Contest Discussion

↳ Starting in T + 10 mins

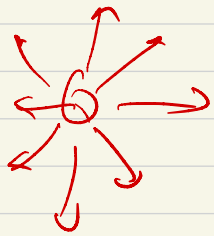
Given a chess Board Configuration with a Queen and a Rook.

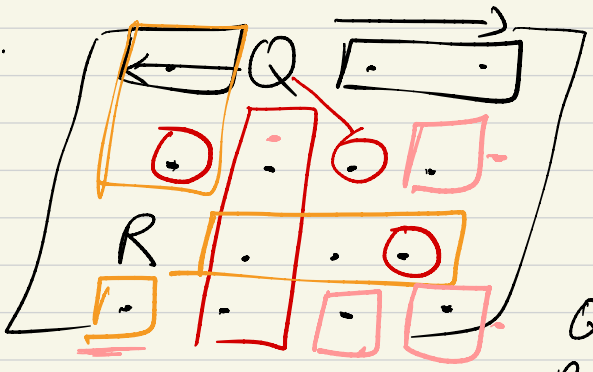
↳ output ⇒ Number of Safe Cells!

N x M

4 x 4

Input:

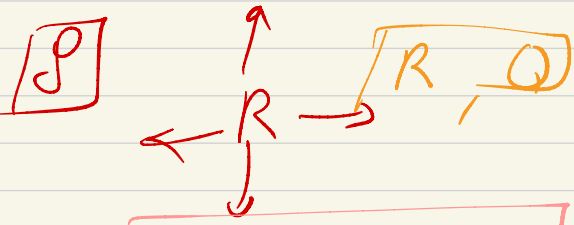




↳ . ⇒ Empty Square

Q → Queen

R → Rook

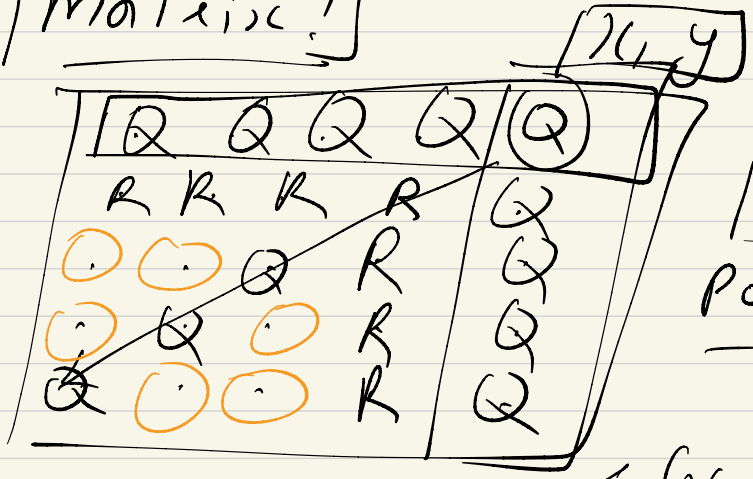


cnt ⇒ 3

Answer ⇒ 3

Solution :→

Matrix !



Whole row → x
whole Column → y
Diogonal

(x, y)
↓
(x+1, y+1)

Whole Row → x
Whole Colum → y

Answer → 6

Pseudocode :-

```
int SafeSpaces (char mat[][], int N)
{ #Find Position of Rank
    int x, y; Position Queen
    for(      i=0; i<N; i++)
```

```
for(j=0; j<N; j++)
    if mat[i][j] == 'Q'
        x = i; y = j;
        break;
```

