42. Trapping Rain Water

Hard 9191 140 Add to List Share

Given n non-negative integers representing an elevation map where the width of each bar is `1`, compute how much water it can trap after raining.

Example 1:

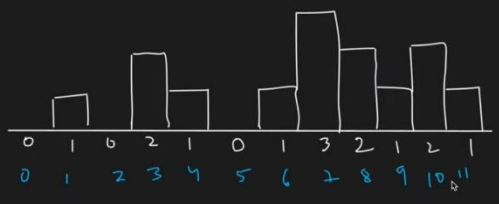

```
Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
Explanation: The above elevation map (black section) is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.
```

Example 2:

```
Input: height = [4,2,0,3,2,5]
Output: 9
```

TUF

TUF



TUF

TUF

$2 - 0 = 2$

TUF

$\min(\text{left}(i), \text{right}(i)) - a[i]$

TUF

i to 0.

min (left(i), right(i)) − a[i]

i to n-1

2 − 2 = 0

TC → O(N²)

SL → O(1)

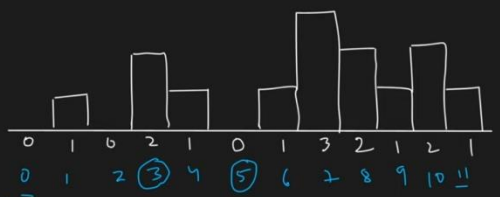
TUF

min (left(i), right(i)) − a[i]

pre 0 1 1 2 2 2 2 3 3 3 3 3

suf 3 3 3 3 3 3 3 2 2 2 1

TUF



TC → O(N) + O(N) + O(N)
≈ O(N)

SL = O(2N)

i

(min (left(i), right(i)) − a[i])

pre 0 1 1 2 2 2 2 3 3 3 3 3 ✓ No

suf 3 3 3 3 3 3 3 2 2 2 1 ✓

TUF

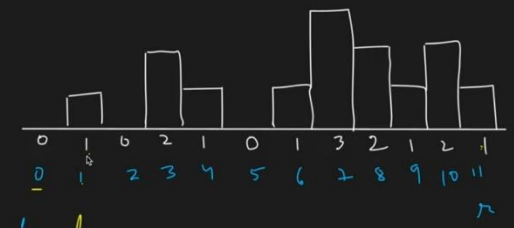
$l = 0$ $res = 0$

$r = n-1$

left_max = 0

right_max = 0

TUF



$l = 0$ $res = 0$

$r = n-1$

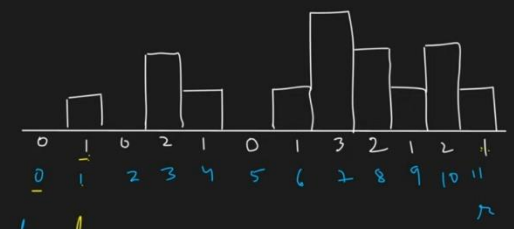
left_max = 0

right_max = 0

if (a[l] <= a[r])

l++;

TUF



$l = 0$ $res = 0$

$r = n-1$

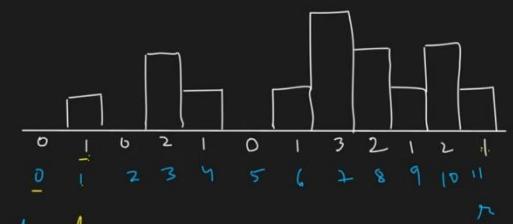
left_max = 1

right_max = 0

if (a[l] <= a[r])

l++;

TUF



$l = 0$ $res = 0$

$r = n-1$

leftmax = ~~0~~ 1

rightmax = 0

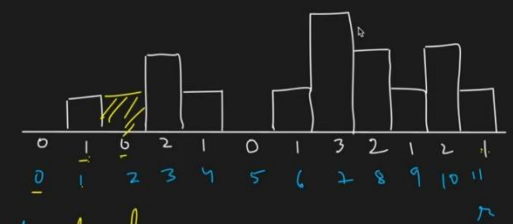
if (a[l] <= a[r])

if (a[l] >= leftmax) leftmax = a[l]

else

l++ ;

TUF



$l = 0$ $res = 0$

$r = n-1$

leftmax = ~~0~~ 1

rightmax = 0

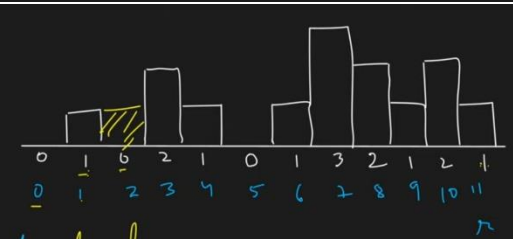
if (a[l] <= a[r])

if (a[l] >= leftmax) leftmax = a[l]

else res += (leftmax - a[l])

l++ ;

