Basic Array Questions ⇒

Q1 ⇒ Find sum of all array elements ⇒

arr = [8 , 4 , 2 , 3 , 1] n = 5

For Loop ⇒ Where I know how many times it will run

while Loop ⇒ Left < Right
{

}

[8 , 4 , 2 , 3 , 1]

int sum = 0 → 8 → 12 → 14 → 17 → 18 ⇒ 18

Pseudo Code :

```
int sum ( arr )
{
    int result = 0
    for ( i -> 0 -> n - 1 ){
        result = result + arr[i]
    }
    return result
}
```

n = size of array

TC -> O(N) -> N is size of array

SC -> O(1)

Q2 -> Find count of odd numbers in an array

arr = [4, 5, 8, 9, 6]

int count

0 + 2 => 2

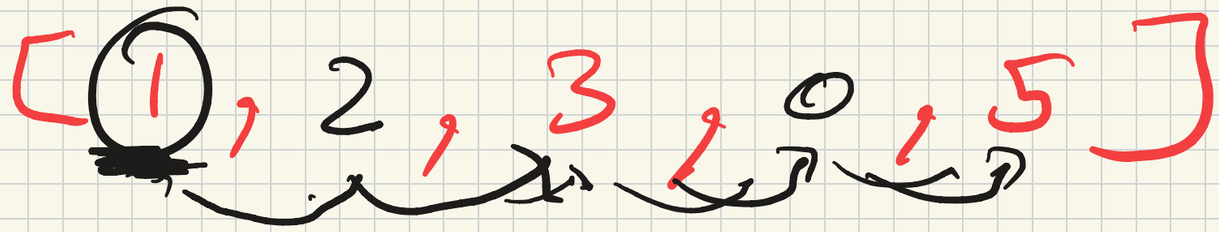
Pseudo Code :)

```
int CountOddNumbers (arr) {
    int count = 0
    for (let i = 0 ; i < n-1) {
        if (arr[i] % 2 === 1) {
            count ++
        }
    }
    return count
}
```

3

Q3 :) Check if Array is sorted or not (In Ascending order)

arr = [1, 2, 3, 9, 5]




```
boolean isArraySorted (arr)
  {
      for (i → 0 → n-1) {
          for (j → i+1 → n-1) {
              if (arr[j] < arr[i]) {
                  return false;
              }
          }
      }
      return true;
  }
```

TC → $O(N^2)$

[1, 2, 3, 4, 5]

SC → $O(1)$

N is size of array

N → 5

N^2 → 25

1 * 4 → 5

1 + 3 → 4

1 + 2 → 3

1 + 1 → 2

1 + 0 → 1

N → 15

→ 15

Optimal Solution →

[1, 2, 3, 4, 5] → Sorted

[2, 5, 4, 8, 10] → Unsorted

boolean checkIfArraySorted(arr)

{

for(i=1; i<n-1){

if(arr[i] < arr[i-1]){

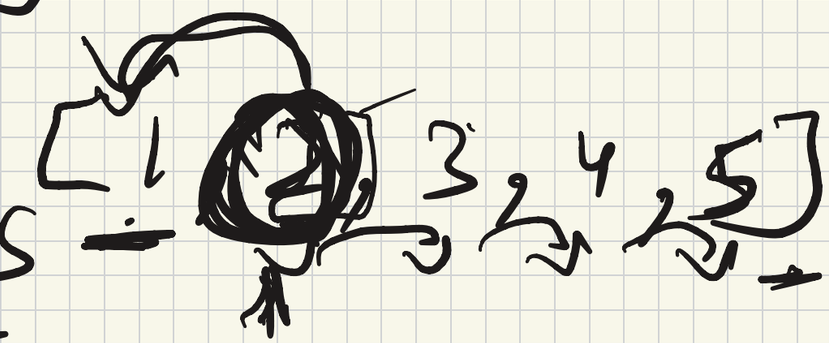
return false

}

}

return true

}



TC => O(N)

SC => O(1)

Q4 → Reverse the given array

arr[] = [1, 2, 3, 4, 5]

