0004. 寻找两个正序数组的中位数

ITCharge 大约 5 分钟

- 标签：数组、二分查找、分治
- 难度：困难

题目链接

- 0004. 寻找两个正序数组的中位数 - 力扣

题目大意

描述：给定两个正序（从小到大排序）数组 $nums1$、$nums2$。

要求：找出并返回这两个正序数组的中位数。

说明：

- 算法的时间复杂度应该为 $O(\log (m + n))$ 。
- $nums1.length == m$。
- $nums2.length == n$。
- $0 \le m \le 1000$。
- $0 \le n \le 1000$。
- $1 \le m + n \le 2000$。
- $-10^6 \le nums1[i], nums2[i] \le 10^6$。

示例：

- 示例 1：

```py
输入：nums1 = [1,2], nums2 = [3,4]
输出：2.50000
解释：合并数组 = [1,2,3,4] ，中位数 (2 + 3) / 2 = 2.5
```

- 示例 2：

```py
输入：nums1 = [1,2], nums2 = [3,4]
输出：2.50000
解释：合并数组 = [1,2,3,4] ，中位数 (2 + 3) / 2 = 2.5
```

解题思路

思路 1：二分查找

单个有序数组的中位数是中间元素位置的元素。如果中间元素位置有两个元素，则为两个元素的平均数。如果是两个有序数组，则可以使用归并排序的方式将两个数组拼接为一个大的有序数组。合并后有序数组中间位置的元素，即为中位数。

当然不合并的话，我们只需找到中位数的位置即可。我们用 $n1$、$n2$ 来表示数组 $nums1$、$nums2$ 的长度，则合并后的大的有序数组长度为 $(n1 + n2)$。

我们可以发现：**中位数把数组分割成了左右两部分，并且左右两部分元素个数相等。**

- 如果 $(n1 + n2)$ 是奇数时，中位数是大的有序数组中第 $\lfloor \frac{(n1+n2)}{2} \rfloor + 1$ 的元素，单侧元素个数为 $\lfloor \frac{(n1+n2)}{2} \rfloor + 1$ 个（包含中位数）。
- 如果 $(n1 + n2)$ 是偶数时，中位数是第 $\lfloor \frac{(n1+n2)}{2} \rfloor$ 的元素和第 $\lfloor \frac{(n1+n2)}{2} \rfloor + 1$ 的元素的平均值，单侧元素个数为 $\lfloor \frac{(n1+n2)}{2} \rfloor$ 个。

因为是向下取整，上面两种情况综合可以写为：单侧元素个数为：$\lfloor \frac{(n1+n2+1)}{2} \rfloor$ 个。

我们用 k 来表示 $\lfloor \frac{(n1+n2+1)}{2} \rfloor$ 。现在的问题就变为了：**如何在两个有序数组中找到前 k 小的元素位置?**

如果我们从 $nums1$ 数组中取出前 $m1(m1 \le k)$ 个元素，那么从 $nums2$ 就需要取出前 $m2 = k - m1$ 个元素。

并且如果我们在 $nums1$ 数组中找到了合适的 $m1$ 位置，则 $m2$ 的位置也就确定了。

问题就可以进一步转换为：**如何从 $nums1$ 数组中取出前 $m1$ 个元素，使得 $nums1$ 第 $m1$ 个元素或者 $nums2$ 第 $m2 = k - m1$ 个元素为中位线位置。**

我们可以通过「二分查找」的方法，在数组 $nums1$ 中找到合适的 $m1$ 位置，具体做法如下：

1. 让 $left$ 指向 $nums1$ 的头部位置 0，$right$ 指向 $nums1$ 的尾部位置 $n1$。
2. 每次取中间位置作为 $m1$，则 $m2 = k - m1$。然后判断 $nums1$ 第 $m1$ 位置上元素和 $nums2$ 第 $m2 - 1$ 位置上元素之间的关系，即 $nums1[m1]$ 和 $nums2[m2 - 1]$ 的关系。

    1. 如果 $nums1[m1] < nums2[m2-1]$，则 $nums1$ 的前 $m1$ 个元素都不可能是第 k 个元素。说明 $m1$ 取值有点小了，应该将 $m1$ 进行右移操作，即 $left = m1 + 1$。
    2. 如果 $nums1[m1] \geq nums2[m2-1]$，则说明 $m1$ 取值可能有点大了，应该将 $m1$ 进行左移。根据二分查找排除法的思路（排除一定不存在的区间，在剩下区间中继续查找），这里应取 $right = m1$。
3. 找到 $m1$ 的位置之后，还要根据两个数组长度和 $(n1 + n2)$ 的奇偶性，以及边界条件来计算对应的中位数。

上面之所以要判断 $nums1[m1]$ 和 $nums2[m2-1]$ 的关系是因为：

> 如果 $nums1[m1] < nums2[m2-1]$，则说明：
>
> - 最多有 $m1 + m2 - 1 = k - 1$ 个元素比 $nums1[m1]$ 小，所以 $nums1[m1]$ 左侧的 $m1$ 个元素都不可能是第 k 个元素。可以将 $m1$ 左侧的元素全部排除，然后将 $m1$ 进行右移。

推理过程：

如果 $nums1[m1] < nums2[m2-1]$，则：

1. $nums1[m1]$ 左侧比 $nums1[m1]$ 小的一共有 $m1$ 个元素（$nums1[0]...nums1[m1-1]$ 共 $m1$ 个）。
2. $nums2$ 数组最多有 $m2 - 1$ 个元素比 $nums1[m1]$ 小（即便是 $nums2[m2-1]$ 左侧所有元素都比 $nums1[m1]$ 小，也只有 $m2 - 1$ 个）。
3. 综上所述，$nums1$、$nums2$ 数组中最多有 $m1 + m2 - 1 = k - 1$ 个元素比 $nums1[m1]$ 小。
4. 所以 $nums1[m1]$ 左侧的 $m1$ 个元素（$nums1[0]...nums1[m1-1]$）都不可能是第 k 个元素。可以将 $m1$ 左侧的元素全部排除，然后将 $m1$ 进行右移。

思路 1：代码

```py
class Solution:
    def findMedianSortedArrays(self, nums1: List[int], nums2: List[int]) -> float:
        n1 = len(nums1)
        n2 = len(nums2)
        if n1 > n2:
            return self.findMedianSortedArrays(nums2, nums1)
```

```python
        k = (n1 + n2 + 1) // 2
        left = 0
        right = n1
        while left < right:
            m1 = left + (right - left) // 2    # 在 nums1 中取前 m1 个元素
            m2 = k - m1                        # 在 nums2 中取前 m2 个元素
            if nums1[m1] < nums2[m2 - 1]:      # 说明 nums1 中所元素不够多，
                left = m1 + 1
            else:
                right = m1

        m1 = left
        m2 = k - m1

        c1 = max(float('-inf') if m1 <= 0 else nums1[m1 - 1], float('-inf') if
m2 <= 0 else nums2[m2 - 1])
        if (n1 + n2) % 2 == 1:
            return c1

        c2 = min(float('inf') if m1 >= n1 else nums1[m1], float('inf') if m2 >=
n2 else nums2[m2])

        return (c1 + c2) / 2
```

思路 1：复杂度分析

- **时间复杂度**：$O(\log(m + n))$ 。
- **空间复杂度**：$O(1)$。