19. Trapping Rain Water

Medium Accuracy: 49.62% Submissions: 42261 Points: 4

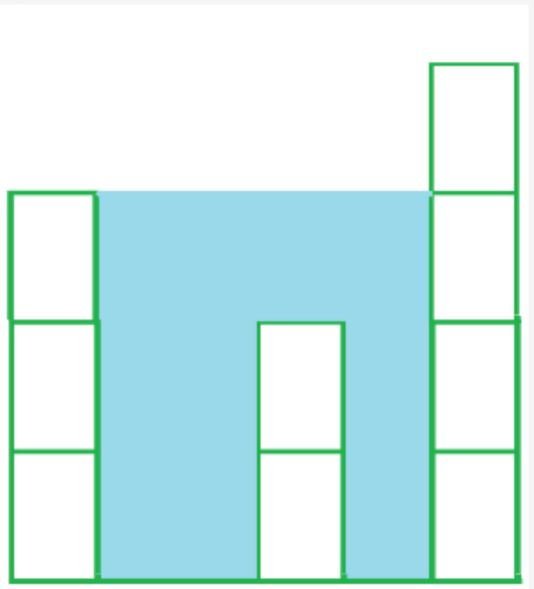
Given an array **arr[]** of **N** non-negative integers representing the height of blocks. If width of each block is 1, compute how much water can be trapped between the blocks during the rainy season.

O(n) T & S

Example 1:

```
Input:
N = 6
arr[] = {3,0,0,2,0,4}
Output:
10
```

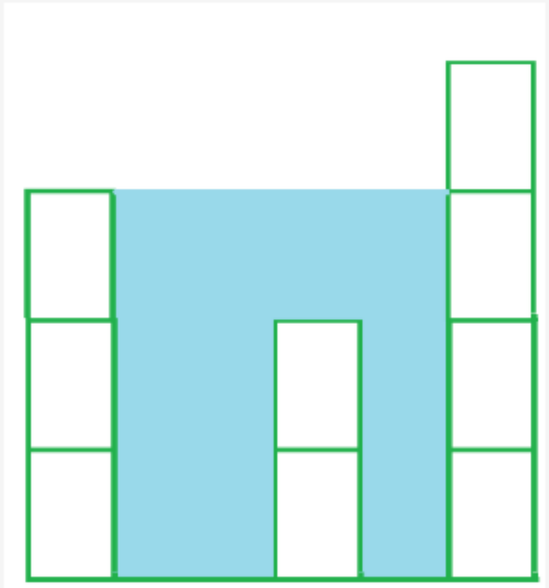
Explanation:



Bars for input {3, 0, 0, 2, 0, 4}
Total trapped water = 3 + 3 + 1 + 3 = 10

Water can't be stored for a grid longer than the tallest left wall

Explanation:



Bars for input {3, 0, 0, 2, 0, 4}
Total trapped water = 3 + 3 + 1 + 3 = 10

for a particular column max water that could be stored would be equal to the minimum of leftmax tower and rightmax.

```
public:
int trappingWater(int arr[], int n){

    // left[i] contains height of tallest bar to the
    // left of bar at ith index including itself.
    vector<int> left(n, 0);

    // right[i] contains height of tallest bar to
    // the right of bar at ith index including itself.
    vector<int> right(n, 0);

    int water = 0;

    // Storing values of tallest bar from first index till ith index.
    left[0] = arr[0];
    for (int i = 1;i < n;i++) {
        left[i] = max(left[i - 1], arr[i]);
    }

    // Storing values of tallest bar from last index till ith index.
    right[n-1] = arr[n-1];
    for (int i = n - 2;i >= 0;i--) {
        right[i] = max(right[i + 1], arr[i]);
    }
```

Part (1) → storing left max & right max.

(1) contains the left max tower which has higher value including ith.

→ 0 → comparing {(i-1) & (i)} max.

(2) → contains right max considering ith itself.

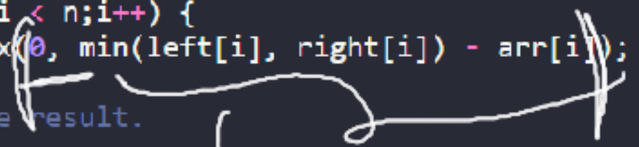
(n-1) → comp. max (i+1, i)

{i : n-2 initialization

Part 2 : Calculating water stored

```
}

// Storing the result by choosing the minimum of heights of tallest bar to
// the right and left of the bar at current index and also subtracting the
// value of current index to get water accumulated at current index.
for (int i = 0;i < n;i++) {
    water += max(0, min(left[i], right[i]) - arr[i]);
}
// returning the result.
return water;
```



min of curr. sides - already filled depth.

if answer negative then 0.