



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

{

3

[8, 4, 2, 3, 1]

int sum = 0 8 ✗ 1 ✗ 3 ✗ 2 ✗ 18 ⇒ 18

Pseudo Code >

int sum (arr)

$n = \text{size}$
of array

{
 int result = 0

 for (i > 0 > n - 1) {

 result = result + arr [i]

}

return result

}

TC $\rightarrow O(N) \Rightarrow N$ is size of array

SC $\rightarrow O(1)$

Q2 > Find count of odd numbers in an array

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

int count

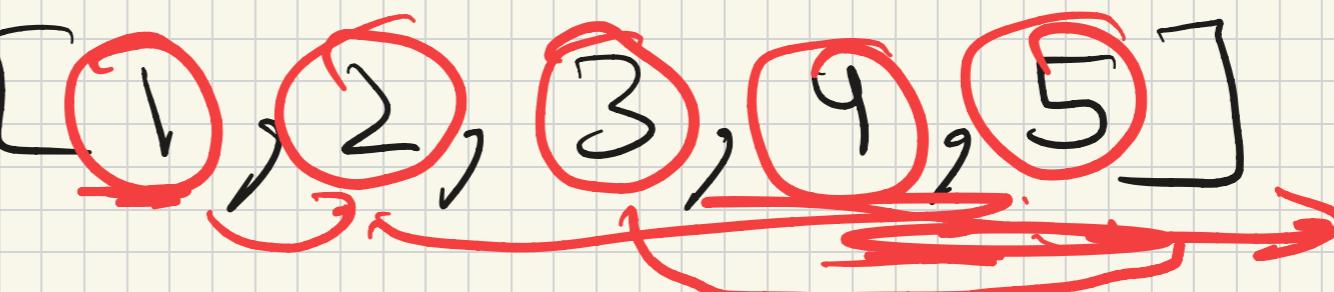
$\emptyset + 2 \Rightarrow 2$

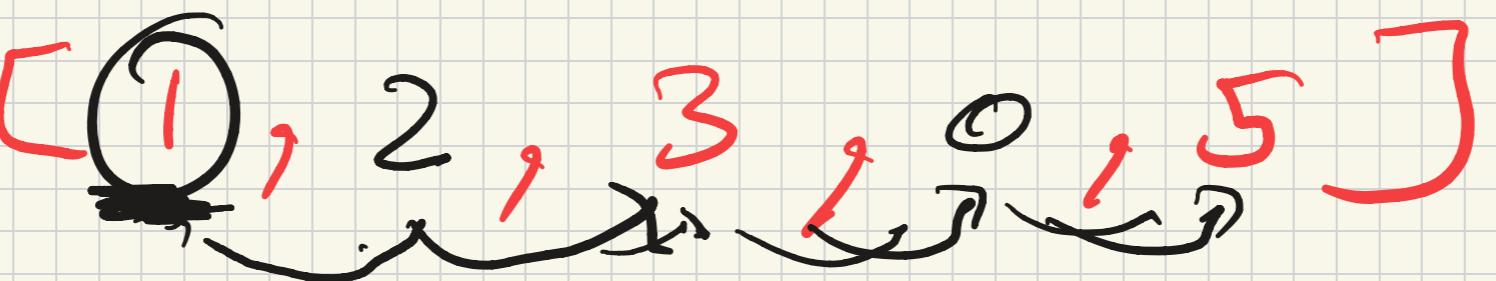
Pseudo Code →

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

Q3 → Check if Array is sorted or not (In Ascending Order)

