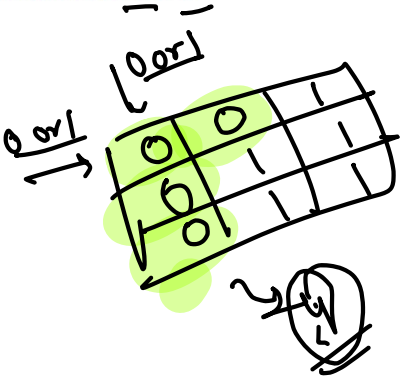
Egg Problem

Problem Statement – Consider a tray of eggs which contains eggs in different placeholders. Your task is to find all the empty placeholders in the given tray.

Input Format -
A 2D array of M*N size will be given where each cell can contain either 0 or 1.
0 - represents empty cell
1 - represents egg present

Output Format -
Number of empty cells in given tray
NOTE - Each row and column is sorted in the given array

Input-
[[0, 0, 1],
[0, 1, 1],
[0, 1, 1]]



output
4

input output

Steps → no. of zeroes
count

1. iterate over each element
   - Check element is 0 or 1

yes

```
class Solution {

    countZeros(A,N){
        //code here
        let count = 0;
        for(let i = 0;i<A.length;i++){
            for(let j = 0;j<A.length;j++){
                if(A[i][j] === 0){
                    count++;
                }
            }
        }
        return count;
    }
}
```

TC → NL

Ticket Queue Problem

Read

Problem Statement – There are multiple train ticket counters and there are long queues in front of each counter. Tickets are given to each person in the queue. Each ticket will contain some number. Your task is to find the sum of numbers on the ticket in each queue.

Input Format -
A 2D array of M*N size will be given (where N = number of ticket counters and each column will represent a queue on the counters)

Output Format -
An array having sum of each queue

45
29
74

Input:

10,	20,	30,	40
15,	25,	35,	45
24,	29,	37,	48
32,	33,	39,	50

2 rows

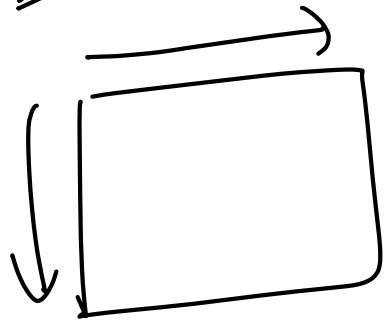
Input and output

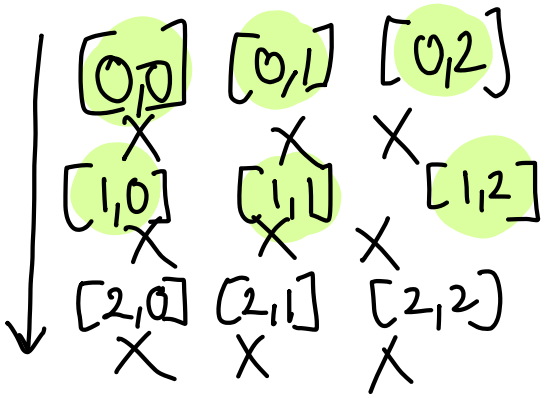
Output -
81, 107, 141, 183

→ Steps

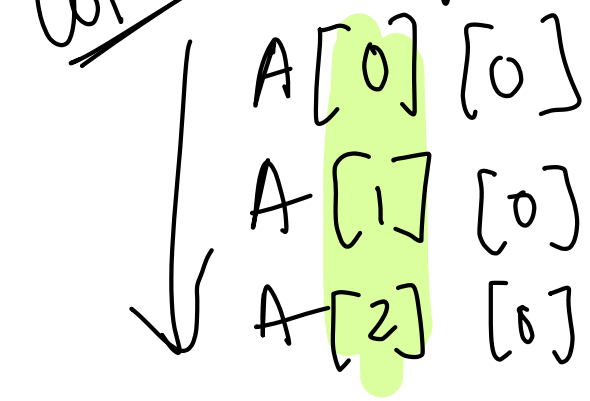
- Iterate to each column
- Sum all columns
- push in array

