# Knuth-Morris-Pratt Algorithm

The principle behind the algorithm that is to generate the prefix table P is that:
Find the length of the longest proper prefix in the subpattern that matches a proper suffix in the same subpattern.

Let us understand through creation of a Prefix Table P using an example.

Pattern: ABAB

| Substring | Proper Prefixes | Proper Suffixes | Longest Match | Length |
|---|---|---|---|---|
| A | NULL | NULL | NULL | 0 |
| AB | A | B | NULL | 0 |
| ABA | A, AB | A, BA | A | 1 |
| ABAB | A, AB, ABA | B, AB, BAB | AB | 2 |

Prefix Table P:

| char | A | B | A | B |
|---|---|---|---|---|
| index | 0 | 1 | 2 | 3 |
| value | 0 | 0 | 1 | 2 |

Now with the help of this generated table, we can apply the algorithm. The formula to compute the shift is given by: k – P[k-1]

For example, if k is the number of matched characters then the shift is computed by:

Shift = 2 – P[2-1]
= 2 - P[1]
= 2 – 0
= 2

Let us see a more detailed example.

2. Search for the pattern ababaca in the text bacbababacaab using KMP algorithm.
Let us first create the prefix table P.

The prefix table will be populated based on the principle stated by the algorithm.

Pattern: ababaca

| Substring | Proper Prefixes | Proper Suffixes | Longest Match | Length |
|---|---|---|---|---|
| a | NULL | NULL | NULL | 0 |
| ab | a | b | NULL | 0 |
| aba | a, ab | a, ba | ab | 1 |
| abab | a, ab, aba | b, ab, bab | ab | 2 |
| ababa | a, ab, aba, abab | a, ba, aba, baba | aba | 3 |

| | | | | |
|---|---|---|---|---|
| ababac | a, ab, aba, abab, ababa | c,ac, bac, abac, babac | NULL | 0 |
| ababaca | a, ab, aba, abab, ababac | a, ca, aca, baca, abaca, babaca | a | 1 |

Prefix Table P:

| char | a | b | a | b | a | c | a |
|---|---|---|---|---|---|---|---|
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| value | 0 | 0 | 1 | 2 | 3 | 0 | 1 |

Now using the generated table we can search for the pattern

Tracing:

Iteration 01:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b | a | c | b | a | b | a | b | a | c | a | a | B |
| x | | | | | | | | | | | | |
| a | b | a | b | a | c | a | | | | | | |

Shift by 1

Iteration 02:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b | a | c | b | a | b | a | b | a | c | a | a | B |
| | √ | x | | | | | | | | | | |
| | a | b | a | b | a | c | a | | | | | |

k = 1

Shift k – P [k-1] = 1 – P[0] = 1

So, shift by 1.

Iteration 03:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b | a | c | b | a | b | a | b | a | c | a | a | B |
| | | x | | | | | | | | | | |
| | | a | b | a | b | a | c | a | | | | |

Shift by 1.

Iteration 04:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b | a | c | b | a | b | a | b | a | c | a | a | B |
| | | | x | | | | | | | | | |
| | | | a | b | a | b | a | c | a | | | |

Shift by 1.

Iteration 05:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b | a | c | b | a | b | a | b | a | c | a | a | B |
| | | | | √ | √ | √ | √ | √ | √ | √ | | |
| | | | | a | b | A | b | a | c | a | | |

Match found at index 4.

Efficiency Analysis:
For KMP, we need
O(m) to compute prefix function values
O(n) to compute pattern
And the total is O(n+m)