DAY 11

INTERVIEW BIT PROBLEMS :

1. Max Continuous Series of 1s

You are given with an array of 1s and 0s. And you are given with an integer M, which signifies number of flips allowed.
Find the **position of zeros which when flipped will produce maximum continuous series of 1s.**
For this problem, return the indices of maximum continuous series of 1s in order.
Example:
Input :
Array = {1 1 0 1 1 0 0 1 1 1 }
M = 1

Output :
[0, 1, 2, 3, 4]

If **there are multiple possible solutions**, return the sequence which has the **minimum start index**.

CODE :

C++

```
vector<int> Solution::maxone(vector<int> &A, int B) {
    int n=A.size();
    int left_ind=0;
    int right_ind=0;
    int final_size=0;
    int final_left=0;
    int z_c=0;
    vector<int>final;

    while(right_ind<n){
        if(z_c<=B){
            if(A[right_ind]==0){
                z_c+=1;
            }
            right_ind+=1;
        }
        if(z_c>B){
            if(A[left_ind]==0){
                z_c-=1;
            }
            left_ind+=1;
        }
        if((right_ind-left_ind)>final_size){
```

```
                final_size=right_ind-left_ind;
                final_left=left_ind;
            }
        }
        for(int i=0;i<final_size;i++){
            final.push_back(final_left+i);
        }
        return final;
}
```

PYTHON

```
class Solution:
    # @param A : list of integers
    # @param B : integer
    # @return a list of integers
    def maxone(self, A, B):
        n=len(A)
        left_ind=0
        right_ind=0
        final_left=0
        final_size=0
        final=[]
        z_c=0
        while right_ind<n:
            if z_c<=B:
                if A[right_ind]==0:
                    z_c+=1
                right_ind+=1
            if z_c>B:
                if A[left_ind]==0:
                    z_c-=1
                left_ind+=1
            if (right_ind-left_ind)>final_size:
                final_size=right_ind-left_ind
                final_left=left_ind
        for i in range(final_size):
            final.append(final_left+i)
        return final
```

2. Array 3 Pointers

You are given 3 arrays A, B and C. All 3 of the arrays are sorted.
Find i, j, k such that :
max(abs(A[i] - B[j]), abs(B[j] - C[k]), abs(C[k] - A[i])) is minimized.
Return the **minimum max(abs(A[i] - B[j]), abs(B[j] - C[k]), abs(C[k] - A[i]))**
Example :
Input :
A : [1, 4, 10]
B : [2, 15, 20]

C : [10, 12]

Output : 5
With 10 from A, 15 from B and 10 from C.

CODE :

PYTHON

```
class Solution:
    # @param A : tuple of integers
    # @param B : tuple of integers
    # @param C : tuple of integers
    # @return an integer
    def minimize(self, A, B, C):
        i=0
        j=0
        k=0
        a=len(A)
        b=len(B)
        c=len(C)
        minimaxi=pow(2,31)-1
        while i<a and j<b and k<c:
            minimaxi=min(minimaxi,max(abs(A[i]-B[j]),abs(B[j]-C[k]),abs(C[k]-A[i])))
            if A[i]==min(A[i],B[j],C[k]):
                i+=1
            elif B[j]==min(A[i],B[j],C[k]):
                j+=1
            else:
                k+=1
        return minimaxi
```

3. Container With Most Water

Given n non-negative integers a1, a2, ..., an,
where each represents a point at coordinate (i, ai).
'n' vertical lines are drawn such that the two endpoints of line i is at (i, ai) and (i, 0).
Find two lines, which together with x-axis forms a container, such that the container contains the most water.
Your program should return an integer which corresponds to the maximum area of water that can be contained (Yes, we know maximum area instead of maximum volume sounds weird. But this is 2D plane we are working with for simplicity).

Note: You may not slant the container.

Example :
Input : [1, 5, 4, 3]
Output : 6

Explanation : 5 and 3 are distance 2 apart. So size of the base = 2. Height of container = min(5, 3) = 3.
So total area = 3 * 2 = 6

CODE :

PYTHON

```
class Solution:
# @param A : list of integers
# @return an integer
def maxArea(self, A):
        l=0
        r=len(A)-1
        maxi_Area=0
        while l<r:
            maxi_Area=max(maxi_Area,min(A[l],A[r])*(r-l))
            if A[l]<A[r]:
                l+=1
            else:
                r-=1
      return maxi_Area
```

4. NUMRANGE

Given an array of non negative integers A, and a range (B, C),
find the number of continuous subsequences in the array which have **sum S in the range [B, C] or B <= S <= C**
Continuous subsequence is defined as all the numbers A[i], A[i + 1], ... A[j]
where 0 <= i <= j < size(A)
Example :
A : [10, 5, 1, 0, 2]
(B, C) : (6, 8)
ans = 3
as [5, 1], [5, 1, 0], [5, 1, 0, 2] are the only 3 continuous subsequence with their sum in the range [6, 8]

NOTE : The answer is guranteed to fit in a 32 bit signed integer.

CODE :

PYTHON

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

```
res=0
i=0
j=0
c=0
while i<n:
    res+=A[j]
    if res in range(B,C+1):
        c+=1
        j+=1
    if res<B:
        j+=1
    elif res>C:
        res=0
        i+=1
        j=i
    if j==n:
        res=0
        i+=1
        j=i
return c
```