DAY 5

INTERVIEW BIT PROBLEMS :

1. Matrix Median

Given a matrix of integers A of size N x M in which each row is sorted.
Find an return the overall median of the matrix A.
Note: No extra memory is allowed.
Note: Rows are numbered from top to bottom and columns are numbered from left to right.

Input Format

The first and only argument given is the integer matrix A.
Output Format
Return the overall median of the matrix A.
Constraints
1 <= N, M <= 10^5
1 <= N*M <= 10^6
1 <= A[i] <= 10^9
N*M is odd
For Example
Input 1:

```
A = [   [1, 3, 5],
        [2, 6, 9],
        [3, 6, 9]   ]
```

Output 1:

```
5
```

Explanation 1:

A = [1, 2, 3, 3, 5, 6, 6, 9, 9]
Median is 5. So, we return 5.

Input 2:

```
A = [   [5, 17, 100]    ]
```

Output 2:

Matrix=17

CODE:

C++

```
int Solution::findMedian(vector<vector<int> > &A) {
    int n=A.size();
    int m=A[0].size();
    int med=n*m/2;
    vector<int>out;
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            out.push_back(A[i][j]);
        }
    }
    std :: nth_element(out.begin(),out.begin()+med,out.end());
    return out[med];
}
```

PYTHON

```
class Solution:
    # @param A : list of list of integers
    # @return an integer
    def findMedian(self, A):
        out=[num for rows in A for num in rows]
        out.sort()
        return out[len(out)//2]
```

2. Sorted Insert Position

Given a sorted array and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You may **assume no duplicates** in the array.
Here are few examples.
[1,3,5,6], 5 → 2
[1,3,5,6], 2 → 1
[1,3,5,6], 7 → 4
[1,3,5,6], 0 → 0

CODE :

C++

```
int Solution::searchInsert(vector<int> &A, int B) {
    // Do not write main() function.
    // Do not read input, instead use the arguments to the function.
    // Do not print the output, instead return values as specified

    int l=0,h=A.size()-1;
    while(l<=h){
        int m=(l+h)/2;
        if(A[m]==B){
            return m;
        }
        else if(A[m]<B){
            l=m+1;
        }
        else{
            h=m-1;
        }
    }
    return l;
}
```

PYTHON

```
class Solution:
    # @param A : list of integers
    # @param B : integer
    # @return an integer
    def searchInsert(self, A, B):
        l=0
        h=len(A)-1
        while(l<=h):
            m=(l+h)//2
            if A[m]==B:
                return m
            elif A[m]<B:
                l=m+1
            else:
                h=m-1
        return l
```

3. Search for a Range

Given a sorted array of integers A(0 based index) of size N, find the starting and ending position of a given integar B in array A.
Your algorithm's runtime complexity must be in the order of O(log n).
Return an array of size 2, such that first element = starting position of B in A and second element = ending position of B in A, if B is not found in A return [-1, -1].

Input Format
The first argument given is the integer array A.
The second argument given is the integer B.
Output Format

Return an array of size 2, such that first element = starting position of B in A and second element = ending position of B in A, if B is not found in A return [-1, -1].

Constraints

1 <= N <= 10^6

1 <= A[i], B <= 10^9

For Example

Input 1:

A = [5, 7, 7, 8, 8, 10]

B = 8

Output 1:

[3, 4]

Explanation 1:

First occurence of 8 in A is at index 3

Second occurence of 8 in A is at index 4

ans = [3, 4]

Input 2:

A = [5, 17, 100, 111]

B = 3

Output 2:

[-1, -1]

CODE :

PYTHON

```
class Solution:
    # @param A : tuple of integers
    # @param B : integer
    # @return a list of integers
    def searchRange(self, A, B):
        output=[-1,-1]
        A=list(A)
        h=len(A)-1
```

```
        l=0
        while l<=high:
            mid=l+(h-l)//2
            if A[mid]>=B:
                h=mid-1
            if A[mid]<B:
                l=mid+1
        if A[l]==B:
            output[0]=l
        l=0
        h=len(A)-1
        while l<=h:
            mid=l+(h-l)//2
            if A[mid]>B:
                h=mid-1
            if A[mid]<=B:
                l=mid+1
        if A[h]==B:
            output[1]=h
        return output
```

C++

```
vector<int> Solution::searchRange(const vector<int> &A, int B) {
    int n=A.size();
    int l=0,h=n-1;
    vector<int>out(2,-1);
    while(l<=h){
        int m=l+(h-l)/2;
        if(A[m]>=B){
            h=m-1;
        }
        if(A[m]<B){
            l=m+1;
        }
```

```
    }
    if(A[l]==B){
        out[0]=l;
    }
    l=0;
    h=n-1;
    while(l<=h){
        int m=l+(h-l)/2;
        if(A[m]>B){
            h=m-1;
        }
        if(A[m]<=B){
            l=m+1;
        }
    }
    if(A[h]==B){
        out[1]=h;
    }
    return out;
}
```

4. Implement Power Function

Implement pow(x, n) % d.
In other words, given x, n and d,
find (xn % d)
Note that remainders on division cannot be negative.
In other words, make sure the answer you return is non negative.

Input : x = 2, n = 3, d = 3
Output : 2

2^3 % 3 = 8 % 3 = 2.

CODE :

PYTHON

```
class Solution:
    # @param x : integer
    # @param n : integer
    # @param d : integer
    # @return an integer
    def pow(self, x, n, d):
        power= 1%d
        while n > 0:
            if n & 1:
                power= (power * x) % d
            x = x**2 % d
            n >>= 1
        return power
```