Good Evening Everyone :)

STOCK PORTFOLIO Performance TRACKING

Tracking the performance of stocks over time is crucial for making informed decisions. You want to develop a feature for a banking app that allows users to quickly assess their stock portfolio's profit or loss over specified periods. To efficiently calculate the total profit or loss over any given range of time, you decide to implement this feature using the prefix sum technique.

Example

	0	1	2	3	4	5	6	7	8	9
Stock_Prices[] = [	-5	10	20	40	50	-10	80	-90	-20	-10]

Queries

Start Day	End Day	Net Stock Price
0	9	65
1	4	120
0	0	-5
7	9	-120
2	7	90

Given N elements and Q queries. For each query, calculate sum of all elements from L to R [0 based index]

	0	1	2	3	4	5	6	7	8	9
A[] =	-3	6	2	4	5	2	8	-9	3	1

Queries

L	R	Solution
4	8	9
3	7	10
1	3	12
0	4	14
7	7	-9

Queries

0	4	8
1	3	7
2	1	3
3	0	4

Brute Force.

Size of array = n
queries size = Q.

```
Function querySum ( Queries[][], Array[],
                    querysize, size) {

    for (i -> 0 to Queries.length - 1) {

        L = Queries[i][0];
        R = Queries[i][1];

        Sum = 0
        for (j -> L to R) {

            sum = sum + Array[j]

        }

        print (sum)

    }

}
```

TC: O(NQ)

SC: O(1)

Quiz 1.

10 overs Scores. [7th over]

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

Score before => 49

Score after 7 => 65

Score in the 7th over => 65 - 49

= 16.

Quiz 2 10 overs Score (6th to 10th over)

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

before 6th score [5]

after 10th score [10]

score [10] - score [5]

97 - 31 => 66

Quiz 3.

10 overs Score [10th over]

[2 8 14 29 31 49 65 79 88 97]

before → score [9]
after → score [10]

97 - 88 = 9.

Quiz 4.

10 overs Score [3rd to 6th over]

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

score [6] - score [2]

49 - 8 ⇒ 41

Quiz 2.

10 overs Score

[4th to 9th Over]

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

lower Upper.

88 - 14 ⇒ 74.

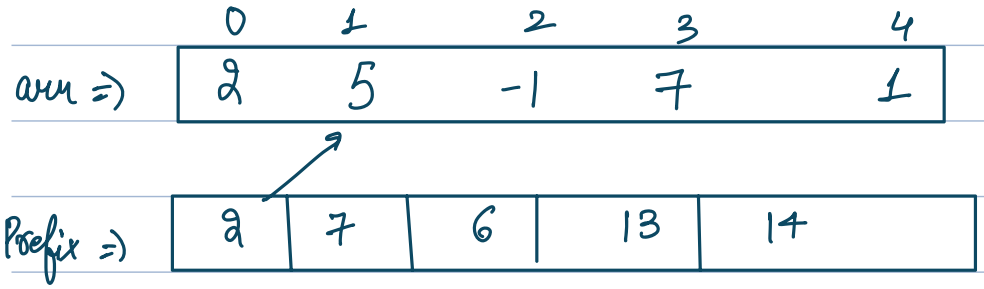
Score [R] - Score [l-1]

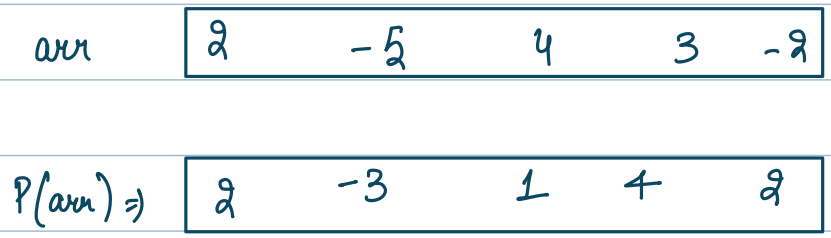
Observations

↳ Answer queries in O(1) time when I have cumulative Sum.

↳ We need to create a cumulative / Prefix Sum Array to answer the question within a range.