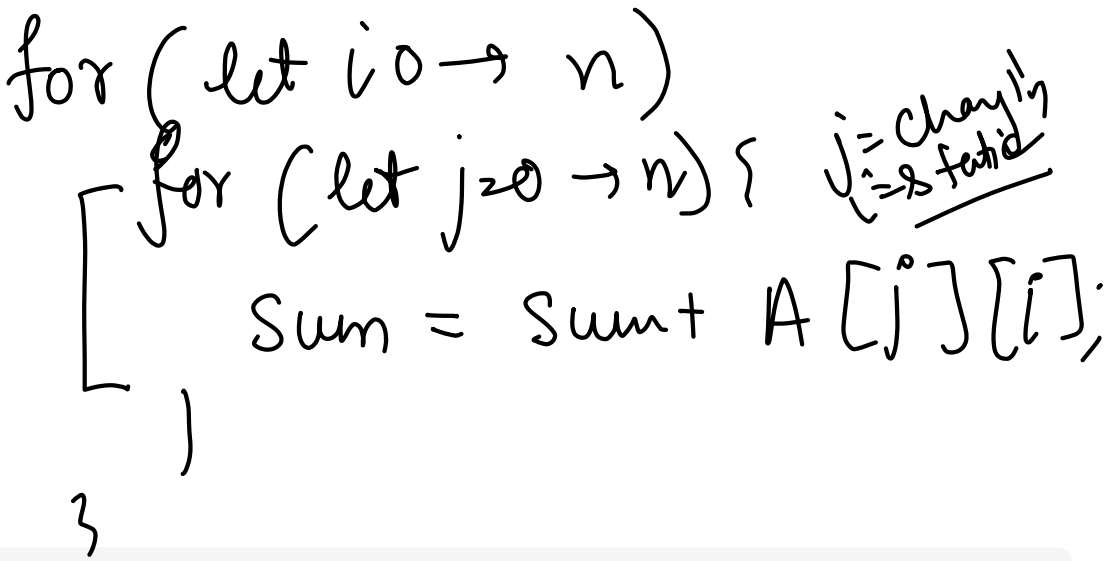
2D





Change static






```
1
2  function sumOfColumn(A){
3      let ansArr = [];
4      for(let i = 0;i<A.length;i++){
5          let sum = 0;
6          for(let j = 0;j<A[0].length;j++){
7              sum = sum + A[j][i];
8          }
9          ansArr.push(sum);
10     }
11     return ansArr;
12 }
13
14 let ans = sumOfColumn([[10,20,30,40],[15,25,35,45],[24,29,37,48],[32,33,39
       ,50]])
15
16 console.log(ans)
```

Chocolates Distribution Problem

Problem Statement – In a class, students are sitting on chairs which are arranged in the form a matrix i.e. 2D array. Chocolates have been distributed to them such that in each row and column, the number of chocolates are in increasing order.
You will be given a matrix which is a sitting arrangement of students having a number of chocolates and a number K.
You need to find the Kth smallest number of chocolates which a student can have.

Input- k = 4

10,	20,	30,	40
15,	25,	35,	45
24,	29,	37,	48
32,	33,	39,	50

Output - 24

input and output

Kth smallest

4th smallest

10th smallest

35

2 min

→ do we know how to find Kth in 1d Array?

→ Convert 2d array into 1d Array

→ Sort the array

→ return $(K-1)^{th}$ element.

rows { | | | |
|---|---|---|
| ⊗ | X | X |
| X | X | X |

```
    // Task 1
    // create 1-D array put all element here
    let arr = [];
    for(let i = 0;i<matrix.length;i++){
        for(let j = 0;j<matrix[0].length;j++){
            arr.push(matrix[i][j]);
        }
    }

    arr = arr.sort((a,b)=>a-b);

    return arr[k-1];
};
```

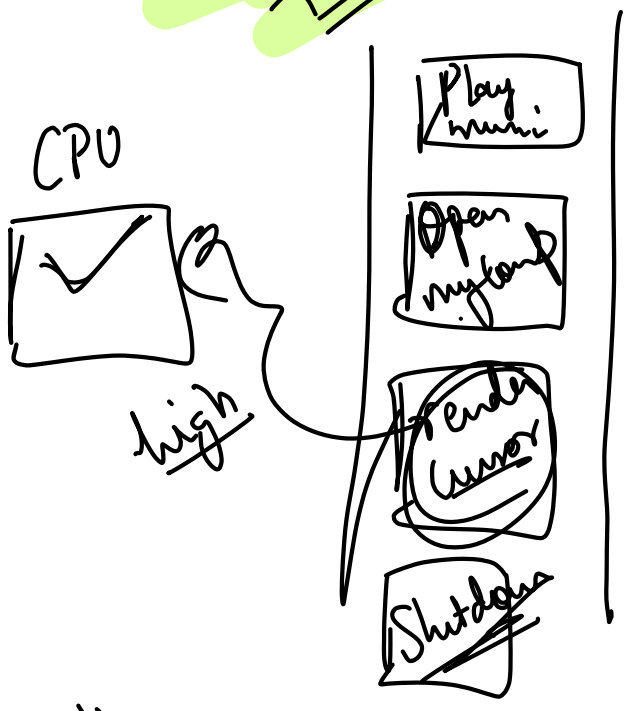
TC - (n x m) log(n x m)

Break

Sol-1

heap

Person with highest Priority will board first



highest Priority task must be picked by CPU

Priority → computer

Priority Queue

(log n) ⇒ Push → element

(log(n)) ⇒ pop → highest priority element

Priority → modify

min heap, max heap

push — ✓ log(n)

pop — ✓ log(n)

