Problem 720: Longest Word in Dictionary

Problem Information

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

Problem Description

Given an array of strings

words

representing an English Dictionary, return

the longest word in

words

that can be built one character at a time by other words in

words

.

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

Note that the word should be built from left to right with each additional character being added to the end of a previous word.

Example 1:

Input:

words = ["w","wo","wor","worl","world"]

Output:

"world"

Explanation:

The word "world" can be built one character at a time by "w", "wo", "wor", and "worl".

Example 2:

Input:

words = ["a","banana","app","appl","ap","apply","apple"]

Output:

"apple"

Explanation:

Both "apply" and "apple" can be built from other words in the dictionary. However, "apple" is lexicographically smaller than "apply".

Constraints:

1 <= words.length <= 1000

1 <= words[i].length <= 30

words[i]

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
string longestWord(vector<string>& words) {

}
};
```

Java:

```
class Solution {
public String longestWord(String[] words) {

}
}
```

Python3:

```
class Solution:
def longestWord(self, words: List[str]) -> str:
```

Python:

```
class Solution(object):
def longestWord(self, words):
"""
:type words: List[str]
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string[]} words
* @return {string}
*/
var longestWord = function(words) {

};
```

TypeScript:

```
function longestWord(words: string[]): string {
```

```
};
```

C#:

```
public class Solution {
public string LongestWord(string[] words) {

}
}
```

C:

```
char* longestWord(char** words, int wordsSize) {

}
```

Go:

```
func longestWord(words []string) string {

}
```

Kotlin:

```
class Solution {
fun longestWord(words: Array<String>): String {

}
}
```

Swift:

```
class Solution {
func longestWord(_ words: [String]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn longest_word(words: Vec<String>) -> String {
```

```
}
}
```

Ruby:

```
# @param {String[]} words
# @return {String}
def longest_word(words)

end
```

PHP:

```
class Solution {

/**
* @param String[] $words
* @return String
*/
function longestWord($words) {

}
}
```

Dart:

```
class Solution {
String longestWord(List<String> words) {

}
}
```

Scala:

```
object Solution {
def longestWord(words: Array[String]): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec longest_word(words :: [String.t]) :: String.t
def longest_word(words) do

end
end
```

Erlang:

```
-spec longest_word(Words :: [unicode:unicode_binary()]) ->
unicode:unicode_binary().
longest_word(Words) ->
.
```

Racket:

```
(define/contract (longest-word words)
(-> (listof string?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Word in Dictionary
* Difficulty: Medium
* Tags: array, string, graph, hash, sort
*
* 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 longestWord(vector<string>& words) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Word in Dictionary
* Difficulty: Medium
* Tags: array, string, graph, hash, sort
*
* 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 longestWord(String[] words) {

}
}
```

Python3 Solution:

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

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 longestWord(self, words: List[str]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def longestWord(self, words):
"""
:type words: List[str]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Longest Word in Dictionary
* Difficulty: Medium
* Tags: array, string, graph, hash, sort
*
* 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[]} words
* @return {string}
*/
var longestWord = function(words) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Word in Dictionary
* Difficulty: Medium
* Tags: array, string, graph, hash, sort
*
* 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 longestWord(words: string[]): string {

};
```

C# Solution:

```
/*
* Problem: Longest Word in Dictionary
* Difficulty: Medium
* Tags: array, string, graph, hash, sort
```

```
*
* 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 LongestWord(string[] words) {

}
}
```

C Solution:

```
/*
* Problem: Longest Word in Dictionary
* Difficulty: Medium
* Tags: array, string, graph, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

char* longestWord(char** words, int wordsSize) {

}
```

Go Solution:

```
// Problem: Longest Word in Dictionary
// Difficulty: Medium
// Tags: array, string, graph, hash, sort
//
// 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 longestWord(words []string) string {

}
```

Kotlin Solution:

```
class Solution {
fun longestWord(words: Array<String>): String {

}
}
```

Swift Solution:

```
class Solution {
func longestWord(_ words: [String]) -> String {

}
}
```

Rust Solution:

```
// Problem: Longest Word in Dictionary
// Difficulty: Medium
// Tags: array, string, graph, hash, sort
//
// 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 longest_word(words: Vec<String>) -> String {

}
}
```

Ruby Solution:

```
# @param {String[]} words
# @return {String}
def longest_word(words)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $words
* @return String
*/
function longestWord($words) {

}
}
```

Dart Solution:

```
class Solution {
String longestWord(List<String> words) {

}
}
```

Scala Solution:

```
object Solution {
def longestWord(words: Array[String]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec longest_word(words :: [String.t]) :: String.t
def longest_word(words) do

end
end
```

Erlang Solution:

```
-spec longest_word(Words :: [unicode:unicode_binary()]) ->
unicode:unicode_binary().
longest_word(Words) ->
.
```

Racket Solution:

```
(define/contract (longest-word words)
(-> (listof string?) string?)
)
```