

1804. Implement Trie II (Prefix Tree)

Medium 104 7 Add to List Share

A 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.
- `int countWordsEqualTo(String word)` Returns the number of instances of the string `word` in the trie.
- `int countWordsStartingWith(String prefix)` Returns the number of strings in the trie that have the string `prefix` as a prefix.
- `void erase(String word)` Erases the string `word` from the trie.

Example 1:

Input
["Trie", "insert", "insert", "countWordsEqualTo", "countWordsStartingWith", "erase", "countWordsEqualTo", "countWordsStartingWith", "erase", "countWordsStartingWith"]
[[], ["apple"], ["apple"], ["apple"], ["app"], ["apple"], ["apple"], ["app"], ["apple"], ["app"]]
Output
[null, null, null, 2, 2, null, 1, 1, null, 0]

Explanation
Trie trie = new Trie();
trie.insert("apple"); // Inserts "apple".
trie.insert("apple"); // Inserts another "apple".
trie.countWordsEqualTo("apple"); // There are two instances of "apple" so return 2.
trie.countWordsStartingWith("app"); // "app" is a prefix of "apple" so return 2.
trie.erase("apple"); // Erases one "apple".
trie.countWordsEqualTo("apple"); // Now there is only one instance of "apple" so return 1.
trie.countWordsStartingWith("app"); // return 1
trie.erase("apple"); // Erases "apple". Now the trie is empty.
trie.countWordsStartingWith("app"); // return 0

Constraints:

- $1 \leq \text{word.length}, \text{prefix.length} \leq 2000$
- `word` and `prefix` consist only of lowercase English letters.
- At most 3×10^4 calls in total will be made to `insert`, `countWordsEqualTo`, `countWordsStartingWith`, and `erase`.
- It is guaranteed that for any function call to `erase`, the string `word` will exist in the trie.

Accepted 4,524 Submissions 7,760

Seen this question in a real interview before? Yes No

Companies 1

0 - 6 months 6 months - 1 year 1 year - 2 years

Apple 2

Related Topics

Hash Table String Design Trie

Similar Questions

Implement Trie (Prefix Tree) Medium

Hide Hint 1

Try to solve the first version first and reuse your code.

Hide Hint 2

To implement the delete function, you should delete the trie nodes of the word if they are not shared with other words.

Hide Hint 3

You should keep for each trie node a counter of how many words share this node.

```
1 *
2 *
3 *
4 *
5 *
6 *
7 *
8 *
9 *
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *
24 *
25 *
26 *
27 *
28 *
29 *
30 *
31 *
```

```
class Trie {
    public Trie() {
    }
    public void insert(String word) {
    }
    public int countWordsEqualTo(String word) {
    }
    public int countWordsStartingWith(String prefix) {
    }
    public void erase(String word) {
    }
}
/**
 * Your Trie object will be instantiated and called as such:
 * Trie obj = new Trie();
 * obj.insert(word);
 * int param_2 = obj.countWordsEqualTo(word);
 * int param_3 = obj.countWordsStartingWith(prefix);
 * obj.erase(word);
 */
```