Arrays : Prefix Sum

Nov 29, 2023

AGENDA

- Intro to prefix sum technique
- 2 interesting questions

```
ans = max(arr)          <-- O(N)

for i in arr:           | T.C: O(N)
    ans = max(ans, i)

arrays.sort()           <-- n log n
```

XX

T.C: O(1)

```
if (ans > i)
    ans = i
```

Range-sum-Query

Q. Given N array elements and Q queries, for each query, calculate sum of all elements from L to R (inclusive) (0-indexed)

0	1	2	3	4	5	6	7	8	9
-3	6	2	4	5	2	8	-9	3	1

	L	R	Ans.
Q1:	1	4	6+2+4+5 = 17
	1	6	27
	1	3	12
	2	4	11
	0	7	15

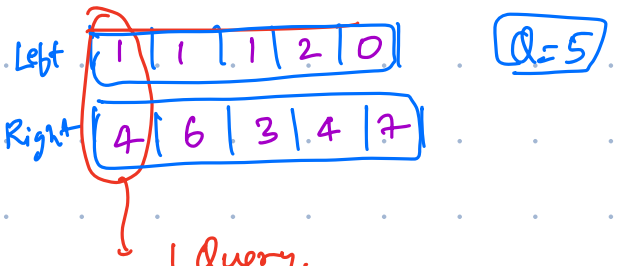
B.F.

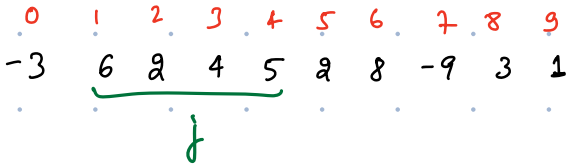
* For each query, iterate from L to R and find the sum.

Q is independent ← `for(int i=0; i<Q; i++)`

```
{
    L = left[i]
    R = right[i]
    // [L,R] is your query.
    sum=0
    for(int j=L; j<=R; j++)
    {
        sum += arr[j]
    }
    print(sum)
}
```

worst case ↓ N iterations ←





L=1
R=4

L=1
R=6

T.C. → O(Q*N)

S.C. → O(1)

Not good enough.

0 1 2
[1, 3, 5]

Q: 0-2
1-2
2-2
1-2
0-2
0-1
0-2
0-1
0-0
1-1

e.g. Cricket score-board

After →	1	2	3	4	5	6	7	8	9	10
	2	8	14	29	31	49	65	79	88	97

Runs scored in

7th over = Runs scored at the end of 7th over − Runs scored at the end of 6th over

= 65 − 49 = 16

2nd over = 8 − 2 = 6

Last over = 97 − 88 = 9

6 - 10th over = Runs scored at the end of 10th over − Runs scored at the end of 5th over

= 97 − 31

= 66

After →	1	2	3	4	5	6	7	8	9	10
	2	8	14	29	31	49	65	79	88	97

3 - 6th over = 49 − 8 = 41

4 - 9th over = 88 − 14 = 74

1 - 6th over = 49

* Obsv.

We were able to answer our range sum queries in constant time (no iteration reqd.) due to the cumulative score-board.

Num. of runs scored in each over

↓

arr: 6 3 0 36 5 15

↓

Convert this into a score-board (cumulative)

↳ 6 9 9 45 50 65

↓

Now, to answer each query, you will need O(1) time.

	0	1	2	3	4	5	6	7	8	9
Original arr:	-3	6	2	4	5	2	8	-9	3	1
(Cumulative) [prefix Sum Array]	-3	3	5	9	14	16	24	15	18	19

Code [to create prefix sum array]

```
int pf[N];
Pf[0] = arr[0]

for(int i=1; i<n; i++)
{
    pf[i] = Pf[i-1] + arr[i]
}
```

[1 0 3 6 2 8]

[1 1 4 10 12 20]

i

T.C. = O(N)

S.C. = O(N)

How to answer the Queries?

0	1	2	3	4	5	6	7	8	9
-3	6	2	4	5	2	8	-9	3	1

0	1	2	3	4	5	6	7	8	9
-3	3	5	9	14	16	24	15	18	19

```
for(int i=0; i<Q; i++)
{
    L = left[i]
    R = right[i]
    // [L,R] is your query.
    if (L==0)
        sum = Pf[R]
    else
        sum = Pf[R] - Pf[L-1]

    print(sum)
}
```

Given L<=R!

