]. Mismatch. q = Π[q] = 0

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

a	b	a	b	a	c	a

Shifts skipped = 5 and first character is not compared. Comparison starts from P[2].

(if) P[1] == T[14]. Match. q++ = 1

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

a	b	a	b	a	c	a

Contd…

n = 15
m = 7

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Step 15: i = 15, q = 1 P[q + 1] == T[i]

(if) P[2] == T[15]. Match. q++ = 2

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

a	b	a	b	a	c	a

Step 16: i = 16

Loop terminates.

Boyer-Moore algorithm

- An efficient string searching algorithm that has been the standard benchmark for practical string search literature.
- Developed by Robert S. Boyer and J Strother Moore in 1977
- A simplified version of it or the entire algorithm is often implemented in text editors for the “search” and “substitute” commands.
- The reason is that it works the fastest when the alphabet is large and the pattern is relatively long.

Contd…

- The execution time of the Boyer-Moore algorithm can be sub-linear.
  - It does not need to check every character of the text, but rather skips over some of them.
- The BM algorithm gets faster as the pattern becomes longer.
  - It utilizes the information gained from each unsuccessful attempt to rule out as many positions as possible of the text where the strings cannot match.

The main ideas…

- Scan the pattern backwards (right to left).
  - The strings are matched from the end of P to the start of P, i.e. P[m], P[m-1], …P[1].
- The **bad character shift rule**.
  - Avoids repeating unsuccessful comparisons against a target character (from the text).
- The **good suffix shift rule**.
  - Aligns only matching pattern characters against target characters already successfully matched.
- Either rule works alone, but they are more effective together.

Backward Scan

- Align the start of P with the start of T.
- Characters in P and T are then compared starting at index m in P and k in T, moving backward.
- The comparisons continue until either the beginning of P is reached (which means there is a match) or a mismatch occurs.
- Example:
  - T: This picture shows a nice view of the park.
  - P: future

Contd…

- T: This picture shows a nice view of the park.
- P: future
  1. P[6] == T[12] = e,   2. P[5] == T[11] = r,
  3. P[4] == T[10] = u,   4. P[3] == T[9] = t,
  5. P[2] == T[8] $\longrightarrow$ u $\neq$ c
  - Mismatch. Shift P to the right (relative to the text) and start comparisons again from the right end.
- The length of shift is given by the two rules:
  - The bad character shift rule.
  - The good suffix shift rule.

Bad Character Shift Rule

- Upon mismatch, skip alignments until
  - Mismatch becomes a match, or
  - P moves past the mismatched character.

1. T: G C T T C T G C T A C C T T T T G C G C G C G C G C G G A A

   P: C C T T T T G C (can be a match as C exists at position 2)

2. T: G C T T C T G C T A C C T T T T G C G C G C G C G C G G A A

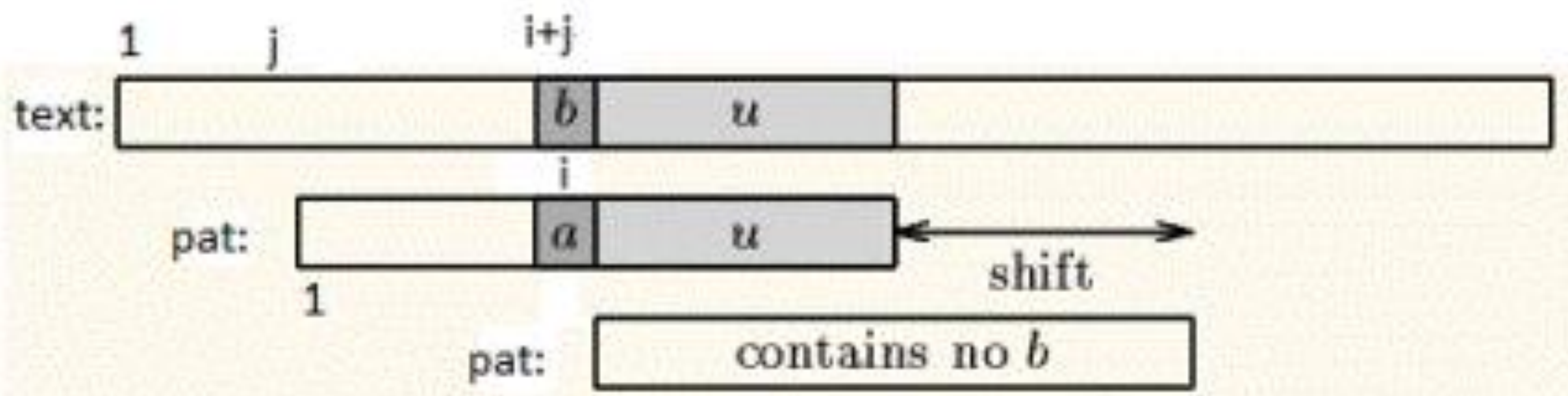
   P: $\xleftrightarrow{s = 3}$ C C T T T T G C (A does not exist in pattern)

