**NAME: ADITYA ABHIJIT PARULEKAR**

**DIV: S2-1**

**ROLL.NO: 2201072**

* **Experiment 12:**
* **AIM: To study and implement Knuth-Morris-Pratt algorithm.**
* **THEORY:**

# **The Knuth-Morris-Pratt (KMP)Algorithm**

Knuth-Morris and Pratt introduce a linear time algorithm for the string matching problem. A matching time of O (n) is achieved by avoiding comparison with an element of 'S' that have previously been involved in comparison with some element of the pattern 'p' to be matched. i.e., backtracking on the string 'S' never occurs

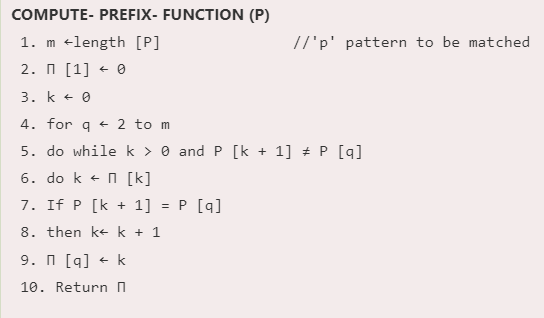
## Components of KMP Algorithm:

**1. The Prefix Function (Π):** The Prefix Function, Π for a pattern encapsulates knowledge about how the pattern matches against the shift of itself. This information can be used to avoid a useless shift of the pattern 'p.' In other words, this enables avoiding backtracking of the string 'S.'

**2. The KMP Matcher:** With string 'S,' pattern 'p' and prefix function 'Π' as inputs, find the occurrence of 'p' in 'S' and returns the number of shifts of 'p' after which occurrences are found.

## The Prefix Function (Π)

Following pseudo code compute the prefix function, Π:

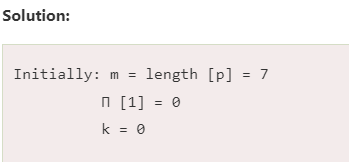


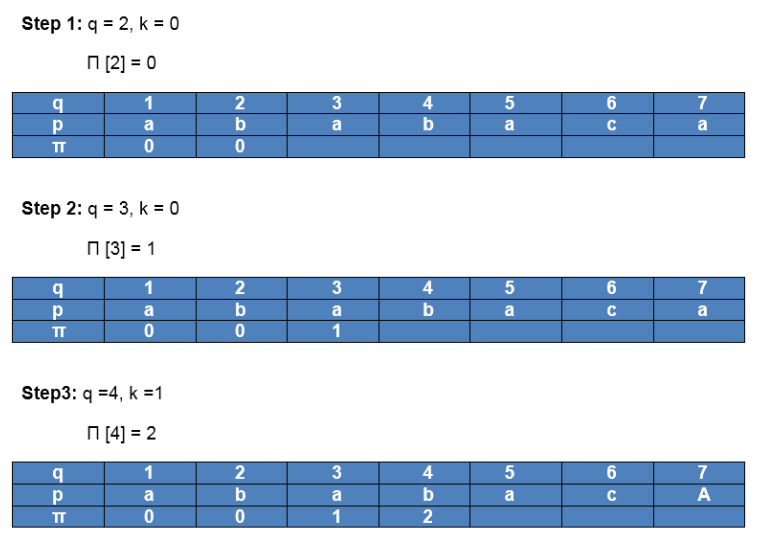
## Running Time Analysis:

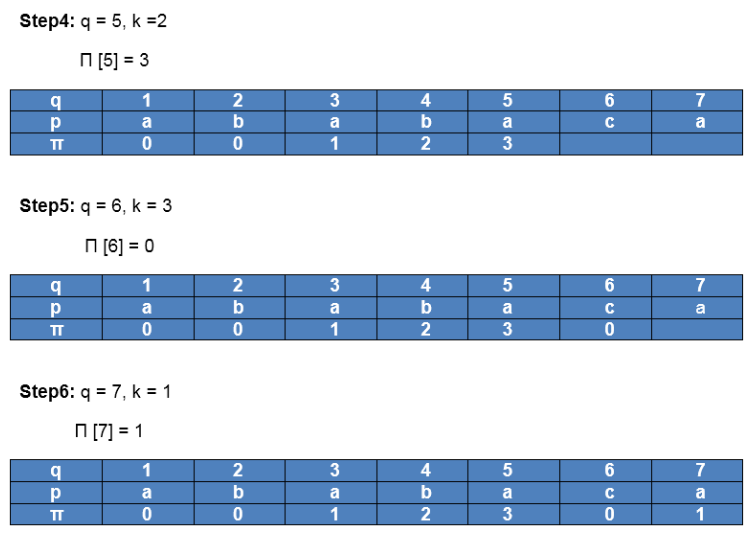
In the above pseudo code for calculating the prefix function, the for loop from step 4 to step 10 runs 'm' times. Step1 to Step3 take constant time. Hence the running time of computing prefix function is O (m).

