ARRAYS - 2

→ Merge Intervals (next class)

2D ARRAYS

↳ 2D Prefix Sums
↳ Submatrices

[↳ Rainwater Problem
↳ Kadane's Algo

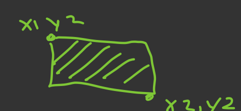
Q. Given a matrix of size N*M, and we are Q queries, find the sum of a given sub matrix. Each query will 4 integers x1,y1, x2, y2 denoting the sub matrix.

Brute Force

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

Q = 3

x1, y1, x2, y2 = (1, 0, 2, 2)



5 + 6 + 7
+ 9 + 10 + 11
= 48

1 query →

```
sum = 0
for(i → x1 to x2)
    for(j → y1 to y2) {
        sum += a[i][j];
    }
```

Time / Space - O(1)

→ O(NM) for 1 query
→ O(Q.NM) for Q Queries

Optimise

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

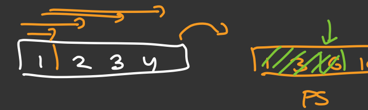
Input Matrix

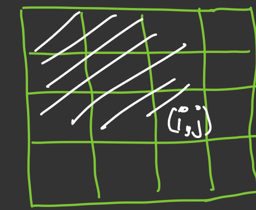
M−1

1	3	6	10
6	14 (1,1)	24	36
15	33	54	78 (N−1, M−1)

N−1 →

O(1)





PS

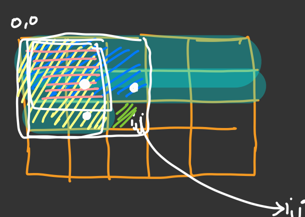
ps(i,j) = hold the sum of submatrix from 0,0 to i,j

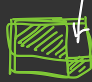
Build a 2D Prefix Sum Matrix

→ Fill first Row & col (just like 1D array)

Precompute O(N.M)

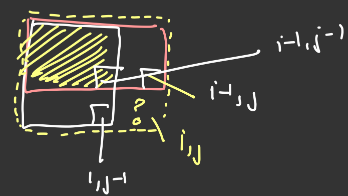
```
for(i=1 ___ N-1) {
    for(j=1 ___ M-1) {
        ps(i,j) =
            = ps[i-1][j] + ps[i][j-1] - ps[i-1][j-1]
              + arr[i][j]
    }
}
```

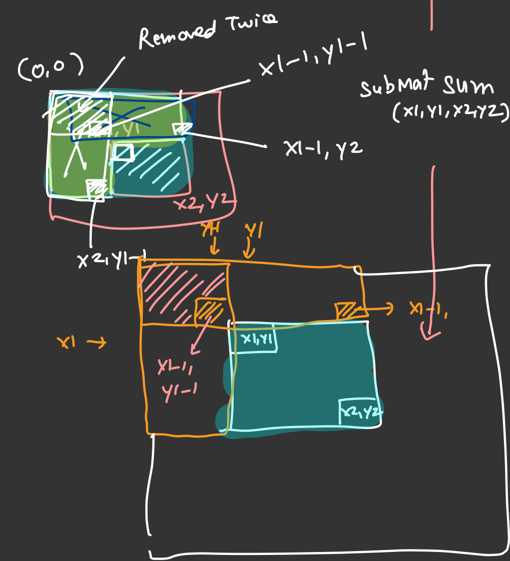




$ps(i,j)$ = (blue) + (yellow) − (red) + (green)

$= ps[i-1][j] + ps[i][j-1] - ps[i-1][j-1] + arr[i][j]$





② Solve for Queries

Q → input()

while (Q > 0) {

x1, y1, x2, y2 = input

$$\text{submat sum}(x1, y1, x2, y2) = \left.\begin{array}{l} ps[x2][y2] \\ -\ ps[x2][y1-1] \\ -\ ps[x1-1][y2] \\ +\ ps[x1-1][y1-1] \end{array}\right\} O(1)$$

}

Total time for query = O(Q)

Time time = O(NM + Q)

Space = O(NM)

Mango Trees

Ramu's father has left a farm organized as an N × N grid. Each square in the grid either has or does not have a mango tree. He has to divide the farm with his three sisters as follows: he will draw one horizontal line and one vertical line to divide the field into four rectangles. His sisters will choose three of the four fields and he gets the last one.

He wants to divide the field so that he gets the maximum number of mangos possible, assuming that his sisters will pick the best three rectangles.

