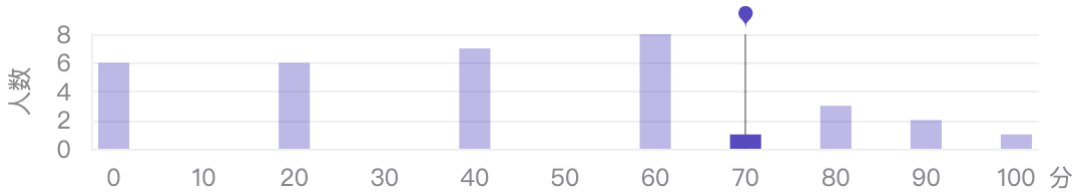
yifan-1 yifan-1

总分
70.00/100

排名
7/34

成绩分布

分数段 70 - 75 共 1 名考生



编程语言 Python3

竞赛名称:
LeetCamp 第二期第四周竞

开始时间:
2021-12-25 04:00

结束时间:
2021-12-25 14:22

登录 IP 地址:
183.157.110.207(中国杭州市电信)

登录设备:
Windows 10 | Chrome 96.0.4664

邮箱:
372104922@qq.com

答题详情

#1 编程题 1000616

分数 20.00/20

统计岛屿数量

仅由 `1`（陆地）和 `0`（水）组成的的二维网格信息记录于二维数组 `grid` 中，请你求出该网格中岛屿的数量。

岛屿总是被水包围，并且每座岛屿只能由水平、竖直以及对角线八个方向上相邻的陆地连接形成。

此外，你可以假设该网格的四条边均被水包围。

示例 1：

> 输入：
> `grid = [[1,0,1],[0,1,1],[1,0,1]]`
>
> 输出：`1`
>
> 解释：
> 如图所示，该网格中仅有一个岛屿。

提示：

- `1 <= grid.length, grid[i].length <=500`
- `grid[i][j]` 仅为 `0` 或 `1`

用时 00:11:38　　提交次数 1　　2021/12/25 12:28

提交结果：	通过测试用例：	语言：
通过	54/54	python3
执行用时：	消耗内存：	
2856 ms	20.3 MB	

```
class Solution:
    def countIslands(self, grid: List[List[int]]) -> int:
        nr = len(grid)
        if not nr:
            return 0
        nc = len(grid[0])
        visited = [[False for _ in range(nc)] for _ in range(nr)]

        num_islands = 0
        for r in range(nr):
            for c in range(nc):
                if grid[r][c] == 1 and visited[r][c] == False:
                    num_islands += 1
                    visited[r][c] = True
                    self.dfs(grid, r, c, visited)
        return num_islands

    def dfs(self, grid, r, c, visited):
        nr, nc = len(grid), len(grid[0])
         for x, y in [(r - 1, c), (r + 1, c), (r, c - 1), (r, c + 1), (r +
1, c + 1), (r + 1, c - 1), (r - 1, c + 1), (r - 1, c - 1)]:
                if 0 <= x < nr and 0 <= y <nc and grid[x][y] == 1 and
visited[x][y] == False:
                visited[x][y] = True
                self.dfs(grid, x, y, visited)
```

#2 编程题 1000617 分数 20.00/20

整数之和

给定含有若干个整数的一维数组 `nums`，求从中取出若干整数相加之和等于 `target` 的方案数。

注意：

- 每种方案中每个整数仅能使用一次；
- `nums` 中的数字各不相同。

示例 1：

> 输入：
> `nums = [31,18,-11,13], target = 20`
>
> 输出：`2`
>
> 解释：
> 两种方案如下：
>
> - `31 + (-11) = 20`
> - `18 + (-11) +13 = 20`

提示：

- `1 <= nums.length <= 15`
- `-10^7 <= nums[i], target <=10^7`

用时 00:24:16 提交次数 1 2021/12/25 12:53

提交结果：	通过测试用例：	语言：
通过	46/46	python3
执行用时：	消耗内存：	
108 ms	17 MB	

```python
class Solution:
    def countPlans(self, nums: List[int], target: int) -> int:
        res = 0
        for i in range(1, len(nums) + 1):
            res += self.kSum(nums, i, target)
        return res

    def kSum(self, A, k, target):
        # write your code here
        if(k == 1):
            count = 0
            for num in A:
                if(num == target):
                    count += 1
            return count
```

```
        def twoSum(start, target):
            count = 0
            i = start
            seen = set()
            while(i<len(A)):
                if(A[i] in seen):
                    count += 1
                seen.add(target - A[i])
                i += 1
            return count

        if(k == 2):
            return twoSum(0, target)

        A.sort()
        @lru_cache(None)
        def findKSum(start, k, target):
            if(k == 2):
                return twoSum(start, target)
            count = 0
            for i in range(start,len(A)):
                if(len(A)-i<k):
                    break
                count += findKSum(i+1, k-1, target-A[i])
            return count

        return findKSum(0, k, target)
```

