Problem 3403: Find the Lexicographically Largest String From the Box I

Problem Information

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

Problem Description

You are given a string

word

, and an integer

numFriends

.

Alice is organizing a game for her

numFriends

friends. There are multiple rounds in the game, where in each round:

word

is split into

numFriends

non-empty

strings, such that no previous round has had the

exact

same split.

All the split words are put into a box.

Find the

lexicographically largest

string from the box after all the rounds are finished.

Example 1:

Input:

word = "dbca", numFriends = 2

Output:

"dbc"

Explanation:

All possible splits are:

"d"

and

"bca"

.

"db"

and

"ca"

.

"dbc"

and

"a"

.

Example 2:

Input:

word = "gggg", numFriends = 4

Output:

"g"

Explanation:

The only possible split is:

"g"

,

"g"

,

"g"

, and

"g"

.

Constraints:

1 <= word.length <= 5 * 10

3

word

consists only of lowercase English letters.

1 <= numFriends <= word.length

Code Snippets

C++:

```
class Solution {
public:
string answerString(string word, int numFriends) {

}
};
```

Java:

```
class Solution {
public String answerString(String word, int numFriends) {

}
}
```

Python3:

```
class Solution:
def answerString(self, word: str, numFriends: int) -> str:
```

Python:

```
class Solution(object):
def answerString(self, word, numFriends):
"""
:type word: str
:type numFriends: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} word
* @param {number} numFriends
* @return {string}
*/
var answerString = function(word, numFriends) {

};
```

TypeScript:

```
function answerString(word: string, numFriends: number): string {

};
```

C#:

```
public class Solution {
public string AnswerString(string word, int numFriends) {

}
}
```

C:

```
char* answerString(char* word, int numFriends) {

}
```

Go:

```
func answerString(word string, numFriends int) string {
```

```
}
```

Kotlin:

```
class Solution {
fun answerString(word: String, numFriends: Int): String {

}
}
```

Swift:

```
class Solution {
func answerString(_ word: String, _ numFriends: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn answer_string(word: String, num_friends: i32) -> String {

}
}
```

Ruby:

```
# @param {String} word
# @param {Integer} num_friends
# @return {String}
def answer_string(word, num_friends)

end
```

PHP:

```
class Solution {

/**
* @param String $word
* @param Integer $numFriends
```

```
* @return String
*/
function answerString($word, $numFriends) {

}
}
```

Dart:

```
class Solution {
String answerString(String word, int numFriends) {

}
}
```

Scala:

```
object Solution {
def answerString(word: String, numFriends: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec answer_string(word :: String.t, num_friends :: integer) :: String.t
def answer_string(word, num_friends) do

end
end
```

Erlang:

```
-spec answer_string(Word :: unicode:unicode_binary(), NumFriends ::
integer()) -> unicode:unicode_binary().
answer_string(Word, NumFriends) ->
.
```

Racket:

```
(define/contract (answer-string word numFriends)
(-> string? exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Lexicographically Largest String From the Box I
* Difficulty: Medium
* Tags: array, string, graph
*
* 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 answerString(string word, int numFriends) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Lexicographically Largest String From the Box I
* Difficulty: Medium
* Tags: array, string, graph
*
* 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 answerString(String word, int numFriends) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Find the Lexicographically Largest String From the Box I
Difficulty: Medium
Tags: array, string, graph

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 answerString(self, word: str, numFriends: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def answerString(self, word, numFriends):
"""
:type word: str
:type numFriends: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Lexicographically Largest String From the Box I
* Difficulty: Medium
* Tags: array, string, graph
*
* 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} word
* @param {number} numFriends
* @return {string}
*/
var answerString = function(word, numFriends) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Lexicographically Largest String From the Box I
* Difficulty: Medium
* Tags: array, string, graph
*
* 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 answerString(word: string, numFriends: number): string {

};
```

C# Solution:

```
/*
* Problem: Find the Lexicographically Largest String From the Box I
* Difficulty: Medium
* Tags: array, string, graph
*
* 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 AnswerString(string word, int numFriends) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Find the Lexicographically Largest String From the Box I
* Difficulty: Medium
* Tags: array, string, graph
*
* 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* answerString(char* word, int numFriends) {

}
```

Go Solution:

```
// Problem: Find the Lexicographically Largest String From the Box I
// Difficulty: Medium
// Tags: array, string, graph
//
// 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 answerString(word string, numFriends int) string {

}
```

Kotlin Solution:

```
class Solution {
fun answerString(word: String, numFriends: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func answerString(_ word: String, _ numFriends: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Find the Lexicographically Largest String From the Box I
// Difficulty: Medium
// Tags: array, string, graph
//
// 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 answer_string(word: String, num_friends: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {String} word
# @param {Integer} num_friends
# @return {String}
def answer_string(word, num_friends)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $word
* @param Integer $numFriends
* @return String
*/
function answerString($word, $numFriends) {
```

```
}
}
```

Dart Solution:

```
class Solution {
String answerString(String word, int numFriends) {

}
}
```

Scala Solution:

```
object Solution {
def answerString(word: String, numFriends: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec answer_string(word :: String.t, num_friends :: integer) :: String.t
def answer_string(word, num_friends) do

end
end
```

Erlang Solution:

```
-spec answer_string(Word :: unicode:unicode_binary(), NumFriends ::
integer()) -> unicode:unicode_binary().
answer_string(Word, NumFriends) ->
.
```

Racket Solution:

```
(define/contract (answer-string word numFriends)
(-> string? exact-integer? string?)
)
```