String algorithm

*→ **String processing problem**

Input: Two strings T and P
Problem: find if P is a substring of T

Example -:①

Input T = gtgatcagatcact P=tcg
output : yes shift = 4, 9

Example -:②

Input T= 18934267089 3, P=1673
Output: No

Solution: (Naïve Algorithm) -:

```
Naïve algorithm (T, P)
{ suppose n = length(T), m = length(P);
for shift s=0 to n-m do
    if ( P[1..m] == T[s+1.. s+m] ) then
        print shift s;
end algo.
```

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

Knuth - Morris - Pratt algorithm :-

KMP flow chart construction :-

Input: P, a string of characters; m, the length of P.

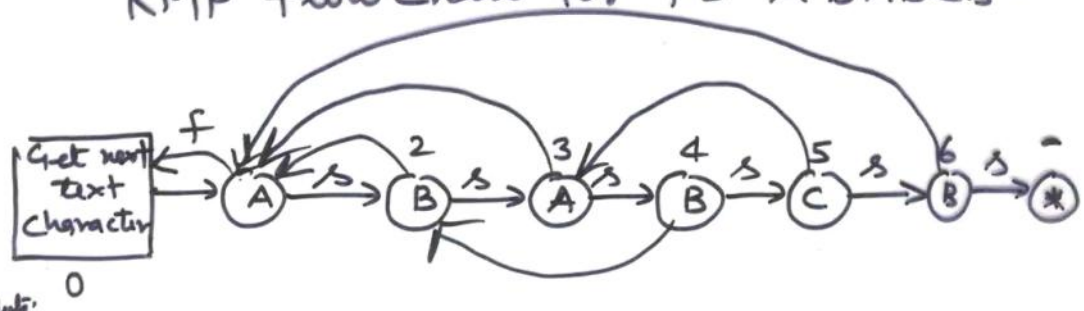
Output: fail, the array of failure links, defined for index 1, ..., m. The array is passed in and the algorithm fills it

```
Void KmpSetup(char[] P, int m, int [] fail)
    int k, s;
{   fail[1] = 0;
    for (k=2; k≤m; k++)
      s = fail[k-1];
     while (s ≥ 1)
       if (Ps == Pk-1)
            break;
       s = fail[s]
    fail[k] = s+1
```

Complexity $O(m^2)$

Example:

KMP flowchart for P = ' A B A B C B '



Note:

* fail [1] = 0

* fail [2]

$s = fail[1] = 0$

$\Rightarrow fail[2] = 1$

* fail [3]

$s = fail[2] = 1$

$\because p_2 \neq p_1$

$s_1 = fail[1]$

$= 0$

$\Rightarrow fail[3] = 1$

* fail [4]

$s = fail[3] = 1$

$\because p_3 = p_1$

$\Rightarrow fail[4] = 2$

* fail [5]

$s = fail[4] = 2$

$p_4 = p_2$

$\Rightarrow fail[5] = 3$

fail [6]

$s = fail[5] = 3$

$p_5 \neq p_3$

$\Rightarrow s_1 = fail[3]$

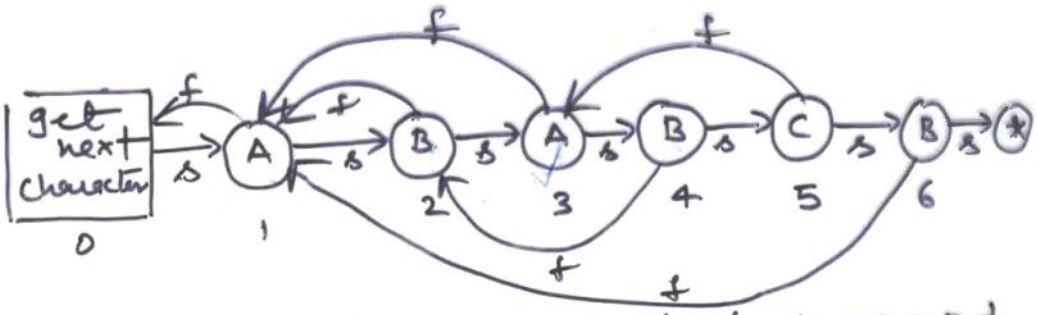
$= 1$

$p_5 \neq p_1$

$\Rightarrow s_2 = fail[1]$

$= 0$

$\Rightarrow fail[6] = 1$



KMP- flow chart for p = 'A B A B C B'

Action of the KMP flow chart for the pattern A B A B C B on the text last line

A C A B A A B A B A
1 2 3 4 5 6 7 8 9 10

{ 4 ✓ 11 none failure

KMP Cell No	Text being processed: index	Text being processed: character	success or failure
1	1	A	s
2	2	C	f
1	2	C	f
1	3	A	s
2	4	B	s
3	5	A	s
4	6	A	f
2	6	A	f
1	6	A	s
2	7	B	s
3	8	A	s
4	9	B	s
5	10	A	s
3	10	A	s

Algorithm KMP scan

Input: P and T
- P → pattern
- T → text string

m = length of P,
fail: array of failure links

Output: The return value is the index in T where a copy of P begins or -1

```
int kmpScan(char[] P, char[] T, int m, int[] fail)
  {  int match;
     int j, k;
  // j indexes text and k indexes pattern & fail array
  match = -1 ; j = 1 ; k = 1
while (endText(T, j) == false)
   if  k > m
       match = j - m  // match found
       break;
    if (k == 0)
        j++;
        k = 1  // start pattern over
    else if (tj == pk)
        j ++;
        k ++;
     else
        k = fail[k];
      // continue loop
return match
```

$\theta(m+n)$

Exercise :- Draw KMP flow chart for the pattern

A B A B A B C B