Topic Array -

Prefix Sum:

1. Given an array of M element d queries For each of the query print sum of ele from index s to e. s and e will be given to you for each query

e.g

0 1 2 3 4 5 6 7 8 9

A[] = { -3, 6, 2,4,5,2,8,-9,3,1}

Q queries: 5

s [] = {4,2,1,0,7}

e[] = {8,7,3,4,7}

Brute Force approach:

Step1: For each Q queries

Step2: Take sum from S[i] to e[i]

Step3: Print Sum

Time complexity: O(Q \* N)

Space complexity: S(1)

Prifix Sum:

Step1: Create Prifix sum array i.e PF[n]

PF[0] = A[0];

for(i = 1; i < n; i++)

PF[i] = PF[i-1] + A[i]

Step2: Iterate Q queries from 0 to Q

Step3: sum = PF[e] – PF[s-1]

Step4: Print Sum

TC = O(Q) + O(N)

TC = O(N + Q)

SC = O(N)

Use of Application:

Application of Prefix Sum Range Sum Queries : where we have to tell sum from s to e in Array

Q2: Given an array of size N court no of equilibrium index. An index is said to be equilibrium index if sum of all ele b4 i th = sum of all elements after ith index

e.g i

[0 i-1] [i+1 N-1]

0 1 2 3 4 5 6

A[] = { -7 1 5 2 -4 3 0}

Brute force:

Index Equibilirium sum is equivil

0 0 == 7 False

1 -7 ==6 False

2 -6 == 1 False

3 -1 == -1 True

4 1 == 3 False

5 -3 ==0 False

6 0 == 0 True

Step1: Iterate i from 0 to N (O(n))

Steps2: Find leftSum from 0 to i -1 (Iterate ) (O(n))

Step3: Find rightSum from i + 1 to N (iterate) (O(n))

Step4: Compare leftSum == rightSum if equal count = count + 1

Step5: return count;

TC = O(n \* 2n) = (n)2

SC = O(1)

Optimised 1:

A[] = { -7 1 5 2 -4 3 0}

Step1: Create PF array

PF = { -7, -6, -1, 1, -3, 0, 0}

// Step2: SF = [0,7,6,1,-1,3,0] // approach – compare PF == SF if match then equi ++

Step3: Iterate A from 0 to N in i

Step4: if i == 0 then leftsum = 0; else leftsum = PF[i-1] Check if leftsum == PF[N-1] – PF[i] then count = count + 1

Step5: return count;

TC = O (N)

SC = O(N)

Approach: 2

A = {1, 2, 3}

Sum = 6

I currentSum sum – rightSum

(Sum – currentSum = rightSum) currentSum – A[i] = leftSum

0 A[0] = 1 5 1-1 = 0

1 3 3 1

2 6 0 3

TC = O(N)

SC = O(1)

A[] = { -7 1 5 2 -4 3 0}

Total sum = 0

I currentSum sum – rightSum

(Sum – currentSum = rightSum) currentSum – A[i] = leftSum

0 -7 7 0

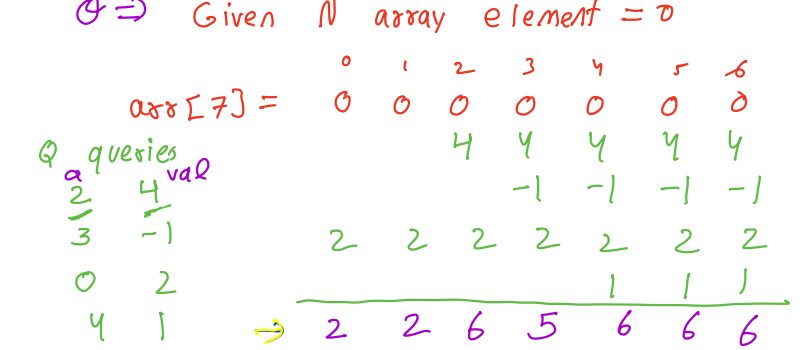
1 -6 6 -5

2 -1 1 6

3 1 -1 -1

Array Advance:

1. Given N array element = 0 and Q queries having index and value



Brute force:

Step1: Iterate Q from 0 to n in (index, value )

