Rain Water Trapped

02 December 2023 12:08 AM

Problem Description
Given a vector **A** of non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.
From < https://www.scaler.com/academy/mentee-dashboard/class/90012/assignment/problems/47>

Problem Constraints
1 <= **|A|** <= 100000

Input Format
First and only argument is the vector **A**

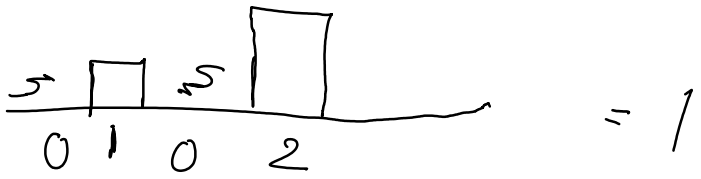
Output Format
Return one integer, the answer to the question

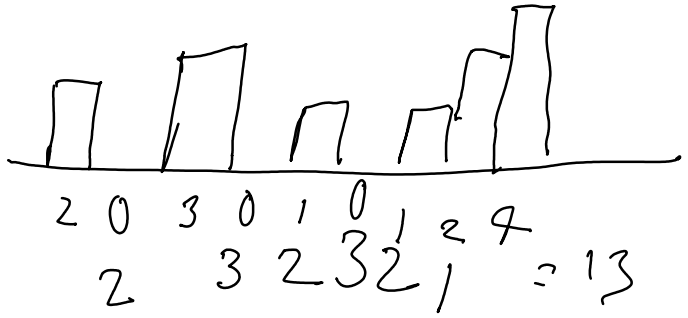
Input 1:
A = [0, 1, 0, 2]

Input 2:
A = [1, 2]

Output 1:
1

Output 2:
0





Bruteforce :-

iterate over each index of an vector.

find out the left highest & right highest element

take the min from both & substract it from current index element.

& Calculate the sum of all values.

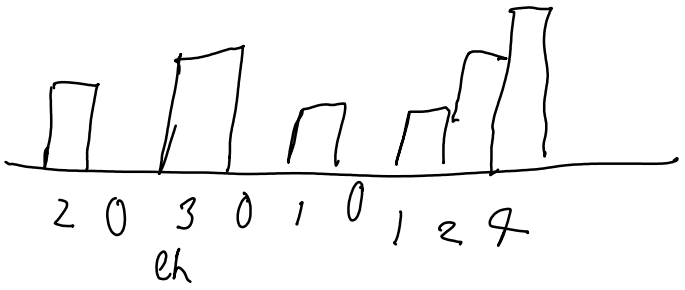
T.C. $O(n^2)$

S.C $O(1)$

Optimization :- Prefix Sum

eg :- [2,0,3,0,1,0,1,2,4] = 13

rh = [4,4,4,4,4,4,4,4,4]

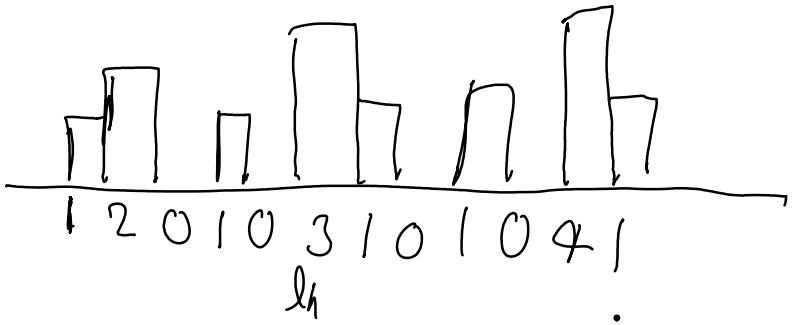


Min(lh, rh) - A[i]
If (i > lh)
lh = i

0 2 0 3 2 3 2 1 0 = 13

eg :- [1,2,0,1,0,3,1,0,1,0,4,1] = 15

rh = [4,4,4,4,4,4,4,4,4,4,4,1]



0 0 2 1 2 0 2 3 2 3 0 0 = 15

Solution :- Calculate the right highest element using prefix sum array.
Iterate over the input array calculate the value by checking.
lh element if it is greater the the current element
Replace it with current element taking the minimum of lh and rh

& Substracting the current element.
& Suming all the values in the end.

