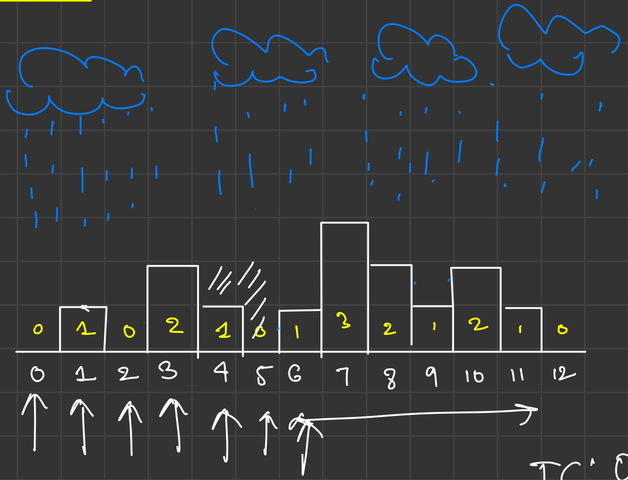
Trapping Rain Water



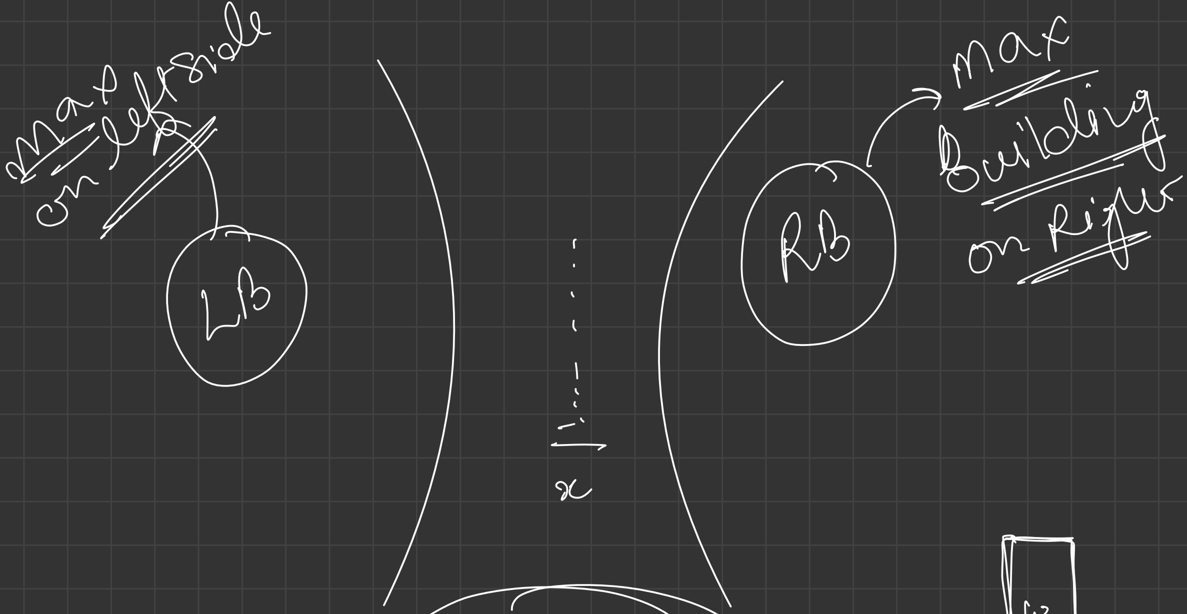
Brute force

RB	3	3	3	3	3	3	3
LB	-1	-1	1	-1	2	2	2
h			1		2	2	2
h'			1		1	2	1

water Stored = 6 units

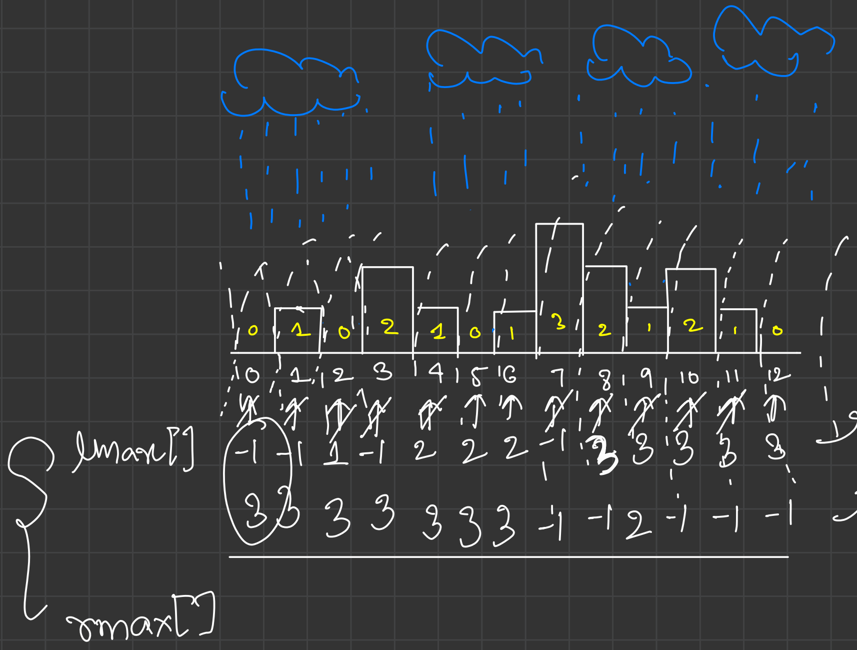
TC: $O(N^2)$

SC: $O(1)$



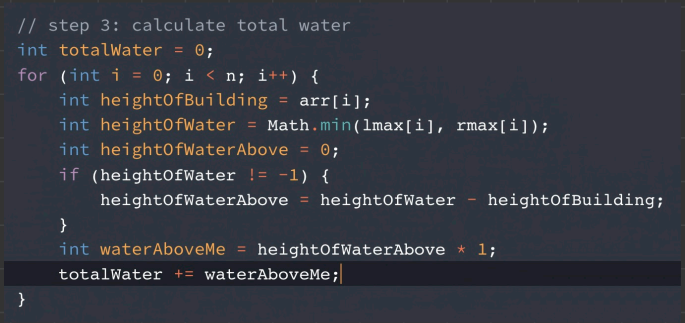
$$\boxed{h = \min(LB, RB)}$$

h above me = $\underline{h - x}$



O(N)

O(N)



