Must Do Coding Questions for Companies like Amazon, Microsoft, Adobe

1. Subarray with given sum

- Positive numbers only

In [ ]:
```
vector<int> subarrayWithSum(vector<int> nums, int sum){

    int n =  nums.size(), start = 0, end = 1, currSum = nums[0];

    while(end <= n){

        while(currSum > sum && start < end - 1)
            currSum -= nums[start++];

        if(currSum == sum)
            return {start, end-1};

        if(end < n)
            currSum += nums[end++];

    }

    return {-1,-1};
}
```

- Negative numbers also included

In [ ]:
```
vector<int> subarrayWithSum(vector<int> nums, int sum){

    // sum -> endIndex
    unordered_map<int,int> prefixSum;

    int currSum = 0;

    for(int end = 0; end < nums.size(); end++){

        currSum += nums[end];

        if(currSum == sum)
            return {0, end};

        if(prefixSum.find(currSum-sum) != prefixSum.end())
            return {prefixSum[currSum-sum] + 1,end};

        prefixSum[currSum] = end;

    }

    return {-1,-1}

}
```

2. Count Triplets such that a+b=c

- All Triplets

In [ ]:
```
int countAllTriplets(vector<int> nums){

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

    for(int i=nums.size()-1; i >= 2; i--){

        int left = 0, right = i-1;

        while(left < right){
            int sum = nums[left] + nums[right];

            if(sum == nums[i])
                count++,left++,right--;
            else if(sum < nums[i])
                right--;
            else
                left++;

        }

    }

    return count;
}
```

- Count distinct triplets

In [ ]:
```
int countDistinctTriplets(vector<int> nums){

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

    for(int i=nums.size()-1; i >= 2; i--){

        int left = 0, right = i-1;

        while(left < right){
            int sum = nums[left] + nums[right];

            if(sum == nums[i]){
                count++;
                break;
            }
            else if(sum < nums[i])
                right--;
            else
                left++;

        }

        int a = nums[i];
        while(i >=2 && nums[i] == a) i--;

    }

    return count;
}
```

4. Missing Number

In [ ]:
```
int findMissingNumber(int arr[], int n){

    int missingNum = n;

    for(int i=0; i < n; i++)
        missingNum ^= i ^ nums[i];

    return missingNum;
}
```

5. Merge Two Sorted Arrays

In [ ]:
```
vector<int> mergeSortedArrays(vector<int> a, vector<int> b){

    int n = a.size(), m = b.size();

    vector<int> res(n+m);

    int i = 0, j = 0, k = 0;

    while( k < n + m){

        int elt1 = i < n ? a[i] : INT_MAX;
        int elt2 = j < m ? b[j] : INT_MAX;

        res[k] = min(elt1,elt2);

        (res[k++] == elt1)? i++ : j++;
    }

    return res;
}
```

6. Rearrange array alternatively

Given a sorted array of positive integers. Your task is to rearrange the array elements alternatively i.e first element should be max value, second should be min value, third should be second max, fourth should be second min and so on...

Note: O(1) extra space is allowed. Also, try to modify the input array as required.

In [ ]:
```
vector<int> rearrange(vector<int> nums){

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

    int maxElt = *nums.end() + 1;

    int maxIndex = nums.size() - 1, minIndex = 0;

    for(int i=0; i < nums.size(); i++){

        if(i & 1)
            nums[i] += (nums[minIndex++] % maxElt) * maxElt;
        else
            nums[i] += (nums[maxIndex--] % maxElt) * maxElt;
    }


    for(int i=0; i< nums.size(); i++)
        nums[i] = nums[i]/maxElt;

    return nums;

}
```

7. Number of pairs

Given two arrays X and Y of positive integers, find number of pairs such that xy > yx (raised to power of) where x is an element from X and y is an element from Y.

In [ ]:
```
int countNoPairs(int x, vector<int> y, int noOfY[]){

    if(x == 0)
        return 0;

    if(x == 1)
        return noOfY[0];

    int ans = 0;
    ans += nums.end() - upper_bound(y.begin(),y.end(),x);

    //Because any number is more than 0 and 1
    ans += noOfY[0] + noOfY[1];

    //Exceptions
    if(x == 2)
        ans -= (noOfY[3] + noOfY[4]);

    if(x == 3)
        ans -= noOfY[4];

    return ans;
}

int CountPairs(vector<int> x, vector<int> y){
    int noOfY[5] = {0};

    for(int i=0; i < y.size(); i++)
        if(y[i] < 5)
            noOfY[i]++;

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

    int pairs = 0;

    for(auto num: x)
        pairs += countNoPairs(num,y,noOfY);

    return pairs;

}
```

8. Inversion Of Array

Given an array of positive integers. The task is to find inversion count of array.

