

Problem 966: Vowel Spellchecker

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given a

wordlist

, we want to implement a spellchecker that converts a query word into a correct word.

For a given

query

word, the spell checker handles two categories of spelling mistakes:

Capitalization: If the query matches a word in the wordlist (

case-insensitive

), then the query word is returned with the same case as the case in the wordlist.

Example:

```
wordlist = ["yellow"]
```

,

```
query = "YellOw"
```

correct = "yellow"

Example:

```
wordlist = ["Yellow"]
```

,

query = "yellow"

:

correct = "Yellow"

Example:

```
wordlist = ["yellow"]
```

,

query = "yellow"

:

correct = "yellow"

Vowel Errors: If after replacing the vowels

('a', 'e', 'i', 'o', 'u')

of the query word with any vowel individually, it matches a word in the wordlist (

case-insensitive

), then the query word is returned with the same case as the match in the wordlist.

Example:

```
wordlist = ["YellOw"]
```

```
,
```

```
query = "yellow"
```

```
:
```

```
correct = "YellOw"
```

Example:

```
wordlist = ["YellOw"]
```

```
,
```

```
query = "yeellow"
```

```
:
```

```
correct = ""
```

(no match)

Example:

```
wordlist = ["YellOw"]
```

```
,
```

```
query = "yllw"
```

```
:
```

```
correct = ""
```

(no match)

In addition, the spell checker operates under the following precedence rules:

When the query exactly matches a word in the wordlist (

case-sensitive

), you should return the same word back.

When the query matches a word up to capitalization, you should return the first such match in the wordlist.

When the query matches a word up to vowel errors, you should return the first such match in the wordlist.

If the query has no matches in the wordlist, you should return the empty string.

Given some

queries

, return a list of words

answer

, where

answer[i]

is the correct word for

query = queries[i]

Example 1:

Input:

```
wordlist = ["KiTe", "kite", "hare", "Hare"], queries =  
["kite", "Kite", "KiTe", "Hare", "HARE", "Hear", "hear", "keti", "keet", "keto"]
```

Output:

```
["kite", "KiTe", "KiTe", "Hare", "hare", "", "", "KiTe", "", "KiTe"]
```

Example 2:

Input:

```
wordlist = ["yellow"], queries = ["YellOw"]
```

Output:

```
["yellow"]
```

Constraints:

```
1 <= wordlist.length, queries.length <= 5000
```

```
1 <= wordlist[i].length, queries[i].length <= 7
```

```
wordlist[i]
```

and

```
queries[i]
```

consist only of only English letters.

Code Snippets

C++:

```
class Solution {  
public:  
    vector<string> spellchecker(vector<string>& wordlist, vector<string>&
```

```
    queries) {  
  
}  
};
```

Java:

```
class Solution {  
public String[] spellchecker(String[] wordlist, String[] queries) {  
  
}  
}
```

Python3:

```
class Solution:  
def spellchecker(self, wordlist: List[str], queries: List[str]) -> List[str]:
```

Python:

```
class Solution(object):  
def spellchecker(self, wordlist, queries):  
    """  
    :type wordlist: List[str]  
    :type queries: List[str]  
    :rtype: List[str]  
    """
```

JavaScript:

```
/**  
 * @param {string[]} wordlist  
 * @param {string[]} queries  
 * @return {string[]}  
 */  
var spellchecker = function(wordlist, queries) {  
  
};
```

TypeScript:

```
function spellchecker(wordlist: string[], queries: string[]): string[] {
```

```
};
```

C#:

```
public class Solution {  
    public string[] Spellchecker(string[] wordlist, string[] queries) {  
  
    }  
}
```

C:

```
/**  
 * Note: The returned array must be malloced, assume caller calls free().  
 */  
char** spellchecker(char** wordlist, int wordlistSize, char** queries, int  
queriesSize, int* returnSize) {  
  
}
```

Go:

```
func spellchecker(wordlist []string, queries []string) []string {  
  
}
```

Kotlin:

```
class Solution {  
    fun spellchecker(wordlist: Array<String>, queries: Array<String>):  
        Array<String> {  
  
    }  
}
```

Swift:

```
class Solution {  
    func spellchecker(_ wordlist: [String], _ queries: [String]) -> [String] {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn spellchecker(wordlist: Vec<String>, queries: Vec<String>) ->  
        Vec<String> {  
  
    }  
}
```

Ruby:

```
# @param {String[]} wordlist  
# @param {String[]} queries  
# @return {String[]}  
def spellchecker(wordlist, queries)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param String[] $wordlist  
     * @param String[] $queries  
     * @return String[]  
     */  
    function spellchecker($wordlist, $queries) {  
  
    }  
}
```

Dart:

```
class Solution {  
    List<String> spellchecker(List<String> wordlist, List<String> queries) {  
  
    }  
}
```

Scala:

```

object Solution {
def spellchecker(wordlist: Array[String], queries: Array[String]): Array[String] = {
    }

}
}