Row x { int R, C → Position of Rook.

```
{for(j=0; j<N; j++) {
    mat[x][j] = 'Q';
}
```

col y

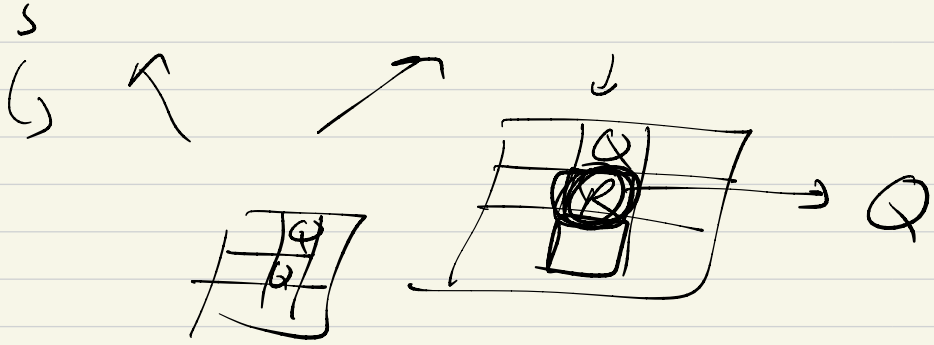
```
for(i=0; i<N; i++)
    mat[i][y] = 'Q';
```

```
i = x;
j = y;
while(i<N and j<N) {
        i++;
        j++;
        mat[i][j] = 'Q';
}
i = x;
j = y;
while(i<N and j≥0) {
        i++;
```

Q(i, y)

(x+1, y-1)

```
        j--;
    mat[i][j] = '0';
```




```
for (i=0; i<N; i++)
    for(j=0; j<N; j++)
        if mat[i][j] == '.'
            cnt++;

return cnt;
}
```

Time Complexity
↳ O(N×N)

Space complexity
↳ O(1) constant

#2

MAXSPROD

Given an Array of integers (size N)

Find maximum special Product possible in this array!

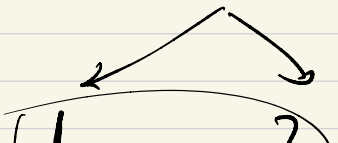
Special Product for `i` is $j \times k$

where A[j] previous greater value

A[k] is next greater value

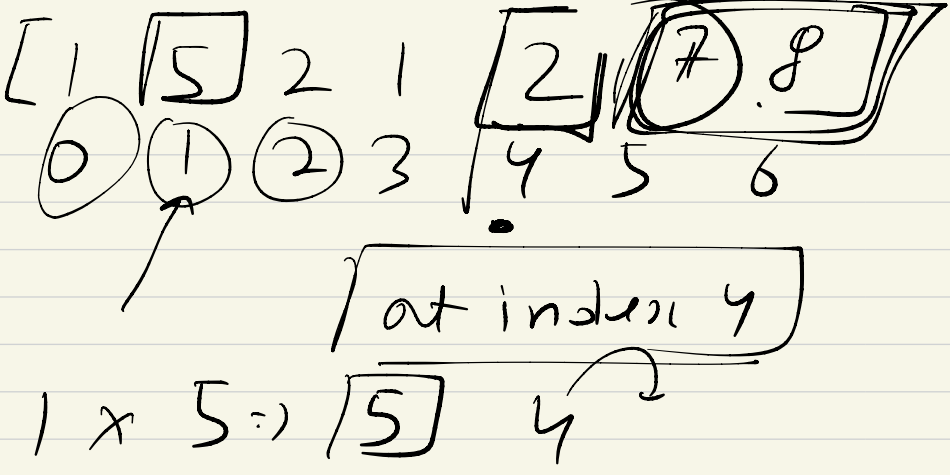
A = [1, 5, 2, 3, 2]

