

Experiment 5

Student Name: Rahul Kr Singh

UID:22BET10284

Branch: BE -IT

Section/Group:22BET/IOT/702/B

Semester: 6th

Date of Performance:21/02/2025

Subject Name: Advanced Programming 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:

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 $O(\log(m+n))$.

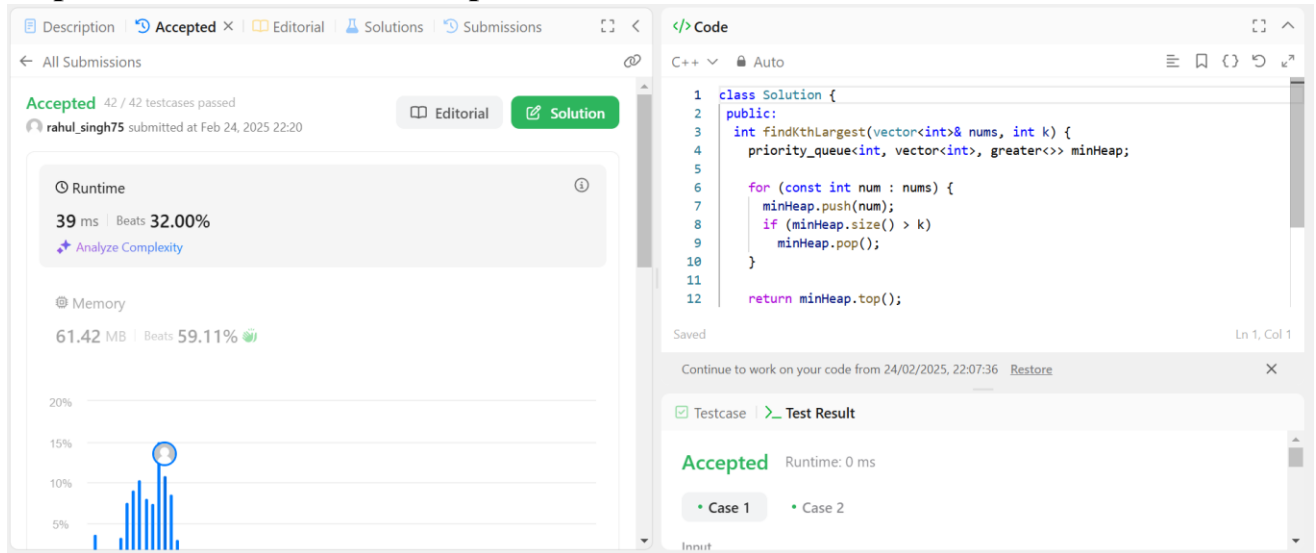
7. Sort Colors :

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.

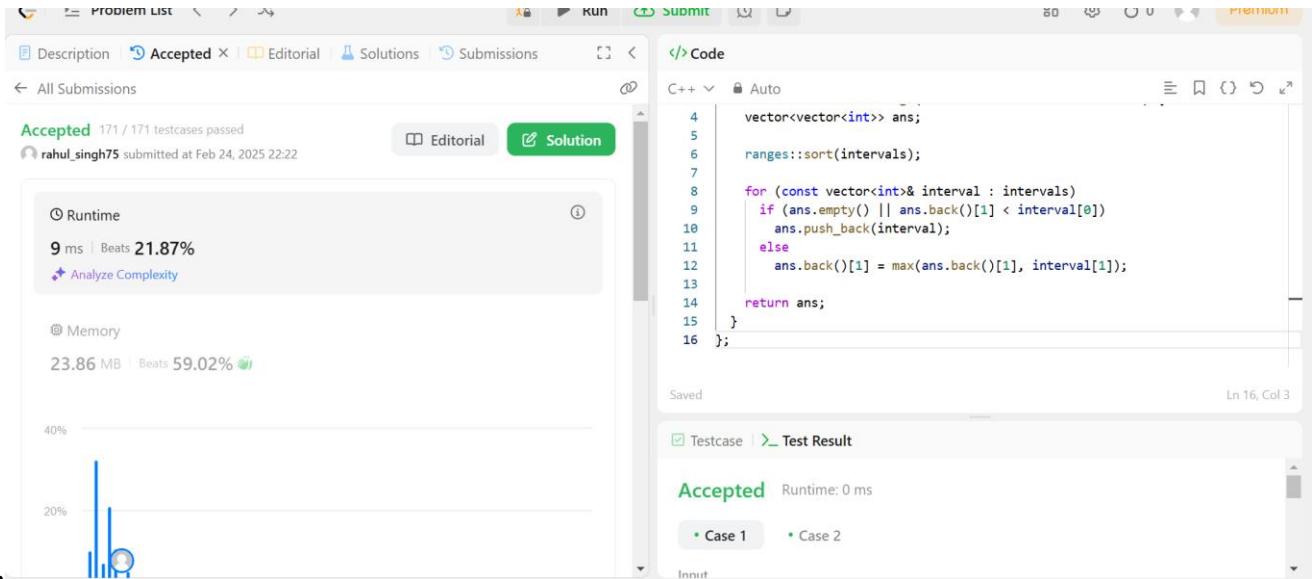
9. Implementation of Code/Output 1 :