How to create Prefix Sum Array





Quiz6:

Calculate prefix sum array:

	0	1	2	3	4	5
ar	10	32	6	12	20	1
pf[0]	10	42	48	60	80	81

Code:

```
pf[N]
for (i -> 0 to N-1)
{
    sum = 0
    for (j -> 0 to i)
    {
        sum = sum + A[j]
    }
    pf[i] = sum
}
```

$O(n^2)$

pf(0) => A[0]
pf(1) = A[0] + A[1]
pf(2) = A[0] + A[1] + A[2]

pf(0) ⇒ A[0]

pf(1) ⇒ pf[0] + A[1]

pf[2] ⇒ pf[1] + A[2]

$$pf[i] = pf[i-1] + A[i]$$

Optimised code:

TC: O(N)

```
pf[N]
pf[0] = A[0]
for (i → 1 to N-1)
{
    pf[i] = pf[i-1] + A[i]
}
```

How to answer the Queries

	1	2	3	4	5	6	7	8	9
A[] =	-3	6	2	4	5	8	8	-9	3	1

pf[] =	-3	3	5	9	14	16	24	15	18	19

L	R	
4	8	pf[8] - pf[3] ⇒ 15 - 5 = 10
3	7	pf[7] - pf[2] ⇒ 24 - 3 ⇒ 21
1	3	pf[3] - pf[0] ⇒ 5.
7	7	pf[7] - pf[6] ⇒ 8

Generalised

$$Sum[L\ R] = pf[R] - pf[L-1]$$

If $L == 1 \Rightarrow Sum[1\ R] = pf[R]$ (1 based indexing)

If $L == 0 \Rightarrow Sum[L\ R] = pf[R]$ (0 based indexing)

	0	1	2	3	4
ar	1	2	-4	3	7

pf[ ]	1	3	-1	2	9

L	R
1	3

pf[2] - pf[0] ⇒

2 - 1 = 1

0	2

pf[R] ⇒

pf[2] ⇒ -1

Code:

```
Function querySum (Queries[][], Array[], querySize, size) {

    pf[N]
    pf[0] = A[0]
    for (i -> 1 to N-1) {
        pf[i] = pf[i-1] + A[i]
    }

    // answer queries

    for (i -> 0 to Queries.length-1) {
        L = Queries[i][0]
        R = Queries[i][1]

        if (L==0) {
            sum = pf[R]
        }
        else{
            sum = pf[R] - pf[L-1]
        }
        print(sum)
    }
}
```

TC : O(N+Q)

SC : O(N)

Sum of Even indexed Elements

Given an array of size N and Q queries with start (s) and (e) index. For every query, return the sum of all even index elements from s to e (0 based index).

	0	1	2	3	4	5
A[] = {	2	3	1	6	4	5 }

Query

1	3	1
2	5	5
0	4	7
3	3	0

Brute force O(NQ).

	0	1	2	3	4	5
A[] ⇒ [	2	3	1	6	4	5]
pf[] ⇒ [	2	2	3	3	7	7]

if i is even

pf[i] ⇒ pf[i-1] + A[i]

else

pf[i] ⇒ pf[i-1].

Quiz 7.

0	1	2	3	4
2	4	3	1	5

pf[]

2	2	5	5	10

Queries

0 [2 4]

1 [0 2]

0	1	2	3	4
2	4	3	1	5

pf[]

2	2	5	5	10

Code:

Function SumOfEvenIndexed (Array[], Queries, N, Q) {

Pse [N]

TC: O(N+Q)

SC: O(N)

Pse [0] = Array [0]

```
for (i -> 1 to N) {
    if (i%2 == 0) {
        Pse [i] = Pse [i-1] + Array [i]
    }
    else
        Pse [i] = Pse [i-1]
    }
}

for (i -> 0 to Queries.length-1)
    s = Queries [i] [0]      // 2   116
    e = Queries [i] [1]      // 0   112

    if (s == 0)
        print (pse [e])
```

}

}

else

print (pse[e] - pse[s-1])

}

}

PROBLEM 3 :- COUNT OF PAIRS ag

given a string s lowercase characters, return the count of pairs (i,j), such that i<j and

s[i] is 'a'
s[j] is 'g'

String s = "abegag"

2 1 = 3.

