Q String
WAP to check whether one string is a rotation of another

Date. ________
Page No. ________

str1 — mypencil   S2 — pencilmy

m y p c ....   p e n c i l m y

so. 2 loop. outer loop => for str1

if for a char of str1 → match in s2 then next char

0 1 2   7 8 9 10
my pencil   pencilmy
j j j j   j j j   x x   x x x x
i i

$TC \Rightarrow O(n^2)$

2nd Method

Make str1 as (mypencilmy penil)

Now find S2 in str1 if found then S2 is a rotation of mypencil else NOT

For string Matching = use KMP Algo
$\Downarrow$
$TC = O(N)$

Problem with the above method or any normal string Matching

s1 =	a	a	a	a	a	a	a	b
	0	1	2	3	4	5	6	7

s2 =	a	a	~~a~~ b	b
	0	1	2	3

first 3 letter matches then suddley b does not match Hence we had to start from 1 index of S1 so all the progress is wasted in this method.

KMP Pattern: $\underset{0}{a}\,\underset{1}{b}\,\underset{2}{c}\,\underset{3}{d}\,\underset{4}{a}\,\underset{5}{b}\,\underset{6}{c}$

prefix: a, ab, abc, abcd

suffix: c, bc, abc, dabc.

these is a prefix which is also in suffix

String: a d s g w a d s x d s g w a d s g z

Pattern: d s g w a d s g z

0	0	0	0	0	1	2	3	0

We caon't go Back in string if these is a mismatch. we will just go the place in Patter where suffix can be ~~[illegible]~~ found

String	Pattern
a	d ✗
d	d ✓
s	s ✓
g	g
d	d ✓
s	s ✓
x	g ✗ → so we won't go back in string we go back to prefix which is equal to the longest suffix

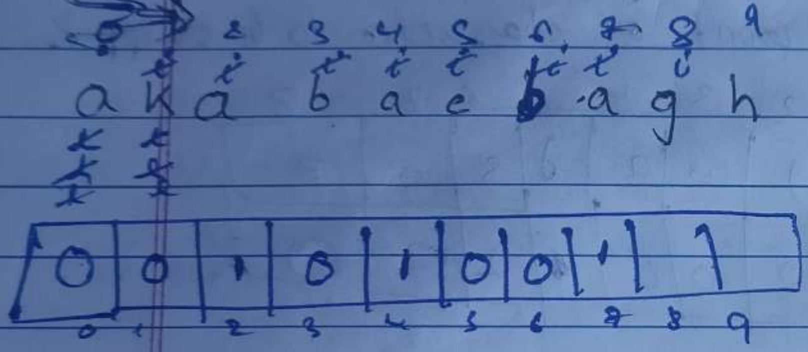
~~As Here we go to last ds~~

string = a dsgwadsxdsgwadsgz
Pattern = dsgwadsgz → not match here go to

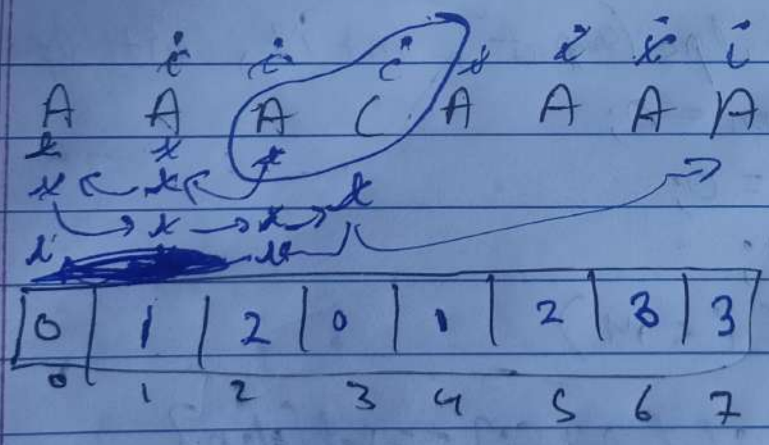
So ~~dello~~ we will know where to go: with help of lps[]
→ How to known how many character to be skipped.
To know this, we ~~pre~~ process pattern and ~~pa~~ prepare an Integer array lps[] that tells us the count of character to be skipped.

