Today's Content:

→ Calculating LPS[]

→ Optimization LPS[]

→ { }

Eg1:

	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

TC → Lps[] : (N^3) ⟶ (N) ?

// Calculating LPS[] → {25%}

Obs1 : Given S of length N & assume lps[i] = 5 (string [0 i])

S_N : $S_0\ S_1\ S_2\ S_3\ S_4 \ldots\ S_{i-5}\ S_{i-4}\ S_{i-3}\ S_{i-2}\ S_{i-1}\ S_i \ldots S_{N-1}$

lps[] : - - - - - . . . - - - - - 5

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

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

lps[i-1] >= 4 }

$S_0 S_1 S_2 S_3 S_4$ = $S_{i-5}\ S_{i-4}\ S_{i-3}\ S_{i-2}\ S_{i-1}$

$S_0 S_1 S_2 S_3 S_4 S_5$ = $S_{i-6}\ S_{i-5}\ S_{i-4}\ S_{i-3}\ S_{i-2}\ S_{i-1}$

Assuming:

lps[i] = x

lps[i-1] >= x-1

lps[i-1] >= lps[i] - 1

lps[i-1] + 1 >= lps[i]

lps[i] x = lps[i-1] + 1

At max

lps[i] = lps[i-1] + 1

// our lps[] value can at max increase by +1 ? →

Step:2 → 50%

ch – unknown

ch == S[3]

Eg1:

	0	1	2	3	4	5	6	7
S =	a	b	a	y	a	b	a	ch
lps[] =	0	0	1	0	1	2	3	

i

i = 4 ?

Eg2:

	0	1	2	3	4	5	6	7	8	9
S =	b	c	a	d	c	b	c	a	d	ch
lps[] =	0	0	0	0	0	1	2	3	4	?

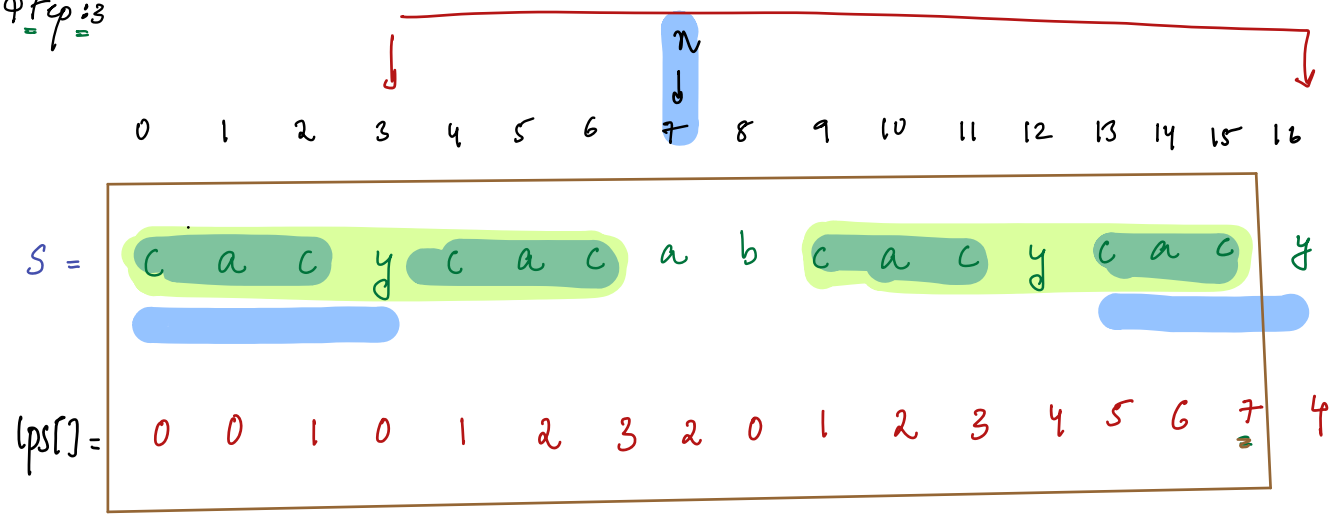
ch == S[4]

// generalize: // Assume lps[0, i-1] are calculated?

