Z-Algorithm

⇓

{Pattern matching Algorithm.}

{Rabin Karp.
Z-Algo.
KMP.} → Rolling Hash.

Q: Given a string s and a pattern t. find if there exist a substring in S which matches the pattern t.

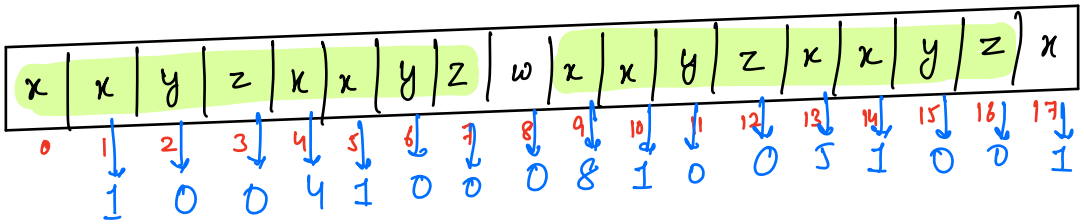
Q: Given two strings A and B. find no. of times B occurs in A as a substring.

Given a string s.

For every index i, find the length of longest substring from it which is equal to prefix of whole string.

⇓ starting from 0^{th} idx.

abcdef.



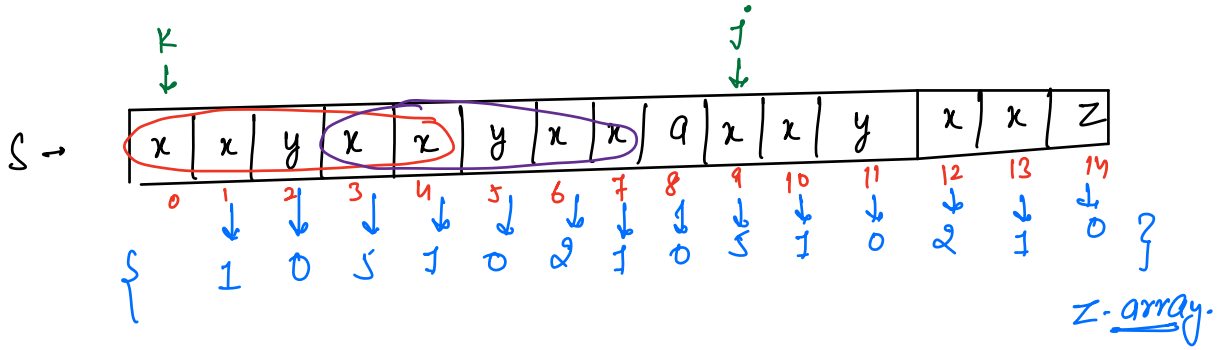
x x

xy ≠ xx

xyz ≠ xxy

⋮ ≠

You need to create Z[].



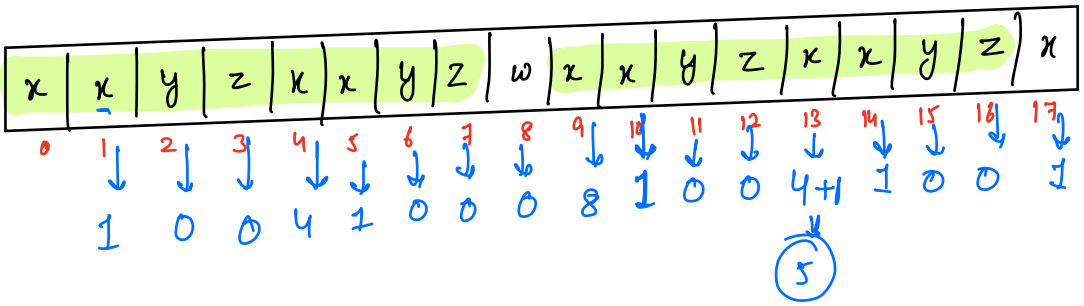
pseudo-code-

```
Z[N];
for( i=1 ; i<N ; i++){
        j = i , k = 0 ;
        while( j<N && S[j] == S[k] ){
                j++
                k++
        }
        Z[i] = k
}
```