Quiz 6. s = "acgdgag"

[0 2] [5 6]
[0,4]
[0,6] ⇒ 4.

Quiz 9.

$S = \text{"}\overset{0}{b}\overset{1}{c}\overset{2}{a}\overset{3}{g}\overset{4}{g}\overset{5}{a}\overset{6}{a}\overset{7}{g}\text{"}$

[2 3]
[2 4]
[2 7]
[5 7]
[6 7]

⇒ 5 pairs.

Brute Force

TC : $O(N^2)$
SC : $O(1)$.

```
function count_ag (str)
result = 0
for (i → 0 to n-1) {
    if (str[i] == 'a') {
        for (j → i+1 to N-1)
            if (str[j] == 'g') {
                result = result + 1
            }
    }
}
```

ans => 2
+3
=> 5
+3
=> 8

	0	1	2	3	4	5	6	7	8
String	a	c	b	a	g	k	a	g	g

count_a => 2

ans = 2 + 3 = 5 + 3 = 8

```
function count_ag (str) {
    result = 0
    count_a = 0
    for (i -> 0 to n-1) {
        if (str[i] == 'a') {
            count_a++;
        }
        else if (str[i] == 'g') {
            result = result + count_a
        }
    }
    return result
}
```

TC : O(N)

SC : O(1)

Introduction to Subarrays

A subarray is a contiguous part of an array. It is formed by selecting a range of elements from the array. A subarray can have one or more elements and must be contiguous part of original array.

{ 4 1 2 3 -1 6 9 8 12 }

✓ { 2 3 -1 6 } → is a subarray of length 4

✓ { 9 }

✓ { 4 1 2 3 -1 6 9 8 12 }

✗ { 4 12 }

✗ { 1 2 6 } not continuous.

✗ { 3 2 1 4 } order has to same and it has to continuous.

Quiz:

A[] = { 2 4 1 6 -3 7 8 4 }

✓ 1 6 8 ✗ not continuous

✓ 1 4 ✗ order

✓ 6 1 4 2 ✗ order and place

✓ 7 8 4 ✓ Subarray.

Representation of a Subarray

(1) By specifying start index → end index

(2) By specifying start and length of subarray.

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

{ 2 3 -1 6 }

s = 2 e = 5 length = 4.

Quiz 11:

How many subarrays start from index 0

0	1	2	3	4	5	6
4	2	10	3	12	-2	15

[4]
[4 2]
[4 2 10]
[4 2 10 3], [4 2 10 3 12]

[4 2 10 3 12 -2]
[4 2 10 3 12 -2 15]

⇒ 7.

Quiz 12:

How many subarrays start from index 1

0	1	2	3	4	5	6
4	2	10	3	12	-2	15

[2] ⇒ 1 → 1
[2 10] ⇒ 1 → 2
[2 10 3] ⇒ 1 → 3
[2 10 3 12] 1 → 4
[2 10 3 12 -2] 1 → 5
[2 10 3 12 -2 15] 1 - 6

PRINT ALL POSSIBLE SUBARRAYS OF ARRAY

Given an array of integers, print all possible subarrays

Input: [1 2 3]

(0 1 2)

s	e
(0,0)	[1]
(0,1)	[1 2]
(0,2)	[1 2 3]
(1,1)	[2]
(1,2)	[2 3]
(2,2)	[3]

(0 1 2)

[1 2 3]

		start	end	
i = 0	j → 0 to 2	0	0	[1]
	j → 1	0	1	[1 2]
	j → 2	0	2	[1 2 3]
i = 1	j → 1 to 2	1	1	[2]
	j → 2	1	2	[2 3]
i = 2	j → 2 to 2	2	2	[3]

Code:

```
function printSubarrays (arr[], n) {
    for (i → 0 to n-1) {
        for (j → i to n-1) {
            start = i
            end = j
            for (k → start to end)
                print arr[k]
        }
    }
}
```

TC: $O(n^3)$

SC: $O(1)$.