Experiment5

Student Name: Rishi soni **UID: 22BCS12768**

Branch: BE-CSE **Section/Group:NTPP_603B**

Semester:6th **DateofPerformance:20/02/25**

SubjectName:Ap **SubjectCode:22CSH-359**

1. Aim:Sort Colors

2. **Objective:**Givenanarraynumswithnobjectscoloredred,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.

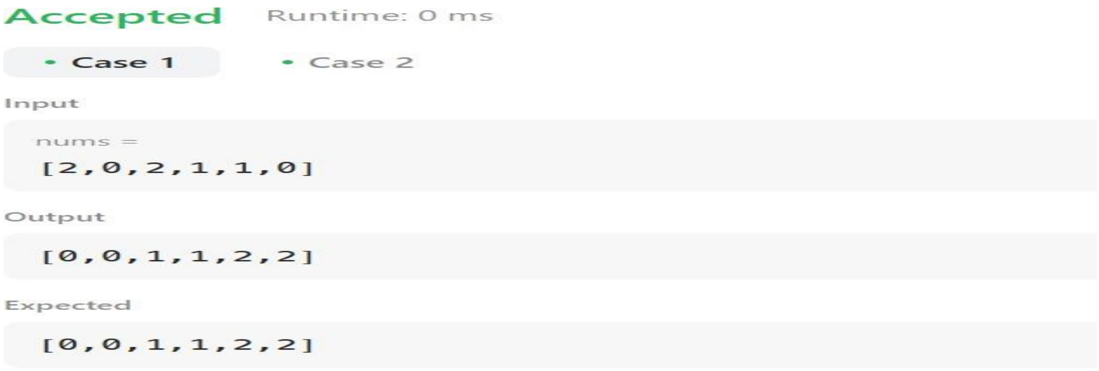
   Wewillusetheintegers 0,1,and2torepresentthecolorred,white, and blue, respectively.

   Youmustsolvethisproblemwithoutusingthelibrary'ssort function.

3. Code:

```
voidsortColors(vector<int>&nums) {

        intlow=0,mid=0,high=nums.size() -1; while

        (mid <= high) {

            if (nums[mid] == 0) {

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

            }elseif(nums[mid]==1){

                mid++;

            }else {

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

            }
        }

    }
```

