算法设计与应用基础: Homework No 2

16337341 朱志儒

1. House Robber II (#213)

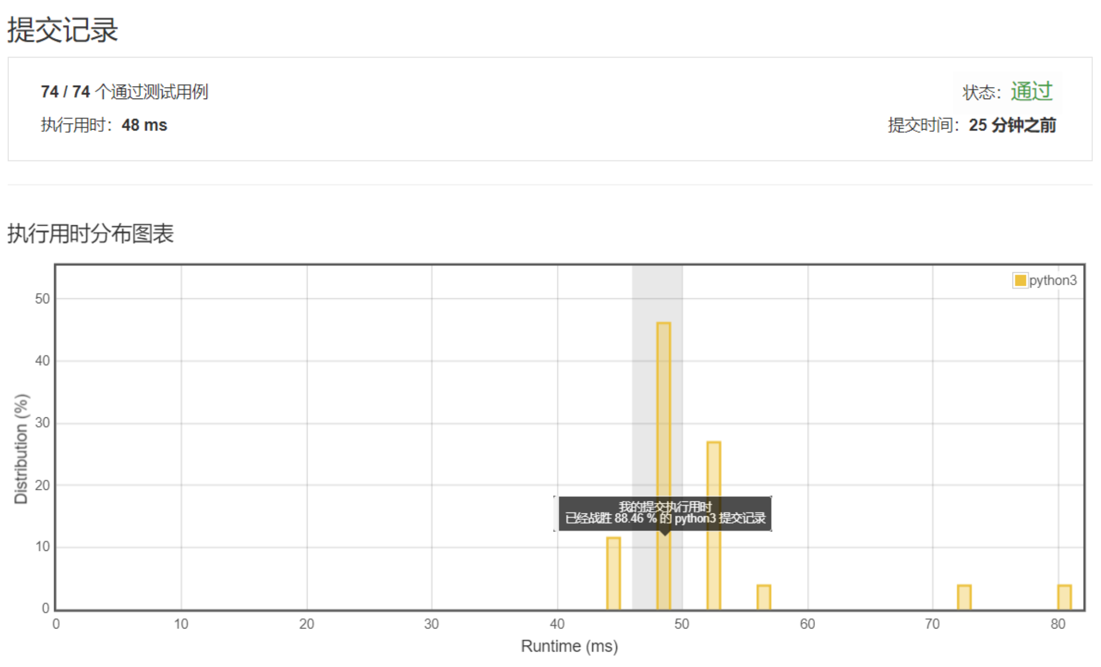
After robbing those houses on that street, the thief has found himself a new place for his thievery so that he will not get too much attention. This time, all houses at this place are arranged in a circle. That means the first house is the neighbor of the last one. Meanwhile, the security system for these houses remain the same as for those in the previous street.
Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight without alerting the police.

解题思路：若列表长度为 0 则输出 0，若长度为 1 则输出第一个元素，若长度为 2 则输出第一个和第二个中的最大的那个，若长度大于 2 则声明两个列表，第一个列表存储 0 到 n-2 最大和，第二个列表存储 1 到 n-1 最大的和，最后返回两个列表最后一个元素中最大的那个。

代码如下：

```
class Solution:
    def rob(self, nums):
        n = len(nums)
        result1 = [i for i in range(0, n - 1)]
        result2 = result1[:]
        if n == 0:
            return 0
        elif n == 1:
            return nums[0]
        elif n == 2:
            return max(nums)
        else:
            result1[0] = nums[0]
            result1[1] = max(nums[0], nums[1])
            for i in range(2, n - 1):
                result1[i] = max(result1[i - 2] + nums[i], result1[i - 1])
            result2[0] = nums[1]
            result2[1] = max(nums[1], nums[2])
            for i in range(3, n):
                result2[i-1] = max(result2[i-3] + nums[i], result2[i-2])
            return max(result2[-1], result1[-1])
```

