Subsequence

Approach 1: Two Pointers (Simple)

- Iterate through t while checking characters of s in order.
- Increment pointer on s when a matching character is found.

Algorithm:

```
1. Initialize i=0 (for s) and j=0 (for t)
2. While j<t.length():
       · If s[i]==t[j], increment i.
       · Increment j.
3. If i==s.length(), return true, else false.
```

Complexity:

- Time: O(|t|) – single pass through t
- Space: O(1)

Follow-Up: Many Queries ($s_1, s_2, \ldots, s_k$):

If we have many s queries and a fixed t:

- Preprocess t:
  - Store positions of each character (a-z) in arrays.
- For each s query:
  - Use binary search to find next valid position in t for each character of s.
- Complexity per query: O(|s| log|t|), preprocessing O(|t|).