DSA SERIES

- Learn Coding

Questions to be Covered today

1. **Sort an Array**
2. **Merge Sorted Array**
3. **Maximum Product of three numbers**
4. **Largest Number At Least Twice of Others**
5. **Minimum Absolute Difference**
6. **Average Salary Excluding The Minimum and Maximum Salary**

7. Mean of Array After Removing Some Element

8. Count Elements with strictly smaller and greater element

9. Maximum Consecutive Floors Without Special Floors

10. Remove Covered Intervals

LETS START TODAY'S LECTURE

LEETCODE - 912

Sort an Array

Difficulty : Medium

Problem statement :

Given an array of integers nums, sort the array in ascending order and return it.

You must solve the problem without using any built-in functions in O(nlog(n)) time complexity and with the smallest space complexity possible.

Code :-

```
class Solution {
public:
    void merge(vector<int> &nums,int left ,int mid,int right){
        vector<int> temp;
        int i = left;
        int j = mid+1;

        while(i<=mid && j<=right){
            if(nums[i]<=nums[j]){
                temp.push_back(nums[i++]);
            } else{
                temp.push_back(nums[j++]);
            }
        }

        while(i<=mid){
            temp.push_back(nums[i++]);
        }

        while(j<=right){
             temp.push_back(nums[j++]);
        }
```

```
        for(int i = left ;i<=right;i++){
            nums[i]=temp[i-left];
        }

    }
    void mergeSort(vector<int> &nums,int left,int right){
        if(left >= right){
            return ;
        }

        int mid = (left+right)/2;
        mergeSort(nums,left,mid);
        mergeSort(nums,mid+1,right);

        merge(nums,left,mid,right);
    }
    vector<int> sortArray(vector<int>& nums) {
        int n = nums.size();
        mergeSort(nums, 0,n-1);

        return nums;

    }
};
```

LEETCODE - 88

Merge Sorted Array

Difficulty : Easy

You are given two integer arrays nums1 and nums2, sorted in **non-decreasing order**,
and two integers m and n, representing the number of elements in nums1 and nums2 respectively.
Merge nums1 and nums2 into a single array sorted in **non-decreasing order**.

The final sorted array should not be returned by the function,
but instead be *stored inside the array* nums1.
To accommodate this, nums1 has a length of m + n,
where the first m elements denote the elements that should be merged,
and the last n elements are set to 0 and should be ignored. nums2 has a length of n.

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int i = m-1; // last index of first vector
        int j = n-1; // last index of second vector
        int k = m+n-1; // last index of num1(complete)

        while(i>=0 && j>=0){
            if(nums1[i] <= nums2[j]){
                nums1[k--]= nums2[j--];
                // k--;
                // j--;
            } else {
                nums1[k--]=nums1[i--];
            }
        }

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

LEETCODE - 628

Maximum Product of three numbers

Difficulty : Easy

Given an integer array nums, *find three numbers whose product is maximum and return the maximum product*.

Example 1:
Input: nums = [1,2,3] **Output:** 6

Example 2:
Input: nums = [1,2,3,4] **Output:** 24

Example 3:
Input: nums = [-1,-2,-3] **Output:** -6

```
class Solution {
public:
    int maximumProduct(vector<int>& nums) {
        int n = nums.size();
        sort(nums.begin(), nums.end());

        int p1 = nums[n - 1] * nums[n - 2] * nums[n - 3];
        int p2 = nums[0] * nums[1] * nums[n - 1];

        return max(p1, p2);
    }
};
```

LEETCODE - 747

Largest Number At Least Twice of Others

Difficulty : Easy

You are given an integer array nums where the largest integer is **unique**.
Determine whether the largest element in the array is **at least twice**
as much as every other number in the array. If it is, return *the **index** of the largest element, or return* -1 *otherwise*.

Example 1:

Input: nums = [3,6,1,0]
Output: 1

Explanation: 6 is the largest integer.
For every other number in the array x, 6 is at least twice as big as x. The index of value 6 is 1, so we return 1.

```
class Solution {
public:
    int dominantIndex(vector<int>& nums) {
        int n = nums.size();

        int maxVal =-1;
        int maxIdx =-1;

        for(int i = 0;i<n;i++){
            if(nums[i]>maxVal){
                maxVal = nums[i];
                maxIdx = i;
            }
        }

        for(int i =0;i<n;i++){
            if(i != maxIdx && maxVal < 2* nums[i]){
                return -1;
            }
        }

        return maxIdx;
    }
};
```

LEETCODE - 1200

Minimum Absolute Difference

Difficulty : Easy

Given an array of **distinct** integers arr,
find all pairs of elements with the minimum absolute difference of any two elements.

Return a list of pairs in ascending order(with respect to pairs), each pair [a, b] follows
- a, b are from arr
- a < b
- b - a equals to the minimum absolute difference of any two elements in arr

Example 1:

Input: arr = [4,2,1,3] **Output:** [[1,2],[2,3],[3,4]]

Explanation:

The minimum absolute difference is 1.
List all pairs with difference equal to 1 in ascending order.

```
class Solution {
public:
    vector<vector<int>> minimumAbsDifference(vector<int>& arr) {
        sort(begin(arr), end(arr));
        // sort(arr.begin(),arr.end());
        int n =arr.size();
        int diff = INT_MAX;

        for (int i = 0; i < n - 1; i++) {
            int val = abs(arr[i + 1] - arr[i]);
            diff = min(diff, val);
        }

        vector<vector<int>> result;

        for (int i = 0; i < n - 1; i++) {
            if (arr[i + 1] - arr[i] == diff) {
                result.push_back({arr[i], arr[i + 1]});
            }
        }

        return result;
    }
};
```

LEETCODE - 1491

Average Salary Excluding The Minimum and Maximum Salary

Difficulty : Easy

You are given an array of **unique** integers salary where salary[i] is the salary of the i^{th} employee.