3. T: G C T T C T G C T A C C T T T T G C G C G C G C G C G G A A

   P: $\xleftrightarrow{s = 10}$ C C T T T T G C

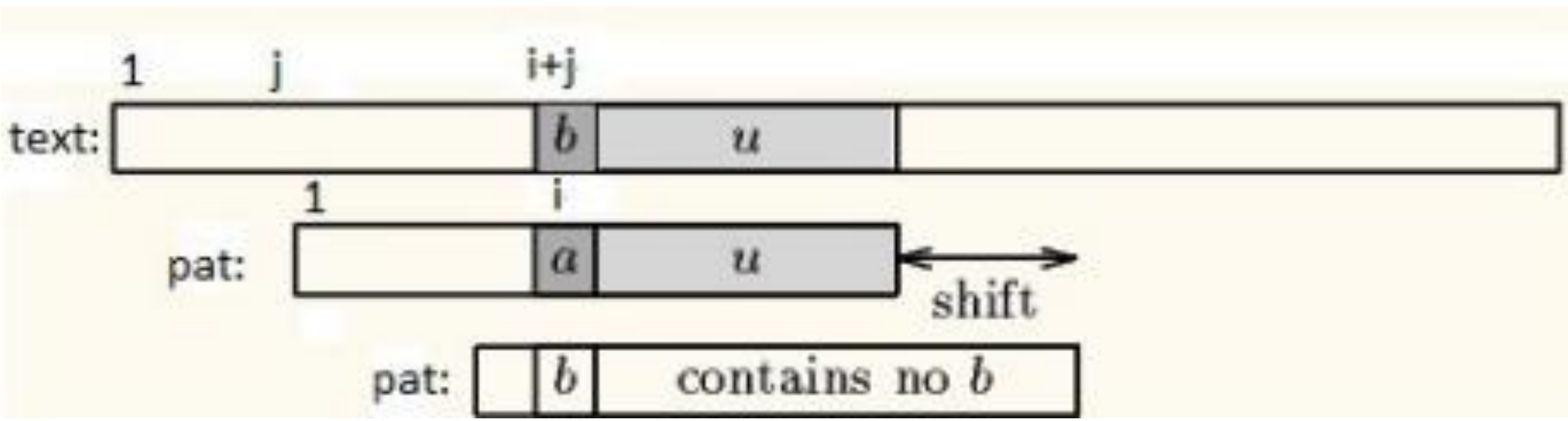
Contd…

- Assume that a mismatch occurs between characters P[i] (= a) and T[i + j] (= b) during an attempt at shift j.
  - That is, P[i + 1 .. m] = T[i + j + 1 .. j + m] = u and P[i] ≠ T[i + j].
- If "b" is not contained anywhere in P, then shift the pattern P completely past T[i + j].



Contd…

- Otherwise, shift the pattern P until the rightmost occurrence of the character “b” in P gets aligned with T[i + j].



Preprocessing

- For all characters x in the alphabet, define the function R(x) to compute the bad character shift.
- R(x) = 0 , if x does not occur in P.

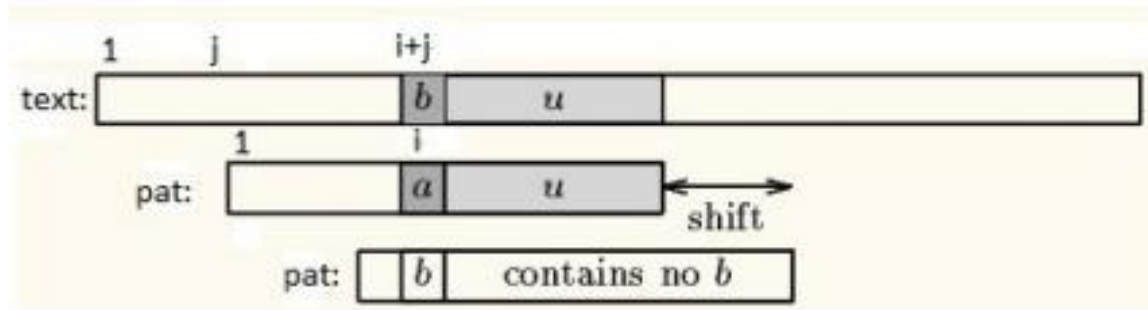
  $\{ i < m \mid P[i] = x\}$, otherwise
- When

  $P[i + 1 .. m] = T[i + j + 1 .. j + m] = u$ and

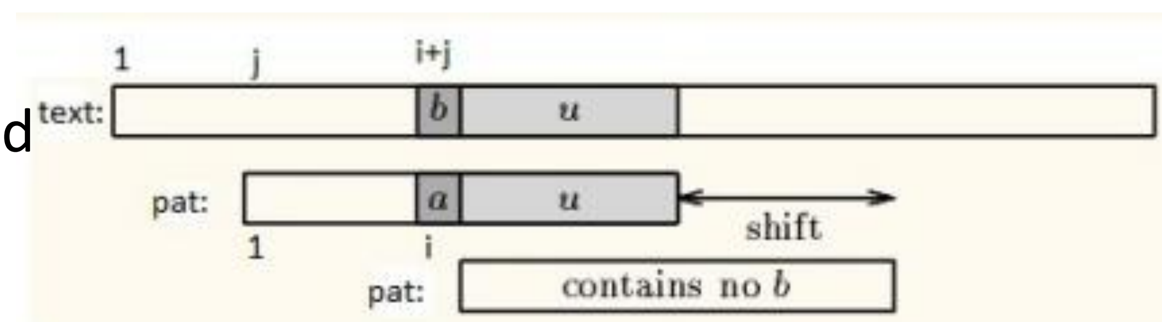
  $P[i] \neq T[i + j]$ and $T[i + j] = b$,
- Then, shift P to the right by max $\{1, i - R(b)\}$.
- R(x) can be computed in O(m) time.

Contd…

If the right-most occurrence of b in P[1..m-1] is at k < i, then P[k] and T[i + j] get aligned.



If b does not occur in P[1..m-1], then shift = i, and the pattern is next aligned with T[i + j + 1.. i + j + m].



Example

Σ	A	C	G	T
R(x)	0	2 → 1	7	6 → 5 → 4 → 3

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
T:	G	C	T	T	C	T	G	C	T	A	C	C	T	T	T	T	G	C	G	C	G	C	G	C	G	C	G	G	A	A
P:	C	C	T	T	T	T	G	C																						

Old Shift = 0. New Shift = 0 + max {1, i - R(b)} = 0 + max {1, 5 – R(C)}
= 0 + max { 1, 5 – 2} = 0 + 3 = 3.

T:	G	C	T	T	C	T	G	C	T	A	C	C	T	T	T	T	G	C	G	C	G	C	G	C	G	C	G	G	A	A
P:				C	C	T	T	T	T	G	C																			

Old Shift = 3. New Shift = 3 + max {1, i - R(b)} = 3 + max {1, 7 – R(A)}
= 3 + max { 1, 7 – 0} = 3 + 7 = 10.

T:	G	C	T	T	C	T	G	C	T	A	C	C	T	T	T	T	G	C	G	C	G	C	G	C	G	C	G	G	A	A
P:											C	C	T	T	T	T	G	C												

Good Suffix Shift Rule

1. T: C G T G C C T A C T T A C T T A C T T A C T T A C G C G A A

   P: C T T A C T T A C

   (TAC can be aligned with the rightmost occurrence TAC)

2. T: C G T G C C T A C T T A C T T A C T T A C T T A C G C G A A

   P: s = 4 C T T A C T T A C

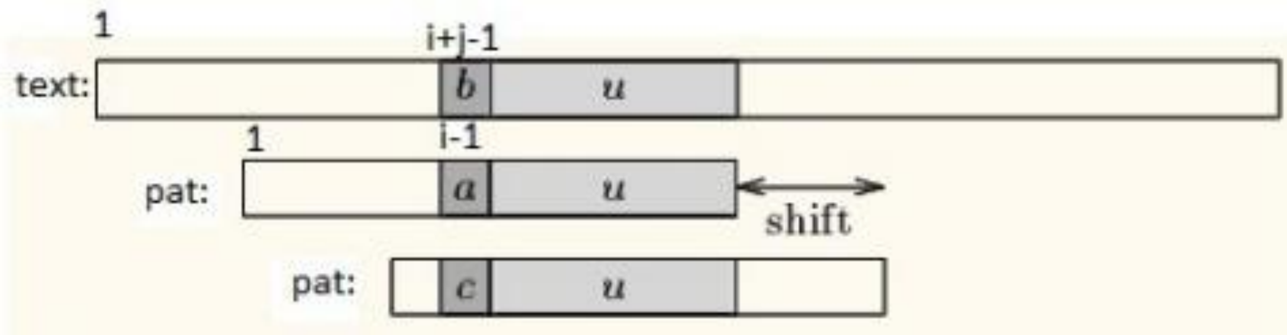
   (TACTTAC has no rightmost occurrence, go for longest prefix-suffix match alignment in between CTTACTTAC and TACTTAC)

