Problem 960: Delete Columns to Make Sorted III

Problem Information

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

Problem Description

You are given an array of

n

strings

strs

, all of the same length.

We may choose any deletion indices, and we delete all the characters in those indices for each string.

For example, if we have

strs = ["abcdef","uvwxyz"]

and deletion indices

{0, 2, 3}

, then the final array after deletions is

["bef", "vyz"]

.

Suppose we chose a set of deletion indices

answer

such that after deletions, the final array has

every string (row) in lexicographic

order. (i.e.,

(strs[0][0] <= strs[0][1] <= ... <= strs[0][strs[0].length - 1])

, and

(strs[1][0] <= strs[1][1] <= ... <= strs[1][strs[1].length - 1])

, and so on). Return

the minimum possible value of

answer.length

.

Example 1:

Input:

strs = ["babca","bbazb"]

Output:

3

Explanation:

After deleting columns 0, 1, and 4, the final array is strs = ["bc", "az"]. Both these rows are individually in lexicographic order (ie. strs[0][0] <= strs[0][1] and strs[1][0] <= strs[1][1]). Note that strs[0] > strs[1] - the array strs is not necessarily in lexicographic order.

Example 2:

Input:

strs = ["edcba"]

Output:

4

Explanation:

If we delete less than 4 columns, the only row will not be lexicographically sorted.

Example 3:

Input:

strs = ["ghi","def","abc"]

Output:

0

Explanation:

All rows are already lexicographically sorted.

Constraints:

n == strs.length

1 <= n <= 100

1 <= strs[i].length <= 100

strs[i]

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int minDeletionSize(vector<string>& strs) {

}
};
```

Java:

```
class Solution {
public int minDeletionSize(String[] strs) {

}
}
```

Python3:

```
class Solution:
def minDeletionSize(self, strs: List[str]) -> int:
```

Python:

```
class Solution(object):
def minDeletionSize(self, strs):
"""
:type strs: List[str]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string[]} strs
* @return {number}
*/
```

```

var minDeletionSize = function(strs) {

};
```

TypeScript:

```
function minDeletionSize(strs: string[]): number {

};
```

C#:

```
public class Solution {
public int MinDeletionSize(string[] strs) {

}
}
```

C:

```
int minDeletionSize(char** strs, int strsSize) {

}
```

Go:

```
func minDeletionSize(strs []string) int {

}
```

Kotlin:

```
class Solution {
fun minDeletionSize(strs: Array<String>): Int {

}
}
```

Swift:

```
class Solution {
func minDeletionSize(_ strs: [String]) -> Int {



}
}
```

Rust:

```
impl Solution {
pub fn min_deletion_size(strs: Vec<String>) -> i32 {

}
}
```

Ruby:

```
# @param {String[]} strs
# @return {Integer}
def min_deletion_size(strs)

end
```

PHP:

```
class Solution {

/**
* @param String[] $strs
* @return Integer
*/
function minDeletionSize($strs) {

}
}
```

Dart:

```
class Solution {
int minDeletionSize(List<String> strs) {

}
}
```

Scala:

```
object Solution {
def minDeletionSize(strs: Array[String]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_deletion_size(strs :: [String.t]) :: integer
def min_deletion_size(strs) do

end
end
```

Erlang:

```
-spec min_deletion_size(Strs :: [unicode:unicode_binary()]) -> integer().
min_deletion_size(Strs) ->
.
```

Racket:

```
(define/contract (min-deletion-size strs)
(-> (listof string?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Delete Columns to Make Sorted III
* Difficulty: Hard
* Tags: array, string, graph, dp, sort
*
* 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:
int minDeletionSize(vector<string>& strs) {

}
};
```

Java Solution:

```
/**
* Problem: Delete Columns to Make Sorted III
* Difficulty: Hard
* Tags: array, string, graph, dp, sort
*
* 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 int minDeletionSize(String[] strs) {

}
}
```

Python3 Solution:

```
"""
Problem: Delete Columns to Make Sorted III
Difficulty: Hard
Tags: array, string, graph, dp, sort

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 minDeletionSize(self, strs: List[str]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minDeletionSize(self, strs):
"""
:type strs: List[str]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Delete Columns to Make Sorted III
* Difficulty: Hard
* Tags: array, string, graph, dp, sort
*
* 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[]} strs
* @return {number}
*/
var minDeletionSize = function(strs) {

};
```

TypeScript Solution:

```
/**
* Problem: Delete Columns to Make Sorted III
* Difficulty: Hard
* Tags: array, string, graph, dp, sort
*
* 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 minDeletionSize(strs: string[]): number {
};
```

C# Solution:

```
/*
* Problem: Delete Columns to Make Sorted III
* Difficulty: Hard
* Tags: array, string, graph, dp, sort
*
* 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 int MinDeletionSize(string[] strs) {

}
}
```

C Solution:

```
/*
* Problem: Delete Columns to Make Sorted III
* Difficulty: Hard
* Tags: array, string, graph, dp, sort
*
* 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
*/

int minDeletionSize(char** strs, int strsSize) {

}
```

Go Solution:

```
// Problem: Delete Columns to Make Sorted III
// Difficulty: Hard
// Tags: array, string, graph, dp, sort
//
// 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 minDeletionSize(strs []string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minDeletionSize(strs: Array<String>): Int {

}
}
```

Swift Solution:

```
class Solution {
func minDeletionSize(_ strs: [String]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Delete Columns to Make Sorted III
// Difficulty: Hard
// Tags: array, string, graph, dp, sort
//
// 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 min_deletion_size(strs: Vec<String>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String[]} strs
# @return {Integer}
def min_deletion_size(strs)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $strs
* @return Integer
*/
function minDeletionSize($strs) {

}
}
```

Dart Solution:

```
class Solution {
int minDeletionSize(List<String> strs) {

}
}
```

Scala Solution:

```
object Solution {
def minDeletionSize(strs: Array[String]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_deletion_size(strs :: [String.t]) :: integer
def min_deletion_size(strs) do
```

```
end
end
```

Erlang Solution:

```
-spec min_deletion_size(Strs :: [unicode:unicode_binary()]) -> integer().
min_deletion_size(Strs) ->
.
```

Racket Solution:

```
(define/contract (min-deletion-size strs)
(-> (listof string?) exact-integer?)
)
```