```

Elixir:

```

defmodule Solution do
@spec spellchecker(wordlist :: [String.t], queries :: [String.t]) :: [String.t]
def spellchecker(wordlist, queries) do
end
end

```

Erlang:

```

-spec spellchecker(Wordlist :: [unicode:unicode_binary()], Queries :: [unicode:unicode_binary()]) -> [unicode:unicode_binary()].
spellchecker(Wordlist, Queries) ->
.
.
```

Racket:

```

(define/contract (spellchecker wordlist queries)
  (-> (listof string?) (listof string?) (listof string?)))
)
```

Solutions

C++ Solution:

```

/*
 * Problem: Vowel Spellchecker
 * Difficulty: Medium
 * Tags: array, string, hash
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)

```

```

* Space Complexity: O(n) for hash map
*/
class Solution {
public:
vector<string> spellchecker(vector<string>& wordlist, vector<string>&
queries) {

}
};


```

Java Solution:

```

/**
 * Problem: Vowel Spellchecker
 * Difficulty: Medium
 * Tags: array, string, hash
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
*/
class Solution {
public String[] spellchecker(String[] wordlist, String[] queries) {

}
}


```

Python3 Solution:

```

"""
Problem: Vowel Spellchecker
Difficulty: Medium
Tags: array, string, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""


```

```
class Solution:
    def spellchecker(self, wordlist: List[str], queries: List[str]) -> List[str]:
        # TODO: Implement optimized solution
        pass
```

Python Solution:

```
class Solution(object):
    def spellchecker(self, wordlist, queries):
        """
        :type wordlist: List[str]
        :type queries: List[str]
        :rtype: List[str]
        """
```

JavaScript Solution:

```
/**
 * Problem: Vowel Spellchecker
 * Difficulty: Medium
 * Tags: array, string, hash
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

/**
 * @param {string[]} wordlist
 * @param {string[]} queries
 * @return {string[]}
 */
var spellchecker = function(wordlist, queries) {
}
```

TypeScript Solution:

```
/**
 * Problem: Vowel Spellchecker
 * Difficulty: Medium
```

```

* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
function spellchecker(wordlist: string[], queries: string[]): string[] {
}

```

C# Solution:

```

/*
* Problem: Vowel Spellchecker
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
public class Solution {
    public string[] Spellchecker(string[] wordlist, string[] queries) {
        }
    }

```

C Solution:

```

/*
* Problem: Vowel Spellchecker
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

```

```
/**  
 * Note: The returned array must be malloced, assume caller calls free().  
 */  
char** spellchecker(char** wordlist, int wordlistSize, char** queries, int  
queriesSize, int* returnSize) {  
  
}
```

Go Solution:

```
// Problem: Vowel Spellchecker  
// Difficulty: Medium  
// Tags: array, string, hash  
//  
// Approach: Use two pointers or sliding window technique  
// Time Complexity: O(n) or O(n log n)  
// Space Complexity: O(n) for hash map  
  
func spellchecker(wordlist []string, queries []string) []string {  
  
}
```

Kotlin Solution:

```
class Solution {  
    fun spellchecker(wordlist: Array<String>, queries: Array<String>):  
        Array<String> {  
  
    }  
}
```

Swift Solution:

```
class Solution {  
    func spellchecker(_ wordlist: [String], _ queries: [String]) -> [String] {  
  
    }  
}
```

Rust Solution:

```

// Problem: Vowel Spellchecker
// Difficulty: Medium
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
    pub fn spellchecker(wordlist: Vec<String>, queries: Vec<String>) -> Vec<String> {
        }

    }
}

```

Ruby Solution:

```

# @param {String[]} wordlist
# @param {String[]} queries
# @return {String[]}
def spellchecker(wordlist, queries)

end

```

PHP Solution:

```

class Solution {

    /**
     * @param String[] $wordlist
     * @param String[] $queries
     * @return String[]
     */
    function spellchecker($wordlist, $queries) {

    }
}

```

Dart Solution:

```

class Solution {
    List<String> spellchecker(List<String> wordlist, List<String> queries) {

```

```
}
```

```
}
```

Scala Solution:

```
object Solution {  
    def spellchecker(wordlist: Array[String], queries: Array[String]):  
        Array[String] = {  
  
    }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec spellchecker(wordlist :: [String.t], queries :: [String.t]) ::  
    [String.t]  
  def spellchecker(wordlist, queries) do  
  
  end  
end
```

Erlang Solution:

```
-spec spellchecker(Wordlist :: [unicode:unicode_binary()]), Queries ::  
  [unicode:unicode_binary()]) -> [unicode:unicode_binary()].  
spellchecker(Wordlist, Queries) ->  
.
```

Racket Solution:

```
(define/contract (spellchecker wordlist queries)  
  (-> (listof string?) (listof string?) (listof string?))  
)
```