Problem 1858: Longest Word With All Prefixes

Problem Information

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

Problem Description

Given an array of strings

words

, find the

longest

string in

words

such that

every prefix

of it is also in

words

.

For example, let

words = ["a", "app", "ap"]

. The string

"app"

has prefixes

"ap"

and

"a"

, all of which are in

words

.

Return

the string described above. If there is more than one string with the same length, return the

lexicographically smallest

one, and if no string exists, return

""

.

Example 1:

Input:

words = ["k","ki","kir","kira", "kiran"]

Output:

"kiran"

Explanation:

"kiran" has prefixes "kira", "kir", "ki", and "k", and all of them appear in words.

Example 2:

Input:

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

Output:

"apple"

Explanation:

Both "apple" and "apply" have all their prefixes in words. However, "apple" is lexicographically smaller, so we return that.

Example 3:

Input:

words = ["abc", "bc", "ab", "qwe"]

Output:

""

Constraints:

1 <= words.length <= 10

5

1 <= words[i].length <= 10

5

1 <= sum(words[i].length) <= 10

5

words[i]

consists only 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 With All Prefixes
* Difficulty: Medium
* Tags: array, string, graph, search
```

```
*
* 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 longestWord(vector<string>& words) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Word With All Prefixes
* Difficulty: Medium
* Tags: array, string, graph, search
*
* 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 longestWord(String[] words) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Word With All Prefixes
Difficulty: Medium
Tags: array, string, graph, search

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 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 With All Prefixes
* Difficulty: Medium
* Tags: array, string, graph, search
*
* 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[]} words
* @return {string}
*/
var longestWord = function(words) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Word With All Prefixes
* Difficulty: Medium
* Tags: array, string, graph, search
*
* 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 longestWord(words: string[]): string {

};
```

C# Solution:

```
/*
* Problem: Longest Word With All Prefixes
* Difficulty: Medium
* Tags: array, string, graph, search
*
* 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 LongestWord(string[] words) {

}
}
```

C Solution:

```
/*
* Problem: Longest Word With All Prefixes
* Difficulty: Medium
* Tags: array, string, graph, search
*
* 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* longestWord(char** words, int wordsSize) {

}
```

Go Solution:

```
// Problem: Longest Word With All Prefixes
// Difficulty: Medium
// Tags: array, string, graph, search
//
// 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 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 With All Prefixes
// Difficulty: Medium
// Tags: array, string, graph, search
//
// 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 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?)
)
```