Agenda

i) Introduction to LPS
   - → prefix and suffix strings
   - → LPS of a string
   - → LPS[] of a string

ii) optimised code of LPS

iii) Pattern matching of a string

⇒ Given a string of N length:

what is **prefix strings** : substrings starting from 0^{th} index

what is **suffix strings** : substring ending at $n-1$ index

S : a b a b (indices 0 1 2 3)

Prefix	Suffix
a	b
ab	ab
aba	bab
abab	abab

S : a b a (indices 0 1 2)

prefix	Suffix
a	a
ab	ba
aba	aba

LPS of a string : length of longest prefix which is also a suffix. } exclude complete string }

s = a b c a b

lps(s) => 2

prefix	suffix
a	b
ab	ab
abc	cab
abca	bcab

s = a a a a

prefix	suffix
a	a
aa	aa
aaa	aaa

lps(s) => 3

s = a b c d a b c

prefix	suffix
a	c
ab	bc
abc	abc
abcd	dabc
abcda	cdabc
abcdab	bcdabc

lps(s) => 3

s = a b c a b

lps(s) => 2

prefix	suffix	
a	b	1
ab	ab	2
abc	cab	3
abca	bcab	4

(1) + (2) + (3) + ... (n-1)

≈ $O(n^2)$

to find LPS value of single string :

TC = $O(n^2)$

Q. Given a string s, return lps[].

lps[i] => lps value of substring 0 to i

	0	1	2	3	4	5	6
s =	a	a	b	a	a	b	c
lps[]	0	1	0	1	2	3	0

lps[0], s[0,0] = a, ans = 0 (-)

lps[1], s[0,1] = aa, ans = 1 (a)

lps[2], s[0,2] = aab, ans = 0 (-)

lps[3], s[0,3] = aaba, ans = 1 (a)

lps[4], s[0,4] = aabaa, ans = 2 (aa)

⋮

	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

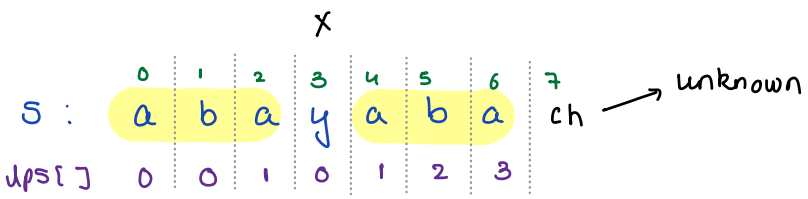
to calculate single string lps value, TC: $O(n^2)$

to calculate lps[], TC: $O(n^3)$

↓

$O(n)$

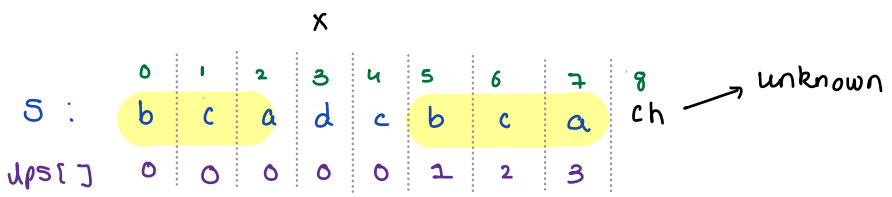
understanding examples




```
if ( s.charAt(i) == s.charAt(x)) {
        lps[i] = x+1;
}
```

i = 7

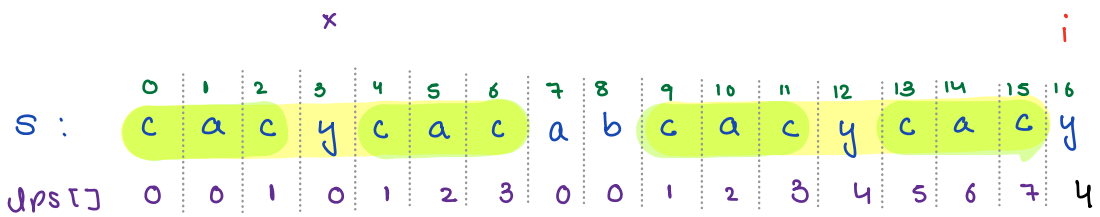
x = lps[i-1]
= lps[6]
= 3




```
if ( s.charAt(i) == s.charAt(x)) {
        lps[i] = x+1;
}
```

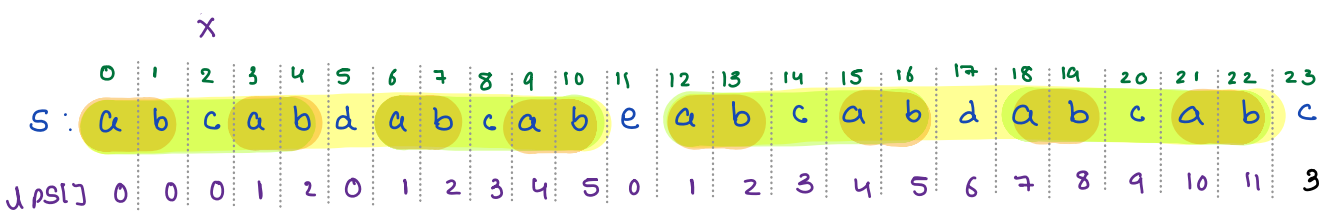
i = 8

x = lps[i-1]
= lps[7]
= 3



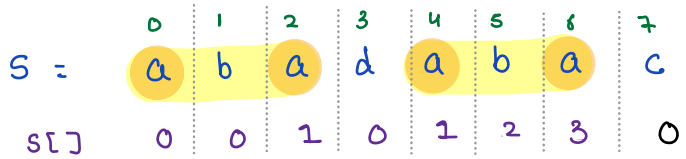
$i = 16, \quad x = lps[i-1] = 7$

x	s.charAt(i) == s.charAt(x)	action
7	s.charAt(16) == s.charAt(7) y == a	no, $x = lps[x-1]$ $x = lps[6] = 3$
3	s.charAt(16) == s.charAt(3) y == y	yes, ans ⇒ x+1



