Problem 524: Longest Word in Dictionary through Deleting

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Given a string

s

and a string array

dictionary

, return

the longest string in the dictionary that can be formed by deleting some of the given string characters

. If there is more than one possible result, return the longest word with the smallest lexicographical order. If there is no possible result, return the empty string.

Example 1:

Input:

s = "abpcplea", dictionary = ["ale","apple","monkey","plea"]

Output:

"apple"

Example 2:

Input:

s = "abpcplea", dictionary = ["a","b","c"]

Output:

"a"

Constraints:

1 <= s.length <= 1000

1 <= dictionary.length <= 1000

1 <= dictionary[i].length <= 1000

s

and

dictionary[i]

consist of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
string findLongestWord(string s, vector<string>& dictionary) {

}
};
```

Java:

```
class Solution {
public String findLongestWord(String s, List<String> dictionary) {

}
}
```

Python3:

```
class Solution:
def findLongestWord(self, s: str, dictionary: List[str]) -> str:
```

Python:

```
class Solution(object):
def findLongestWord(self, s, dictionary):
"""
:type s: str
:type dictionary: List[str]
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {string[]} dictionary
* @return {string}
*/
var findLongestWord = function(s, dictionary) {

};
```

TypeScript:

```
function findLongestWord(s: string, dictionary: string[]): string {

};
```

C#:

```
public class Solution {
public string FindLongestWord(string s, IList<string> dictionary) {
```

```
}
}
```

C:

```
char* findLongestWord(char* s, char** dictionary, int dictionarySize) {

}
```

Go:

```
func findLongestWord(s string, dictionary []string) string {

}
```

Kotlin:

```
class Solution {
fun findLongestWord(s: String, dictionary: List<String>): String {

}
}
```

Swift:

```
class Solution {
func findLongestWord(_ s: String, _ dictionary: [String]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn find_longest_word(s: String, dictionary: Vec<String>) -> String {

}
}
```

Ruby:

```
# @param {String} s
# @param {String[]} dictionary
# @return {String}
def find_longest_word(s, dictionary)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param String[] $dictionary
* @return String
*/
function findLongestWord($s, $dictionary) {

}
}
```

Dart:

```
class Solution {
String findLongestWord(String s, List<String> dictionary) {

}
}
```

Scala:

```
object Solution {
def findLongestWord(s: String, dictionary: List[String]): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_longest_word(s :: String.t, dictionary :: [String.t]) :: String.t
def find_longest_word(s, dictionary) do
```

```
end
end
```

Erlang:

```
-spec find_longest_word(S :: unicode:unicode_binary(), Dictionary ::
[unicode:unicode_binary()]) -> unicode:unicode_binary().
find_longest_word(S, Dictionary) ->
.
```

Racket:

```
(define/contract (find-longest-word s dictionary)
(-> string? (listof string?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Word in Dictionary through Deleting
* Difficulty: Medium
* Tags: array, string, graph, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string findLongestWord(string s, vector<string>& dictionary) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Word in Dictionary through Deleting
* Difficulty: Medium
* Tags: array, string, graph, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String findLongestWord(String s, List<String> dictionary) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Word in Dictionary through Deleting
Difficulty: Medium
Tags: array, string, graph, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def findLongestWord(self, s: str, dictionary: List[str]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def findLongestWord(self, s, dictionary):
"""
:type s: str
:type dictionary: List[str]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Longest Word in Dictionary through Deleting
* Difficulty: Medium
* Tags: array, string, graph, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} s
* @param {string[]} dictionary
* @return {string}
*/
var findLongestWord = function(s, dictionary) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Word in Dictionary through Deleting
* Difficulty: Medium
* Tags: array, string, graph, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function findLongestWord(s: string, dictionary: string[]): string {

};
```

C# Solution:

```
/*
* Problem: Longest Word in Dictionary through Deleting
* Difficulty: Medium
```

```
* Tags: array, string, graph, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string FindLongestWord(string s, IList<string> dictionary) {

}
}
```

C Solution:

```
/*
* Problem: Longest Word in Dictionary through Deleting
* Difficulty: Medium
* Tags: array, string, graph, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* findLongestWord(char* s, char** dictionary, int dictionarySize) {

}
```

Go Solution:

```
// Problem: Longest Word in Dictionary through Deleting
// Difficulty: Medium
// Tags: array, string, graph, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func findLongestWord(s string, dictionary []string) string {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun findLongestWord(s: String, dictionary: List<String>): String {

}
}
```

Swift Solution:

```
class Solution {
func findLongestWord(_ s: String, _ dictionary: [String]) -> String {

}
}
```

Rust Solution:

```
// Problem: Longest Word in Dictionary through Deleting
// Difficulty: Medium
// Tags: array, string, graph, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn find_longest_word(s: String, dictionary: Vec<String>) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {String[]} dictionary
# @return {String}
def find_longest_word(s, dictionary)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param String[] $dictionary
* @return String
*/
function findLongestWord($s, $dictionary) {

}
}
```

Dart Solution:

```
class Solution {
String findLongestWord(String s, List<String> dictionary) {

}
}
```

Scala Solution:

```
object Solution {
def findLongestWord(s: String, dictionary: List[String]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_longest_word(s :: String.t, dictionary :: [String.t]) :: String.t
def find_longest_word(s, dictionary) do

end
end
```

Erlang Solution:

```
-spec find_longest_word(S :: unicode:unicode_binary(), Dictionary ::
[unicode:unicode_binary()]) -> unicode:unicode_binary().
find_longest_word(S, Dictionary) ->
.
```

Racket Solution:

```
(define/contract (find-longest-word s dictionary)
(-> string? (listof string?) string?)
)
```