

PROBLEM-2

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 $(2 + 3) / 2 = 2.5$.

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`
- $0 \leq m \leq 1000$
- $0 \leq n \leq 1000$
- $1 \leq m + n \leq 2000$
- $-10^6 \leq \text{nums1}[i], \text{nums2}[i] \leq 10^6$

PROBLEM-2

SOLUTION

```
class Solution {  
  
    public double findMedianSortedArrays(int[] nums1, int[] nums2) {  
  
        if (nums1.length > nums2.length) {  
  
            return findMedianSortedArrays(nums2, nums1);  
  
        }  
  
  
        int m = nums1.length;  
  
        int n = nums2.length;  
  
  
        int start = 0;  
  
        int end = m;  
  
  
        while (start <= end) {  
  
  
            int partitionNums1 = (start + end) / 2;  
  
            int partitionNums2 = (m + n + 1) / 2 - partitionNums1;  
  
  
            int maxLeftNums1 = partitionNums1 == 0 ? Integer.MIN_VALUE : nums1[partitionNums1 - 1];  
  
  
            int minRightNums1 = partitionNums1 == m ? Integer.MAX_VALUE : nums1[partitionNums1];  
  
  
            int maxLeftNums2 = partitionNums2 == 0 ? Integer.MIN_VALUE : nums2[partitionNums2 - 1];  
  
            int minRightNums2 = partitionNums2 == n ? Integer.MAX_VALUE : nums2[partitionNums2];  
  
  
            if (maxLeftNums1 <= minRightNums2 && maxLeftNums2 <= minRightNums1) {
```

PROBLEM-2

```
        if ((m + n) % 2 == 0) {

            return (Math.max(maxLeftNums1, maxLeftNums2) + Math.min(minRightNums1,
minRightNums2)) / 2.0;

        } else {

            return Math.max(maxLeftNums1, maxLeftNums2);

        }

    }

    else if (maxLeftNums1 > minRightNums2) {

        end = partitionNums1 - 1;

    }

    else {

        start = partitionNums1 + 1;

    }

}

throw new IllegalArgumentException();

}
```

PROBLEM-2

OUTPUT

The screenshot displays a code editor with lines 45 and 46. Line 45 contains a closing curly brace '}', and line 46 is empty. Below the editor, a tabbed interface shows 'Testcase', 'Run Code Result', and 'Debugger' (locked). The 'Run Code Result' tab is active, showing a green 'Accepted' status and a runtime of 0 ms. The input is '[1,3]' and '[2]', the output is '2.00000', and the expected result is '2.00000'. A 'Diff' button is next to the output. At the bottom, there are navigation buttons ('prev', '2/330', 'Next >'), a 'Console' tab, a link to 'Use Example Testcases', and 'Run Code' and 'Submit' buttons. The Windows taskbar at the bottom shows icons for the Start menu, File Explorer, Word, and a blue icon, along with system tray icons for network, battery, and language (ENG), and a clock showing 23:07 on 20-06-2021.

```
45 }
46
```

Testcase Run Code Result Debugger

Accepted Runtime: 0 ms

Your input [1,3]
[2]

Output 2.00000

Expected 2.00000

prev 2/330 Next > Console [Use Example Testcases](#)

Windows taskbar: Start, File Explorer, Word, Blue icon, Network, Battery, ENG, 23:07, 20-06-2021, 3 notifications