Input- k = 4

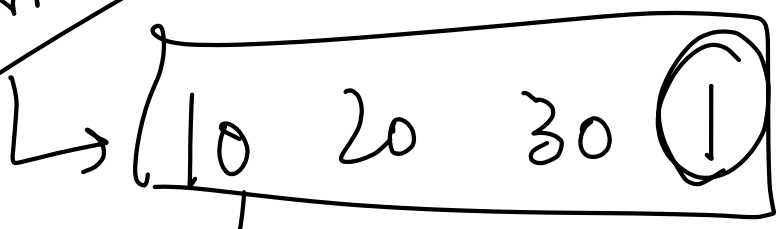
```
10, 20, 30, 40
15, 25, 35, 45
24, 29, 37, 48
32, 33, 39, 50
```

n^2 → heap → $n^2 \log(n^2)$

- put all ele in min heap $K \log(n^2)$
- pop k element
- Return K^{th} poped element

n^2 → 1 ele → $\log(n^2)$ → $n^2 \log(n^2)$

min heap
↳ min ele has max priority



pop() → 1

pop() → 10

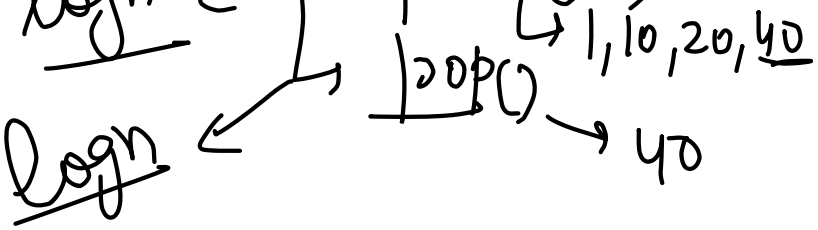
→ 20

max-heap
↳ element which is max has highest priority

1, 10, 20, 30

logn ← pop() → 30
1, 10, 20

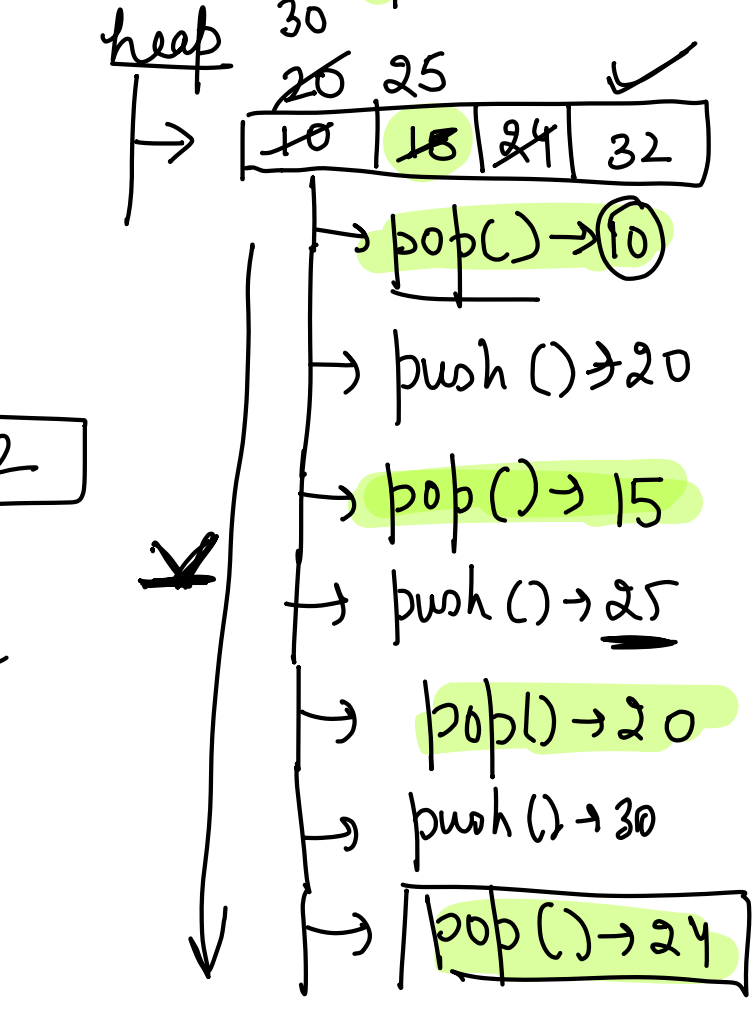
logn ← push(40)

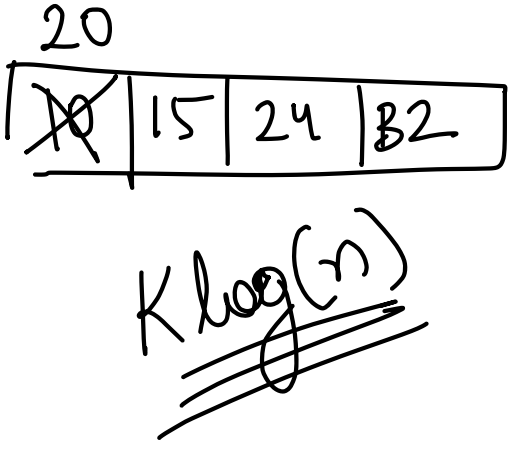


Input- k = 4

```
10, 20, 30, 40
15, 25, 35, 45
24, 29, 37, 48
32, 33, 39, 50
```

Can we improve?