T.C. = O(Q)
S.C. = O(1)

1-6

24 - (-3) = 27

[Very important]

Total

T.C. → O(N+Q)

S.C. → O(N)

can be improved if you modify the original array itself for prefix sum.

(should be avoided or discussed with the interviewer ☺)

Break till 8:08 AM

Q.

Given array of size N and Q queries = [L, R] :
For every query, return the sum of all even-indexed elements from L to R.

0	1	2	3	4	5
2	3	1	6	4	5

L	R	
1	3	→ 1
2	5	→ 1 + 4 = 5
0	4	→ 2 + 1 + 4 = 7
3	3	→ 0

is 0 even? Yes :)

0	1	2	3	4	5
2	3	1	6	4	5

↓

L	R	
1	3	2 3 7
2	5	
0	4	
3	3	

0	1	2	3	4	5
2	3	1	6	4	5

Sum of even-indexed elements from L to R

= Sum of even-indexed elements till R

− Sum of even-indexed elements till L-1.

In your prefix sum array.

→

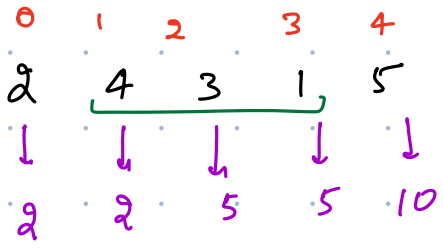
Pf[i] → Sum of even-indexed elements till i.

0	1	2	3	4	5
2	3	1	6	4	5
↓	↓	↓	↓	↓	↓
pf: 2	2	3	3	7	7

Have clarity on: what pf[i] should contain.

pf[i]

↳ denotes sum of even-indexed elements upto i.



L - R Pf[R] - Pf[L-1] = 5 - 2 = 3

Code.

```
int pf[N]
pf[0] = arr[0]
for(int i=1; i<n; i++)
{
    if (i%2 == 0)
    {
        pf[i] = pf[i-1] + arr[i]
    } else
    {
        pf[i] = pf[i-1]
    }
}
```

T.C. → O(N+Q)
S.C. → O(N)

```
for(int i=0; i< Q; i++)
{   // L, R as input.
    if (L==0)   sum = pf[R]
    else        sum = pf[R] - pf[L-1]
}
```

Extension

Q. Q queries : (L-R)

Return the sum of odd-indexed elements..

```
int pf[N]
pf[0] = 0
for(int i=1; i<n; i++)
{
    if (i%2 != 0)
    {
        pf[i] = pf[i-1] + arr[i]
    } else
    {
        pf[i] = pf[i-1]
    }
}
```

1 3 6 2

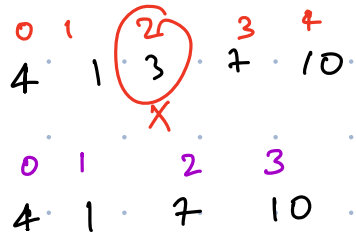
Q. **Special index**

Given an array of size N, count the no. of special indices in the array.

Note: A special index is an index after removing which, sum of even-indexed elements become equal to sum of odd-indexed elements.

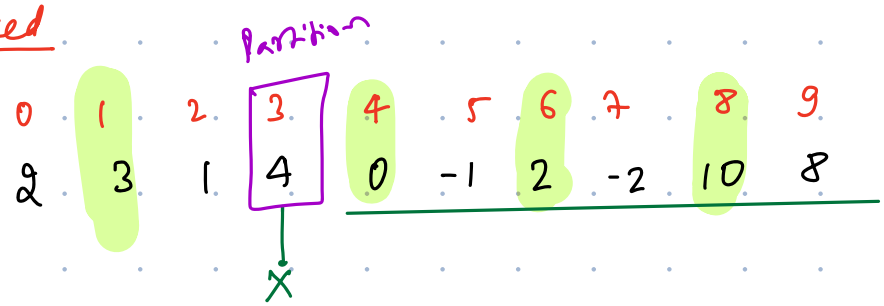
	0	1	2	3	4	5
	4	3	2	7	6	-2

i=0	0	1	2	3	4	S_e = 8	S_o = 8 ✓
	3	2	7	6	-2		
i=1	0	1	2	3	4	S_e = 9	S_o = 8 ✗
	4	2	7	6	-2		
i=2	0	1	2	3	4	S_e = 9	S_o = 9 ✓
	4	3	7	6	-2		
i=3	0	1	2	3	4	S_e = 4	S_o = 9 ✗
	4	3	2	6	-2		
i=4							✗
i=5							✗



