class Solution {

private:

    int findkth(int\* num1, int n1, int\* num2, int n2, int k)

    {

        if(n1>n2)

            return findkth(num2, n2, num1, n1, k); // ensure the first array is shorter than the second one

        if(n1==0)

            return num2[k-1];

        if(k==1)

            return min(num1[0], num2[0]);

        int k1 = min(k/2, n1);

        int k2 = k - k1;

        if(num1[k1-1]==num2[k2-1])

            return num1[k1-1];

        else if(num1[k1-1]<num2[k2-1])

            return findkth(num1+k1, n1-k1, num2, n2, k-k1);

        else

            return findkth(num1, n1, num2+k2, n2-k2, k-k2);

    }

public:

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

        int n1 = nums1.size();

        int\* num1=NULL;

        if(n1>0)

            num1 = (int\*) malloc(n1\*sizeof(int));

        for(int i=0; i<n1; i++)

            num1[i] = nums1[i];

        int n2 = nums2.size();

        int\* num2=NULL;

        if(n2>0)

            num2 = (int\*) malloc(n2\*sizeof(int));

        for(int j=0; j<n2; j++)

            num2[j] = nums2[j];

        double result;

        int n = n1+n2;

        if(n%2==1)

            result = 1.0\*findkth(num1, n1, num2, n2, (n+1)/2);

        else

        {

            result = 0.5\*findkth(num1, n1, num2, n2, n/2);

            result += 0.5\*findkth(num1, n1, num2, n2, n/2+1);

        }

        if(n1>0)

            free(num1);

        if(n2>0)

            free(num2);

        return result;

    }

};