0 1 2 3 4



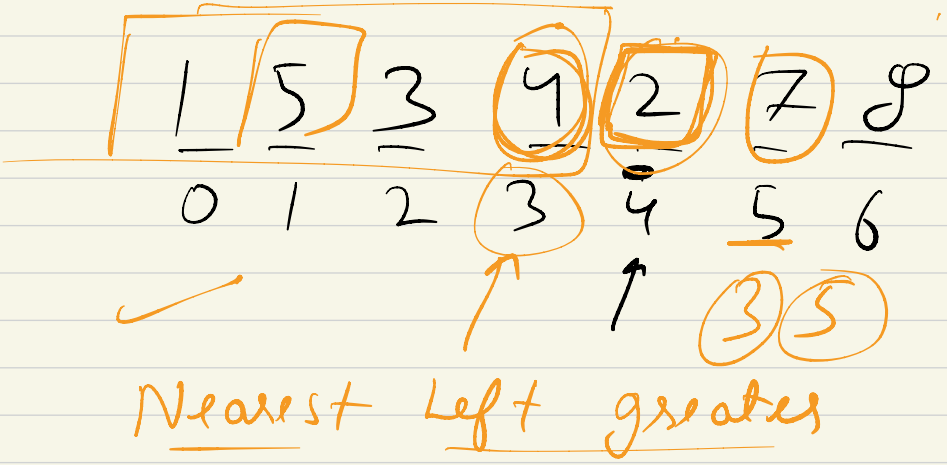
Special Product at index => 2

1×3 | 3



Just Previous greater

Next Greater Element

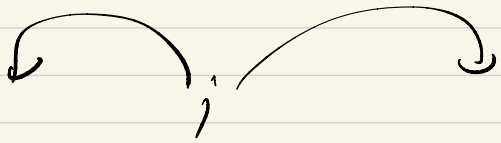


Question ? Yes or No!

Nearest Greater on Left!

[Nearest Greater on Right]

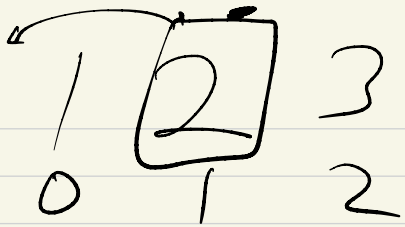
Stacks

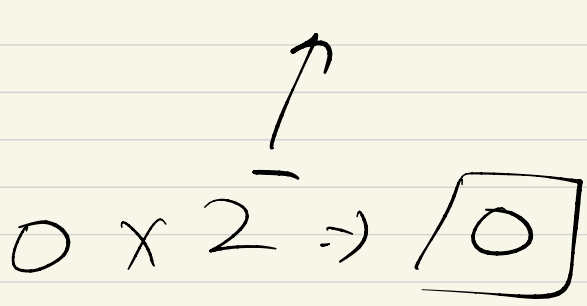


Pseudocode :->

```
int maxProd ( int arr[] ) {
    int left[] = nearest greater Left (arr);
    int right[] = nearest greater Right (arr);
    int product = 0;
    for (int i=0 ; i< N; i++) {
        int P= left[i]x right[i];
        product = max (product, P);
    }
    return product;
}
```

Revisit





Jumping Along Towers

Given an Array of integers (size N)

Initially you are at index 1

Goal is to reach index N of this array. Min cost

Jump #1 $i^{th} \rightarrow (i+1)^{th}$ $\rightarrow B \times |A[i+1] - A[i]|$

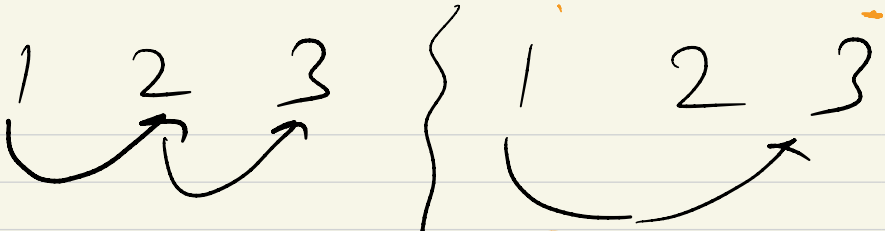
Jump #2 $i^{th} \rightarrow (i+2)^{th}$ $\rightarrow C \times |A[i+2] - A[i]|$

