Problem 3324: Find the Sequence of Strings Appeared on the Screen

Problem Information

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

Problem Description

You are given a string

target

.

Alice is going to type

target

on her computer using a special keyboard that has

only two

keys:

Key 1 appends the character

"a"

to the string on the screen.

Key 2 changes the

last

character of the string on the screen to its

next

character in the English alphabet. For example,

"c"

changes to

"d"

and

"z"

changes to

"a"

.

Note

that initially there is an

empty

string

""

on the screen, so she can

only

press key 1.

Return a list of

all

strings that appear on the screen as Alice types

target

, in the order they appear, using the

minimum

key presses.

Example 1:

Input:

target = "abc"

Output:

["a","aa","ab","aba","abb","abc"]

Explanation:

The sequence of key presses done by Alice are:

Press key 1, and the string on the screen becomes

"a"

.

Press key 1, and the string on the screen becomes

"aa"

.

Press key 2, and the string on the screen becomes

"ab"

.

Press key 1, and the string on the screen becomes

"aba"

.

Press key 2, and the string on the screen becomes

"abb"

.

Press key 2, and the string on the screen becomes

"abc"

.

Example 2:

Input:

target = "he"

Output:

["a","b","c","d","e","f","g","h","ha","hb","hc","hd","he"]

Constraints:

1 <= target.length <= 400

target

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
vector<string> stringSequence(string target) {

}
};
```

Java:

```
class Solution {
public List<String> stringSequence(String target) {

}
}
```

Python3:

```
class Solution:
def stringSequence(self, target: str) -> List[str]:
```

Python:

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

JavaScript:

```
/**
* @param {string} target
```

```
* @return {string[]}
*/
var stringSequence = function(target) {

};
```

TypeScript:

```
function stringSequence(target: string): string[] {

};
```

C#:

```
public class Solution {
public IList<string> StringSequence(string target) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** stringSequence(char* target, int* returnSize) {

}
```

Go:

```
func stringSequence(target string) []string {

}
```

Kotlin:

```
class Solution {
fun stringSequence(target: String): List<String> {

}
}
```

Swift:

```
class Solution {
func stringSequence(_ target: String) -> [String] {

}
}
```

Rust:

```
impl Solution {
pub fn string_sequence(target: String) -> Vec<String> {

}
}
```

Ruby:

```
# @param {String} target
# @return {String[]}
def string_sequence(target)

end
```

PHP:

```
class Solution {

/**
* @param String $target
* @return String[]
*/
function stringSequence($target) {

}
}
```

Dart:

```
class Solution {
List<String> stringSequence(String target) {

}
```

```
}
```

Scala:

```
object Solution {
def stringSequence(target: String): List[String] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec string_sequence(target :: String.t) :: [String.t]
def string_sequence(target) do

end
end
```

Erlang:

```
-spec string_sequence(Target :: unicode:unicode_binary()) ->
[unicode:unicode_binary()].
string_sequence(Target) ->
.
```

Racket:

```
(define/contract (string-sequence target)
(-> string? (listof string?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Sequence of Strings Appeared on the Screen
* Difficulty: Medium
* Tags: string
```

```
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
vector<string> stringSequence(string target) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Sequence of Strings Appeared on the Screen
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public List<String> stringSequence(String target) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Sequence of Strings Appeared on the Screen
Difficulty: Medium
Tags: string

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
```

```
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Sequence of Strings Appeared on the Screen
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} target
* @return {string[]}
*/
var stringSequence = function(target) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Sequence of Strings Appeared on the Screen
* Difficulty: Medium
```

```
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function stringSequence(target: string): string[] {

};
```

C# Solution:

```
/*
* Problem: Find the Sequence of Strings Appeared on the Screen
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public IList<string> StringSequence(string target) {

}
}
```

C Solution:

```
/*
* Problem: Find the Sequence of Strings Appeared on the Screen
* Difficulty: Medium
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** stringSequence(char* target, int* returnSize) {

}
```

Go Solution:

```
// Problem: Find the Sequence of Strings Appeared on the Screen
// Difficulty: Medium
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func stringSequence(target string) []string {

}
```

Kotlin Solution:

```
class Solution {
fun stringSequence(target: String): List<String> {

}
}
```

Swift Solution:

```
class Solution {
func stringSequence(_ target: String) -> [String] {

}
}
```

Rust Solution:

```
// Problem: Find the Sequence of Strings Appeared on the Screen
// Difficulty: Medium
```

```
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn string_sequence(target: String) -> Vec<String> {

}
}
```

Ruby Solution:

```
# @param {String} target
# @return {String[]}
def string_sequence(target)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $target
* @return String[]
*/
function stringSequence($target) {

}
}
```

Dart Solution:

```
class Solution {
List<String> stringSequence(String target) {

}
}
```

Scala Solution:

```
object Solution {
def stringSequence(target: String): List[String] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec string_sequence(target :: String.t) :: [String.t]
def string_sequence(target) do

end
end
```

Erlang Solution:

```
-spec string_sequence(Target :: unicode:unicode_binary()) ->
[unicode:unicode_binary()].
string_sequence(Target) ->
.
```

Racket Solution:

```
(define/contract (string-sequence target)
(-> string? (listof string?))
)
```