Minimum Window Substring Documentation

Objective

Given two strings s and t of lengths m and n respectively, the goal is to find 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 "".

Constraints

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

FollowUp

- Find an algorithm that runs in O(m + n) time.

Examples

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 the empty string.

Approach

The solution uses the sliding window technique along with hashmaps to keep track of character counts.

1. **Initialization:**

- Create a dictionary dict_t to count all characters in t.
- Initialize the required variable to the number of unique characters in t.
- Use two pointers, left and right, to represent the current window.
- Create a dictionary window_counts to keep track of the counts of characters in the current window.
- Initialize formed to count how many unique characters in the current window match the required count in t.
- Initialize ans to store the length of the smallest window found, and its left and right pointers.

2. **Expand the Window:**

- Move the right pointer to expand the window by adding one character from s to the window_counts.
- If the character added is part of t and its count in the window matches its count in t, increment formed.

3. **Contract the Window:**

- While the window is desirable (i.e., all characters in t are in the window with correct counts):
- Save the smallest window by updating ans.
- Move the left pointer to contract the window by removing characters from the window.
- If a character removed is part of t and its count falls below the required count, decrement formed.

4. **Return the Result:**

- If ans was updated, return the substring of s using the pointers stored in ans.
- If ans was not updated, return the empty string.

Time Complexity

- The algorithm runs in O(m + n) time because each character in s and t is processed a constant number of times.

Space Complexity

- The space complexity is O(m + n) due to the storage of the character counts in the dictionaries.