3. T: C G T G C C T A C T T A C T T A C T T A C T T A C G C G A A

   P: s = 8 C T T A C T T A C

Contd…

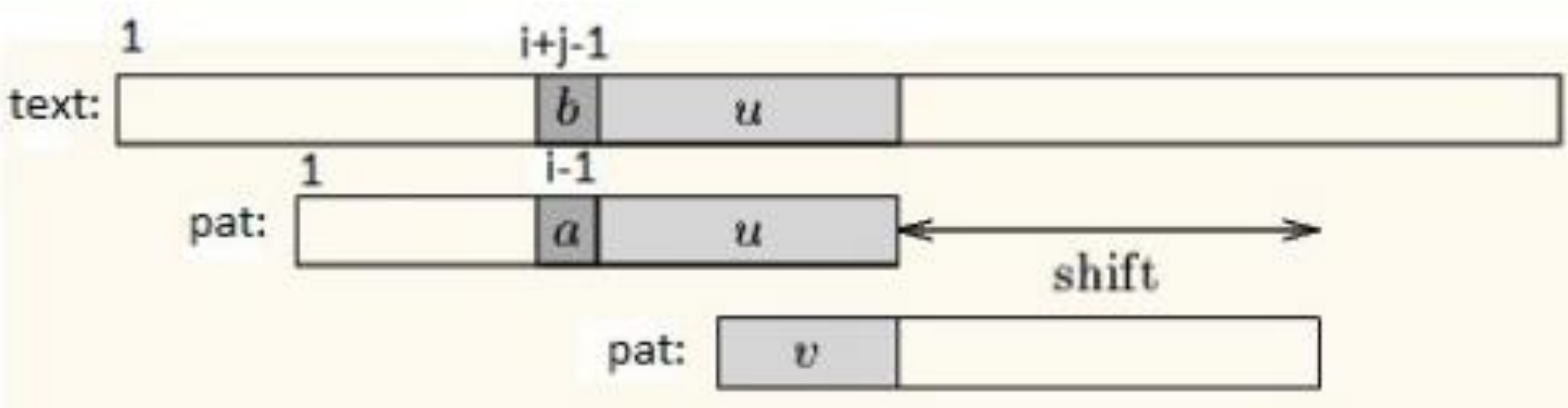
- Let P[i .. m] = T[i + j .. j + m] = u and P[i - 1] ≠ T[i + j - 1].
- The good-suffix shift consists in aligning the segment u = T[i + j .. j + m] (= P[i .. m]) with its rightmost occurrence in pattern (which is not a suffix) that is preceded by a character different from P[i – 1].



- The original weaker version of this rule did not require that the character preceding the rightmost occurrence of u should not be P[i – 1].

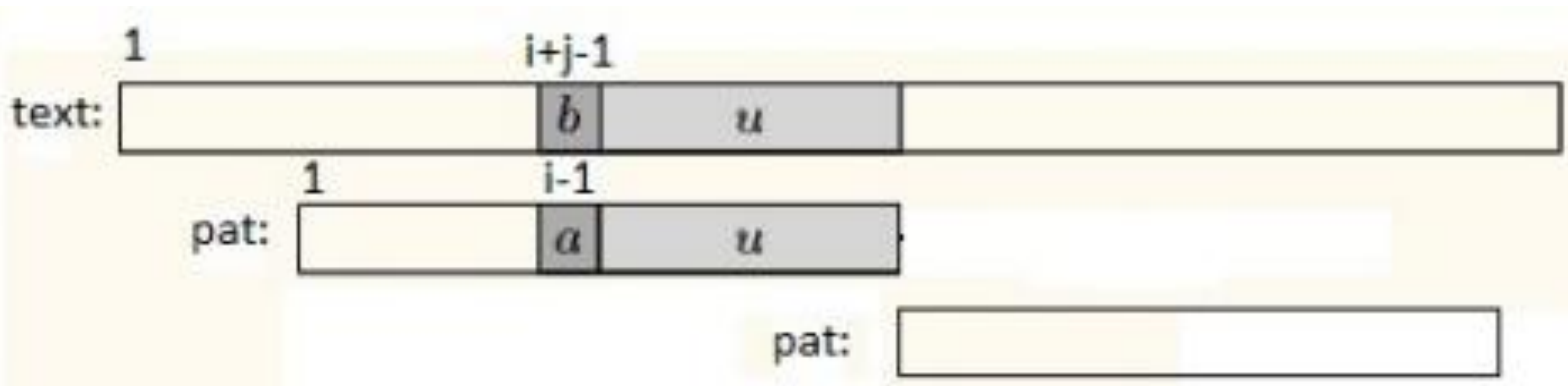
Contd…

- If there exists no such segment, the shift consists in aligning the longest suffix v of T[i + j .. j + m] with a matching prefix of P.



Contd…

- If no such shift is possible, i.e., the longest suffix v of T[i + j .. j + m] which matches a prefix of P, is empty, then shift P to the right by P.length (= m).



Example – 1

1. T: C G T G C C T A C T T A C T T A C T T A C T T A C G C G A A

   P: C T T A C T T A C

(TAC can’t be aligned with the rightmost occurrence TAC as preceding character (T) is same in both the occurrences.)

Thus, the longest prefix-suffix match alignment in between CTTACTTAC and TAC is used to compute the shift increment)

2. T: C G T G C C T A C T T A C T T A C T T A C T T A C G C G A A

   P: C T T A C T T A C

Example – 2

```
        1...3456789......
text: I visited Helsinki by bike and I pedaled a lot.
pat:    pedaled
pat:        pedaled
```

- Align text[7..9] = "ted" with some character sequence "xed" in the pattern, such that x ≠ l.
- In this example the character sequence pat[1..3] = "ped" is the required one, so shift pattern such that pat[1..3] gets aligned with text[7..9].

