Problem 943: Find the Shortest Superstring

Problem Information

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

Problem Description

Given an array of strings

words

, return

the smallest string that contains each string in

words

as a substring

. If there are multiple valid strings of the smallest length, return

any of them

.

You may assume that no string in

words

is a substring of another string in

words

.

Example 1:

Input:

words = ["alex","loves","leetcode"]

Output:

"alexlovesleetcode"

Explanation:

All permutations of "alex","loves","leetcode" would also be accepted.

Example 2:

Input:

words = ["catg","ctaagt","gcta","ttca","atgcatc"]

Output:

"gctaagttcatgcatc"

Constraints:

1 <= words.length <= 12

1 <= words[i].length <= 20

words[i]

consists of lowercase English letters.

All the strings of

words

are

unique

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (shortest-superstring words)
(-> (listof string?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Shortest Superstring
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Find the Shortest Superstring
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public String shortestSuperstring(String[] words) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Shortest Superstring
Difficulty: Hard
Tags: array, string, tree, dp

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

```
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Shortest Superstring
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Find the Shortest Superstring
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function shortestSuperstring(words: string[]): string {

};
```

C# Solution:

```
/*
* Problem: Find the Shortest Superstring
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public string ShortestSuperstring(string[] words) {

}
}
```

C Solution:

```
/*
* Problem: Find the Shortest Superstring
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

}
```

Go Solution:

```
// Problem: Find the Shortest Superstring
// Difficulty: Hard
// Tags: array, string, tree, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func shortestSuperstring(words []string) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Shortest Superstring
// Difficulty: Hard
// Tags: array, string, tree, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (shortest-superstring words)
(-> (listof string?) string?)
)
```