In [ ]:
```
//Brute Force
int noOfInversions(vector<int> arr){

    int inv = 0;

    for(int i=0; i < arr.size() - 1; i++)
        for(int j=i+1; j < arr.size(); j++)
            if(arr[i] > arr[j])
                inv++;

    return inv;
}
```

In [ ]:
```cpp
int noOfInversions(vector<int> &arr, vector<int> &temp, int low, int high){
    int ans = 0;

    if(low < high){
        int mid = low + (high-low)/2;
        ans += noOfInversions(arr,temp,low,mid);
        ans += noOfInversions(arr,temp,mid+1,high);
        ans += merge(arr,temp,low,mid,high);
    }

    return ans;
}

int merge(vector<int> &arr, vector<int> &temp, int low, int mid, int high){

    int i = low , j = mid + 1, k = low, inv = 0;

    while(i<=mid && j <= high){

        if(arr[i] < arr[j])
            temp[k++] = arr[i++];
        else{
            temp[k++] = arr[j++];
            inv += mid - i;
        }

    }

    while(i<=mid)
        temp[k++] = arr[i++];

    while(j<=mid)
        temp[k++] = arr[i++];

    for(int i=low;i<=high;i++)
        arr[i+low] = temp[i];

    return inv;

}
```

9. Sort an array of 0s, 1s and 2s

In [ ]:
```cpp
void sort(vector<int> nums){

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

    while(equal <= high){

        switch(nums[equal]){

            case 0: swap(nums[low++],nums[equal++]);
                    break;

            case 1: equal++;
                    break;

            case 2: swap(nums[equal],nums[high--]);
                    break;
        }

    }

}
```

10. Equillibrium Point

Given an array A of N positive numbers. The task is to find the position where equilibrium first occurs in the array. Equilibrium position in an array is a position such that the sum of elements before it is equal to the sum of elements after it.

In [ ]:
```
int equilibriumPoint(vector<int> nums){

    int cumSum = 0;

    for(int i=0; i < nums.size();i++)
        cumSum += nums[i];

    int leftSum = 0;
    for(int i=0; i < nums.size();i++){

        if(leftSum == cumSum - nums[i])
            return i;

        leftSum += nums[i];
        cumSum -= nums[i];

    }

}
```

11. Leaders in an array

Given an array of positive integers. Your task is to find the leaders in the array. Note: An element of array is leader if it is greater than or equal to all the elements to its right side. Also, the rightmost element is always a leader.

In [ ]:
```
vector<int> findLeaders(vector<int> nums){

    vector<int> res;
    int maxElt = INT_MIN;

    for(int i=nums.size()-1;i >=0; i--){
        maxElt = max(maxElt, nums[i]);
        if(nums[i] == maxElt)
            res.push_back(nums[i]);
    }

    reverse(res.begin(),res.end());

    return res;
}
```

12. Minimum Platforms

Given arrival and departure times of all trains that reach a railway station, find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays which represent arrival and departure times of trains that stop.

In [ ]:
```
int minPlatforms(vector<int> arrival, vector<int> dept){

    int n = arrival.size();

    sort(arrival.begin(), arrival.end());
    sort(dept.begin(), dept.end());

    //One train has arrival and 0 has left
    int i = 1, platforms = 1, j = 0, platformsReqd = 1;

    while(i < n && j < n){

        //If another train has arrived before the departure of the train that supposed to leave first
        if(arrival[i] <= dept[j])
            platforms++,i++;
        else
            platfoems--,j++;

        platformsReqd = max(platforms, platformsReqd);

    }

    return platformReqd;
}
```

13. Reverse in groups

Given an array arr[] of positive integers of size N. Reverse every sub-array of K group elements.

