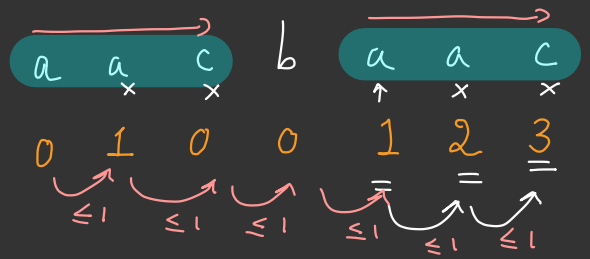
"Pattern Matching"

Pattern → abc

Text → xyz abc af abc

Concept . LPS



$O(N^3)$ Algo (Brute Force)

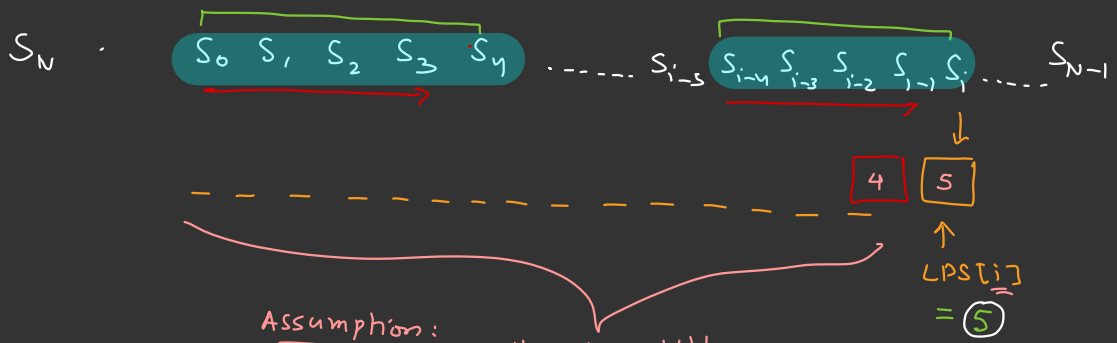
→ O(N) Algo (Pseudo Code) ← ?

"Hardest Algo in CS"

KMP Algorithm

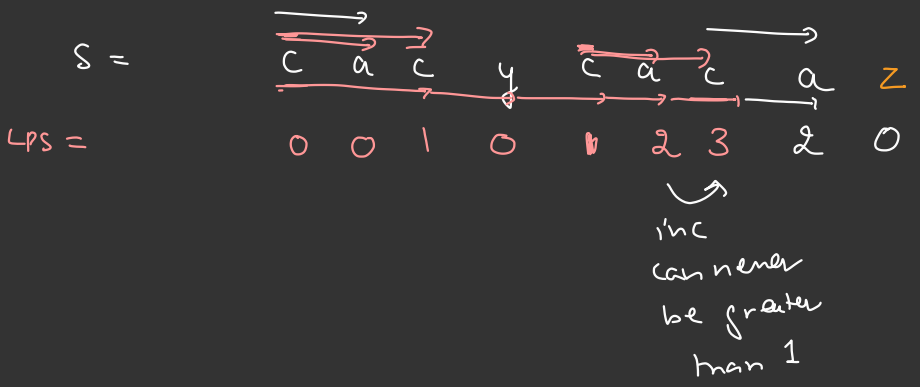
Scientists { Knuth, Morris, Pratt } 1970-77

Calc LPS[]



