MEDIAN OF TWO SORTED ARRAYS

✡ In this problem, we are given 2 arrays of integers which are sorted. Our job is to find the median of the addition of these 2 arrays.

Bruteforce solution is to obviously merge both the sorted arrays using 2 pointer approach and find middle element.

Better solution is to not use an extra merger array OR while merging, just stopping at N/2 as that is median.

Optimal solution is to use Binary Search for this. In case N is even, we split the merged array into two halves. If we are able to figure out what the first half looks like without merging, by considering all possible configurations, we can find the median.

We start by trying to create first half by checking how many elements should be taken from each array while maintaining

sorted condition.
Lets say n1 + n2 = 10, i.e. half is of length 5. We can either take 0 from arr1 and 5 from arr2, 1 from arr1 and 4 from arr2 etc. To figure out which config we should choose to maintain sorted array. For this we run a binary search on array with smaller size.
When our total number of elements is odd, a lot doesn't change we just fix length of our left half

Pseudocode :

```
medianSortedArrays(arr1, n1, arr2, n2){
    if(n1 > n2) {
        return medianSortedArrays(arr2, n2,
                                  arr1, n1)
    }
    low, high = 0, n1
    while(low <= high) {
        mid = (low + high)/2
        oMid = (n1 + n2 + 1)/2 - mid
        L1, L2 = INT_MIN, INT_MIN
        r1, r2 = INT_MAX, INT_MAX
        if(mid < n1) r1 = arr1[mid]
        if(oMid < n2) r2 = arr2[oMid]
        if(mid-1 > 0) l1 = arr1[mid-1]
        if(oMid-1 > 0) l2 = arr2[oMid-1]
        if(l1 <= r2 && l2 <= r1) {
            if(n1 + n2 % 2 != 0) {
```

```
            return max(l1, l2)
        } else {
            return (max(l1, l2) + min(r1, r2)) / 2
        }
    } else if (l1 > r2) {
        high = mid - 1
    } else {
        low = mid + 1
    }
}
return 0
```