1Q) Given N array elements & Q queries, for each query calculate sum of all even index in given range

// Note: All Queries on Same Array

	0	1	2	3	4	5	6	7
ar[8] :	3	4	-2	8	6	2	1	3

4 Queries

	Sum
[2 5] :	4
[3 7] :	7
[0 7] :	8
[7 7] :	0
[4 4] :	6

For every query iterate and only add even index.

```
sum = 0
i = L; i <= R; i++){
    if(i%2==0){
        sum = sum + ar[i]
    }
}
return ans;
```

TC: For single Query is O(N)

For Q queries

TC: Q * N

SC: O(1)

// Optimization: 1

: increment i by 2 : (N/2) : O(N)

Iterations

// Optimization: 2

// pf[i] = { // Sum of all even index from [0-i] }

Ex 1:

	0	1	2	3	4	5	6	7
ar[8] :	3	4	-2	8	6	2	1	3
Pf[8] :	3	3	1	1	7	7	8	8

Ex 2:

	[0]	1	2	3	4	5
ar[6] :	2	4	-1	3	9	4
Pf[6] :	2	2	1	1	10	10

// Pseudocode

```
Pfeven[N];
Pfeven[0] = arr[0]
i = 1 ; i < N; i++) {
    if ( i % 2 == 1) {
        Pfeven[i] = Pfeven[i-1
    }
    else { // i is even
        Pfeven[i] = Pfeven[i-1] + ar[i]
    }                  -> i == 0 // error
}
```

-> TC : O(N) SC : O(N)

Answering Queries:

[4 8] :

0 1 2 3 4 5 6 7 8

Pf[8] = Sum of all even indices from 0-8

= Sum of all even indi from [0-3] → Pf[3]

Sum of all even indi from [4-8]

Pf[8] = Pf[3] + Sum of all even indices from [4-8]

[4-8] = Pf[8] - Pf[3]

// Query [L R] get sum of all even indices

sumeven[L-R] = Pf[R] - Pf[L-1] } if L==0 ✗

Given L & R

```
if (L==0){
   sumeven[L-R] = Pf[R]
}
else{
   sumeven[L-R] = Pf[R] - Pf[L-1]
}
```

// For Each Query

O(1)

TC: (N + Q)

SC: O(N)

20) Given N array elements & Q queries, for each query calculate sum of all odd index in given range

TODO :

Pfodd[i] = sum of all odd index elements from [0 - i]

Ex:

	0	1	2	3	4	5
ar[6] :	-3	-1	6	2	4	5
Pfodd[6] :	0	-1	-1	1	1	6

	0	1	2	3	4
ar[5] :	2	4	3	1	5
Pfodd[5] :	0	4	4	5	5

Delete index

Ex1:

	0	1	2	3	4	5	6
				~~odd~~	even	odd	even
ar[7] ⇒	3	-2	~~6~~	8	9	10	4

// Delete 2nd index

ar[6] ⇒	3	-2	8	9	10	4
			even	odd	even	odd

Ex2:

	0	1	2	3	4	5	6	7
ar[8] ⇒	2	3	9	6	4	-2	1	8

// Delete 3rd index

ar[7] ⇒	2	3	9	4	-2	1	8

Ex3:

	0	1	2	3	4	5
ar[6] ⇒	-1	6	3	8	0	7

// Delete 0th index

ar[5] ⇒	6	3	8	0	7

Observations:

All Elements after deleted index after shifting to the leftside by 1 step

Special Index { Above Medium }

An index is said to be special, if after deleting the element Sum of All Even Indices = Sum of all odd indices

Count how many special index are there?

$\rightarrow$ Sum even = 12

Ex1:

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

Sum odd = 8 $\Rightarrow$ 1

Check index = 0?

Delete index 0

0	1	2	3	4
3	2	7	6	-2

Sum of all even : 8

Sum of all odd : 8

Index 0 is special

Check index = 1?

Delete index 1

0	1	2	3	4
4	2	7	6	-2

Sum of all even : 9

Sum of all odd : 8

Index 1 ✗

Check index = 2?