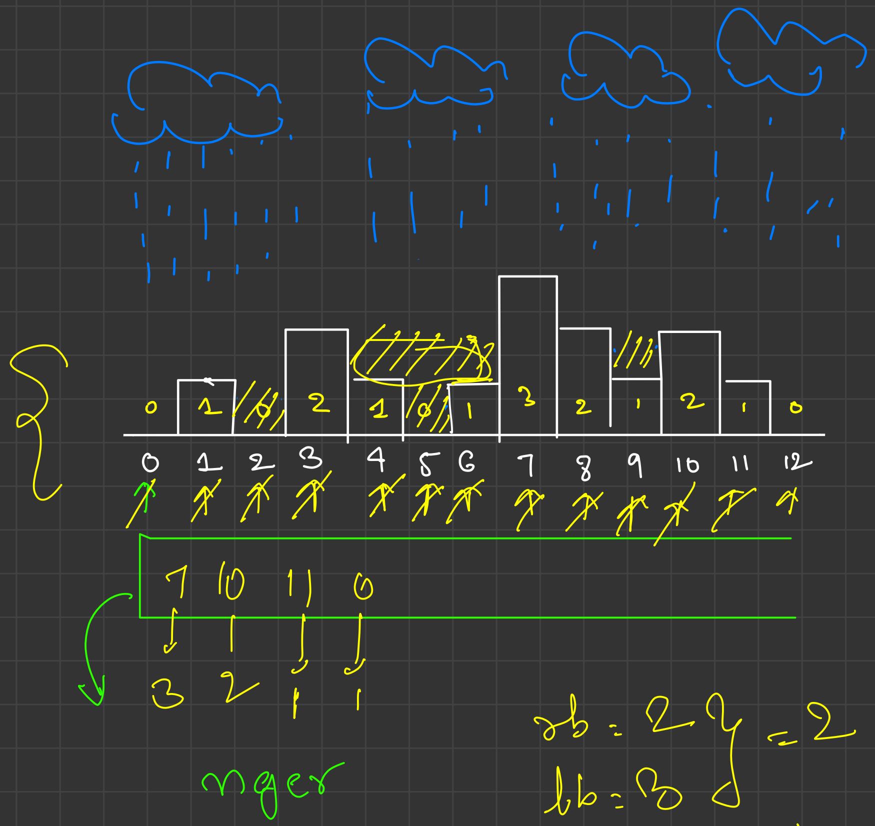
```
// step 3: calculate total water
int totalWater = 0;
for (int i = 0; i < n; i++) {
    int heightOfBuilding = arr[i];
    int heightOfWater = Math.min(lmax[i], rmax[i]);
    int heightOfWaterAbove = 0;
    if (heightOfWater != -1) {
        heightOfWaterAbove = heightOfWater - heightOfBuilding;
    }
    int waterAboveMe = heightOfWaterAbove * 1;
    totalWater += waterAboveMe;
}
```

3

RMax = ~~-1~~ ~~0~~ ~~1~~ ~~2~~

O(N)

TC : O(N)   SC : O(N)



