**Median of Two Sorted Arrays**

**Hard**

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.

**Example 3:**

**Input:** nums1 = [0,0], nums2 = [0,0]

**Output:** 0.00000

**Example 4:**

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

**Output:** 1.00000

**Example 5:**

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

**Output:** 2.00000

**Constraints:**

* nums1.length == m
* nums2.length == n
* 0 <= m <= 1000
* 0 <= n <= 1000
* 1 <= m + n <= 2000
* -106 <= nums1[i], nums2[i] <= 106
* class Solution {
* public:
* int kth(vector<int> :: iterator arr1, vector<int> :: iterator arr2,
* vector<int> :: iterator end1, vector<int> :: iterator end2, int k) {
* if (arr1 == end1)
* return arr2[k];
* if (arr2 == end2)
* return arr1[k];
* int mid1 = (end1 - arr1) / 2;
* int mid2 = (end2 - arr2) / 2;
* if (mid1 + mid2 < k) {
* if (arr1[mid1] > arr2[mid2])
* return kth(arr1, arr2 + mid2 + 1, end1, end2,
* k - mid2 - 1);
* else
* return kth(arr1 + mid1 + 1, arr2, end1, end2,
* k - mid1 - 1);
* }
* else {
* if (arr1[mid1] > arr2[mid2])
* return kth(arr1, arr2, arr1 + mid1, end2, k);
* else
* return kth(arr1, arr2, end1, arr2 + mid2, k);
* }
* }
* double findMedianSortedArrays(vector<int>& a1, vector<int>& a2) {
* int size=a1.size()+a2.size();
* /\*vector<double> v;
* int i=0, j=0;
* while (i<a1.size() and j<a2.size()) {
* if (a1[i]<a2[j]) {
* v.push\_back((double)a1[i]);
* i++;
* }
* else {
* v.push\_back((double)a2[j]);
* j++;
* }
* }
* while (i<a1.size()) {
* v.push\_back((double)a1[i]);
* i++;
* }
* while (j<a2.size()) {
* v.push\_back((double)a2[j]);
* j++;
* }
* int x, y;
* if (size%2==0) {
* x=(size-1)/2;
* y=(size+1)/2;
* return (v[x]+v[y])/2;
* }
* else return v[(size-1)/2];\*/
* if (size%2==0) {
* int x=(size-1)/2, y=size/2;
* int m1=kth(a1.begin(), a2.begin(), a1.end(), a2.end(), x);
* int m2=kth(a1.begin(), a2.begin(), a1.end(), a2.end(), y);
* double ans=((double)m1+(double)m2)/2;
* return ans;
* }
* else {
* return (double)kth(a1.begin(), a2.begin(), a1.end(), a2.end(), size/2);
* }
* }
* };