**4. Median of Two Sorted Arrays: -**

Hard Accepted: 2.1M Submissions: 5.6M Acceptance Rate: 37.8%

Given two sorted arrays nums1 and nums2 of size m and n respectively, return **the median** of the two sorted arrays.

The overall run time complexity should be O(log (m+n)).

**Example 1:**

**Input:** nums1 = [1,3], nums2 = [2]

**Output:** 2.00000

**Explanation:** merged array = [1,2,3] and median is 2.

**Example 2:**

**Input:** nums1 = [1,2], nums2 = [3,4]

**Output:** 2.50000

**Explanation:** merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.

**Constraints:**

* nums1.length == m
* nums2.length == n
* 0 <= m <= 1000
* 0 <= n <= 1000
* 1 <= m + n <= 2000
* -106 <= nums1[i], nums2[i] <= 106

**Code: -**

class Solution {

public:

    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {

      // The whole Logic will work considering 1st Array size < 2nd Array size

      int n1 = nums1.size(), n2 = nums2.size();

      if(n1 > n2)

        return findMedianSortedArrays(nums2, nums1);

      int l1, l2, r1, r2;

      int low = 0, high = n1;

      while(low <= high){

        int mid1 = (low + high) / 2;

        int mid2 = ((n1 + n2 + 1) / 2) - mid1;

        l1=INT\_MIN, l2=INT\_MIN, r1=INT\_MAX, r2=INT\_MAX;

        if(mid1 - 1 >= 0)    l1 = nums1[mid1 - 1];

        if(mid2 - 1 >= 0)    l2 = nums2[mid2 - 1];

        if(mid1 < n1)        r1 = nums1[mid1];

        if(mid2 < n2)        r2 = nums2[mid2];

        // case1 = answer found

        if(l1 <= r2 and l2 <= r1){

          // total count is EVEN

          if((n1 + n2) % 2 == 0){

            cout << "(" << max(l1, l2) << "," << min(r1, r2) << ")";

            return ((double)(max(l1, l2) + min(r1, r2))) / 2.0;

          }

          // total count is ODD

          else{

            cout << max(l1, l2);

            return (double)max(l1, l2);

          }

        }

        // case2 = answer will be from the left half

        if(l1 > r2)

          high = mid1 - 1;

        // case3 = answer will be from the right half

        else

          low = mid1 + 1;

      }

      return -1;

    }

};

**T.C: - O(log min(n1, n2))**

**S.C: - O(1)**