For example, suppose the field looks as follows:

Q2 Q1

```
.##...
#..##.
.#....
.##..#
#..##.
.#....
```

Q3 Q4

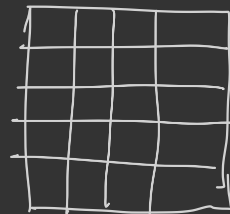
Not the Best cut

Q1 → 1 → Ramu

Q2 → 5
Q3 → 5
Q4 → 2
→ Sisters

Decide how to cut the field to maximise his mangoes

Goal → maximise the min mangoes by changing cut



Ramu can ensure that he gets at least 3 mango trees by cutting as follows:

Best cut?

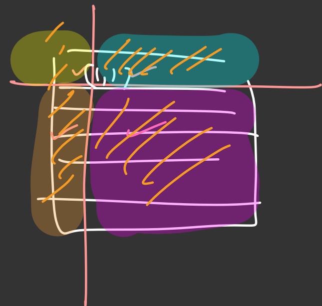
```
.#|#...
#.|.##.
.#|....
------+---------
.#|#..#
#.|.##.
.#|....
```

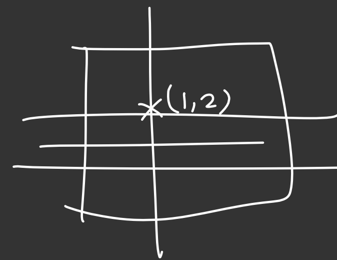
Q1 – 3 — Ramu

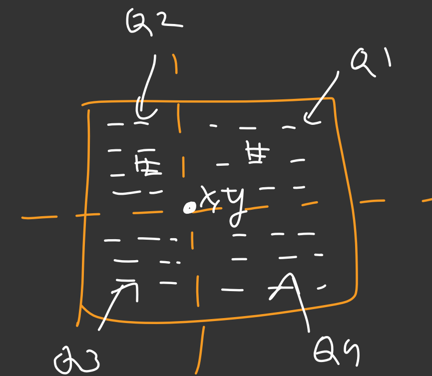
Q2 – 3
Q3 – 3
Q4 – 4

Sisters

Soln Try cutting all locations (x,y) & find out Best cut







Algo-1

N^2 cuts $\times$ N^2

$= O(N^4)$

ans = 0

for(x = 0 —— N-1 } N^2 cuts

for(y = 0 —— N-1] {

O(1) after optimis

Q1 → ?

Q2 → ?

Q3 → ?

Q4 → ?

→ Ramu = min (Q1, Q2, Q3, Q4),

→ ans = max(ans, Ramu),

}

option-1 ① iterate everytime over submatrix N^2 cells

OR

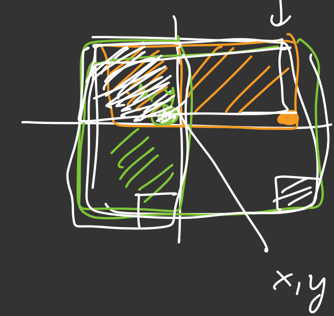
option-2 ② use a PS matrix

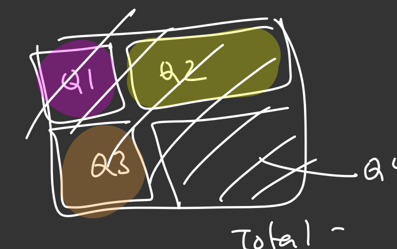
3

print(ans)

Algo-2

$$
\begin{cases}
Q1 = PS[x,y] \\
Q2 = PS[x][N-1] - Q1 \\
Q3 = PS[N-1][y] - Q1 \\
Q4 = \underbrace{PS[N-1][N-1]}_{\text{Total}} - Q1 - Q2 - Q3
\end{cases}
$$

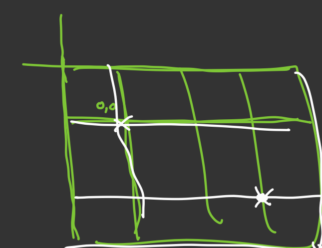




1. Build PS Matrix $O(N^2)$
2. Make N^2 cuts & find Best cut $O(N^2)$

overall Time = $O(N^2)$

Space = $O(N^2)$



Q3. Given a Matrix of Size N * M, calculate the sum of all sub matrices., output single int

$$\begin{bmatrix} 3 & 1 \\ -1 & -2 \\ 2 & 4 \end{bmatrix} \qquad \begin{bmatrix} 3 & 1 \\ -1 & -2 \\ 2 & 4 \end{bmatrix} = [3], \begin{bmatrix} 3 & 1 \end{bmatrix}, \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \begin{bmatrix} 3 \\ -1 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 & 1 \\ -1 & -2 \end{bmatrix}, \begin{bmatrix} 3 & 1 \\ -1 & -2 \\ 2 & 4 \end{bmatrix},$$

