JavaScript Median of Two Sorted Arrays

Challenge

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))`.

1st Example

```
Input: nums1 = [1,3], nums2 = [2]
Output: 2.00000
Explanation: merged array = [1,2,3]
             and median is 2.
```

2nd Example

```
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.
```

Constraints

- `nums1.length == m`

- `nums2.length == n`
- `0 <= m <= 1000`
- `0 <= n <= 1000`
- `1 <= m + n <= 2000`
- `-10⁶ <= nums1[i], nums2[i] <= 10⁶`

Solution

```
const findMedianSortedArrays = (nums1, nums2) => {
    const merge = [...nums1, ...nums2]
                  .sort((a, b) => a - b),
              n = merge.length;

    if (n < 2) {
        return merge.join('');
    }

    if (n % 2 === 0) {
        const mid = n / 2;

        return (merge[mid - 1] + merge[mid]) / 2;
    } else {
        const mid = Math.floor(n / 2);

        return merge[mid];
    }
};
```

Explanation

I've written a function called `findMedianSortedArrays` that takes in two arrays, `nums1` and `nums2`. The purpose of this function is to

merge and sort the arrays, and then find the median of the merged array.

Inside the function, the arrays `nums1` and `nums2` are merged into a new array called `merge` using the spread operator. The `merge` array is then sorted in ascending order using the `sort` method with a comparison function that subtracts one element from another.

The length of the `merge` array is stored in a variable called `n`. If the length is less than `2`, indicating that there is only one element or no elements in the array, the function joins the elements of the `merge` array into a string and returns it.

If the length of the `merge` array is even, the function calculates the middle index by dividing the length by `2`. The median is then calculated by adding the element at the index `mid - 1` and the element at the index `mid`, and dividing the sum by `2`. This value is returned as the median.

If the length of the `merge` array is odd, the function calculates the middle index using the `Math.floor` function to round down the division of the length by `2`. The element at the index `mid` of the `merge` array is returned as the median.

In summary, the `findMedianSortedArrays` function takes two arrays, merges and sorts them, and then finds the median of the merged array.

Author: Trevor Morin