$i = 23, \quad x = lps[i-1] = 11$

x	s.charAt(i) == s.charAt(x)	action $x = lps[x-1]$
11	s.charAt(23) == s.charAt(11)	no, $x = lps[10] = 5$
5	s.charAt(23) == s.charAt(5)	no, $x = lps[4] = 2$
2	s.charAt(23) == s.charAt(2)	yes, ans = x+1 = 3



if (x == 0)
↳ don't use
lps[x-1]

i = 7, x = lps[i-1] = 3

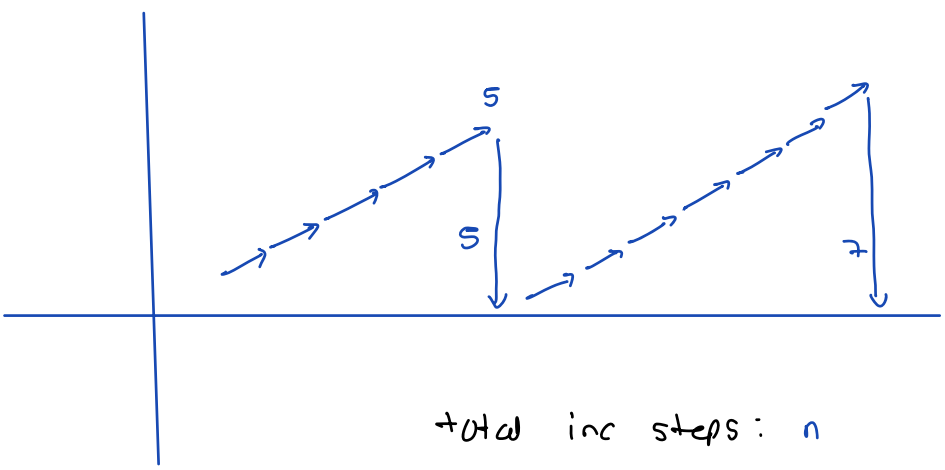
code

```
int[] LPS(String s) {
    int n = s.length();
    int[] lps = new int[n];
    lps[0] = 0;
    for(int i=1; i<n; i++) {
        int x = lps[i-1];
        while(s.charAt(i) != s.charAt(x)) {
            if(x==0) {
                x = -1;
                break;
            }
            x = lps[x-1];
        }
        lps[i] = x+1;
    }
    return lps;
}
```

	0	1	2	3	4	5	6	7
s :	a	b	a	y	a	b	a	b
lps[]	0	0	1	0	1	2	3	2

```
for(int i=1; i<n; i++) {
    int x = lps[i-1];
    while (s.charAt(i) != s.charAt(x)) {
        if (x==0) {
            x=-1;
            break;
        }
        x = lps[x-1];
    }
    lps[i] = x+1;
}
```

i	x	
1	~~0~~ -1	lps[1] = 0
2	0	lps[2] = 1
3	~~1~~ ~~0~~ -1	lps[3] = 0
4	0	lps[4] = 1
5	1	lps[5] = 2
6	2	lps[6] = 3
7	~~3~~ 1	lps[7] = 2



total inc steps : n

total dec steps : n

itr : 2n TC : O(n)

Q. Count occurrences of pattern P in Text T.

eg1

T = a a b a c d

P = a b a c

ans = 1

eg2

T = a b a d c a b a b a e

P = a b a

ans = 3

length of T => n

length of P => m

Brute force: Match every substring of P.length() in T with P

TC : O(n*m)

T = a b a d c a b a b a e

P = a b a

Searching P in T.

Str = P + "#" + T

	a	b	a	#	a	b	a	d	c	a	b	a	b	a	e
lps[]	0	0	1	0	1	2	3	0	0	1	2	3	2	3	0

```
if (lps[i] = P.length()) {
        ans++;
}
```

T = a b d a b d a a b e a b d a a

P = a b d a

Str = P + "#" + T

Str = a b d a # a b d a b d a a b e a b d a a

lps[] 0 0 0 1 0 1 2 3 4 2 3 4 1 2 0 1 2 3 4 1

```
if (lps[i] = P.length()) {
        ans++;
}
```

Algo → KMP Algo for pattern matching

↓

Knuth - morris - pratt

```
int patternMatching (String T, String P) {
    String str = P + "#" + T;
    int[] lps = LPS(str);
    int ans = 0;
    for (int i=0; i< lps.length(); i++) {
        if (lps[i] == P.length()) {
            ans++;
        }
    }
    return ans;
}
```

TC: O(n+m)

Doubts

i) Break : No class on sat & tues

next class → Thursday

topic → Linked list

(memory management & Class and object)

→ utilise your break

ii) Practice contest

watch recording of contest guidance class first and then