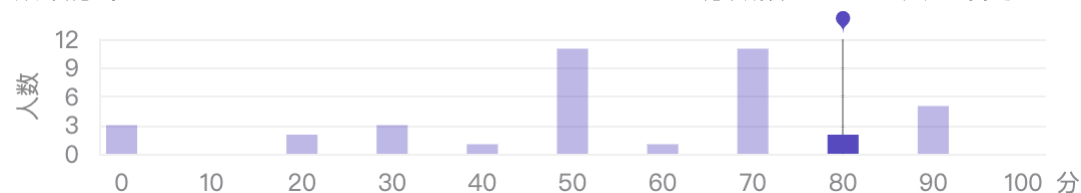


yifan-1 yifan-1

总分
80.00/100排名
6/39

成绩分布



编程语言 Python3

竞赛名称:

LeetCode 第二期第二周竞

结束时间:

2021-12-11 14:40

登录设备:

Windows 10 | Chrome 96.0.4664

开始时间:

2021-12-11 04:00

登录 IP 地址:

115.220.226.55(中国杭州市电信)

邮箱:

372104922@qq.com

答题详情

#1 编程题 1000582



分数 20.00/20

最小化排队等待时间

有 n 个人在等待办理事务，其中第 i 个人的事务需要 $w[i]$ 分钟完成。现在希望你安排他们办理事务的顺序，从而使得每个人的等待时间之和最小，并返回最小的排队等待时间总和。

示例 1:

输入: $w = [1, 3, 2]$

输出: 4

解释:

- 首先安排第一个人办理, 第一个人等待时间为 0。需要 1 分钟。
- 然后安排第三个人办理, 第三个人等待第一个人等待了 1 分钟, 他自己的业务办理需要 2 分钟。
- 最后安排第二个人办理, 他已经等待了 3 分钟, 他自身的业务办理需要 3 分钟。

第三个人等待了 1 分钟, 第二个人等待了 3 分钟, 因此总的等待时间是 4 分钟。这是最少的等待方案。

提示:

- `1 <= w.length <= 10^4`
- `1 <= w[i] <= 100`

用时 00:11:15

提交次数 1 2021/12/11 12:03

提交结果:

通过

通过测试用例:

51/51

语言:

python3

执行用时:

88 ms

消耗内存:

15.7 MB

```
class Solution:
    def minimumWaitingTime(self, w: List[int]) -> int:
        n = len(w)
        w.sort()
        dp = [0] * n
        dp[0] = w[0]
        for i in range(1, n):
            dp[i] = dp[i - 1] + w[i]
        res = 0
        for i in range(0, len(dp) - 1):
            res += dp[i]
        return res
```

#2 编程题 1000583 

分数 30.00/30

统计出现次数

给定一个有序数组 `nums`，以及一个目标数 `target`。返回数组 `nums` 中 `target` 的出现次数。

请你实现时间复杂度为 $O(\log n)$ 并且只使用 常数级别额外空间 的解决方案。

示例 1:

输入: `nums = [1,1,2,2,2,2,3,3,5,6,7,7,7]`, `target = 3`

输出: `2`

提示:

- `1 <= nums.length <= 10^5`
- `-10^5 <= nums[i], target <= 10^5`

用时 00:08:25

提交次数 1 2021/12/11 12:12 

提交结果:

通过

通过测试用例:

52/52

语言:

python3

执行用时:


96 ms

消耗内存:

24 MB

```
class Solution:
    def countOccurrences(self, nums: List[int], target: int) -> int:
        n = len(nums) - 1
        l, r = 0, n
        while l < r:
```

```
        mid = (l + r) // 2
        if nums[mid] >= target:
            r = mid
        else:
            l = mid + 1
    L = r
    l, r = 0, n
    while l < r:
        mid = (l + r + 1) // 2
        if nums[mid] <= target:
            l = mid
        else:
            r = mid - 1
    return r - L + 1
```