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





boolean isArraySorted (arr) {

```
for ( i >= 0 >= n - 1 ) {
```

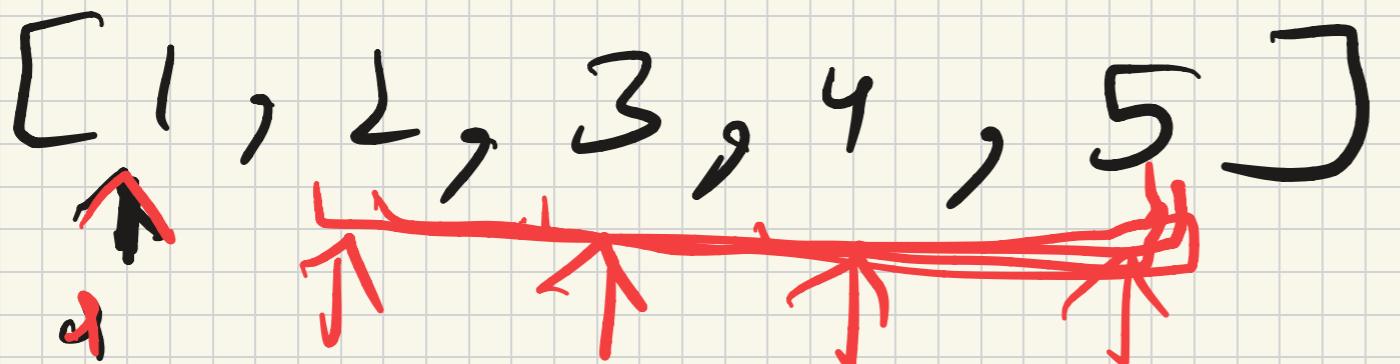
```
    for ( j >= i + 1 >= n - 1 ) {
```

```
        if ( arr[j] < arr[i] ) {
```

~~return false~~

3 3 3
return true;