11

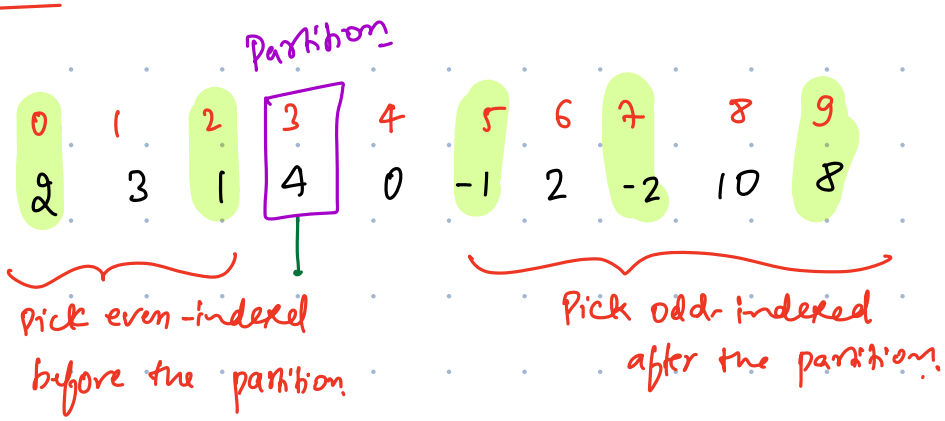
Sum of odd-indexed



0 1 2 3 4 5 6 7 8
2 3 1 0 -1 2 -2 10 8

3 + 0 + 2 + 10 = 15 ✓

Sum of even-indexed



2 + 1 + (-1) + (-2) + 8

= 8

Special index

0	1	2	3	4	5
4	3	2	7	6	-2

i = 0 Check whether i=0 is a special index.

Sum of odd-indexed elements after removing i
= Sum of even-indexed elements from 1 to n-1.
= 8 (2+6)

Sum of even-indexed elements after removing i
= Sum of odd-indexed elements from 1 to n-1
= 3+7-2 = 8

0	1	2	3	4	5
4	3	2	7	6	-2

(F)

i = 2

Sum of odd-indexed elements after removing i
= Sum of odd-indexed from 0 to i-1 +
Sum of even-indexed from i+1 to n-1
= 3 + 6 = 9

Sum of even-indexed elements after removing i

= Sum of even-indexed from 0 to i-1 +
Sum of odd-indexed from i+1 to n-1

= 4 + (7 + -2)

= 4 + 5 = 9

Code.

```
// Calculate pf sum of odd-indexed elem.  Pf odd[]
//          "          "   even-indexed elem.  pf Even[]
```

	0	1	2	3	4	5
	4	3	2	7	6	-2
pfOdd:	0	3	3	10	10	8
pfEven:	4	4	6	6	12	12

cnt=0

```
for( int i=0 ; i<n; i++)
{
    // Check if i is a special index.

    if(i==0)
    {
        // Left partition doesn't exist.
        // Sum-odd = sum of even indexed elements in right side.
        sum-odd = pfEven[n-1] - pfEven[i]
```

↳ Sum of even indexed elements from i+1 to n-1.

```
        // Sum-even = sum of odd indexed elements in right side.
        sum-even = pfOdd[n-1] - pfOdd[i]
```

↳ Sum of odd indexed elements from i+1 to n-1.

```
    } else

    {
```

// sum-odd = Sum of odd elements on left side of i + Sum of even elements on right side of i

sum-odd = pfOdd[i-1] + pfEven[n-1] - pfEven[i]

pfOdd[i-1]: Sum of odd indexed from 0 to i-1

pfEven[n-1] - pfEven[i]: Sum of even-indexed from i+1 to n-1

// sum-even = Sum of even elements on left side of i + Sum of odd elements on right side of i

sum-even = pfEven[i-1] + pfOdd[n-1] - pfOdd[i]

pfEven[i-1]: Sum of even indexed from 0 to i-1

pfOdd[n-1] - pfOdd[i]: Sum of odd.-indexed from i+1 to n-1

```
    }

    if (sum_odd == sum_even)
    {
        // It is a special index.
        cnt++
    }
}
return cnt;
```

$O(Q+N)$

size of array.

Q and N are independent:

$1 <= N <= 10^5$

but Q

$1 <= Q <= 10^{15}$