结果：



提交记录

74 / 74 个通过测试用例

执行用时：48 ms

状态：通过

提交时间：25 分钟之前

执行用时分布图表

2. Triangle (#120)

Given a triangle, find the minimum path sum from top to bottom. Each step you may move to adjacent numbers on the row below.
For example, given the following triangle

```
[
     [2],
    [3,4],
   [6,5,7],
  [4,1,8,3]
]
```

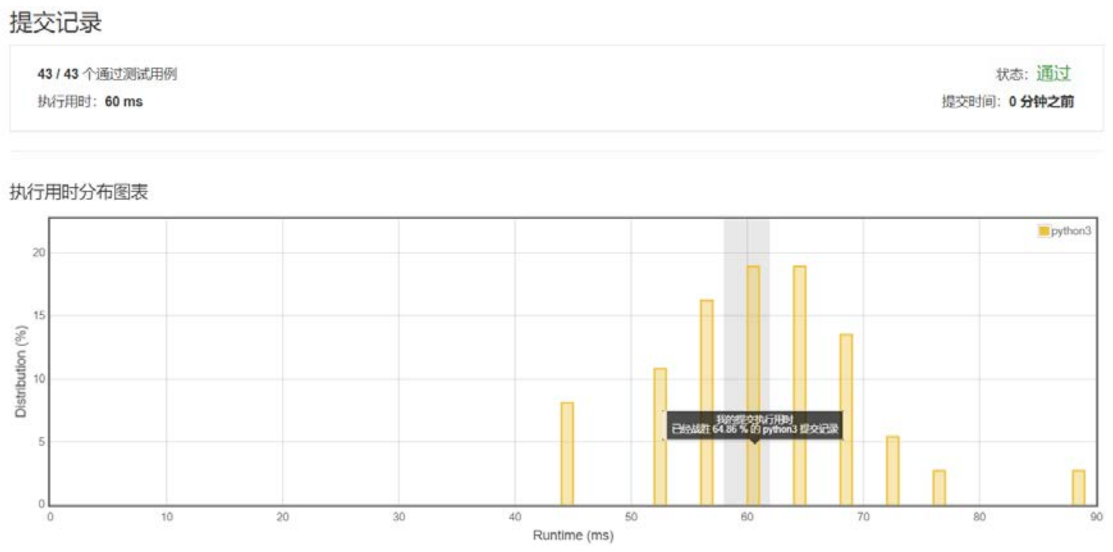
The minimum path sum from top to bottom is 11 (i.e., 2 + 3 + 5 + 1 = 11).
Note: Bonus point if you are able to do this using only O(n) extra space, where n is the total number of rows in the triangle.

解题思路：构建动态规划数组，保存前一行和当前行符合题目要求的路径和，以自底向上的方式，最终归一到第 0 行的 0 元素位置。那么动态规划数组的第 0 个元素即为最小路径和。

代码如下：

```
class Solution:
    def minimumTotal(self, triangle):
        """
        :type triangle: List[List[int]]
        :rtype: int
        """
        result = triangle[-1]
        for i in range(len(triangle) - 2, -1, -1):
            for j in range(len(triangle[i])):
                result[j] = min(result[j], result[j + 1]) + triangle[i][j]
        return result[0]
```

结果：


提交记录

43 / 43 个通过测试用例　　　　状态：通过

执行用时：60 ms　　　　提交时间：0 分钟之前

执行用时分布图表

3. Jump Game (#55)

Given an array of non-negative integers, you are initially positioned at the first index of the array.
Each element in the array represents your maximum jump length at that position.
Determine if you are able to reach the last index.

For example:

A = [2,3,1,1,4], return true.

