Problems on Arrays..

Q.1 Special subsequences "ag"

Find how many subsequences of string "ag" are present in given string.

eg. a b c g a g ⇒ ans = 3

obs: ① 'g' comes before 'a' → contribute to answer NO!

..... g .... a .... ⇒ No match

--- a ---- g ---- g .... ⇒ 2 matches.

Brute Force: For every 'a' check how many 'g' occur after this ind

```
count=0
for (i=0 ... i<n, i++)
{   if (s[i]=='a')
    {
        for (j=i+1; j<n; j++)
        {   if (s[j]=='g')
                count++;
        }
    }
}
return count;
```

a g a g a g a g

T.C ⇒ $O(N^2)$

Obs ②

a b c g a g a a g

a b c g a g a a g b c a

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

O(N)

Rolling count of 'g'

update

ans + 1 => 1

+1 = 2

+2 = 4

+3

T.C => O(N)

S.C ~~O(N)~~ => O(1)

right

a a b c g b a g c g a g

g_count =

g_count = 1

ans = 0

ans + = g_count

ans = 1

g_count = 2

g_count = 3

ans + = g_cor

ans = 1 + 4

= 5

g_count = 1

T.C => O(N)

S.C => O(1)

ans = 5 + 4 = 9

ans = 9 + 4 = 13

ans =

a b c g | a g | a a g

'ag'

cont = 3

T.C => O(N)

S.C => O(1)

ans = 3

cont = 2

ans = 5

cont = 1

ans = 6

= 7

cont = 0

a ... g ... a ... a ... g ... g ... a ... g

4 3 2 1

cont g => 4 → O(N)

3 O(N)

2

1

Q.2 Return this 2-D array (N×N)

N = 15

N = 4

$$\begin{array}{c} \\ 0\ 0\ 0\ 1 \\ 0\ 0\ 2\ 1 \\ 0\ 3\ 2\ 1 \\ 4\ 3\ 2\ 1 \end{array} \quad \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}$$

N = 3

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 1 \\ 3 & 2 & 1 \end{bmatrix}$$

N

N

Brute → 2 Loops

T.C ⇒ $O(N^2)$

N=15 → 15 14 13 ... 2

a[N][N]

pseudo

```
for(i=0; i<n; i++)
    for(j=0; j<n-i-1; j++)
        a[i][j]=0

~~for(i=0; i<n; i++)~~
    count = i+1
    for(j=n-i-1; j<N; j++)
        a[i][j]= count--;
}
```

Q.3 Closest Min Max !

Given array "A", find size of smallest subarray that contains both minimum & max. value (array can have duplicat

eg ⇒ [1, 2, 3, 1, 3, 4, 2, 4]

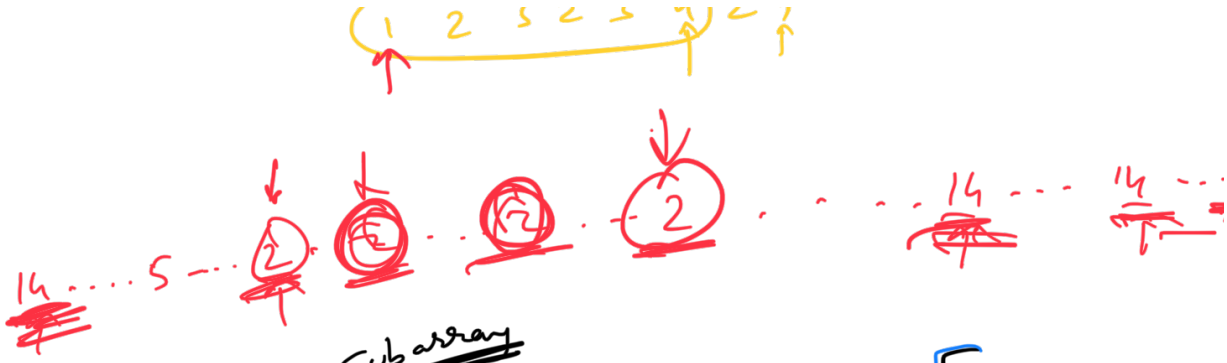
Ans = 3

min = 1
max = 4

size = 2

[1 4] size = 2

[2 7 4 -1 6 5 1]



Obs ①: Subarray

[min - - - - - - - max] or [max - - - - - mi

if ans

ans? No.

