GeeksforGeeks

Welcome All

@

Mastering DSA Basic to Advance

(Course for Product-Based Companies)

(Sorting + Searching)

Presented By: Puneet Kansal

Sorting Techniques

Binary Array Sorting

Given a binary array A[] of size N. The task is to arrange the array in increasing order.
Note: The binary array contains only 0 and 1.

Example :
Input: N=5, A[]={1 0 1 1 0}

Output:
0 0 1 1 1

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

https://practice.geeksforgeeks.org/problems/binary-array-sorting-1587115620/1?page=1&difficulty[]=-1&category[]=Sorting&sortBy=submissions

Three way partitioning

Given an array of size n and a range [a, b]. The task is to partition the array around the range such that array is divided into three parts.
1) All elements smaller than a come first.
2) All elements in range a to b come next.
3) All elements greater than b appear in the end.
The individual elements of three sets can appear in any order. You are required to return the modified array.

Note: The generated output is 1 if you modify the given array successfully.

Input:
n = 3
A[] = {1, 4, 3, 5, 2, 1}
[a, b] = [2, 3]
Output: 1

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

https://practice.geeksforgeeks.org/problems/three-way-partitioning/1

Types of Sorting algorithms

Sorting based

Interview Question

Which sorting algorithm is best if array is almost sorted?

Sort() in C++/Java/Python

- ✓The sort function in C++ uses the IntroSort internally which is a kind of hybrid sort. IntroSort is a combination of Quick Sort, Heap Sort, and Insertion Sort. The average complexity is N * log(N).

- ✓Timsort is a fast sorting algorithm working at stable O(N log(N)) complexity. Timsort is a blend of Insertion Sort and Mergesort. This algorithm is implemented in Java's Arrays. sort() as well as Python's sort().

Permutations in array

Use inbuilt sort() function

Given two arrays of equal size N and given an integer K. The task is to check if after permuting both arrays, we get sum of their corresponding element greater than or equal to k i.e $A_i + B_i >= K$ for all i (from 0 to N-1). Return true if possible, else false.

Example 1:

Input : a[] = {2, 1, 3}, b[] = { 7, 8, 9 }, k = 10.
Output : True
Explanation: Permutation a[] = { 1, 2, 3 }
and b[] = { 9, 8, 7 }
satisfied the condition a[i] + b[i] >= K.

Expected Time Complexity: O(N. Log(N))
Expected Auxiliary Space: O(1)

Hint: Check if min(arr1) + max(arr2) >= K or not

https://practice.geeksforgeeks.org/problems/permutations-in-array1747/1?page=1&difficulty[]=-1&category[]=Sorting&sortBy=submissions

Sorting based

Interview Question

Can we apply the binary search in insertion sort to find exact position of an element?
If Yes, then what will be the time complexity of insertion sort?

Stable sort + Inplace algo

Algorithm	Stable(Yes/No)	Inplace(Yes/No)
Bubble Sort	**Yes**	**Yes**
Selection Sort	**No**	**Yes**
Insertion Sort	**Yes**	**Yes**
Merge sort	**Yes**	**No**
Quick sort	**No**	**Yes**
Heap sort	**No**	**Yes**

- Selection sort is unstable for this sequence 5, 5, 3, 2, 1

Stable Sort and Position

Given an array arr[] of N integers which may contain duplicate elements, the index of an element of this array is given to us, the task is to find the final position of this element (for which the index is given) in the array after the stable sort is applied on the array.

Example :
Input: N = 10, INDEX = 5
arr[]= {3, 4, 3, 5, 2, 3, 4, 3, 1, 5}
Output: 4

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

https://practice.geeksforgeeks.org/problems/stable-sort-and-position1359/1?page=2&difficulty[]=-1&category[]=Sorting&sortBy=submissions

P r o b l e m

Merge Two Sorted Array:

- *How to merge two sorted arrays to generate one combined sorted array?*

Sort the Half Sorted

Microsoft

Given an integer array of which both the first halve and second halve are sorted. The task is to merge these two sorted halves of the array into a single sorted array.
Note: The two halves can be of arbitrary sizes (i.e. if first halve of size k then the second halve is of size N-k where 0<=k<=N).

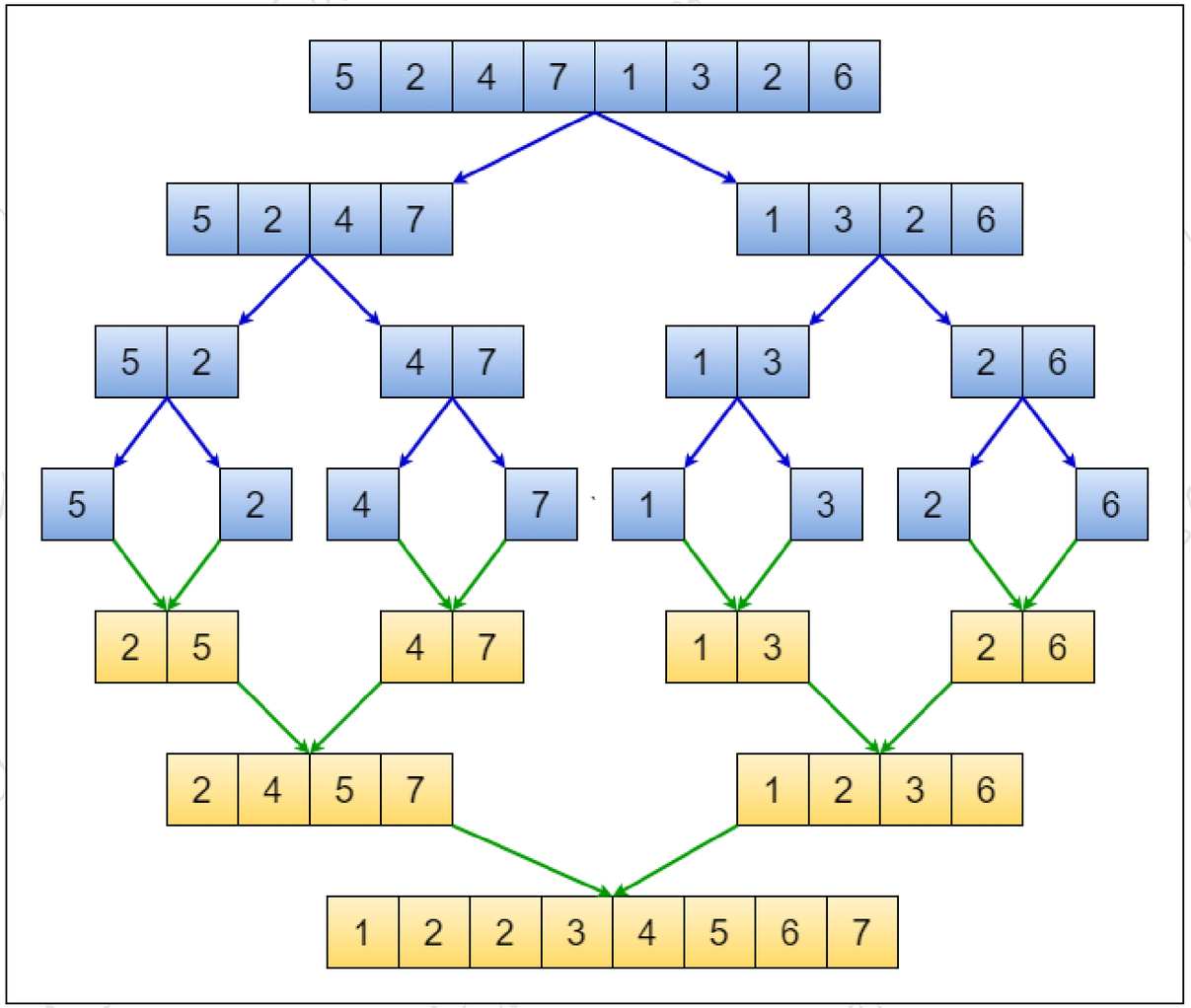
Example :
Input: N = 6
arr[] = {2 3 8 -1 7 10}
Output: -1 2 3 7 8 10
Explanation: {2 3 8} and {-1 7 10} are sorted in the original array. The overall sorted version is {-1 2 3 7 8 10}

Expected Time Complexity: O(n).
Expected Auxiliary Space: O(n).

https://practice.geeksforgeeks.org/problems/sort-the-half-sorted2157/1?page=2&difficulty[]=-1&category[]=Sorting&sortBy=submissions

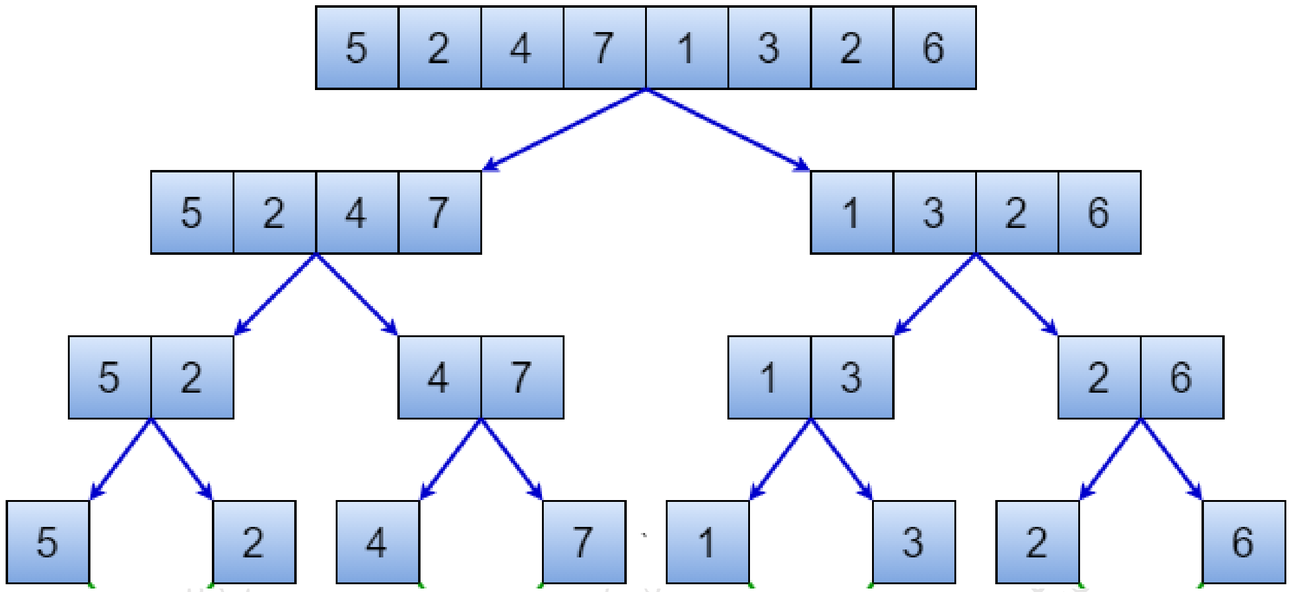
Merge Sort Algorithm

```
MERGE-SORT(A, p, r)
1  if p < r
2      q = ⌊(p + r)/2⌋
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```



How Merge Sort Works

```
MERGE-SORT(A, p, r)
1  if p < r
2      q = ⌊(p + r)/2⌋
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```



How Merge Sort Works

(Time Complexity)

```
MERGE-SORT(A, p, r)
1  if p < r
2        q = ⌊(p + r)/2⌋
3        MERGE-SORT(A, p, q)
4        MERGE-SORT(A, q + 1, r)
5        MERGE(A, p, q, r)
```

Merge Without Extra Space

Given two sorted arrays **arr1[]** and **arr2[]** of sizes **n** and **m** in non-decreasing order. Merge them in sorted order without using any extra space. Modify arr1 so that it contains the first N elements and modify arr2 so that it contains the last M elements.

Example :

Input:

n = 4, arr1[] = [1 3 5 7]

m = 5, arr2[] = [0 2 6 8 9]

Output:

arr1[] = [0 1 2 3]

arr2[] = [5 6 7 8 9]

Expected Time Complexity: O((n+m) log(n+m))

Expected Auxiliary Space: O(1)

https://www.geeksforgeeks.org/problems/merge-two-sorted-arrays-1587115620/1?page=1&category=Sorting&difficulty=Hard&sortBy=submissions

Quick Sort

Quick Sort
(Time Complexity)

```
QuickSort(A, p, r)
        if p < r
        q = Partition(A, p, r)
        QuickSort(A, p, q - 1)
        QuickSort(A, q + 1, r)

Partition(A, p, r){
        x = A[r];
        i = p -1;
        for(j = p to r-1){
                if(A[j] <= x){
                        i++;
                        exchange A[i] with A[j]
                        }
                }
        Exchange A[i+1] with A[r];
        return i+1;
}
```

https://www.geeksforgeeks.org/problems/quick-sort/1?page=1&category=Sorting&difficulty=School,Basic,Easy,Medium,Hard&sortBy=submissions

Kth smallest element

Given an array arr[] and an integer K where K is smaller than size of array, the task is to find the Kth smallest element in the given array. It is given that all array elements are distinct.

Note :- **l** and **r** denotes the **starting** and **ending** index of the array.

Example :
Input: N = 6 arr[] = 7 10 4 3 20 15 K = 3
Output : 7
Explanation :
3rd smallest element in the given array is 7.

Constraints:
$1 <= N <= 10^5$
$1 <= arr[i] <= 10^5$
$1 <= K <= N$

Expected Time Complexity: O(n)
Expected Auxiliary Space: O(log(n))

https://www.geeksforgeeks.org/problems/kth-smallest-element5635/1?page=1&category=Sorting&difficulty=Medium&sortBy=submissions

Important Point

The merging process in merge sort and partition process in quicksort are good examples of two pointer approach.

Heap sort

Introduction to heap

- Heap is a data structure in the form of binary tree or 3-ary tree…….n-ary tree.

- Heap is almost complete tree.

Type of heap

Height / Depth of Tree

- No. of edges b/w last level leaf node to the root node is called depth of the binary tree.
- The depth/height of the almost complete binary tree with "n" nodes is $\log_2 n$.
- Level of CBT = height +1

Priority Queue in C++

- Priority Queue is a STL container in C++, in which the top element is either the largest or the smallest of all the elements(depending on the type of the priority queue). However, the highest element is always the default in the C++ STL.

- The time complexity of operations like insertion and deletion in the priority queue in C++ is O(logn).

- priority_queue<int> max_q; //creating max queue or max heap
- priority_queue<int, vector<int>, greater<int>> min_q;//creating min queue or min heap

- int arr[10] = {3,4,1,2,6,8,5,7,9,0};
- int length = sizeof(arr)/sizeof(arr[0]); // n=array.size();
- priority_queue<int> pq(arr, arr + n); //creating max heap from array

- vector<int> v(arr, arr + N);
- priority_queue <int,vector<int>,greater<int>> pq ;(v.begin() , v.end());//creating min heap from vector

Priority Queue in Java

- The PriorityQueue is based on the priority heap. The head of this queue is the least element with respect to the specified ordering. If multiple elements are tied for the least value, the head is one of those elements — ties are broken arbitrarily.

- PriorityQueue doesn’t permit null.

```
PriorityQueue<Integer> pQueue = new PriorityQueue<Integer>();       // min heap
PriorityQueue<Integer> pQueue = new PriorityQueue<Integer>(Collections.reverseOrder());// max heap
```

Heap queue (or heapq) in Python

```
import heapq

# Initialize a list with some values
values = [5, 1, 3, 7, 4, 2]

# Convert the list into a heap
heapq.heapify(values) #minheap
heapq._heapify_max(values) #maxheap

# Print the heap
print("Heap:", values)

# Add a new value to the heap
heapq.heappush(values, 6)

# Remove and return the smallest element from the heap
smallest = heapq.heappop(values)

# Get the n smallest elements from the heap
n_smallest = heapq.nsmallest(3, values)

# Print the n smallest elements
print("Smallest 3 elements:", n_smallest)

# Get the n largest elements from the heap
n_largest = heapq.nlargest(2, values)

# Print the n largest elements
print("Largest 2 elements:", n_largest)
```

K largest elements

Given an array of N positive integers, print k largest elements from the array.

Example :
Input:
N = 5, k = 2
arr[] = {12,5,787,1,23}
Output: 787 23

Expected Time Complexity: O(N+K log N)

https://practice.geeksforgeeks.org/problems/k-largest-elements3736/1?page=1&difficulty[]=-2&difficulty[]=-1&category[]=Heap&sortBy=submissions

Counting sort

```
1.  Counting-Sort(A, B, k)
2.  Let C[0.....k] be a new array
3.  for i=0 to k
4.      C[i]= 0;
5.  for j=1 to A.length  or n
6.      C[ A[j] ] = C[ A[j] ]  + 1;
7.  for i=1 to k
8.      C[i] = C[i] + C[i-1];
9.  for j=n or A.length down to 1
10.     B[  C[ A[j] ]  ] = A[j];
11.     C[ A[j] ] = C[ A[j] ]  - 1;
```

Merge and Sort

Given two arrays of length N and M, print the merged array in ascending order containing only unique elements.

Example :

Input: N = 2 a[] = {1, 8} M = 2 b[] = {11, 10}

Output: answer[] = {1, 8, 10, 11}

Constraints:

$1 \le \mathbf{N} \le 10^5$

$1 \le \mathbf{M} \le 10^5$

$1 \le \mathbf{A_i} \le 10^5$

$1 \le \mathbf{B_i} \le 10^5$

Expected Time Complexity: O(N)

Expected Auxiliary Space: O(N)

Note: We can apply counting sort if the size of input array elements is small and we need to sort array in O(n) time.

https://practice.geeksforgeeks.org/problems/merge-and-sort5821/1?page=2&difficulty[]=-1&category[]=Sorting&sortBy=submissions

Note: Range of int = 10^9 and long long int = 10^{18}

The Latest Time to Catch a Bus

You are given a 0-indexed integer array buses of length n, where buses[i] represents the departure time of the ith bus. You are also given a 0-indexed integer array passengers of length m, where passengers[j] represents the arrival time of the jth passenger. All bus departure times are unique. All passenger arrival times are unique.

You are given an integer capacity, which represents the maximum number of passengers that can get on each bus.

When a passenger arrives, they will wait in line for the next available bus. You can get on a bus that departs at x minutes if you arrive at y minutes where y <= x, and the bus is not full. Passengers with the earliest arrival times get on the bus first.

More formally when a bus arrives, either:

If capacity or fewer passengers are waiting for a bus, they will all get on the bus, or
The capacity passengers with the earliest arrival times will get on the bus.
Return the latest time you may arrive at the bus station to catch a bus. You cannot arrive at the same time as another passenger.

Note: The arrays buses and passengers are not necessarily sorted.

- **Example1:**

  Input: buses = [10,20], passengers = [2,17,18,19], capacity = 2

  Output: 16

Example2:

Input: buses = [20,30,10], passengers = [19,13,26,4,25,11,21], capacity = 2

Output: 20

https://leetcode.com/problems/the-latest-time-to-catch-a-bus/

Sort Characters By Frequency

Given a string s, sort it in decreasing order based on the frequency of the characters. The frequency of a character is the number of times it appears in the string.

Return the sorted string. If there are multiple answers, return any of them.

Example :
Input: s = "tree"
Output: "eert“

Example 2:

Input: s = "cccaaa"
Output: "aaaccc"

https://leetcode.com/problems/sort-characters-by-frequency/description/

Searching Techniques

Searching Techniques Problem-1

What is the function of given code?

```
bool fun(int arr[], int N, int x)    // N is size of array
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return true;
    return false;
}
```

Time Complexity: *O(n)*
Auxiliary Space : *O(1)*

Output: Linear Search

What will be the “Best case” time complexity to find minimum element in an unsorted array?

Searching Techniques
Problem-2

What is the function of given code?

```
def fun(arr, key, index):
    if index >= len(arr):
        return -1
    if arr[index] == key:
        return index
    return fun(arr, key, index + 1)

print(fun([1,2,3,4], 3, 0))
```

Time Complexity: *O(n)*
Auxiliary Space : *O(n)*

Output: Linear Search (recursion)

Value equal to index value

Problem-3

Given an array Arr of N positive integers. Your task is to find the elements whose value is equal to that of its index value (Consider 1-based indexing).

Note: There can be more than one element in the array which have the same value as its index. You need to include every such element's index. Follows 1-based indexing of the array.

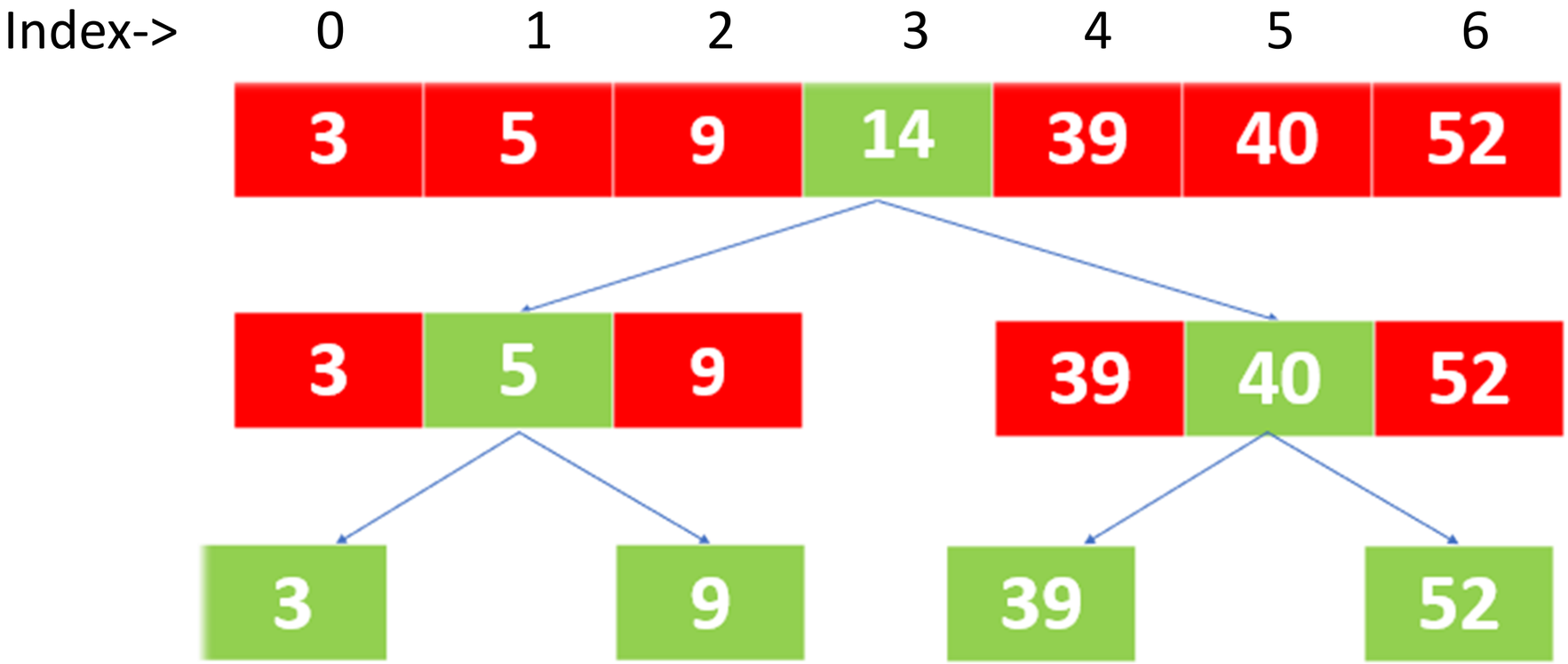
- **Example:**

  Input: *N = 5 Arr[] = {15, 2, 45, 12, 7}*

  Output: *{2}*

  Explanation: *Only Arr[2] = 2 exists here.*

https://practice.geeksforgeeks.org/problems/value-equal-to-index-value1330/1?page=1&difficulty[]=-2&category[]=Searching&category[]=Binary%20Search&sortBy=submissions



Binary Search

mid = low+(high-low)/2 = (high+low)/2

Is there any difference between low+(high-low)/2 & LB((high+low)/2) statements ?

Binary Search

Problem-4

accenture

Given a sorted array of size N and an integer K, find the position(0-based indexing) at which K is present in the array using binary search. If element not found return -1.

- **Example:**

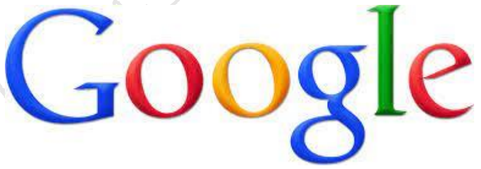
  Input: *N = 5 arr[] = {1 2 3 4 5} K = 4*

  Output: *3*

  Explanation: *4 appears at index 3.*

https://practice.geeksforgeeks.org/problems/binary-search-1587115620/1

Interview Question



What is the time complexity to determine if an integer appears more than n/2 times in a sorted array of n integers?

Time Complexity: $O(log_2 n)$

Hint:

As the element is present more than n/2 times, then definitely it will be present at the middle index position.

Searching Techniques
Problem-5

What is the function of given code?

```
int foo(int nums[ ], int n, int val)
    {
    int l = 0, h = n - 1;

    while (l <= h)
        {
        int m = (l + h) / 2;
        if (nums[m] == val)
            return m;
        else if (val > nums[m])
                l = m + 1;
            else
                h = m - 1;
        }
    return -1;
    }
```

Time Complexity: *O(logn)*
Auxiliary Space : *O(1)*

Output: Binary Search using iteration

Search insert position of K in a sorted array

Microsoft

Given a sorted array Arr[](0-index based) consisting of N distinct integers and an integer k, the task is to find the index of k, if its present in the array Arr[]. Otherwise, find the index where k must be inserted to keep the array sorted.

Example1:

Input: N = 4 Arr = {1, 3, 5, 6} k = 5

Output: 2

Example2:

Input: N = 4 Arr = {1, 3, 5, 6} k = 4

Output: 2

Expected Time Complexity: O(logN)
Expected Auxiliary Space: O(logN)

https://practice.geeksforgeeks.org/problems/search-insert-position-of-k-in-a-sorted-array/1

Sqrt(x)

Goldman Sachs

Given a non-negative integer x, return the square root of x rounded down to the nearest integer. The returned integer should be non-negative as well.

You must not use any built-in exponent function or operator.

For example, do not use pow(x, 0.5) in c++ or x ** 0.5 in python.

Example 1:

Input: x = 4

Output: 2

Example 2:

Input: x = 8

Output: 2

https://leetcode.com/problems/sqrtx/description/

Search in Rotated Sorted Array

media.net

There is an integer array **nums** sorted in ascending order (with distinct values).

Prior to being passed to your function, nums is possibly rotated at an unknown pivot index k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0-indexed).

For example, [0,1,2,4,5,6,7] might be rotated at pivot index 3 and become [4,5,6,7,0,1,2].

Given the array nums after the possible rotation and an integer **target**, return the **index of target** if it is in nums, or **-1** if it is not in nums.

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

Example:

Input: nums = [4,5,6,7,0,1,2], target = 0

Output: *4*

Input2: nums = [6,7,0,1,2,3,4,5], target = 5

Output: *7*

https://leetcode.com/problems/search-in-rotated-sorted-array/description/

Search in Rotated Sorted Array II

There is an integer array nums sorted in non-decreasing order (not necessarily with distinct values).

Before being passed to your function, nums is rotated at an unknown pivot index k (0 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). For example, [0,1,2,4,4,4,5,6,6,7] might be rotated at pivot index 5 and become [4,5,6,6,7,0,1,2,4,4].

Given the array nums after the rotation and an integer target, return true if target is in nums, or false if it is not in nums.

You must decrease the overall operation steps as much as possible.

Example1: ***Input:*** nums = [2,5,6,0,0,1,2], target = 0 ***Output:*** *true*

Example2: ***Input:*** nums = [7,4,5,7,7,7,7], target = 5 ***Output:*** *true*

Example3: ***Input:*** nums = [7,7,7,7,4,5,7], target = 5 ***Output:*** *true*

https://leetcode.com/problems/search-in-rotated-sorted-array-ii/description/

Search in a matrix

Given a matrix mat[][] of size N x M, where every row and column is sorted in increasing order, and a number X is given. The task is to find whether element X is present in the matrix or not.

Example:

Input: *N = 3, M = 3*

mat[][] = 3 30 38

44 52 54

57 60 69

X = 62

Output: *0*

Explanation: *62 is not present in the matrix, so output is 0*

Expected Time Complexity: O(N+M).
Expected Auxiliary Space: O(1).

https://www.geeksforgeeks.org/problems/search-in-a-matrix17201720/1

https://leetcode.com/problems/search-a-2d-matrix-ii/description/

Find Peak Element

A peak element is an element that is strictly greater than its neighbors.

Given a 0-indexed integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to any of the peaks.

You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in O(log n) time.

Example1:

Input: *N = 10*

arr[] = {1,2,3,4,2,1,5,6,4,1}

Output: *7*

Intuition:

If nums[mid] > nums[mid+1] then confirm one peak is at LHS, otherwise confirm one peak is at RHS.

https://leetcode.com/problems/find-peak-element/description/

Find a Peak Element II

A peak element in a 2D grid is an element that is strictly greater than all of its adjacent neighbors to the left, right, top, and bottom.

Given a 0-indexed m x n matrix mat where no two adjacent cells are equal, find any peak element mat[i][j] and return the length 2 array [i,j].

You may assume that the entire matrix is surrounded by an outer perimeter with the value -1 in each cell.

You must write an algorithm that runs in O(m log(n)) or O(n log(m)) time.

Example:

Input: *mat = [[10,20,15],[21,30,14],[7,16,32]]*

Output: *[1,1]*

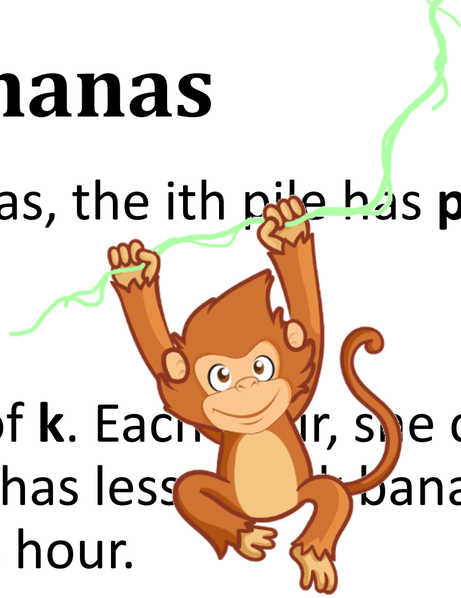
Explanation: Both 30 and 32 are peak elements

so [1,1] and [2,2] are both acceptable answers.

-1	-1	-1	-1	-1
-1	10	20	15	-1
-1	21	30	14	-1
-1	7	16	32	-1
-1	-1	-1	-1	-1

https://leetcode.com/problems/find-a-peak-element-ii/submissions/

KoKo Eating Bananas



Koko loves to eat bananas. There are **n piles** of bananas, the ith pile has **piles[i]** bananas. The guards have gone and will come back in **h hours**.

Koko can decide her bananas-per-hour eating speed of **k**. Each hour, she chooses some pile of bananas and eats k bananas from that pile. If the pile has less than k bananas, she eats all of them instead and will **not** eat any more bananas during this hour.

Koko likes to **eat slowly** but still wants to finish eating **all** the bananas before the guards return.

Return the **minimum integer k** such that she can **eat all the bananas** within **h** hours.

Example 1:

Input: piles = [3,6,7,11],
h = 8

Output: 4

Example 2:

Input: piles = [30,11,23,4,20],
h = 5

Output: 30

How many maximum bananas Koko can eat in 1 hour?

How many Minimum bananas Koko can eat in 1 hour?

https://leetcode.com/problems/koko-eating-bananas/description/


THANK
YOU!


THANK
YOU!