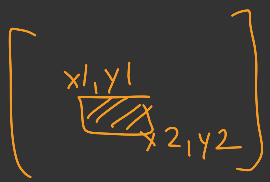
$$[1], \begin{bmatrix} -1 \\ -2 \end{bmatrix}, \begin{bmatrix} 1 \\ -2 \\ 4 \end{bmatrix} + + +$$

$$[-1] \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \begin{bmatrix} -1 & -2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} -2 \\ 4 \end{bmatrix}, [-2], [2], [4]$$

$(i+1)(j+1)(n-i)(m-j)$

$= 1 \cdot 1 \cdot (3-0)(2-0)$

$= 6$



N x M

Submatrices

```
for(x1 ——)          N
  for (x2 ——)       N
    for (y1 ——)     N
      for (y2 —— ){ N
        // compute the sum
```

3

$O(N^4)$ submatrices.

Brute Force → $O(N^6)$ time $\quad N^4 \times N^2$

PS $O(1)$ $\quad$ Brute $O(N^2)$

Brute Force generate all sub + PS
find sum
$O(1)$

$O(N^4 \times 1)$

$= O(N^4)$

how many submatrices will inc given $a[i][j]$

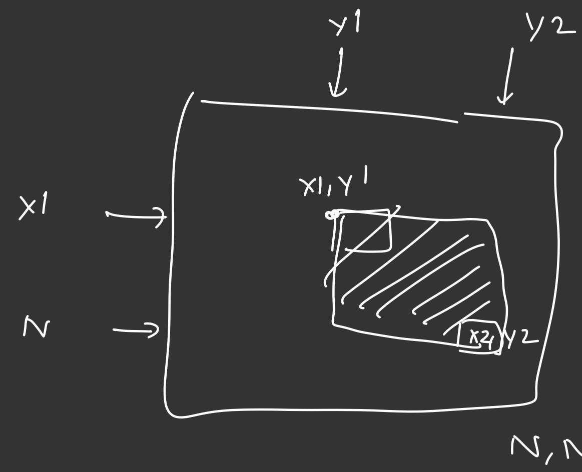
Anand's hint

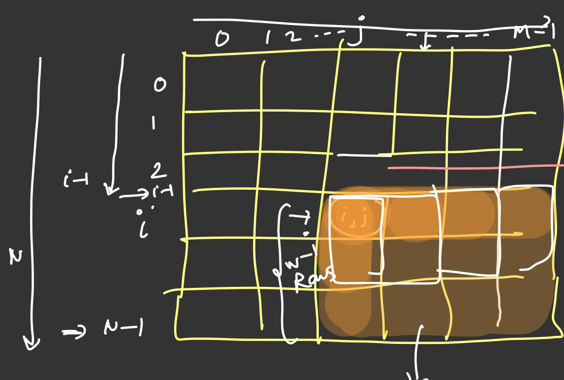
→ we can find out which element is being repeated how many times and then traverse matrix and multiple it with and then sum .. ?

10·30

$N \cdot N \cdot N \cdot N$

$= \underline{O(N^4)}$ ways.



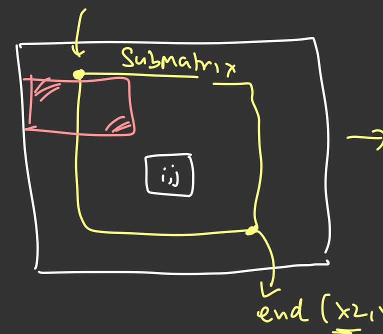


→ Total contribution in final sum

= No of submatrices in which a(i)(j) is present $\times$ $a[i][j]$ = $(N-i)(M-j)$