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

Claim : Observation-1 LPS[i] exceeds LPS[i-1] by atmost 1

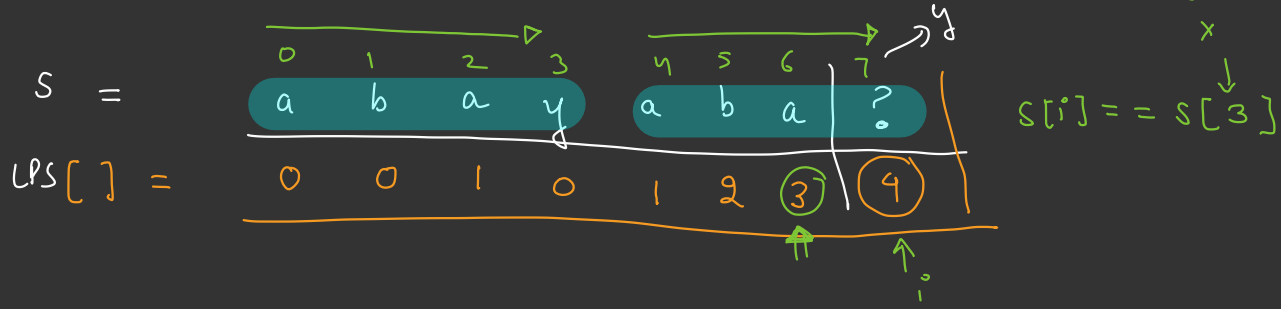


At max

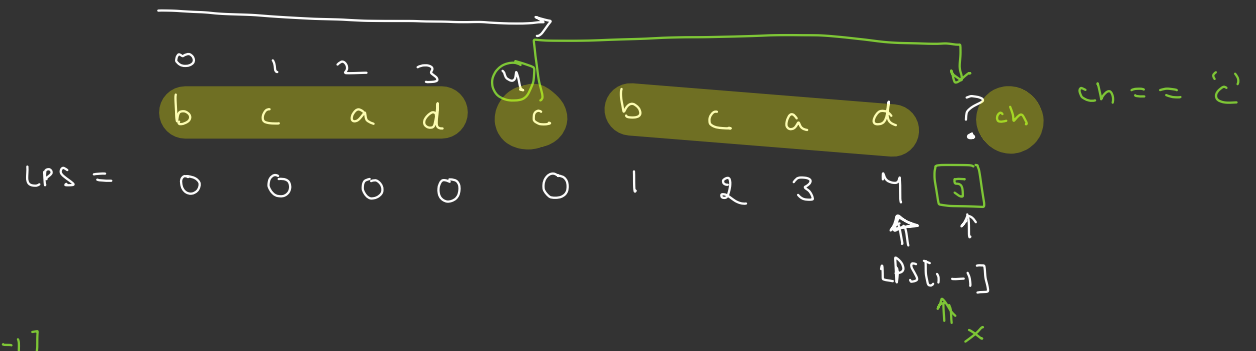
Claim $LPS[i] = LPS[i-1] + 1$

Step - 2

Ex-1

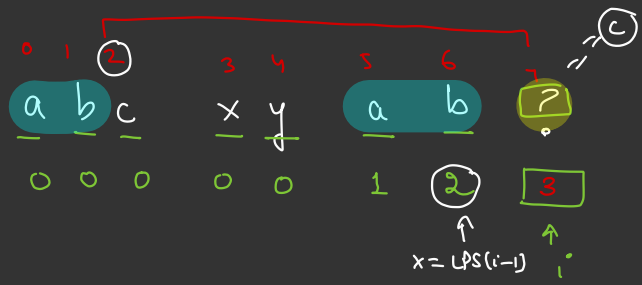


Ex-2




```
x = LPS[i-1]
if ( S[x] == S[i] ){
        LPS[i] = LPS[i-1] + 1
}
```

Ex-3

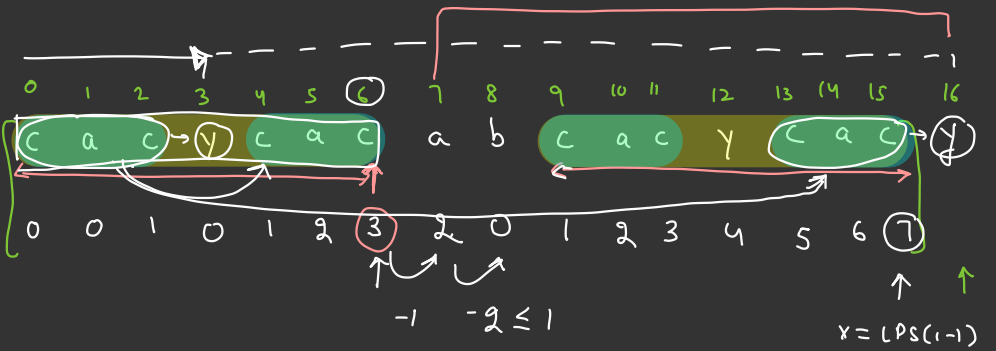


$x = LPS(i-1)$

if $(s[i] == s[x])$

$LPS(i) = 1 + LPS(i-1)$

Step -3



i = 16

x	str[i] == str[x]	
7	s[16] == s[7] y == a	No
3	s[16] == s[3] y == y	Yes

x = LPS[x-1]

x = LPS[6]

