Experiment 1.4

Name: Parvesh **UID:** 22BCS13548
Branch: BE CSE **Section:**22BCS_IOT_624_A
Semester: 6th **DOP:**12/02/2025
Subject: Advanced Programming Lab II **Subject Code:** 22CSP-351

1) Aim: To efficiently merge two sorted arrays nums1 and nums2 into nums1 in a nondecreasing order without using extra space.

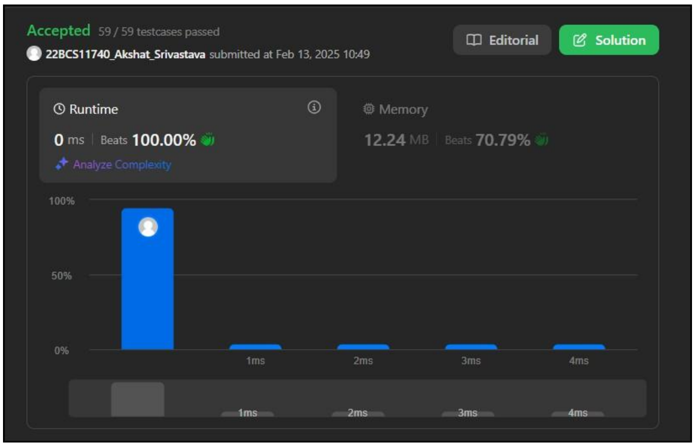
Objective:

- Understand and implement the **two-pointer approach** to merge two sorted arrays efficiently.
- Modify `nums1` **in-place**, ensuring the correct order is maintained.
- Optimize the merging process by starting from the **end of the arrays** to avoid unnecessary shifting.

Code:

```
class Solution { public:    void merge(vector<int>& nums1, int m,

vector<int>& nums2, int n) {

        int i = m - 1;

        int j = n - 1;

        int k = m + n - 1;

        while (i >= 0 && j >= 0) {

            if (nums1[i] > nums2[j]) {

                nums1[k] = nums1[i];
                i--;

            } else {

                nums1[k] = nums2[j];

                j--;

}

            k--; }

        while (j >= 0) {

            nums1[k] = nums2[j];

            j--;

            k--;

        }}};
```

Output:



Accepted 59 / 59 testcases passed

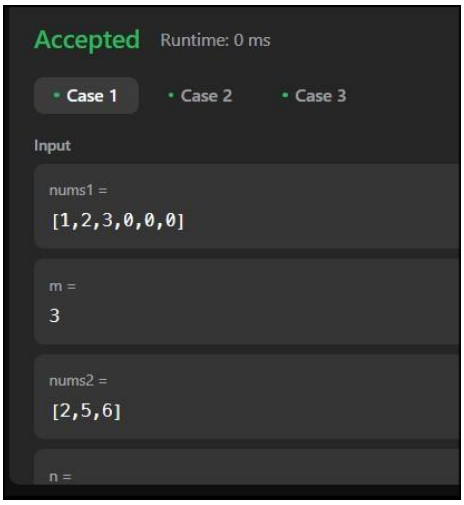
22BCS11740_Akshat_Srivastava submitted at Feb 13, 2025 10:49

Runtime

0 ms | Beats 100.00%

Memory

12.24 MB | Beats 70.79%



Accepted Runtime: 0 ms

Input

nums1 =

[1,2,3,0,0,0]

m =

3

nums2 =

[2,5,6]

n =

Learning Outcome:

- Gain an understanding of **in-place merging** in arrays.
- Learn how to use the **two-pointer technique** efficiently.
- Improve problem-solving skills by **handling edge cases** like empty arrays and different lengths.
- Enhance your ability to **write optimized algorithms** with **O(m + n) time complexity** and **O(1) space complexity**.

2) Aim: To efficiently find the first bad version using the **minimum number of API calls** by implementing an optimized search algorithm.

Objective:

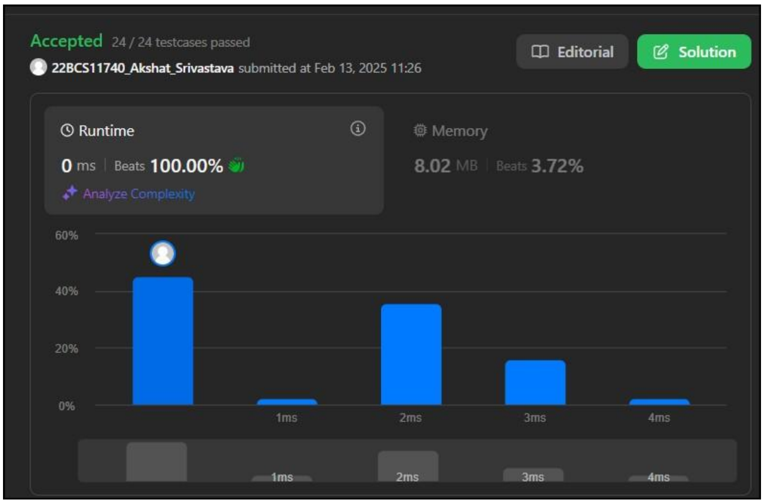
- Utilize **binary search** to minimize API calls while searching for the first bad version.
- Reduce the search space efficiently instead of checking each version sequentially.
- Implement an **O(log n) solution** instead of a naive **O(n) approach**.

