Contest Discussion

Content

01. Toggle case
02 Positive in Range
03 Super stream Engineers

01. **Toggle case**

You are given a character string str having length N, consisting of uppercase & lowercase characters. You have to toggle every character.

str = "YogEsH" ⇒ yOGeSh

str = "scAleR" ⇒ SCaLEr

'A' = 65 'a' = 97

'B' = 66 'b' = 98

'C' = 67 'c' = 99

⋮ ⋮

'Z' = 90 'z' = 122

```
String ans="";
for(i=0; i<n; i++){
    char ch=str[i]
    if(ch≥'A' && ch≤'Z'){
        ch=(char)ch+32;   // ch=(char)(ch-'A'+'a');
    }
    else{
        ch=(char)ch-32;
    }
    ans=ans+ch;
}
```

uppercase ⟶ lowercase

ch = 'A' ⟶ ch: 'A' - 'A' + 'a'

ch = 'B' ⟶ ch: 'B' - 'A' + 'a'

* lowercase ⟶ uppercase

ch = 'a' ⟶ ch: 'a' - 'a' + 'A'

* By using XOR operator

* In our lower case characters → 5th bit is set

In our uppercase characters → 5th bit is unset

'a' 97 = 1 1 0 0 0 0 1 (bits 6 5 4 3 2 1 0)

↓ unset the 5th bit

'A' 65 = 1 0 0 0 0 0 1

```
for ( i=0 ; i<n ; i++){
    char ch = str[i];
    char newch = ch ^ 32;
    ans = ans + newch;
}
```

TC : O(n)
SC : O(1)

* Positive in Range

You have been given an array A with profit for N days. You also have Q queries represented by a 2D Array B.

For every query, your task is to find the count of non negative profit in range A[L] to A[R]

$$A = \{\ \underset{0}{1}\quad \underset{1}{-1}\quad \underset{2}{-2}\quad \underset{3}{5}\quad \underset{4}{0}\ \}$$

(n) → length

B

L	R
0	3
1	4
0	4

→ 2

→ 2

→ 3

* Solution

→ Create a pf array which holds the count of non-negative no. at every 'i' idx.

```
int [] pf = new int [n]

Pf [0] = A[0] ≥ 0 ? 1 : 0
```

```
for (i=1; i<n; i++){
    if (A[i] ≥ 0){
        pf[i] = pf[i-1] + 1;
    }
    else {
        pf[i] = pf[i-1] + 0;
    }
}

// answer all queries

int[] ans = new int[B.length]

for (i = 0; i < B.length; i++){
    int l = B[i][0]
    int R = B[i][1]

    if (l == 0) { ans[i] = (pf[R]) };
    else    ans[i] = pf[R] - pf[l-1];
}
```

TC : O(n+B)

SC : O(n)

* Inplace pf array

```
A[0] = A[0] ≥ 0 ? 1 : 0

for (i=1; i<n; i++){
    if (A[i] ≥ 0)   A[i] = A[i-1] + 1
    else    A[i] = A[i-1] + 0;
}
```

* Superstream Engineers

Given an array A, where each entry represents the ack time for individual packet, and two integers B & C. Determine if there's a continous sequence of B packets with an avg ack time less than or equal to C.

window size = B

avg of elem in window $\leq C$

$A = [\underset{0}{30}, \underset{1}{25}, \underset{2}{18}, \underset{3}{22}, \underset{4}{15}, \underset{5}{40}]$

$B = 3$

$C = 30$

$avg = \frac{30+25+18}{3} = 24.33 < C$ return 1

```
int sum = 0
for (i=0; i<B; i++)
    sum = sum + A[i];
if (sum/B ≤ c) return 1;
```

```
int s = 1
int e = B
while (e < n) {
    sum = sum - A[s-1] + A[e];
    if (sum/B <= c) return 1;
    s++;
    e++;
}
return 0;
```

A = [30 , 25 , 18 , 22 , 15 , 40]
0 1 2 3 4 5

B = 3

C = 20

st	end	
0	2	$\Rightarrow 24 \leq 20$ X
1	3	
2	4	
3	5	

sum = 0

sum = 30 + 25 + 18

$= \frac{73}{3} \leq 20$ X

s = 1

e = 3

sum = 73 − 30 + 22 = $\frac{65}{3}$

s = 2

e = 4