**LEETCODE PROBLEM 4:**

**Median of Two Sorted Arrays**

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

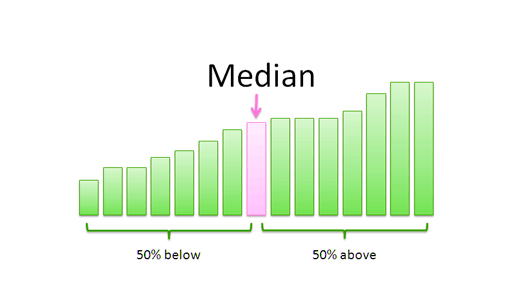
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.

**Solution: First we will try to understand what is median problem in Statistics.**

Median, in statistics, is the middle value of the given list of data, when arranged in an order. The arrangement of data or observations can be done either in ascending order or descending order.

Example: The median of 2,3,4 is 3.

In Maths, the median is also a type of average, which is used to find the center value.



The formula to calculate the median of the data set is given as follows:

**Odd Number of Observations**

If the total number of observation given is odd, then the formula to calculate the median is:

**Median = {(n+1)/2}thterm**

where n is the number of observations

**Even Number of Observations**

If the total number of observation is even, then the median formula is:

**Median  = [(n/2)th term + {(n/2)+1}th]/2**

where n is the number of observations

NOW THE ALGORITHM FOR THIS PROBLEM:

class Solution {

public double findMedianSortedArrays(int[] nums1, int[] nums2) {

int size = nums1.length + nums2.length;

if(size%2 == 1) {

return (double)getMedian(nums1, nums2, size/2);

} else {

int x = getMedian(nums1, nums2, size/2);

int y = getMedian(nums1, nums2, size/2-1);

return ((double)(x+y)/2);

}

}

int getMedian(int[] nums1, int[] nums2, int k) {

int i=0, j=0;

int median=0;

while(i<nums1.length && j<nums2.length) {

if(nums1[i] < nums2[j]) {

median = nums1[i];

i+=1;

} else {

median = nums2[j];

j+=1;

}

if(k==0)

return median;

k-=1;

}

while(i<nums1.length) {

median = nums1[i];

if(k==0)

return median;

k-=1;

i+=1;

}

while(j<nums2.length) {

median = nums2[j];

if(k==0)

return median;

k-=1;

j+=1;

}

return 0;

}

}