No of submatrices here = $(N-i)(M-j)$
Rows cols

$N-1-(i-1)$
$= N-\cancel{i}-1+\cancel{i}$
$= N-1$



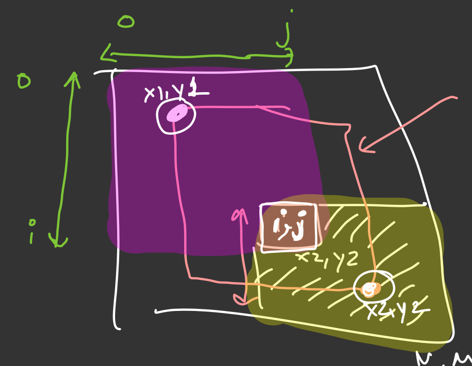
Goal

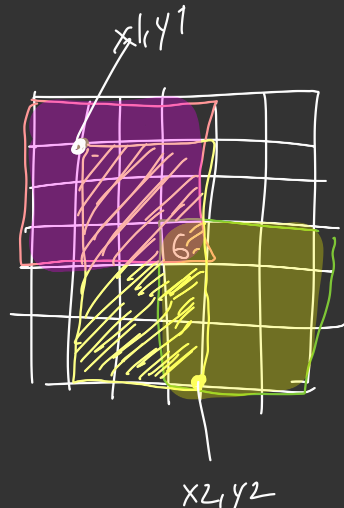
Count the no of valid ways to choose $x1, y1, x2, y2$ s.t. $a[i][j]$ is included in submatrix

Total ways

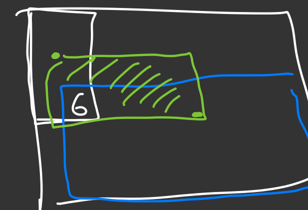
ways (x1, y1. x2. y2.)

$$\underbrace{(i+1)\,(j+1)\,(N-i)\,(M-j)}_{\text{No of times } a[i][j] \text{ will contribute in final ans.}}$$





ways # x1, ways # x2, # y1 # y2



N^6

$\downarrow$

N^4

$\downarrow$

N^2

Code

sum = 0

```
for (i=0 ; i< N ; i++ ){
    for (j=0 , j< M, j++){
        sum = sum + a[i][j] * (i+i)(j+1)(N-i)(M-j)
    }
}
```

$$\underbrace{a[i][j]}_{x} * \underbrace{(i+1)(j+1)(N-i)(M-j)}$$

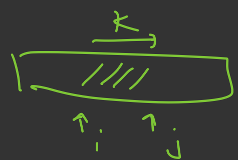
$= O(N \cdot M)$

1D Arrays

(1) Maximum Subarray Sum / Kadane's Algo $O(N)$

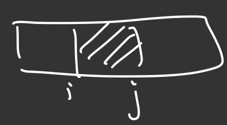
3	5	-10	2	6	4	-1	5	-3	7	-12	4

(1) Brute Force



$O(N^2 \cdot N) = O(N^3)$

(2) PS Sum



sum

$PS[j] - PS[i-1]$

$O(N^2 \times 1)$
$= O(N^2)$

(3) Kadane's Algo

-ve +ve

3	5	-10	2	6	4	-1	5	-3	7	-12	4

CS = 0 + 3 8 ~~-2~~ 0 2 8 12 11 16 13 20 8 12

MS = 0 3 8 8 8 8 12 12 16 16 20 20 20

CS = 0, MS = 0

```
for (i=0, i<n, i++) {

    CS = CS + arr[i],
    if (CS<0) { CS = 0, }
    MS = Max (CS, MS);
}
print (MS)
```

→ O(N)

→ O(1)

