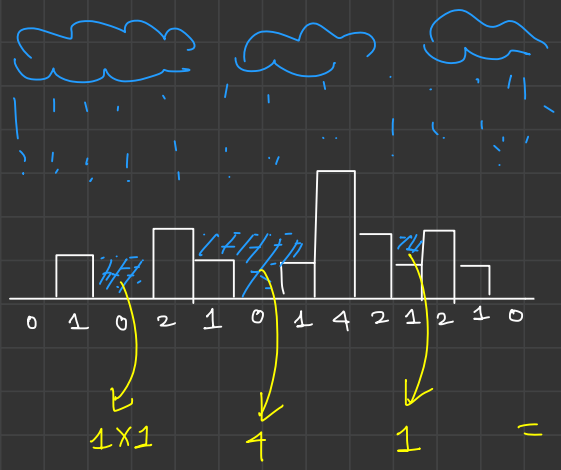
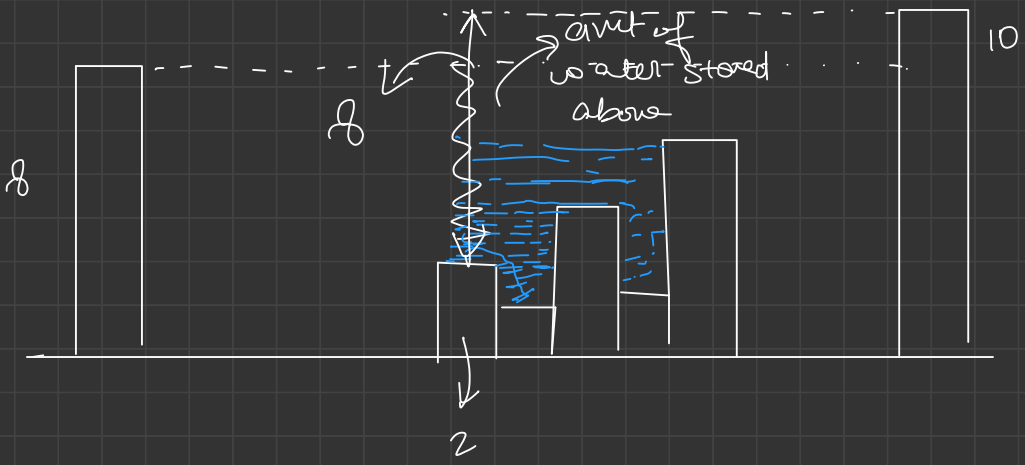
Trapping Rain Water

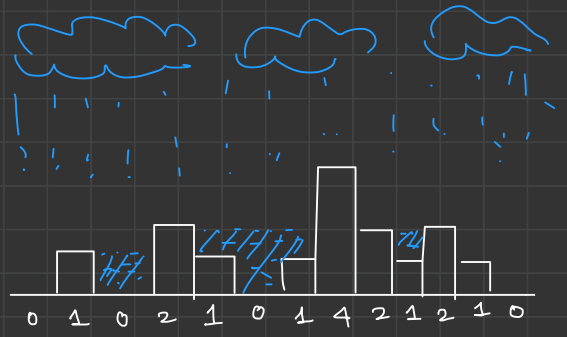


= 6 unit of water



height of water above a building

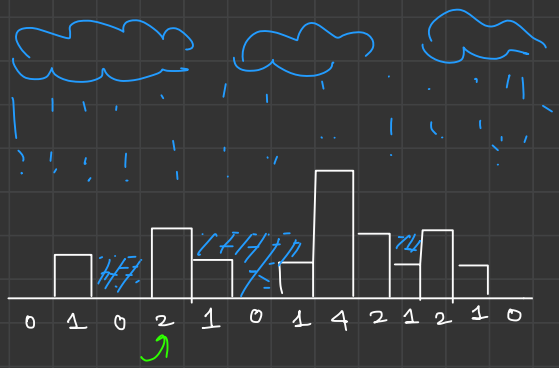
depends on min(Max height on left, Max height on Right)



TC : $O(N^2)$
SC : $O(1)$

Brute force

- → go to each Building
- → scan right array to find tallest Building (RB)
- → scan left array to find tallest Building (LB)
- → calc. th of water = min(LB, RB) − th of Building
- → Cumulate total water



h =

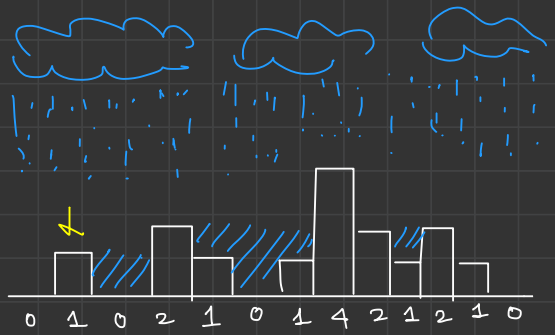
AB =

LB =

th =

two =

total = 1



lmax 0 0 1 1 2 2 2 2 4 4 4 4 4

rmax 4 4 4 4 4 4 4 2 2 2 1 0 0

waters

O(N) ✓

```
int[] lmax = new int[n];
lmax[0] = 0;
for(i -> 1 -> n)
    lmax[i] = max(lmax[i-1], arr[i-1]);
```