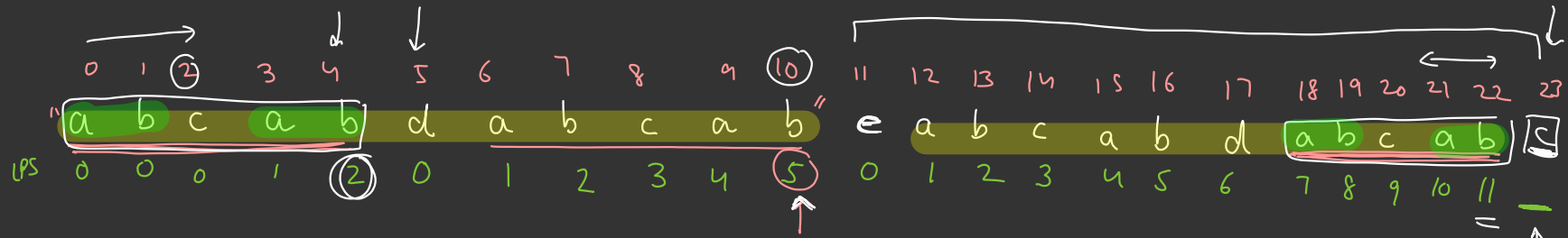
= 3

LPS[i] = x+1

= 3+1

= 4

finding Subpatterns that can possibly match



$x = LPS[i-1]$

$= 11$

Loop

x	Str[x] == Str[i]	
11	e == c	No
5	d == c	No
2	c == c	Yes

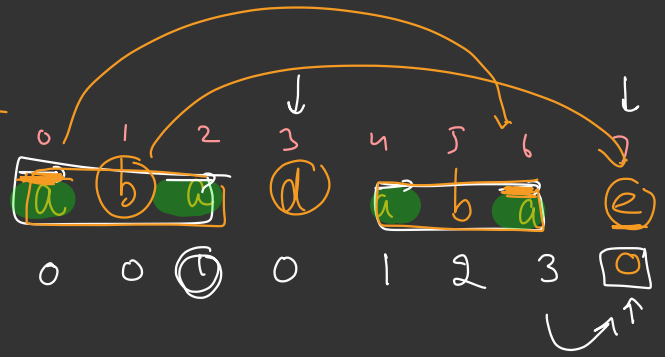
$x = LPS[x-1]$

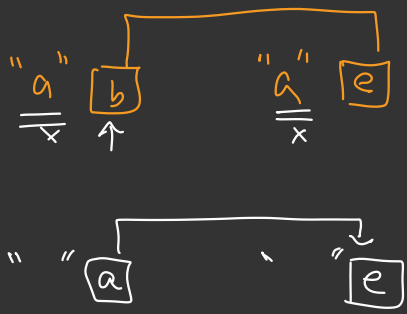
$x = LPS[x-1]$

$LPS[i] = x+1$

$= 3$

No Match





$x = LPS(i-1) = 3$

x	str[x] == str[i]	
3	d == e	NO
1	b == e	NO
0	a == e	NO

$x = LPS(x-1)$ Reducing pattern length

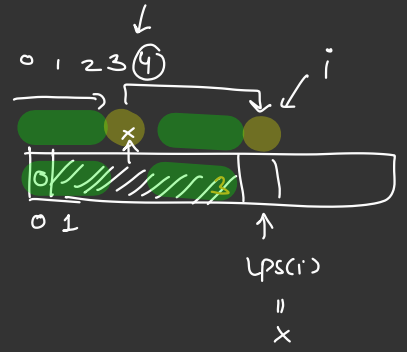
→ $x = LPS(x-1)$

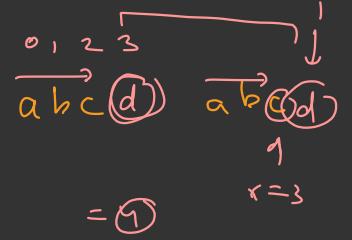
$x = LPS(x-1) = LPS[-1] = ?$

