Pattern matching 2

Monday, 12 April 2021 8:06 PM

⇒ Z-Algo:

↳ Z-Value: for each character we calculate Z Value

Z[i] = length of longest substring starting at (i) which is also a prefix string

Str: a b c a b c x a b

Z[]: N 0 0 0 3 0 0 0 2 0

a / a / a b

a b / a b / a b c

a b c

a b c x *

a b c a b c x a b ✓

Str: x a y x a b x a y x a b

Z[]: 0 0 0 2 0 0 6 0 0 3 0 0

x / x a / x a b / x a y / x a y x / x a y x a b

x / x a / x a y / x a y x / x a y x a b

S[i]

a_i prefix string

$a_i\, a_{i+1}$ ✓

$a_i\, a_{i+1}\, a_{i+2}$ ✓

Str2: x a b y x a b y x a

Z[]: 0 0 0 8 0 0 0 4 0 0 0 0

[4,4] x - [0-0]

[4,5] x a - [0-1]

[4,6] x a b - [0-2]

[4,7] x a b y - [0-3]

[4,8] x a b y x - [0-4]

[4,9] x a b y x a - [0-5]

[4,10] x a b y x a b - [0-6]

[4,11] x a b y x a b y - [0-7] → 8

[4,12] x a b y x a b y a - [0-8] → x a b y x a b y x

⇒ [0-0] x *

Str4: a a a a

Z[]: 0 4 3 2 1

[2-2] : a : [0-0] ✓

[2-3] : aa : [0-1] ✓

[2-4] : aaa : [0-2] ✓

[1-1] : a : [0-0] ✓ start

[1-2] : aa : [0-1] ✓ 1st & character

[1-3] : aaa : [0-2] ✓ 1st 3 character ✓

[1-4] : aaaa : [0-3] ✓ len = 4

[1-5] *

Text ⇒ a b c x a b a a b c x

Pattern ⇒ a b a a

Str: a b a a $ a b c x a b a a b c x

Pattern | Text

Z[]: 0 0 1 1 0 2 0 0 0 4

a ✓

a b ✓

a ✓

a b ✓

a b a ✓

a b a a ✓

a b a a b * a b a a $

⇒ Given T & P ← [Searching in T] Note:

M N

Step1: Construct New String

Str = P $ T

Step2: For str construct Z[]

Step3: If for any of Z[i] value = N ⇒ [Pattern found] ✓

If N is found x times in Z[]: Pattern is repeating x times in Text

when we consider T & P

2QD If 2 String A & B of same length N. find if any cyclic Rotation of B is same as A.

[C-E char by char]

A = a b a c

B = a c a b

B' = b a c a

B^2 = a b a c = A] True

A = a b c B = a b c [True]

Eg: A = a b a c B = a a b c

Sol2: Sort & Comp most work

caab

baca

abca

aabc

Sol:

Get 1 by 1 rotation of B & comp to A.

⇒ TC: N × [N]

TC: O(N²) SC: O(1)

// If a rotation of B = A

A = $A_1 A_2 A_3$

B = $B_0 B_1 B_2 B_3$ = A

B' = $B_3 B_0 B_1 B_2$ = A

B^2 = $B_2 B_3 B_0 B_1$ = A

B_3 = $B_1 B_2 B_3 B_0$ = A

⇒ New update Str = $B_0 B_1 B_2 B_3 B_0 B_1 B_2 B_3$ Str = B B

All rotation of B are present

Now I just have to check if A is present in Str

A: Pattern

Str: Text

⇒ A $ Str & get Z[]

Str: a b c a a b c a a b c

Z[]: 0 0 0 1 1

String, str = N, Z[]

Z[0] = 0; // Insignificant

for(int i=1; i<N; i++) {

P_1 = 0, P_2 = i;

while(P_2<N && str[P_1] == str[P_2]) {

Z[i]++; P_1++; P_2++;

}

}

TC: O(N²)

Str: x x y a x x y a x x y a x x y d x x y

Z[]: 0 1 0 0 4 1 0 0 2 1 0

8th char is our 8 segment ⇒ [4-7]

↳ we have to make & calculate Z[]

Str1: x x y a x x y a d x x y x x y a x x y d

Z[]: 0 1 0 0 4 1 0 0 8 1 0 0 4 1 0 0 4

3 1 0 7 3

Z[5] Z[6]

Given String S, length: N

int Z[N] = 0;

int s = 0, e = 0;

for(int i=1; i<N; i++){

if(i > e) & Z[i] = 0 // outside seg

else, & Z[i] = min(Z[i-s], e-i+1);

// inside segment

// Z[i] ⇒ Iterate on Z[i]

int P_1 = Z[i] int P_2 = i+Z[i]

while(P_2<N && S[P_1] == S[P_2]){

P_1++; P_2++; Z[i]++;

}

// Check Z[i] Value

// we update [s e]

// i ⇒ Z[i] : New end : i+Z[i]-1

if(e < (i+Z[i]-1)){

update s&e

s = i; e = i+Z[i]-1

}

}

for(int i=1; i<N; i++){

if(i ≤ e){

Z[i] = min(Z[i-s], e-i+1);

}

int P_1 = Z[i], P_2 = i+Z[i]

while(P_2<N && S[P_1] == S[P_2]){

P_1++; P_2++; Z[i]++;

}

if(e < (i+Z[i]-1)){

s = i; e = i+Z[i]-1

}

}

⇒ O(N)

TC:

char if not in segment

Z[i-s] Z[i-s]

⇒ Check & Z[i] value ≥ ① : Z[i] = 0

char in in segment

Z[i] = Z[i-s]

s start of segment → ②

Z[i] is max possible one

→ we further have to check → ③

Step3 Z[i] with = Z[i-s]

Z[i] = min(Z[i-s], e-i+1)

end of segment

How much we can extend on right side

Step1: i is not in segment

Z[i] = 0

Step2: i is in in segment

Z[i] = Z[i-s] ← start of segment

Z[i] = min(Z[i-s], e-i+1)

copy

max extend on right

we have min Z[i]

Step3: Get actual Z[i] by extending

Step4: Extend your boundaries