Code:

```
class Solution { public:     int
firstBadVersion(int n) {
        int left = 1, right = n;

        while (left < right) {              int
mid = left + (right - left) / 2;

            if (isBadVersion(mid)) {
                right = mid;
            } else {
                left = mid + 1;
            }
}
          return left;
```

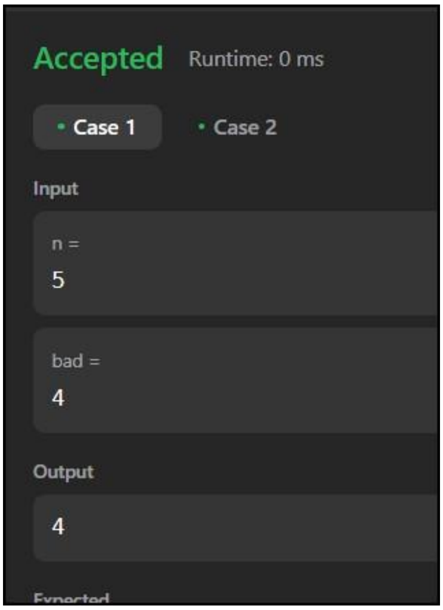
```
  } };
```

**Outp
ut:**



Accepted 24 / 24 testcases passed

22BCS11740_Akshat_Srivastava submitted at Feb 13, 2025 11:26



Accepted Runtime: 0 ms

Input

n =
5

bad =
4

Output

4

Expected

Learning Outcome:

- Understand **binary search** and its applications in optimization problems.
- Learn how to **interact with external APIs** efficiently.
- Improve problem-solving skills by working with **search space reduction techniques**.
- Learn how to handle **edge cases** where the first version itself is bad.

3) Aim: To implement an efficient **in-place sorting algorithm** to sort an array containing three distinct values (0, 1, and 2) without using built-in sorting functions.

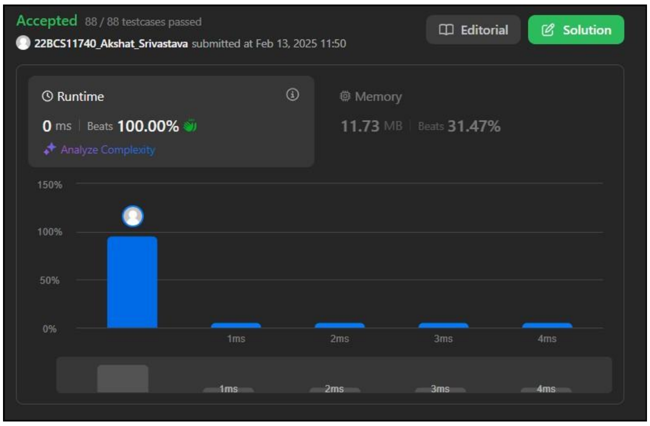
Objective:

- To understand and apply Dutch National Flag Algorithm for sorting a three-element array efficiently.
- To sort the array in O(n) time complexity using a constant amount of extra space (O(1)).
- To learn how to manipulate array elements in-place while maintaining correct order.
- To practice optimizing sorting algorithms for real-world problems.
-

Code:

```
class Solution { public:     void

sortColors(vector<int>& nums) {

     int low = 0, mid = 0, high = nums.size() - 1;

    while (mid <= high) {

      if (nums[mid] == 0) {

        swap(nums[mid], nums[low]);

        low++;

        mid++;

      }

      else if (nums[mid] == 1) {

        mid++;

}

      else {

        swap(nums[mid], nums[high]);

        high--;

} } } };
```

Output:



Accepted 88 / 88 testcases passed

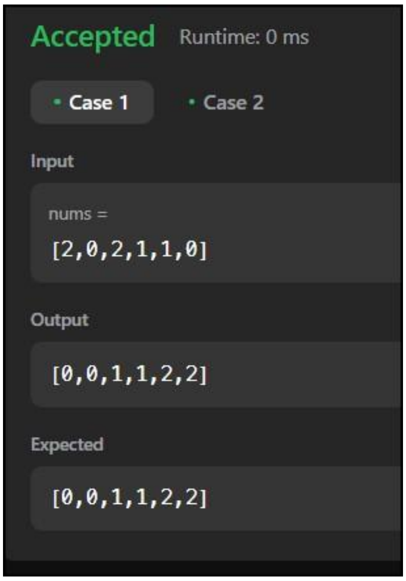
22BCS11740_Akshat_Srivastava submitted at Feb 13, 2025 11:50

Runtime

0 ms | Beats 100.00%

Memory

11.73 MB | Beats 31.47%



Accepted Runtime: 0 ms

Input

```
nums =
[2,0,2,1,1,0]
```

Output

```
[0,0,1,1,2,2]
```

Expected

```
[0,0,1,1,2,2]
```

Learning Outcome:

- **Understanding of the Dutch National Flag Algorithm** and how it partitions an array into three sections.
- **Efficient array manipulation techniques** for solving problems in-place.
- **Comparison of different sorting approaches** (counting sort, two-pass sort, one-pass sort).
- **Improved problem-solving skills** in competitive programming and technical interviews.