TUF



$l = 0$ $res = 0$ 1

$r = n-1$

leftmax = ~~0~~ 1

rightmax = 0

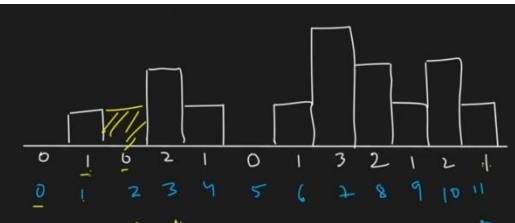
if (a[l] <= a[r])

if (a[l] >= leftmax) leftmax = a[l]

else res += (leftmax - a[l])

l++ ;

TUF

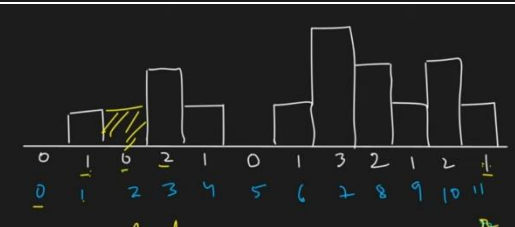


t t t l

l = 0 res = 0 1
r = n-1
leftmax = 1
rightmax = 0

if (a[l] <= a[r])
if (a[i] >= leftmax) left-a[i]
else res += (leftmax - a[i])
l++;

TUF



t t t l

else

if(a[r] >= rightmax)
rightmax = a[r]
else
{

l = 0 res = 0 1
r = n-1
leftmax = 1
rightmax = 0

if (a[l] <= a[r])
if (a[i] >= leftmax) left-a[i]
else res += (leftmax - a[i])
l++;

TUF

t t t t t l

else

if(a[r] >= rightmax)
rightmax = a[r]
else
{ r--

l = 0 res = 0 +2 4
r = n-1
leftmax = 1 +2
rightmax = 0 1

if (a[l] <= a[r])
if (a[i] >= leftmax) left-a[i]
else res += (leftmax - a[i])
l++;

TUF

t t t t t t l l r r

else

if(a[r] >= rightmax)
rightmax = a[r]
else
r--

l = 0 res = 0 1 2 4 5
r = n-1
leftmax = 1 2
rightmax = 1 2

if(a[l] <= a[r])
if(a[i] > = leftmax) leftmax - a[i]
else res += (leftmax - a[i])
l++;

TUF

t t t t t t l l r r

else

if(a[r] >= rightmax)
rightmax = a[r]
else
r--

l = 0 res = 0 1 2 4 5
r = n-1
leftmax = 1 2
rightmax = 1 2

if(a[l] <= a[r])
if(a[i] > = leftmax) leftmax - a[i]
else res += (leftmax - a[i])
l++;