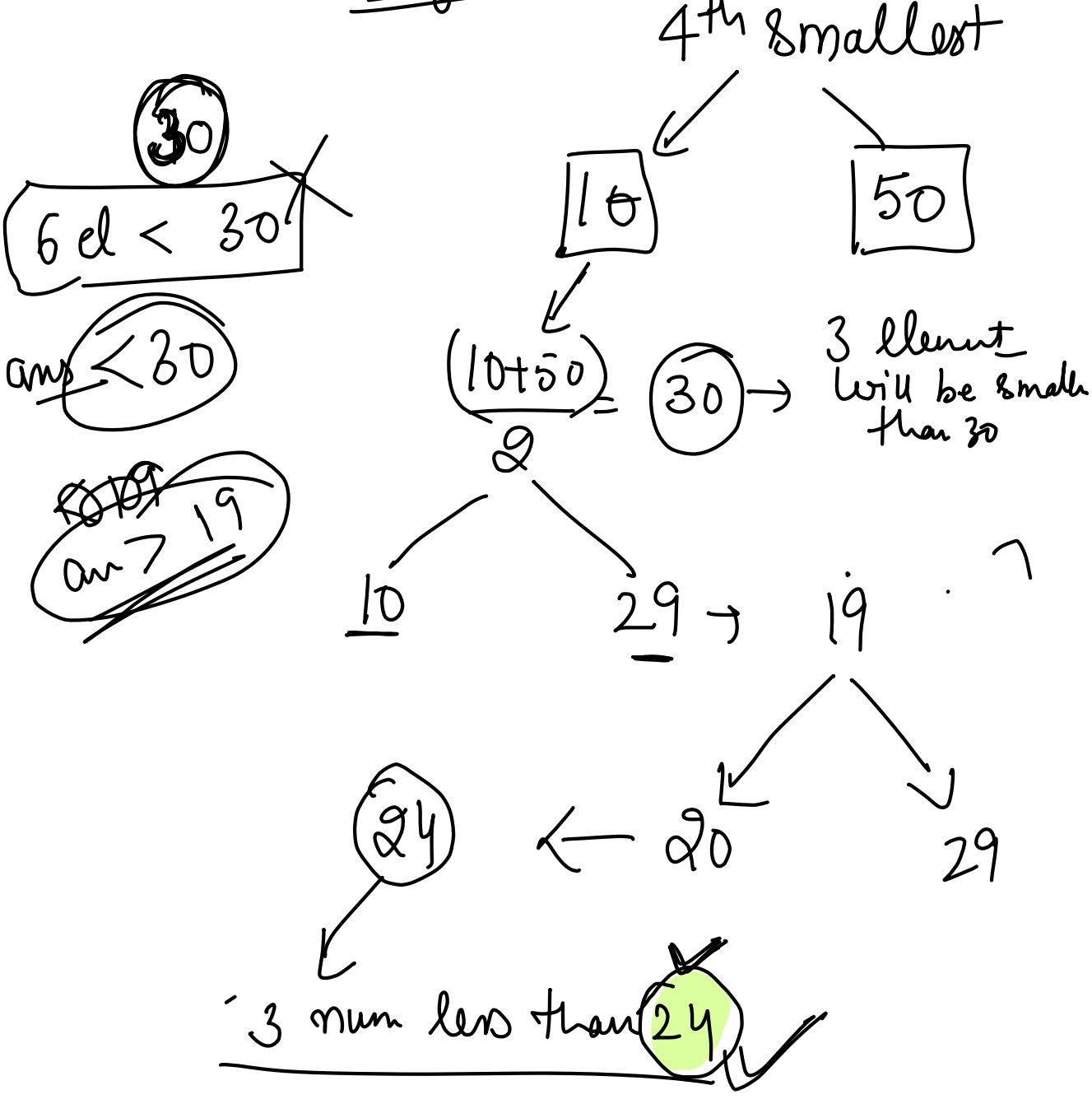


Input- k = 4

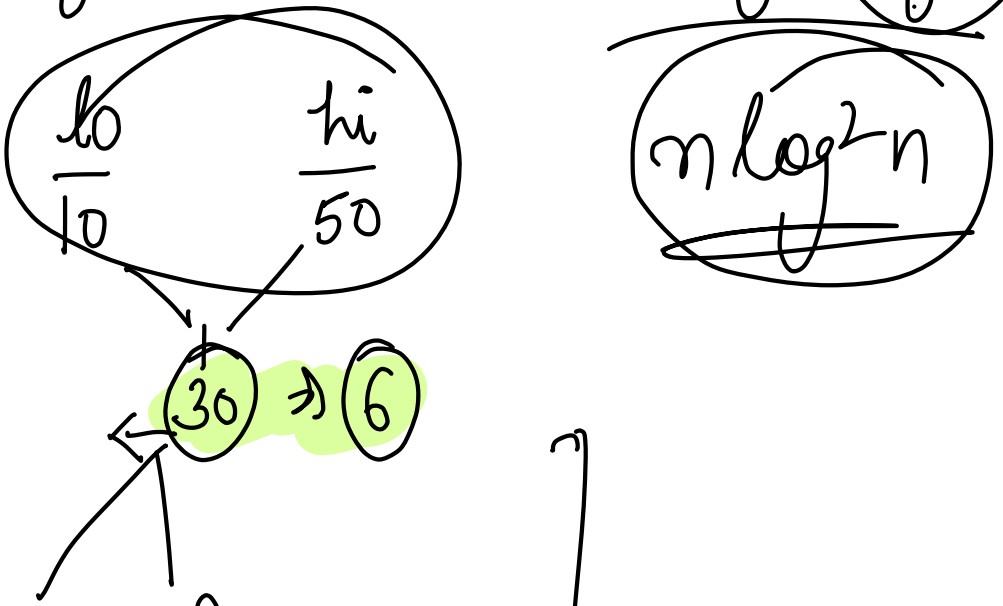
```
10, 20, 30, 40
15, 25, 35, 45
24, 29, 37, 48
32, 33, 39, 50
```

