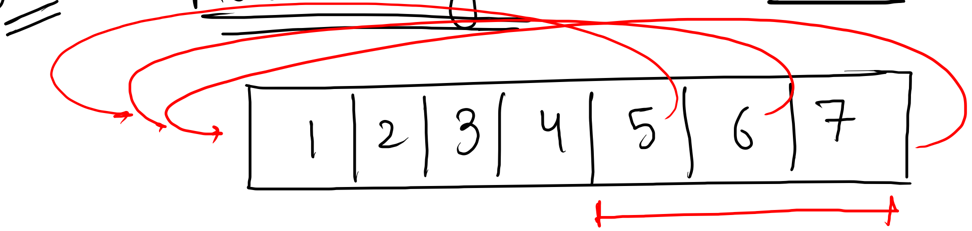
# Topics

- Sorting → lambda $f^n$
- Subarrays → $N^2 * N$
- imp ✓ 2 pointer
- Prefix array →
- Array as HM
- 2D array → transpose
- String
- imp BS → BSUB, BSLB
- AL → inbuilt
- Stack
- HM
- → Queue
- → PQ

Que Rotate Right size = n



|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

$K = 3$

a) $K = K + n$ ✓

b) $K = K \% n$ ✓

Step1 reverse last K elements

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 7 | 6 | 5 |

$(n-K, n-1)$

Step2 reverse remaining element

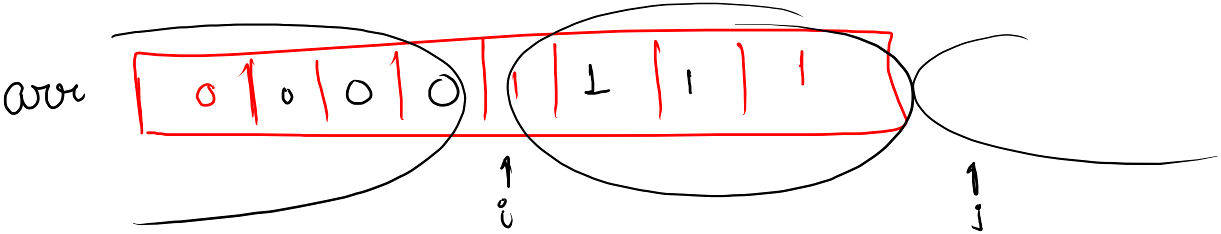
|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| 4 | 3 | 2 | 1 | 7 | 6 | 5 |

$(0, n-k-1)$

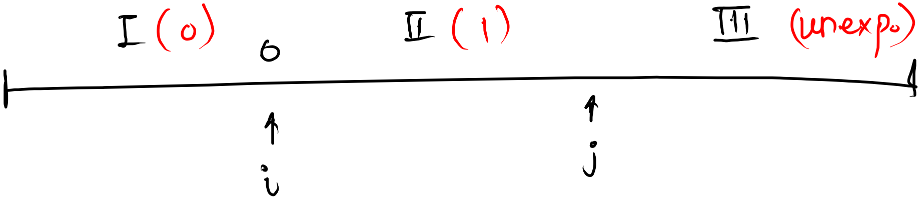
Step3 reverse all elements

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| 5 | 6 | 7 | 1 | 2 | 3 | 4 |