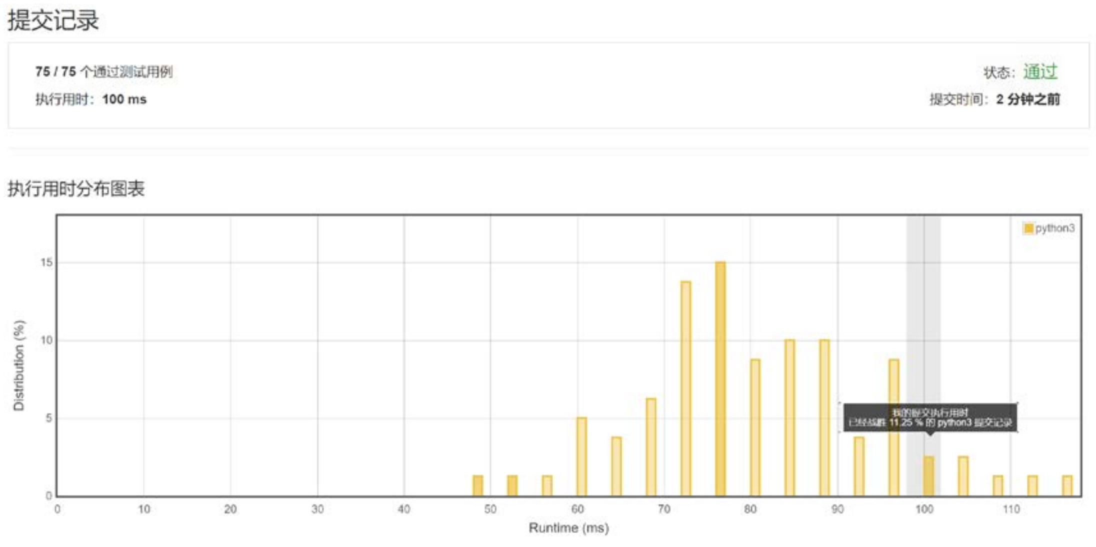
A = [3,2,1,0,4], return false.

解题思路：遍历列表，根据列表元素的值递增下标 pos，当 pos >= len(nums) - 1 时，则返回 true，如果 pos 的增长为 0 时，则返回 false。

代码如下：

```
class Solution:
    def canJump(self, nums):
        """
        :type nums: List[int]
        :rtype: bool
        """
        pos = 0
        while (pos < len(nums) - 1):
            index = 0
            max_num = 0
            for i in range(1, nums[pos] + 1):
                if pos + i >= len(nums) - 1:
                    return True
                if i + nums[pos + i] > max_num:
                    max_num = i + nums[pos + i]
                    index = i
            if max_num == 0:
                return False
            pos += index
        return True
```