In [ ]:
```
void reverseArrayInGroups(vector<int> nums, int K){

    for(int i=0;i<nums.size();i+=K){

        int j = i + K;

        if(j > nums.size())
            j = nums.size();

        reverse(nums.begin()+i,nums.begin()+j);

    }

}
```

14. Trapping Rain Water

In [ ]:
```
int trapWater(vector<int> heights){

    int left_max = 0, right_max = 0;

    int left = 0, right = heights.size() - 1, ans = 0;

    while(left < right){

        if(heights[left] < heights[right]){

            if(heights[left] > left_max)
                left_max = heights[left];
            else
                ans += left_max - heights[left];
        }
        else {
            if(heights[right] > right_max)
                left_max = heights[left];
            else
                ans += right_max - heights[right];
        }
    }

    return ans;
}
```

15. Pythagoran Triplet

In [ ]:
```
bool triplets(vector<int> nums){

    for(int i=0; i<nums.size(); i++)
        nums[i] = nums[i] * nums[i];

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

    for(int i=nums.size()-1; i >=2; i--){
        int low = 0, high = i-1;
        while(low < high){
            if(nums[low]+nums[high] == nums[i]){
                return true;
            }
            else if(nums[low]+nums[high] < nums[i])
                low++;
            else
                high--;
        }
    }

    return false;
}
```

16. Chocolate Distribution

In [ ]:
```
int minDiffernce(vector<int> packets, int n, int students){

    if(students > n) return -1;

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

    int minDiff = INT_MAX;

    for(int i=0; i+student-1 < n; i++){

        //Last student gets max and first get min
        int diff = packets[i+student-1] - packets[i];
        minDiff = min(diff,minDiff);
    }

    return minDiff;
}
```

17. Buy and sell stock

In [ ]:
```
vector<vector<int>> buySellStocks(vector<int> prices){
    int i = 0;
    vector<vector<int>> res;
    while(i < prices.size()){

        //Find min price
        while(i < prices.size()-1 && price[i] > price[i+1]) i++;

        int j = i;

        while(j < prices.size()-1 && prices[j] < prices[j+1]) j++;

        if(j-i > 0)
            res.push_back({i,j});

        i = j;
    }

}
```

18. Find element

Find the element before which all the elements are smaller than it, and after which all are greater

In [ ]:
```
int findElt(vector<int> nums){

    vector<int> leftMax(nums.size());
    leftMax[0] = INT_MIN;

    for(int i=1; i < nums.size(); i++)
        leftMax[i] = max(leftMax[i-1],nums[i]);

    int rightMin = INT_MAX;

    for(int i=nums.size()-1; i>=0; i--)
        if(leftMax[i] <= nums[i] && nums[i] <= rightMin)
                return i;
        else
            rightMin = min(nums[i],rightMin);

    return -1;
}
```

19. Convert array into Zig-Zag fashion

Given an array A (distinct elements) of size N. Rearrange the elements of array in zig-zag fashion. The converted array should be in form a < b > c < d > e < f. The relative order of elements is same in the output i.e you have to iterate on the original array only.

In [ ]:
```
void zigzag(vector<int> nums){

    for(int i=1; i < nums.size(); i += 2){

        if(nums[i-1] > nums[i])
            swap(nums[i-1], nums[i]);

        if(i+1 < nums.size() && nums[i+1] > nums[i])
            swap(nums[i+1], nums[i]);

    }

}
```

20. Largest number formed from an array of numbers

In [ ]:
```
int myCompare(string X, string Y){
    string XY = X.append(Y);

    string YX = Y.append(X);

    return XY.compare(YX) > 0 ? 1: 0;
}

void printLargest(vector<string> arr)
{
    sort(arr.begin(), arr.end(), myCompare);

    for (int i=0; i < arr.size() ; i++ )
        cout << arr[i];
    cout<<endl;
}
```

21. Spirally traversing a matrix

In [ ]:

```
void spiralTraversal(int matrix[][], int rows, int cols){

    int r = 0, c = 0;

    while(r < rows && c < cols){

        //Print first Row
        for(int i=c; i < cols; i++)
            cout<<matrix[r][i];
        r++;

        //Print last col
        for(int i=r;i<rows;i++)
            cout<<matrix[i][cols-1];
        cols--;

        //Print last Row
        if(r < rows){
            for(int i=cols-1; i>=c;i--)
                cout<<matrix[rows-1][i];
            rows--;
        }

        //Print first col
        if(c < cols){
            for(int i=row-1;i>=r;i--)
                cout<<matrix[i][c];
            c++;
        }
    }

}
```