$(0, n-1)$



faith



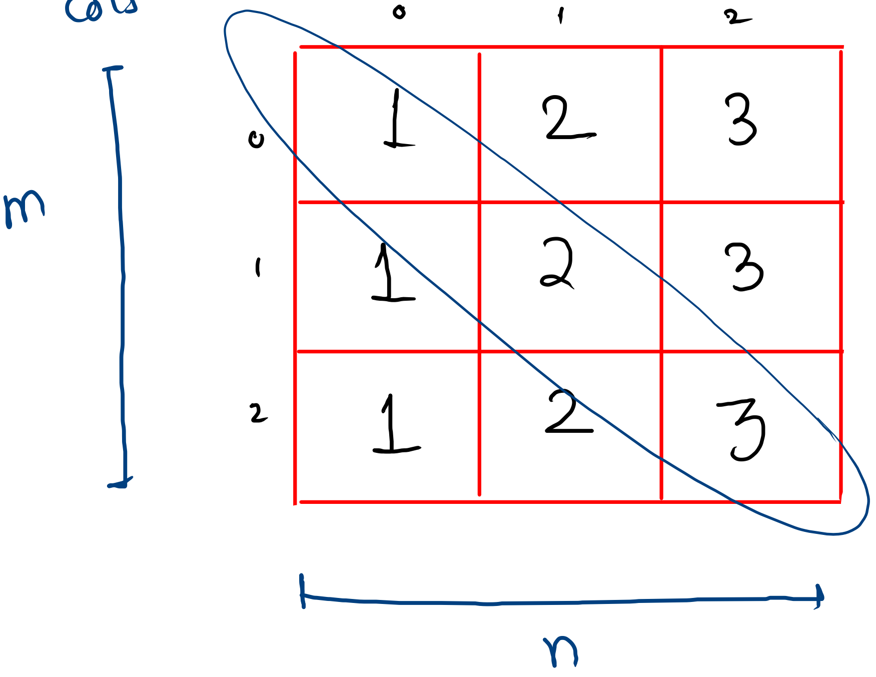
## Ques Transpose ( row → col and col– rows )

|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 2 | 2 | 2 |
| 2 | 3 | 3 | 3 |

|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 1 | 1 | 2 | 3 |
| 2 | 1 | 2 | 3 |

Swap (i, j) index with (j, i)

rows = m
cols = n



$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 1 & 2 & 3 \\ 2 & 1 & 2 & 3 \end{array}$$

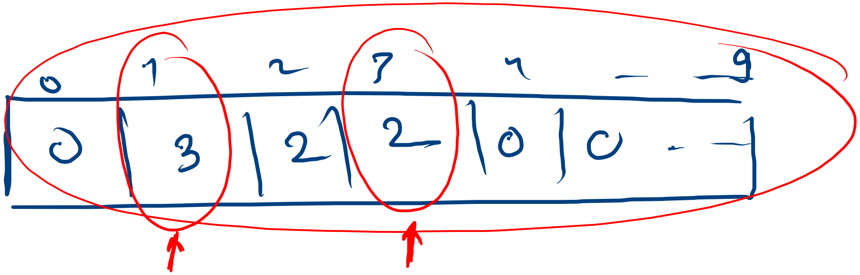
```
for ( int i=0 → m)
    for (j=0 → n)
```

