0727. 最小窗口子序列

ITCharge 大约 3 分钟

- 标签：字符串、动态规划、滑动窗口
- 难度：困难

题目链接

- 0727. 最小窗口子序列 - 力扣

题目大意

给定字符串 `s1` 和 `s2`。

要求：找出 `s1` 中最短的（连续）子串 `w`，使得 `s2` 是 `w` 的子序列。如果 `s1` 中没有窗口可以包含 `s2` 中的所有字符，返回空字符串 `""`。如果有不止一个最短长度的窗口，返回开始位置最靠左的那个。

解题思路

这道题跟「76. 最小覆盖子串」有点类似。但这道题中字符的相对顺序需要保持一致。求解的思路如下：

- 向右扩大窗口，匹配字符，直到匹配完 `s2` 的最后一个字符。
- 当满足条件时，缩小窗口，并更新最小窗口的起始位置和最短长度。
- 缩小窗口到不满足条件为止。

这道题的难点在于第二步中如何缩小窗口。当匹配到一个子序列时，可以采用逆向匹配的方式，从 `s2` 的最后一位字符匹配到 `s2` 的第一位字符。找到符合要求的最大下标，即是窗口的左边界。

整个算法的解题步骤如下：

- 使用两个指针 `left`、`right` 代表窗口的边界，一开始都指向 `0`。`min_len` 用来记录最小子序列的长度。`i`、`j` 作为索引，用于遍历字符串 `s1` 和 `s2`，一开始都为 `0`。
- 遍历字符串 `s1` 的每一个字符，如果 `s1[i] == s2[j]`，则说明 `s2` 中第 `j` 个字符匹配了，向右移动 `j`，即 `j += 1`，然后继续匹配。

- 如果 `j == len(s2)` ，则说明 `s2` 中所有字符都匹配了。
  - 此时确定了窗口的右边界 `right = i` ，并令 `j` 指向 `s2` 最后一个字符位置。
  - 从右至左逆向匹配字符串，找到窗口的左边界。
  - 判断当前窗口长度和窗口的最短长度，并更新最小窗口的起始位置和最短长度。
  - 令 `j = 0` ，重新继续匹配 `s2` 。
- 向右移动 `i` ，继续匹配。
- 遍历完输出窗口的最短长度（需要判断是否有解）。

代码

```py
class Solution:
    def minWindow(self, s1: str, s2: str) -> str:
        i, j = 0, 0
        min_len = float('inf')
        left, right = 0, 0
        while i < len(s1):
            if s1[i] == s2[j]:
                j += 1
            # 完成了匹配
            if j == len(s2):
                right = i
                j -= 1
                while j >= 0:
                    if s1[i] == s2[j]:
                        j -= 1
                    i -= 1
                i += 1
                if right - i + 1 < min_len:
                    left = i
                    min_len = right - left + 1
                j = 0
            i += 1
        if min_len != float('inf'):
            return s1[left: left + min_len]
        return ""
```

参考资料

- 【题解】[c++ 简单好理解的 滑动窗口解法 和 动态规划解法 - 最小窗口子序列 - 力扣]

- 【题解】727. 最小窗口子序列 C++ 滑动窗口 - 最小窗口子序列 - 力扣