Input

-> [1, 2, 3] -> A

2 -> B

3 -> C



$2 \times (2-1) + 2 \times (3-2)$

4

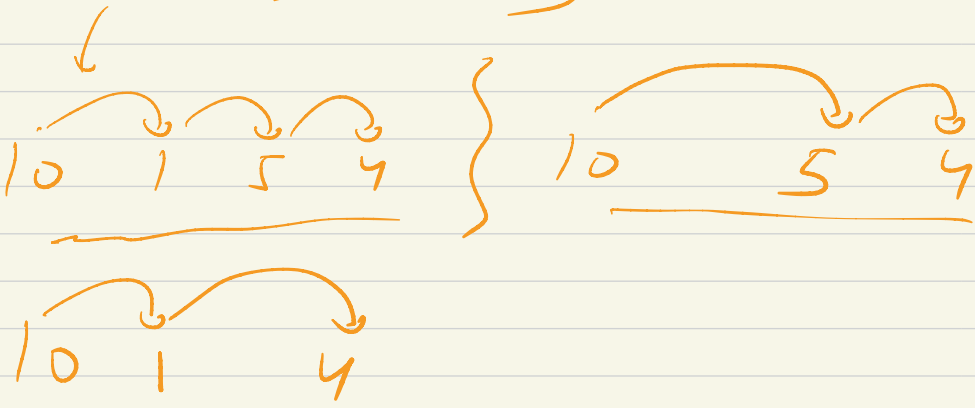
?

$3 \times (3-1)$

6

Answer => 4

[10, 1, 5, 4]



Consider all Possibilities

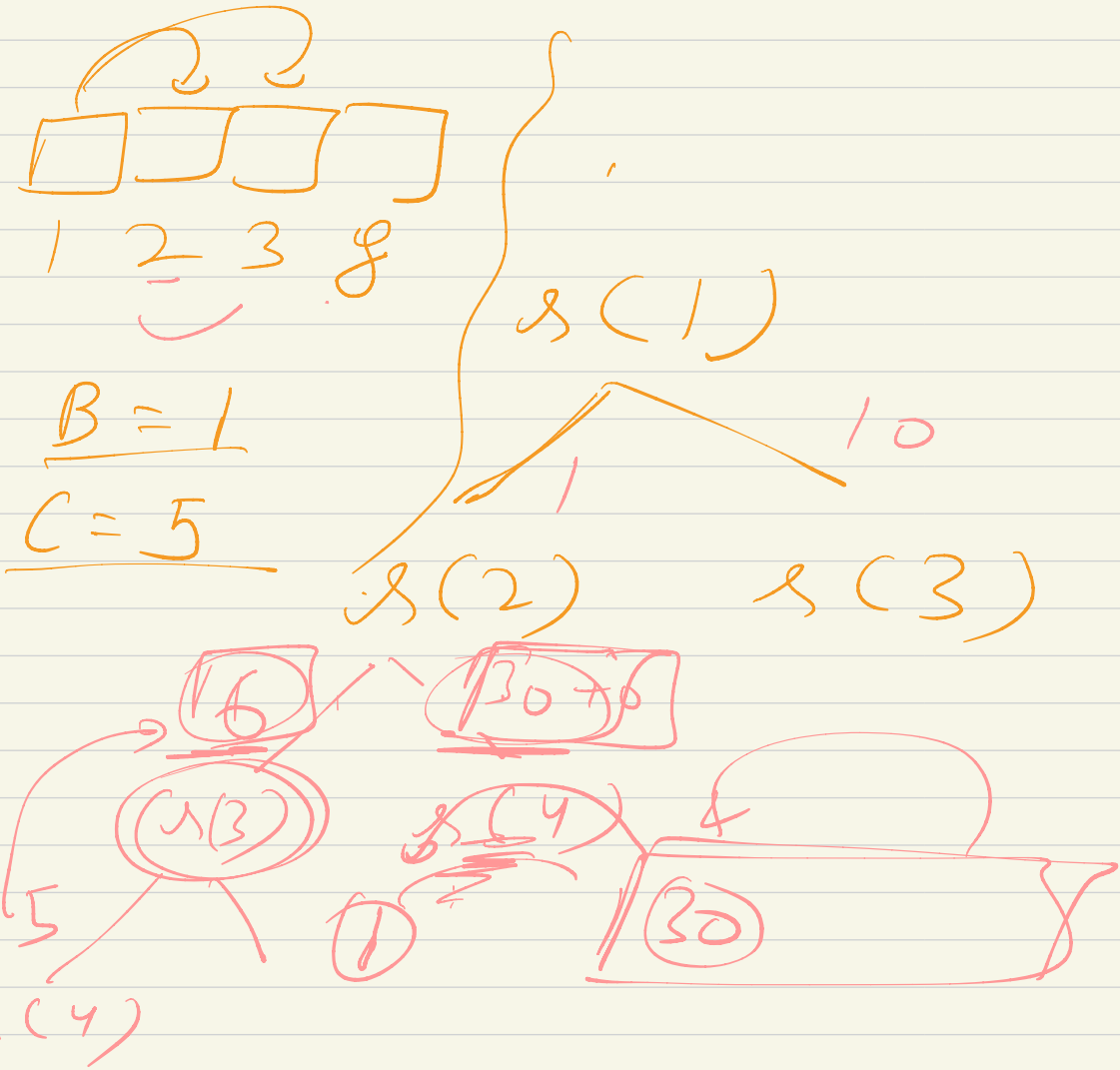
Recursion

```
int recursion(int pos, int[] A, int B
                                  int C)
{
    if (pos == N)
        return 0;
    if (dp[pos] != -1)
        # Jump 1 Place ahead
    int cost1 = B x abs(A[pos+1] - A[pos])
                + r(pos+1, A, B, C);

    # Jump 2
    if (pos+2 ≤ N)
    int cost2 =
        C x abs(A[pos+2] - A[pos]);
        + r(pos+2, A, B, C);

    return dp[pos] = min(cost1, cost2);
}
```