Example – 3

```
text: When the phone ranged he was disturbed.
pat:                      disturbed
```

- The character sequence "ed" is not appearing again in pattern, so check for longest suffix matching. In this case a single character 'd' is the required match, thus align pat[1] with text[21].

```
text: When the phone ranged he was disturbed.
pat:                              disturbed
```

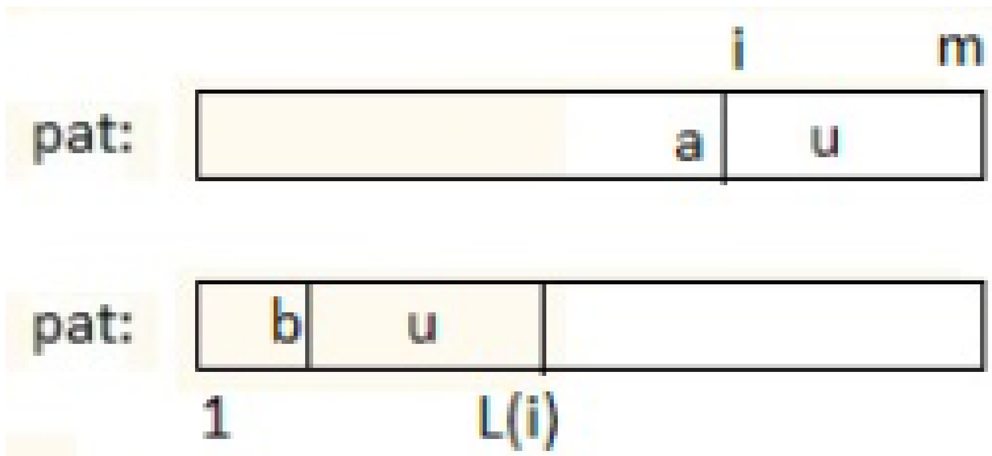
Example – 4

```
text: I travelled by bike, so I pedaled a lot.
pat:      pedaled
pat:                pedaled
```

- In this case,
  - The character sequence "led" is not appearing again in pattern.
  - The longest suffix matching also results in an empty string.
- So shift the pattern completely.

Preprocessing (Compute L(i))

- For i = 2,.., m + 1, compute L(i) as follows:
  - L(i) is the largest position, less than m, such that P[i..m] matches a suffix of P[1..L(i)] and, moreover, this suffix is not preceded by character P[i-1].
- If no such copy of P[i..m] exists, then let L(i) = 0.



Contd… (Compute L(i))

- Since $P[m + 1..m] = \varepsilon$, $L(m + 1)$ is the right-most position j such that $P[j] \neq P[m]$ (or 0 if all characters are equal).
- L(i) gives the right end position of the rightmost copy of $P[i..m]$, which is not a suffix of pattern, and is not preceded by $P[i - 1]$.
- The values L(i) can be computed in O(m) time.

Contd…

- For a given i, the value $L(i) < m$ is computed as follows:
  - Take the suffix $P[i..m] = u$.
  - Search for the rightmost occurrence of u in P such that the preceding letter is not $P[i-1]$.
  - If there is such an occurrence then $L(i)$ = the right-end position of this occurrence of u.
  - Otherwise, $L(i) = 0$

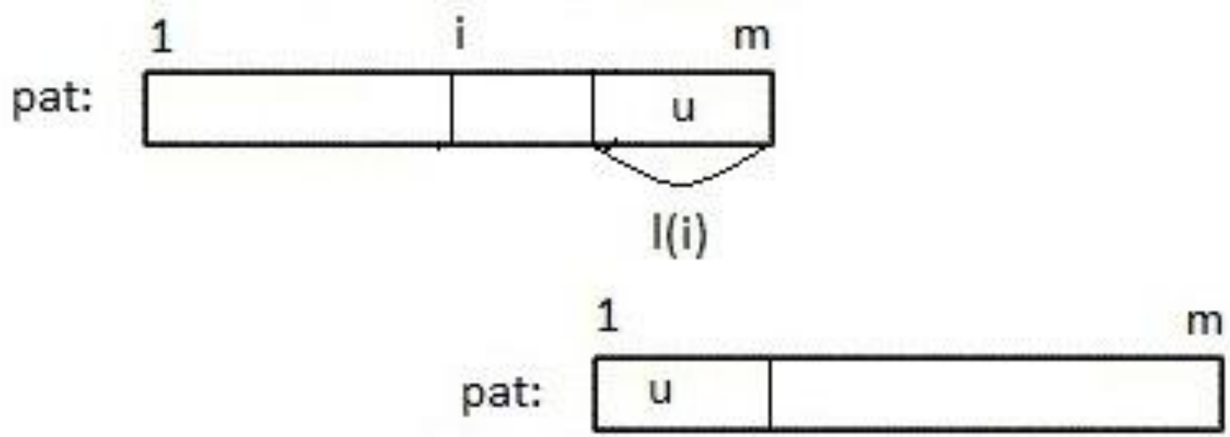
Example

- P: antecedence
- P.length = 11, i = 2, 3, …, 11, 12

- L(12) = 10, P[12..11] = ε, antecedence
- L(11) = 8, P[11] = e, antecedence
- L(10) = 6, P[10..11] = ce, antecedence
- L(9) = 0, P[9..11] = nce antecedence
- L(8) = … = L(2) = 0.

Contd… (Compute l(i))

- For i ≥ 2, let l(i) = the length of the largest suffix of P[i..m] that is also a prefix of pattern, if one exists.
- Otherwise, l(i) = 0.



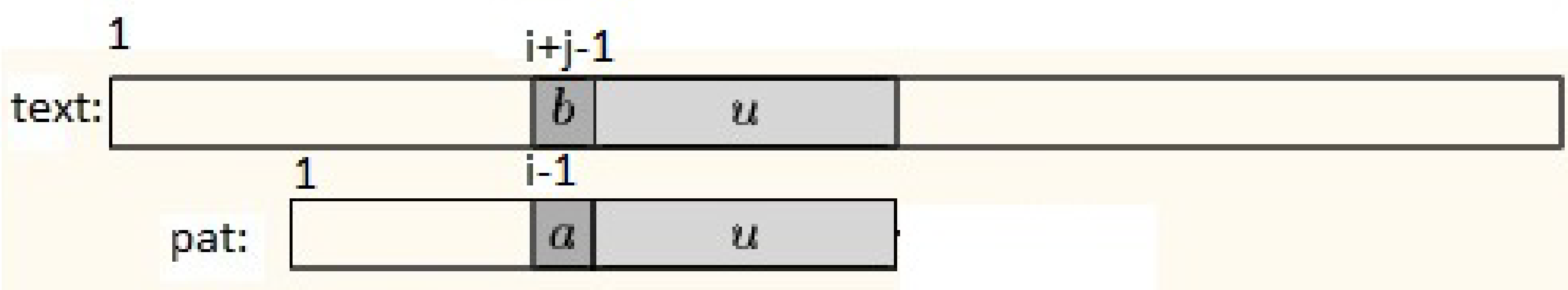
Example

- P: ababa
- P.length = 5, i = 2, 3, 4, 5, 6

- l(6) = 0 (since P[m+1..m] = ε)
- l(5) = 1 (P[5] = a, the largest suffix is 'a')
- l(4) = 1 (P[4..5] = ba, the largest suffix is 'a')
- l(3) = 3 (P[3..5] = aba, the largest suffix is 'aba')
- l(2) = 3 (P[2..5] = baba, the largest suffix is 'aba')