Step2: Iterate A from index to N and Add the value in A i.e A[i] = A[i] + value

Step4: print Array

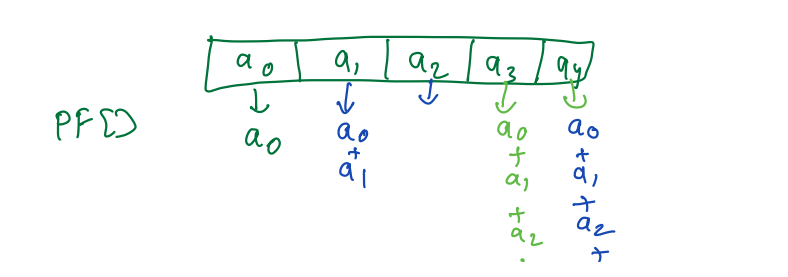
TC = O(Q \* N)

SC = O (1)

Optimised Approach:

A[] = 2, 0, 4,-1,1,0,0

PF = 2,2,6,5,6,6,6



Step1: Iterate Q queries from 0 to N in index, value

Step2: Set value in respective index of A

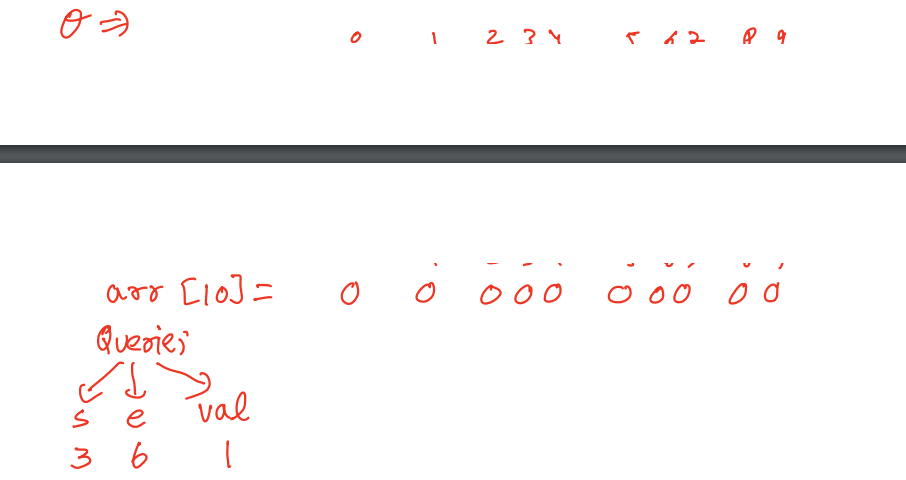
Step3: Calculate PF sum

Step4: Return A

TC = O(Q + N)

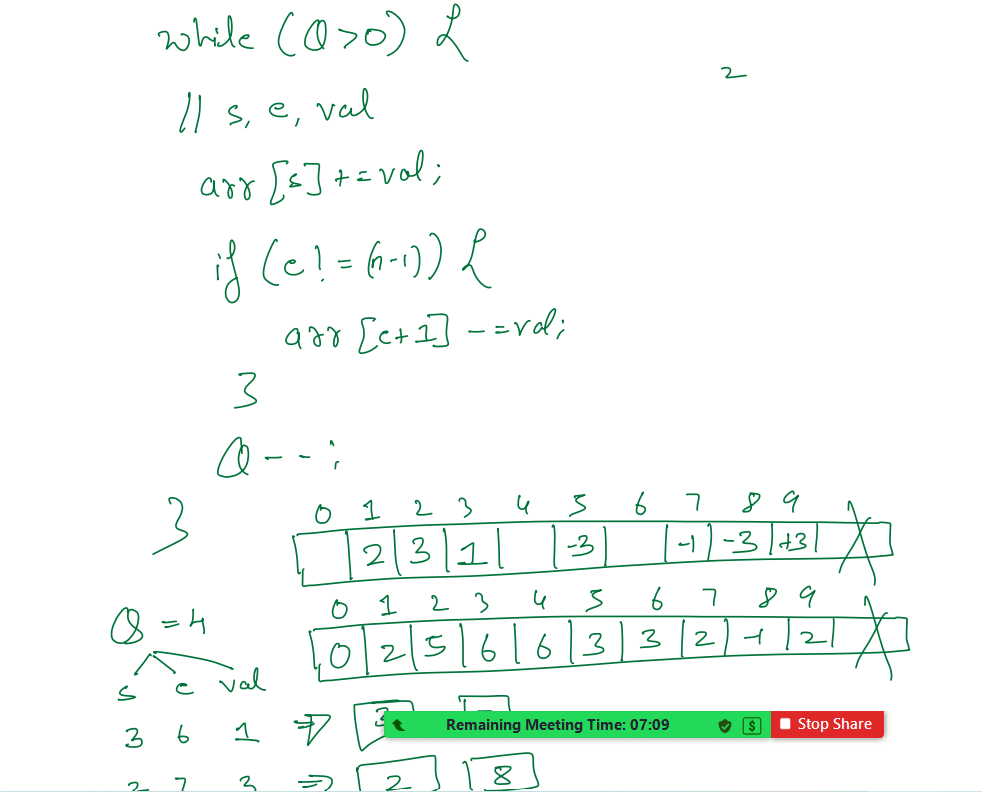
SC = O(1)