	0	1	2	3	4	5	6	7	8
pattern	d	s	g	w	a	d	s	g	z
lps	0	0	0	0	0	1	2	3	0

```
                  size of Pat
Compute lps (String 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 // when pat[i] != Pat[len]
if(len!=0)
    len = lps[len-1];
else // when len=0
{ lps[i] = 0;
    i++;
}
}
```



0	0	1	0	1	0	0	1	1	
0	1	2	3	4	5	6	7	8	9



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

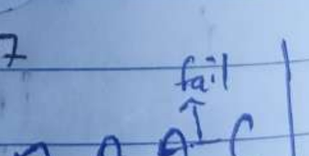
$A \neq C$

(len != 0)

len = lps[len-1]

len = lps[1]

$\boxed{\text{len = 1}}$



when A and C [illegible]

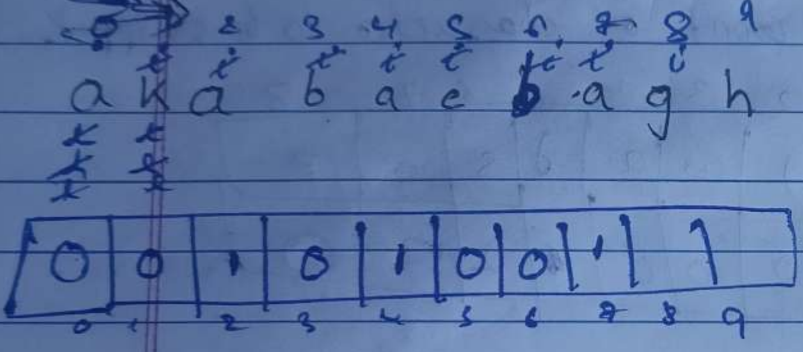
AA Pre AC Suff

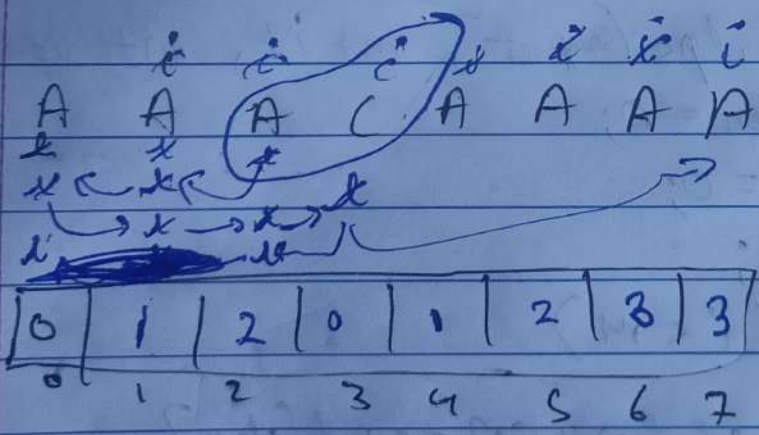
so we need to check A-C are eq or NOT

Date. ______

Page No. ______

```
{
if(len!=0)
    len = lps[len-1];
else // when len=0
    {lps[i] = 0;
     i++;
    }
}
```





$A \neq C$

(len != 0)

len = lps[len-1]

len = lps[1]

$\boxed{len = 1}$

fail

A A A C

when

A and C

AA AC

Pre Suff

so we need to check A C are eq or NOT

Date.
Page No.

```
when l at C  ant i = 7 (A)
          (C != A)
          (len != 0)
          len = lps[len-1]
          len = 2
```

$\underline{A\ A\ A}\ C\quad A\ \underbrace{A\ A\ A}$

```
KMP(string pat, string txt)
{     int n = pat.size(), m = txt.size();
    int lps[m];

    Compute lps(______);
                 ↗txt   ↗ Pat
    int i=0, J=0;
    while ((N-i) >= (M-J))
    {
      if(pat[J] = = txt[i])
       {     J++;
             i++;
       }
       if(J = = M)
       { found pattern at << (i-J);
         J = lps[J-i];  → To found 2nd or 3rd — pattern
                           in the txt
       }
      else if (i<N && pat[J] != txt[i])
       {    if(J != 0)
              J = lps[J-1];
           else
```

```
            i=i+1;
        }
    }
}
```

txt = a d s g w a d s x d s g w a d s g z
(0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18)

Pat = d s g w a d s g z
(0 1 2 3 4 5 6 7 8)

[0|0|0|0|0|1|2|3|0]
0 1 2 3 4 5 6 7 8

i — J

a — d ✗

J=0 (i++)

d — d ✓

s — s ✓

⋮ ⋮

w w ✓

a a ✓

d d ✓

s s ✓

x g ✗

J!=0 J = lps[J−1] = 2

longest ~~Postfix~~ sufix that is also a ~~suffix~~ prefix is length 2 (ds ---- ds g z)

mismatch

Sowe ~~start as~~ search from index 2

i J(2)

x g mismatch

J!=0 J = lps[J-1] = 0
lps [1]

i J(0)

x d mismatch

But J=0 Hence i++

d - d match ✓

s - s

g - g

: :

g - g

z - z ✓

if (J == $\overset{(9)}{M}$)

yes

{ first pattern found at i-J index
17 - 8
$\boxed{9}$

$\boxed{J = \cdot lps[J-1]}$
↳ to found 2nd pattern in txt

}