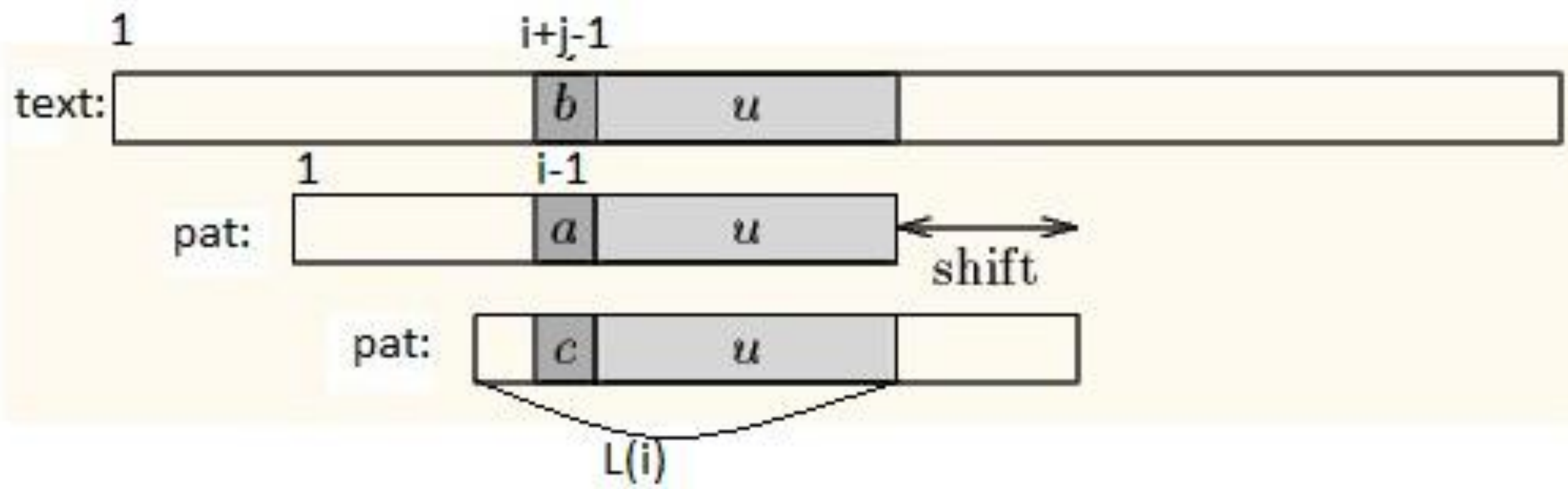
Computing Shifts

- The computed values L(i) and l(i) are used to get the length of a shift using the good suffix rule.
- Suppose, an occurrence of P[i..m] exists but there is a mismatch on P[i – 1]



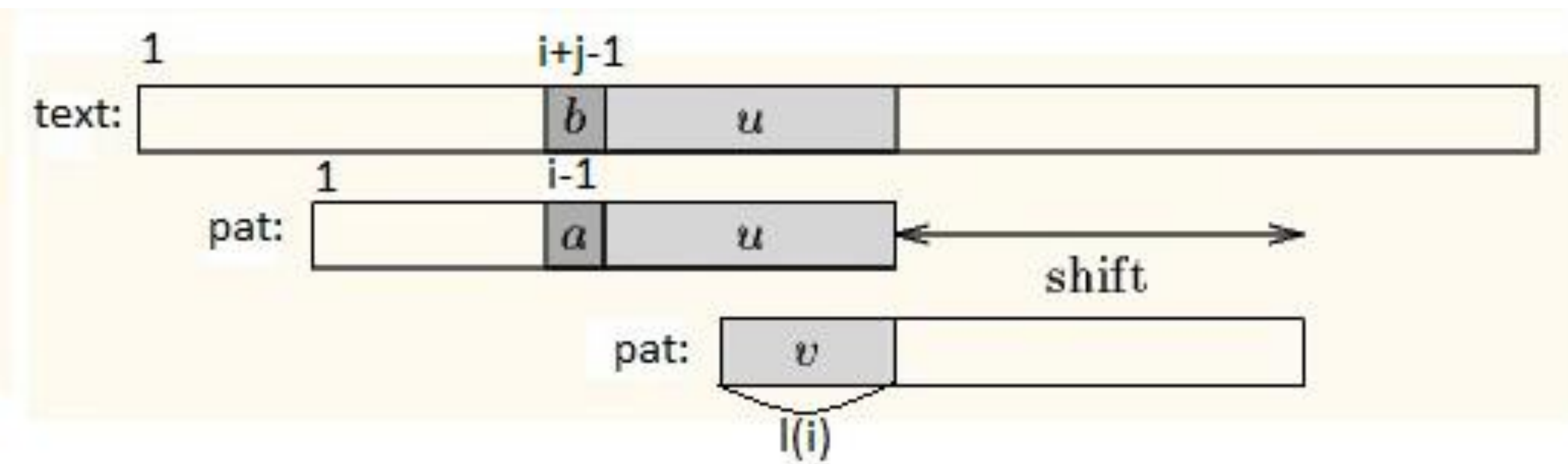
Contd…

- If L(i) > 0, then shift P by m – L(i) characters to the right, i.e., the prefix of length L(i) of the shifted pattern aligns with the suffix of length L(i) of the unshifted pat.



Contd…

- If L(i) = 0, then the good shift rule shifts pattern by m – l(i) characters.



Contd…

- If mismatch occurs for the first comparison, i.e., on P[m], then shift the pattern P by m – L(m + 1) positions to the right.
- When an occurrence of P has been found, then shift pattern to the right by m – l(2) positions, to align a prefix of pattern with the longest matching proper suffix of the occurrence.
  - l(2) = the length of the largest suffix of P[2..m] that is also a prefix of pattern, if one exists.

