0076. 最小覆盖子串

ITCharge 大约 1 分钟

- 标签：哈希表、字符串、滑动窗口
- 难度：困难

题目链接

- 0076. 最小覆盖子串 - 力扣

题目大意

描述：给定一个字符串 `s` 、一个字符串 `t` 。

要求：返回 `s` 中涵盖 `t` 所有字符的最小子串。如果 `s` 中不存在涵盖 `t` 所有字符的子串，则返回空字符串 `""` 。

说明：

- $1 \leq s.length, t.length \leq 10^5$。
- `s` 和 `t` 由英文字母组成。

示例：

- 示例 1：

```py
输入：s = "ADOBECODEBANC", t = "ABC"
输出："BANC"
```

- 示例 2：

```py
输入：s = "a", t = "a"
输出："a"
```

解题思路

思路 1：滑动窗口

1. `left` 、 `right` 表示窗口的边界，一开始都位于下标 `0` 处。 `need` 用于记录短字符串需要的字符数。 `window` 记录当前窗口内的字符数。
2. 将 `right` 右移，直到出现了 `t` 中全部字符，开始右移 `left` ，减少滑动窗口的大小，并记录下最小覆盖子串的长度和起始位置。
3. 最后输出结果。

思路 1：代码

py

```
import collections

class Solution:
    def minWindow(self, s: str, t: str) -> str:
        need = collections.defau    ct(int)
        window = collections.defaultdict(int)
        for ch in t:
            need[ch] += 1

        left, right = 0, 0
        valid = 0
        start = 0
        size = len(s) + 1

        while right < len(s):
            insert_ch = s[right]
            right += 1

            if insert_ch in need:
                window[insert_ch] += 1
                if window[insert_ch] == need[insert_ch]:
                    valid += 1

            while valid == len(need):
                if right - left < size:
```

```
                start = left
                size = right - left
            remove_ch = s[left]
            left += 1
            if remove_ch in need:
                if window[remove_ch] == need[remove_ch]:
                    valid -= 1
                window[remove_ch] -= 1
        if size == len(s) + 1:
            return ''
        return s[start:start+size]
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$。其中 n 是字符串 s 的长度。
- **空间复杂度**：$O(|\sum|)$。$|\sum|$ 是 s 和 t 的字符集大小。