Return *the average salary of employees excluding the minimum and maximum salary*.
Answers within 10^{-5} of the actual answer will be accepted.

Example 1:

Input: salary = [4000,3000,1000,2000]

Output: 2500.00000

Explanation:

Minimum salary and maximum salary are 1000 and 4000 respectively.
Average salary excluding minimum and maximum salary is (2000+3000) / 2 = 2500

```
class Solution {
public:
    double average(vector<int>& salary) {
        sort(salary.begin(),salary.end());

        int n = salary.size();

        double sum =0;

        for(int i =1;i<n-1;i++){
            sum += salary[i];
        }

        double avg = sum/(n-2);
        return avg;
    }
};
```

LEETCODE - 1619

Mean of Array After Removing Some Element

Given an integer array arr, return *the mean of the remaining integers after removing the smallest 5% and the largest 5% of the elements.*

Example 1:

Input: arr = [1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3]
Output: 2.00000
Explanation: After erasing the minimum and the maximum values of this array, all elements are equal to 2, so the mean is 2.

Example 2:

Input: arr = [6,2,7,5,1,2,0,3,10,2,5,0,5,5,0,8,7,6,8,0]
Output: 4.00000

Example 3:

Input: arr = [6,0,7,0,7,5,7,8,3,4,0,7,8,1,6,8,1,1,2,4,8,1,9,5,4,3,8,5,10,8,6,6,1,0,6,10,8,2,3,4]
Output: 4.77778

```
class Solution {
public:
    double trimMean(vector<int>& arr) {
        int n = arr.size();

        sort(arr.begin(),arr.end());

        int remove = n * 0.05;
        double sum =0;

        for(int i = remove;i<n-remove;i++){
            sum +=arr[i];
        }

        double avg = sum/(n-2*remove);

        return avg;
    }
};
```

LEETCODE - 2148

Count Elements with strictly smaller and greater element

Given an integer array nums, return *the number of elements that have* ***both*** *a strictly smaller and a strictly greater element appear in* nums.

Example 1:

Input: nums = [11,7,2,15]

Output: 2

Explanation:
The element 7 has the element 2 strictly smaller than it and the element 11 strictly greater than it.
Element 11 has element 7 strictly smaller than it and element 15 strictly greater than it.
In total there are 2 elements having both a strictly smaller and a strictly greater element appear in nums.

```
class Solution {
public:
    int countElements(vector<int>& nums) {
        int n = nums.size();

        sort(nums.begin(),nums.end());

        int count =0;
        int minVal = nums[0];
        int maxVal = nums[n-1];

        for(int i = 1;i<n-1;i++){
            if(nums[i]>minVal && nums[i]<maxVal){
                count++;
            }
        }

        return count;
    }
};
```

LEETCODE - 2274

Maximum Consecutive Floors Without Special Floors

Alice manages a company and has rented some floors of a building as office space.
Alice has decided some of these floors should be special floors, used for relaxation only.

You are given two integers bottom and top, which denote that Alice has rented all the floors from bottom to top (inclusive).
You are also given the integer array special, where special[i] denotes a special floor that Alice has designated for relaxation.
Return *the maximum number of consecutive floors without a special floor*.

Example 1:

Input: bottom = 2, top = 9, special = [4,6]
Output: 3

Explanation:
The following are the ranges (inclusive) of consecutive floors without a special floor: -
(2, 3) with a total amount of 2 floors. - (5, 5) with a total amount of 1 floor. - (7, 9) with a total amount of 3 floors.
Therefore, we return the maximum number which is 3 floors.

```
class Solution {
public:
    int maxConsecutive(int bottom, int top, vector<int>& special) {
        sort(special.begin(),special.end());

        int ans = 0;

        ans = max(ans,special[0]-bottom);

        for(int i =1;i<special.size();i++){
            ans = max(ans,special[i]-special[i-1]-1);
        }

        ans = max(ans,top-special.back());

        return ans;
    }
};
```

LEETCODE - 1288

Remove Covered Intervals

Given an array intervals where intervals[i] = [l_i, r_i] represent the interval [l_i, r_i), remove all intervals that are covered by another interval in the list.

The interval [a, b) is covered by the interval [c, d) if and only if c <= a and b <= d.
Return *the number of remaining intervals*.

Example 1:

Input:
intervals = [[1,4],[3,6],[2,8]]
Output: 2

Explanation:
Interval [3,6] is covered by [2,8], therefore it is removed.

Example 2:

Input: intervals = [[1,4],[2,3]]
Output: 1

```
class Solution {
public:
    int removeCoveredIntervals(vector<vector<int>>& A) {
        sort(A.begin(), A.end(), [](auto& a, auto& b) {
            if (a[0] == b[0])
                return a[1] > b[1];

            return a[0] < b[0];
        });
        int result = 0;
        int right = 0;
        for (auto& v : A) {
            if (v[1] > right) {
                result++;
                right = v[1];
            }
        }
        return result;
    }
};
```

Learn coding

THANK YOU