Today's Content: ⟶ {Pen/Paper}

- → Prefix & Suffix Strings
- → LPS of a given String
- → LPS[] of a given String

Problems based on LPS[]

- → Pattern matching by LPS
- → Cyclic rotations
- → Min characters to be added at start to make entire string palindrome
- → Period of a String → {Assignment}

Given a String S of N size :

Prefix Strings : Substrings starting at index 0

Suffix Strings : Substrings ending at index N-1

Eg: S = a b a b

0	1	2	3
a	b	a	b

Prefix String	Suffix Strings
a	b
a b	a b
a b a	b a b
a b a b	a b a b

→

LPS of a String : length of longest prefix which is also suffix string

Note: except full string

Eg1: S = a b c a b (0 1 2 3 4)

Prefix Strings	Suffix Strings	
a	b	
a b	a b	: len = 2
a b c	c a b	
a b c a	b c a b	
a b c a b	a b c a b	

S = a

Prefix: a — Suffix: a *

ans = 0

S = a a a a a (0 1 2 3 4)

Prefix	Suffix	
a	a	
a a	a a	
a a a	a a a	
a a a a	a a a a	: ans = 4
a a a a a	a a a a a	

TC to calculate LPS? : $O(N^2)$

0 1 2 3 — N-2 N-1

S_6 = S_0 S_1 S_2 S_3 S_4 S_5

Prefix	Suffix	iterations
S_0	S_5	1
$S_0 S_1$	$S_4 S_5$	2
$S_0 S_1 S_2$	$S_3 S_4 S_5$	3
$S_0 S_1 S_2 S_3$	$S_2 S_3 S_4 S_5$	4
$S_0 S_1 S_2 S_3 S_4$	$S_1 S_2 S_3 S_4 S_5$	5

Total = $\frac{(5)(6)}{2}$

// generalize: S_N

Pref	Suff	itr
S_0	S_{N-1}	: 1
S[0 1]	S[N-2 N-1]	: 2
S[0 2]	S[N-3 N-1]	: 3
S[0 3]	S[N-4 N-1]	: 4
:		
S[0 N-2]	S[1 N-1]	: N-1

Total iterations : $O(N^2)$

Given a string S of length N, Return the LPS array.

LPS[i] = LPS value of Substring [0 i]

TC : O(N) * {N2} → O(N3)

TC : O(N) {✓}

Ex:

	0	1	2	3	4	5	6
S =	a	a	b	a	a	b	a
LPS[7] =	0	1	0	1	2	3	4

LPS[0] = LPS value of S[0 0] : "a"

Prefix	Suffix
a	a

} ans = 0

LPS[1] = LPS value of S[0 1] : "aa"

Prefix	Suffix
a	a
aa	aa

} ans = 1

LPS[2] = LPS value of S[0 2] : "aab"

Prefix	Suffix
a	b
aa	ab
aab	aab

} = 0

LPS[3] = LPS value of S[0 3] : "aaba"

Prefix	Suffix
a	a
aa	ba
aab	aba
aaba	aaba

LPS[4] = LPS value of S[0 4] : "aabaa"

Prefix	Suffix
a	a
aa	aa
aab	baa
aaba	abaa
aabaa	aabaa

} ans = 0

LPS[5] =
LPS value of S[0 5] : aabaab

Prefix	Suffix
a	b
aa	ab
aab	aab
aaba	baab
aabaa	abaab
aabaab	aabaab

LPS[6] =
LPS value of S[0 6] : aabaaba

Prefix	Suffix
a	a
aa	ba
aab	aba
aaba	aaba
aabaa	baaba
aabaab	abaaba
aabaaba	aabaaba

Quiz: S = a a b a c a a b a

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

lps[4] = LPS value of s[0 4] = "aabac"

Pref	Suf
a	c
a a	a c
a a b	b a c
a a b a	a b a c
a a b a c	a a b a c

LPS[5] = LPS value of s[0 4] = "aabaca"

Pref	Suf
a	a
a a	c a
a a b	a c a
a a b a	b a c a
a a b a c	a b a c a
a a b a c a	a a b a c a

Q1) Search for a given Pattern P in Text T

lowercase

Eg1: T_N : a a b a c d (0 1 2 3 4 5)

m is length of Pattern

P_m : a b a c

P + T : a b a c a a b a c d (0 1 2 3 4 5 6 7 8 9)

LPS[10] : 0 0 1 0 1 1 2 3 4 0

{0 3} = {5 8}

? : [P] → in Text

↳ found ar Patten : { lps[i] == m } { Pattern present in Text }

Q2) : #Count no: of given Pattern P in Text T

Eg1: T : d a b c c a b c f → ans = 2

P : a b c

Eg2: T : a a a a (0 1 2 3)

