

Experiment 5

Student Name Lokesh

UID 22BET10170

Branch: BE -IT

Section/Group: 702/B

Semester: 6th

DOP :21/02/2025

Subject Name: AP Lab-2

Subject Code: 22ITP-351

1. Aim 1 : Kth Largest Element in an Array

Given an integer array `nums` and an integer `k`, return *the k^{th} largest element in the array.*

Note that it is the k^{th} largest element in the sorted order, not the k^{th} distinct element. Can you solve it without sorting?

2. Merge Intervals :

Given an array of intervals where `intervals[i] = [starti, endi]`, merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input.

3. Search in Rotated Sorted Array:

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` ($1 \leq k < \text{nums.length}$) 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.*

4. Search a 2D Matrix II:

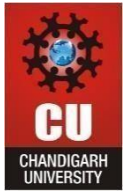
Write an efficient algorithm that searches for a value `target` in an `m x n` integer matrix `matrix`. This matrix has the following properties:

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

5. Kth Smallest Element in a Sorted Matrix:

Given an `n x n` matrix where each of the rows and columns is sorted in ascending order, return the k^{th} smallest element in the matrix. Note that it is the k^{th} smallest element in the sorted order, not the k^{th} distinct element. You must find a solution with a memory complexity better than $O(n^2)$.

6. Median of Two Sorted Arrays:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return **the median** of the two sorted arrays. The overall run time complexity should be $O(\log(m+n))$.

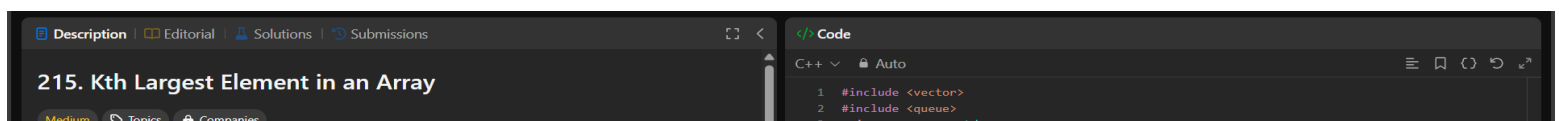
7. Sort Colors <https://leetcode.com/problems/beautiful-array/> :

Given an array `nums` with `n` objects colored red, white, or blue, sort them **in-place** so that objects of the same color are adjacent, with the colors in the order red, white, and blue. We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively. You must solve this problem without using the library's sort function.

8. Objective:

- Find the `k`th largest element efficiently without sorting the entire array.
- Given overlapping intervals, merge them into a minimal set of non-overlapping intervals.
- Find the target element in a rotated sorted array in $O(\log n)$ time.
- Search for a target efficiently in a row-wise and column-wise sorted matrix.
- Find the `k`th smallest element in a sorted matrix efficiently.
- Find the median of two sorted arrays in $O(\log(m+n))$ time.
- Sort an array containing 0s, 1s, and 2s in-place without using built-in sort functions.

9. Implementation of Code/Output 1 :





DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

10.Code 2 :

56. Merge Intervals
Medium Topics Companies

Given an array of intervals where $intervals[i] = [start_i, end_i]$, merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input.

Example 1:
Input: $intervals = [[1,3], [2,6], [8,10], [15,18]]$
Output: $[[1,6], [8,10], [15,18]]$
Explanation: Since intervals $[1,3]$ and $[2,6]$ overlap, merge them into $[1,6]$.

Example 2:
Input: $intervals = [[1,4], [4,5]]$
Output: $[[1,5]]$
Explanation: Intervals $[1,4]$ and $[4,5]$ are considered overlapping.

Constraints:

- $1 \leq intervals.length \leq 10^4$
- $intervals[i].length == 2$
- $0 \leq start_i \leq end_i \leq 10^4$

23K 177 468 Online

Code
C++ Auto

```
1 #include <vector>
2 #include <algorithm>
3 using namespace std;
4
5 class Solution {
6 public:
7     vector<vector<int>> merge(vector<vector<int>>& intervals) {
8         if (intervals.empty()) return {};
9         sort(intervals.begin(), intervals.end());
10    }
11 }
```

Saved Ln 9, Col 9

Testcase | **Test Result**

Accepted Runtime: 0 ms

Case 1 Case 2

Input

intervals =
[[1,3], [2,6], [8,10], [15,18]]

Output

[[1,6], [8,10], [15,18]]

Expected

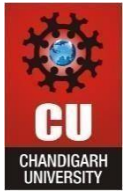
11.Code 3 :

33. Search in Rotated Sorted Array
Medium Topics Companies

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

Code
C++ Auto

```
1 #include <vector>
2 using namespace std;
3
4 class Solution {
5 public:
6     int search(vector<int>& nums, int target) {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

12.Code 4 :

Description | Editorial | Solutions | Submissions

Run Ctrl

240. Search a 2D Matrix II

Medium Topics Companies

Write an efficient algorithm that searches for a value `target` in an $m \times n$ integer matrix `matrix`. This matrix has the following properties:

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

Example 1:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

12.3K 81 71 Online

Code

C++ Auto

```
1 class Solution {
2 public:
3     bool searchMatrix(vector<vector<int>>& matrix, int target) {
4         if (matrix.empty() || matrix[0].empty()) return false;
5
6         int rows = matrix.size();
7         int cols = matrix[0].size();
8
9         int row = 0, col = cols - 1;
10
11         while (row < rows && col >= 0) {
```

Saved Ln 8, Col 5

Testcase Test Result

Accepted Runtime: 5 ms

Case 1 Case 2

Input

matrix =

[1,4,7,11,15], [2,5,8,12,19], [3,6,9,16,22], [10,13,14,17,24], [18,21,23,26,30]

target =

5

Output

19°C Cloudy

Search

ENG IN 21:17 19-02-2025

13.Code 5:

Description | Editorial | Solutions | Submissions

378. Kth Smallest Element in a Sorted Matrix

Medium Topics Companies

Given an $n \times n$ matrix where each of the rows and columns is sorted in ascending order, return the k^{th} smallest element in the matrix.

Note that it is the k^{th} smallest element in the sorted order, not the k^{th} distinct element.

You must find a solution with a memory complexity better than $O(n^2)$.

Example 1:

Input: matrix = [[1,5,9],[10,11,13],[12,13,15]], k = 8

Output: 13

Code

C++ Auto

```
1 #include <vector>
2 using namespace std;
3
4 class Solution {
5 public:
6     int kthSmallest(vector<vector<int>>& matrix, int k) {
7         int n = matrix.size();
8         int low = matrix[0][0], high = matrix[n-1][n-1];
9
10         while (low < high) {
11             int mid = low + (high - low) / 2;
```

Saved Ln 1, Col 1

Testcase Test Result



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

15.Code 6 :

DescriptionEditorialSolutionsSubmissions

4. Median of Two Sorted Arrays

HardTopicsCompanies

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return **the median** of the two sorted arrays.

The overall run time complexity should be $O(\log(m+n))$.

Example 1:

Input: `nums1 = [1,3]`, `nums2 = [2]`
Output: `2.00000`
Explanation: merged array = `[1,2,3]` and median is `2`.

Example 2:

Input: `nums1 = [1,2]`, `nums2 = [3,4]`
Output: `2.50000`
Explanation: merged array = `[1,2,3,4]` and median is $(2 + 3) / 2 = 2.5$.

Constraints:

- `nums1.length == m`
- `nums2.length == n`

29.6K615635 Online

Code

C++Auto

```
1 #include <vector>
2 #include <limits.h>
3 using namespace std;
4
5 class Solution {
6 public:
7     double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
8         if (nums1.size() > nums2.size()) {
9             return findMedianSortedArrays(nums2, nums1);
10        }
11    }
```

SavedLn 9, Col 59

TestcaseTest Result

AcceptedRuntime: 0 ms

Case 1Case 2

Input

nums1 =
[1,3]

nums2 =
[2]

Output

16.Code 7 :

DescriptionEditorialSolutionsSubmissions

75. Sort Colors

MediumTopicsCompaniesHint

Given an array `nums` with `n` objects colored red, white, or blue, sort them **in-place** so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers `0`, `1`, and `2` to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

Example 1:

Input: `nums = [2,0,2,1,1,0]`
Output: `[0,0,1,1,2,2]`

Code

C++Auto

```
1 #include <vector>
2 using namespace std;
3
4 class Solution {
5 public:
6     void sortColors(vector<int>& nums) {
7         int low = 0, mid = 0, high = nums.size() - 1;
8
9         while (mid <= high) {
10             if (nums[mid] == 0) {
11                 swap(nums[low], nums[mid]);
```

SavedLn 23, Col 1

TestcaseTest Result



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

17.Learning Outcome:

- Using a Min-Heap (Priority Queue) to maintain the k largest elements.
- QuickSelect (Hoare's Selection Algorithm) for finding the kth largest element in $O(n)$ average time complexity.
- Sorting + Merging Technique to process overlapping intervals.
- Binary Search in a Rotated Array.
- Matrix traversal from the top-right or bottom-left for $O(m + n)$ complexity.
- Using a Min-Heap to extract the smallest k elements efficiently.
- Optimal $O(\log(\min(m, n)))$ solution instead of naive merging ($O(m+n)$).
- Three-way partitioning using three pointers (low, mid, high).