AP Assignment 5

Name : Chirag Yadav

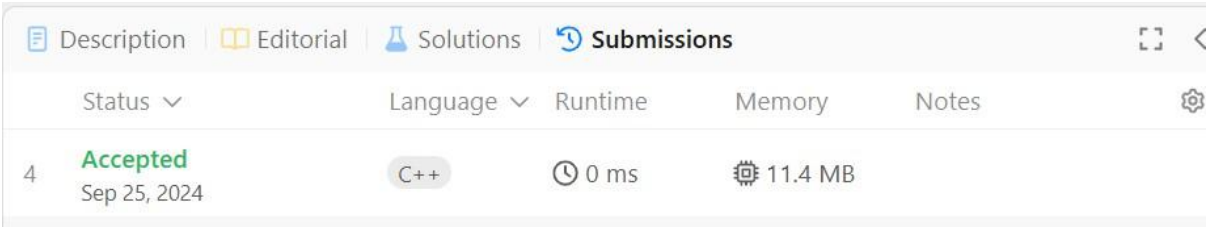
UID : 22BCS16377

Section : 608-B

1) Merge Sorted Array

```
#include <iostream>
#include <vector>
using namespace std;

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--];
            } else {
                nums1[k--] = nums2[j--];
            }
        }

        while (j >= 0) {
            nums1[k--] = nums2[j--];
        }
    }
};

int main() {
    Solution sol;
    vector<int> nums1 = {1, 2, 3, 0, 0, 0};
    vector<int> nums2 = {2, 5, 6};
    int m = 3, n = 3;
```

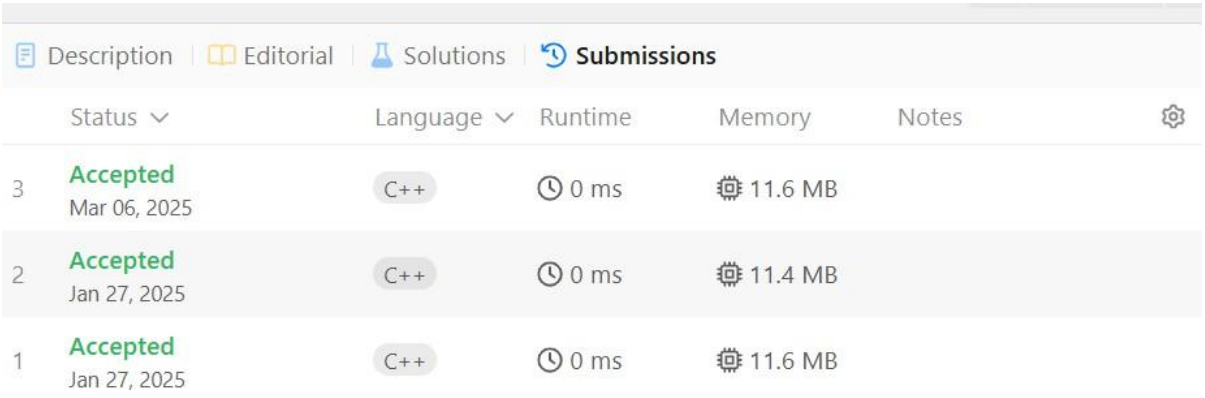
```
    sol.merge(nums1, m, nums2, n);

    cout << "Merged Array: ";
    for (int num : nums1) {
        cout << num << " ";
    }

    return 0;
}
```



2) Sort Colors

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int n = nums.size();
        int j = 0, k = 0, l = 0;
        for (int i = 0; i < n; i++) {
            if (nums[i] == 0) {
                j++;
            } else if (nums[i] == 1) {
                k++;
            } else {
                l++;
            }
        }
        int index = 0;