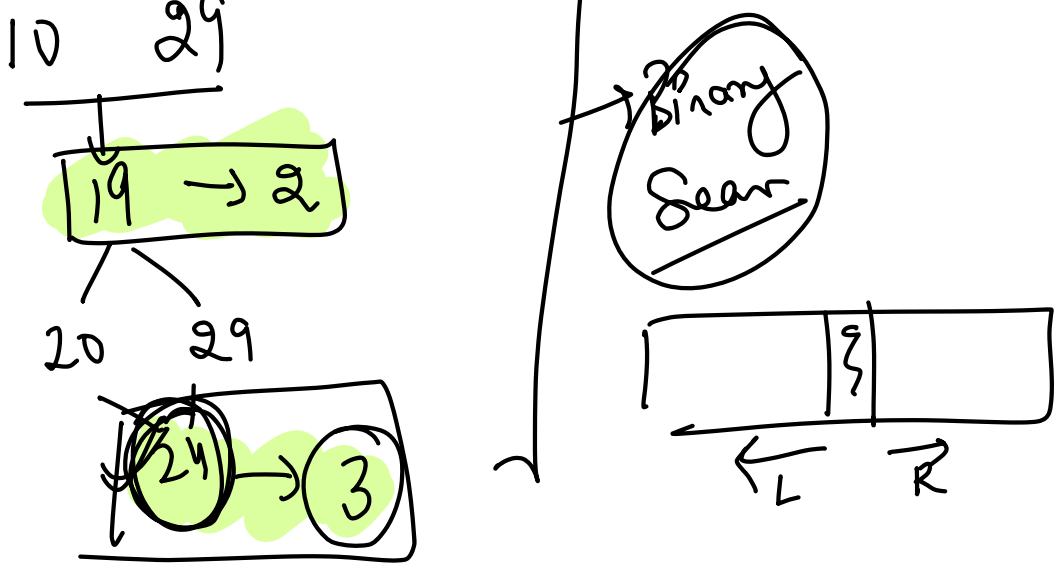
(1) all row and col are sorted

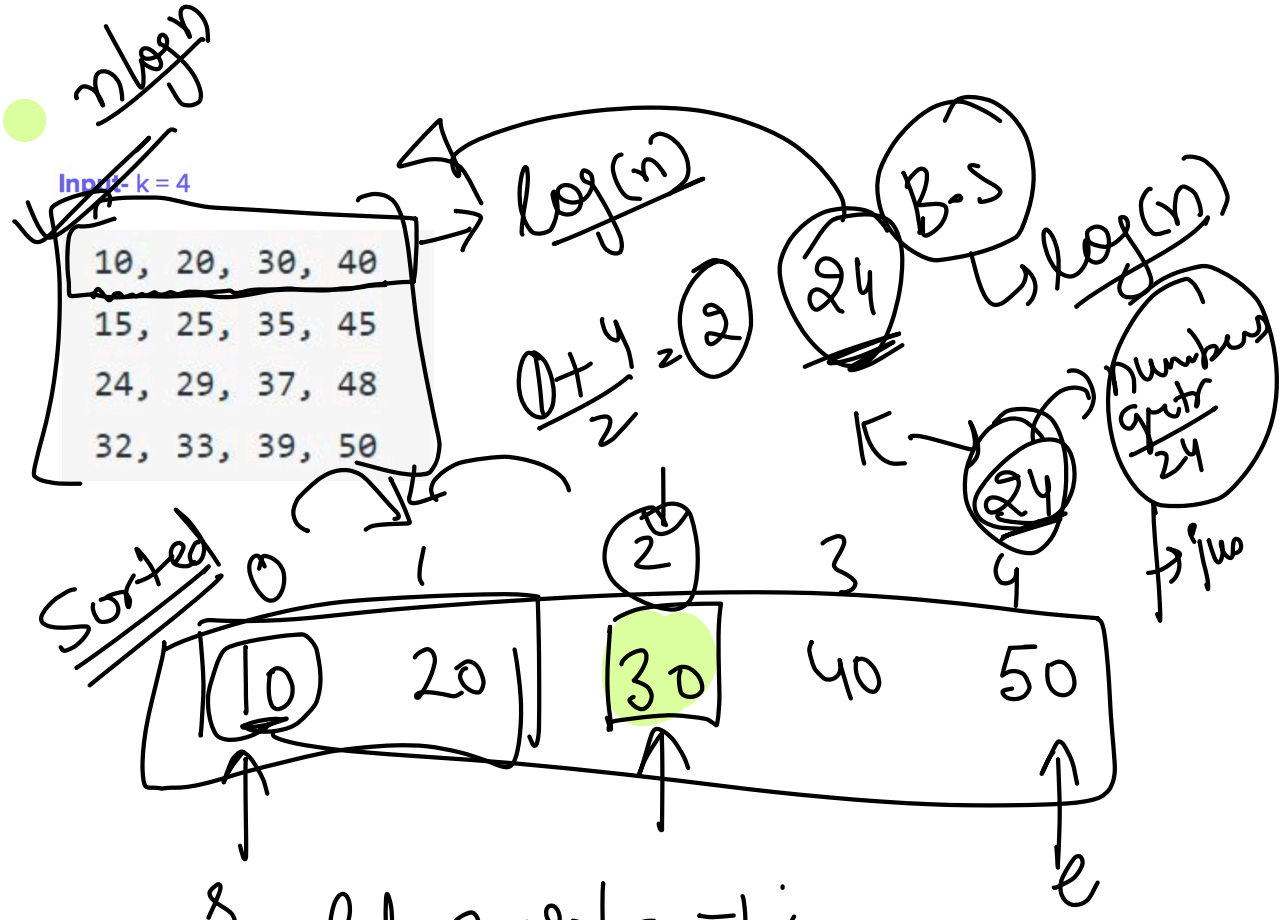
largest



(1) Binary Search → $n \log n$ $\log(n)$

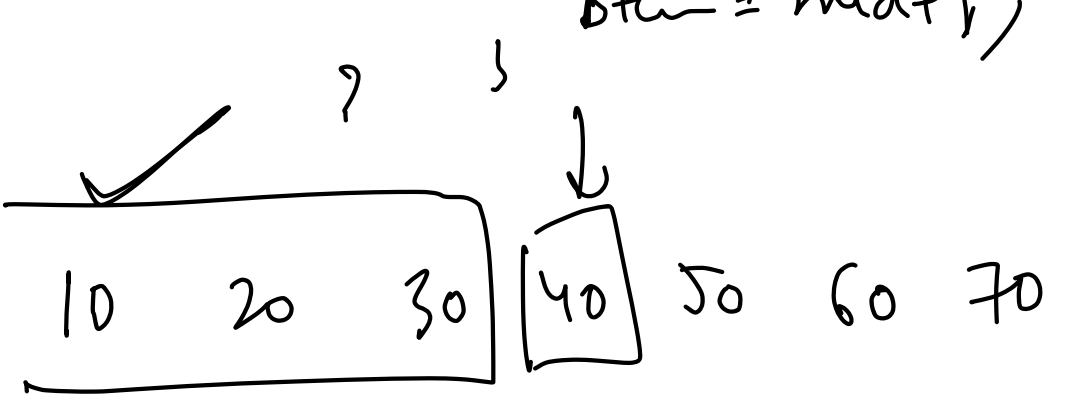


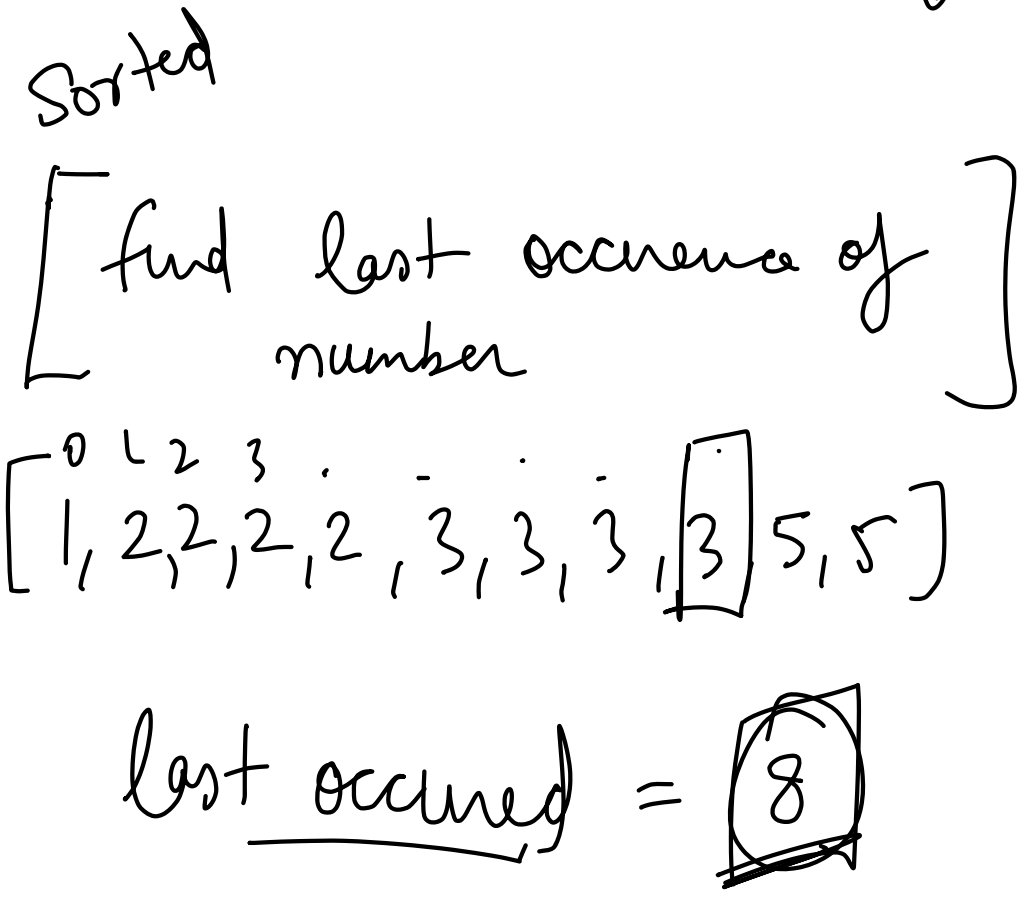





```
let possAns = -1;
while (s <= e) {
    mid = (s+e)/2
    if (arr[mid] > ans) {
        ans = mid;
        e = mid-1;
    }
    else {
```

s=0
e=1
0+1/2
=0





34. Find First and Last Position of Element in Sorted Array

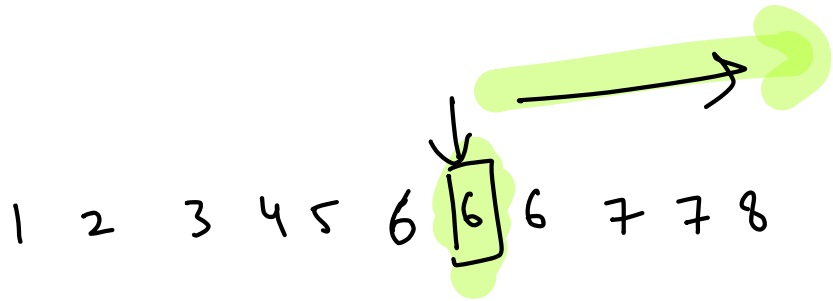
Medium 14236 350 Add to List Share

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given `target` value.