$TC \rightarrow O(N^2)$



$SC \rightarrow O(1)$

N is size of array

$N \rightarrow 5$

$N^2 \rightarrow 25$

$1+4 \rightarrow 5$

$1+3 \rightarrow 4$

$1+2 \rightarrow 3$

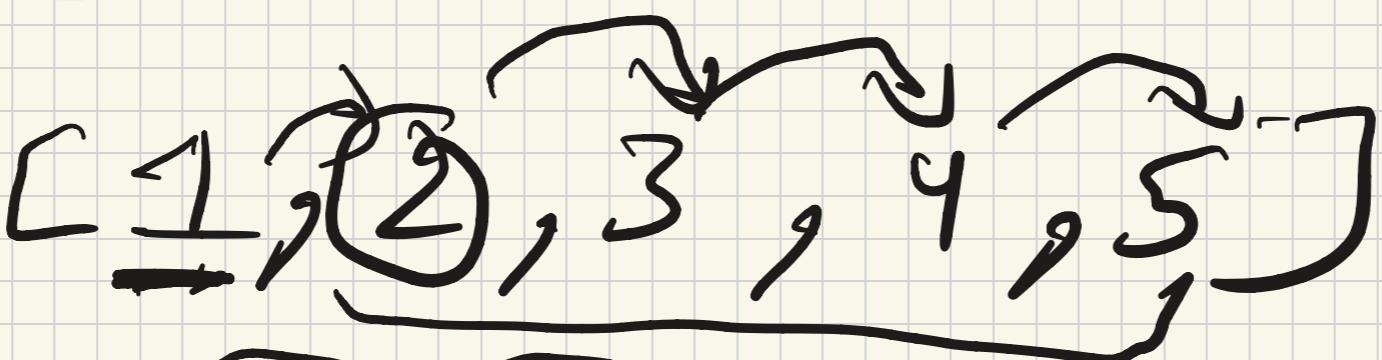
$1+1 \rightarrow 2$

$1+0 \rightarrow 1$

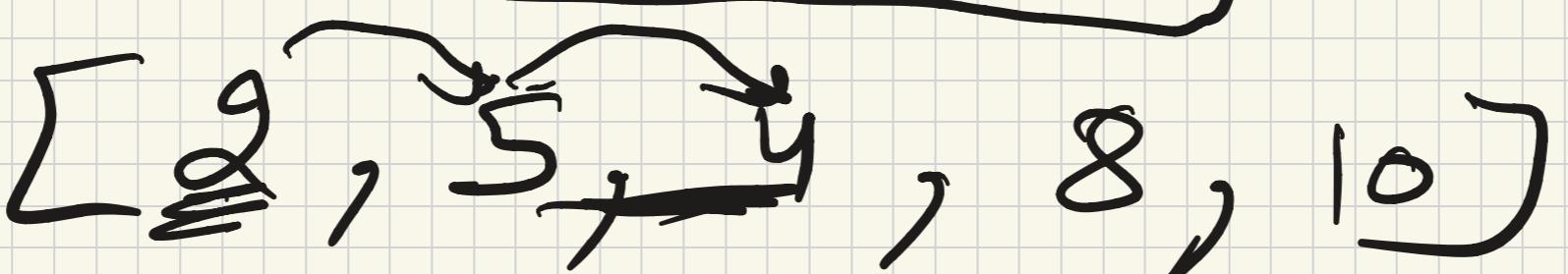
$N \rightarrow 5$

$\rightarrow 15$

Optimal Solution \rightarrow



\rightarrow Sorted

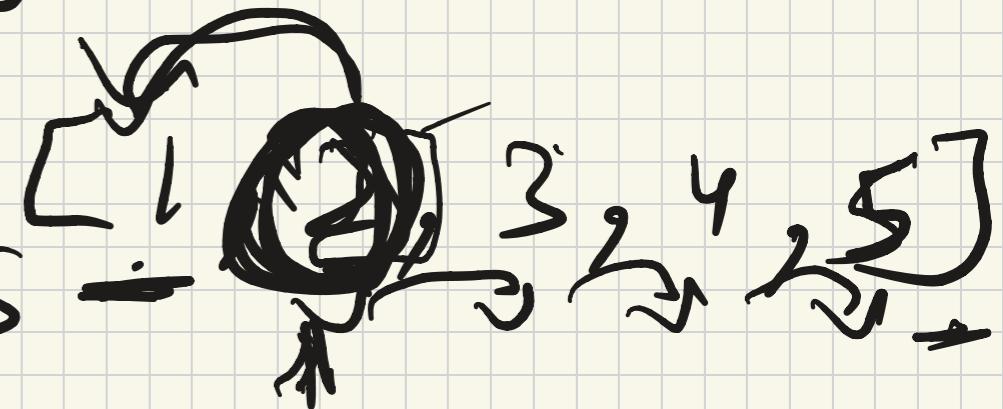


\rightarrow Unsorted

Boolean check If Array Sorted (arr)

{

for ($i \geq 1 \geq n - 1$) {



if ($arr[i] < arr[i - 1]$) {
 return false

 3
 3
 return true

3

TC $\Rightarrow \Theta(N)$

SC $\Rightarrow \Theta(1)$

}

Q9 →

Reverse the given array

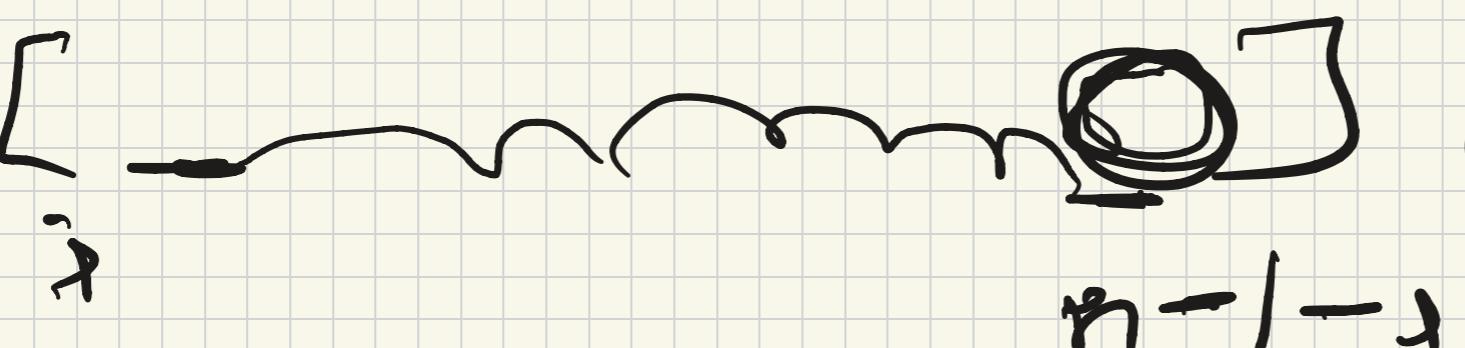
$$arr[] = [\underline{1, 2, 3, 4, 5}]$$

$$\Rightarrow [5, 4, 3, 2, 1]$$

void reverseArr (arr) { }

$$arr[] = [\underline{1, 2, 3, 4, 5}] \quad r \rightarrow 5$$

$$temp[] = [\underline{5}, \underline{4}, \underline{3}, \underline{2}, \underline{1}] \quad n-1-i$$