Q4:



BF: As above Q3

Optimised:



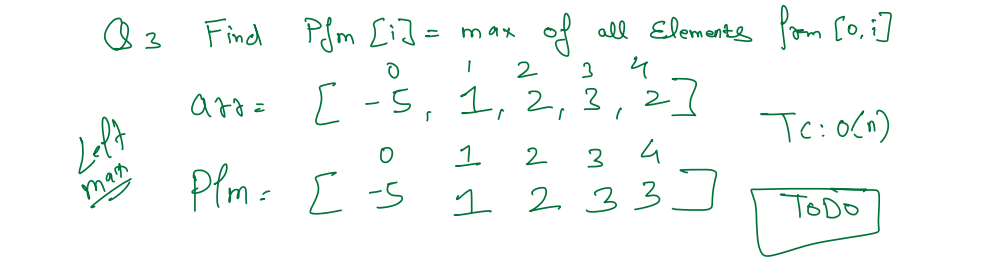
A [] = 0 0 0 0 0 0 0 0 0

TC = O(Q + N)

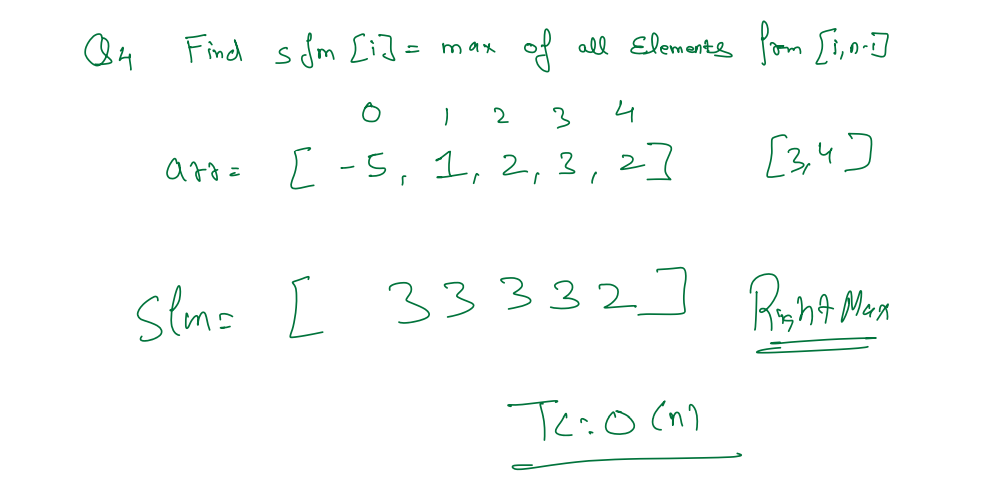
SC = O(1)

Q5:

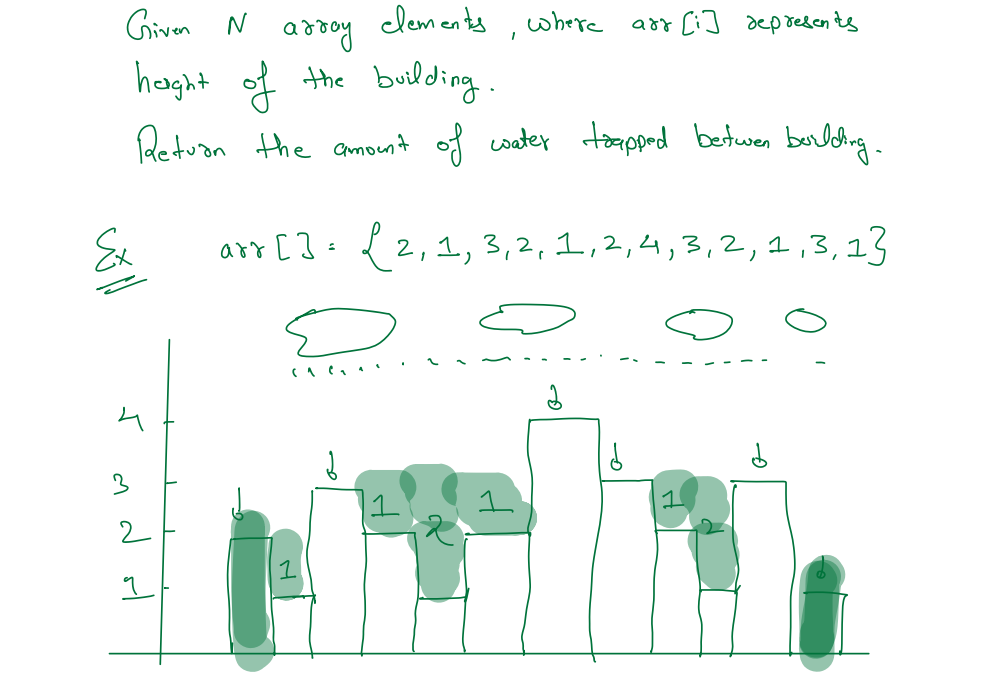
Left Max



Q6: Right Max

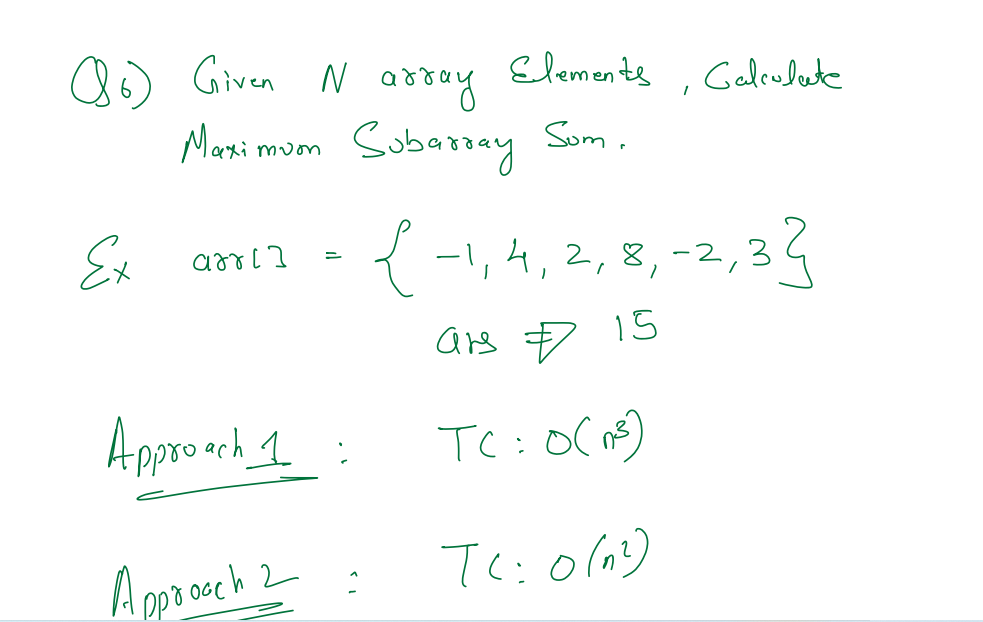


Q7: Very important



Subarray Problem

Kadane algorithm



Boot force:

List<List<int>> SubArrayResult;

int maxSubArray = int.Min;

for (i = 0; i< n ; i++)

for (j = i; j < n; j++)

List<int> subarray; int currentSum = 0;

for(k = i; k <= j; k++)

{

subarray.Add(A[k]);

currentSum = currentsum + A[k]

}

maxSubArray = Max(maxSubArray, currentSum);

subArrayResult.Add(subarray);

}

}

i j k subarray

0 0 0 -1

0 1 1 -1,4

0 2 2 -1,4,2

1 1 1 4

1 2 1 4,2

Begger outside temple

A = 5

B = [[1, 2, 10], [2, 3, 20], [2, 5, 25]]

Brute force

Result[A] = 0;

for (i =0 ; i< B.Length; i++)

{

s = B[i][0];

nIndex = B[i][1];

p = B[i][2];

for (i =s ; i<= nIndex; i++)

{

Result[i-1] = Result[i-1] + P

}

}

return Result;

TC = O(N\*N)

1, 2 ,3 , 4 ,5

0, 0, 0, 0, 0

10 0 -10 0 0 10,45,-10,-20,0

0 20 0 -20

0 25 0 0 0

10,45,-10,-20,0

PF 10, 55, 45, 25,25

Optimised Approach:

Result[A] = 0;

for (i =0 ; i< B.Length; i++)

{

s = B[i][0];

nIndex = B[i][1];

p = B[i][2];

Result[s-1] = Result[s-1] + p;

Result[nIndex] = Result[nIndex] - P;

}

for (var i = 1; i < A; i++)

Result[i] = Result[i - 1] + Result[i]

return Result;