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.