when to swap → ✓ $i <= j$

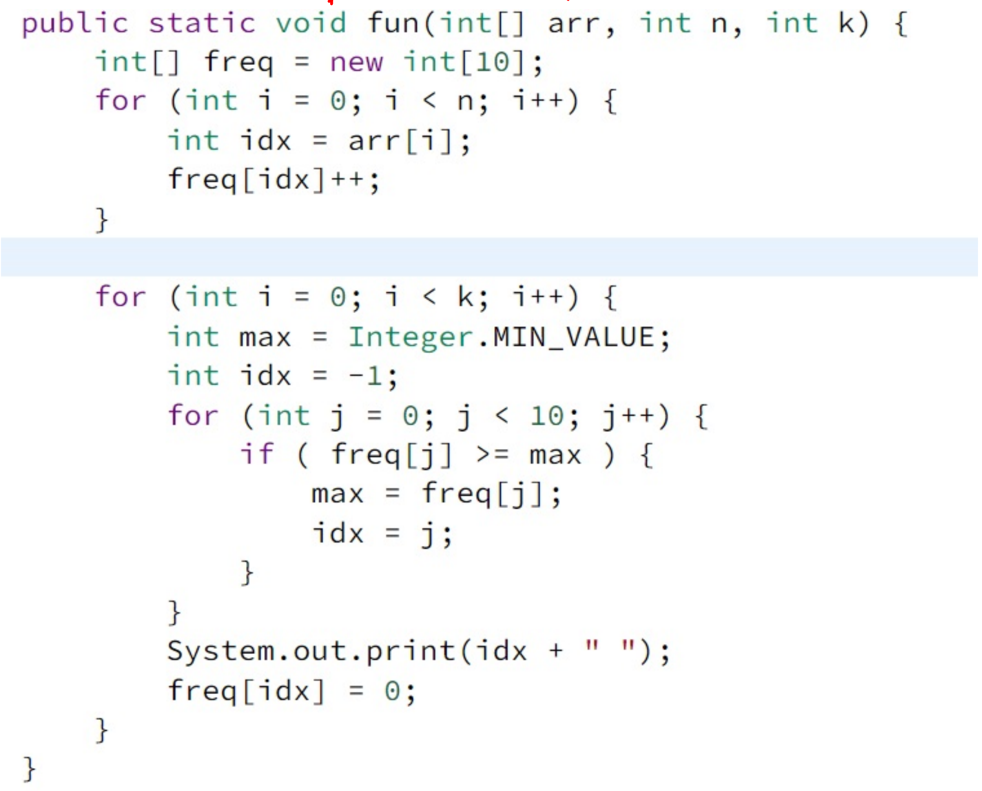
✓ $i < j$

| | | |
|---|---|---|
| (0,0) | → | (0,0) |
| (0,1) | → | (1,0) |
| (0,2) | → | (2,0) |
| (1,0) | → | X |
| (1,1) | → | (1,1) |
| (1,2) | → | (2,1) |
| (2,0) | → | X |
| (2,1) | → | X |
| (2,2) | → | (2,2) |

k = 2



10



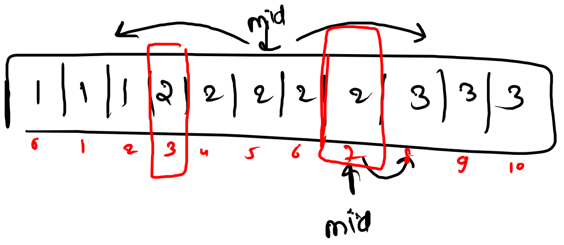
```
public static void fun(int[] arr, int n, int k) {
    int[] freq = new int[10];
    for (int i = 0; i < n; i++) {
        int idx = arr[i];
        freq[idx]++;
    }

    for (int i = 0; i < k; i++) {
        int max = Integer.MIN_VALUE;
        int idx = -1;
        for (int j = 0; j < 10; j++) {
            if ( freq[j] >= max ) {
                max = freq[j];
                idx = j;
            }
        }
        System.out.print(idx + " ");
        freq[idx] = 0;
    }
}
```

$T.C = O(n+k)$

# ↳ Binary Search

sorted



target = 2

## Template :-

```
int i = 0, j = n-1;
while ( i<=j) {
    int mid = (i+j)/2;
    if( arr[mid] == target){
        if(arr[mid] == arr[mid+1]){
            i = mid+1;
        }else{
            return mid;
        }
    } else if( arr[mid] > target){
        j = mid-1;
    } else{
        i = mid+1;
    }
}
```

```
if( arr[mid] == arr[mid-1])
    j = mid-1;
else
    return mid;
```

