Leetcode Virtual Contest - 271

Problem - 1 {Difficulty - Easy}

Stat: Rings and Rods

Cond^{n}: The first char of the i^{th} pair denotes the i^{th} ring's color ('R', 'G', 'B').

The second char of the i^{th} pair denotes the rod that i^{th} ring is placed on ('0' to '9').

// Return number of rods that have all those colors of rings

Input: rings = "B0B6G0R6R0R6G9"
Output: 1

Thought Process:

by reading problem stat. get an idea count no. of ~~xxx~~ rings of all diff color.

↓

if count is even of each color

↓

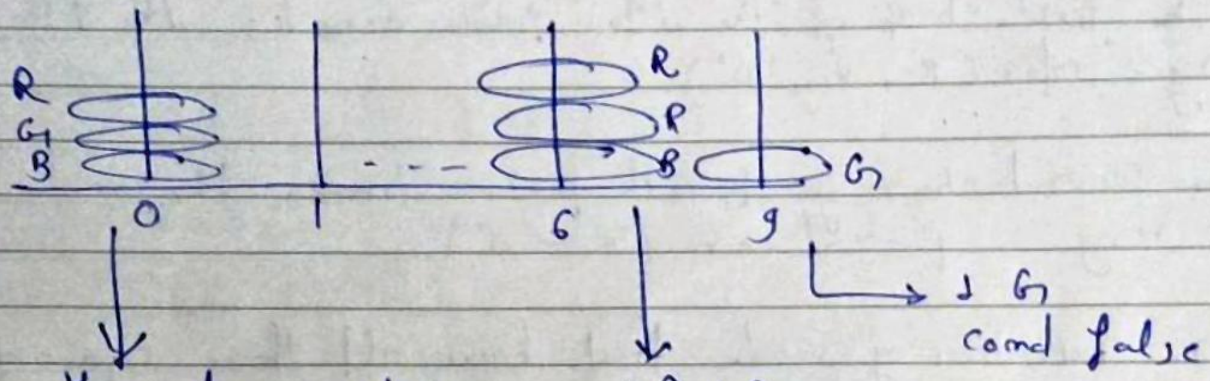
2-B, 2-R, 2-G → make 2 pair

↓

if count is odd make odd pair

like ↓R, ↓G, ↓B

↓



↳ ↓G cond false

0th index contain all three color of ring ⇒ cnt = 1

2R, 1B cond^n false

ans = 1 //.

code: // take 3-vector → R, G, B of size (10)
↳ total no. of read.

// int cnt = 0 ; int ringrod ⇒

// for (i = 0 to i = n)

ringrod = rings[i+1] - '0' . → string to int

// if (rings[i] == 'R') { red[rings[i]] = true } ;

// → same with green and blue

// for (i = 0 to 10 ; i++)

if (red[i] and green[i] and blue[i])

cnt++

return cnt;

Problem-② {Difficulty- Medium}

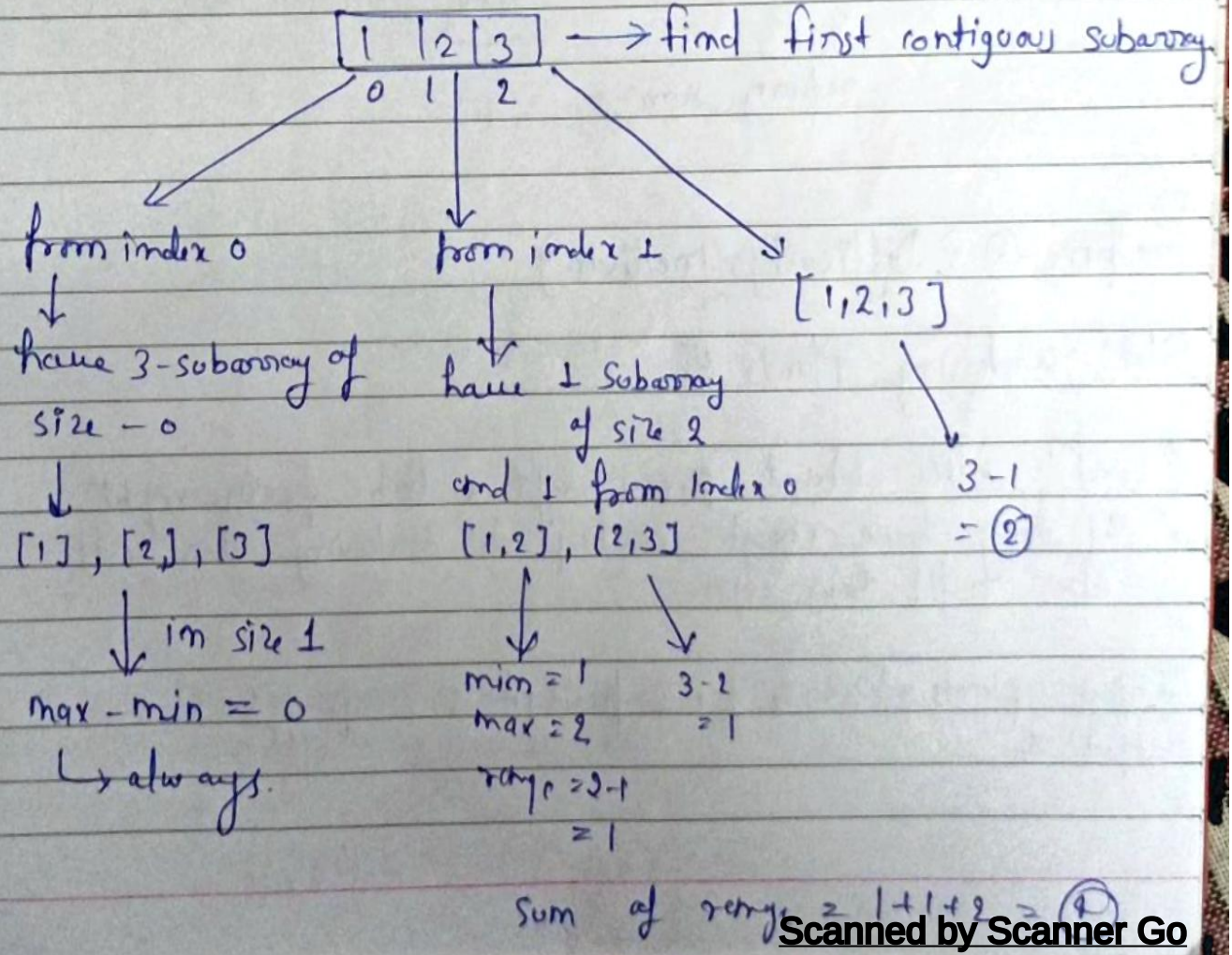
Stat: Sum of subarray Ranges

Condⁿ: given an array, ranges of subarray of nums is the difference b/w the largest and smallest element in the subarray.

Return sum of all subarray ranges of nums.

Input: nums = [1,2,3]
Output: 4

Thought Process:



we maintain min^m and max^m every subarray have diff max and min.

↓ better understanding with

Code Hint:

```
// BC : if (size == 20) → return 0
// int sum = 0
// for (i = 0 to i = n)
//    int mini = INT_MAX
//    int maxi = INT_MINI
// for (int j = i ; j < n ; j++)
{
    mini = min(mini, num[j]);
    maxi = max(maxi, num[j]);
    sum += maxi - mini;
}
return sum //.
```

Problem - ③ {Difficulty - medium}

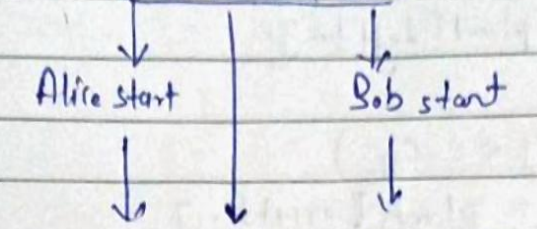
Stat: Watering Plants II

// cond^n Alice start from left, Bob from right
// if they have enough water to watering then cont.
else refil their can

Input: plants = [2,2,3,3], capacityA = 5, capacityB = 5
output: 1

Thought Process:

plants: [2|2|3|3] $c_A = 5$, $c_B = 5$

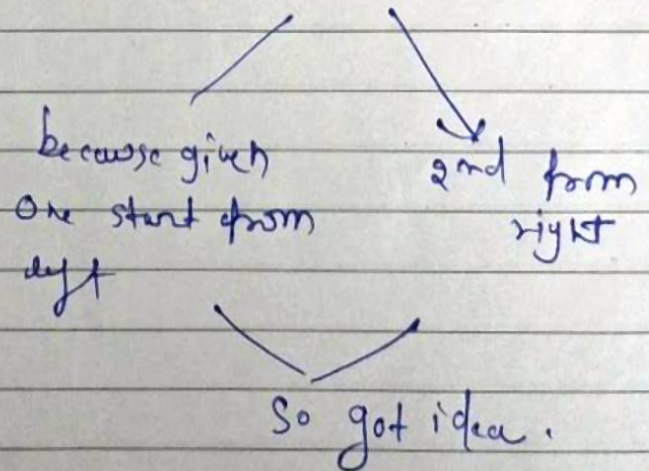


5-2 = 3 | 3-2 = 1 | 5-3 = (2) ↓ don't have water to watering plant n-1 So it refill

In this only one time they refil so ans = (1) //

⟹ We used two pointer approach.

How I know



Code line: int ans = 0, int left = 0, right = n-1, a = c_A, b = c_B

```
while ( left < right)
{
if (plants[left] <= cA)
```

```
        cA -= plants[left++];
    else
        {
            ans++;
            cA = a - plant[left++];
        }
    if (plants[right] <= cB )
        cB -= plants[right--]
    else
        {
            ans++
            cB = b - plant[right--];
    // if (left == right)
        ans += max (cA, cB) < plant[left] ? 1 : 0
    // return ans;
```