Boyer Moore Algorithm

- Given a pattern P of length m and text T of length n.
- Pre-processing
  - Compute the values L(i) and l(i) for $2 \leq i \leq m+1$.
  - Also compute R(x) for all characters x from the alphabet.
- Matching
  - Let index k represents the right-end of the current occurrence of pattern that is being matched to the text.
  - Thus, a shift of pattern is implemented by increasing k with the appropriate amount of positions.

Contd…

```
k = m
while k ≤ n do begin
        i = m, h = k
        while i > 0 and P[i] == T[h] do begin
                i - -,  h - -
        end
        if i = 0 then begin
                pattern found at a shift of h
                k = k + m – I(2)
        end
        else
                shift pattern (increase k) by the maximum amount
                determined by the bad character rule and the good suffix
                rule.
        end
end
```

Example – 1

	2	3	4	5	6	7	8	9	10
L:	0	0	0	0	0	6	0	7	8
l:	1	1	1	1	1	1	1	1	0

	A	C	G	T
R:	3	8 → 5	7 → 6 → 4 → 1	2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
T:	G	T	T	A	T	A	G	C	T	G	A	T	C	G	C	G	G	C	G	T	A	G	C	G	G	C	G	A	A
P:	G	T	A	G	C	G	G	C	G																				

Step 1: k = 9. Mismatch at i = 9.
Shift using BC = max{ 1, i – R(T)} = max(1,9-2) = 7.
Shift using GS = m – L(m + 1) = 9 – 8 = 1
Maximum shift = 7.
k = k + 7 = 9 + 7 = 16

Contd...

	A	C	G	T
R:	3	$8 \rightarrow 5$	$7 \rightarrow 6 \rightarrow 4 \rightarrow 1$	2

	2	3	4	5	6	7	8	9	10
L:	0	0	0	0	0	6	0	7	8
l:	1	1	1	1	1	1	1	1	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
T:	G	T	T	A	T	A	G	C	T	G	A	T	C	G	C	G	G	C	G	T	A	G	C	G	G	C	G	A	A
P:								G	T	A	G	C	G	G	C	G													
								1	2	3	4	5	6	7	8	9 ← i													

Step 2: k = 16. Mismatch at i = 6.

Shift using BC = max{ 1, i – R(C)} = max(1,6-5) = 1.

Shift using GS:

As L(i + 1) = L(7) = 6 ≠ 0.

Thus, shift = m – L(i + 1) = 9 – 6 = 3.

Maximum shift = 3.

k = k + 3 = 16 + 3 = 19

Contd...

	A	C	G	T
R:	3	$8 \rightarrow 5$	$7 \rightarrow 6 \rightarrow 4 \rightarrow 1$	2

	2	3	4	5	6	7	8	9	10
L:	0	0	0	0	0	6	0	7	8
l:	1	1	1	1	1	1	1	1	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
T:	G	T	T	A	T	A	G	C	T	G	A	T	C	G	C	G	G	C	G	T	A	G	C	G	G	C	G	A	A
P:											G	T	A	G	C	G	G	C	G										
											1	2	3	4	5	6	7	8	9 ← i										

Step 3: k = 19. Mismatch at i = 3.

Shift using BC = max{ 1, i – R(C)} = max(1,3-0) = 3.

Shift using GS:

As L(i + 1) = L(4) = 0.

Thus, shift = m – l(i + 1) = 9 – 1 = 8.

Maximum shift = 8.

k = k + 8 = 19 + 8 = 27

Contd...

	A	C	G	T
R:	3	$8 \rightarrow 5$	$7 \rightarrow 6 \rightarrow 4 \rightarrow 1$	2

	2	3	4	5	6	7	8	9	10
L:	0	0	0	0	0	6	0	7	8
l:	1	1	1	1	1	1	1	1	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
T:	G	T	T	A	T	A	G	C	T	G	A	T	C	G	C	G	G	C	G	T	A	G	C	G	G	C	G	A	A
P:																			G	T	A	G	C	G	G	C	G		
																			1	2	3	4	5	6	7	8	9 ← i		

Step 4: k = 27. Pattern found at T[k – m + 1..k]. i = 0.
Shift using BC = 1 (pattern found).
Shift using GS = m – l(2) = 9 – 1 = 8.
Maximum shift = 8.
k = k + 8 = 27 + 8 = 35.

Step 5: k > n. 35 > 29. Loop terminates.

Example – 2

	2	3	4	5	6	7	8	9
L:	0	0	0	6	0	4	1	7
l:	1	1	1	1	1	1	1	0

	A	C	G	T
R:	$7 \rightarrow 5 \rightarrow 3$	2	$6 \rightarrow 4 \rightarrow 1$	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	G	C	A	T	C	G	C	A	G	A	G	A	G	T	A	T	A	C	A	G	T	A	C	G
P:	G	C	A	G	A	G	A	G																

Step 1: k = 8. Mismatch at i = 8.

Shift using BC = max{ 1, i – R(A)} = max(1,8-7) = 1.

Shift using GS = m – L(m + 1) = 8 – 7 = 1

Maximum shift = 1.

k = k + 1 = 8 + 1 = 9

Contd…

	A	C	G	T
R:	$7 \rightarrow 5 \rightarrow 3$	2	$6 \rightarrow 4 \rightarrow 1$	0

	2	3	4	5	6	7	8	9
L:	0	0	0	6	0	4	1	7
l:	1	1	1	1	1	1	1	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	G	C	A	T	C	G	C	A	G	A	G	A	G	T	A	T	A	C	A	G	T	A	C	G
P:		G	C	A	G	A	G	A	G															
		1	2	3	4	5	6	7	8 ← i															

Step 2: k = 9. Mismatch at i = 6.

Shift using BC = max{ 1, i – R(C)} = max(1,6-2) = 4.

Shift using GS:

As L(i + 1) = L(7) = 4 ≠ 0.

Thus, shift = m – L(i + 1) = 8 – 4 = 4.

Maximum shift = 4.

k = k + 4 = 9 + 4 = 13

Contd...

	2	3	4	5	6	7	8	9
L:	0	0	0	6	0	4	1	7
l:	1	1	1	1	1	1	1	0

	A	C	G	T
R:	7 → 5 → 3	2	6 → 4 → 1	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	G	C	A	T	C	G	C	A	G	A	G	A	G	T	A	T	A	C	A	G	T	A	C	G
P:						G	C	A	G	A	G	A	G											
						1	2	3	4	5	6	7	8 ← i											

Step 3: k = 13. Pattern found at T[k – m + 1..k]. i = 0.
Shift using BC = 1 (pattern found).
Shift using GS = m – l(2) = 8 – 1 = 7.
Maximum shift = 7.
k = k + 7 = 13 + 7 = 20

Contd...

