Book

A Simplified Approach to

Data Structures

Prof.(Dr.)Vishal Goyal, Professor, Punjabi University Patiala

Dr. Lalit Goyal, Associate Professor, DAV College, Jalandhar

Mr. Pawan Kumar, Assistant Professor, DAV College, Bhatinda

Shroff Publications and Distributors

Edition 2014

<<KMP Algorithm>>

By - Anupriya Garg (140454100)

Department of Computer Science, Punjabi University Patiala

Contents for Today's Lecture

- Introduction to KMP Algorithm
- Terminologies
- Working of KMP Algorithm
- Algorithm to create a prefix array **'π'**
- Algorithm to search the pattern **'P'**
- Complexity Analysis

Introduction to KMP Algorithm

- KMP is a pattern matching algorithm which is named after its inventors **Donald Knuth , James H. Morris and Vaughan Pratt .**
- This was conceived in 1974 and was formally published in 1977.
- Knuth , Morris and Pratt developed a method which keeps the information gathered and never re-compares the symbols that have matched in earlier.
- This methodology reduces the no. of comparisons w.r.t other native approaches such as BruteForce, and achieves the running time of **O(m+n).**
- It uses the pre-processing of the patterns to analyze its structure.
- It uses the concept of *prefix* and *suffix* in the pattern.

Terminologies

- **Prefix :** A *prefix* of a string P = P1.. Pn is a string P' = P1.. Pm where **m ≤n.** A proper prefix of a string is not equal to the string itself i.e m < n. This means a prefix is the special case of a substring.
- **Cyber :** it is the prefix of string cyberspace , similarly, cybersp is prefix of string cyberspace.
- **Suffix :** A suffix of a string P = P1… Pn is a string P' = Pn-m .. .. Pn. where m <= n. A proper suffix of a string is not equal to the string itself i.e, m < n. This means, a suffix is the special case of a substring.
- **Space :** space is the suffix of string cyberspace, similarly, erspace is suffix of string cyberspace.
- **Border :** A border is suffix and prefix of the same string. That is, border of a string is a substring which is a prefix as well as suffix. For example, consider a string P = "abacab".

The proper prefixes of P are : a , ab , aba , abac , abaca
The proper suffixes of P are : b , ab, cab , acab , bacab
Here the border comes out to be : **ab** as it is suffix as well as prefix.

Working of KMP Algorithm

- KMP algorithm uses the concept of the width of the border. Eg. in previous example border width is 2.
- Here, larger the width of the border, more efficient will be the pattern matching algorithm.
- It performs the preprocessing step by analyzing the pattern, After analyzing the pattern, it creates the auxiliary army denoted by **π** which is defined as,

π [i] is the larger integer smaller than i such that P1.. P π[i] is a sufix of p1…Pi.

- Consider a pattern P = a b a b a b a b c a , the values in the array π will be ,

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
P	a	b	a	b	a	b	a	b	c	a
π	0	0	1	2	3	4	5	6	0	1

Working of KMP Algorithm (continued)

• Here, **π[10] = 1** as in the pattern with length 10(abababababca), border is of length 1(border is a)

•**π[9] = 0** as in the pattern with length 9(ababababc), border is of length 0(here , border is nil)

• **π[8] = 6** as in the pattern with length 9(ababababc), border is of length 0(here , border is nil)

•**π[7] = 5** as in the pattern with length 9(ababababc), border is of length 0(here , border is nil)

•**π[6] = 4** as in the pattern with length 9(ababababc), border is of length 0(here , border is nil)

•**π[5] = 3** as in the pattern with length 9(ababababc), border is of length 0(here , border is nil)

•**π[4] = 2** as in the pattern with length 9(ababababc), border is of length 0(here , border is nil)

•**π[3] = 1** as in the pattern with length 3(aba), border is of length 1(here , border is a)

•**π[2] = 0** as in the pattern with length 2(ab), border is of length 0(here , border is nil)

Working of KMP Algorithm (continued)

- **π[1] = 0** as in the pattern with length 1(a), border is of length 0(here, border is nil)
- Here, the array **π** is very important which is used in the shift function. The shift function f(k) gives a value by which we will shift the pattern. Whenever there is mismatch between the pattern and the text, the pattern is shifted by the amount that is calculated by the shift function. Here, the shift function is,

f (k) = k- π(k), where **k** is number of characters matched before the mismatch occur .

- Consider the text(having two white spaces denoted here by filled box) and pattern as

T : a b c ■ a b c d a b ■ a b c d a b c d a b d e

P : a b c d a b d

- The auxiliary array it is defined as: [here , **π[0] is taken as -1**]

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P	a	b	c	d	a	b	d
Π	0	0	0	0	1	2	0

Working of KMP Algorithm (continued)

- **PASS 1:**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
A	B	C		A	B	C	D	A	B		A	B	C	D	A	B	C	D	A	B	D	E
A	B	C	D	A	B	D																

The three characters of the pattern match with the text (k = 3) and mismatch occurs at position 4 of the text. The shift function :

$$f(k) = k - \boldsymbol{\pi(k) = 3 - \pi(3) = 3 - 0 = 3}$$

Therefore , we shift the pattern by three positions as shown in the second pass.

