SREACHING-1

- https://leetcode.com/problems/search-insert-position/

```
class Solution {
public int searchInsert(int[] nums, int target) {
int n=nums.length;
int ans=0;
for(int i=0;i<n;i++)
{
if(nums[i]==target)
{
ans=i;
}
if(nums[i]!=target && nums[i]<target)
{
ans=i+1;
}
}
return ans;
}
}
```

- https://leetcode.com/problems/find-peak-element/

```
class Solution {
public int findPeakElement(int[] nums) {

int n=nums.length;
if(n==1|| nums[0]>nums[1]) return 0;

if(nums[n - 1] > nums[n - 2]) return n - 1;

int l = 0;
int r = n - 1;

while(l <= r)
{
int m = (l + r)/2;
int minus = m == 0 ? Integer.MIN_VALUE : nums[m-1];
int plusUltra = m == n - 1 ? Integer.MIN_VALUE : nums[m+1];
if(minus < nums[m] && plusUltra < nums[m]) return m;

if(minus < nums[m]) l = m + 1;
```

```
else r = m - 1;
}

return l;

}
}
```

- https://leetcode.com/problems/search-in-rotated-sorted-array/

```
class Solution {
public int search(int[] nums, int target) {
int res = 0;
int a = 0;

for(int i=0; i<nums.length; i++){
if(nums[i] == target){
res=i;
a++;
}
}
if(a>0){
return res;
}
return -1;
}
}
```

- https://leetcode.com/problems/search-a-2d-matrix/

```
class Solution {
public boolean searchMatrix(int[][] matrix, int target) {
int m=matrix.length;
int n=matrix[0].length;

// binnary implementation
int low=0,high=m*n-1;
while(low<=high) {
int midIdx,midEle,rowIdx,colIdx;
midIdx=low+(high-low)/2;
rowIdx=midIdx/n;
colIdx=midIdx%n;
midEle=matrix[rowIdx][colIdx];
if(midEle==target)
return true;
else if(midEle<target)
low=midIdx+1;
else
```

```
high=midIdx-1;
}
return false;
}
}
```

- https://leetcode.com/problems/median-of-two-sorted-arrays/

```
class Solution {
public double findMedianSortedArrays(int[] nums1, int[] nums2) {
int n1 = nums1.length;
int n2 = nums2.length;
int n = n1 + n2;
int[] new_arr = new int[n];

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

while (i<=n1 && j<=n2) {
if (i == n1) {
while(j<n2) new_arr[k++] = nums2[j++];
break;
} else if (j == n2) {
while (i<n1) new_arr[k++] = nums1[i++];
break;
}

if (nums1[i] < nums2[j]) {
new_arr[k++] = nums1[i++];
} else {
new_arr[k++] = nums2[j++];
}
}
```

- https://www.interviewbit.com/problems/matrix-median/

```
public class Solution {
    public int findMedian(ArrayList<ArrayList<Integer>> A) {

        int n = A.size();
        int m = A.get(0).size();

        int start = 0;
        int end = 1000000002;
        int ans = -1;
```

```
    while(start <= end) {

        int mid = start + (end - start) / 2;

        int count = getLessThan(mid, A);
        //System.out.println(mid + " " + count);

        if(count <= (n * m) / 2) {
            ans = mid;
            start = mid + 1;
        } else {
            end = mid - 1;
        }
    }

    return ans;
}

public int getLessThan(int mid, ArrayList<ArrayList<Integer>> A) {

    int n = A.size();
    int m = A.get(0).size();
    int count = 0;

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

        if(A.get(i).get(m - 1) < mid) {
            count += m;
        } else if(A.get(i).get(0) < mid) {
            int start = 0;
            int end = m - 1;
            int ans = 0;

            while(start <= end) {
                int mi = start + (end - start) / 2;
                if(A.get(i).get(mi) < mid) {
```

```
                    ans = mi;
                    start = mi + 1;
                } else {
                    end = mi - 1;
                }
            }

            count += ans + 1;
        }
    }

    return count;
  }
}
```