结果：



提交记录

75 / 75 个通过测试用例

执行用时： 100 ms

状态： 通过

提交时间： 2 分钟之前

执行用时分布图表

我的提交执行用时
已经战胜 11.25 % 的 python3 提交记录

4. Gas Station (#134)

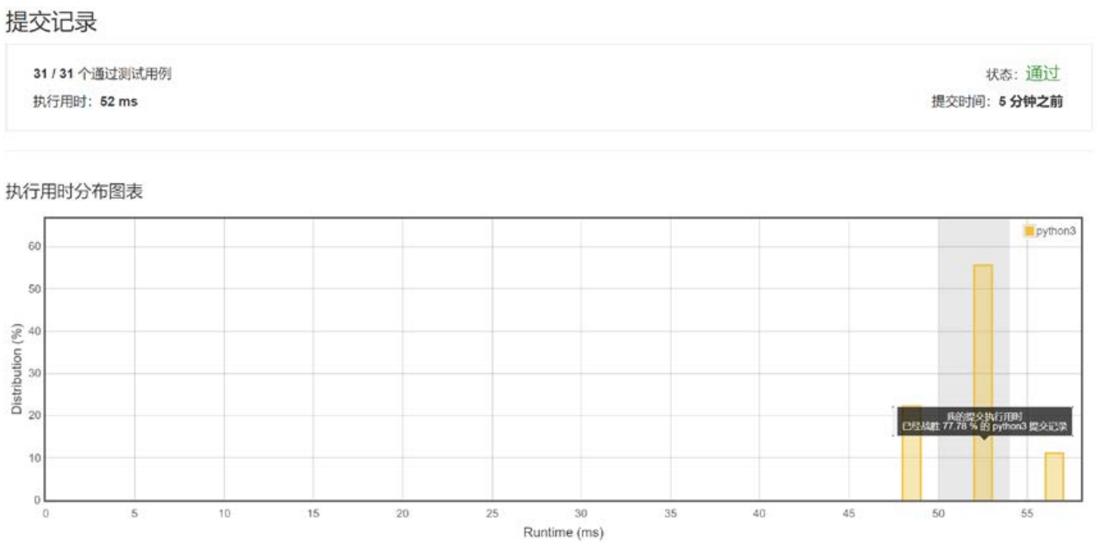
There are N gas stations along a circular route, where the amount of gas at station i is gas[i].
You have a car with an unlimited gas tank and it costs cost[i] of gas to travel from station i to its next station (i+1). You begin the journey with an empty tank at one of the gas stations.
Return the starting gas station's index if you can travel around the circuit once, otherwise return -1.
Note: The solution is guaranteed to be unique.

解题思路：若 gas 的总和小于 cost 的总和则返回-1。累计 gas 与 cost 对应元素的差值，若出现负数则将下标指向下一个，这样遍历整个列表，最后返回下标。

代码如下：

```
class Solution:
    def canCompleteCircuit(self, gas, cost):
        total, rest, start = 0, 0, 0
        for i in range(len(gas)):
            rest = rest + gas[i] - cost[i]
            total = total + gas[i] - cost[i]
            if rest < 0:
                start = i + 1
                rest = 0
        if total >= 0:
            return start
        else:
            return -1
```

结果:



提交记录

31 / 31 个通过测试用例　　状态: 通过

执行用时: 52 ms　　提交时间: 5 分钟之前

执行用时分布图表

5. Sort List (#148)

Sort a linked list in O(n log n) time using constant space complexity.

Example 1:

Input: 4->2->1->3

Output: 1->2->3->4

Example 2:

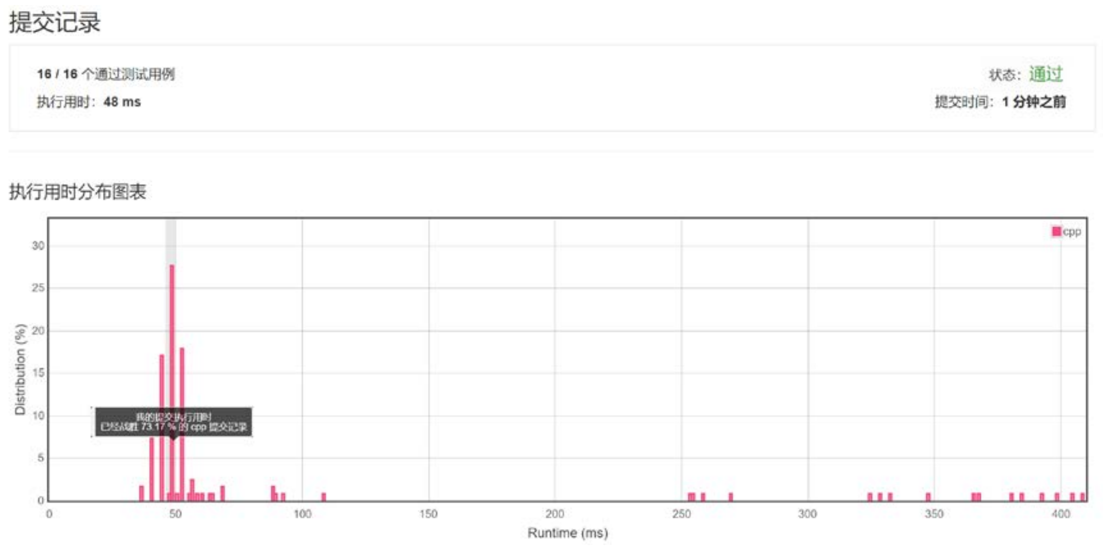
Input: -1->5->3->4->0

Output: -1->0->3->4->5

解题思路：使用归并排序的算法。

代码如下：
```
class Solution {
public:
    ListNode * sortList(ListNode* head) {
        if (head == nullptr || head->next == nullptr) return head;
        ListNode *fast = head;
        ListNode *slow = head;
        while (fast->next != nullptr && fast->next->next != nullptr) {
            fast = fast->next->next;
            slow = slow->next;
        }
        fast = slow->next;
        slow->next = nullptr;
        ListNode *p1 = sortList(head);
        ListNode *p2 = sortList(fast);
        return merge(p1, p2);
    }
    ListNode *merge(ListNode *p1, ListNode *p2) {
        if (p1 == nullptr) return p2;
        if (p2 == nullptr) return p1;
        ListNode tmp = ListNode(0);
        ListNode *p = &tmp;
        while (p1 && p2) {
            if (p1->val > p2->val) {
                p->next = p2;
                p2 = p2->next;
            }
            else {
                p->next = p1;
                p1 = p1->next;
            }
            p = p->next;
        }
        if (p1 == nullptr) p->next = p2;
        else if (p2 == nullptr) p->next = p1;
        return tmp.next;
    }
};
```

结果：


提交记录

16 / 16 个通过测试用例
执行用时：48 ms

状态：通过
提交时间：1 分钟之前

执行用时分布图表

6. Kth Largest Element in an Array (#215)

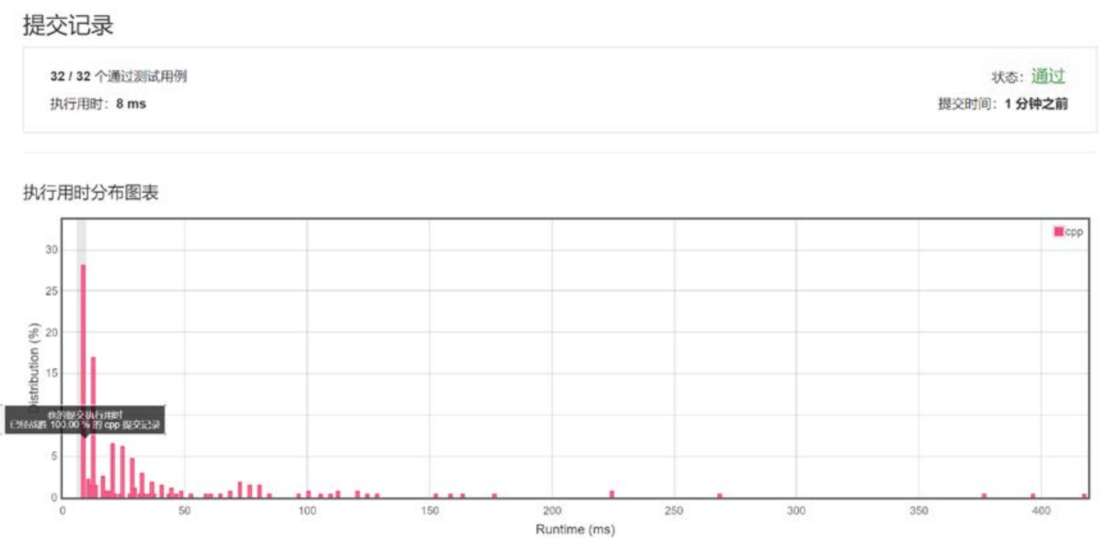
Find the **k**th largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.
For example, given [3,2,1,5,6,4] and k = 2, return 5.
Note: You may assume k is always valid, 1 ≤ k ≤ array's length.