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

$arr[4] \rightarrow temp[0]$

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

$temp[] = [-, -, -, -, -]$

void reverse (arr) {

temp $\square \rightarrow [5, 4, 3, 2, 1]$

for ($i \geq 0 \rightarrow n-1$) {

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

3

for ($i > 0 \rightarrow n-1$) {

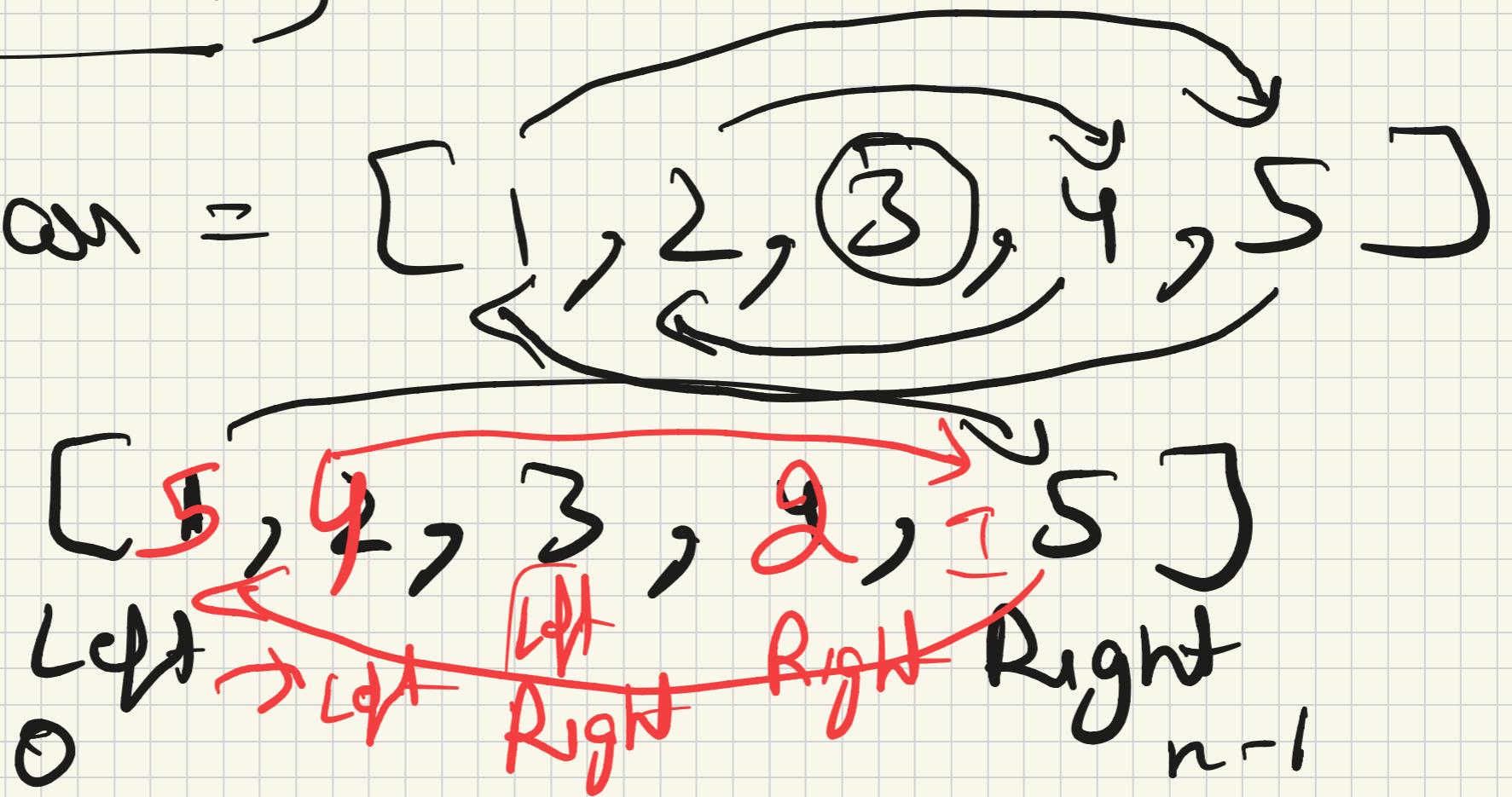
$arr[i] \rightarrow temp[i]$

3

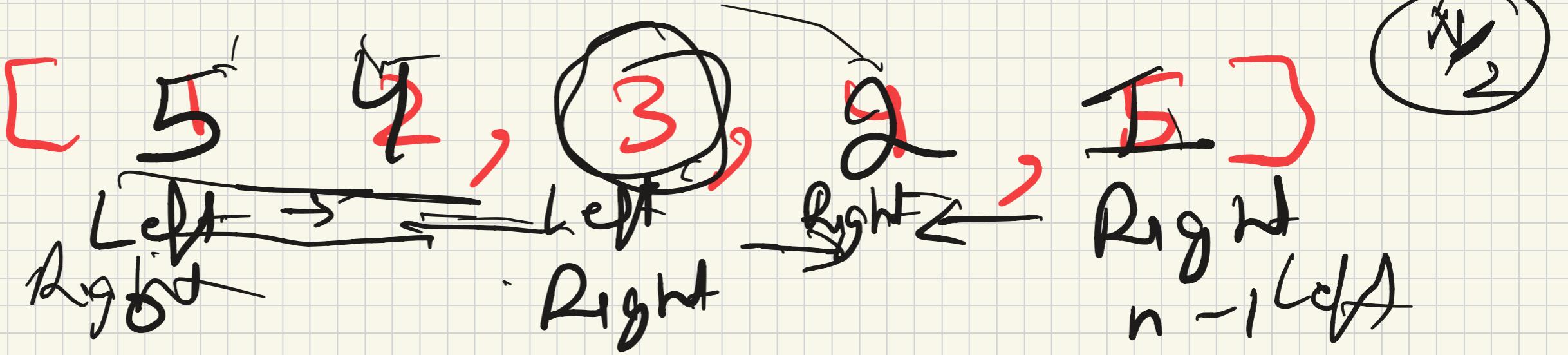
3

$TC \rightarrow O(N) + O(N) \rightarrow O(2N)$
 $SC \rightarrow O(N)$ $\Rightarrow N \rightarrow \text{size of array}$

Optimal Solution



while ($left < right$) {
 cout temp $\geq arr[left]$
 $arr[left] = arr[right]$
 $arr[right] = temp$
 left $\dagger\dagger$
 right $--$
}



$[5, 4, 3, 2, 1]$

$TC \rightarrow O(N)$

$SC \rightarrow O(1)$

Q5 \rightarrow Richest Customer Wealth

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

Ram \Rightarrow 6

A hand-drawn diagram on grid paper. It shows the numbers 1, 2, and 3 arranged vertically on the left, each with a curved arrow pointing towards a circled sum of 10 in the center. A final curved arrow points from the circled 10 to the number 8 on the right.