TC: O(N)

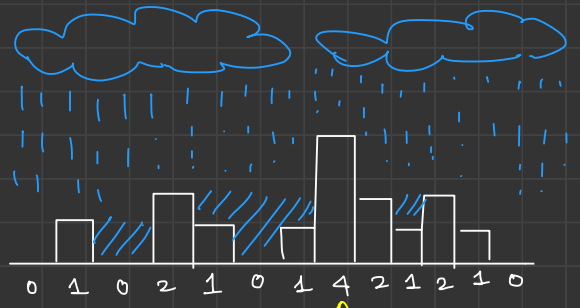
```
int[] rmax = new int[n];
rmax[n-1] = 0;
for(i -> n-2 -> 0)
    rmax[i] = max(arr[i+1], rmax[i+1]);
```

while (l <= r)

{ TC : O(N) }
{ SC : O(1) }

water = 6

LB > RB



else

{
if(arr[r] < arr[RB])
{
int w = arr[RB] - arr[r];
ans += w;
} else {
RB = arr[r];
}
r--;
}

max building on left of l.

max building on right of r

if (LB <= RB)
{
if (arr[l] < arr[LB])
{
int w = arr[LB] - arr[l];
ans += w;
} else {
LB = arr[l];
l++;
}
}

Min Stack

- push()
- pop()
- peek()
- size()
- getMin()

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

getMin() → get you the minimum Element present in the Stack

TC: O(1)

Brute force

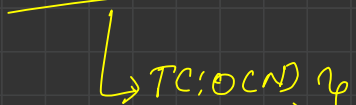
push(10)

push(20)

push(-10)

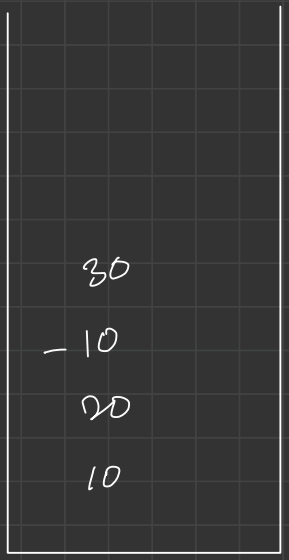
push(30)

getMin()



TC: O(N) ✓

SC: O(N) ✓



Stack

temp

minValue = ~~20~~, ~~30~~ -10

push(10)

getMin()
↳ 10

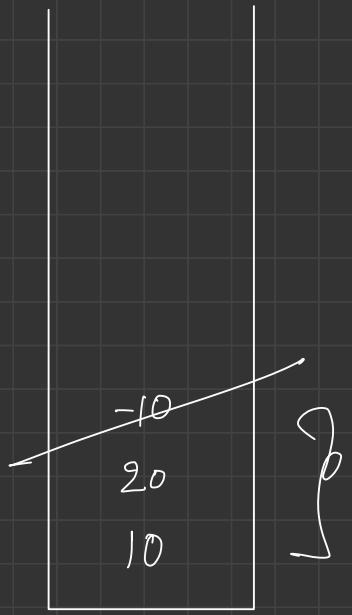
push(20)

push(-10)

getMin()
↳ -10

pop()

getMin()
↳ ~~-10~~
Wrong!



minValue = ~~0~~ ~~10~~ -10

push(10)

getMin()
↳ 10

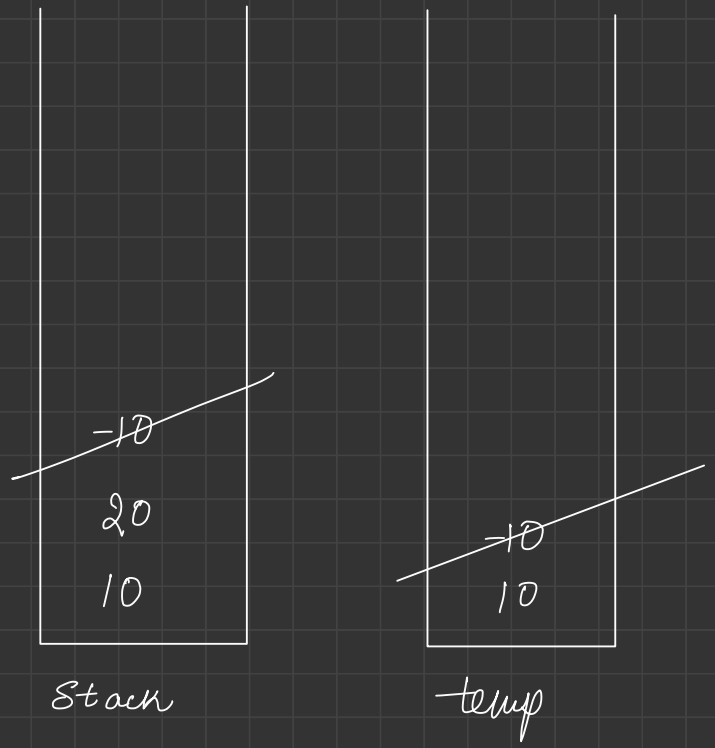
push(20)