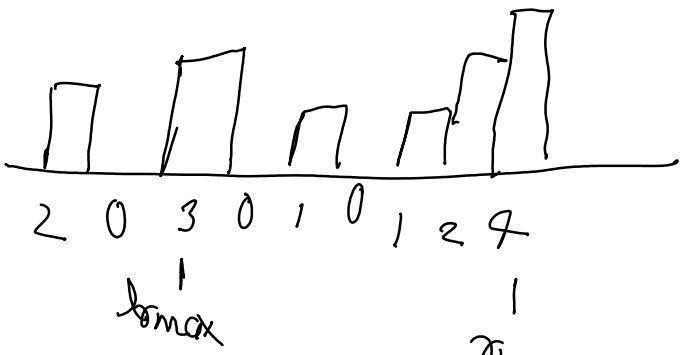
T. C. $O(n)$ — Creating Prefix sum highest array.
$O(n)$ — to find the answer.
$= O(2n) = O(n)$

S. C. $O(n)$ — for Psum array.

— x — x — x — x — x — x — x — x — x —

Optimization using pointers :-

eg :- [2, 0, 3, 0, 1, 0, 1, 2, 4] = 13



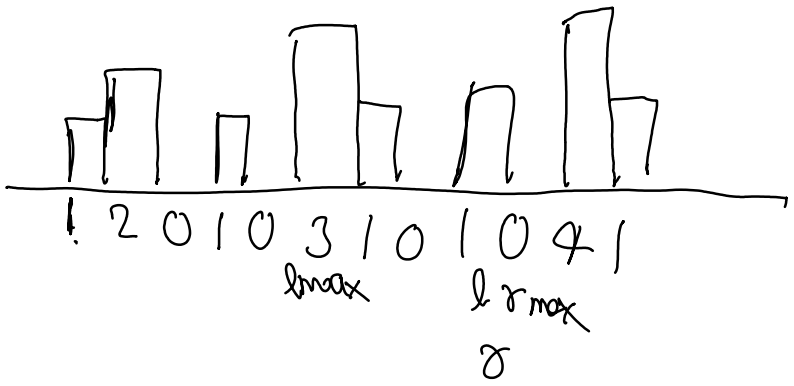
$l = 0$
$r = n - 1$
$lmax = 0$
$rmax = 0$
total $= 0$

$l = 0$
$r = 8$

```
while (l <= r) {
  if (lmax <= rmax) {
    int sum = lmax - A.get(l);
      if (sum > 0) {
        total += sum;
    } else {
        lmax = A.get(l);
    } l++
  } else {
    int sum = rmax - A.get(r);
     if (sum > 0) {
        total += sum;
    } else {
       rmax = A.get(r);
    }
      r--;
  }
}
return total;
```

lmax = 3
rmax = 4
total = 2 + 3 + 2 + 3 + 2 + 1 = 13

2



l = 0
r = 11
lmax = 3
rmax = 4
Total = 2 + 1 + 2 + 2 + 3 + 2 + 3
= 15

Solution :- Use two pointer approach use four variables l, r, lmax & rmax initialize l to be 0 & r to last index of vector & lmax & rmax to be 0 & iterate till r<=r if lmax <= rmax calculate the sum by substracting current element from lmax if it is greater than zero add it in the output total. else assign the current element to the lmax. & Increment the l

Whereas if rmax > lmax calculate the sum by substracting current element from lmax if it is greater than zero add it in the output total. else assign the current element to the rmax & decrement r

T.C. = O(n)

S.C = O(1)