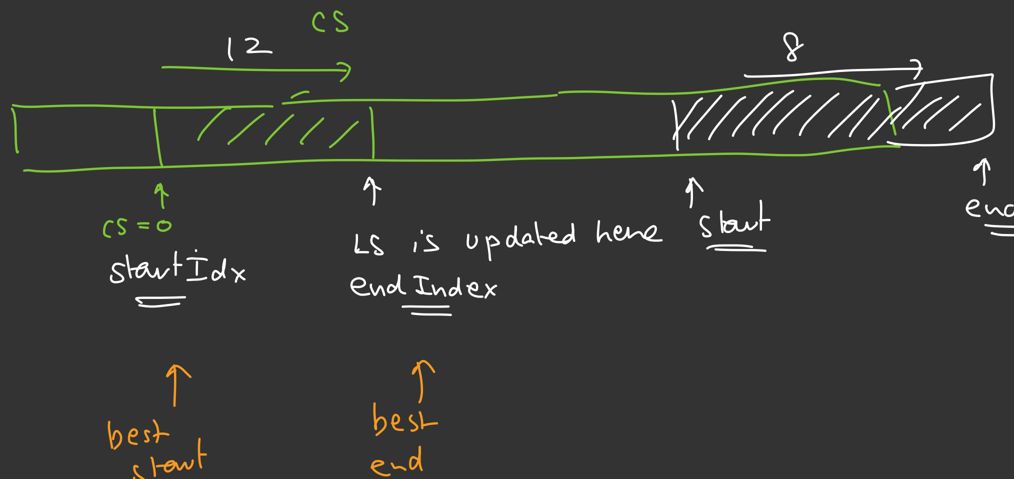
start

end,

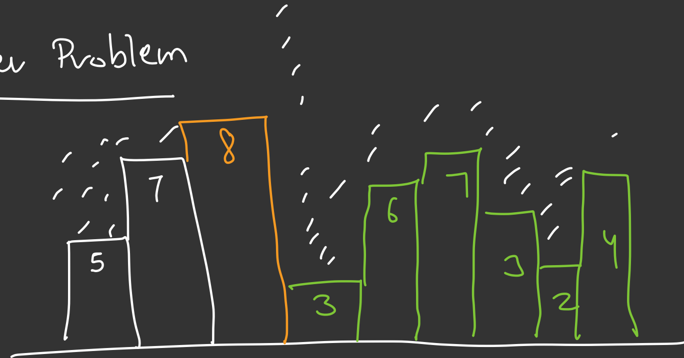
best start, best end.

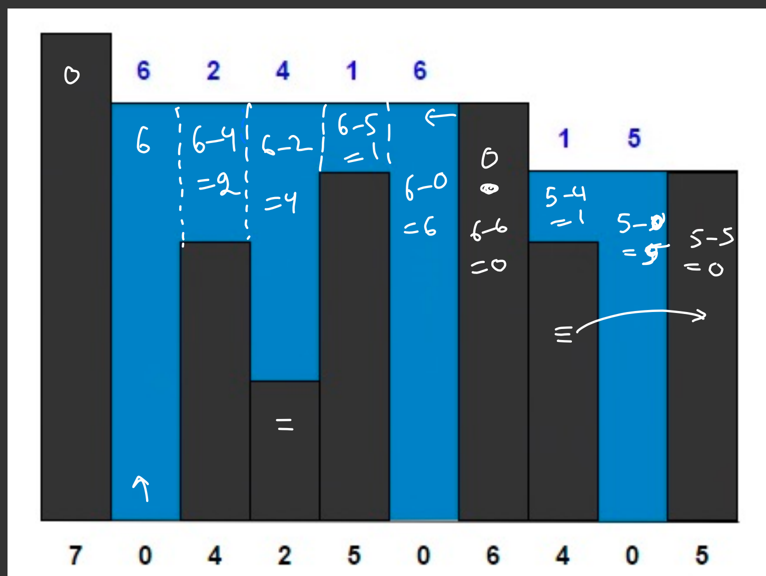
-8, -1, -3, -5, -4, -2

largest



Rainwater Problem

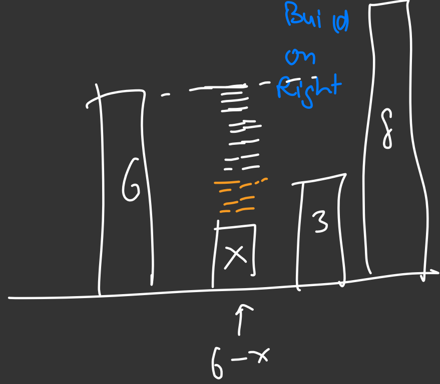




buildings

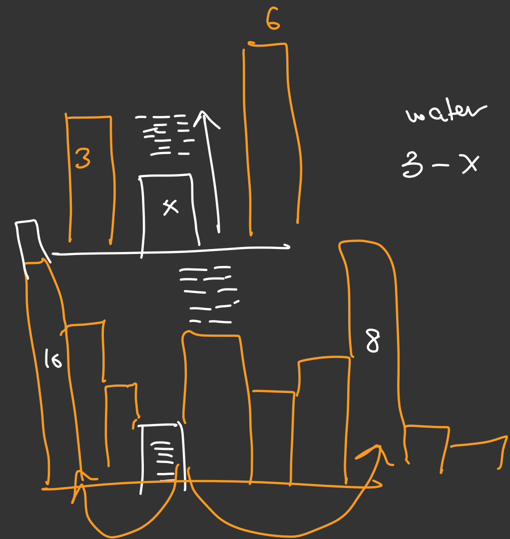
largest

Calc the total water buildings will hold.