4. **Output:**



Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

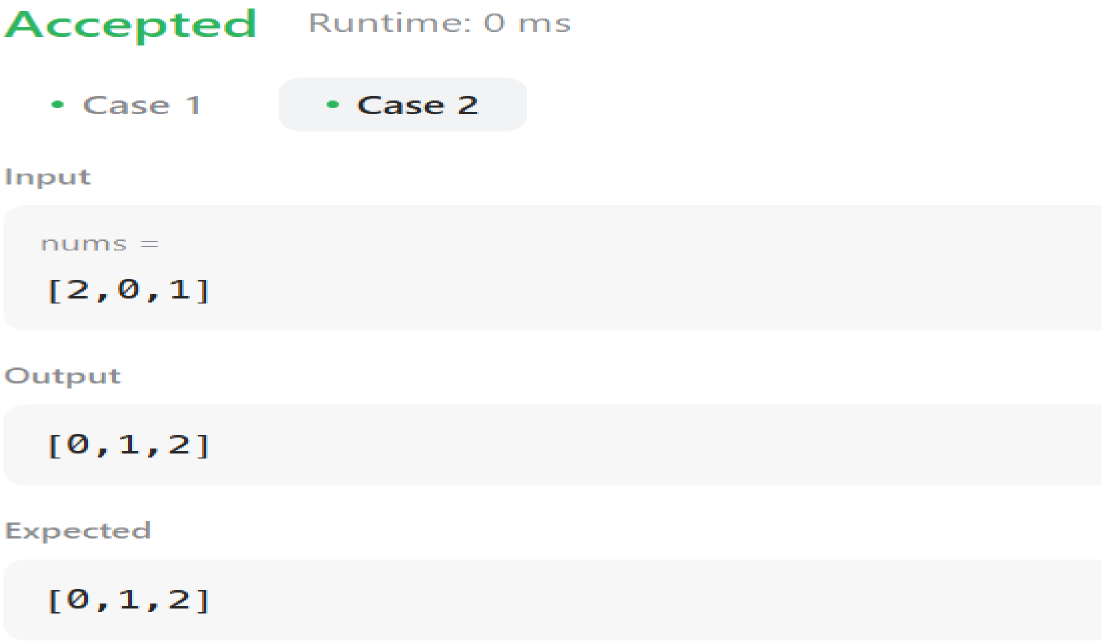
```
nums =
[2,0,2,1,1,0]
```

Output

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

Expected

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



Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

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

Output

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

Expected

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

5. LearningOutcomes

- UseofBinarySearchApproachinit.
- UseofSwapFunctoninit.
- UseofPointerApproachinit.

1. **Aim**:-**MedianofTwo SortedArrays**

2. **Objective**:-Giventwosortedarrays nums1andnums2 of sizemandnrespectively,return **themedian**ofthetwosortedarrays. The overall run time complexity should be O(log (m+n)).

3. **Code**:-

```
 voidmerge(vector<int>&nums1,vector<int>&nums2,vector<longlong>&v){

        int k =
        0;intidx1=0
        ; intidx2 =0;

        while(idx1<nums1.size()andidx2<nums2.size()){

            if(nums1[idx1]<=nums2[idx2]){
                v.push_back(nums1[idx1++]);
            }else{
                v.push_back(nums2[idx2++]);
            }

        }

        while(idx1 < nums1.size()){
            v.push_back(nums1[idx1++]);
        }
        while(idx2 < nums2.size()){
            v.push_back(nums2[idx2++]);
        }

    }
public:
    doublefindMedianSortedArrays(vector<int>&nums1,vector<int>&nums2){

        vector<long long > v;

        merge(nums1,nums2,v);

        int index =v.size()/ 2;

        if(v.size()%2!=0){ return
            v[index];
        }else{

            return(double)(v[index]+v[index -1])/2;

        }
    }
```

4. **OutPut**:-

```
Case1
Input
nums1=[1,2]
nums2=[3,4]
Output
2.50000
Expected
2.50000

Input
nums1=[1,2]
nums2=[3,4]
Output
2.50000
Expected
2.50000
```

5. **LearningOutcomes:-**

   1. UsingMergeSortApproach.
   2. wecanmakeitasanidealapproachin it.
   3. usingoffunctioncall.

1. **Aim:-**KthSmallestElementinaSorted Matrix

2. **Objective:-**Givenannxnmatrixwhereeachoftherowsandcolumnsis sorted in ascending order, return *the* kth*smallest element in the matrix*.
   Notethatitisthekthsmallestelementinthesortedorder,not the kth
   distinct element.
   YoumustfindasolutionwithamemorycomplexitybetterthanO(n^2).

3. **Code:-**

```
intkthSmallest(vector<vector<int>>&matrix,intk){

      priority_queue<int>ans;
      for(int i=0;i<matrix.size();i++){
          for(intj=0;j<matrix.size();j++){

             if(ans.size()<k){
              ans.push(matrix[i][j]);
             }else{

              if(matrix[i][j]<ans.top()){
                  ans.pop();
                  ans.push(matrix[i][j]);
              }
             }
          }
      }
      returnans.top();}
```

4. **Output:-**

Case1
Input
matrix=[[1,5,9],[10,11,13],[12,13,15]]
k = 8
Output
13
Expected
13

Input
matrix=[[-5]]
k =1
Output
-5
Expected
-5

5. Learning Outcomes:-

1. UseofPriorityQueue.
2. Useofneastedloop.
3. Usingtheapproachofqueue.