median of two sorted arrays leetcode solution

Median of Two Sorted Arrays is a classic problem that frequently appears in coding interviews and competitive programming. The challenge lies in finding the median of two sorted arrays without merging them, with a focus on achieving an efficient solution. This article will delve into the problem's requirements, provide an efficient algorithm to find the median, and illustrate the solution through examples.

Understanding the Problem

The problem statement for the median of two sorted arrays is as follows:

Given two sorted arrays, `nums1` and `nums2`, of sizes `m` and `n` respectively, the goal is to find the median of the two sorted arrays combined. The median is defined as:

- If the combined length of the arrays is odd, the median is the middle element.
- If the combined length is even, the median is the average of the two middle elements.

For example:
- For `nums1 = [1, 3]` and `nums2 = [2]`, the median is `2.0`.
- For `nums1 = [1, 2]` and `nums2 = [3, 4]`, the median is `(2 + 3) / 2 = 2.5`.

Input and Output

- Input: Two sorted arrays `nums1` and `nums2`.
- Output: A single float value representing the median.

Constraints

To efficiently solve the problem, we should keep the following constraints in mind:
- The total number of elements in both arrays can be very large (up to 2×10^6).
- Both `nums1` and `nums2` can have different sizes.
- The elements of the arrays are integers.

Brute Force Approach

A straightforward solution would be to combine the two arrays and then sort the combined array. The median can then be easily calculated. However, this approach has a time complexity of $O((m+n) \log(m+n))$ due to the sorting step, which can be inefficient for large arrays.

Optimal Approach

The optimal solution to the median of two sorted arrays problem can be achieved through a binary search algorithm, which operates in $O(\log(\min(m, n)))$ time. This method takes advantage of the properties of sorted arrays to find the median without merging.

Algorithm Explanation

1. Identify the Shorter Array:
Always perform the binary search on the smaller of the two arrays for efficiency.

2. Binary Search Setup:
Define pointers for the binary search. Let `i` be the partition index for `nums1` and `j` be the partition index for `nums2`. The partitions divide the arrays into left and right halves.

3. Conditions for Partitioning:
- The left side of `nums1` (i.e., `nums1[i-1]`) must be less than or equal to the right side of `nums2` (i.e., `nums2[j]`).
- The left side of `nums2` (i.e., `nums2[j-1]`) must be less than or equal to the right side of `nums1` (i.e., `nums1[i]`).

4. Finding the Median:
If the combined length of the arrays is odd, the median is the maximum of the left sides. If even, it is the average of the maximum of the left sides and the minimum of the right sides.

Pseudocode

Here is a pseudocode representation of the algorithm:

```python
def findMedianSortedArrays(nums1, nums2):
if len(nums1) > len(nums2):
nums1, nums2 = nums2, nums1

x, y = len(nums1), len(nums2)
low, high = 0, x

while low <= high:
partitionX = (low + high) // 2
partitionY = (x + y + 1) // 2 - partitionX

maxX = float('-inf') if partitionX == 0 else nums1[partitionX - 1]
minX = float('inf') if partitionX == x else nums1[partitionX]

maxY = float('-inf') if partitionY == 0 else nums2[partitionY - 1]
minY = float('inf') if partitionY == y else nums2[partitionY]
```

```
if maxX <= minY and maxY <= minX:
if (x + y) % 2 == 0:
return (max(maxX, maxY) + min(minX, minY)) / 2
else:
return max(maxX, maxY)
elif maxX > minY:
high = partitionX - 1
else:
low = partitionX + 1
```

Example Walkthrough

Let's walk through an example to solidify our understanding of the algorithm.

Consider:
- `nums1 = [1, 3]`
- `nums2 = [2]`

1. Since `nums1` is shorter, we proceed with binary search on it.
2. In the first iteration:
- `low = 0`, `high = 2` (length of `nums1`)
- `partitionX = 1`, `partitionY = 1`
- `maxX = 1`, `minX = 3`, `maxY = 2`, `minY = inf`
3. Check conditions:
- `maxX (1) <= minY (inf)` (True)
- `maxY (2) <= minX (3)` (True)
4. Since both conditions are satisfied and the total length is odd:
- The median is `max(1, 2) = 2.0`.

In another example:
- `nums1 = [1, 2]`
- `nums2 = [3, 4]`

1. Run binary search:
- `partitionX = 1`, `partitionY = 1`
- `maxX = 1`, `minX = 2`, `maxY = 3`, `minY = 4`
2. Conditions:
- `maxX (1) <= minY (4)` (True)
- `maxY (3) <= minX (2)` (False), so adjust `low` to `partitionX + 1`.
3. Repeat until the correct partitions are found:
- Finally, compute the median as `(2 + 3) / 2 = 2.5`.

Conclusion

The median of two sorted arrays problem is a well-known algorithmic challenge that can be solved

efficiently using binary search. By leveraging the properties of sorted arrays, we can find the median in logarithmic time, which is a significant improvement over the naive merging approach. This problem not only tests a candidate's understanding of binary search but also their ability to think critically about array manipulations. As such, mastering this solution is essential for anyone preparing for technical interviews or looking to enhance their algorithmic skills.

Frequently Asked Questions

What is the problem statement for 'Median of Two Sorted Arrays' on LeetCode?

The problem requires finding the median of two sorted arrays, nums1 and nums2, of sizes m and n respectively, in O(log(min(m,n))) time complexity.

What approach can be used to solve the 'Median of Two Sorted Arrays' problem?

A binary search approach can be used, where we partition both arrays and ensure that all elements on the left are less than or equal to the elements on the right.

What is the time complexity of the optimal solution for this problem?

The optimal solution has a time complexity of O(log(min(m,n))), where m and n are the lengths of the two arrays.

How do you handle edge cases in the 'Median of Two Sorted Arrays' problem?

You should consider cases where one or both arrays are empty, as well as when the total number of elements is odd or even to correctly calculate the median.

What is the median when the combined length of the two arrays is odd?

When the combined length is odd, the median is the middle element of the combined sorted array.

What is the median when the combined length of the two arrays is even?

When the combined length is even, the median is the average of the two middle elements of the combined sorted array.

Can you provide a brief outline of the algorithm to find the median?

1. Ensure nums1 is the smaller array. 2. Use binary search on nums1 to find the correct partition. 3. Calculate the left and right max/min values for both partitions. 4. Check if the partition is valid, and if so, compute the median based on the combined array length.

Median Of Two Sorted Arrays Leetcode Solution

Median Of Two Sorted Arrays Leetcode Solution

Related Articles

- mental health level of care assessment
- meet me in st louis dvd
- mental health in occupational therapy

Back to Home