1 →

2 →

3 →

10

8

$$\left[\begin{array}{c} [1, 5] \\ [7, 3] \\ [3, 5] \end{array} \right] \rightarrow 10$$

Int wealth

The image shows four handwritten digits on a sheet of lined paper. The first digit, '7', is written in black ink. The second digit, '8', is written in red ink. The third digit, '9', is also written in red ink. The fourth digit, '10', is written in red ink and includes a small red circle at the bottom right.

wealth →

Row Cop

A diagram on grid paper illustrating a search space or graph. The nodes are represented by black outlines. The path taken is highlighted in black, starting from node 0 and ending at node 5. Nodes 0, 1, 2, 3, and 4 are crossed out with large red marks, while node 5 is marked with a red checkmark.

Int Max ~~0~~

18

$$w_{\text{calt}} \rightarrow \cancel{\emptyset} \times \cancel{13} = 18$$

int FindMaxWealth (arr)

{
 int Max = 0

 for (now $\rightarrow 0 \rightarrow n - 1$)

 int wealth $\Rightarrow 0$

 for (col $\rightarrow 0 \rightarrow arr[now] - 1$)

 wealth = wealth + arr[now][col]

 3

 max = Math.max(max, wealth)

 3

 return max

3

T $\rightarrow O(m \times n)$
S $\rightarrow O(1)$