0005. 最长回文子串

ITCharge 大约 2 分钟

- 标签：字符串、动态规划
- 难度：中等

题目链接

- 0005. 最长回文子串 - 力扣

题目大意

描述：给定一个字符串 s。

要求：找到 s 中最长的回文子串。

说明：

- **回文串**：如果字符串的反序与原始 串相同，则该字符串称为回文字符串。
- $1 \leq s.length \leq 1000$。
- s 仅由数字和英文字母组成。

示例：

- 示例 1：

```py
输入：s = "babad"
输出："bab"
解释："aba" 同样是符合题意的答案。
```

- 示例 2：

```py
输入：s = "cbbd"
输出："bb"
```

解题思路

思路 1：动态规划

1. 划分阶段

按照区间长度进行阶段划分。

2. 定义状态

定义状态 $dp[i][j]$ 表示为：字符串 s 在区间 $[i, j]$ 范围内是否是一个回文串。

3. 状态转移方程

- 当子串只有 1 位或 2 位的时候，如果 $s[i] == s[j]$，该子串为回文子串，即：`dp[i][j] = (s[i] == s[j])`。
- 如果子串大于 2 位，则如果 $s[i+1:j-1]$ 是回文串，且 $s[i] == s[j]$，则 $s[i...j]$ 也是回文串，即：`dp[i][j] = (s[i] == s[j]) and dp[i + 1][j - 1]`。

4. 初始条件

- 初始状态下，默认字符串 s 的所有子串都不是回文串。

5. 最终结果

根据我们之前定义的状态，$dp[i][j]$ 表示为：字符串 s 在区间 $[i, j]$ 范围内是否是一个回文串。当判断完 $s[i:j]$ 是否为回文串时，同时判断并更新最长回文子串的起始位置 max_start 和最大长度 max_len。则最终结果为 $s[max\_start, max\_start + max\_len]$。

思路 1：代码

```py
class Solution:
    def longestPalindrome(self, s: str) -> str:
        n = len(s)
        if n <= 1:
            return s

        dp = [[False for _ in range(n)] for _ in range(n)]
        max_start = 0
        max_len = 1

        for j in range(1, n):
            for i in range(j):
                if s[i] == s[j]:
                    if j - i <= 2:
                        dp[i][j] = True
                    else:
                        dp[i][j] = dp[i + 1][j - 1]
                if dp[i][j] and      i + 1) > max_len:
                    max_len = j - i + 1
                    max_start = i
        return s[max_start: max_start + max_len]
```

思路 1：复杂度分析

- **时间复杂度**：$O(n^2)$，其中 n 是字符串的长度。
- **空间复杂度**：$O(n^2)$。