Problem of the Week – Autocomplete System

Company: Twitter
Difficulty: Medium
Topic: Strings, Trie, Hashing

📌 Scenario

Autocomplete is a widely used feature in search engines and applications like Twitter, Google, and IDEs. When a user starts typing, the system suggests possible completions based on known query strings.

Your task is to **implement an autocomplete system** that, given a query prefix and a set of strings, returns all the words that begin with that prefix.

📌 Problem Statement

You are given:

- A **query string** `s`.
- A **set of possible query strings** `dict[]`.

Return all strings in `dict` that have `s` as a prefix.

💡 **Hint:** Preprocessing the dictionary into a **Trie (prefix tree)** makes lookups much faster compared to checking every word linearly.

🔷 Input Format

- First line: Integer `N`, number of words in the dictionary.
- Second line: N space-separated strings (the dictionary).
- Third line: A string `s` (the query prefix).

🔷 Output Format

- List of strings from the dictionary that start with prefix `s`.

🔷 Constraints

- $1 \leq N \leq 10^5$
- Each word length ≤ 50
- Query string length ≤ 50

🔷 Sample Input 0

```
3
dog deer deal
de
```

🔷 Sample Output 0

```
deer deal
```

🔑 Approaches

1. **Brute Force Search**
   - Iterate through every word in the dictionary.
   - Check if it starts with prefix `s`.
   - Time: **O(N * L)** (N = words, L = length of prefix).
2. **Trie (Prefix Tree) Approach – Efficient**
   - Preprocess dictionary into a Trie.
   - Traverse Trie using prefix `s`.
   - DFS/BFS from that node to collect words.
   - Lookup time: **O(L + K)** (L = prefix length, K = number of results).

🔗 Practice Links

- LeetCode – Implement Trie (Prefix Tree)
- GeeksforGeeks – Autocomplete feature using Trie