Cal lps[i] = ?

```
x = lps[i-1]
if ( s[x] == s[i] ){
    lps[i] = x+1
}
```

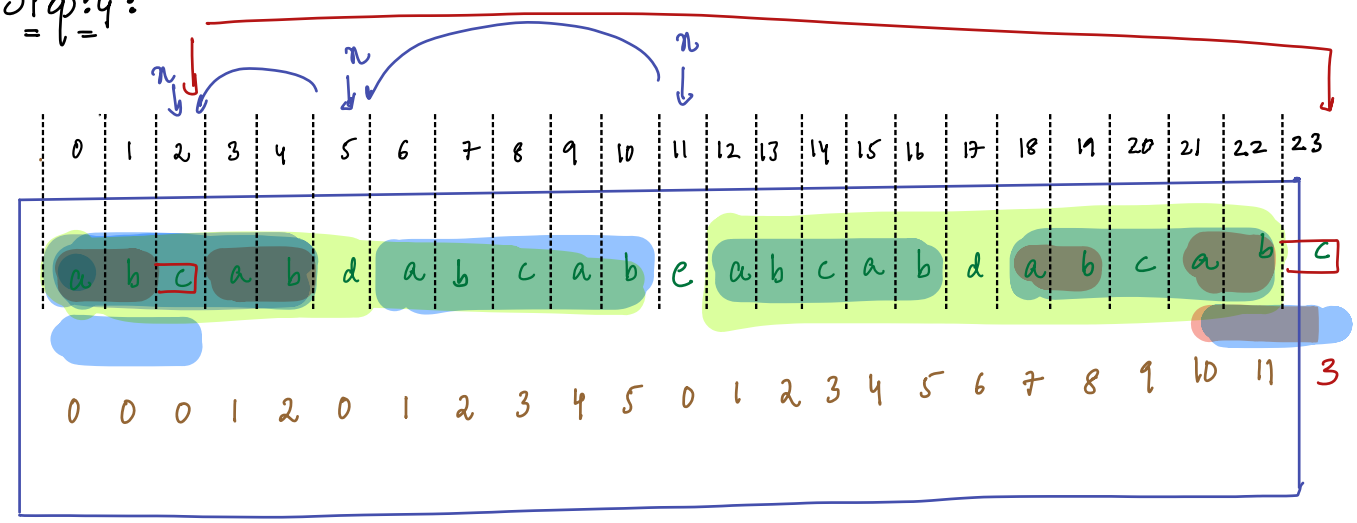
Step :3



i = 16, n = lps[i]

n	str[n] == str[i]		
7	str[7] == str[16]		n = lps[n-1]
3	str[3] == str[16]		lps[i] = n+1 ; lps[i] = 4

Step:4:



$n = lps[i-1] = 11$

n	$str[n] == str[i]$		
11	$str[11] == str[23]$		$n = lps[n-1]$
5	$str[5] == str[23]$		$n = lps[n-1]$
2	$str[2] == str[23]$		$lps[i] = n+1$, $lps[i] = 3$

	0	1	2	3	4	5	6	7
Ex:	a	b	a	d	a	b	a	c
lps[]:	0	0	1	0	1	2	3	

$i = lps[i-1]$

i	str[i] == str[j]	
3	str[3] == str[7]	$i = lps[i-1]$
1	str[1] == str[7]	$i = lps[i-1]$, $i = lps[0]$ $i = 0$ (out of bounds)
0	str[0] == str[7]	(At 0 not matching: $i = lps[i-1]$ & break)

// Pseudocode:

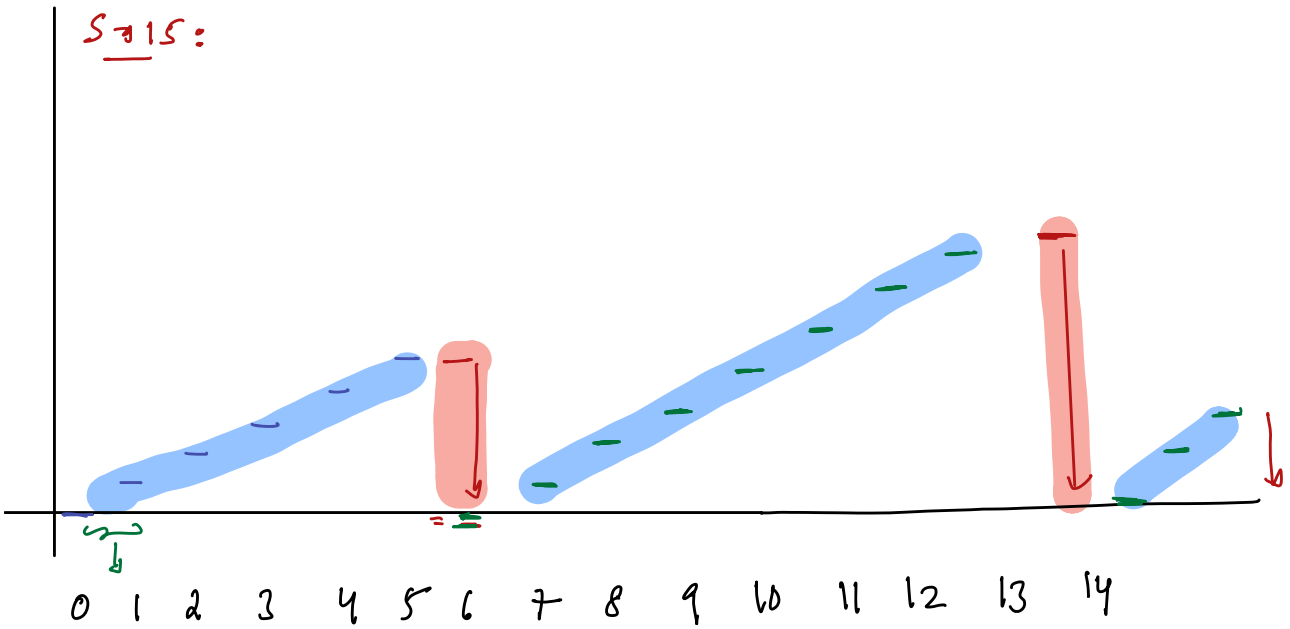
```
lpsVal(String s) {

    int N = s.size(); lps[N];        → 10:35 break
    lps[0] = 0

    for(i=1; i<N; i++) {
        // Say we need lps[i]
        i = lps[i-1]
        while(s[i] != s[i]) {
            if(i==0) { i=-1; break }
            i = lps[i-1]
        }
        lps[i] = i+1
    }
}
```

O(N)

//



Total inc steps in all val : O(N) → ? { At max we can inc → 1 }

Total dec steps in all val : O(N)

→ Total steps in all val : O(N)

// KMP : Pattern matching using LPS[] → KMP { Knuth - Morris - Pratt }

2Q] Given 2 strings A & B check if they are Anagrams → {V} Each other?

N N

Anagrams:

{ 2 strings A & B are said to be anagrams, if they are permutations of each other

Eg:

tac cat → YES

satya ytas → NO

lps pls → YES

Madam dadam → NO

1d 2d

ramas → sort → aamsr

smaa → sort → aamsr

Given strings A & B

Idea1: Sort & Compare ✓ → O(1)

Idea2: Take star of all character of A, B → {*}

A: aa B: bb } nr of all chr = 0 *

Idea3: { frequency of all character in Both strings have to be same, using hashmap }

{ Hashmap < char, Int > hm1, hm2
A → in hm1, B → in hm2 : O(N)

{ You need to check if hm1 & hm2 are exactly same }

TC: { N + N + 26 }

To insert in hm1, hm2 — to compare hm1 & hm2

28)

Given 2 strings A (N) & B (M), (N >= M), Count no. of substrings of A are anagrams of B

	0	1	2	3	4	5	6	7	8	9	10	11
Eg: A: (12)	a	b	c	a	c	c	b	a	b	c	a	c

B: a b a c c
(5)

Idea 1: For all substrings of len = 5 in A
Check if they are anagrams with B.