	A	C	G	T
R:	$7 \rightarrow 5 \rightarrow 3$	2	$6 \rightarrow 4 \rightarrow 1$	0

	2	3	4	5	6	7	8	9
L:	0	0	0	6	0	4	1	7
l:	1	1	1	1	1	1	1	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	G	C	A	T	C	G	C	A	G	A	G	A	G	T	A	T	A	C	A	G	T	A	C	G
P:													G	C	A	G	A	G	A	G				
													1	2	3	4	5	6	7	8 ← i				

Step 4: k = 20. i = 6.

Shift using BC = max{ 1, i – R(C)} = max(1,6-2) = 4.

Shift using GS:

As L(i + 1) = L(7) = 4 ≠ 0.

Thus, shift = m – L(i + 1) = 8 – 4 = 4.

Maximum shift = 4.

k = k + 4 = 20 + 4 = 24

Contd…

	A	C	G	T
R:	7	2	6	0

	2	3	4	5	6	7	8	9
L:	0	0	0	6	0	4	1	7
l:	1	1	1	1	1	1	1	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
T:	G	C	A	T	C	G	C	A	G	A	G	A	G	T	A	T	A	C	A	G	T	A	C	G	
P:																	G	C	A	G	A	G	A	G	
																	1	2	3	4	5	6	7	8	← i

Step 5: k = 24. i = 7.

Shift using BC = max{ 1, i – R(C)} = max(1,7-2) = 5.

Shift using GS:

As L(i + 1) = L(8) = 1 ≠ 0.

Thus, shift = m – L(i + 1) = 8 – 1 = 7.

Maximum shift = 7.

k = k + 7 = 24 + 7 = 31

Step 6: k > n. 31 > 24. Loop terminates.

Complexity

- If BM algorithm uses only the strong good suffix rule, then it has O(n) worst-case time complexity if the pattern does not occur in the text.
- If the pattern does appear in the text, then the algorithm runs in O(mn) worst-case.
- However, by slightly modifying this algorithm, it can achieve O(n) worst-case time complexity in all cases.
  - This was first proved by Galil in 1979.
- It can be proved that the BM algorithm has O(n) time complexity also when it uses both shift rules (Bad character and Good suffix shift rules)
- On natural language texts the running time is almost always sub-linear.

Example

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:	X	Y	X	Y	Y	X	Y	X	Y	X	X													

- Solve it using all the string matching algorithm giving total
  - Number of shifts and
  - Number of character comparisons.

- For Rabin-Karp
  - $\Sigma = \{0, 1\}$, $q = 13$, $X = 0$, $Y = 1$

Naïve approach

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:	X	Y	X	Y	Y	X	Y	X	Y	X	X													
P:		X	Y	X	Y	Y	X	Y	X	Y	X	X												
P:			X	Y	X	Y	Y	X	Y	X	Y	X	X											
P:				X	Y	X	Y	Y	X	Y	X	Y	X	X										
P:					X	Y	X	Y	Y	X	Y	X	Y	X	X									
P:						X	Y	X	Y	Y	X	Y	X	Y	X	X								
P:							X	Y	X	Y	Y	X	Y	X	Y	X	X							
P:								X	Y	X	Y	Y	X	Y	X	Y	X	X						
P:									X	Y	X	Y	Y	X	Y	X	Y	X	X					
P:										X	Y	X	Y	Y	X	Y	X	Y	X	X				
P:											X	Y	X	Y	Y	X	Y	X	Y	X	X			
P:												X	Y	X	Y	Y	X	Y	X	Y	X	X		
P:													X	Y	X	Y	Y	X	Y	X	Y	X	X	
P:														X	Y	X	Y	Y	X	Y	X	Y	X	X

Total shifts
= 24 – 11 + 1 = 14

Total character comparisons
= 4 + 1 + 2 + 5 + 1 + 11 + 1
+ 3 + 1 + 1 + 5 + 1 + 11 + 1
= 48

Rabin–Karp

$\Sigma = \{0, 1\}$, $q = 13$, $X = 0$, $Y = 1$

$h = 2^{11-1} \bmod 13 = 10$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
	0	1	0	0	1	0	1	0	1	1	0	1	0	1	0	1	1	0	1	0	1	0	0	1
P:	X	Y	X	Y	Y	X	Y	X	Y	X	X													
	0	1	0	1	1	0	1	0	1	0	0													

- $p = 9$.
- $t_7 = 9$. Compare T[8..18] with P[1..11].
  - T[8] = P[1] = X
  - T[9] = P[2] = Y
  - T[10] = Y and P[3] = X. Mismatch
- $t_{12} = 9$. Compare T[13..23] with P[1..11].
  - Match.

Total shifts = 24 – 11 + 1 = 14

Total character comparisons = 3 + 11 = 14

- $p = (2^9 + 2^7 + 2^6 + 2^4 + 2^2) \bmod 13 = 9$
- T[1..11] = $t_0 = (2^9 + 2^6 + 2^4 + 2^2 + 2^1) \bmod 13 = 0$
- T[2..12] = $t_1 = (2(0 - 10.0) + 1) \bmod 13 = 1$
- T[3..13] = $t_2 = (2(1 - 10.1) + 0) \bmod 13 = 8$
- T[4..14] = $t_3 = (2(8 - 10.0) + 1) \bmod 13 = 4$
- T[5..15] = $t_4 = (2(4 - 10.0) + 0) \bmod 13 = 8$
- T[6..16] = $t_5 = (2(8 - 10.1) + 1) \bmod 13 = 10$
- T[7..17] = $t_6 = (2(10 - 10.0) + 1) \bmod 13 = 8$
- T[8..18] = $t_7 = (2(8 - 10.1) + 0) \bmod 13 = 9$
- T[9..19] = $t_8 = (2(9 - 10.0) + 1) \bmod 13 = 6$
- T[10..20] = $t_9 = (2(6 - 10.1) + 0) \bmod 13 = 5$
- T[11..21] = $t_{10} = (2(5 - 10.1) + 1) \bmod 13 = 4$
- T[12..22] = $t_{11} = (2(4 - 10.0) + 0) \bmod 13 = 8$
- T[13..23] = $t_{12} = (2(8 - 10.1) + 0) \bmod 13 = 9$
- T[14..24] = $t_{13} = (2(9 - 10.0) + 1) \bmod 13 = 6$

KMP

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:	X	Y	X	Y	Y	X	Y	X	Y	X	X													
Π:	0	0	1	2	0	1	2	3	4	3	1													

- i = 1, q = 0. T[1] = P[1] (Shift = 0). q = 1.
- i = 2, q = 1. T[2] = P[2] (Shift = 0). q = 2.
- i = 3, q = 2. T[3] = P[3] (Shift = 0). q = 3.
- i = 4, q = 3. T[4] ≠ P[4] (Shift = 0). q = Π[3] = 1.
  T[4] ≠ P[2] (Shift = 2). q = Π[1] = 0.
  T[4] = P[1] (Shift = 3). q = 1.
- i = 5, q = 1. T[5] = P[2] (Shift = 3). q = 2.
- i = 6, q = 2. T[6] = P[3] (Shift = 3). q = 3.
- i = 7, q = 3. T[7] = P[4] (Shift = 3). q = 4.
- i = 8, q = 4. T[8] ≠ P[5] (Shift = 3). q = Π[4] = 2.
  T[8] = P[3] (Shift = 5). q = 3.
- i = 9, q = 3. T[9] = P[4] (Shift = 5). q = 4.
- i = 10, q = 4. T[10] = P[5] (Shift = 5). q = 5.

