Problem 211: Design Add and Search Words Data Structure

Problem Information

Difficulty: Medium
Acceptance Rate: 47.80%
Paid Only: No
Tags: String, Depth-First Search, Design, Trie

Problem Description

Design a data structure that supports adding new words and finding if a string matches any previously added string.

Implement the `WordDictionary` class:

* `WordDictionary()` Initializes the object. * `void addWord(word)` Adds `word` to the data structure, it can be matched later. * `bool search(word)` Returns `true` if there is any string in the data structure that matches `word` or `false` otherwise. `word` may contain dots `'.'` where dots can be matched with any letter.

Example:

Input
["WordDictionary","addWord","addWord","addWord","search","search","search","search"] [[],["bad"],["dad"],["mad"],["pad"],["bad"],[".ad"],["b.."]] **Output** [null,null,null,null,false,true,true,true] **Explanation** WordDictionary wordDictionary = new WordDictionary(); wordDictionary.addWord("bad"); wordDictionary.addWord("dad"); wordDictionary.addWord("mad"); wordDictionary.search("pad"); // return False wordDictionary.search("bad"); // return True wordDictionary.search(".ad"); // return True wordDictionary.search("b.."); // return True

Constraints:

* `1 <= word.length <= 25` * `word` in `addWord` consists of lowercase English letters. * `word` in `search` consist of `'.'` or lowercase English letters. * There will be at most `2` dots in

`word` for `search` queries. * At most `104` calls will be made to `addWord` and `search`.

Code Snippets

C++:

```
class WordDictionary {
public:
WordDictionary() {

}

void addWord(string word) {

}

bool search(string word) {

}
};

/**
* Your WordDictionary object will be instantiated and called as such:
* WordDictionary* obj = new WordDictionary();
* obj->addWord(word);
* bool param_2 = obj->search(word);
*/
```

Java:

```
class WordDictionary {

public WordDictionary() {

}

public void addWord(String word) {

}

public boolean search(String word) {
```

```
}
}

/**
* Your WordDictionary object will be instantiated and called as such:
* WordDictionary obj = new WordDictionary();
* obj.addWord(word);
* boolean param_2 = obj.search(word);
*/
```

Python3:

```
class WordDictionary:

def __init__(self):


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


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



# Your WordDictionary object will be instantiated and called as such:
# obj = WordDictionary()
# obj.addWord(word)
# param_2 = obj.search(word)
```