→ Largest on left, Right

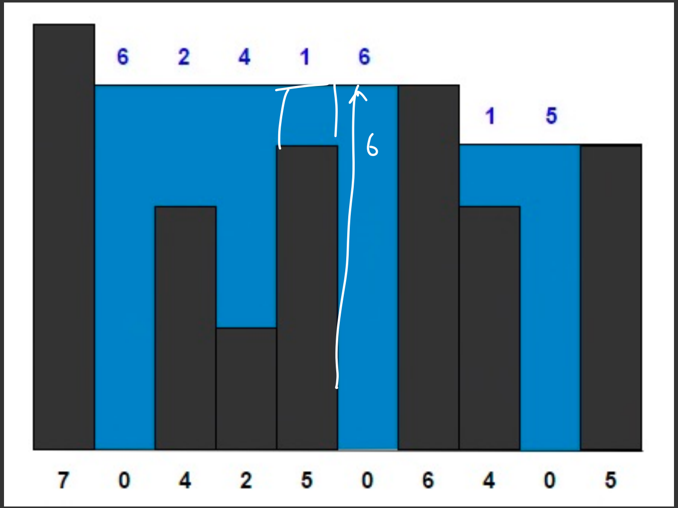
Smaller one decide the water ht



water

3 − x

water vol = waterlevel − building ht



loop

0 N-1

left = 7

Right = 6

Water Level = Min(7,6) = 6

water = 6 - BuildHt

= 6 - 5

= 1

ans = 0

for(i=0 —— N-1){

L = ✓ } loop

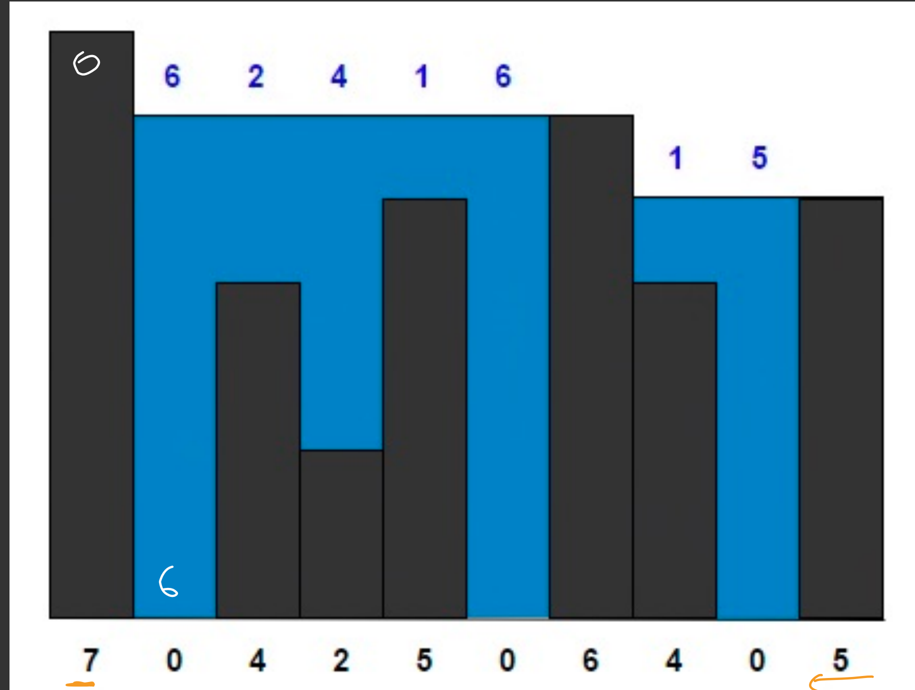
R = ✓

ans = ans + min(L,R) - a(i);

}

$O(N^2)$

Interview



→ 25

Left Largest	7	7	7	7	7	7	7	7	7	7	→ O(N)
Right Largest	7	6	6	6	6	6	6	5	5	5	→ O(N)
Min	7	6	6	6	6	6	6	5	5	5	→ O(N)
B	7	0	4	2	5	0	6	4	0	5	→ O(N)

water = 0 + 6 + 2 + 4 + 1 + 6 + 0 + 1 + 5 + 0 → O(N) = 25

① Build left[ ] , Right[ ] using CF.

O(N) Space

O(N) time

```
for( i=0 ——— n-1 ){
    ans = ans + min( left[i], right[i] ) - arr(i);
}
```

Happy weekend :)