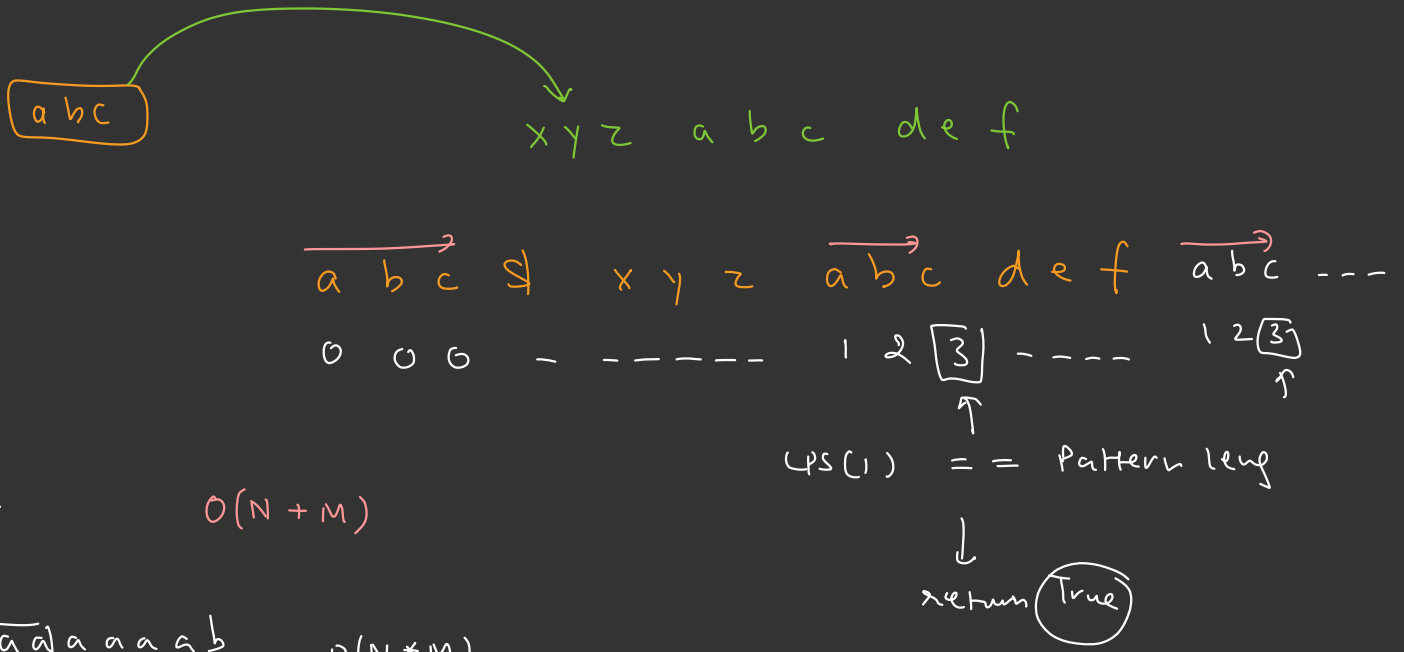
↳ Spl Case if (x == 0 && No match) Break;

Pseudo Code –

```
Lps Array ( String S )
    N = S.length()
    LPS[N].
→ LPS[0] = 0
→ for (i=1 to i< N ) {
        x = LPS(i-1)
        while ( Str[x] != Str[i]) {
          if(x==0) { x=-1, break; }
          x = lps[x-1]
        }
→       LPS[i] = x+1;   ← Match
    }
}
```







~~Brut~~ $O(N+M)$

1. aaaaaaab / aab → $O(N*M)$
2. LPS → $O(N^3)$
   - ↳ → $O(N+M)$
   - KMP

10.40

(a) Given a string of len N, calc min no of chars needed to add at start to make entre string as palindrome

