76. Minimum Window Substring

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<sup>5</sup>`
- `s` and `t` consist of uppercase and lowercase English letters.

Follow up: Could you find an algorithm that runs in `O(m + n)` time?

```
def minWindow(self, s: str, t: str) -> str:
        if len(s)<len(t):
            return ""
        ans = ""
        mapT = collections.Counter(t)
        mapS = {}
        allowedLength = len(t)
        currLength = 0
        i = -1
        j = -1
        while True:
            f1 = False
            f2 = False
            while i<len(s)-1 and currLength<allowedLength:
                i = i+1
                ch = s[i]
                mapS[ch] = mapS.get(ch,0)+1
                if mapS[ch]<=mapT.get(ch,0):
                    currLength = currLength+1
                f1 = True

            while j<i and currLength==allowedLength:
                pAns = s[j+1:i+1]
                if len(ans)==0 or len(pAns)<len(ans):
                    ans = pAns
                j = j+1
                ch = s[j]
                if mapS[ch]==1:
                    mapS[ch]=0
                else:
                    mapS[ch] = mapS[ch]-1

                if mapS[ch]<mapT.get(ch,0):
                    currLength = currLength-1
                f2 = True

            if f1 == False and f2 == False:
                break
        return ans
```