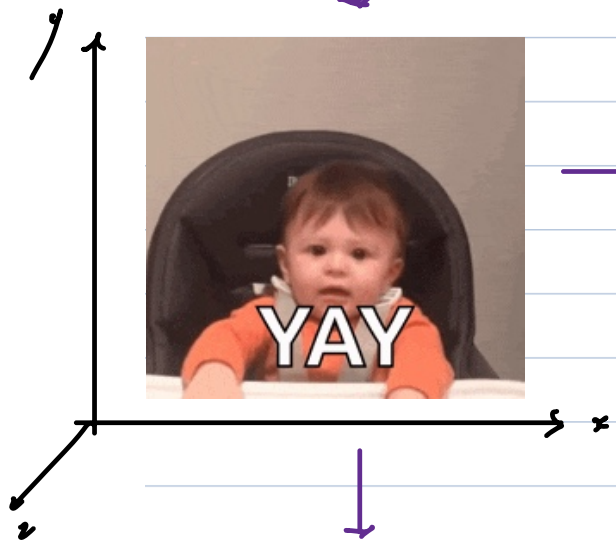
TODAY:

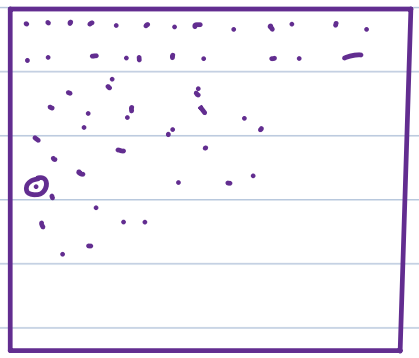
→ flip an image

→ Search in a sorted 2D matrix.

Flipping image

1D 2D





hello

ollleh

image/object is represented usually in the form of matrix.

Reversing the word

This is a image

0 → 1

0.00 0.2 0.5 .4 ... 1

1	1	0
1	0	1
0	0	1

original image

① reverse each row

0	1	1
1	0	1
1	0	0

② Invert input

1	0	0
0	1	0
0	1	1

flipped image

4x4

1	1	0	0
1	0	0	1
0	1	1	1
1	0	1	0

① reverse each row

0	0	1	1
1	0	0	1
1	1	1	0
0	1	0	1

invert input

1	1	0	0
0	1	1	0
0	0	0	1
1	0	1	0

↑ answer.

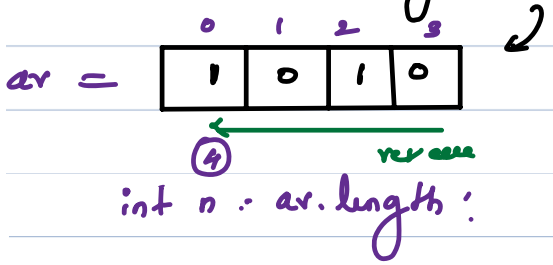
2D array

1	1	0
1	0	1
0	0	1

1D array

1	1	0

i) Reverse an 1D array

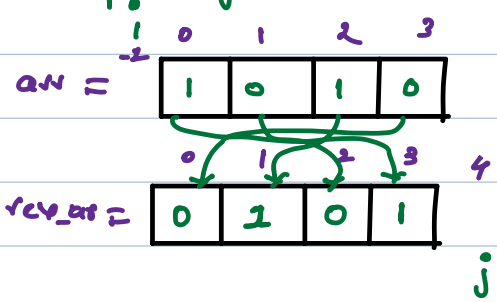


int n = arr.length;

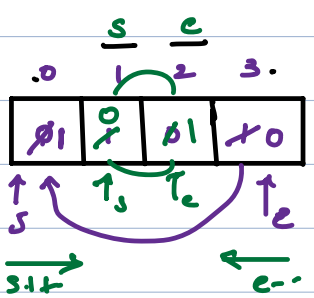
```
int j=0;
int rev_arr = new int[n];
for(int i=n-1; i>=0; i--){
    rev_arr[j] = arr[i];
    j++;
}
```

come out of loop.