Memoization ?

DP -> Dynamic Programming


0
1 2 3 4 5
Overlapping
Subproblem
O(N)



Time → O(N)

Space → O(N)

In a trouble

Given an integer Array A of size N., Perform N operations.

Each operation :→

1) Select minimum (A) and maximum from this Array. B

↳ Find Avg ($\frac{A+B}{2}$)

↳ Replace → (A) ← with A + Avg

↳ Replace → B with → B - Avg

A = [3, 2, 7]

A = 2 { Minimum }

B = 7 { Maximum }

Avg = $\left\{\frac{2+7}{2}\right\}$ -> 4

N times

A -> A + AVG

B -> B - AVG

2 -> 2 + 4 (6)

7 -> 7 - 4 {3}

{3, 2, 7}

{3, 6, 3}

{3, 6, 3}

A = 3
B = 6
} Avg → 4

A = 3 + 4 { 7 }

B = 6 − 4 { 2 }

{3, 2, 7} → 2 operation

→ 1 operation

Answer

3, 3, 6

Hints

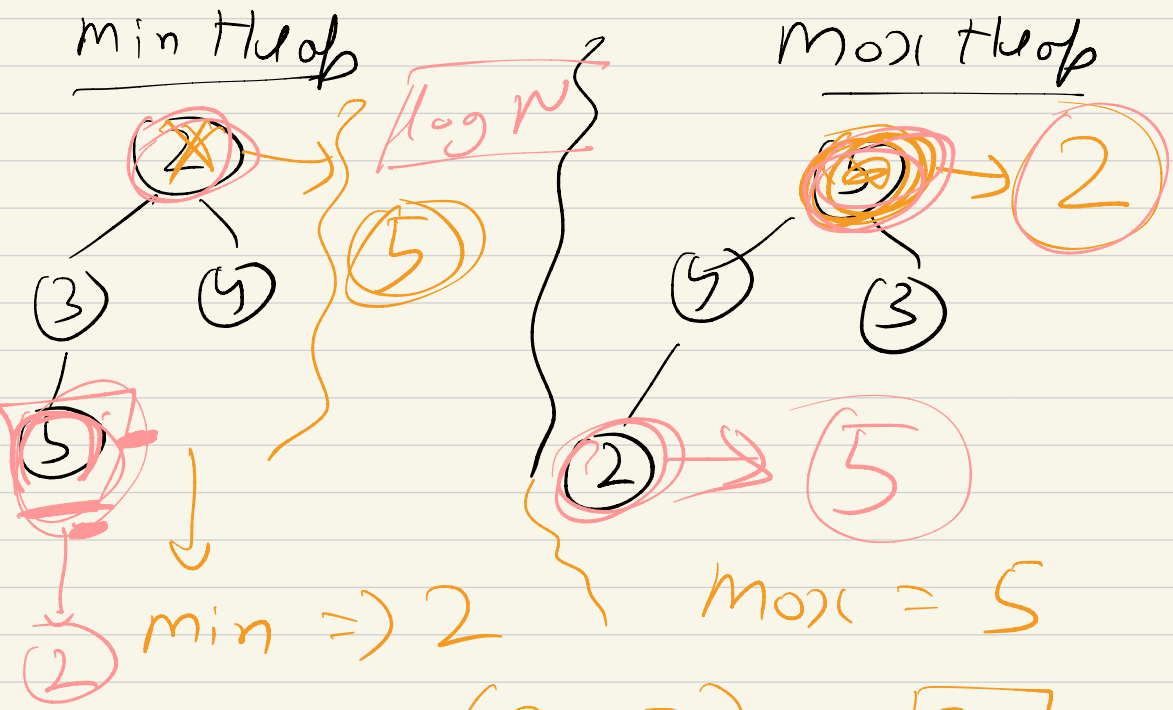
↳ Find Max and Replace it with something

{ Max Heap ↙

() Find Min

() min Heap

{3, 4, 5, 2}



min => 2

max = 5

Avg => $\frac{2+5}{2}$ => 3

~~min~~ => 2 => 2+3 = 5

~~max~~ => 5 => 5 - 3 => 2

it should be done?

Yes! OR No!

2 3, 4, 5

it could be done?

Yes OR No?