⇒ [5, 4, 3, 2, 1]

void reverseArr (arr) {—3

arr[] = [1, 2, 3, 4, 5] n → 5

temp[] = [5, 4, 3, 2, 1]; n-1-i

[———]

i → n-1-i

5-1-0

arr[0] → temp[4]
arr[1] → temp[3]
arr[2] → temp[2]
arr[3] → temp[1]

arr[4] → temp[0]

arr[] = [1, 2, 3, 4, 5]

temp[] = [_ , _ , _ , _ , _]

void reverse (arr) {

temp[] → [5, 4, 3, 2, 1]

for (i → 0 → n-1) {

temp[n-1-i] = arr[i]

3

for (i → 0 → n-1) {

arr[i] → temp[i]

3

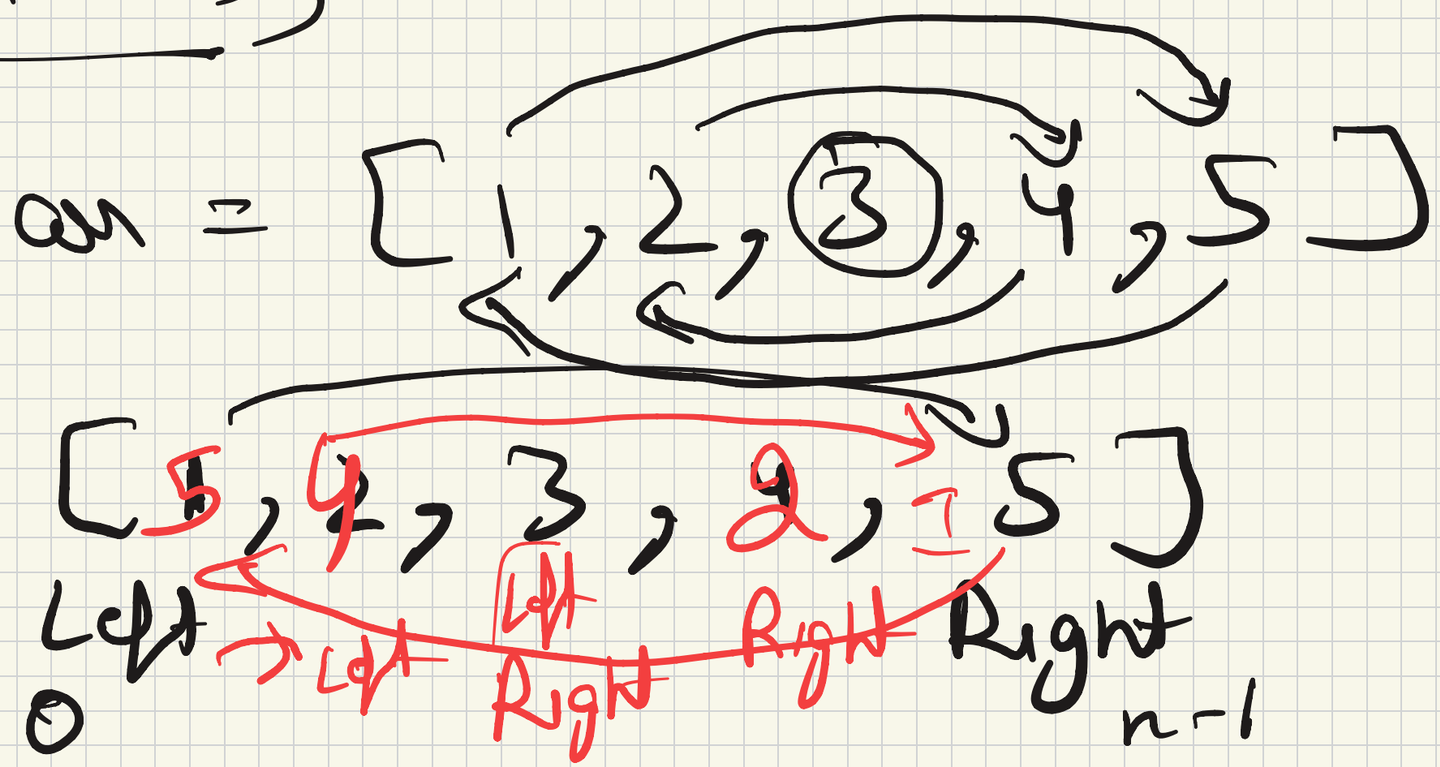
3

TC → O(N) + O(N) → O(2N)

