```
        0 1 2 3 4 5 6 7 8
S:      a a b a c a a b a
LPS[]   0 1 0 1 0 1 2 3 4
```

P: aaba ⇒ N

T: a a b a c a a b a ⇒ M

P $ T ⇒ aaba $ a a b a c a a b a

LSP[] ⇒ 0 1 0 1 0 1 2 3 4 0 1 2 3 4

$$TC: O(N+M)^3$$

⇒ KMP (Knuth Morris Prat)

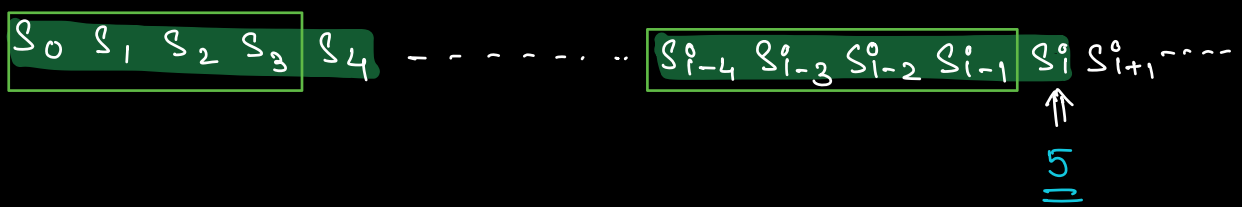
Quiz

```
        0 1 2 3 4 5 6
S:      a b a y a b a
LPS[]:  0 0 1 0 1 2 3
```

Quiz

```
        0 1 2 3 4 5 6 7
S:      c a c y c a c a
LPS[]:  0 0 1 0 1 2 3 2
```

LPS[i] : Length of longest prefix that is also a suffix in the string from [0, i].



$LPS[i] = 5$

$$S_0 S_1 S_2 S_3 S_4 = S_{i-4} S_{i-3} S_{i-2} S_{i-1} S_i$$

$(LPS[i-1])_{min} = 4$

$S_0 S_1 S_2 S_3 = S_{i-4} S_{i-3} S_{i-2} S_{i-1}$

$$LPS[i-1] >= 4$$

$LPS[i] = x$

$LPS[i-1] >= x-1$

$LPS[i-1] >= LPS[i] - 1$

$$LPS[i] <= LPS[i-1] + 1$$

⇒ Note: The LPS value can atmax increase by (1).

Quiz

LPS[i] = 5

LPS[i-1] >= 5 - 1

LPS[i-1] >= 4

↳ #3

Ex

i

	0	1	2	3	4	5	6	?
S:	a	b	a	y	a	b	a	
LPS[]:	0	0	1	0	1	2	3 (x)	4

LPS[i] = ?

LPS[i-1] = x = 3

LPS[i] <= x+1

LPS[i] <= 4

if LPS[i] = 4

S[i] = S[x]

