Median of Two Sorted Arrays

Solving the "Median of Two Sorted Arrays" problem – a very famous question with an optimal solution in $O(\log(\min(m, n)))$ using binary search.

Goal: Find the median of two sorted arrays nums1 and nums2 in logarithmic time, not by merging.

Core Idea – Binary Search Partitioning:

We want to partition nums1 and nums2 such that:

- Left halves of both arrays together contain half of the total elements.
- All elements on the left side ≤ all elements on the right side.
- We binary search only on the smaller array (nums1), trying different cut points.

Partition Logic: Let:

- i = cut index in nums1
- $j = (m+n+1)/2 - i$ in nums2 (so left half has $(m+n+1)//2$ elements in total)

We define:

maxLeftX = (i == 0) ? −∞ : nums1[i-1]

minRightX = (i == m) ? +∞ : nums1[i]

maxLeftY = (j == 0) ? −∞ : nums2[j-1]

minRightY = (j == n) ? +∞ : nums2[j]

Valid Partition: If: maxLeftX <= minRightY && maxLeftY <= minRightX

Then:

- If (m+n) is even → median = (max(maxLeftX, maxLeftY) + min(minRightX, minRightY)) / 2
- If odd → median = max(maxLeftX, maxLeftY)

Binary Search: Adjust the binary search:

- If maxLeftX > minRightY → move high = i - 1
- Else → move low = i + 1

Example: For nums1 = [2, 3], nums2 = [2]:

- Partition such that left = [1], [2] → median is 2

Time Complexity: · $O(\log(\min(m, n)))$ – binary search on shortest array.