getMin()
↳ 10

push(-10)

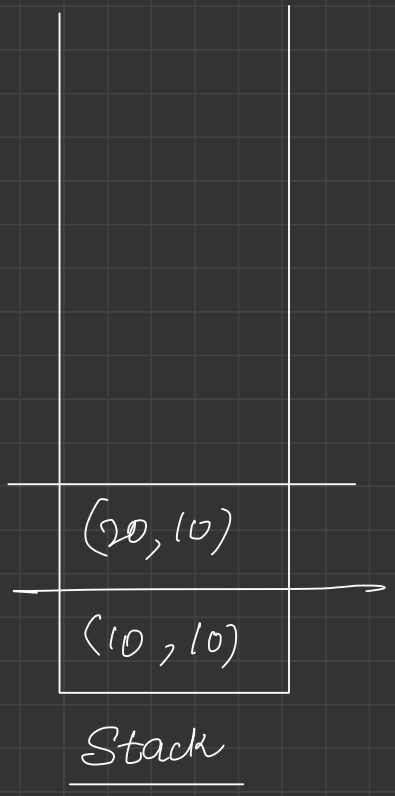
pop()

getMin()
↳ 10



{ TC : O(N)
SC : O(N) }

push(10)
getMin()
↳ 10
push(20)
getMin()
↳ 10
push(-10)
pop()
getMin()



Stack

minVal = ~~∞~~ ~~10~~ -10

```
class Pair
{
  int val;
  int minVal;
};
```

Sum of Subarray Minimums

0	1	2	3	4	5
3	2	4	1	5	2

arr[] = {3, 2, 4, 1, 5, 2}

(3) 3 ✓ (3,2) 2 ✓ (3,2,4) 2 ✓ (3,2,4,1) 1 ✓ (3,2,4,1,5) 1 ✓ (3,2,4,1,5,2) 1 ✓

(2) 2 ✓ (2,4) 2 ✓ (2,4,1) 1 ✓ (2,4,1,5) 1 ✓ (2,4,1,5,2) 1 ✓

(4) 4 ✓ (4,1) 1 ✓ (4,1,5) 1 ✓ (4,1,5,2) 1 ✓

(1) 1 ✓ (1,5) 1 ✓ (1,5,2) 1 ✓

(5) 5 ✓ (5,2) 2 ✓

(2) ↳ 2

Sum = 36

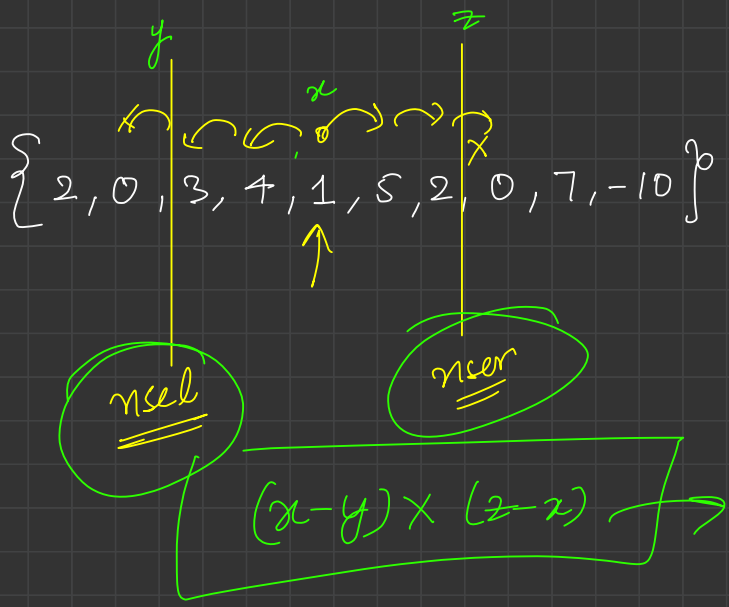
Brute force

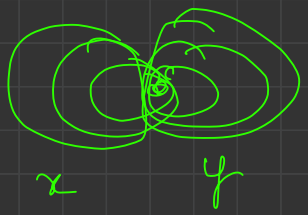
↳ generate all subarrays, and get min in it }