解题思路：将列表排序返回第 k 个最大的数。

代码如下：

```
#include<algorithm>
using namespace std;

class Solution {
public:
    int findKthLargest(vector<int>& nums, int k) {
        sort(nums.begin(), nums.end());
        return nums[nums.size() - k];
    }
};
```

结果:



提交记录

32 / 32 个通过测试用例

执行用时: 8 ms

状态: 通过

提交时间: 1 分钟之前

执行用时分布图表

您的提交执行用时
已经战胜 100.00 % 的 cpp 提交记录

7. Merge K Sorted Lists (#23)

Merge k sorted linked lists and return it as one sorted list. Analyze and describe its complexity.

Example:

Input:

[

  1->4->5,

  1->3->4,

  2->6

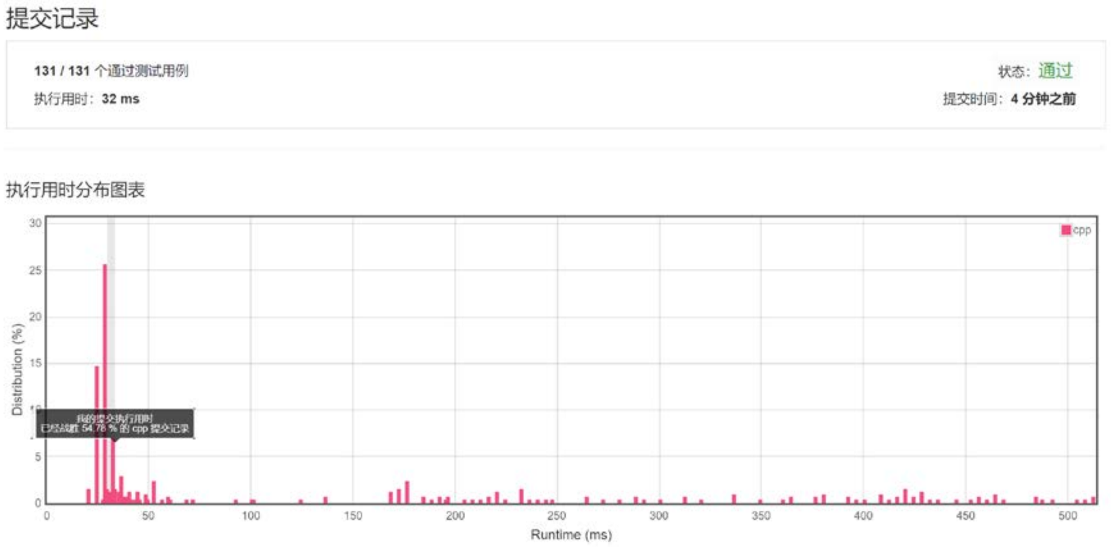
]

Output: 1->1->2->3->4->4->5->6

解题思路: 将所有的列表元素存储在一个数组中，然后将该数组排序，最后将该数组中的元素组成一个列表。

代码如下:

```
#include <algorithm>
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        int data[10000];
        int index = 0;
        bool flag = 0;
        if (lists.size() == 0) return nullptr;
        for (int i = 0; i < lists.size(); ++i) {
            while (lists[i] != nullptr) {
                data[index++] = lists[i]->val;
                lists[i] = lists[i]->next;
                flag = 1;
            }
        }
        if (flag == 0) return nullptr;
        sort(data, data + index);
        ListNode *head = new ListNode(data[0]);
        ListNode *p = head;
        for (int i = 1; i < index; ++i) {
            p->next = new ListNode(data[i]);
            p = p->next;
        }
        return head;
    }
};
```
结果:


提交记录

131 / 131 个通过测试用例

执行用时: **32 ms**

状态: 通过

提交时间: **4 分钟之前**

执行用时分布图表

8. Median of Two Sorted Arrays (#4)

There are two sorted arrays **nums1** and **nums2** of size m and n respectively.
Find the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).
Example 1:

```
nums1 = [1, 3]

nums2 = [2]

The median is 2.0
```