f = 3
l = 7

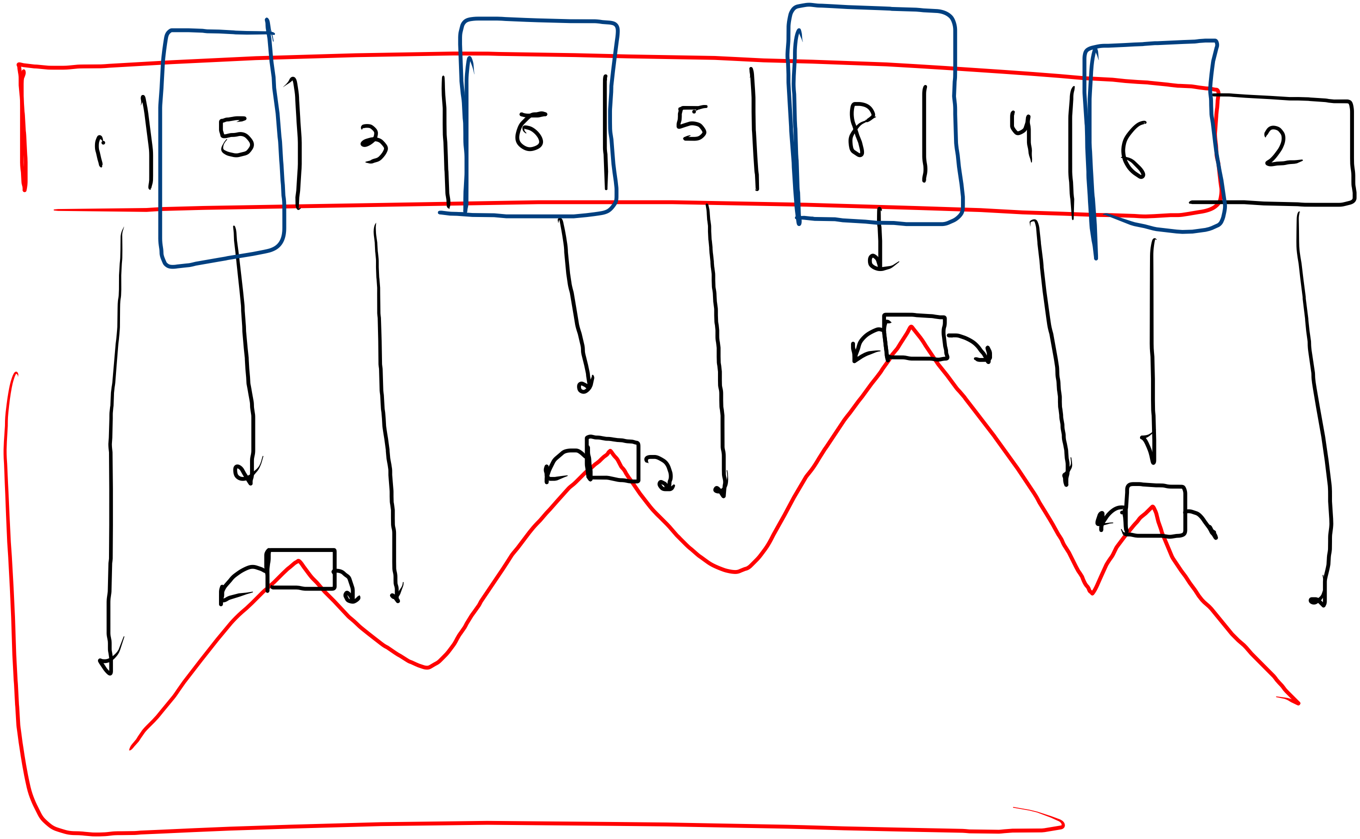
= 5

O(log N)