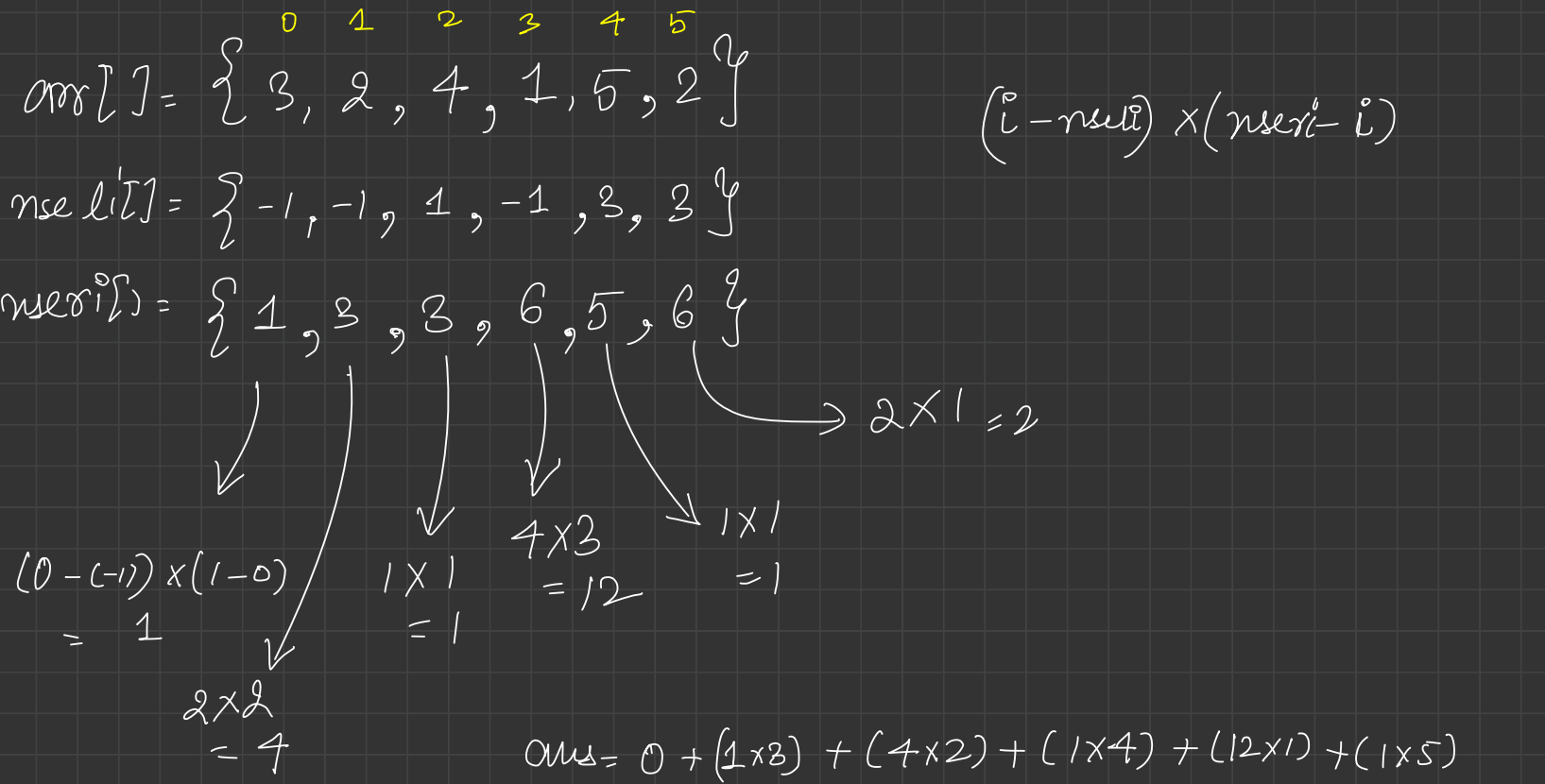
↳ add all minimums

TC: $O(N^2)$
SC: $O(1)$

arr[] = { 3, 2, 4, 1, 5, 2 }
0 1 2 3 4 5






0 1 2 3 4 5
arr[] = {3, 2, 4, 1, 5, 2}
nseli[] = {-1, -1, 1, -1, 3, 3}
nseri[] = {1, 3, 3, 6, 5, 6}
(i - nseli) x (nseri - i)
2 x 1 = 2
(0 - (-1)) x (1 - 0) = 1
1 x 1 = 1
4 x 3 = 12
1 x 1 = 1
2 x 2 = 4
ans = 0 + (1x3) + (4x2) + (1x4) + (12x1) + (1x5) + (2x2)
= 3 + 8 + 4 + 12 + 5 + 4 = 36