TUF

t t t t t t l l r r r r

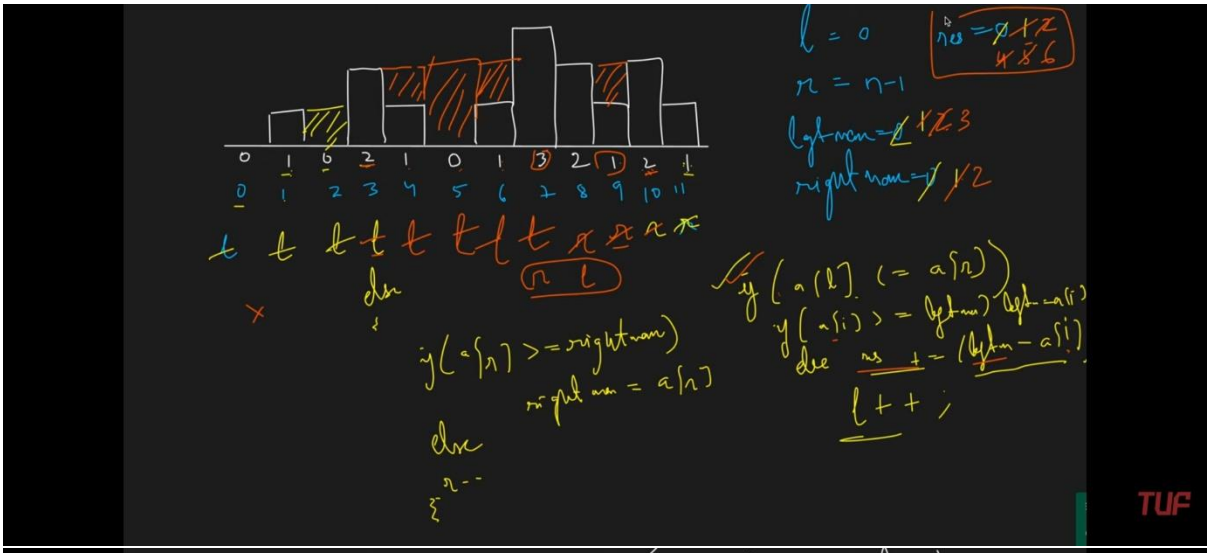
else

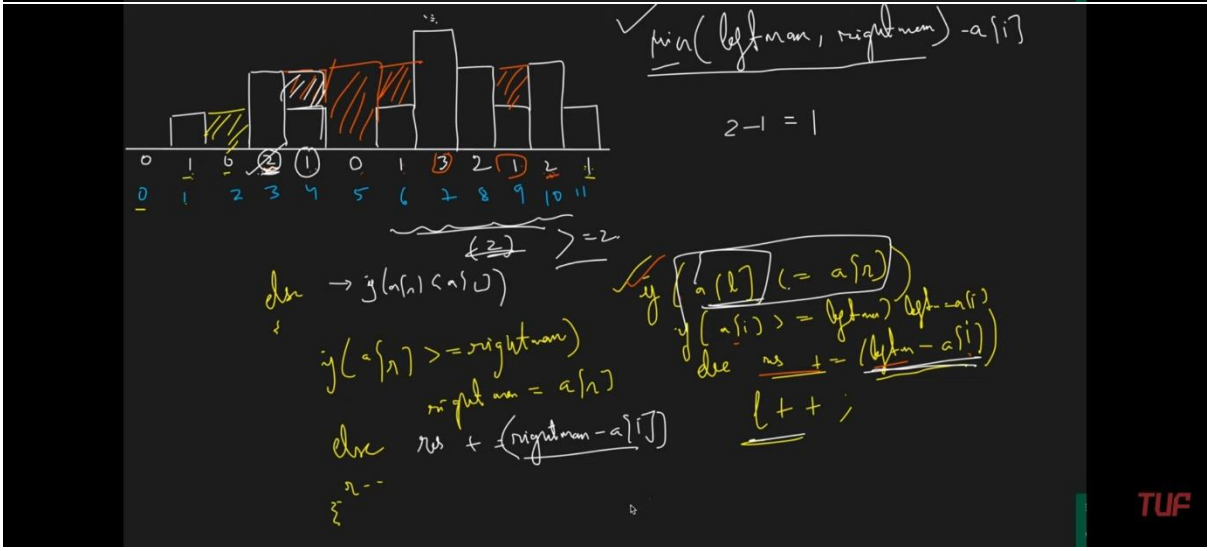
if(a[r] >= rightmax)
rightmax = a[r]
else
r--

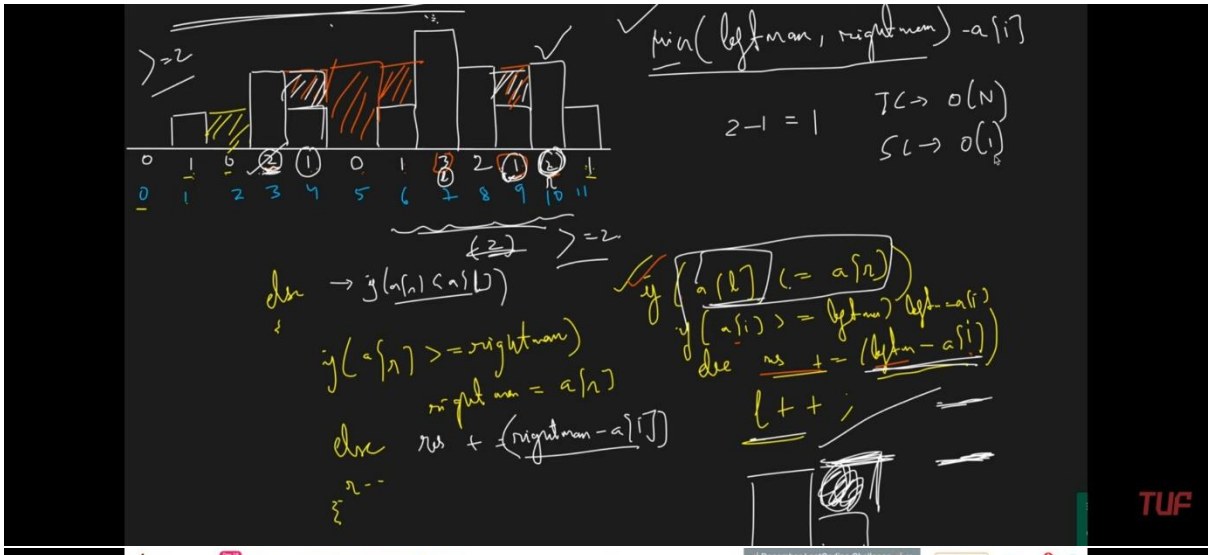
l = 0 res = 0 1 2 4 5 6
r = n-1
leftmax = 1 2 3
rightmax = 1 2