        while (j--) {
```

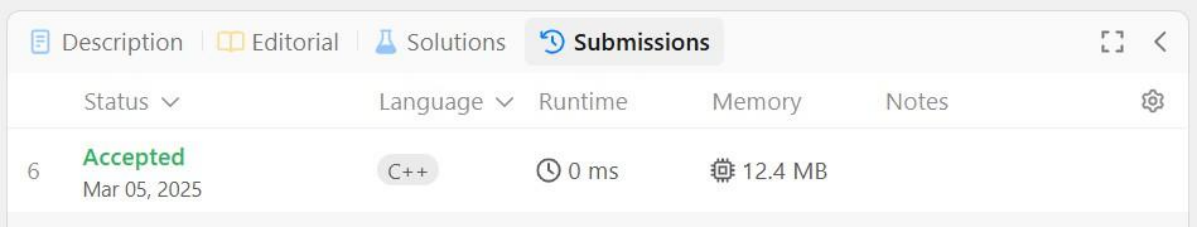
```
            nums[index++] = 0;
        }
        while (k--) {
            nums[index++] = 1;
        }
        while (l--) {
            nums[index++] = 2;
        }
    }
};
```



3) Find Peak Element

```
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        int n = nums.size();
        int low = 0, high = n - 1;

        while (low < high) {
            int mid = low + (high - low) / 2;
            if (nums[mid] > nums[mid + 1]) {
                high = mid;
```

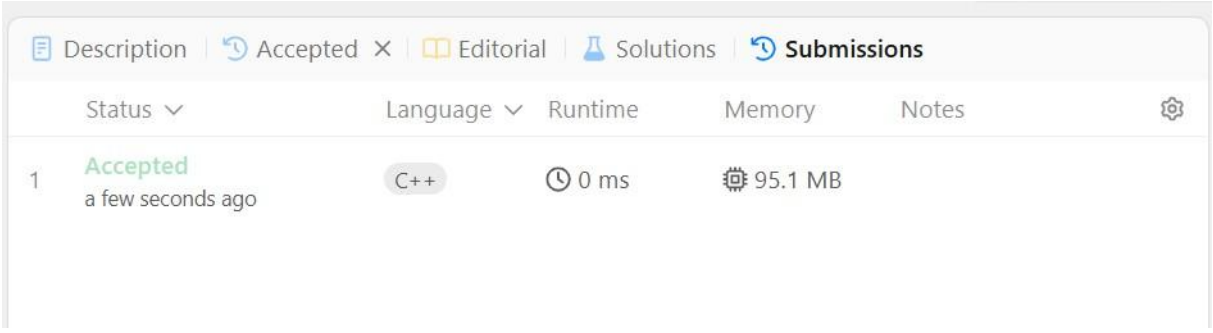
```
        } else {
            low = mid + 1;
        }
    }
    return low;
  }
};
```



4) Median of two sorted Array

```
class Solution {
public:
  double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
    if (nums1.size() > nums2.size()) {
      return findMedianSortedArrays(nums2, nums1);
    }

    int m = nums1.size();
    int n = nums2.size();
    int low = 0, high = m;

    while (low <= high) {
      int partition1 = low + (high - low) / 2;
      int partition2 = (m + n + 1) / 2 - partition1;

      int maxLeft1 = (partition1 == 0) ? INT_MIN : nums1[partition1 - 1];
```

```
            int minRight1 = (partition1 == m) ? INT_MAX : nums1[partition1];

            int maxLeft2 = (partition2 == 0) ? INT_MIN : nums2[partition2 - 1];
            int minRight2 = (partition2 == n) ? INT_MAX : nums2[partition2];

            if (maxLeft1 <= minRight2 && maxLeft2 <= minRight1) {
                if ((m + n) % 2 == 0) {
                    return (max(maxLeft1, maxLeft2) + min(minRight1, minRight2)) / 2.0;
                } else {
                    return max(maxLeft1, maxLeft2);
                }
            } else if (maxLeft1 > minRight2) {
                high = partition1 - 1;
            } else {
                low = partition1 + 1;
            }
        }
        throw invalid_argument("Input arrays are not sorted.");
    }
};
```



5) First Bad Version

bool isBadVersion(int version);

```
class Solution {
public:
    int firstBadVersion(int n) {
        int low = 1, high = n;
        while (low < high) {
            int mid = low + (high - low) / 2;
            if (isBadVersion(mid)) {
                high = mid;
            } else {
                low = mid + 1;
            }
        }
        return low
    }
};
```