#3 编程题 1000584 

分数 20.00/20

上一个排列

实现获取 **上一个排列** 的函数，算法需要将给定数字序列重新排列成字典序中上一个更小的排列。

注意：如果不存在前一个更小的排列，则将数字重新排列成最大的排列（即降序排列）。

示例 1：

输入：nums = [3,0,4,2,1]

输出：[3,0,4,1,2]

示例 2：

输入：nums = [0,1,2,3]

输出：[3,2,1,0]

示例 3：

输入：nums = [0,0,2,1,0]

输出: `[0,0,2,0,1]`

提示:

- `1 <= nums.length <= 1000`

用时 00:29:03

提交次数 1

2021/12/11 12:33

提交结果:

通过

通过测试用例:

79/79

语言:

python3

执行用时:

40 ms

消耗内存:

15.6 MB

```
class Solution:
    def previousPermutation(self, permutation: List[int]) -> List[int]:
        n = len(permutation)
        if n == 1:
            return permutation
        for i in range(n - 2, -1, -1):
            if permutation[i] > permutation[i + 1]:
                break
        if permutation[i] < permutation[i + 1]:
            permutation.reverse()
            return permutation
        cur = -sys.maxsize
        cur_idx = -1
        for j in range(i + 1, n):
            if permutation[j] < permutation[i]:
                if permutation[j] >= cur:
                    cur = permutation[j]
                    cur_idx = j
        permutation[i], permutation[cur_idx] = permutation[cur_idx], permutation[i]
        l, r = i + 1, n - 1
        while l < r:
            permutation[l], permutation[r] = permutation[r], permutation[l]
```

```
l += 1
r -= 1
return permutation
```

#4 编程题 1000585 

分数 0.00/10

最大价值

给定整数数组 `nums` 和 `arr`，定义 `arr` 中每个数的「价值」为 `nums` 中比该数大的元素个数与比该数小的元素个数之积，求 `arr` 中「价值」最大的数的「价值」，并将结果对 `1e9+7` 取模。

示例 1：

输入： `nums = [1,4,5,8,2]`, `arr = [3,5,4]`

输出： 6


解释：

- `arr` 中第一个元素 3 的「价值」为 $2*3 = 6$ (`nums` 中有 2 个数比 3 小，有 3 个数比 3 大)。
- `arr` 中第二个元素 5 的「价值」为 $3*1 = 3$ (`nums` 中有 3 个数比 5 小，有 1 个数比 5 大)。
- `arr` 中第三个元素 4 的「价值」为 $2*2 = 4$ (`nums` 中有 2 个数比 4 小，有 2 个数比 4 大)。

提示：

- `1 <= nums.length, arr.length <= 10^5`
- `-10^5 <= nums[i], arr[i] <= 10^5`

用时 00:46:39

提交次数 5 2021/12/11 14:40 

提交结果：

解答错误

通过测试用例：

24/35

语言：

python3

执行用时：

消耗内存：

N/A

N/A

```
class Solution:
    def maximumValue(self, nums: List[int], arr: List[int]) -> int:
        def value(nums, target):
            l, r = 0, n - 1
            while l < r:
                mid = l + (r - l) // 2
                if nums[mid] >= target:
                    r = mid
                else:
                    l = mid + 1
            small, big = -1, -1
            res = 0
            for j in range(r, -1, -1):
                if nums[j] < target:
                    small = j
                    break
            for k in range(r, n - 1):
                if nums[k] > target:
                    big = k
                    break
            if small == -1 or big == -1:
                return 0
            res = (small + 1) * (n - big)
            return res
        n = len(nums)
        nums.sort()
        maxvalue = -sys.maxsize
        for num in arr:
            res = value(nums, num)
            if res > maxvalue:
                maxvalue = res
        if maxvalue > 1e9+7:
            maxvalue %= 1e9+7
        return maxvalue
```

#5 编程题 1000586

分数 10.00/10

打怪兽

你正在玩一个叫做「打怪兽」的游戏，游戏初始得分为 `0`，初始攻击值为 `atk`。二维数组 `monsters` 存储了 `n` 只怪兽的属性，其中 `monsters[i] = [d, t]`，其中 `d` 表示该怪兽的防御值，`t` 表示打败该怪兽可以提升的攻击值。游戏规则如下：

