算法分析和复杂性理论

第 5 次作业

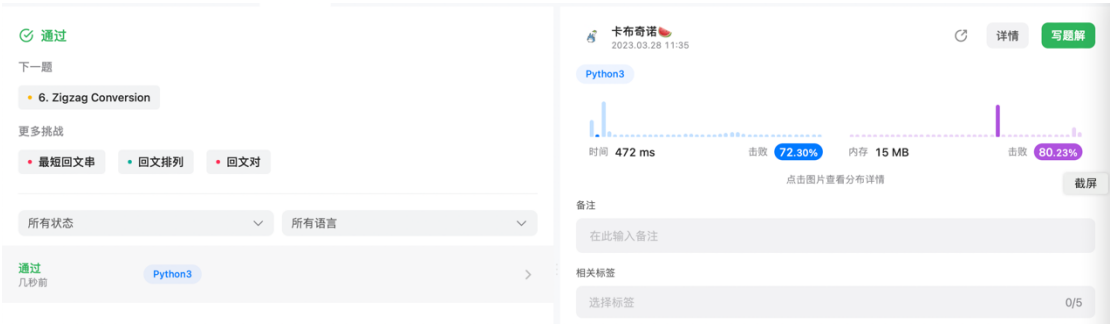
张瀚文 2201212865

1 最长回文子串（005）

题目描述：给你一个字符串 s，找到 s 中最长的回文子串。如果字符串的反序与原始字符串相同，则该字符串称为回文字符串。

解题思路：对于一个子串而言，如果它是回文串，并且长度大于 2，那么将它首尾的两个字母去除之后，它仍然是个回文串。例如对于字符串“ababa”，如果我们已经知道“bab”是回文串，那么“ababa”一定是回文串，这是因为它的首尾两个字母都是“a”。

提交记录：



测试代码：

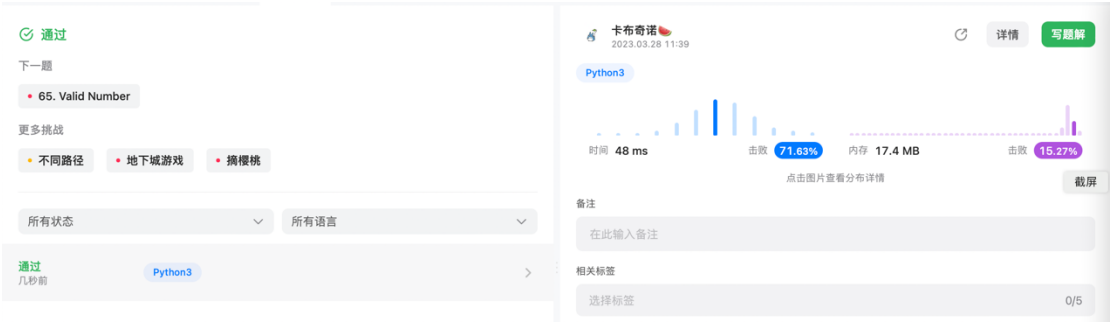
```
class Solution:
    def expandAroundCenter(self, s, left, right):
        while left >= 0 and right < len(s) and s[left] == s[right]:
            left -= 1
            right += 1
        return left + 1, right - 1

    def longestPalindrome(self, s: str) -> str:
        start, end = 0, 0
        for i in range(len(s)):
            left1, right1 = self.expandAroundCenter(s, i, i)
            left2, right2 = self.expandAroundCenter(s, i, i + 1)
            if right1 - left1 > end - start:
                start, end = left1, right1
            if right2 - left2 > end - start:
                start, end = left2, right2
        return s[start: end + 1]
```

2 最小路径和（064）

题目描述：给定一个包含非负整数的 m x n 网格 grid ，请找出一条从左上角到右下角的路径，使得路径上的数字总和为最小。说明：每次只能向下或者向右移动一步。

解题思路：由于路径的方向只能是向下或向右，因此网格的第一行的每个元素只能从左上角元素开始向右移动到达，网格的第一列的每个元素只能从左上角元素开始向下移动到达，此时的路径是唯一的，因此每个元素对应的最小路径和即为对应的路径上的数字总和。对于不在第一行和第一列的元素，可以从其上方相邻元素向下移动一步到达，或者从其左方相邻元素向右移动一步到达，元素对应的最小路径和等于其上方相邻元素与其左方相邻元素两者对应的最小路径和中的最小值加上当前元素的值。由于每个元素对应的最小路径和与其相邻元素对应的最小路径和有关，因此可以使用动态规划求解。

提交记录：



测试代码：

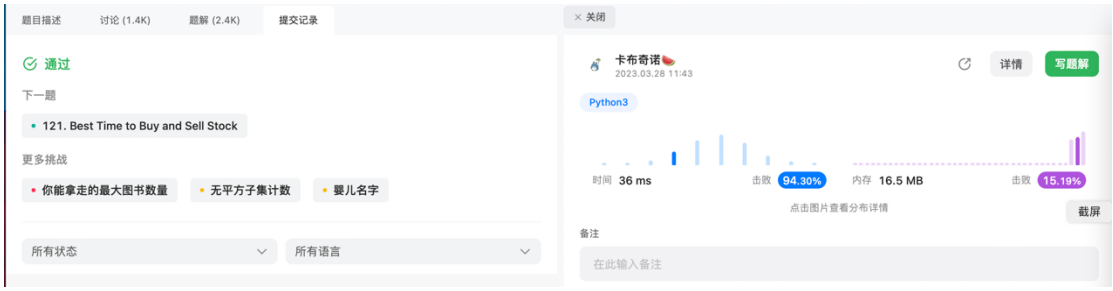
```
class Solution:
    def minPathSum(self, grid: List[List[int]]) -> int:
        if not grid or not grid[0]:
            return 0

        rows, columns = len(grid), len(grid[0])
        dp = [[0] * columns for _ in range(rows)]
        dp[0][0] = grid[0][0]
        for i in range(1, rows):
            dp[i][0] = dp[i - 1][0] + grid[i][0]
        for j in range(1, columns):
            dp[0][j] = dp[0][j - 1] + grid[0][j]
        for i in range(1, rows):
            for j in range(1, columns):
                dp[i][j] = min(dp[i - 1][j], dp[i][j - 1]) + grid[i][j]

        return dp[rows - 1][columns - 1]
```

3　三角形最小路径和（120）

题目描述：给定一个三角形 triangle ，找出自顶向下的最小路径和。每一步只能移动到下一行中相邻的结点上。相邻的结点 在这里指的是 下标 与 上一层结点下标 相同或者等于 上一层结点下标 + 1 的两个结点。也就是说，如果正位于当前行的下标 i ，那么下一步可以移动到下一行的下标 i 或 i + 1 。

解题思路：我们可以将任一点到底边的最小路径和，转化成了与该点相邻两点到底边的最小路径和中的较小值，再加上该点本身的值。

提交记录：



测试代码：

```
class Solution:
    def minimumTotal(self, triangle: List[List[int]]) -> int:
        n = len(triangle)
        f = [[0] * n for _ in range(n)]
        f[0][0] = triangle[0][0]

        for i in range(1, n):
            f[i][0] = f[i - 1][0] + triangle[i][0]
            for j in range(1, i):
                f[i][j] = min(f[i - 1][j - 1], f[i - 1][j]) + triangle[i][j]
            f[i][i] = f[i - 1][i - 1] + triangle[i][i]

        return min(f[n - 1])
```