DAY 10

INTERVIEW BIT PROBLEMS :

1. Divide Integers

Divide two integers without using multiplication, division and mod operator.
Return the floor of the result of the division.
Example:
5 / 2 = 2
Also, consider if there can be overflow cases. For overflow case, return INT_MAX.
Note: INT_MAX = 2^31 - 1

CODE :

PYTHON

```
class Solution:
    # @param A : integer
    # @param B : integer
    # @return an integer
    def divide(self, A, B):
        sign=(-1 if ((A<0)^(B<0)) else 1)
        A=abs(A)
        B=abs(B)
        q=0
        temp=0

        for i in range(31,-1,-1):
            if(temp+(B<<i)<=A):
                temp+=B<<i
                q|=1<<i
                if (sign*q)>0 and (sign*q)>(pow(2,31)-1):
                    return sign*(pow(2,31)-1)
                elif (sign*q)<0 and (sign*q)<(-1*(pow(2,31)-1)):
                    return sign*pow(2,31)
        return sign*q
```

2. Intersection Of Sorted Arrays

Find the intersection of two sorted arrays.
OR in other words,
Given 2 sorted arrays, find all the elements which occur in both the arrays.
Example :
Input :
A : [1 2 3 3 4 5 6]
B : [3 3 5]

Output : [3 3 5]

Input :
A : [1 2 3 3 4 5 6]
B : [3 5]

Output : [3 5]

CODE :

PYTHON

```
class Solution:
    # @param A : tuple of integers
    # @param B : tuple of integers
    # @return a list of integers
    def intersect(self, A, B):
        a=len(A)
        b=len(B)
        out=[]
        i=0
        j=0
        while (i<a) and (j<b):
            if A[i]>B[j]:
                j+=1
            elif A[i]<B[j]:
                i+=1
            else:
                out.append(A[i])
                i+=1
                j+=1
        return out
```

3. Merge Two Sorted Lists II

Given two sorted integer arrays A and B, merge B into A as one sorted array.
If the number of elements initialized in A and B are m and n respectively, the resulting size of array A after your code is executed should be m + n
Example :
Input :
A : [1 5 8]
B : [6 9]

Modified A : [1 5 6 8 9]

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
```

```
    # @param B : list of integers
    def merge(self, A, B):
        merged=[]
        i=0
        j=0
        a=len(A)
        b=len(B)
        A.extend(B)
        A.sort()
        return A
```

(OR)

```
class Solution:
    # @param A : list of integers
    # @param B : list of integers
    def merge(self, A, B):
        if not A:
            A = B
            return

        if not B:
            return

        n = len(A)
        m = len(B)
        i=0
        j=0

        while i<n and j<m:
            if A[i]<B[j]:
                i+=1
            else:
                A.insert(i,B[j])
                n+=1
                i+=1
                j+=1

        if i==n and j!=m:
            while j<m:
                A.append(B[j])
                j+=1
```

C++

```
void Solution::merge(vector<int> &A, vector<int> &B) {
    vector<int> C = A;
    A.clear();
    int p1 = 0 , p2 = 0;
    while(true){
        if(p1 == C.size() || p2 == B.size())
            break;
        if(C[p1] < B[p2]){
            A.push_back(C[p1]);
            p1++;
```