Heap X

Heap supports ->
top

{1, 5, 7, 8}

5, 5, 2
7, 8, 10

2, 7, 5
5, 10, 5

AVG => (1 + 8) / 2 => 4

0 -> 4 + 1 => 5

8 -> 8 - 4 => 4

map
8 -> 2

Don't use it

min

Max

Ignored

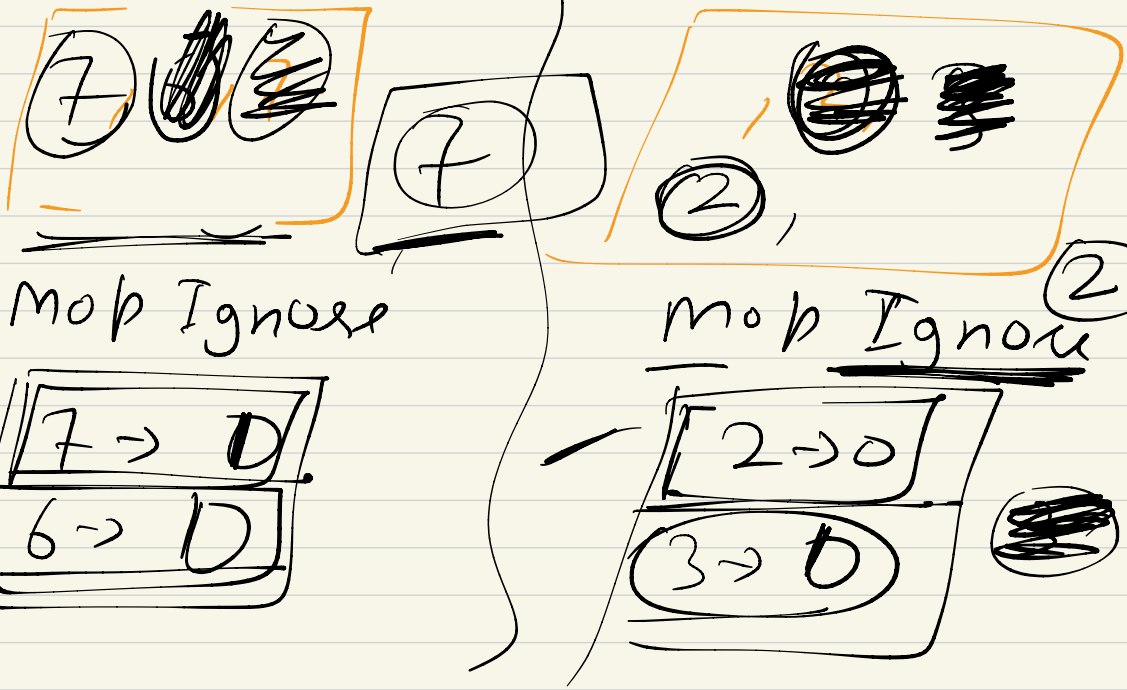
map
1 -> 1

A = [3, 2, 7]

3 2 7

Min Heap

Max Heap



Map Ignore

7 → 0

6 → 0

Map Ignore

2 → 0

3 → 0

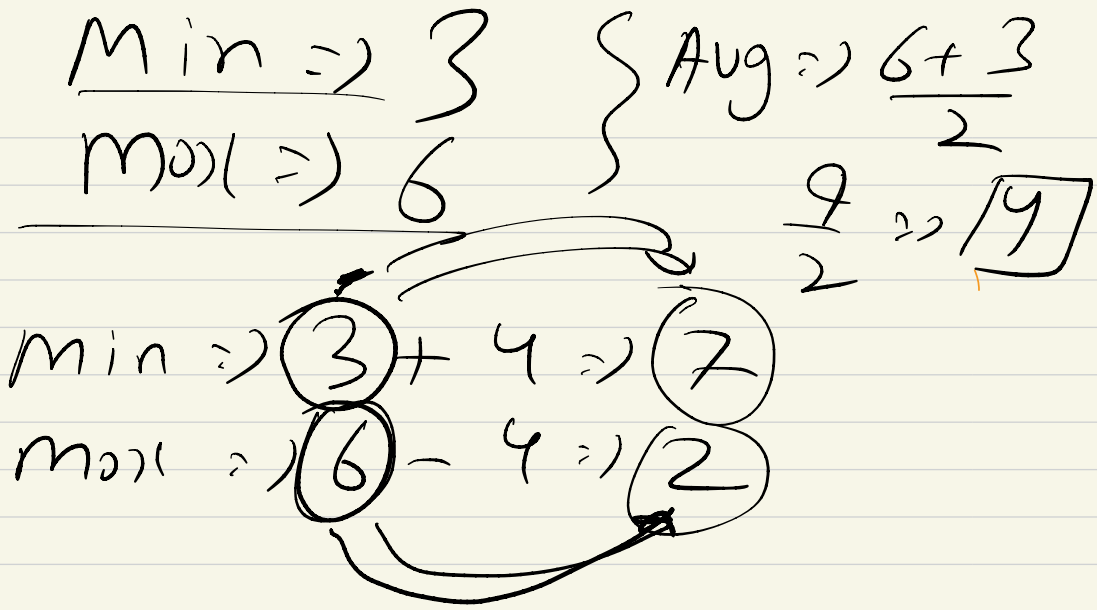
Min = 2

Max = 7

Avg = $\frac{2+7}{2}$ = 4

2 → 2 + 4 → 6

7 → 7 − 4 → 3


Min => 3
Max => 6
Avg => 6+3/2
9/2 => 4
Min => 3 + 4 => 7
Max => 6 - 4 => 2