P : a a

→ How many times P occurs in T

P + T : a a a a a a

LPS[6] : 0 1 2 3 4 5

≥ 2

separator

add a delimiter to separate text & pattern

Separator: $, @, . -

pick a separator, which is not present in both of them

$T_{N=4}$: 0 1 2 3
a a a a

$P_{M=2}$: a a

P $ T : a a $ a a a a

LPS[7] : 0 1 0 1 2 2 2

c = 0 → +1 +1 +1 → 3

Idea: S = P $ T → Size ⇒ (N + M + 1)

#optimize

1) Construct lps[] → TC: $(N+M+1)^3$ → $O(N+M+1)$?

{ Wednesday Session }

c = 0

2) Iterate in lps[]

```
if (lps[i] == M) {
    c = c + 1
}
```

3) return c

10:55 pm ?

Q2) Given a binary string S → {0, 1}

Find no. of (start-end) cyclic rotations which same a given String

At most only String len Rotations can be done?

Ex: 1 0 1 0

r_1 : 0 1 0 1

r_2 : 1 0 1 0 : +1

r_3 : 0 1 0 1

r_4 : 1 0 1 0 : +1

Ex: a b c d (original string)

r_1 : b c d a → org

r_2 : c d a b → org

r_3 : d a b c → org

r_4 : a b c d → org

// Idea: S = a b c d

S_1 = a b c d a b c d

Every string rotation of S is present as substring in S_1

$S_1 = S + S$

Ex: N

S = 1 0 1 0

0 1 2 3 4 5 7 6

2N ← S_1 = S S = 1 0 1 0 1 0 1 0

In S_1 we simply need len of S

T = S_1 → [2N]

P = S → [N]

P $ T → len → 3N+1

LPS[] → O(3N) → O(N)

If we want to search for pattern in Text P

38: Given a string N. Calculate minno: of characters we need to add at start to make entire string palindrome

Eg1: d c a b a c d

Eg2: d c a a b b a a c d

Eg3: f c d a b c b a d e f

Eg4: h g a a a e a a a g h

Idea: N - length of longest Palindrome starting at 0^{th} index

N - {length of longest Prefix Palindrome}

Eg1: a b a c d d c a b a

// Eg: S → $S_0 S_1 S_2 S_3 S_4$ longest palindrom at 0^{th}

Eg2: $S_0 S_1 S_2 S_3 S_4 S_4 S_3 S_2 S_1 S_0$

$S_0 S_1 S_2 S_3$ = $S_3 S_2 S_1 S_0$ → { $S_0 S_1 S_2 S_3$ is palindrom }

↳ should have been longest palindrom * {Contradiction}

// $S = s_0\ s_1\ s_2\ s_3\ s_4\ s_5\ s_6\ s_7\ s_8\ s_9$

$s_0\ s_1\ s_2\ s_3\ s_4\ s_5\ s_6\ s_7\ s_8\ s_9\ \underset{9}{s}\ \underset{8}{s}\ \underset{7}{s}\ \underset{6}{s}\ \underset{5}{s}\ \underset{4}{s}\ \underset{3}{s}\ \underset{2}{s}\ \underset{1}{s}\ \underset{0}{s}$ → lps = 5

// S = a a a a (0 1 2 3)

	0	1	2	3	4	5	6	7
$S + rev(S)$ =	a	a	a	a	a	a	a	a
lps[] =	0	1	2	3	4	5	6	7

Both String & rev → Edge case

// S = a a a a

	0	1	2	3	4	5	6	7	8
$S + \$ + rev(S)$ =	a	a	a	a	$	a	a	a	a
lps[] =	0	1	2	3	0	1	2	3	4

ans = N − { len of longest palindrome starting at 0^{th} index = 4

= 4 − 4 = 0

Idea: // given S

$S_1 = S + "\$" + rev(S)$

lps[2N+1] : Construct lps[] for String S_1 → TC: $O(2N) \to O(N)$

len of longest palindrome substring starting at 0^{th}

ans = N − lps[2N]

Pseudocode // for a given S_N, we need to calculate lps[N]

```
lps[0] = 0;

for(i=1; i<N; i++) {

    // Say we need to get lps[i]

    x = lps[i-1]

    while( str[i] != str[x] ) {

        if(x == 0) { x = -1; break }

        x = lps[x-1]

    }

    lps[i] = x+1

}
```

T.C → O(N)

//

Ex: a b c a a c b d d b c a a c b a