Contd…

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:	X	Y	X	Y	Y	X	Y	X	Y	X	X													
Π:	0	0	1	2	0	1	2	3	4	3	1													

- i = 11, q = 5. T[11] = P[6] (Shift = 5). q = 6.
- i = 12, q = 6. T[12] = P[7] (Shift = 5). q = 7.
- i = 13, q = 7. T[13] = P[8] (Shift = 5). q = 8.
- i = 14, q = 8. T[14] = P[9] (Shift = 5). q = 9.
- i = 15, q = 9. T[15] = P[10] (Shift = 5). q = 10.
- i = 16, q = 10. T[16] ≠ P[11] (Shift = 5). q = Π[10] = 3.
  T[16] = P[4] (Shift = 12). q = 4.
- i = 17, q = 4. T[17] = P[5] (Shift = 12). q = 5.
- i = 18, q = 5. T[18] = P[6] (Shift = 12). q = 6.
- i = 19, q = 6. T[19] = P[7] (Shift = 12). q = 7.
- i = 20, q = 7. T[20] = P[8] (Shift = 12). q = 8.
- i = 21, q = 8. T[21] = P[9] (Shift = 12). q = 9.
- i = 22, q = 9. T[22] = P[10] (Shift = 12). q = 10.
- i = 23, q = 10.
  T[23] = P[11] (Shift = 12). q = 11.
  Pattern found at shift 12. q = Π[11] = 1.
- i = 24, q = 1.
  T[24] = P[2] (Shift = 22). q = 2.

Total shifts = 6

Total character comparisons = 28

Boyer–Moore

	2	3	4	5	6	7	8	9	10	11	12
L:	0	0	0	0	0	0	0	0	0	10	9
l:	1	1	1	1	1	1	1	1	1	1	0

	X	Y
R:	$10 \rightarrow 8 \rightarrow 6 \rightarrow 3 \rightarrow 1$	$9 \rightarrow 7 \rightarrow 5 \rightarrow 4 \rightarrow 2$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:	X	Y	X	Y	Y	X	Y	X	Y	X	X													

- Step 1: k = 11. Shift = 0.
  BC = max(1,10 – 9) = 1; GS = 11 – 10 = 1;
  k = 11 + 1 = 12. Shift = 0 + 1 = 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:		X	Y	X	Y	Y	X	Y	X	Y	X	X												

- Step 2: k = 12. Shift = 1.
  BC = max(1,11 – 9) = 2; GS = m – L(m + 1) = 11 – 9 = 2;
  k = 12 + 2 = 14. Shift = 1 + 2 = 3.

Contd…

	X	Y
R:	$10 \rightarrow 8 \rightarrow 6 \rightarrow 3 \rightarrow 1$	$9 \rightarrow 7 \rightarrow 5 \rightarrow 4 \rightarrow 2$

	2	3	4	5	6	7	8	9	10	11	12
L:	0	0	0	0	0	0	0	0	0	10	9
l:	1	1	1	1	1	1	1	1	1	1	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:				X	Y	X	Y	Y	X	Y	X	Y	X	X										

- Step 3: $k = 14$. Shift $= 3$.
  $BC = \max(1, 11 - 9) = 2$; $GS = m - L(m + 1) = 11 - 9 = 2$;
  $k = 14 + 2 = 16$. Shift $= 3 + 2 = 5$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:						X	Y	X	Y	Y	X	Y	X	Y	X	X								

- Step 4: $k = 16$. Shift $= 5$.
  $BC = \max(1, 11 - 9) = 2$; $GS = m - L(m + 1) = 11 - 9 = 2$;
  $k = 16 + 2 = 18$. Shift $= 5 + 2 = 7$.

Contd…

	2	3	4	5	6	7	8	9	10	11	12
L:	0	0	0	0	0	0	0	0	0	10	9
l:	1	1	1	1	1	1	1	1	1	1	0

	X	Y
R:	$10 \rightarrow 8 \rightarrow 6 \rightarrow 3 \rightarrow 1$	$9 \rightarrow 7 \rightarrow 5 \rightarrow 4 \rightarrow 2$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:								X	Y	X	Y	Y	X	Y	X	Y	X	X						

- Step 5: k = 18. Shift = 7.
  BC = max(1,10 – 9) = 1; GS = 11 – 10 = 1;
  k = 18 + 1 = 19. Shift = 7 + 1 = 8.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:									X	Y	X	Y	Y	X	Y	X	Y	X	X					

- Step 6: k = 19. Shift = 8.
  BC = max(1,11 – 9) = 2; GS = m – L(m + 1) = 11 – 9 = 2;
  k = 19 + 2 = 21. Shift = 8 + 2 = 10.

Contd...

	X	Y
R:	$10 \rightarrow 8 \rightarrow 6 \rightarrow 3 \rightarrow 1$	$9 \rightarrow 7 \rightarrow 5 \rightarrow 4 \rightarrow 2$

	2	3	4	5	6	7	8	9	10	11	12
L:	0	0	0	0	0	0	0	0	0	10	9
l:	1	1	1	1	1	1	1	1	1	1	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:											X	Y	X	Y	Y	X	Y	X	Y	X	X			

- Step 7: k = 21. Shift = 10.
  BC = max(1,11 – 9) = 2; GS = m – L(m + 1) = 11 – 9 = 2;
  k = 21 + 2 = 23. Shift = 10 + 2 = 12.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T:	X	Y	X	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	Y	Y	X	Y	X	Y	X	X	Y
P:													X	Y	X	Y	Y	X	Y	X	Y	X	X	

- Step 8: k = 23. Shift = 12.
  Pattern found.
  k = 21 + 11 - 1 = 31. Shift = 12 + 10 = 22.

Total shifts = 8

Total character comparisons = 20