Delete index 2

0	1	2	3	4
4	3	7	6	-2

Sum of all Even : 9

Sum of all Odd : 9

Index 2 is special ✓

$\rightarrow$ index = 3 ✗

$\rightarrow$ index = 4 ✗

$\rightarrow$ index = 5 ✗

// 2 indices are special

0 & 2

Approach : 1

Check if every index is special or Not ?

: Delete that index

get sum of even = sum of odd

TC : $N * \{ N + N \}$

N checks

To delete a index

Sum of Even Index & Sum of odd Index

TC : $O(N^2)$

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

sumeven = 12

sumodd = 8

sumodd = 8 - 7 = 1

Delete index 3

0	1	2	3	4
4	3	2	6	-2

sumeven = 4

sumodd = 9

Eg =

ar[14]:

0 1 2 3 4 5 6 7 8 9 10 11 12 13

sumeven[0 4] + sumodd [6 13] = sumeven

sumodd [0 4] + sumeven [6 13] = sumodd

0 1 2 3 4 5 6 7 8 9 10 11 12 13

sumeven[0 7]

sumodd [0 7]

sumeven [9 13] = sumodd

sumodd [9 13] = sumeven

Even Sum:

sumeven[0 7] + sumodd [9 13]

Odd Sum

sumodd [0 7] + sumeven [9 13]

if(Evensum == OddSum)

// Generalize ean

// i^{th} index special a Not?

0   1   2   3   ..   i-1   i   i+1   - -   N-1

Sum even [0, i-1]   Sum even [i+1, N-1] = oddsum

Sum odd [0, i-1]   Sum odd [i+1, N-1] → evensum

// → Construct pfeven[] → O(N)
// → Construct pfodd[] → O(N)

i = 1; i < N-1; i++

if i == 0 Edge case

EvenSum = Sumeven [0, i-1] + Sumodd [i+1, N-1]

OddSum = Sumodd [0, i-1] + Sumeven [i+1, N-1]

if i == N-1

if (EvenSum == OddSum) {
    cnt++;
}

}

// Check if 0^{th} index special a Not
// Check if $N-1^{th}$ index special a Not

10:45 break

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

char [ ] : N

208) Given a String, calculate Number of pairs

i, j such that i < j && S[i] = 'a' && S[j] = 'g'

// All characters are lower case english Alphabets

Ex1:

0	1	2	3	4	5	6	7
b	a	a	g	d	c	a	g

Pairs: {1 3}, {2, 3}

{1 7}, {2, 7}

{6, 7} ✓

Ex2:

0	1	2	3	4	5	6
a	c	g	d	g	a	g

Pairs {0, 2}, {0, 4}, {0, 6}

{5, 6}

Solution

1) Check all pairs } BF

```
i = 0; i < N; i++) {
    j = i+1 ; j < N; j++) {
        if (S[i] == 'a' &&
            S[j] == 'g') {
            cnt++
        }
    }
}
return cnt;
```

```
i = 0; i < N; i++){        BF2
    if(s[i] == 'a'){        }if s[i] == 'a'
        j = i+1; j < N; j++){   => {Number of g's on right side}
            if(s[j] == 'g'){
            }  cnt++
        }
    }
}
```

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

Eg:

0	1	2	3	4	5	6	7	8
a	d	g	a	g	a	g	f	g
ans+=2, ans=9	c=4	c++, c=4	ans+=, c=, ans=5	c++	ans+=c, ans=2	c++, c=2	c=1	c++, c=1

ans=0
c = 0

c=0, ans=0

```
i = N-1; i >= 0; i--){
    if(ar[i] == 'g'){ c++ }
    else if(ar[i] == 'a'){ ans += c }
}
return ans;
```

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

// For every 'g' we need no: of a's in leftside

```
ans = 0, c = 0

i = 0; i < N; i++){
    if( s[i] == 'a'){
        c++
    }
    else if (s[i] == 'g'){
        // Number of a's to the left of g
        ans += c;
    }
}
return ans;
```

TC: O(N)

SC: O(1)

Closest Min Max

→ Continuous part of array
→

Given an array find the {length of smallest subarray} which contains Both Min & Max of array

$\frac{(N)(N+1)}{2}$

Ex:

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

Min : 1
Max : 6

[3 6] : 4

Ex1:

0	1	2	3	4	5	6	7	8	9	10
2	2	6	4	5	1	5	2	6	4	1

Min : 1
Max : 6

[2 10] : len = 8
[2 5] : len = 4
[8 10] : len = 3

Ex2:

0	1	2	3	4	5	6	7	8	9	10	11
1	6	4	2	7	7	5	1	3	1	1	5

Max : 7
Min : 1

[5 7] len = 3

Ex4: 8 8 8 8 8

Max : 8
Min : 8

len = 1

Final Subarray

obs1: Our Subarray should only contain single min & single max

Min   Min   Max | Max   Min   Max

obs2: Min & Max should be on corners

[Min   Max]   [Max   Min]

Subarray:
- [Min   Max]
- [Max   Min]

Max = 6, Min = 1

Eg1:

0	1	2	3	4	5	6	7	8	9	10
2	2	6	4	5	1	5	2	6	4	1
X	X	[2 5]	X	X	[5 8]	X	X	[8 10]	X	

len = 4   len = 4   len = 3

```
// Calculate min_val & max_val, ans = N
i = 0; i < N; i++) {
    if (ar[i] == min_val) {
        // Iterate in right to get 1st max index
        j = i+1; j < N; j++) {
            if (ar[j] == max_val) { break }
        }
        // subarray [i - j]
        ans = min (ans, j-i+1)
    }
    else if (ar[i] == max_val) {
        // Iterate in right to get 1st min index
        j = i+1; j < N; j++) {
            if (ar[j] == min_val) { break }
        }
        // Subarray [i j]
        ans = min (ans, j-i+1)
    }
}
```

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

// iterate & from R-L & get Min & Max Index

ans = N

0	1	2	3	4	5	6	7	8	9	10
2	2	6	4	5	1	5	2	6	4	i
X	X		X	X		X	X		X	

minI = -1

maxI = -1

minI = 10

maxI = -1

maxI = 8

minI = 10

len = 3

ans = 3

minI = 5

maxI = 8

len = 4

ans = 3

maxI = 2

minI = 5

len = 4

ans = 3

Pseudocode

Step 1: Calculate Min Value & Max value

Step 2: ans = N, MinI = -1, & MaxI = -1

if (min_value == Max_value) return 1

```
i = N-1; i >= 0; i--) {
    if (ar[i] == Min_value){
        MinI = i; // Get 1st max on right
        if (MaxI != -1){ [Min  Max]
            ans = min (ans, MaxI - MinI + 1) } }
        }
    }
    else if (ar[i] == Max_Value){ => [Max  Min]
        MaxI = i; // Get 1st min on right
        if (MinI != -1){
            ans = min (ans, MinI - MaxI + 1)
        }
    }
}
return ans;
```

Doubts

	0	1	2	3	4	5
Eg 1:	4	3	2	7	6	-2
pfeven[]:	4	4	6	6	12	12
pfodd[]:	0	3	3	10	10	8

3 is special

left:

Sumeven [0 2]

pfeven [2]

odd even → [0 2]

Sumodd [0 2]

Pfodd [2]

odd → [0 2]

Right

sumeven [4 5]

Pfodd [5] − Pfeven [3]

even [0 5] = even [0 3]

→ Sumeven → [4 5]

Sumodd [4 5]

	0	1	2	3	4	5	6	7
ar[]:	3	2	4	5	8	9	10	11

// pf[i] = pf[i-1] + ar[i]

pf[i+1] = pf[i] + ar[i+1]

	0	1	2	3	4	5	6
ar[] :	3	4	2	1	6	8	9
pf[] :	3	7	9	10	16	24	33

pf[5] = 24
pf[3] = 10

pf[5] - pf[3] = 14

// Sum of all elem [5 4]

ar[5] + ar[4] + pf(3) - pf(3)

ar[3] x

pf(5) = pf(4) + ar[5]

pf(4) = pf(3) + ar[4]

pf(3) = pf(2) + ar(2)

Sum [i+1, - N-1)

Pfeven [N-1] - Pfeven [i]

Sum of All Even 0, N-1 - Sum of all Even 0, i

→ Sum of All Even i+1, - N-1 → i