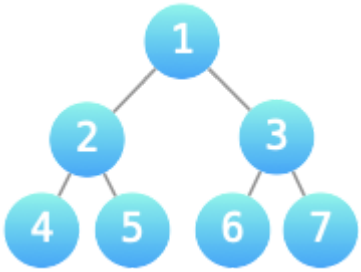
#3 编程题 1000618 分数 20.00/20

二叉树的反向层序遍历

给你一棵二叉树的根结点 `root` ，请你返回其按 **反向层序遍历** 得到的结点值。 （即逐层地，**从右到左** 访问所有结点）。

示例 1：

> 输入：`root = [1,2,3,4,5,6,7]`
>
> 输出：`[[1],[3,2],[7,6,5,4]]`
>
> 解释：
> 如图所示，逐层地**从右到左** 访问所有结点得到的结点值为 `[[1],[3,2],[7,6,5,4]]` 。



提示：

- `1 <= 结点数 <=10^4`

- `1 <= 结点值 <=10^4`

用时 00:02:53 提交次数 1 2021/12/25 12:56

提交结果：	通过测试用例：	语言：
通过	81/81	python3
执行用时：	消耗内存：	
532 ms	20.2 MB	

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None
class Solution:
    def solve(self, root: TreeNode) -> List[List[int]]:
        if not root:
            return []
        q = collections.deque()
        q.append(root)
        res = []
        while q:
            size = len(q)
            tmp = []
            for _ in range(size):
                x = q.popleft()
                tmp.append(x.val)
                if x.left: q.append(x.left)
                if x.right: q.append(x.right)
            res.append(tmp[::-1])
        return res
```

#4 编程题 1000619 分数 0.00/20

走迷宫

迷宫地宫地形信息记录于一维字符串数组 `maze` 中，`maze[i]` 为仅由 `"."`、`"X"`、`"S"` 和 `"E"` 组成的字符串，其中：

- `"X"` 表示障碍物，不可通过；
- `"."` 表示空地，可以通行；
- `"S"` 表示迷宫入口；
- `"E"` 表示迷宫出口。

请你求出入口到出口的最短路径的长度，如果从入口无法到达出口，请返回 `-1`。

注意：

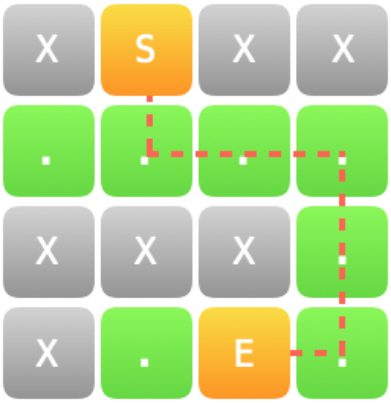
- 迷宫中仅有一个入口和一个出口。

示例 1：

输入：
`maze = ["XSXX","....","XXX.","X.E."]`

输出：`6`

解释：
下图是一种可行的方案：



提示：

- `1 <= maze.length, maze[i].length <= 500`

用时 01:01:53　　提交次数 6　　2021/12/25 14:22

提交结果：	通过测试用例：	语言：
解答错误	6/53	python3
执行用时：	消耗内存：	
N/A	N/A	

```
class Solution:
    res = sys.maxsize
    def getDistance(self, maze: List[str]) -> int:
        m, n = len(maze), len(maze[0])
        s_x, s_y = -1, -1
        for i in range(m):
            for j in range(n):
                if maze[i][j] == "S" :
                    s_x, s_y = i, j
        if s_x < 0 or s_x >= m or s_y < 0 or s_y >= n:
            return -1
        visited = [[False for _ in range(n)] for _ in range(m)]
        visited[s_x][s_y] = True
        for i in range(m):
            for j in range(n):
                if maze[i][j] == "X" :
                    visited[i][j] = True
        path = 0
        self.dfs(maze, s_x, s_y, visited, path)
```