SC → O(N)

→ N → size of array

Optimal Solution

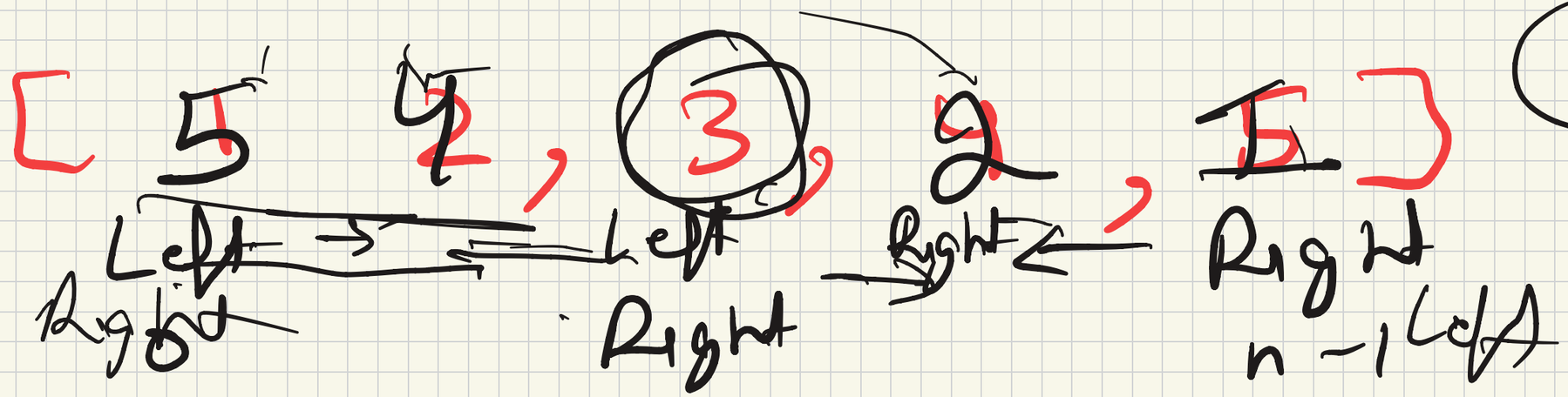



```
while (left < right) {
    const temp = arr[left]
    arr[left] = arr[right]
    arr[right] = temp
```

3

left ++

right --



[5, 4, 3, 2, 5]

TC → O(N/2)

SC → O(1)

Q 5 → Richest Customer Wealth

John → 6

Ram → 6

$$\begin{bmatrix} [1, 2, 3] \\ [3, 2, 1] \end{bmatrix}_{m \times n} \Rightarrow 6$$

I → 6

2 → 10

3 → 8

[[1, 5] [7, 3] [3, 5]] ⇒ 10

int wealth ⇒ ~~0~~ ~~4~~ ~~8~~ 6

Row Col

0 1 2

[~~1~~, 2, ~~3~~]

[~~8~~, ~~6~~, ~~4~~]

[~~4~~, 2, 5]

int Max ⇒ ~~0~~ 18

wealth ⇒ ~~0~~ ~~8~~ ~~11~~ ~~18~~

wealt ⇒ ~~0~~ ~~11~~ ~~13~~ 18

```
int findMaxWealth(arr)
{
    int Max = 0
    for (row i -> 0 -> n-1)
        int wealth = 0
        for (col -> 0 -> arr[row]-1)
            wealth = wealth + arr[row][col]
    }
    max = Math.max(max, wealth)
}
return max
```

T.C -> O(m*n)

S.C -> O(1)