Ex

0	1	2	3	4	5	6	7	8	9
b	c	a	d	c	b	c	a	d	?
0	0	0	0	0	1	2	3	4	

n

LPS[i] <= LPS[i-1] + 1

n

LPS[i] <= 5

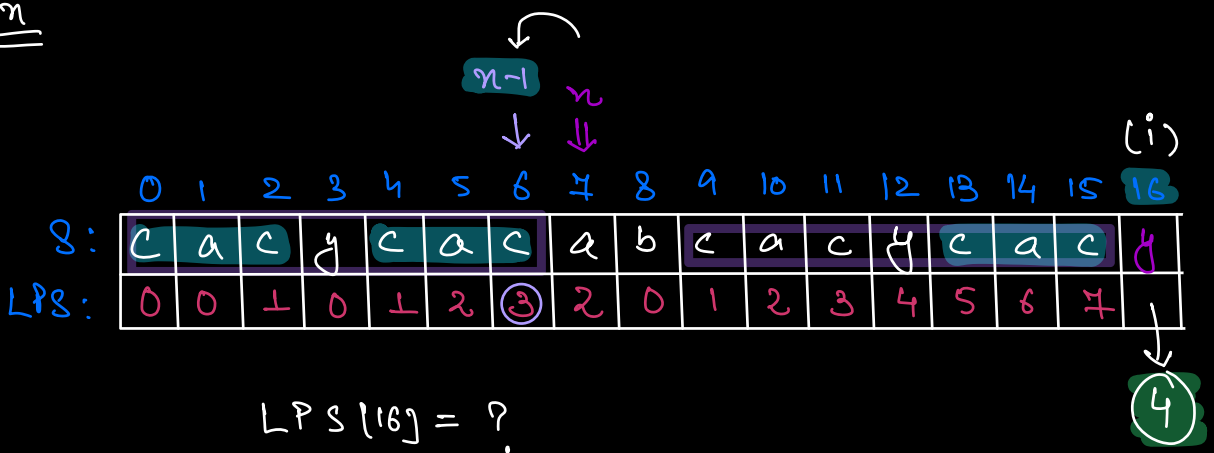
if LPS[i] = 5

⇒ S[i] should be equal to S[4].

S[i] = c

```
n = LPS[i-1]
if (s[i] == s[n]) {
        LPS[i] = n+1;
}
```

Eg



LPS[16] = ?

LPS[15] = 4 = n

LPS[16] <= 8.

if (S[16] == S[7]) ×

$S_0 S_1 S_2 S_3 S_4 S_5 S_6 = S_9 S_{10} S_{11} S_{12} S_{13} S_{14} S_{15}$

$S_0 S_1 S_2 = S_{13} S_{14} S_{15} \Rightarrow S[16] == S[3] \Rightarrow$ ✓

i = 16,
LPS[i-1] = 7 = n

n	if (S[i] == S[n])	n_new
7	S[16] == S[7] y == a ×	n = LPS[n-1] = LPS[6] = 3
3	S[16] == S[3] y == y ✓ ⇒ LPS[i] = n+1 = 4	

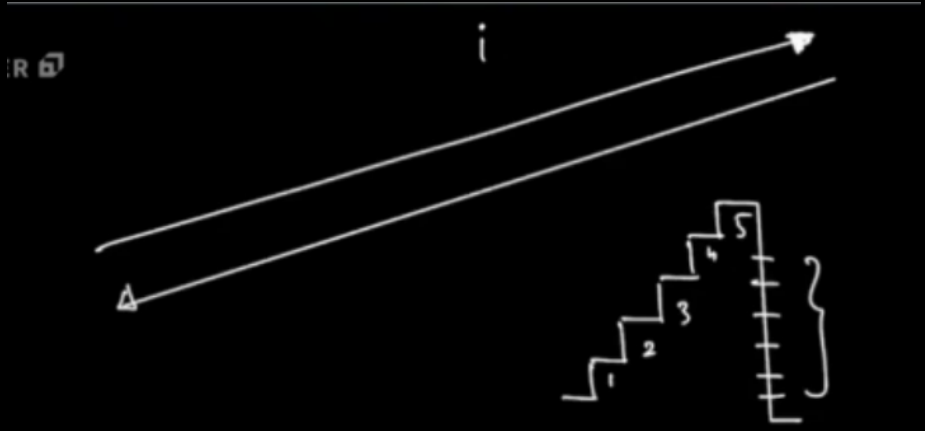
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	a	b	c	a	b	d	a	b	c	a	b	e	a	b	c	a	b	d	a	b	c	a	b	c
	0	0	0	1	2	0	1	2	3	4	5	0	1	2	3	4	5	6	7	8	9	10	11	3

(n) (i)

i = 23, LPS[i] = ?