Note:- si → just greater element

ei → just smaller element

## Ques peak element



# Find The Index of Rotation

sorted rotated array

n = 10

arr

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

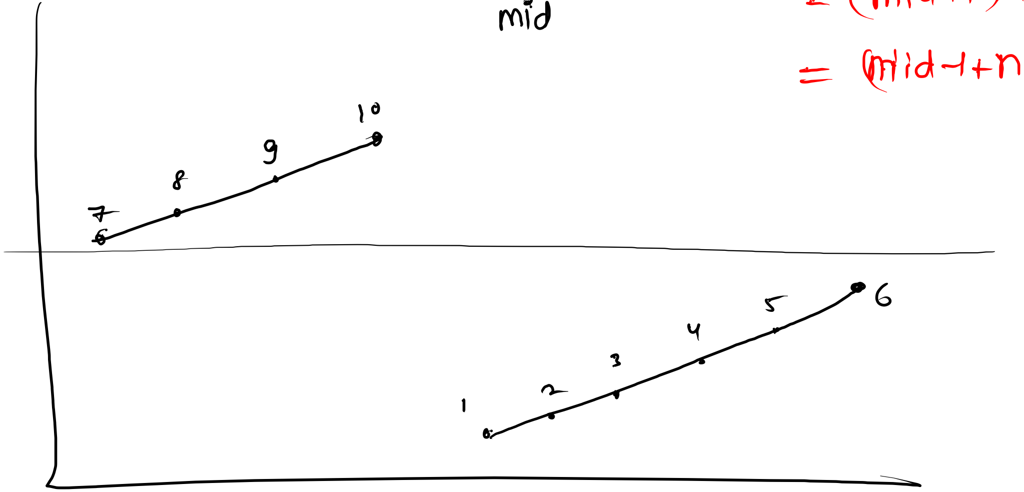
si

ei

mid

$= (mid+1)\%n;$

$= (mid-1+n)\%n;$



Note:-

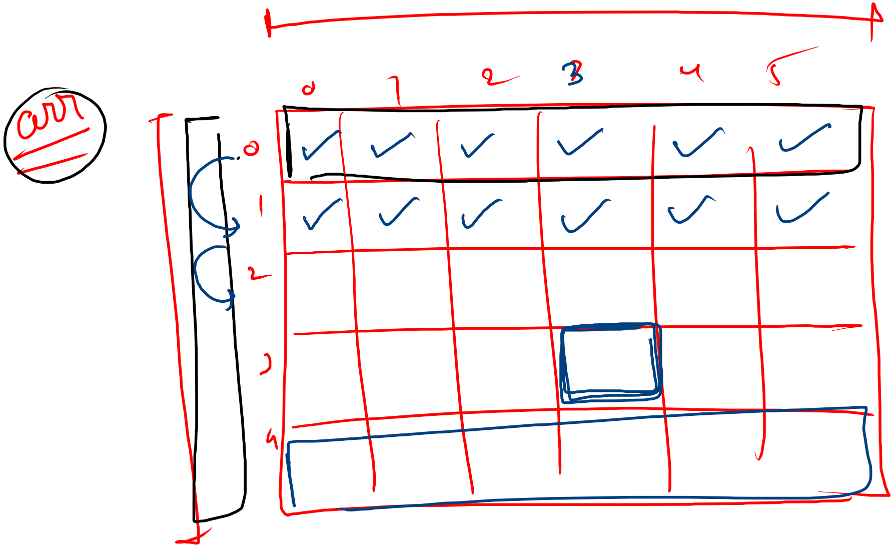
$$\boxed{mod = 10^9 + 7}$$

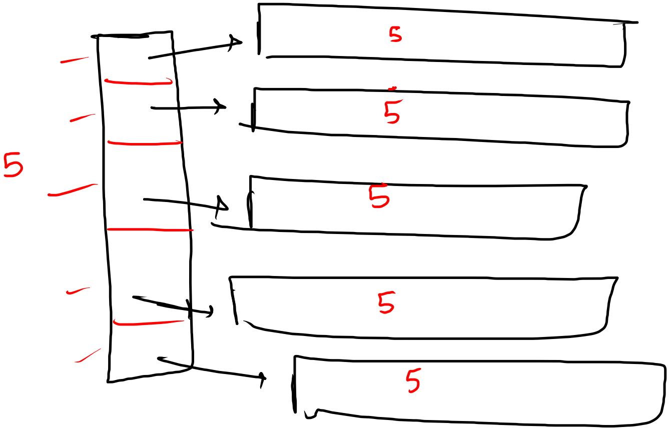
(num1 + num2) % mod ;

(num1 - num2 + mod) % mod ;

(num1 * num2) % mod ;

