MedianOfTwoSortedArray.java

```
   package com.example.array;

   import java.util.Arrays;

   public class MedianOfTwoSortedArray {
       // Brute-Force
       public static double medianBruteForce(int[] nums1, int[] nums2) {
           // Get the sizes of both input arrays.
           int n = nums1.length;
           int m = nums2.length;

           // Merge the arrays into a single sorted array.
1          int[] merged = new int[n + m];
           int k = 0;
2          for (int i = 0; i < n; i++) {
1              merged[k++] = nums1[i];
           }
2          for (int i = 0; i < m; i++) {
1              merged[k++] = nums2[i];
           }

           // Sort the merged array.
1          Arrays.sort(merged);

           // Calculate the total number of elements in the merged array.
           int total = merged.length;

2          if (total % 2 == 1) {
               // If the total number of elements is odd, return the middle element as the
               // median.
2              return (double) merged[total / 2];
           } else {
               // If the total number of elements is even, calculate the average of the two
               // middle elements as the median.
2              int middle1 = merged[total / 2 - 1];
1              int middle2 = merged[total / 2];
3              return ((double) middle1 + (double) middle2) / 2.0;
           }
       }

       // Better-Approach
       public static double medianBetter(int[] nums1, int[] nums2) {
           int n = nums1.length;
           int m = nums2.length;
           int i = 0, j = 0, m1 = 0, m2 = 0;

           // Find median.
4          for (int count = 0; count <= (n + m) / 2; count++) {
               m2 = m1;
2              if (i != n && j != m) {
2                  if (nums1[i] > nums2[j]) {
1                      m1 = nums2[j++];
                   } else {
1                      m1 = nums1[i++];
                   }
2              } else if (i < n) {
1                  m1 = nums1[i++];
               } else {
1                  m1 = nums2[j++];
               }
           }

           // Check if the sum of n and m is odd.
3          if ((n + m) % 2 == 1) {
1              return (double) m1;
           } else {
1              double ans = (double) m1 + (double) m2;
2              return ans / 2.0;
           }
       }

       // Optimal-Approach
       public static double medianOptimal(int[] nums1, int[] nums2) {
           int n1 = nums1.length, n2 = nums2.length;
75
76          // Ensure nums1 is the smaller array for simplicity
77  2       if (n1 > n2)
78  1           return medianOptimal(nums2, nums1);
79
80  1       int n = n1 + n2;
81  3       int left = (n1 + n2 + 1) / 2; // Calculate the left partition size
82          int low = 0, high = n1;
83
84  2       while (low <= high) {
85  2           int mid1 = (low + high) >> 1; // Calculate mid index for nums1
86  1           int mid2 = left - mid1; // Calculate mid index for nums2
87
88              int l1 = Integer.MIN_VALUE, l2 = Integer.MIN_VALUE, r1 = Integer.MAX_VALUE, r2 = Integer.MAX_
89
90              // Determine values of l1, l2, r1, and r2
91  2           if (mid1 < n1)
92                  r1 = nums1[mid1];
93  2           if (mid2 < n2)
94                  r2 = nums2[mid2];
95  3           if (mid1 - 1 >= 0)
96  1               l1 = nums1[mid1 - 1];
97  3           if (mid2 - 1 >= 0)
98  1               l2 = nums2[mid2 - 1];
99
100 4           if (l1 <= r2 && l2 <= r1) {
101                 // The partition is correct, we found the median
102 2               if (n % 2 == 1)
103 1                   return Math.max(l1, l2);
104                 else
105 3                   return ((double) (Math.max(l1, l2) + Math.min(r1, r2))) / 2.0;
106 2           } else if (l1 > r2) {
107                 // Move towards the left side of nums1
108 1               high = mid1 - 1;
109             } else {
110                 // Move towards the right side of nums1
111 1               low = mid1 + 1;
112             }
113         }
114
115         return 0; // If the code reaches here, the input arrays were not sorted.
116     }
117 }
```