nger $\boxed{7} =$

$\boxed{7} - \boxed{3} - \boxed{1} = 3$

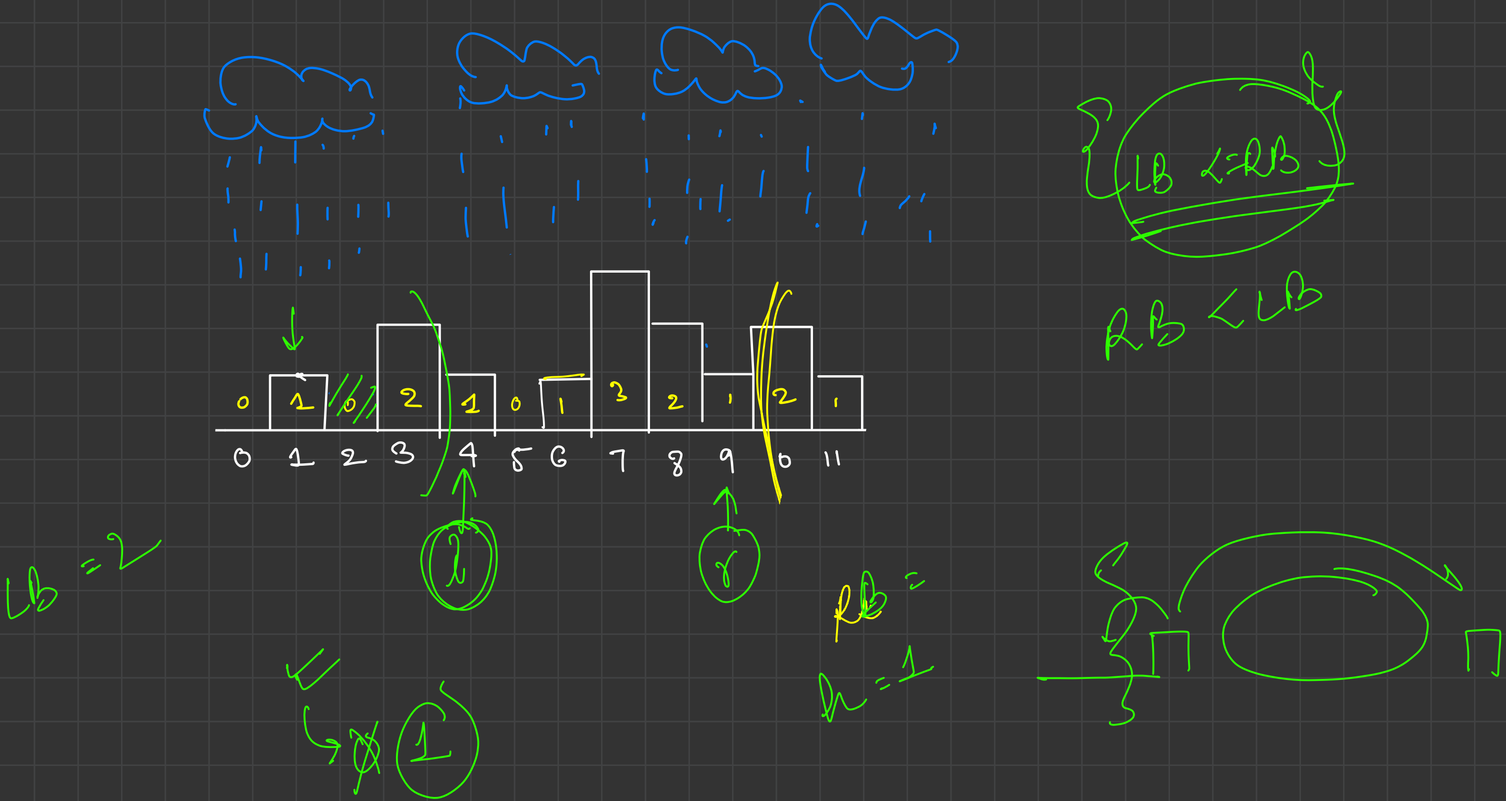
~~$1 \times 3 = 3$~~ $\boxed{3}$

