Problem 208: Implement Trie (Prefix Tree)

Problem Information

Difficulty: Medium
Acceptance Rate: 68.80%
Paid Only: No
Tags: Hash Table, String, Design, Trie

Problem Description

A [**trie**](https://en.wikipedia.org/wiki/Trie) (pronounced as "try") or **prefix tree** is a tree data structure used to efficiently store and retrieve keys in a dataset of strings. There are various applications of this data structure, such as autocomplete and spellchecker.

Implement the Trie class:

* `Trie()` Initializes the trie object. * `void insert(String word)` Inserts the string `word` into the trie. * `boolean search(String word)` Returns `true` if the string `word` is in the trie (i.e., was inserted before), and `false` otherwise. * `boolean startsWith(String prefix)` Returns `true` if there is a previously inserted string `word` that has the prefix `prefix`, and `false` otherwise.

Example 1:

Input ["Trie", "insert", "search", "search", "startsWith", "insert", "search"] [[], ["apple"], ["apple"], ["app"], ["app"], ["app"], ["app"]] **Output** [null, null, true, false, true, null, true] **Explanation** Trie trie = new Trie(); trie.insert("apple"); trie.search("apple"); // return True trie.search("app"); // return False trie.startsWith("app"); // return True trie.insert("app"); trie.search("app"); // return True

Constraints:

* `1 <= word.length, prefix.length <= 2000` * `word` and `prefix` consist only of lowercase English letters. * At most `3 * 104` calls **in total** will be made to `insert`, `search`, and `startsWith`.

Code Snippets

C++:

```
class Trie {
public:
Trie() {

}

void insert(string word) {

}

bool search(string word) {

}

bool startsWith(string prefix) {

}
};

/**
* Your Trie object will be instantiated and called as such:
* Trie* obj = new Trie();
* obj->insert(word);
* bool param_2 = obj->search(word);
* bool param_3 = obj->startsWith(prefix);
*/
```

Java:

```
class Trie {

public Trie() {

}

public void insert(String word) {

}
```

```
public boolean search(String word) {

}

public boolean startsWith(String prefix) {

}
}

/**
* Your Trie object will be instantiated and called as such:
* Trie obj = new Trie();
* obj.insert(word);
* boolean param_2 = obj.search(word);
* boolean param_3 = obj.startsWith(prefix);
*/
```

Python3:

```
class Trie:

def __init__(self):


def insert(self, word: str) -> None:


def search(self, word: str) -> bool:


def startsWith(self, prefix: str) -> bool:



# Your Trie object will be instantiated and called as such:
# obj = Trie()
# obj.insert(word)
# param_2 = obj.search(word)
# param_3 = obj.startsWith(prefix)
```