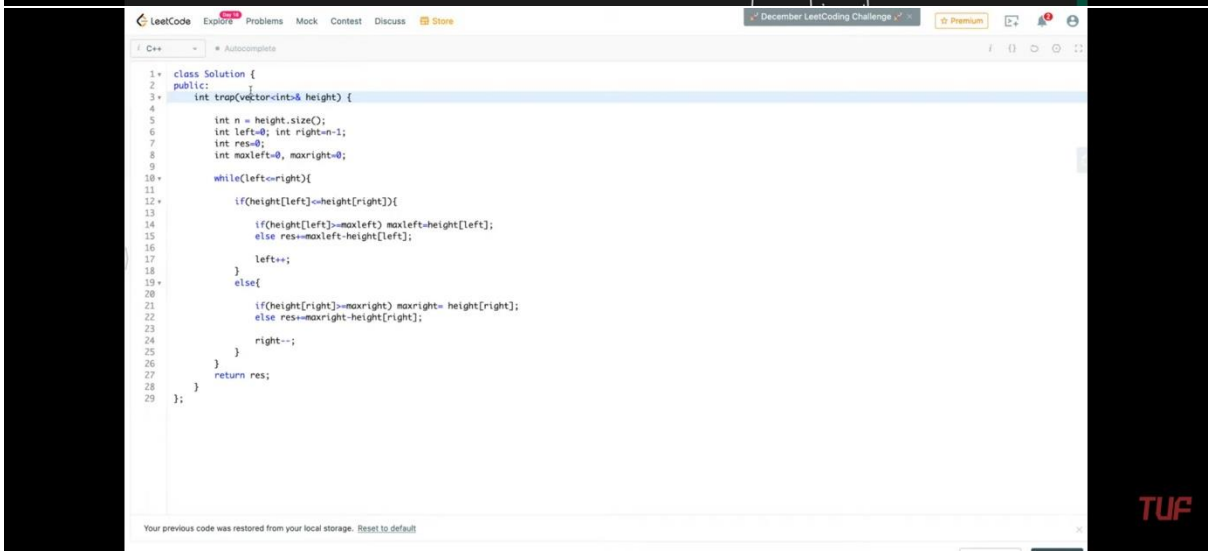
if(a[l] <= a[r])
if(a[i] > = leftmax) leftmax - a[i]
else res += (leftmax - a[i])
l++;

TUF


0 1 0 2 1 0 1 3 2 1 2 1
0 1 2 3 4 5 6 7 8 9 10 11
l = 0
r = n-1
leftmax
rightmax
res
else
if(a[r] >= rightmax)
rightmax = a[r]
else
r--
if(a[l] <= a[r])
if(a[i] > = leftmax) leftmax-a[i]
else res += (leftmax - a[i])
l++;
TUF


0 1 0 2 1 0 1 3 2 1 2 1
0 1 2 3 4 5 6 7 8 9 10 11
min(leftmax, rightmax) - a[i]
2-1 = 1
>=2
else → if(a[r] < a[l])
if(a[r] >= rightmax)
rightmax = a[r]
else res += (rightmax - a[i])
r--
if(a[l] <= a[r])
if(a[i] > = leftmax) leftmax-a[i]
else res += (leftmax - a[i])
l++;
TUF


min(leftmax, rightmax) - a[i]
2-1 = 1
TC → O(N)
SC → O(1)
else → if(a[l] < a[r])
if(a[r] >= rightmax)
rightmax = a[r]
else res + (rightmax - a[i])
r--
if(a[l] <= a[r])
if(a[i] >= leftmax) leftmax = a[i]
else res += (leftmax - a[i])
l++;
TUF


LeetCode
Explore
Problems
Mock
Contest
Discuss
Store
December LeetCoding Challenge
Premium
C++
Autocomplete
class Solution {
public:
int trap(vector<int>& height) {
int n = height.size();
int left=0; int right=n-1;
int res=0;
int maxleft=0, maxright=0;
while(left<=right){
if(height[left]<=height[right]){
if(height[left]>=maxleft) maxleft=height[left];
else res+=maxleft-height[left];
left++;
}
else{
if(height[right]>=maxright) maxright= height[right];
else res+=maxright-height[right];
right--;
}
}
return res;
}
};
Your previous code was restored from your local storage. Reset to default
TUF