Problem List

Run Submit

Description | Accepted | Editorial | Solutions | Submissions

42. Trapping Rain Water

Solved

Hard | Topics | Companies

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:

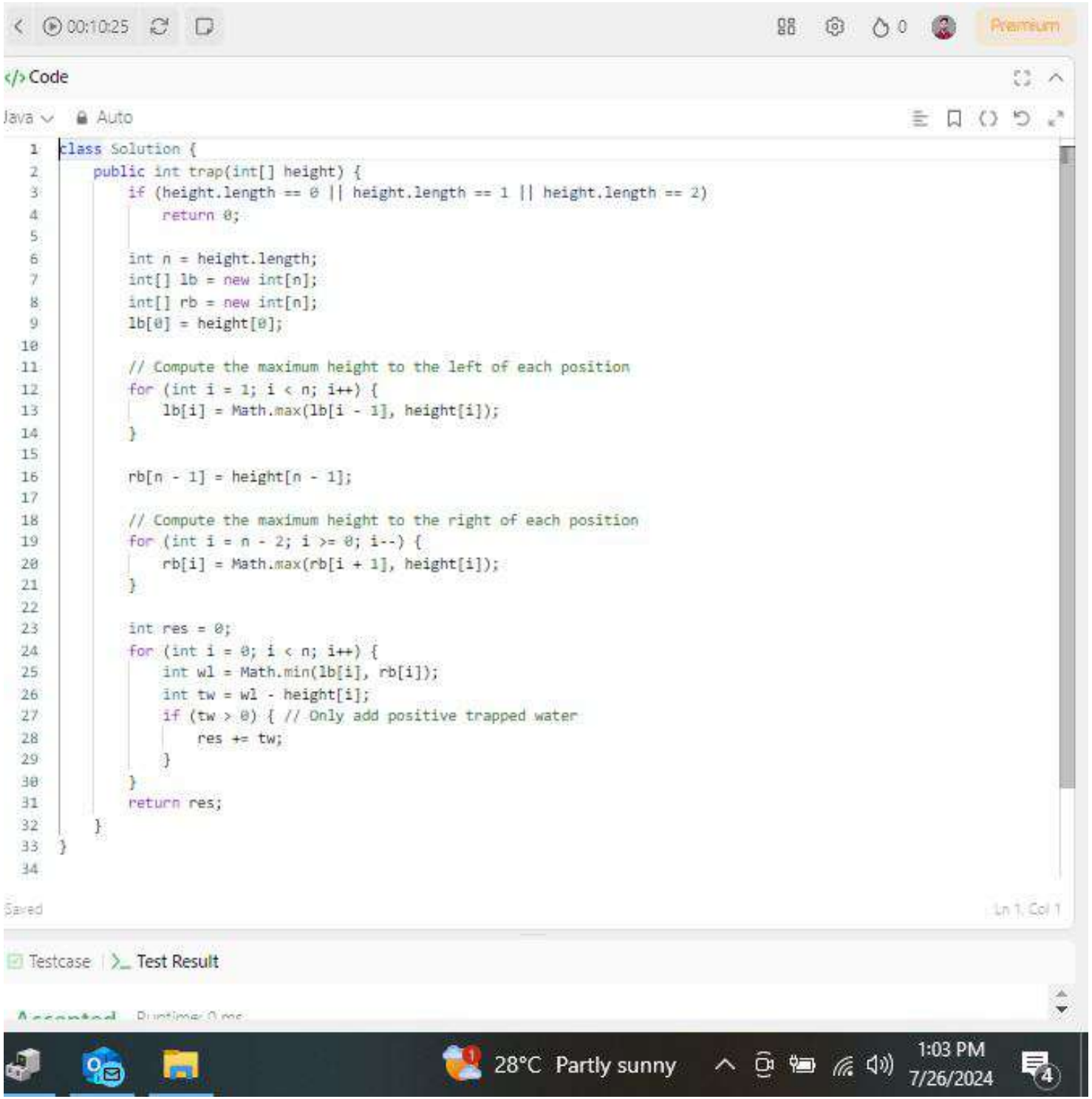
```
Input: height = [4,2,0,3,2,5]
Output: 9
```

Constraints:

- `n == height.length`
- `1 <= n <= 2 * 10^4`

32.1K | 271

Type here to search


00:10:25
0
Premium
Code
Java
Auto
class Solution {
public int trap(int[] height) {
if (height.length == 0 || height.length == 1 || height.length == 2)
return 0;
int n = height.length;
int[] lb = new int[n];
int[] rb = new int[n];
lb[0] = height[0];
// Compute the maximum height to the left of each position
for (int i = 1; i < n; i++) {
lb[i] = Math.max(lb[i - 1], height[i]);
}
rb[n - 1] = height[n - 1];
// Compute the maximum height to the right of each position
for (int i = n - 2; i >= 0; i--) {
rb[i] = Math.max(rb[i + 1], height[i]);
}
int res = 0;
for (int i = 0; i < n; i++) {
int wl = Math.min(lb[i], rb[i]);
int tw = wl - height[i];
if (tw > 0) { // Only add positive trapped water
res += tw;
}
}
return res;
}
}
Saved
Ln 1, Col 1
Testcase
Test Result
28°C Partly sunny
1:03 PM
7/26/2024

< Trapping Rain Water > Leetcode Problem

First Solution

Brute Force Solution is in $O(n^2)$

Approach

① Step 1 Find Left bar, right bar
② Step 2 Find water level (minimum(Lb, Rb))
③ Step 3 Find trapped water (WL - height of bar)
③ Step 3 Add trapped water and return

Code

```
class Solution {
    public int trap(int [] height) {
IF(height.length == 0 || height.length == 1 ||
  height.length == 2)          return 0;
```

```
int res=0;
for(int i=1; i<height.length-1; i++){
 int lb = height[i];                    // Guess
 for(int j= 0; j< i; j++){
      If(height[i] > lb) { lb = height[i]; }
  }
 int rb=height[i];
 for(int j= i; j< height.length; j++){
      If (height[j] >rb){
         rb=height[j];
        } }
int wl;
   If(rb > lb){ wl= rb; }
   else wl=lb;

 int tw=wl- height[i];
     res+=tw;
  }
  return res;
 }
}
```

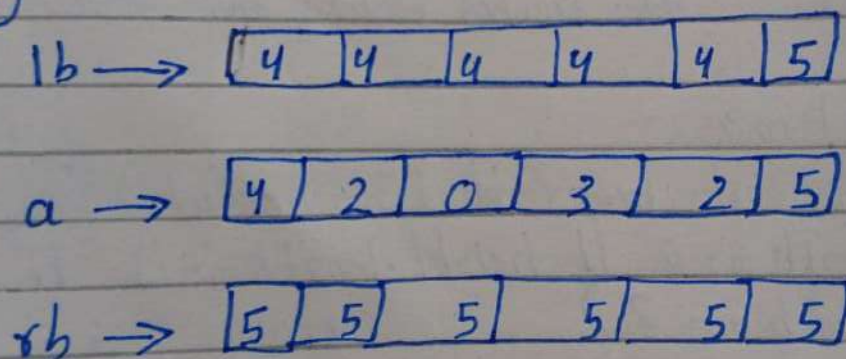
Solution(2) Time complexity O(n)

Approach>

① If length is equal to 0, 1, 2 return 0

② Create two newe arrays of same size

③



④ Calculate trapped water

```
class Solution{
    public int trap(int[] height){
    If(height.length==0 || height.length==1 || height.length==2
      ){    return 0;}

    int   n= height.length;
    int[] lb= new int[n];
    int   rb = new int[n];

     lb[0]=height[0];
//Calculate maximum height to left of each element
     for(int i=1; i<n; i++){
       lb = Math.max(lb[i-1], height[i]);
       }

     rb[n-1] = height[n-1];
// calculate the maximum height to the right of each
  position
     for(int i=n-2; i>=0; i--){
         rb[i] = Math.max(rb[i+1], height[i]);
       }

     int res =0;
// Calculate the trapped water
     for(int i=0; i<n; i++){
       int wl = Math.min(lb[i], rb[i]);
       int tw = wl - height[i];
        if(tw>0){    res+=tw; }
       return res;
     }
    }
```

Problem List

Run Submit

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted

Azan imtiaz submitted at Jul 26, 2024 12:57

Editorial Solution

Runtime

1 ms | Beats 69.24%

Analyze Complexity

Memory

45.94 MB | Beats 61.46%

Code | Java

```java
class Solution {
    public int trap(int[] height) {
        if (height.length == 0 || height.length == 1 || height.length == 2)
            return 0;

        int n = height.length;
        int[] lb = new int[n];
        int[] rb = new int[n];
```

View more

Type here to search