n = LPS[i-1] = 11

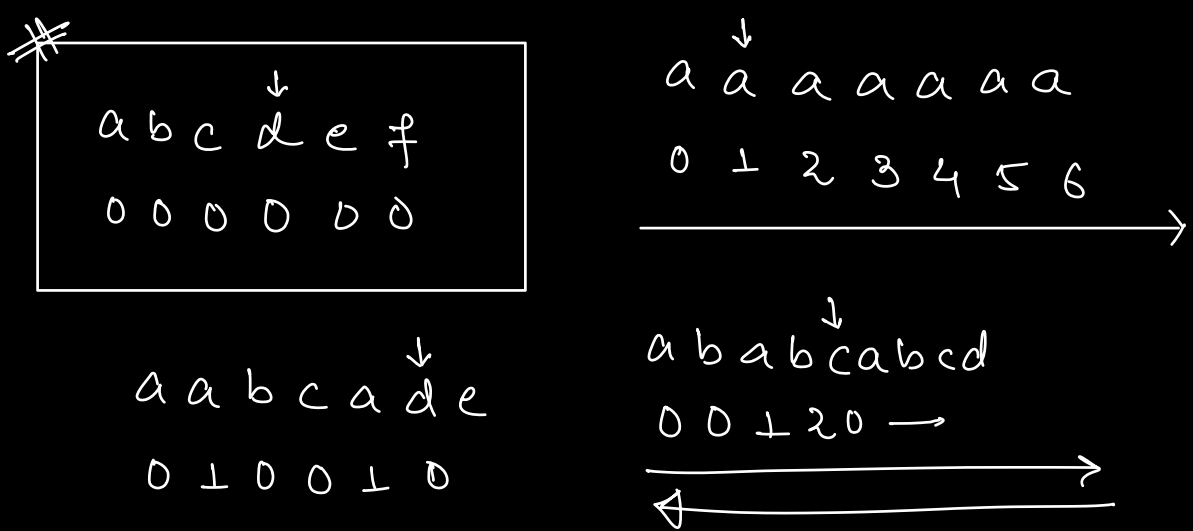
n	if (S[i] == S[n])	n_new
11	S[23] == S[11] c == e ? x	n = LPS[n-1] = 5
5	S[23] == S[5] c == d x	n = LPS[n-1] = LPS[4] = 2
2	S[23] == S[2] c == c ✓ LPS[i] = n+1 = 3	



every index will be touched at max 1 by increasing 1 by one and also by decreasing

```
int LPS[N];
LPS[0] = 0;
for(i = 1; i < N; i++) {
        x = LPS[i-1];
        while ( s[i] != s[x] ) {
                if(x == 0) { x = -1, break; }
                x = LPS[x-1]
        }
        LPS[i] = x + 1;
}
```

TC : O(N)



Pattern Matching

P ⇒ N

T ⇒ M >> N.

S : P$T ⇒ N+M

↳ find LPS[] for (S).

⇒ Count occurrences of N in LPS[].

TC ⇒ O(N+M)

SC ⇒ O(N+M)

Q: Given a String, Count the no of different rotations (<=N) that gives us the original String back.

S : abcd

bcda

cdab

dabc

abcd ⇒ (1)

S: abab
baba
abab
baba
abab

⇒ (2)

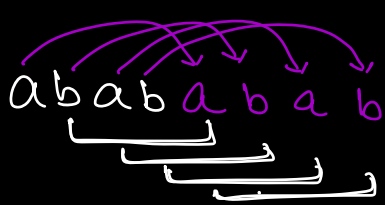
S: aaa
aaa
aaa
aaa

⇒ (3)

S: abacd
bacda
acdab
cdaba
dabac
abacd

⇒ (1)

⇒



⇒ Count the no. of occurrences of abab in this string

P = abab
T: abababab → S+S.

T : abababab.

P : abab

P$T ⇒ abab$ababab.

0 0 1 2 0 1 2 3 4 3 4 3 4

Count − 1

P/S: aaa

T : aaaaaaa

⇒ P$T ⇒ aaa$aaaaaaa

1st 2nd 3rd

0 1 2 0 1 2 3 3 3 3

⇒ 4 − 1 = 3

—— * ——