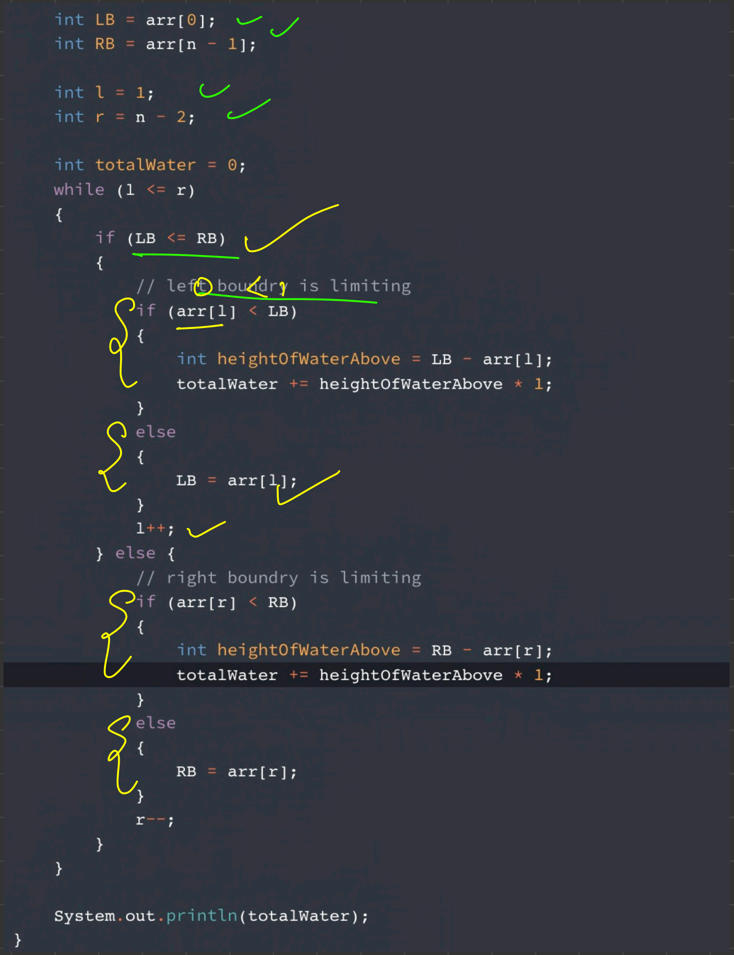
$$\left.\begin{array}{l} rb = 2 \\ lb = 3 \end{array}\right\} = 2$$