Mutations

13	1. Replaced integer addition with subtraction → KILLED
15	1. changed conditional boundary → KILLED
	2. negated conditional → KILLED
16	1. Changed increment from 1 to -1 → KILLED
18	1. changed conditional boundary → KILLED
	2. negated conditional → KILLED
19	1. Changed increment from 1 to -1 → KILLED
23	1. removed call to java/util/Arrays::sort → KILLED
28	1. negated conditional → KILLED
	2. Replaced integer modulus with multiplication → KILLED
31	1. replaced double return with 0.0d for com/example/array/MedianOfTwoSortedArray::medianBruteForce → KIL
	2. Replaced integer division with multiplication → KILLED
35	1. Replaced integer subtraction with addition → KILLED
	2. Replaced integer division with multiplication → KILLED
36	1. Replaced integer division with multiplication → KILLED
37	1. Replaced double division with multiplication → KILLED
	2. replaced double return with 0.0d for com/example/array/MedianOfTwoSortedArray::medianBruteForce → KIL
	3. Replaced double addition with subtraction → KILLED
48	1. Replaced integer addition with subtraction → KILLED
	2. changed conditional boundary → KILLED
	3. negated conditional → KILLED
	4. Replaced integer division with multiplication → KILLED
50	1. negated conditional → KILLED
	2. negated conditional → KILLED
51	1. negated conditional → KILLED
	2. changed conditional boundary → SURVIVED
52	1. Changed increment from 1 to -1 → KILLED
54	1. Changed increment from 1 to -1 → KILLED
56	1. negated conditional → NO_COVERAGE
	2. changed conditional boundary → NO_COVERAGE
57	1. Changed increment from 1 to -1 → NO_COVERAGE
59	1. Changed increment from 1 to -1 → NO_COVERAGE
64	1. Replaced integer addition with subtraction → KILLED
	2. Replaced integer modulus with multiplication → KILLED
	3. negated conditional → KILLED

65	1. replaced double return with 0.0d for com/example/array/MedianOfTwoSortedArray::medianBetter → KILLED
67	1. Replaced double addition with subtraction → KILLED
68	1. replaced double return with 0.0d for com/example/array/MedianOfTwoSortedArray::medianBetter → KILLED
	2. Replaced double division with multiplication → KILLED
77	1. changed conditional boundary → KILLED
	2. negated conditional → KILLED
78	1. replaced double return with 0.0d for com/example/array/MedianOfTwoSortedArray::medianOptimal → NO_COVE
80	1. Replaced integer addition with subtraction → KILLED
81	1. Replaced integer addition with subtraction → KILLED
	2. Replaced integer division with multiplication → KILLED
	3. Replaced integer addition with subtraction → KILLED
84	1. negated conditional → KILLED
	2. changed conditional boundary → SURVIVED
85	1. Replaced integer addition with subtraction → KILLED
	2. Replaced Shift Right with Shift Left → KILLED
86	1. Replaced integer subtraction with addition → KILLED
91	1. negated conditional → KILLED
	2. changed conditional boundary → SURVIVED
93	1. negated conditional → SURVIVED
	2. changed conditional boundary → SURVIVED
95	1. negated conditional → SURVIVED
	2. Replaced integer subtraction with addition → SURVIVED
	3. changed conditional boundary → SURVIVED
96	1. Replaced integer subtraction with addition → KILLED
97	1. changed conditional boundary → SURVIVED
	2. Replaced integer subtraction with addition → SURVIVED
	3. negated conditional → KILLED
98	1. Replaced integer subtraction with addition → KILLED
100	1. negated conditional → KILLED
	2. changed conditional boundary → SURVIVED
	3. negated conditional → KILLED
	4. changed conditional boundary → SURVIVED
102	1. negated conditional → KILLED
	2. Replaced integer modulus with multiplication → KILLED
103	1. replaced double return with 0.0d for com/example/array/MedianOfTwoSortedArray::medianOptimal → KILLED
105	1. Replaced integer addition with subtraction → KILLED
	2. Replaced double division with multiplication → KILLED
	3. replaced double return with 0.0d for com/example/array/MedianOfTwoSortedArray::medianOptimal → KILLED
106	1. negated conditional → KILLED
	2. changed conditional boundary → SURVIVED
108	1. Replaced integer subtraction with addition → NO_COVERAGE
111	1. Replaced integer addition with subtraction → TIMED_OUT

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NEGATE_CONDITIONALS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- com.example.array.MedianOfTwoSortedArrayTest.testMedianBruteForce_OddLengthArrays(com.example.array.MedianOfTwoSortedArrayTest) (0 ms)
- com.example.array.MedianOfTwoSortedArrayTest.testMedianBruteForce_EvenLengthArrays(com.example.array.MedianOfTwoSortedArrayTest) (0 ms)
- com.example.array.MedianOfTwoSortedArrayTest.testMedianOptimal_EvenLengthArrays(com.example.array.MedianOfTwoSortedArrayTest) (0 ms)
- com.example.array.MedianOfTwoSortedArrayTest.testMedianBetter_OddLengthArrays(com.example.array.MedianOfTwoSortedArrayTest) (0 ms)
- com.example.array.MedianOfTwoSortedArrayTest.testMedianBetter_EvenLengthArrays(com.example.array.MedianOfTwoSortedArrayTest) (0 ms)
- com.example.array.MedianOfTwoSortedArrayTest.testMedianOptimal_OddLengthArrays(com.example.array.MedianOfTwoSortedArrayTest) (0 ms)

Report generated by PIT 1.15.0