1) Loop from back of array



ii) Reverse 1D array without using extra array.



temp=0

s=0 e=1

```
while(s<e){
    swapElements(arr,s,e);
    s++; e--;
}
```

temp = arr[s]

arr[s] = arr[e] → swaping element

arr[e] = temp

swaprow

```
static void swapElement(int[] ar, int s, int e){
    int temp = ar[s];
    ar[s] = ar[e];
    ar[e] = temp;
}
```

→ use swaping method to reverse each row in 2D array.

10:24pm → 10:35pm

→ solve the remaining problem....

for → i • → n-1 swap

for → j • — n-1

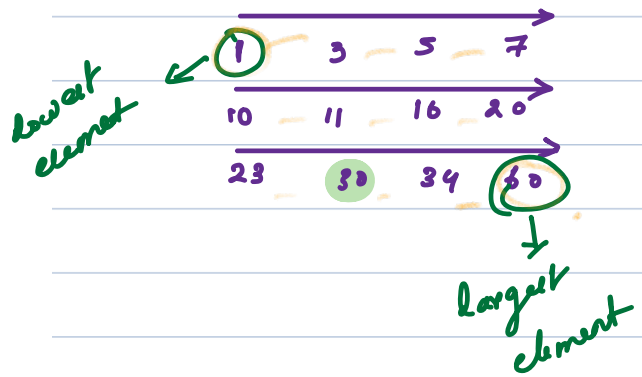
→ row

```
static void swapEleRow(int[][] ar, int i, int s, int e){
    temp = ar[i][s];
    ar[i][s] = ar[i][e];
    ar[i][e] = temp;
}
```

```
static void swapNflip(int[][] ar, int i, int s, int e){
    int temp = 0;
    if(ar[i][s] == 0){
        temp = 1;
    }
    if(ar[i][e] == 0){
        ar[i][s] = 1;
    }else{
        ar[i][s] = 0;
    }
```

```
    }
    arr[i][c] = temp;
}
```

Q) Given a **sorted 2D matrix**, search the given element **k**.



k = 30 → true

k = 37 → false

k = 120 → false.

Brute force solution

Solution :- Iterate over the entire 2D matrix (array and check each element is == **k**.

n → rows

m → cols.

k = 34.

n x m

```
for i 0 ⟶ n-1
    for j 0 ⟶ m-1
        if arr[i][j] == k
            return true;
return false.
```

Exchange sorted

Note:- In above solution we are not taking advantage of the **sorted array**.

100 x 20 = 2000

Case 1) Check if k is with range.

Top left

12 iterations

1	3	5	7
10	11	16	20
23	30	34	60

Bottom right

3x4

k = 34 →

k = 14 →

k = -5

k = -78

k = 89

k = 100.

(1 - 60)

1) if (k < arr[0][0] || k > arr[n-1][m-1]) {
    return false;
}

2)

i
[0][m-1]

	0	1	2	3
0	1	3	5	7
1	10	11	16	20
2	23	30	34	60

34 > 7 skip this row.

34 > 20 skip this row

34 > 60 → no-so → iterate over the row

```
for (int i=0; i<n; i++) {
    if(arr[i][m-1] < k) {
        continue;
    }
    for(int j=0; j<m; j++) {
        if(arr[i][j] == k) {
            return true;
        }
    }
}
return false;
```

(n+m)

3x4

120 → iterations

```
static boolean findk ( int[][] ar , int k ) {
    int n = ar.length;
    int m = ar[0].length;
    if ( k < ar[0][0]  ||  k > ar[n-1][m-1] ) {
        return false;
    }
    for (int i=0; i<n; i++) {
        if(ar[i][m-1] < k ) {
            continue
        }
        for (int j=0; j<m ; j++) {
            if(ar[i][j] == k ) {
                return true;
            }
        }
    }
    return false;
}
```

iterate our whole row