```
        return Solution.res if Solution.res != sys.maxsize else -1

    def dfs(self, maze, s_x, s_y, visited, path):
        m, n = len(maze), len(maze[0])
        if 0 <= s_x < m and 0 <= s_y < n and maze[s_x][s_y] == "E":
            Solution.res = min(Solution.res, path)
            return
         for x, y in [(s_x + 1, s_y), (s_x -1, s_y), (s_x, s_y + 1), (s_x,
s_y - 1)]:
            if 0 <= x < m and 0 <= y < n and visited[x][y] == False:
                path += 1
                visited[x][y] = True
                self.dfs(maze, x, y, visited, path)
                path -= 1
                visited[x][y] = False
```

#5 编程题 1000621 分数 10.00/10

岛屿间的最短路径

仅由 `1` （陆地）和 `0` （水）组成的的二维网格信息记录于二维数组 `grid` 中，该网格中 **有且仅有两座岛屿**。请你求出这两座岛屿间的最短路径的长度。

岛屿总是被水包围，并且每座岛屿只能由水平、竖直四个方向上相邻的陆地连接形成。此外，你可以假设该网格的四条边均被水包围。

注意：测试数据保证给定的二维表格中一定恰好存在两座岛屿。

示例 1：

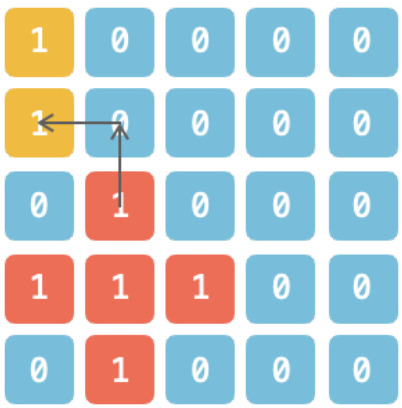
输入：

`grid = [[1,0,0,0,0],[1,0,0,0,0],[0,1,0,0,0],[1,1,1,0,0],[0,1,0,0,0]]`

输出：`2`

解释：

如图所示，是一条可能的最短路径，其长度为 `2` 。



提示：

- `5 <= grid.length, grid[i].length <=500`

- `grid[i][j]` 仅为 `0` 或 `1`

用时 00:09:57 提交次数 1 2021/12/25 13:43

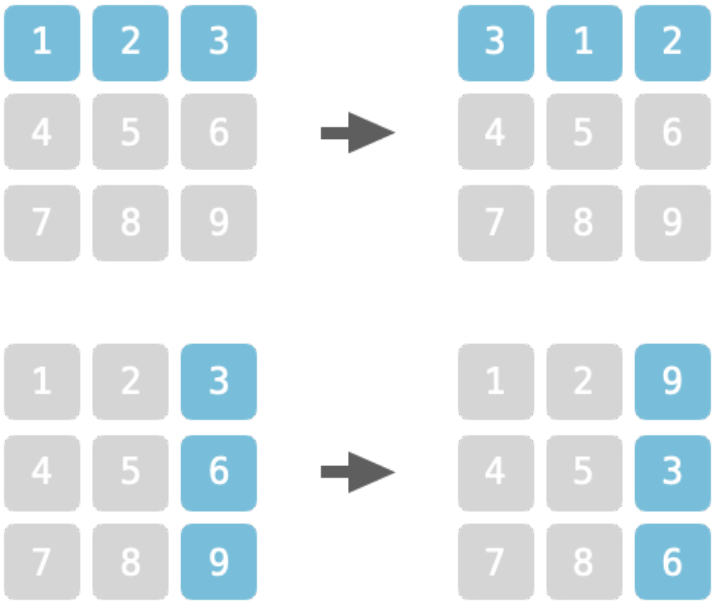
提交结果:	通过测试用例:	语言:
通过	39/39	python3
执行用时:	消耗内存:	
1900 ms	36.8 MB	

```python
class Solution:
    def getDistance(self, grid: List[List[int]]) -> int:
        R, C = len(grid), len(grid[0])

        def neighbors(r, c):
            for nr, nc in ((r-1,c),(r,c-1),(r+1,c),(r,c+1)):
                if 0 <= nr < R and 0 <= nc < C:
                    yield nr, nc

        def get_components():
            done = set()
            components = []
            for r, row in enumerate(grid):
                for c, val in enumerate(row):
                    if val and (r, c) not in done:
                        # Start dfs
                        stack = [(r, c)]
                        seen = {(r, c)}
                        while stack:
                            node = stack.pop()
                            for nei in neighbors(*node):
                                    if grid[nei[0]][nei[1]] and nei not in
seen:
                                        stack.append(nei)
                                        seen.add(nei)
                        done |= seen
                        components.append(seen)
            return components

        source, target = get_components()

        queue = collections.deque([(node, 0) for node in source])
        done = set(source)
        while queue:
            node, d = queue.popleft()
            if node in target: return d
            for nei in neighbors(*node):
                if nei not in done:
                    queue.append((nei, d+1))
                    done.add(nei)
```

#6 编程题 1000620 分数 0.00/10

方阵变换

每次 **变换** 操作可以选择方阵的某一行或者某一列的值进行轮换，其示意图如下：



给定两个仅含有数字 `1~9` 的 `3*3` 方阵 `source` 和 `target`，求从方阵 `source` 到方阵 `target` 所需的最小 **变换** 次数，若方阵 `source` 不能由 **变换** 操作得到方阵 `target`，请返回 `-1`。

示例 1：

> 输入：`source = [[1,2,3],[4,5,6],[7,8,9]], target = [[3,1,9],[4,5,2],[7,8,6]]`
>
> 输出：`2`
>
> 解释：
> 下面是一种可行的方案：
>
> - 对 `source` 方阵的第一行做变换操作，`source` 变为 `[[3,1,2],[4,5,6],[7,8,9]]`
> - 再对 `source` 方阵的第三列做变换操作，`source` 变为 `[[3,1,9],[4,5,2],[7,8,6]]`

提示：

- `source.length = target.length = 3`
- `source[i].length = target[i].length = 3`
- 两个方阵中的元素仅为数字 `1～9`，且方阵中不存在值相同的元素

用时 00:14:36 提交次数 0