If `target` is not found in the array, return `[-1, -1]`.

You must write an algorithm with `O(log n)` runtime complexity.


1 2 3 3 3 4 5


1 2 3 4 5 6 6 6 7 7 8

704. Binary Search

Easy 6829 148 Add to List Share

Given an array of integers `nums` which is sorted in ascending order, and an integer `target`, write a function to search `target` in `nums`. If `target` exists, then return its index. Otherwise, return `-1`.

You must write an algorithm with `O(log n)` runtime complexity.

Example 1:

```
Input: nums = [-1,0,3,5,9,12], target = 9
Output: 4
Explanation: 9 exists in nums and its index is 4
```

Example 2:

```
Input: nums = [-1,0,3,5,9,12], target = 2
Output: -1
Explanation: 2 does not exist in nums so return -1
```

input output clear

$\frac{0+9}{2} = 4.5$

target = 6

1 2 3 4 5 6 7 8

Arr[mid] < target

```
/**
 * @param {number[]} nums
 * @param {number} target
 * @return {number}
 */
var search = function(nums, target) {

    let s = 0;
    let e = nums.length - 1;
    let ans = -1;
    while(s <= e){
        let mid = Math.floor((s + e)/2);
        if(nums[mid] == target){
            ans = mid;
            break;
        }
        else if(nums[mid] < target){
            // right
            s = mid + 1;
        }
        else{
            // left
            e = mid - 1;
        }
    }

    return ans;

};
```

