KMP

Text: abc abc abc abc

Pat: abc

Str: abc # abc abc abc abc

LPS:																		
								3			3			3			3	

LPS ↳ length of longest proper prefix which is also a proper suffix.

St:	a	a	b	a	a	c	a	a	b	a	a	b
LPS:	0	1	0	1	2	0	1	2	3	4	5	3
	0	1	2	3	4	5	6	7	8	9	10	11

Complexity

$i \rightarrow len$

$n \downarrow n$

$\rightarrow \Sigma i \Rightarrow i_0 \Rightarrow n_0$

$i_1 \Rightarrow n_1$

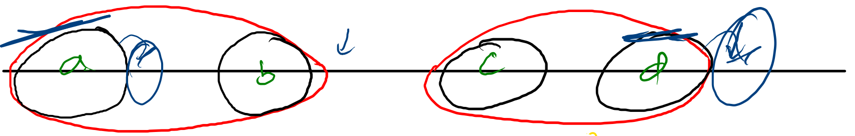
$i_2 \Rightarrow n_2$

$i_3 \Rightarrow n_3$

$O(n)$

```
p s void LPS(String st){
int[] lps = new int[st...];

 int i = 1;
  int len = 0;
while( i < st.length() ){
  if(st.charAt(i) == st.charAt(len)){
        len++;  ←
        lps[i] = len;
        i++;  ←
  } else {
     if(len > 0){
        len = lps[len-1];
     } else {
           i++;
  }    }
}
}
```



Shortest Palindrome

$$\begin{array}{cccccccccccc} & & & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \text{St:} & e & d & c & a & a & b & a & a & c & d & e \end{array}$$

string ↙ reversed ↙

LPS

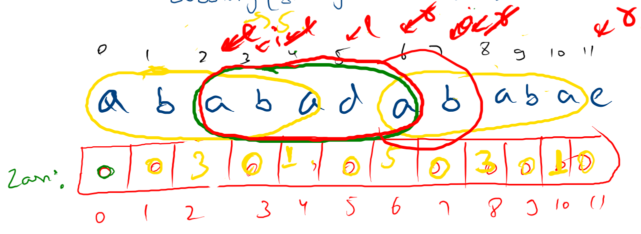
$$\text{Str: } \underline{a\ a\ b\ a\ a}\ c\ d\ e\ \#\ e\ d\ c\ \underline{a\ a\ b\ a\ a}$$

$\boxed{5}$

Z-algo

txt: abc abc abc ab c
Pat: abc

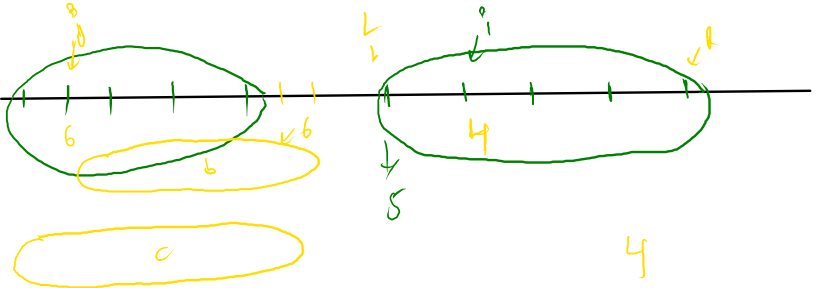
zarr → longest length proper prefix which is also a substring (starting at ith index)

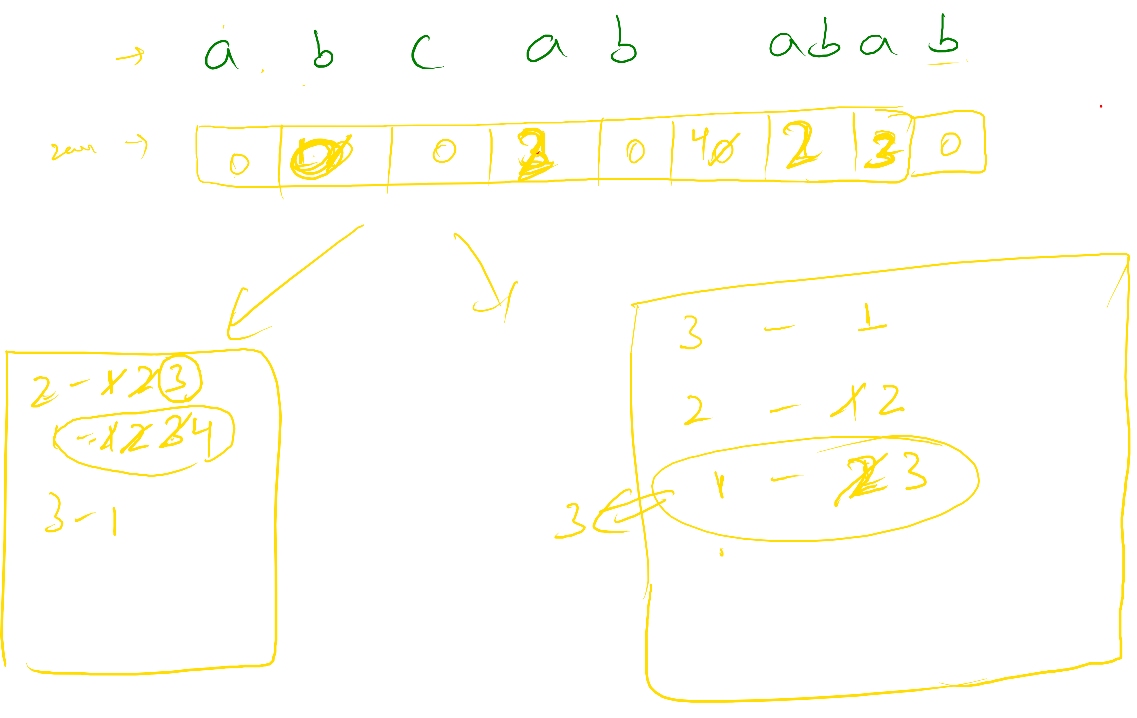


i_0 → r_0 → 5
i_1 → r_1 → 1
i_2 → r_2 → 1
i_3 → r_4 → 1

```
P S void Z(String st) {
int[] zarr = new int[st.length()];
int i=1;
int l=0, r=0;
while(i < st.length()) {
  if(i<=r){
  zarr[i] = math.min(r-i+1, zarr[i-l])
  }
  while(i+zarr[i] < st.length() &&
        st[i+zarr[i]] == st[zarr[i]]){
        zarr[i]++;

  }
  if(i+ zarr[i] - 1 > r){
     l = i;
     r = i + z[i]-1;
  }
  i++;
}
```





_ _ _ _ _ _ 10%6 = 4

a 4 4 4 4

Hashmap

↳ subarray sum ⇒ tar?:

⇓

1st elem ⇒ remove

2nd elem ⇒

2+1st elem ⇒

3+1st elem ⇒