The screenshot displays a coding platform interface with the following details:

- Submission Status:** Accepted (42 / 42 testcases passed). Submitted by `rahul_singh75` on Feb 24, 2025 at 22:20.
- Runtime:** 39 ms | Beats 32.00%. A link to [Analyze Complexity](#) is provided.
- Memory:** 61.42 MB | Beats 59.11%.
- Code Editor:** C++ code for finding the kth largest element using a min-heap.
- Test Results:** Accepted (Runtime: 0 ms). Cases 1 and 2 are listed.

```
1 class Solution {
2 public:
3     int findKthLargest(vector<int>& nums, int k) {
4         priority_queue<int, vector<int>, greater<>> minHeap;
5
6         for (const int num : nums) {
7             minHeap.push(num);
8             if (minHeap.size() > k)
9                 minHeap.pop();
10        }
11
12        return minHeap.top();
13    }
14 }
```



Problem List

Description Accepted Editorial Solutions Submissions

All Submissions

Accepted 171 / 171 testcases passed

rahul_singh75 submitted at Feb 24, 2025 22:22

Editorial Solution

Runtime

9 ms | Beats 21.87%

Analyze Complexity

Memory

23.86 MB | Beats 59.02%

40%

20%

Code

```
4 vector<vector<int>> ans;
5
6 ranges::sort(intervals);
7
8 for (const vector<int>& interval : intervals)
9     if (ans.empty() || ans.back()[1] < interval[0])
10        ans.push_back(interval);
11    else
12        ans.back()[1] = max(ans.back()[1], interval[1]);
13
14 return ans;
15 }
16 ;;
```

Saved

Ln 16, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

10.
Code 2 :



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

11.Code 3 :

Accepted 196 / 196 testcases passed

rahul_singh75 submitted at Feb 24, 2025 22:25

[Editorial](#) [Solution](#)

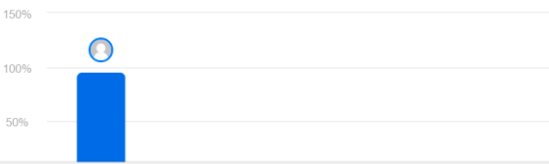
Runtime

0 ms | Beats 100.00%

[Analyze Complexity](#)

Memory

15.31 MB | Beats 3.54%



```
14         else
15             l = m + 1;
16     } else { // nums[m..n - 1] are sorted.
17         if (nums[m] < target && target <= nums[r])
18             l = m + 1;
19         else
20             r = m - 1;
21     }
22 }
23
24 return -1;
25 }
26 ;
```

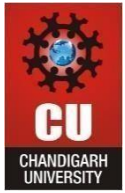
Saved Ln 26, Col 3

☒ Testcase [Test Result](#)

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

12.Code 4 :

Accepted 130 / 130 testcases passed

rahul_singh75 submitted at Feb 24, 2025 22:27

Runtime: 46 ms | Beats 83.92%

Memory: 18.79 MB | Beats 37.03%

```
7 while (r < matrix.size() && c >= 0) {
8     if (matrix[r][c] == target)
9         return true;
10    if (matrix[r][c] > target)
11        --c;
12    else
13        ++r;
14 }
15
16 return false;
17 }
18 };
19
```

Accepted Runtime: 3 ms

Case 1 Case 2



13.Code 5 :

← All Submissions

Accepted 87 / 87 testcases passed

rahul_singh75 submitted at Feb 24, 2025 22:31

Editorial Solution

Runtime 20 ms Beats 12.16% [Analyze Complexity](#)

Memory 17.10 MB Beats 84.45%

```
14 minHeap.emplace(i, 0, matrix[i][0]);
15
16 while (k-- > 1) {
17     const auto [i, j, _] = minHeap.top();
18     minHeap.pop();
19     if (j + 1 < matrix[0].size())
20         minHeap.emplace(i, j + 1, matrix[i][j + 1]);
21 }
22
23 return minHeap.top().num;
24 }
25 };
26
```

Saved Ln 26, Col 1

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

14.Code 6 :

The screenshot displays a code submission interface for a C++ problem. The top navigation bar includes tabs for Description, Accepted (selected), Editorial, Solutions, and Submissions. Below this, the submission status is "Accepted" with 2096 / 2096 testcases passed, submitted by rahul_singh75 on Feb 24, 2025 at 22:33. The interface is divided into three main sections: Runtime, Memory, and Code.

Runtime: Shows 0 ms execution time, beating 100.00% of other solutions. A link to "Analyze Complexity" is available.

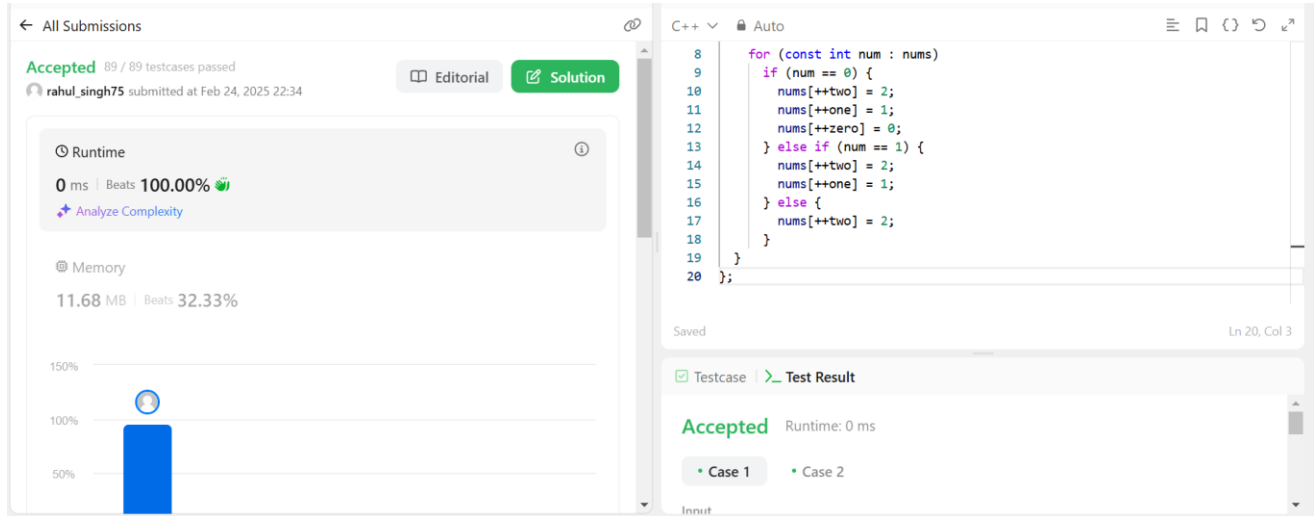
Memory: Shows 95.11 MB memory usage, beating 64.03% of other solutions.

Code: The C++ code is displayed in a syntax-highlighted editor. It implements a recursive function to calculate the maximum sum of a subarray of size k. The code is as follows:

```
20     return (n1 + n2) % 2 == 0
21         ? (max(maxLeft1, maxLeft2) + min(minRight1, minRight2)) *
22           0.5
23         : max(maxLeft1, maxLeft2);
24     else if (maxLeft1 > minRight2)
25         r = partition1 - 1;
26     else
27         l = partition1 + 1;
28     }
29     throw;
30 }
31 ;;
```

Below the code editor, the "Testcase" tab is selected, showing "Accepted" status with a runtime of 0 ms. Two test cases are listed: Case 1 and Case 2.

15.Code 7 :



All Submissions

Accepted 89 / 89 testcases passed

rahul_singh75 submitted at Feb 24, 2025 22:34

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

11.68 MB | Beats 32.33%

150%

100%

50%

C++

```
8 for (const int num : nums)
9     if (num == 0) {
10         nums[++two] = 2;
11         nums[++one] = 1;
12         nums[++zero] = 0;
13     } else if (num == 1) {
14         nums[++two] = 2;
15         nums[++one] = 1;
16     } else {
17         nums[++two] = 2;
18     }
19 }
20 ;
```

Saved

Ln 20, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

16.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.
- Inary Search in a Rotated Array.
- Matrix traversal from the top-right or bottom-left for $O(m + n)$ complexity.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

- 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).