```
        }
        else if(C[p1] > B[p2]){
            A.push_back(B[p2]);
            p2++;
        }else if(C[p1] == B[p2]){
            A.push_back(C[p1]);
            A.push_back(B[p2]);
            p1++;
            p2++;
        }
    }
    if(p1 == C.size() && p2 < B.size()){
        while(p2 < B.size()){
            A.push_back(B[p2]);
            p2++;
        }
    }
    if(p2 == B.size() && p1 < C.size()){
        while(p1 < C.size()){
            A.push_back(C[p1]);
            p1++;
        }
    }}
```

4. Minimize the absolute difference

Given three sorted arrays A, B and Cof not necessarily same sizes.
Calculate **the minimum absolute difference between the maximum and minimum number from the triplet a, b, c** such that a, b, c belongs arrays A, B, C respectively.
i.e. **minimize | max(a,b,c) - min(a,b,c) |.**
Example :
Input:
A : [1, 4, 5, 8, 10]
B : [6, 9, 15]
C : [2, 3, 6, 6]
Output:
1
Explanation: We get the minimum difference for a=5, b=6, c=6 as | max(a,b,c) - min(a,b,c) | = |6-5| = 1.

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
    # @param B : list of integers
    # @param C : list of integers
    # @return an integer
    def solve(self, A, B, C):
```

```
        a=len(A)
        b=len(B)
        c=len(C)
        i=0
        j=0
        k=0
        min_abs=pow(2,31)-1
        while i<a and j<b and k<c:
            maxi,mini=max(A[i],B[j],C[k]),min(A[i],B[j],C[k])
            absi=maxi-mini
            min_abs=min(min_abs,absi)
            if mini==A[i]:
                i+=1
            elif mini==B[j]:
                j+=1
            else:
                k+=1
        return min_abs
```

5. 3 Sum

Given an array S of n integers, find three integers in S such that the sum is closest to a given number, target.
Return the sum of the three integers.
Assume that there will only be one solution
Example:
given array S = {-1 2 1 -4},
and target = 1.
The sum that is closest to the target is 2. (-1 + 2 + 1 = 2)

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
    # @param B : integer
    # @return an integer
    def threeSumClosest(self, A, B):
        n=len(A)
        A.sort()
        res=0
        sums=pow(2,31)-1
        for i in range(n-2):
            j=i+1
            k=n-1
            while j<k:
                res=A[i]+A[j]+A[k]
                if (abs(B-res)<abs(B-sums)):
                    sums=res
```

```
            if res>B:
                k-=1
            else:
                j+=1
    return sums
```

6. 3 Sum Zero

Given an array S of n integers, are there elements a, b, c in S such that a + b + c = 0? Find **all unique triplets in the array** which gives the sum of zero.

Note:

Elements in a triplet (a,b,c) must be in non-descending order. (ie, a ≤ b ≤ c)

The solution set must not contain duplicate triplets.

For example, given array S = {-1 0 1 2 -1 -4},

A solution set is:

(-1, 0, 1)

(-1, -1, 2)

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
    # @return a list of list of integers
    def threeSum(self, A):
        #import itertools
        n=len(A)
        out_list=[]
        in_list=[]
        res_list=[]
        A.sort()
        sums=0
        for i in range(n-2):
            j=i+1
            k=n-1
            while j<k:
                sums=A[i]+A[j]+A[k]
                if sums==0:
                    in_list=[A[i],A[j],A[k]]
                    in_list.sort()
                    if in_list not in out_list:
                        out_list.append(in_list)
                    j+=1
                    k-=1
                elif sums<0:
                    j+=1
                else:
                    k-=1
        #out_list.sort()
```

```
        #res_list=list(out_list for out_list,_ in itertools.groupby(out_list))
        #return res_list
        return sorted(out_list)
```

7. Counting Triangles

You are given an array of N non-negative integers, A0, A1 ,…, AN-1.
Considering each array element Ai as the edge length of some line segment, count the number of triangles which you can form using these array values.
Notes:
You can use any value only once while forming each triangle. Order of choosing the edge lengths doesn't matter. Any triangle formed should have a positive area.
Return answer modulo 10^9 + 7.

For example,
A = [1, 1, 1, 2, 2]
Return: 4

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
    # @return an integer
    def nTriang(self, A):
        n=len(A)
        A.sort(reverse=True)
        c = 0
        #print(A)
        for i in range(n- 2):
            side_3=A[i]
            #print('i,side3',i,side_3)
            j = i + 1
            k = len(A) - 1
            #print('k',k)
            while(j < k):
                if(A[j] + A[k] >side_3):
                    #print('a(j),a(k)',A[j],A[k])
                    c+=(k - j)
                    #print('count',c)
                    j += 1
                else:
                    k -= 1
        return c % (10**9 + 7)
```

8. Diffk

Given an array 'A' of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[i] - A[j] = k, i != j.

Example:
Input :
A : [1 3 5]
k : 4
Output : YES
as 5 - 1 = 4
Return 0 / 1 (0 for false, 1 for true) for this problem

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
    # @param B : integer
    # @return an integer
    def diffPossible(self, A, B):
        n=len(A)
        for i in range(n):
            for j in range(i+1,n):
                if abs(A[i]-A[j])==B:
                    return 1
        return 0
```