{T.C → $O(N^2)$}

⇓

$O(N)$



$x = x$

$xy \neq xx$

xyz

$xyzx$

$xyzxx$

$xyzxxy$

$xyzxxyz$

x

xy

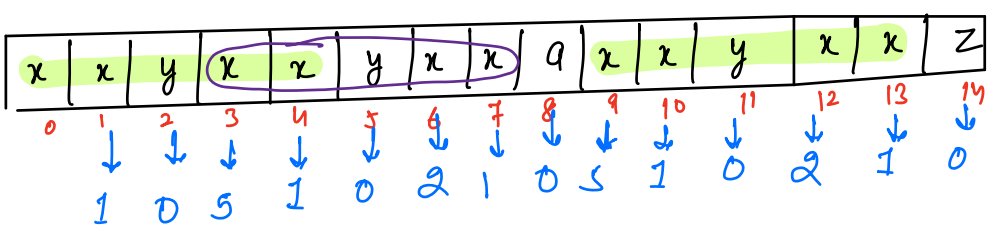
xyz

$xyzx$

$xyzxx$

$xyzxxy$

$xyzxxyz$



* { z-value of big. brother } < { no. of matching characters } ⇒ trust & copy answer

idea:

if char → not inside green box {segment}

↓

Brute force → update segments.

else

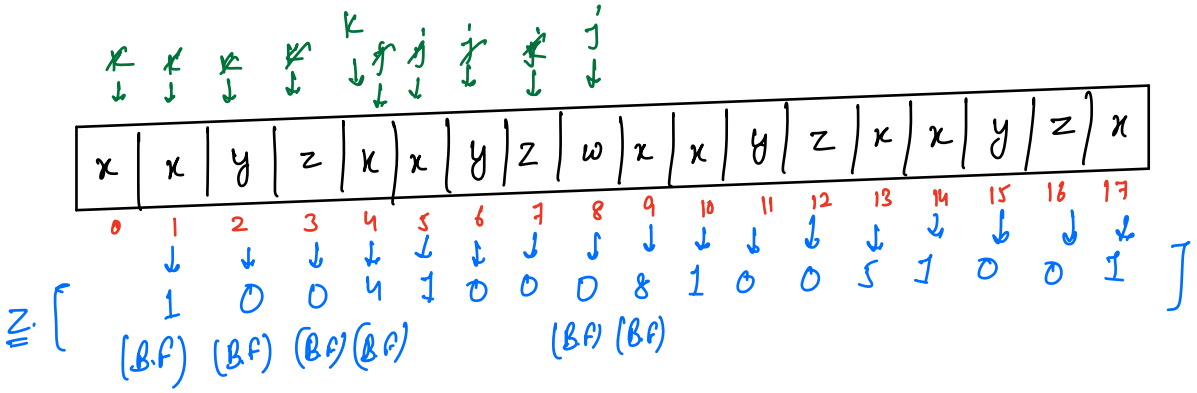
talk to elder brother.

z(brother) < common characters

trust & copy

z(brother) ≥ common characters

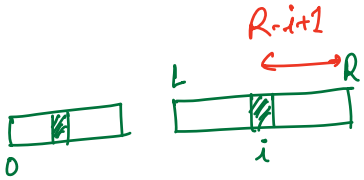
common characters + B.f length



⇒ Every element will be considered only once in the Brute force.

// maintain the segment.

```
L = 0 , R = 0
for ( i = 1 ; i < N ; i++ ) {
    if ( i > R ) {          // curr. element is not lying inside green box.
        j = i , k = 0
        while ( j < N && s[j] == s[k] ) {
            j++ , k++
        }
        z[i] = k
        L = i , R = j - 1   // update the segment
    }
    else {
        if ( z[i-L] < R-i+1 ) {
            z[i] = z[i-L]
        }
        else {
            j = R+1 , k = R-i+1 ;
            while ( j < N && s[j] == s[k] ) {
                j++ , k++
            }
            z[i] = k
            L = i , R = j-1   // update the segment
        }
    }
}
```



L → 0
L+1 → 1
L+2 → 2
L+3 → 3
i → i-L

T.C → O(N)
S.C → O(N)

Q: Given a string s and a pattern t. Find if there exists a substring in S which matches the pattern t

S = my name is rahul

p = meis

str → meis $ my name is rahul

0 0 0 0 1 0 0 4

[∀i, z[i] = pattern.length
true.
else, false.]

Q: Given two strings A and B. Find no. of times B occurs in A as a substring.

B → abc

A → d a b c a b a a b c a b c c

Str = B + $ + A.

= a b c $ d a b c a b a a b c a b c c

0 0 0 0 3 0 0 2 0 1 3 0 0 3 0 0 0

z →

∀i, z[i] = pattern.length ⇒ count++

→ Watch recording once & then forget on Saturday.

⇒ [L-2 Trees]

{ You need to revise it before 2-3 days of OA / interview. }