Ex-1

+ dc → a b a c d

Left

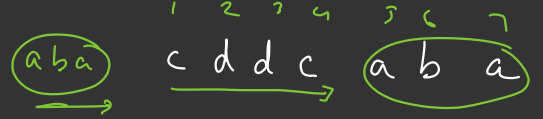
Ex-2

+ dca → a bb a a c d

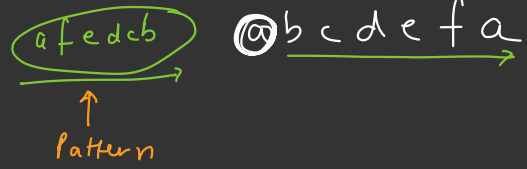
largest prefix which is a palindrome

Ans = N − len of largest prefix palindrome

Ex-3

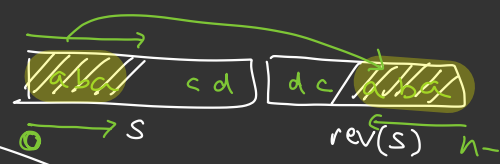


Ex-4

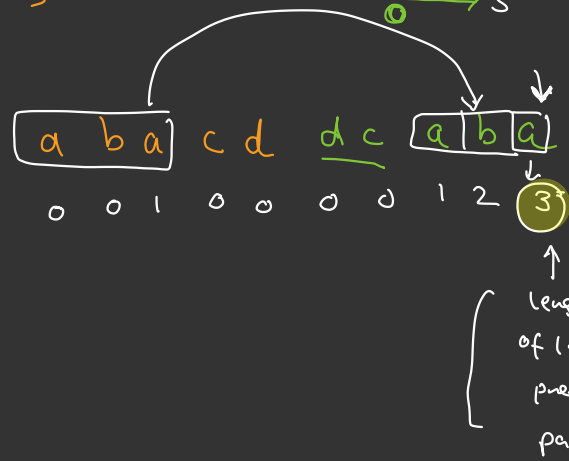


Logic

s = a b a c d



s' = s + rev(s) =

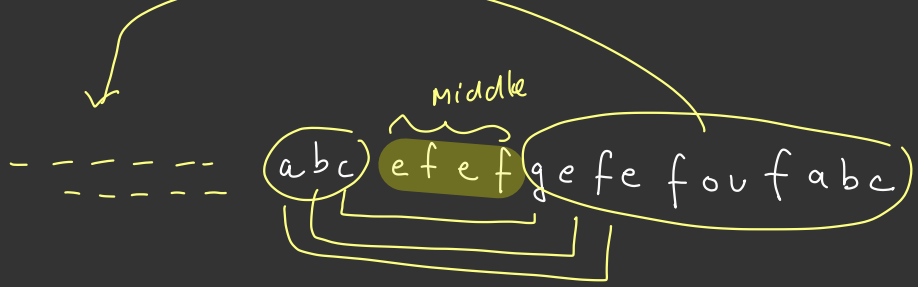


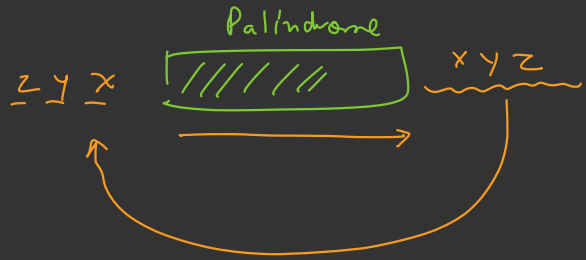
N − LPS(last_idx)

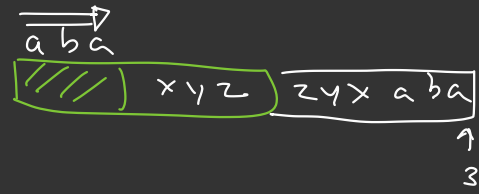
= 5 − 3

= 2

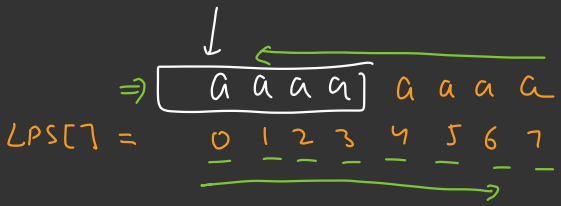
S = "a a a a"







S + "$" + rev(S)



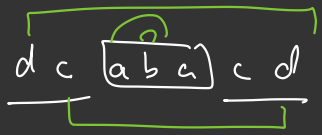
a a a a $ a a a a
0 1 2 3 0 1 2 3 4

N − len
= 4 − 4
= 0

s = a b a c d

	a	b	a	c	d	$	d	c	a	b	a
lps =	0	0	1	0	0	0	0	0	1	2	3

N = 5



ans = 5 - lps

= 5 - 3

= 2

Hashing