Agenda

13 June 2024 13:13

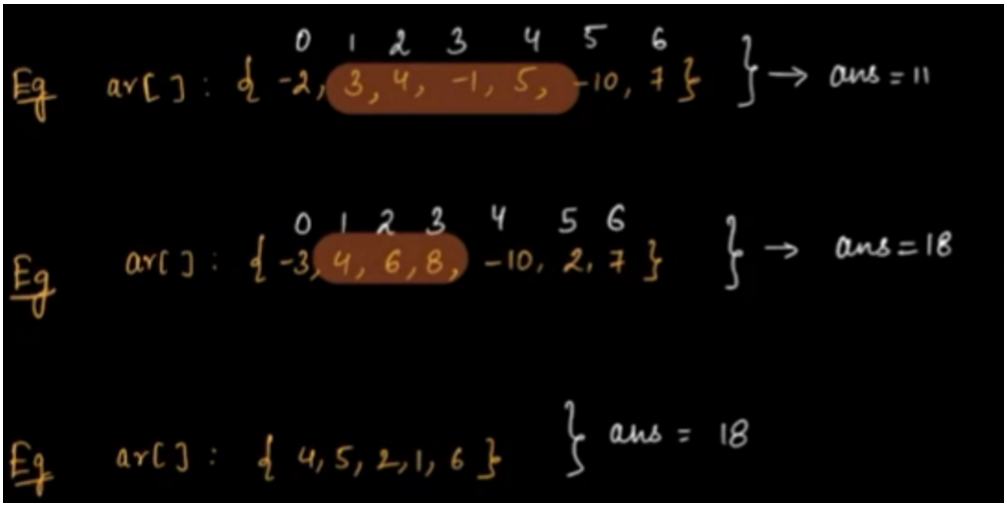
1. Max Subarray Sum
2. Add x from i to N-1
3. Add x from i to j
4. Rain water Trapping

Problems

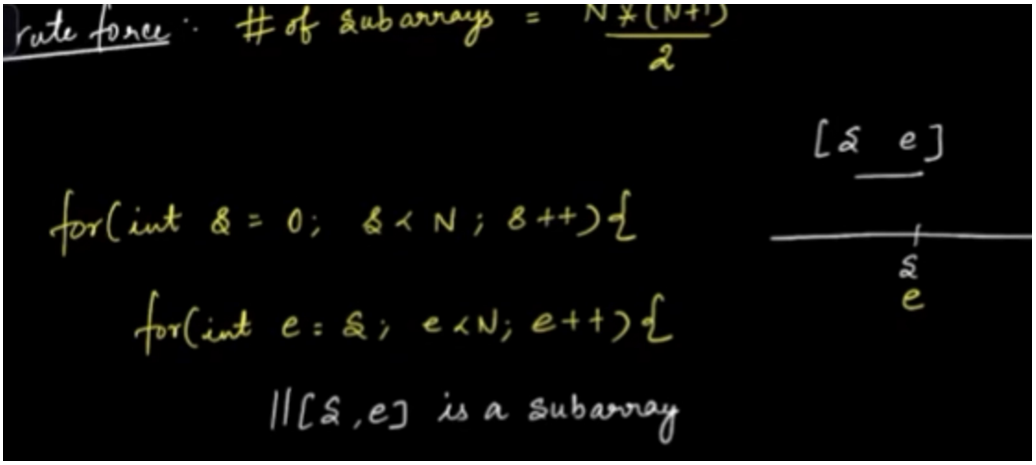
13 June 2024 13:15

Problem1:

Given an array. Find the max subarray sum.



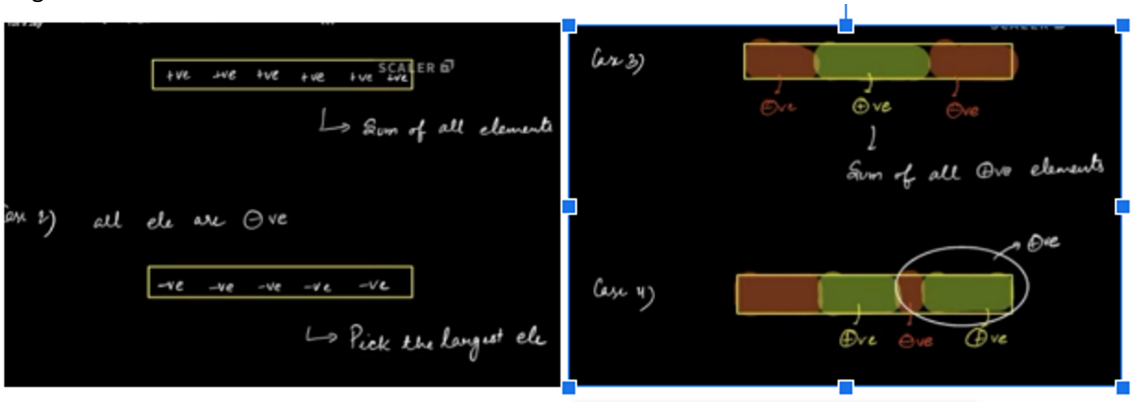
1) Brute Force:

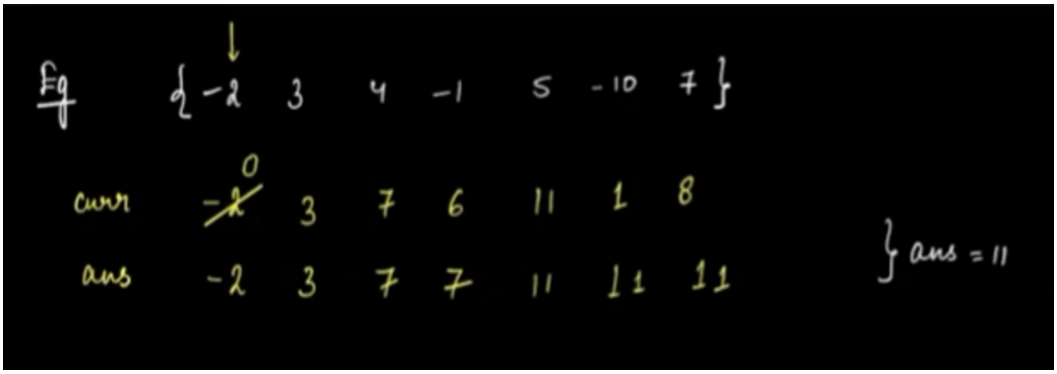


2) Using Prefix sum : Calculate the prefix sum and using prefix sum find the subarray. For each i and j calculate difference in prefix sum (this gives sum of element in subarray i to j)

3) Using carry forward : use a variable which holds sum of each consecutive elements of consecutive subarrays .

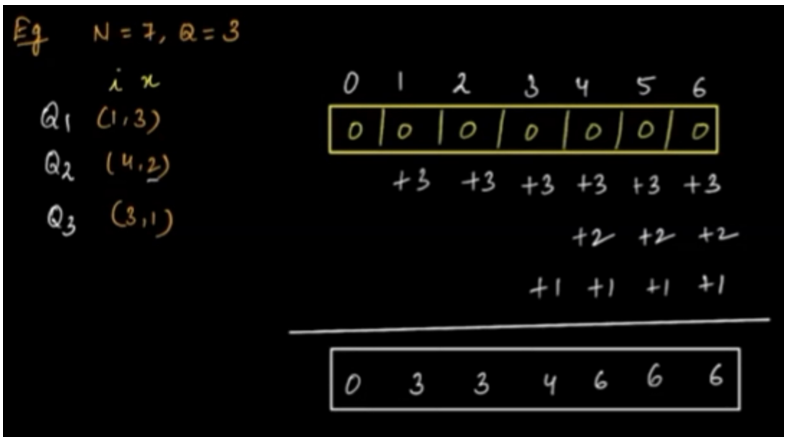
4) Kadane's Algorithm :





Problem 2:

Given an array where every element is 0. Find the final array after performing multiple queries.
Query(i, x) :- Add x to all numbers from index i to N-1

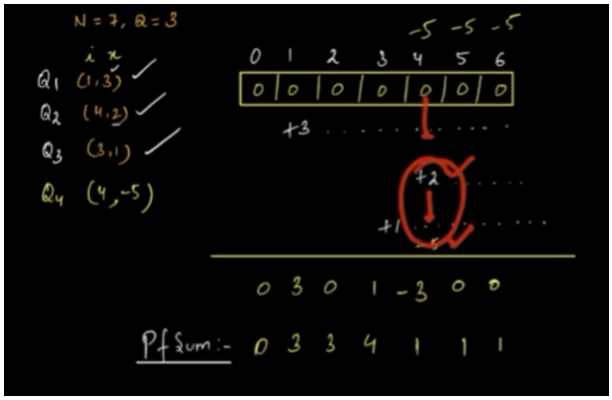


Brute Force:

For every query add respective elements in array.

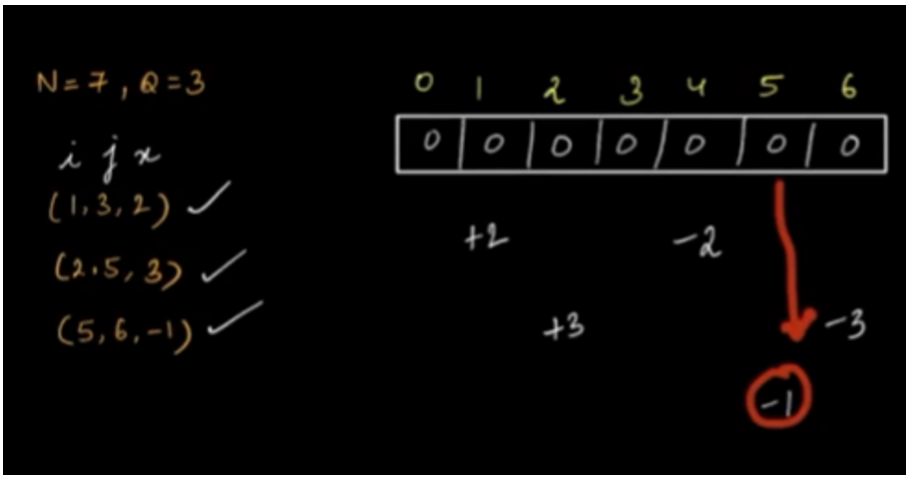
Using Prefix Sum :

