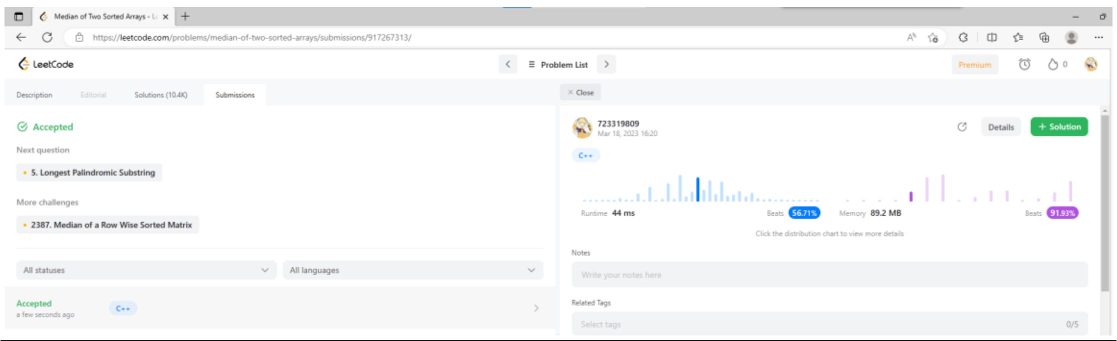
Median of Two Sorted Arrays - L
https://leetcode.com/problems/median-of-two-sorted-arrays/submissions/917267313/
LeetCode
Problem List
Premium
Description
Editorial
Solutions (10.4K)
Submissions
Close

Accepted
Next question
5. Longest Palindromic Substring
More challenges
2387. Median of a Row Wise Sorted Matrix

All statuses | All languages

Accepted a few seconds ago C++

723319809
Mar 18, 2023 16:20
C++

Runtime 44 ms Beats 56.71%
Memory 89.2 MB Beats 91.93%

Click the distribution chart to view more details

Notes
Write your notes here
Related Tags
Select tags 0/5

源代码如下：

```cpp
#include<iostream>
#include<vector>
using namespace std;

class Solution {
private:
    int findKthMinimum(const vector<int>& nums1, const vector<int>&
nums2,
        const int index1left, const int index1right,
        const int index2left, const int index2right,
        const int k) {
        if (index1left > index1right)
            return nums2[index2left + k - 1];
        if (index2left > index2right)
            return nums1[index1left + k - 1];
        if (k == 1)
            return min(nums1[index1left], nums2[index2left]);
        int m = index1right - index1left + 1,
            n = index2right - index2left + 1;
        if (m == 1 && n == 1)
            return max(nums1[index1left],nums2[index2left]);
        if (m == 1) {
            if (nums1[index1left] < nums2[index2left + k - 2])
                return nums2[index2left + k - 2];
            else if (nums1[index1left] > nums2[index2left + k - 1])
                return nums2[index2left + k - 1];
            else
                return nums1[index1left];
        }
        if (n == 1) {
            if (nums2[index2left] < nums1[index1left + k - 2])
                return nums1[index1left + k - 2];
            else if (nums2[index2left] > nums1[index1left + k - 1])
                return nums1[index1left + k - 1];
            else
```

```cpp
                return nums2[index2left];
        }
        if (nums1[index1left + m / 2] < nums2[index2left + n / 2])
            if (k - 1 < (m + n) / 2)
                return findKthMinimum(nums1, nums2,
                    index1left, index1right,
                    index2left, index2left + n / 2 - 1,
                    k);//throw the right side of nums2
            else
                return findKthMinimum(nums1, nums2,
                    index1left + m / 2, index1right,
                    index2left, index2right,
                    k - m / 2);//throw the left side of nums1
        else
            if (k - 1 < (m + n) / 2)
                return findKthMinimum(nums1, nums2,
                    index1left, index1left + m / 2 - 1,
                    index2left, index2right,
                    k);//throw the right side of nums1
            else
                return findKthMinimum(nums1, nums2,
                    index1left, index1right,
                    index2left + n / 2, index2right,
                    k - n / 2);//throw the left side of nums2
    }
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>&
nums2) {
        int k = nums1.size() + nums2.size();
        if (k & 1)
            return findKthMinimum(nums1, nums2, 0, nums1.size() - 1, 0,
nums2.size() - 1, k / 2 + 1);
        return (findKthMinimum(nums1, nums2, 0, nums1.size() - 1, 0,
nums2.size() - 1, k / 2)
            + findKthMinimum(nums1, nums2, 0, nums1.size() - 1, 0,
nums2.size() - 1, k / 2 + 1)) / 2.0;
    }
    void findAllTheNums(vector<int>& nums1, vector<int>& nums2) {
        int k = nums1.size() + nums2.size();
        for (int i = 1; i <= k; i++) {
            cout << "The " << i;
            switch (i % 10) {
                case 1:
                    cout << "st";
                    break;
                case 2:
                    cout << "nd";
                    break;
                case 3:
                    cout << "rd";
                    break;
                default:
                    cout << "th";
                    break;
            }
            cout << " number is: " << findKthMinimum(nums1, nums2, 0,
nums1.size() - 1, 0, nums2.size() - 1, i) << endl;
        }
    }
};
int main() {
    Solution test;
    /*vector<int>nums1 = {0,3,5,7,9,11,13},
        nums2 = { 1,2,4,8,16 };*/
    //vector<int>nums1 = { 3,5,9,15,25,36,49,63 },
        //nums2 = { 1,2,4,8,16,32,64,128 };
    vector<int>nums1 = { 1 },
        nums2 = { 1 };
    //1,2,3,4,
    //5,8,9,15,
    //16,25,32,36,
    //49,63,64,128
    cout << "The median is: " << test.findMedianSortedArrays(nums1,
nums2) << endl;
    //test.findAllTheNums(nums1, nums2);
}
```