(OR)

```
    def diffPossible(self, A, B):
        n = len(A)
        if(n<=1):
            return 0
        for i in range(n):
            l,h = i+1,n-1
            k = B+A[i]
            while(l<=h):
                mid =(l+h)//2
                if(A[mid]==k):
                    return 1
                elif(k>A[mid]):
                    l=mid+1
                elif(k<A[mid]):
                    h=mid-1
        return 0
```

9. Remove Element from Array

Given an array and a value, remove all the instances of that value in the array.
Also return the number of elements left in the array after the operation.
It does not matter what is left beyond the expected length.

Example:
If array A is [4, 1, 1, 2, 1, 3]

and value elem is 1,
then new length is 3, and A is now [4, 2, 3]

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
    # @param B : integer
    # @return an integer
    def removeElement(self, A, B):
        A[:] = [ele for ele in A if ele != B]
        return len(A)
```

C++

```
int Solution::removeElement(vector<int> &A, int B) {
    int count = 0;
    int n=A.size();
    for (int i = 0; i < n; i++) {
        if (A[i] == B) continue;
        else {
            A[count] = A[i];
            count++;
        }
    }
    return count;
}
```

10. Remove Duplicates from Sorted Array II

Given a sorted array, remove the duplicates in place such that **each element can appear atmost twice** and return the new length.
Do not allocate extra space for another array, you must do this in place with constant memory.
Note that even though we want you to return the new length, make sure to change the original array as well in place
For example,
Given input array A = [1,1,1,2],
Your function should return length = 3, and A is now [1,1,2].

CODE :

C++

```
int Solution::removeDuplicates(vector<int> &A) {
    if(A.size() == 0){
        return 0;
    }
```

```
    int count = 1;
    int j = 1;
    for(int i = 1 ; i < A.size(); i++){
        if(A[i] != A[i - 1]){
            A[j] = A[i];
            count = 1;
            j++;
        }
        else if(count == 1){
            A[j] = A[i];
            count = 2;
            j++;
        }
    }
    return j;
}
```

PYTHON

```
class Solution:
    # @param A : list of integers
    # @return an integer
    def removeDuplicates(self, A):
        n=len(A)
        c=1
        j=1
        for i in range(1,n):
            if A[i]!=A[i-1]:
                A[j]=A[i]
                j+=1
                c=1
            elif c==1:
                A[j]=A[i]
                j+=1
                c=2
        return j
```

11. Remove Duplicates from Sorted Array

Remove duplicates from Sorted Array
Given a sorted array, remove the duplicates in place such that each element appears only once and return the new length.
Note that even though we want you to return the new length, make sure to change the original array as well in place
Do **not allocate extra space** for another array, you must do this in place with constant memory.

Example:
Given input array A = [1,1,2],
Your function should return length = 2, and A is now [1,2].

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
    # @return an integer
    def removeDuplicates(self, A):
        n=len(A)
        j=1
        for i in range(1,n):
            if A[i]!=A[i-1]:
                A[j]=A[i]
                j+=1
        return j
```

12. Sort by Color

Given an array with n objects colored red, white or blue,
sort them so that objects of the same color are adjacent, with the colors in the order red, white and blue.
Here, we will use the integers 0, 1, and 2 to represent the color red, white, and blue respectively.

Note: Using library sort function is not allowed.

Example :

Input : [0 1 2 0 1 2]

Modify array so that it becomes : [0 0 1 1 2 2]

CODE :

PYTHON

```
class Solution:
    # @param A : list of integers
    # @return A after the sort
    def sortColors(self, A):
        n=len(A)
        l=0
        h=n-1
        m=0
        while m<=h:
            if A[m]==0:
                A[l],A[m]=A[m],A[l]
                l+=1
                m+=1
            elif A[m]==1:
```

```
            m+=1
        else:
            A[m],A[h]=A[h],A[m]
            h-=1
return A
```