[... min .... max ....] ✗

Brute:

① Find 'min' & 'max' element → O(N)

[1 2 3 1 3 4 2 4]

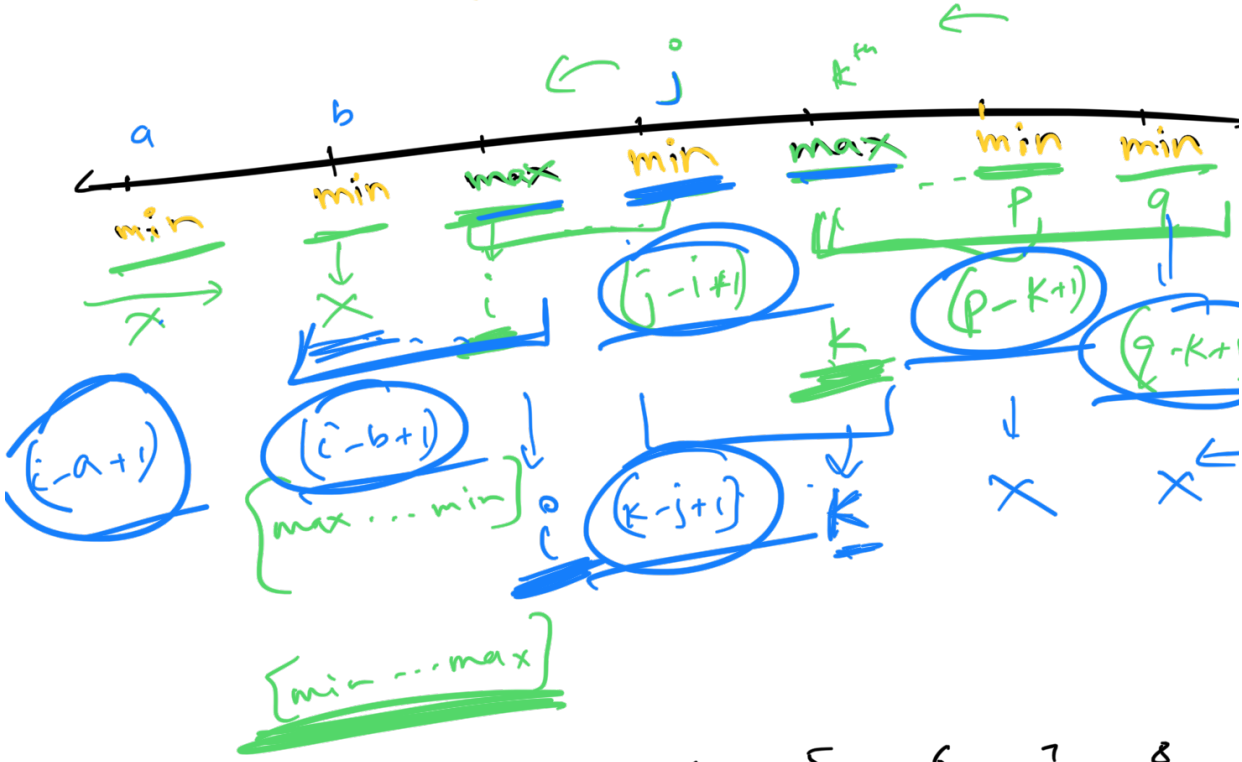
↳ Min = 1

↳ Max = 4

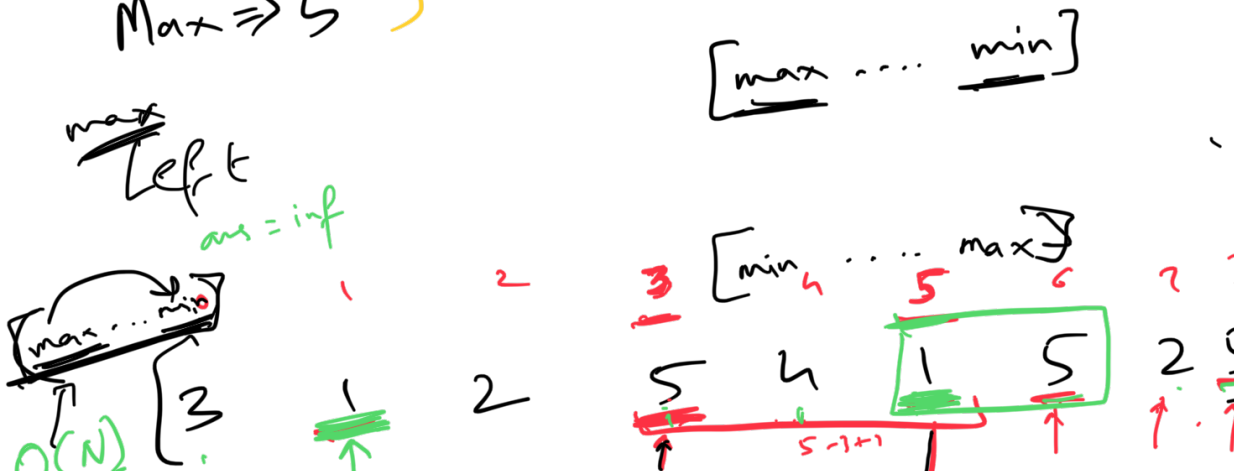
② 2 for loops, For every min, find 1st max on both sides.