$10 - 8 - 1 = 1$

$(2-1) * 1 = \boxed{1}$

TC : O(SC : O(1)



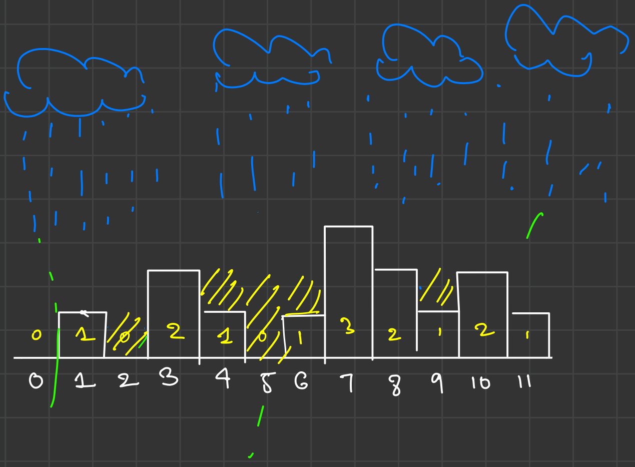


```
    int LB = arr[0];
    int RB = arr[n - 1];

    int l = 1;
    int r = n - 2;

    int totalWater = 0;
    while (l <= r)
    {
        if (LB <= RB)
        {
            // left boundry is limiting
            if (arr[l] < LB)
            {
                int heightOfWaterAbove = LB - arr[l];
                totalWater += heightOfWaterAbove * 1;
            }
            else
            {
                LB = arr[l];
            }
            l++;
        } else {
            // right boundry is limiting
            if (arr[r] < RB)
            {
                int heightOfWaterAbove = RB - arr[r];
                totalWater += heightOfWaterAbove * 1;
            }
            else
            {
                RB = arr[r];
            }
            r--;
        }
    }

    System.out.println(totalWater);
}
```



LB = ~~1~~ 0 ~~1~~ ~~2~~ 3

RB = ~~1~~ 2

Sum of Subarray Minimums

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

TC : O(N²)

SC : O(1)

Subarrays :

(3) → 3 (2) → 2 (4) → 4 (1) → 1 (5) → 5 (2) → 2

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

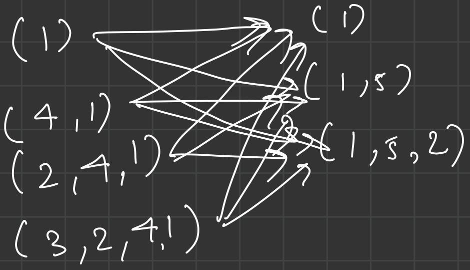
(2,4) → 2
(2,4,1) → 1
(2,4,1,5) → 1
(2,4,1,5,2) → 1

(4,1) → 1
(4,1,5) → 1
(4,1,5,2) → 1

(1,5) → 1
(1,5,2) → 1

(5,2) → 2

$$\begin{array}{lcccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ arr[\,] : & \{3, & 2, & 4, & 1, & 5, & 2\} \\ nseli[\,] : & \{-1, & -1, & 1, & -1, & 3, & 3\} \\ nseri[\,] : & \{1, & 3, & 3, & 6, & 5, & 6\} \end{array}$$



$$\underline{\underline{(idx - nseli) \times (nseri - idx)}}$$

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

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