	A	→	B
0-4	a b c a c	→	a b a c c ✓
1-5	b c a c c		a b a c c
2-6	c a c c b		a b a c c
3-7	a c c b a		a b a c c
4-8	c c b a b		a b a c c
5-9	c b a b c		a b a c c
6-10	b a b c a		a b a c c
7-11	a b c a c		a b a c c

Idea2: Using Sliding Window

Ex: A: a b c a c a b a b c a c
(12)

B: a b a c c
(5)

{ We need to compare }

	Add	Rem	fA	c=0	fB == fA
0-4:			{a:2, b:1, c:2}	c=c+1	{a:2, b:1, c:2}
1-5:	5	0	{a:1, b:1, c:3}		{a:2, b:1, c:2}
2-6:	6	1	{a:1, b:1, c:3}		{a:2, b:1, c:2}
3-7:	7	2	{a:2, b:1, c:2}	c=c+1	{a:2, b:1, c:2}
4-8:	8	3	{a:1, b:2, c:2}		{a:2, b:1, c:2}
5-9:	9	4	{a:1, b:2, c:2}		{a:2, b:1, c:2}
6-10:	10	5	{a:2, b:2, c:1}		{a:2, b:1, c:2}
7-11:	11	6	{a:2, b:1, c:2}	c=c+1	{a:2, b:1, c:2}

// TODO

ans = 3

// Idea of how hm1 & hm2

```
if (hm1.size() == hm2.size()){
    // for every in hm1 the frequency
    in both hm1 & hm2 have to
    be same
}
```

→ Friday : Oops + LinkedList

Monday : ———

Wednesday : ———

Friday : ——

//

// Soy.

→ { LinkedList }

{ Trees }

{ Dp }

{ Graphs }

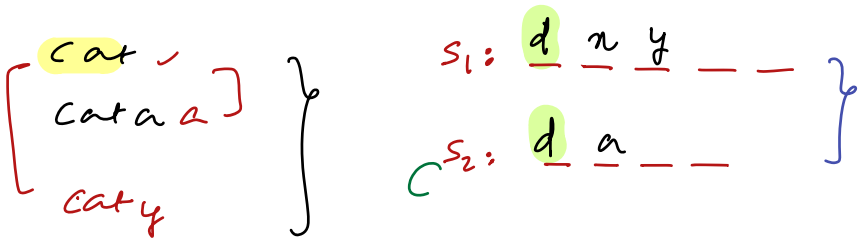
{ 9573546915 → { Saturday / Sunday } }

//→

//

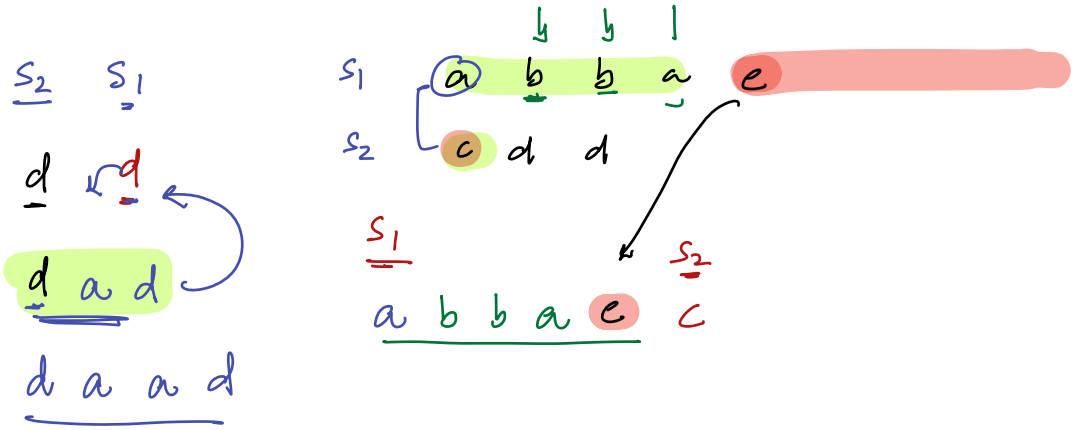
```
if (hm1.size() == hm2.size()){
    // for every in hm1 the frequency
    in both hm1 & hm2 have to
    be same
    for(auto x = hm1.begin(); x < hm1.end(); x++){
        k = x.first  v = x.second
        if( v != hm2[k]){
            return false
        }
    }
    return True
}
```

//



S_1 d n y b c

S_2 d a a c c



// S_1 S_2 :

// Decide whom sho com first y last