- 每次你能 **任意** 挑选 **未被打败** 的怪兽与其对战；
- 当你的攻击值 `atk` 大于等于怪兽的防御值 `d` 时，你才可以打败该只怪兽；
- 当打败了一只怪兽后，你的得分可增加 `atk - d` 分，攻击值可增加 `t` 点。

求击败所有怪兽后的最大得分，若无法打败所有的怪兽请返回 `-1`。

注意：

- 测试数据保证不存在属性完全相同的怪兽；
- 测试数据保证最大得分在 `int` 表示范围内。

示例 1：

输入： `atk = 2, monsters = [[4,3],[2,1],[0,5],[4,1]]`

输出： `20`

解释：

- 与下标为 `2` 的怪兽对战，得分增加 `2 - 0 = 2` 分变为 `2` 分，攻击值增加 `5` 点变为 `7` 点。
- 与下标为 `0` 的怪兽对战，得分增加 `7 - 4 = 3` 分变为 `5` 分，攻击值增加 `3` 点变为 `10` 点。
- 与下标为 `1` 的怪兽对战，得分增加 `10 - 2 = 8` 分变为 `13` 分，攻击值增加 `1` 点变为 `11` 点。
- 与下标为 `3` 的怪兽对战，得分增加 `11 - 4 = 7` 分变为 `20` 分，攻击值增加 `1` 点变为 `12` 点。

提示：

- `0 <= atk <= 10^3`
- `1 <= monsters.length <= 10^3`
- `0 <= monsters[i][0] <= 10^3`
- `0 <= monsters[i][1] <= 50`

用时 00:42:22

提交次数 1

2021/12/11 14:04

提交结果:

通过

通过测试用例:

60/60

语言:

python3

执行用时:

60 ms

消耗内存:

15.8 MB

```
class Solution:
    def maximumScore(self, atk: int, monsters: List[List[int]]) -> int:
        monsters.sort(key = lambda x:x[1], reverse = True)
        que = collections.deque()
        for num in monsters:
            que.append(num)
        res = 0
        stk = []
        while que:
            while que[0][0] > atk:
                stk.append(que.popleft())
            if not que:
                return -1
            res += atk - que[0][0]
            atk += que[0][1]
            que.popleft()
            while stk:
                que.appendleft(stk.pop())
        return res
```

#6 编程题 1000587



分数 0.00/10

区间最大值

给定整数数组 `nums` 和查询数组 `queries`，其中 `queries[i]=[l,r]`。对于每个查询 `i`，求子数组 `nums[l...r]` 中的最大值。

请你返回 `ans` 数组，其中 `ans[i]` 是第 `i` 个查询的答案。

示例 1：

输入： `nums = [-3,-1,2,-3,-2,-5]`, `queries = [[2,5],[1,1],[2,3],[4,5],[0,5]]`

输出： `[2,-1,2,-2,2]`

解释：

- 第 1 个查询区间 `[2,5]` 中的最大值为 2。
- 第 2 个查询区间 `[1,1]` 中的最大值为 -1。
- 第 3 个查询区间 `[2,3]` 中的最大值为 2。
- 第 4 个查询区间 `[4,5]` 中的最大值为 -2。
- 第 5 个查询区间 `[0,5]` 中的最大值为 2。

提示：

- `1 <= nums.length, queries.length <= 10^5`
- `-10^9 <= nums[i] <= 10^9`
- `0 <= queries[i][0] <= queries[i][1] < nums.length`

用时 00:30:03

提交次数 2

2021/12/11 14:37



提交结果：

解答错误

通过测试用例：

1/28

语言：

python3

执行用时：

N/A

消耗内存：

N/A

```
class Solution:
    def solve(self, nums: List[int], queries: List[List[int]]) -> List[int]:
        res = []
        if not queries:
            return res
```

```
for q in queries:
    l, r = q[0], q[1]
    max = nums[l]
    for i in range(l, r):
        if nums[i] > max:
            max = nums[i]
    res.append(max)
return res
```