LeetCode 392: Is Subsequence (Two Pointers Approach)

■ Problem Statement:

Given two strings s and t, return true if s is a subsequence of t, or false otherwise. A subsequence of a string is a new string formed from the original string by deleting some (or no) characters without changing the order of the remaining characters.

■ Approach: Two Pointers

We use two pointers — one for s and one for t. Move both pointers when characters match, and move only the t pointer when they don’t. If we reach the end of s, it means all characters in s were found in order within t.

■ Example:

Input: s = 'abc', t = 'ahbgdc'
Output: true
Explanation: We can form 'abc' by deleting 'h', 'g', and 'd' from 'ahbgdc'.

■ Python Solution:

```python
def isSubsequence(s: str, t: str) -> bool:
    i = j = 0
    while i < len(s) and j < len(t):
        if s[i] == t[j]:
            i += 1
        j += 1
    return i == len(s)

# Example
print(isSubsequence("abc", "ahbgdc"))  # Output: True
```

■ C++ Solution:

```cpp
#include <iostream>
using namespace std;

bool isSubsequence(string s, string t) {
    int i = 0, j = 0;
    while (i < s.size() && j < t.size()) {
        if (s[i] == t[j]) i++;
        j++;
    }
    return i == s.size();
}

int main() {
    string s = "abc", t = "ahbgdc";
    cout << (isSubsequence(s, t) ? "true" : "false");
    return 0;
}
```

■ Java Solution:

```java
public class Main {
    public static boolean isSubsequence(String s, String t) {
        int i = 0, j = 0;
        while (i < s.length() && j < t.length()) {
            if (s.charAt(i) == t.charAt(j)) {
                i++;
            }
            j++;
        }
        return i == s.length();
    }

    public static void main(String[] args) {
        String s = "abc", t = "ahbgdc";
        System.out.println(isSubsequence(s, t)); // true
    }
}
```