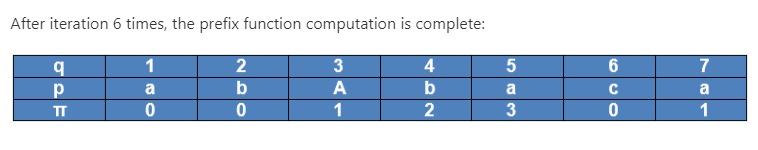
**Example:** Compute Π for the pattern 'p' below:

Knuth-Morris-Pratt Algorithm



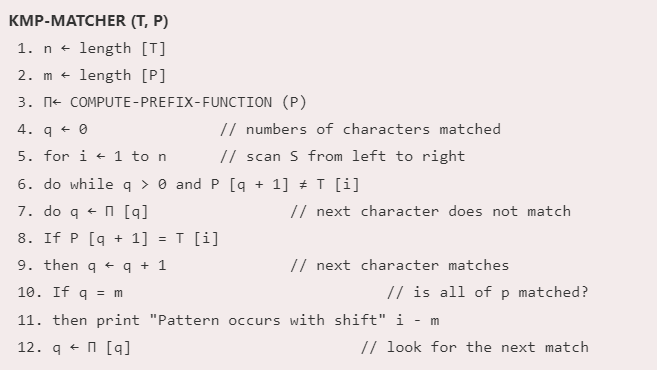






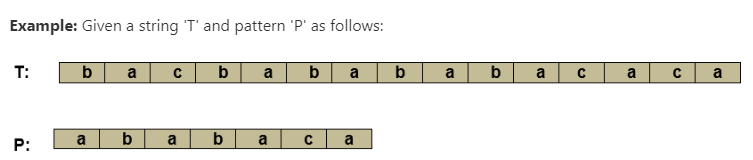
## The KMP Matcher:

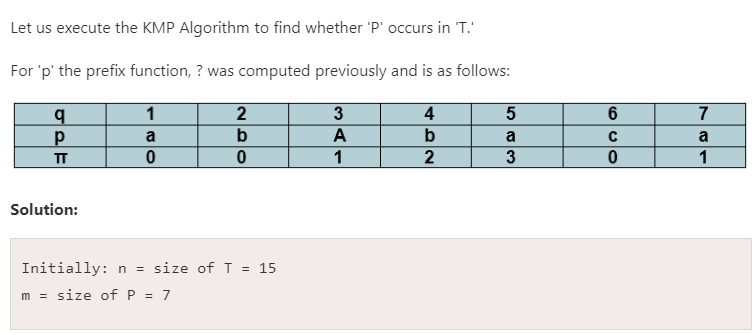
The KMP Matcher with the pattern 'p,' the string 'S' and prefix function 'Π' as input, finds a match of p in S. Following pseudo code compute the matching component of KMP algorithm:

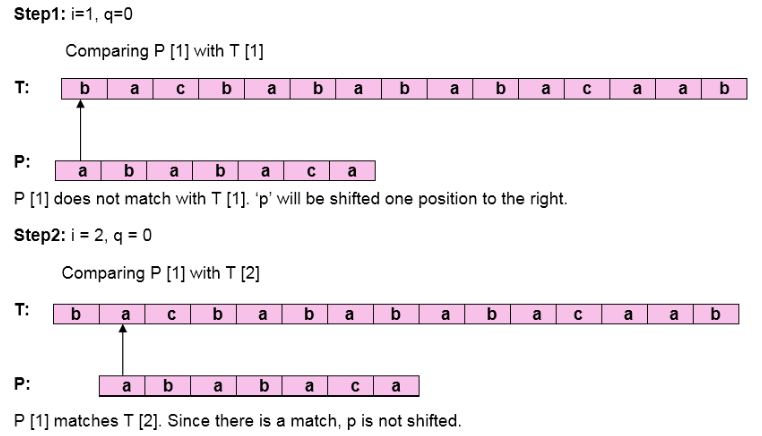


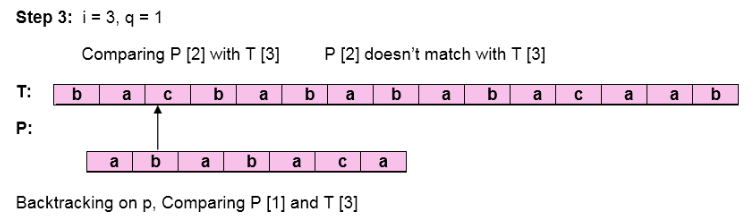
## Running Time Analysis:

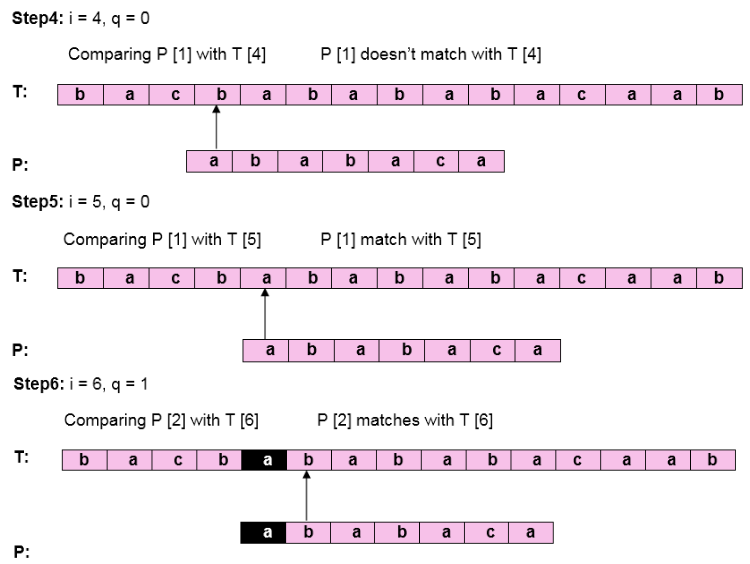
The for loop beginning in step 5 runs 'n' times, i.e., as long as the length of the string 'S.' Since step 1 to step 4 take constant times, the running time is dominated by this for the loop. Thus running time of the matching function is O (n).