- **PASS 2:**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
A	B	C		A	B	C	D	A	B		A	B	C	D	A	B	C	D	A	B	D	E
			A	B	C	D	A	B	D													

No character of the pattern match with the text (k = 0) and mismatch occurs at position 4 of the text. The shift function:

$$f(k) = k - \boldsymbol{\pi(k) = 0 - \pi(0) = 0 - (-1) = 1}$$

Therefore , we shift the pattern by one positions as shown in the third pass.

Working of KMP Algorithm (continued)

- **PASS 3:**

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
A	B	C		A	B	C	D	A	B		A	B	C	D	A	B	C	D	A	B	D	E
				A	B	C	D	A	B	D												

Six characters of the pattern match with the text (k = 6) and mismatch occurs at position 11 of the text. The shift function :

$$f(k) = k - \boldsymbol{\pi(k) = 6 - \pi(6) = 6 - (2) = 4}$$

Therefore , we shift the pattern by four positions as shown in the fourth pass.

- **PASS 4:**

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
A	B	C		A	B	C	D	A	B		A	B	C	D	A	B	C	D	A	B	D	E
								A	B	C	D	A	B	D								

Two character of the pattern match with the text (k = 2) and mismatch occurs at position 11 of the text. The shift function:

$$f(k) = k - \boldsymbol{\pi(k) = 2 - \pi(2) = 2 - (0) = 2}$$

Therefore , we shift the pattern by two positions as shown in the fifth pass.

Working of KMP Algorithm (continued)

- **PASS 5:**

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
A	B	C		A	B	C	D	A	B		A	B	C	D	A	B	C	D	A	B	D	E
										A	B	C	D	A	B	D						

No character of the pattern match with the text (k = 0) and mismatch occurs at position 11 of the text. The shift function:

$$f(k) = k - \boldsymbol{\pi(k) = 0 - \pi(0) = 0 - (-1) = 1}$$

Therefore , we shift the pattern by one positions as shown in the sixth pass.

- **PASS 6:**

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
A	B	C		A	B	C	D	A	B		A	B	C	D	A	B	C	D	A	B	D	E
											A	B	C	D	A	B	D					

Six characters of the pattern match with the text (k = 6) and mismatch occurs at position 18 of the text. The shift function :

$$f(k) = k - \boldsymbol{\pi(k) = 6 - \pi(6) = 6 - (2) = 4}$$

Therefore , we shift the pattern by four positions as shown in the seventh pass.

Working of KMP Algorithm (continued)

- **PASS 7:**

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
A	B	C		A	B	C	D	A	B		A	B	C	D	A	B	C	D	A	B	D	E
															A	B	C	D	A	B	D	

All the seven characters of the pattern match with the text (k = 7) and no mismatch occurs. It means the pattern has been found in the text. In case we need to find the other occurrences of the same pattern , we need to shift the pattern in the similar manner using shift function.

Implementation of the KMP algorithm starts with the preprocessing of the pattern. It calculates the border and generates the array **π.**

Algorithm : Create the prefix array '**π**' for the given pattern **'P'**

```
Step 1: Set $m = \text{Length}(P)$
Step 2: Set $i = 0$ , $j = -1$
Step 3: Set $\pi[i] = j$
Step 4: Repeat While $i < m$
            Repeat While $j \geq 0$ AND $P[i+1] \neq P[j+1]$ Then
                    Set $j = \pi[j]$
            [End Loop]
            Set $i = i + 1$
            Set $j = j + 1$
                    Set $\pi[j] = j$
        [End Loop]
Step 5: Exit.
```

Algorithm : search the pattern **'P' from text 'T'**

- Given : the prefix array π which is calculated in preprocessing step.

```
Step 1: Set m = Length(P)  // m being length of the pattern
Step 2: Set n = Length(T) // n being the length of the Text
Step 3: Set i = 0 , j =  0
Step 4: Repeat While i < n
Step 5: Repeat While j ≥0 AND T[i+1] ≠ P[j+1] Then
                    Set j = π[j]
               [End Loop]
Step 6: Set  i =  i +1
Step 7: Set  j =  j +1
Step 8: If  j = m Then
               Print : "Desired Pattern Found"
               [End If]
     [End Loop]
Step 9: Print: "Desired Pattern is not found in the text"
Step 10: Exit.
```

Complexity Analysis of KMP Algorithm

• The first three steps of the KMP algorithm take constant time which does not affect the running time of the algorithm.
• The while loop in step 4 runs **n** times i.e. as long as the length of the text **T** .
• So, the running time is dominated by this loop Thus running time of the algorithm is **0(n).**
• But the complexity of this algorithm is much less than **n** as the pattern shifts more than one because of prefix function.
• The calculation of the prefix array **π** has the complexity of **0(m)** as the length of the pattern is **m**.
• So, the complexity of KMP algorithm takes **0(m+n).**