# ⇒ 2D array



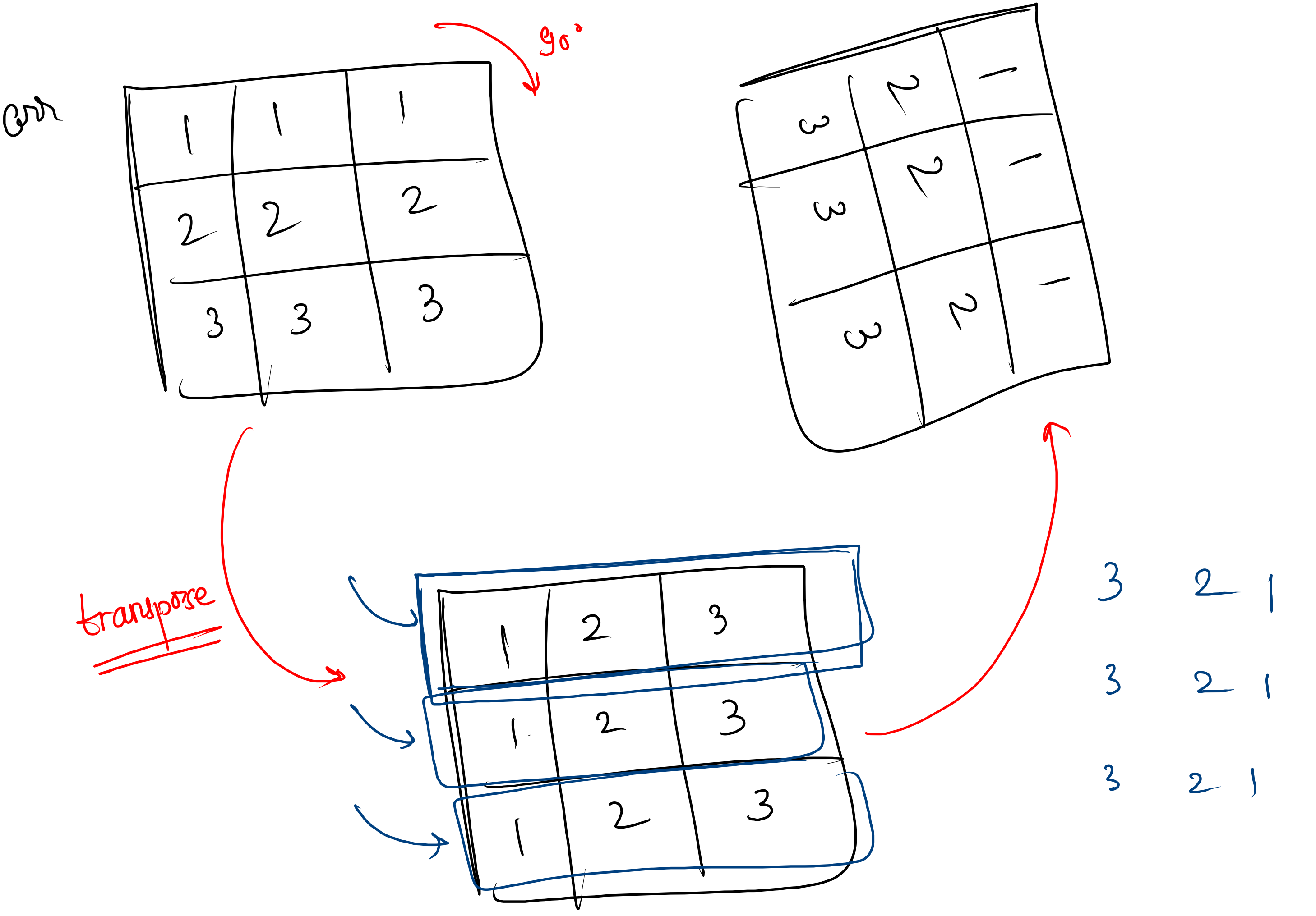


arr[3][3]

arr[4]

rows :- arr.length; ✓

cols :- arr[0].length ✓



arr

$$\begin{bmatrix}1&1&1\\2&2&2\\3&3&3\end{bmatrix}$$

90°

$$\begin{bmatrix}3&2&1\\3&2&1\\3&2&1\end{bmatrix}$$

transpose

$$\begin{bmatrix}1&2&3\\1&2&3\\1&2&3\end{bmatrix}$$

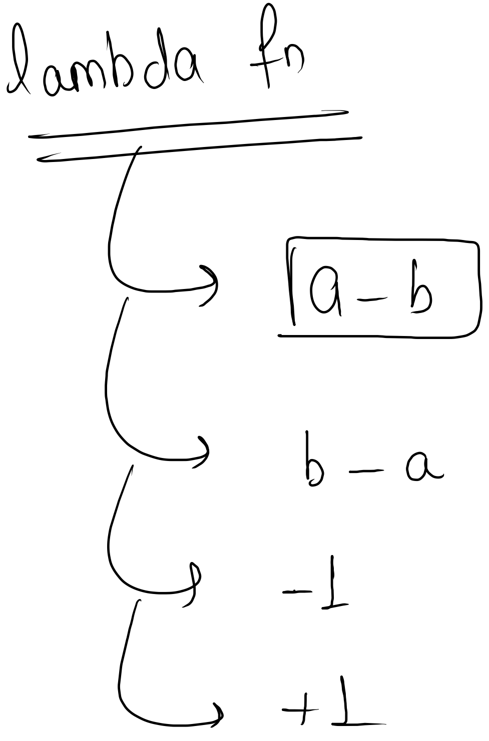
3 2 1

3 2 1

3 2 1

row

arr[i] → row each time



```
Arrays.sort(arr, (a, b) -> {
        if ( a = even & b = odd ) {
            return -1;
        } else if( a = odd, & b = even) {
            return +1;
        } else {
            return a-b;
        }
});
```

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | |