



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

WORKSHEET - 5

Student Name: Mayank Bhardwaj

UID: 22BCS10988

Branch: CSE

Section/Group: IOT_640-B

Semester: 6th

Date of Performance: 24/02/2025

Subject Name: AP-II

Subject Code:22CP_351

1. Aim:- Write a program for Sorting and searching, which are fundamental concepts in computer science

2. Source Code:-

88. Merge Sorted Array:--

CODE:---

```
import java.util.Arrays;
```

```
public class Solution {
```

```
    public void merge(int[] nums1, int m, int[] nums2, int n) {
```

```
        // Pointers for nums1, nums2, and the end of merged array int p1 = m -
```

```
        1;                                // Last element in nums1 (excluding zeros) int p2
```

```
        = n - 1;                        // Last element in nums2
```

```
        int p = m + n - 1;              // Last position in nums1 (including zeros)
```

```
        // Merge from the back to avoid overwriting values in nums1 while (p1  
>= 0 && p2 >= 0) {
```

```
            if (nums1[p1] > nums2[p2]) {
```

```
                nums1[p] = nums1[p1]; // Place larger value at the end p1--;
```

```
            } else {
```

```
                nums1[p] = nums2[p2]; p2--;
```

```
            };
```

```
        }
```

```
        p--;  
    }  
  
    // If any elements are left in nums2, copy them while (p2 >=  
    0) {  
        nums1[p] = nums2[p2]; p2-  
        -;  
        p--;  
    }  
}  
  
public static void main(String[] args) {  
    Solution solution = new Solution();  
  
    // Example input  
    int[] nums1 = {1, 2, 3, 0, 0, 0};  
    int m = 3;  
    int[] nums2 = {2, 5, 6}; int n  
    = 3;  
  
    System.out.println("Original nums1: " + Arrays.toString(nums1));  
    solution.merge(nums1, m, nums2, n);  
    System.out.println("Merged Array: " + Arrays.toString(nums1));  
}  
}
```

347. Top K Frequent Elements:--

Code:--

```
import java.util.*; public  
  
class Solution {  
    public int[] topKFrequent(int[] nums, int k) {
```

```
// Frequency map to count occurrences of each element
Map<Integer, Integer> freqMap = new HashMap<>(); for (int
num : nums) {
    freqMap.put(num, freqMap.getOrDefault(num, 0) + 1);
}

// Priority queue (min-heap) based on frequency PriorityQueue<Integer> minHeap
= new PriorityQueue<>((a, b) ->
freqMap.get(a) - freqMap.get(b));

// Add elements to the heap and maintain its size as k for (int
num : freqMap.keySet()) {
    minHeap.add(num);
    if (minHeap.size() > k) {
        minHeap.poll(); // Remove least frequent element
    }
}

// Prepare result array int[]
result = new int[k]; int index
= 0;
while (!minHeap.isEmpty()) {
    result[index++] = minHeap.poll();
}

return result;
}

public static void main(String[] args) {
    Solution solution = new Solution();

    int[] nums1 = {1, 1, 1, 2, 2, 3};
    int k1 = 2;
    System.out.println("Output for Example 1: " +
Arrays.toString(solution.topKFrequent(nums1, k1))); // Output: [1, 2]

    int[] nums2 = {1};
```

```
        int k2 = 1;
        System.out.println("Output for Example 2: " +
Arrays.toString(solution.topKFrequent(nums2, k2))); // Output: [1]
    }
}
```

56. Merge Intervals:---

Code:--

```
import java.util.*;

public class Solution {
    public int[][] merge(int[][] intervals) { if
        (intervals.length <= 1) {
            return intervals;
        }

        // Sort intervals based on the start time Arrays.sort(intervals, (a, b) ->
        Integer.compare(a[0], b[0]));

        List<int[]> merged = new ArrayList<>();

        // Initialize the first interval int[]
        current = intervals[0];
        merged.add(current);

        // Iterate through intervals for
        (int[] interval : intervals) {
            if (interval[0] <= current[1]) {
                // Overlapping intervals, merge them current[1] =
                Math.max(current[1], interval[1]);
            } else {
                // No overlap, move to the next interval
                current = interval; merged.add(current);
            }
        }
    }
}
```

```
}

// Convert list to array
return merged.toArray(new int[merged.size()]());
}

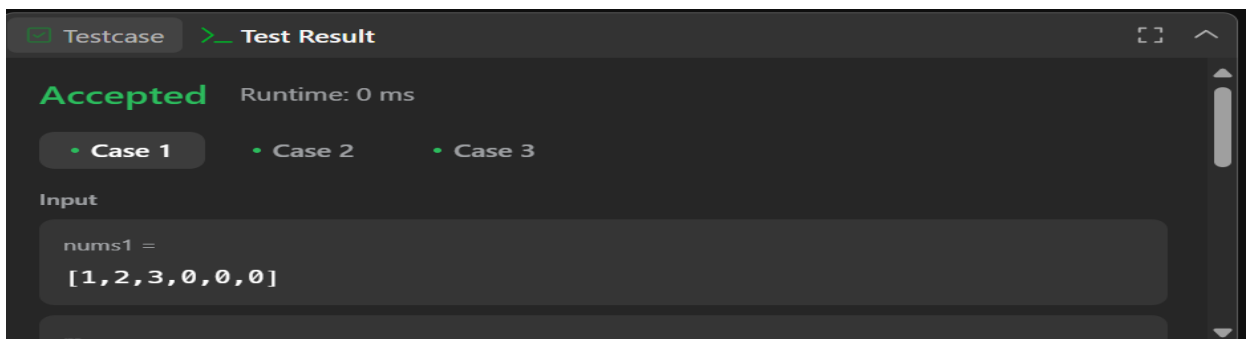
public static void main(String[] args) {
    Solution solution = new Solution();

    int[][] intervals1 = {{1, 3}, {2, 6}, {8, 10}, {15, 18}};
    System.out.println("Output for Example 1: " +
        Arrays.deepToString(solution.merge(intervals1)));
    // Output: [[1,6],[8,10],[15,18]]

    int[][] intervals2 = {{1, 4}, {4, 5}}; System.out.println("Output for
        Example 2: " +
        Arrays.deepToString(solution.merge(intervals2)));
    // Output: [[1,5]]
}
}
```

2. Screenshot of Outputs:

88. Merge Sorted Array:--



347. Top K Frequent Elements:--

☒ Testcase | [Test Result](#)

Accepted Runtime: 0 ms

• Case 1

• Case 2

Input

nums =
[1,1,1,2,2,3]

k =
2

56. Merge Intervals:---

☒ Testcase | [Test Result](#)

Accepted Runtime: 0 ms

• Case 1

• Case 2

Input

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

Output