Prefix Sum Intro 2

1) Equilibrium Index

Define an Array, find the {Index of Equilibrium} which satisfies Both, Index and Sum of Array Subarray, Common part of array $\Rightarrow \frac{(N)(N+1)}{2}$

Ex: $\{1, 2, 3, 1, 3, 4, 6, 3\}$ — Mid: 4, Mid: 1

0 1 2 3 4 5 6 7

2 = 5 = 3

Mid not there

0 + 6 = 1+4 = 7

3 = 6 = 1+4 = 7 ✓ ans = 4

0 1 2 3 4 5 6 7 8 9 10

Ex: $\{2, 2, 2, 4, 5, 5, 1, 5, 2, 4, 4\}$

5 = 8 ; len = 4

8 = 10 ; len = 3

ans = 3

Observations

1) Only 1 mid 1 mid

2) Mid 1 mid will only be a boundary

Steps: Iterate & get mid-sum value & mid-sum[T]

M[mid] : mid[mid] = 6

0 1 2 3 4 5 6 7 8 9

Ex: $\{2, 2, 2, 4, 5, 5, 1, 5, 2, 4, 4\}$

[2 2 2 4 5 5 1 5 2 4 4]

[2, 4] [8, 10]

Pseudo code:

if arr[T] == Mid_Ele: get from mid-sum & print

if arr[T] == Mid_Ele: get from mid-sum & print

ans = N

get mid_ele & mid-ele

if mid_ele == mid_ele: return 1

for T = 0; T < N; T++{

if (arr[T] == Mid): get mid_ele & mid-ele & break

else if (arr[T] == Mid): get mid_ele & mid-ele & break

ans = min(ans, T-Mid)

}

return ans;

If multiple times present we need to get from R.H.S

Ex: arr[2] : 2 2 2 4 5 5 1 5 2 4 4

mid = 8

ans = 2

mid-sum = 8

ans = 2

Steps: Calculate mid-value & mid-value } O(N)

if (mid-value == mid-value) return 1

Steps: mid-T = 2N, mid-T = 2N, ans = N

Steps: for T = N-1; T > 0; T--{

if (arr[T] == mid-value){ mid-T = T }

else if (arr[T] == mid-value){ ans = min(ans, mid-T-T) mid = T }

if (ans == min(mid-mid, ans)){ ans = min(ans, mid-mid) }

}

return ans;

Ex: 0 1 2 3 4 5 6 7 8 9

2 2 2 4 5 5 1 5 2 4 4

ans = min(ans, 8-T+1)

ans = 3

mid-T = 2

mid-T = 2

mid-sum = 8

mid-T = 2

ans = min(ans, 8-T+1)

ans = 2

mid-T = 2

mid-sum = 8

mid-T = 2

O(N)

mid-T = 18

mid-mid = 18

0 1 2 3 4 5 6 7 8

3 -1 4 8 -1 4 6 5 0

mid = -1

mid = 8

ans = min(ans, 18-8+1)

ans = 11

return ans;

TC: O(N) SC: O(1)

Equilibrium Index: Index of an array.

An index (T) said to be equilibrium, if

Sum of all array elements before T^th index = Sum of all array elements after T^th index.

Sum[0, T-1] = Sum[T+1, N-1]

Note: for T = 0, we can assume sum of all T-1 is = 0.

[0, -1]

Note: for T = N-1, we can assume sum of all T+1 is = 0.

[N-1, N-1]

Ex: arr[2] = -7 1 5 2 -4 3 0

0 1 2 3 4 5 6

Equilibrium Index

Sum[0, T-1] = -1 ; Sum[T+1, N-1] = -1 ✓ ; [T] = 3

Sum[0, T-1] = 0 ; Sum[T+1, N-1] = 0 ✓ ans = 3

Ex: arr[5] = 3 -1 2 -1 1 2 1

0 1 2 3 4 5 6

Sum[0, T-1] = 4 ; Sum[T+1, N-1] = 4 ; [T] = 3

Ex3: arr[3] = -2 3 1 -4 -1 4 -3

0 1 2 3 4 5 6

Equilibrium Index

0 : Sum[0, T-1] = 0 ; [T-1] = 0

5 : Sum[0, T-1] = -3 ; [T] = -3

Solution:

1) For each index value check if this equilibrium or not.

O(N):

if return T is equilibrium

Sum[0, T-1] = Sum[T+1, N-1]

And arr[T] is both 0 or 1

Sum[0, T-1] + arr[T] = arr[T] + Sum[T+1, N-1]

Sum[0, T-1] = Sum[T, N-1]

C = 0;

T = 0; T < N; T++{

to get T^th Equilibrium

leftSum = [0 : T-1] ; T=0 error

rightSum = [T+1 : N-1] ; T=N-1

if (leftSum == rightSum)

{ C++ }

}

→ O(N^2) SC: O(1)

C = 0;

T = 0; T < N; T++{

to get T^th Equilibrium

leftSum = Sum[0 : T-1]

rightSum = Sum[T+1 : N-1]

if (leftSum == rightSum){

C++

}

}

optimize leftSum & rightSum

Both queries can be both prefix

Both queries can be solved using a single prefix sum

TC: O(N + N) SC: O(N)

Because we are calculating all prefix sum for the entire array.

for a i:

leftSum : [0 : T-1]

rightSum : [T+1 : N-1]

Calculate right-sum using

leftSum = ?

Right-Sum = Total-Sum − Left-Sum + arr[T]

[0, 1, 2, 3] T+1, T+2 ... N-1 ; [0, 1, 2, 3]

Iterate & calculate total sum.

C = 0

leftSum = 0

T = 0; T < N; T++{

to get T^th Equilibrium

leftSum = leftSum + arr[T-1]

rightSum = TotalSum − leftSum − arr[T]

if (leftSum == rightSum){

C++

}

}

return C;

We can use carry forward & calculate leftSum on the fly

leftSum = 0

T	leftSum	rightSum
0	arr[0]	
1	arr[0]+arr[1]	

T = 0; T < N; T++{

if (arr[T] == 0){

ans = ans + arr[T]

}

return ans;

}

Q2) Count N-th Spoiled Element

An element is spoiled if after removing the element from array, Sum of all elements at even index = Sum of all elements at odd index

Ex: 0 1 2 3 4 — 2 4 3 3 3

Ex: 0 1 2 3 4 5 — 1 3 2 7 4 -2