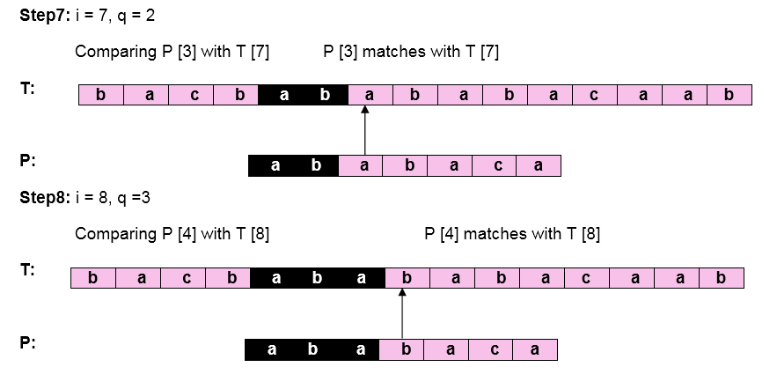


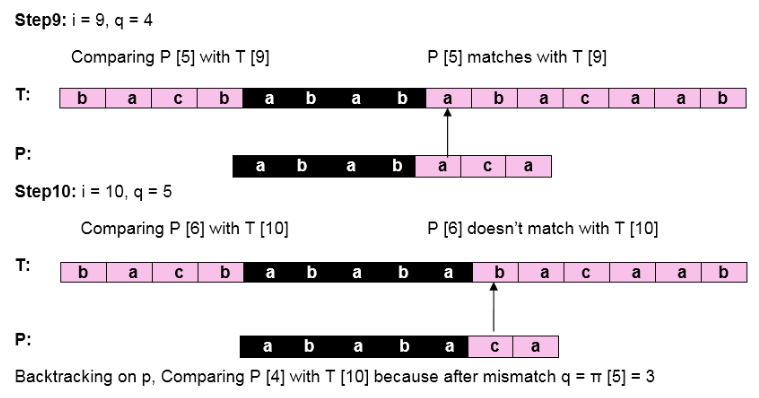


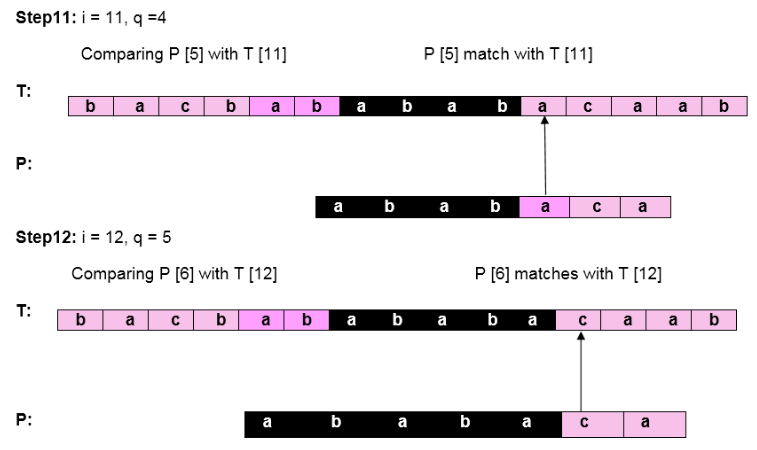


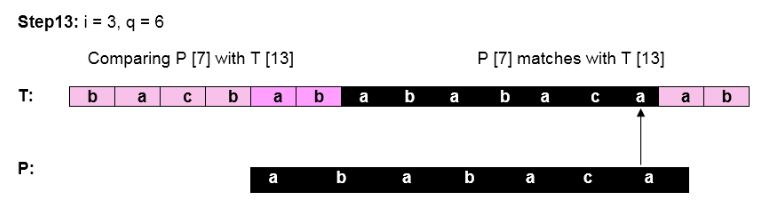












Pattern 'P' has been found to complexity occur in a string 'T.' The total number of shifts that took place for the match to be found is i-m = 13 - 7 = 6 shifts.

* **C PROGRAM:**

#include <stdio.h>

#include <string.h>

void computeLPSArray(char\* pat, int M, int\* lps);

void KMPSearch(char\* pat, char\* txt)

{

int M = strlen(pat);

int N = strlen(txt);

int lps[M];

computeLPSArray(pat, M, lps);

int i = 0;

int j = 0;

while ((N - i) >= (M - j)) {

if (pat[j] == txt[i]) {

j++;

i++;

}

if (j == M) {

printf("Found pattern at index %d\n", i - j);

j = lps[j - 1];

}

else if (i < N && pat[j] != txt[i]) {

if (j != 0)

j = lps[j - 1];

else

i = i + 1;

}

}

}

void computeLPSArray(char\* pat, int M, int\* lps)

{

int len = 0;

lps[0] = 0;

int i = 1;

while (i < M) {

if (pat[i] == pat[len]) {

len++;

lps[i] = len;

i++;

}

else {

if (len != 0) {

len = lps[len - 1];

}

else {

lps[i] = 0;

i++;

}

}

}

}

int main()

{

char txt[100], pat[100];

printf("ADITYA PARULEKAR-S21-2201072\n");

printf("------------ KMP ALGORITHM ---------------\n\n");

printf("Enter the text: ");

scanf("%s", txt);

printf("Enter the pattern: ");

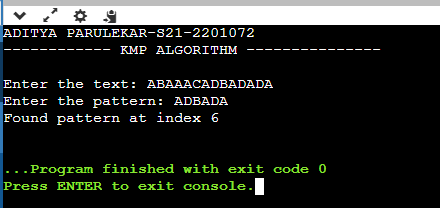
scanf("%s", pat);

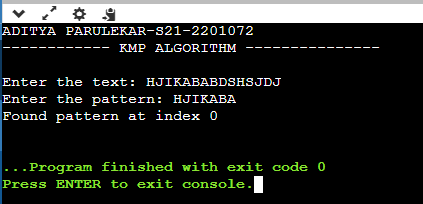
KMPSearch(pat, txt);

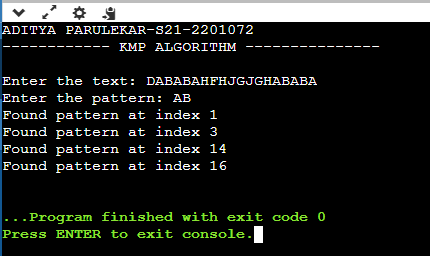
return 0;

}

* **OUTPUT:**







* **CONCLUSION: Hence, we have successfully implemented Knuth-Morris-Pratt algorithm; LO 1, LO 2.**