Example 2:

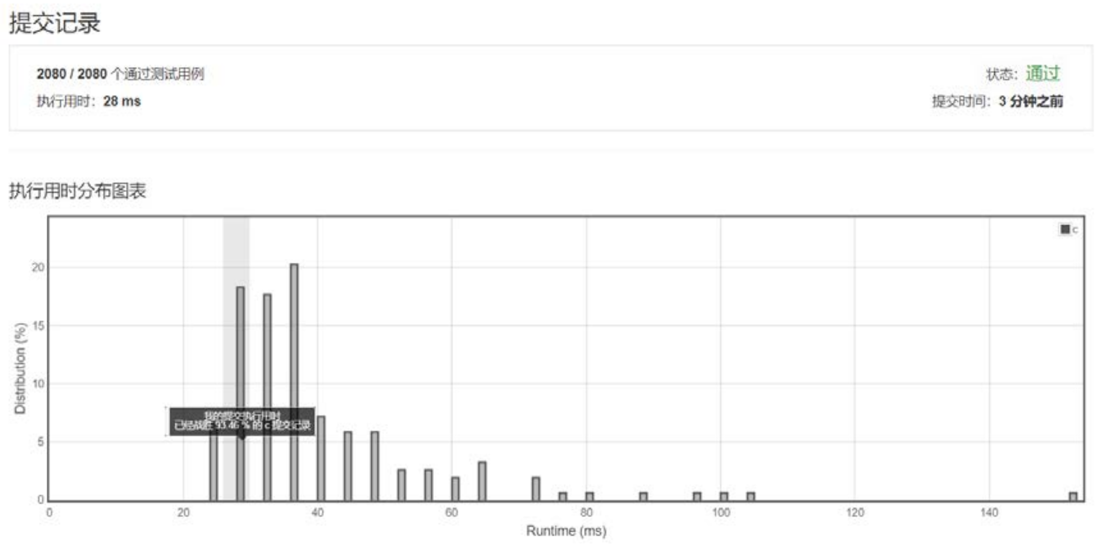
```
nums1 = [1, 2]

nums2 = [3, 4]

The median is (2 + 3)/2 = 2.5
```

解题思路：将两个数组合并取中间值。

代码如下：

```c
double findMedianSortedArrays(int* nums1, int nums1Size, int* nums2, int
nums2Size) {
    int count = 0, total = nums1Size + nums2Size;
    int data[10000];
    if (nums1Size == 0 && nums2Size != 0) {
        if (total % 2 == 1) return (double)nums2[total / 2];
        else return (nums2[total / 2] + nums2[total / 2 - 1]) / 2.0;
    }
    else if (nums1Size != 0 && nums2Size == 0) {
        if (total % 2 == 1) return (double)nums1[total / 2];
        else return (nums1[total / 2] + nums1[total / 2 - 1]) / 2.0;
    }
    else {
        int p1 = 0, p2 = 0;
        while (count != total / 2 + 1) {
            if (p1 >= nums1Size)
                data[count++] = nums2[p2++];
            else if (p2 >= nums2Size)
                data[count++] = nums1[p1++];
            else if (p1 < nums1Size && p2 < nums2Size) {
                if (nums1[p1] < nums2[p2])
                    data[count++] = nums1[p1++];
                else
                    data[count++] = nums2[p2++];
            }
            else
                break;
        }
        if (total % 2 == 1) return (double)data[total / 2];
        else
            return (((double)data[total / 2 - 1] + (double)data[total /
2]) / 2.0);
    }
}
```

结果：



提交记录

2080 / 2080 个通过测试用例

执行用时： **28 ms**

状态： 通过

提交时间： **3 分钟之前**

执行用时分布图表

我的提交执行用时
已经战胜 93.46 % 的 c 提交记录