true

0 <= 5

s = 0 2+1=3

e = 5

3 <= 5

true

ans = -1

mid = 2

nums = [-1,0,3,5,9,12],

target = 9

0 1 2 3 4

3+5 = 4

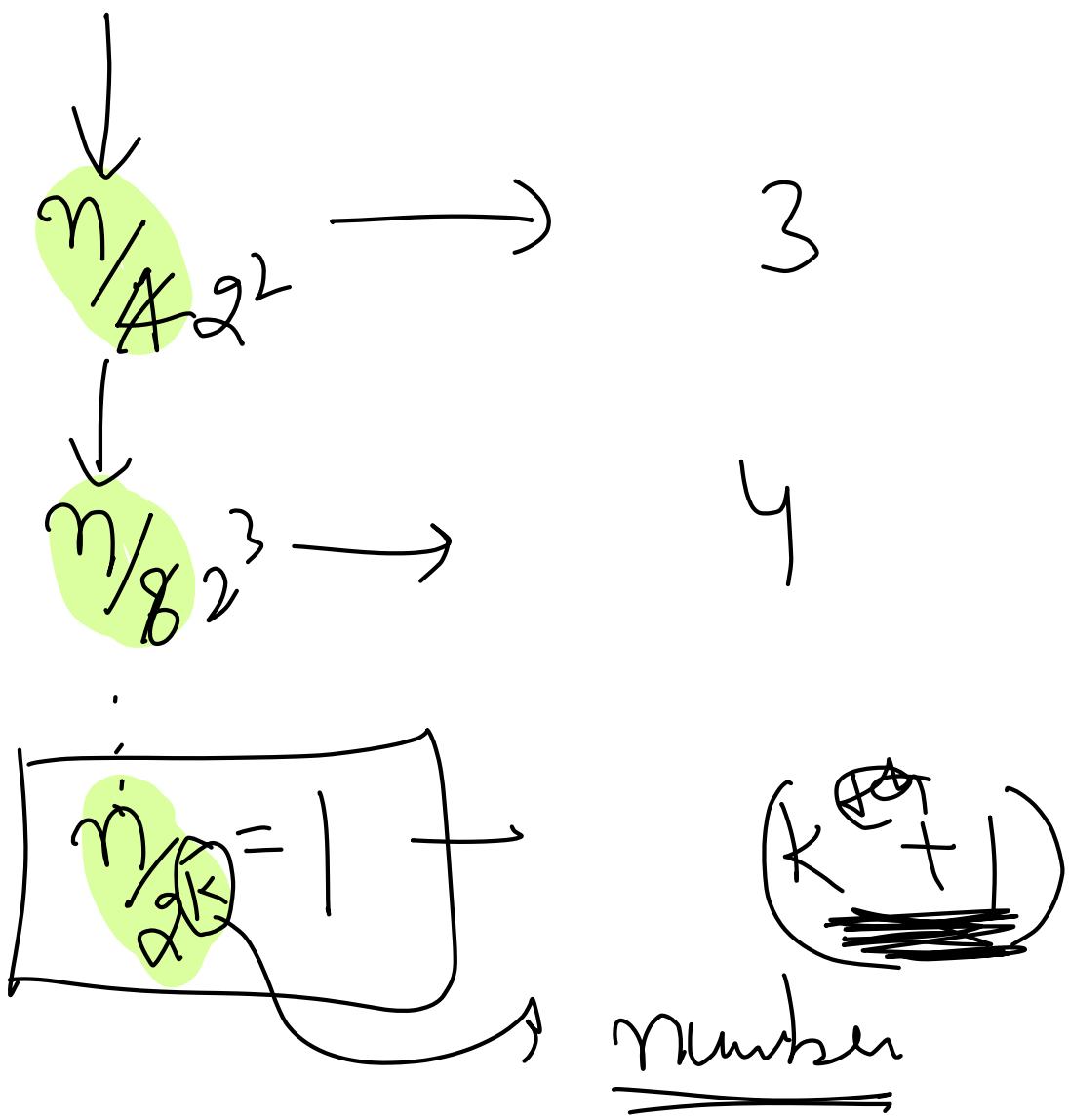
2

```javascript
/**
 * @param {number[]} nums
 * @param {number} target
 * @return {number}
 */
var search = function(nums, target) {

    let s = 0;
    let e = nums.length - 1;
    let ans = -1;
    while(s <= e){
        let mid = Math.floor((s + e)/2);
        if(nums[mid] == target){
            ans = mid;
            break;
        }
        else if(nums[mid] < target){
            // right
            s = mid + 1;
        }
        else{
            // left
            e = mid - 1;
        }
    }

    return ans;
};
```

nums = [-1,0,3,5,9,12],
target = 2



$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$\log n = k \log_2 2$$

$$\log(n) = k \log_2 2 \to 1$$

$k <= \log(n)$

15 min Break

① [1, 2, 3, 4, 5, 6, 7, 10, 11, 15, 17]

target = 9

→ number equal or just greater than target

11 − 7 = 4

[1, 2, 3, 4, 5, 6, 7, 10, 11, 15, 19]

tw 9 → s = 0 ⇒ 5
→ e = 10

6 < 9

11 > 9

11 < 9

ans = 8

8

==

<

→ left

>

← right

Save ans