[4 3 2 1] ⇒ 4

max
min
min
min
min
m
O(N)
O(N)


a
b
j
k^th
min
max
min
min
min
max
p
q
(i-a+1)
(i-b+1)
(j-i+1)
(p-k+1)
(q-k+1)
(k-j+1)
max ... min
min ... max

index ⇒ 0 1 2 3 4 5 6 7 8
A = [3 1 2 5 4 1 5 2 5]
infin
Min ⇒ 1
Max ⇒ 5
⇒ O(N)


[max .... min]
max
Left
ans = inf
[min .... max]
max ... min
O(N)
1 2 3 4 5 6 7
3 1 2 5 4 1 5 2
5-1+1

Left: inf, inf, inf, 3 → 3, 3, 6, 6, 8

Left: inf, 3

Right: 3, 3, 3, 3, 6, 6, 6, 8, 8

3-1+1 = 3

6-5+1 = 2

O(N)

[min ... max]

$T.C \Rightarrow O(N)$

$S.C \Rightarrow O(1)$

Similar to 1st question

-10^{10} $(-10^9, 10^9)$ $+10^{10}$

int_ma

int_m

min = inf, max = inf

Find min/max

```
for (i=0; i<n; i++)
{
    min = min (a[i], min)
    max = max (a[i], max)
}
```

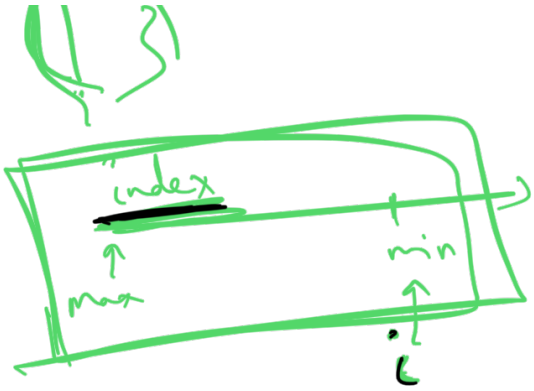
Last ind = -inf; ans = N;

```
for (i=0; i<n; i++)
{
    if (a[i] == max)
        index = i
    else if (a[i] == min)
    {
        ans = min (ans, i - index + 1)
    }
```

→ O(N)

Left

For Right → index - i + 1



0 1 2 [3 4 5]

5 - 3 + 1

[6 6 6 6]

abs (index - i) + 1

Q.4 Given a string of 0's & 1's, swap 'atmost' one '0' with one '1'. Find length of longest contiguous substring possible with all 1's.

eg. S = "1110111100 11"

length = 8

11 1 1 1111 0001

S = "011100" → 3. → No change.

110 1110 11110 10 11011 ⇒ 8

atmost	≤
at least	≥
exact	=

① Find longest 1s

1111 0 1111 000 1111110 1

9

11 0000 1 0 → 3

111 0 0 0010

Obs ①

2+1+0

0+1+1=2

00 1110 11110 0 11 0001

1Ls + 1 + 1Rs

0+1+2=3

0+1+0 =1

3 + 1 + 4

4+1=5

Corner Case

1111110

000 1110111 0000 → 6

L + 0 + R

3 + 0 + 3 = 6

9 → 0000 1110 11111000

3 + 1 + 5 = 9 → 8

00000 11111111

For each 0,

check ans if we swapped this 0.

Right

Left + 1 + Right

1 1 0 0 1 0 1 1 1 1 0

Left[] 1 2 0 0 1 0 1 2 3 4 0

how many 1's are on left including this in

0 → 1 → 2 → 0 → 0 → 1 → 0 → 1 → 2

Right[] 2 1 0 0 1 0 4 3 2 1 0

For '0' at i^{th} position,

max (Left[i-1] + 1 + Right[i+1]) ⟹ ans

corner (ans > count_of_total-1 ans = count_of_total-1)

1 1 1 0 1 1 1

3 + 1 + 3 → 7 − 1 ⟹ 6

0 1 1 1 0 1 1 0 0 1 1 0

L [0 1 2 3 0 1 2 0 0 1 2 0]

3 + 1 + 2 = 6

R [0 1 2 1 0 2 1 0 0 2 1 0]

0+1+3=4

3+1+2=6

2+1+0=3

0+1+2=3

2+1+0=3

Ans = 6

0 1 1 1 1 1 0 0

0 1 1 0 1 1 0 → 5

L [0 1 2 0 1 2 3 0] ⇒ O(N)

R [0 2 1 0 3 2 1 0] ⇒ O(N)

2+1=3

2+3+1=6

3+0+1=4 → O(N)

6 → 5

T.C ⇒ O(N)

S.C ⇒ O(N)

S.C ⇒ O(1)

1 1 1 0 0 1 1 1 0 1 1 1 0 0 1 1 0 1

2+1+1

1 1 1 1 1 1 1

4+1+3

1st 2nd

1111111 0 11 0 0 0 1 1 1 1 1