- Add all the values in the respective index.
- Do the prefix sum in end;



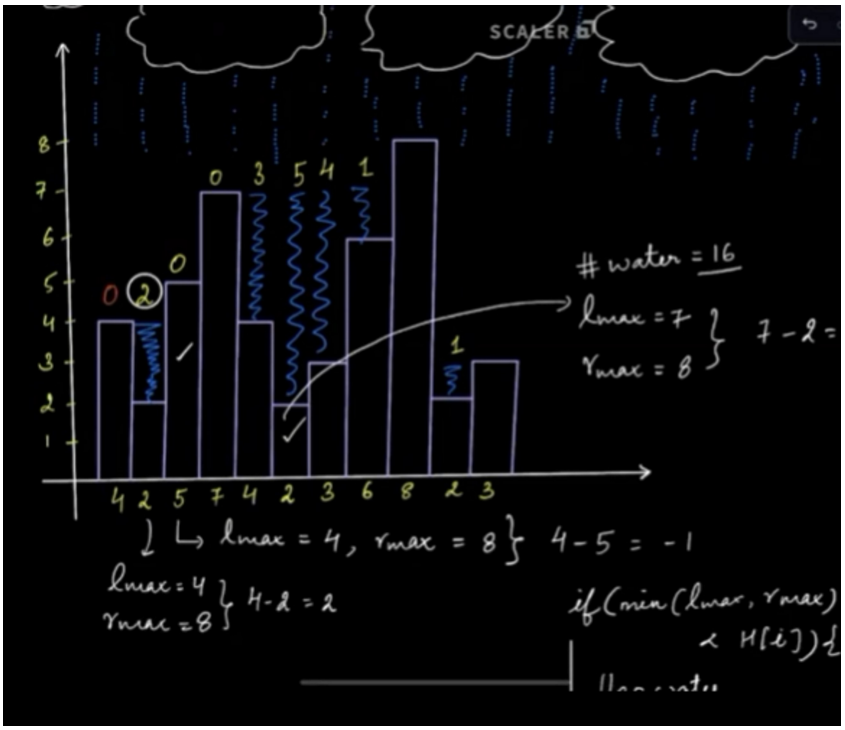
Problem 3:

Given an array where every element is 0. Find the final array after performing multiple queries.
Query(i, j, x) :- Add x to all numbers from index i to j



Problem 4:
Rain Water Trapping

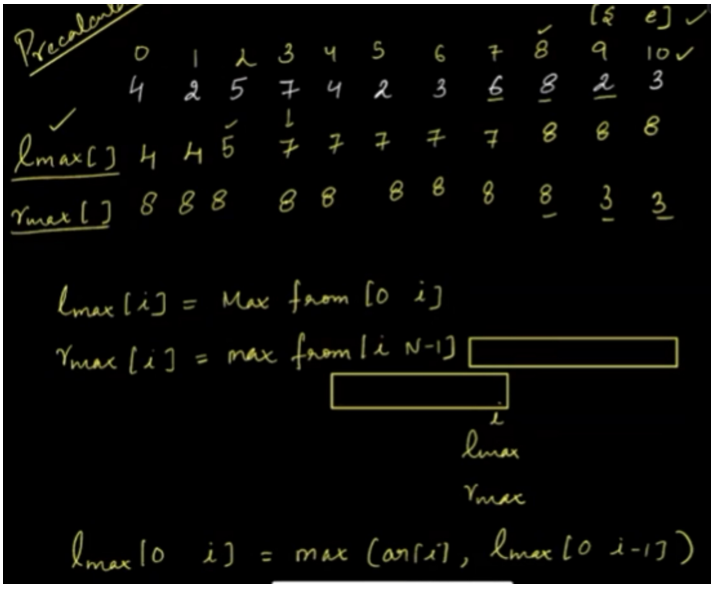
Given N buildings with height of each building. Find the rain water trapped in between the buildings.



Brute Force Approach:

Find the left maximum and right maximum for each building
Find the minimum among them.
If minimum is greater then the current building size = calculate the difference between minmax and current building.
Do for each building sum up and return.

Optimised Code:
Keep two arrays which stores leftMax(Maximum among all elements present in left side) and rightMax.


Precalculate
[2 e]
0 1 2 3 4 5 6 7 8 9 10
4 2 5 7 4 2 3 6 8 2 3
lmax[] 4 4 5 7 7 7 7 7 8 8 8
rmax[] 8 8 8 8 8 8 8 8 8 3 3
lmax[i] = Max from [0 i]
rmax[i] = max from [i N-1]
i
lmax
rmax
lmax[0 i] = max (arr[i], lmax[0 i-1])