76. Minimum Window Substring

Created	@February 4, 2022 8:10 AM
Difficulty	Hard
LC Url	https://leetcode.com/problems/minimum-window-substring/
Importance	****
Tag	Sliding Window String
Video	https://www.youtube.com/watch?v=qvou9R_7m10&list=PL2rWx9cCzU84DLLKCkx-9kLQZzieHo3r_&index=5

Given two strings `s` and `t` of lengths `m` and `n` respectively, return *the **minimum window substring** of* `s` *such that every character in* `t` *(**including duplicates**) is included in the window. If there is no such substring, return the empty string* `""`.

The testcases will be generated such that the answer is **unique**.

A **substring** is a contiguous sequence of characters within the string.

Example 1:

```
Input: s = "ADOBECODEBANC", t = "ABC"
Output: "BANC"
Explanation: The minimum window substring "BANC" includes 'A', 'B', and 'C' from string t.
```

Example 2:

```
Input: s = "a", t = "a"
Output: "a"
Explanation: The entire string s is the minimum window.
```

Example 3:

```
Input: s = "a", t = "aa"
Output: ""
Explanation: Both 'a's from t must be included in the window.
Since the largest window of s only has one 'a', return empty string.
```

Constraints:

- `m == s.length`
- `n == t.length`
- `1 <= m, n <= 10` `5`
- `s` and `t` consist of uppercase and lowercase English letters.

Follow up:

Could you find an algorithm that runs in

```
O(m + n)
```

time?

Solution

```
class Solution:
    def minWindow(self, s: str, t: str) -> str:
        if not t or not s:
            return ''

        dict_t = Counter(t)
        required = len(dict_t)

        filtered_s = []
        for i, char in enumerate(s):
            if char in dict_t:
                filtered_s.append((i, char))

        left, right = 0, 0
        formed = 0
        window_counts = {}
        ans = [float('inf'), None, None]

        while right < len(filtered_s):
            character = filtered_s[right][1]
            window_counts[character] = window_counts.get(character, 0) + 1

            if window_counts[character] == dict_t[character]:
                formed += 1

            while left <= right and formed == required:
                character = filtered_s[left][1]

                start = filtered_s[left][0]
                end = filtered_s[right][0]
                if end - start + 1 < ans[0]:
                    ans = (end - start + 1, start, end)

                window_counts[character] -= 1
                if window_counts[character] < dict_t[character]:
                    formed -= 1
                left += 1

            right += 1

        return '' if ans[0] == float('inf') else s[ans[1]: ans[2] + 1]
```