944. Delete Columns to Make Sorted

Solved

Easy | Topics | Companies

You are given an array of `n` strings `strs`, all of the same length.

The strings can be arranged such that there is one on each line, making a grid.

- For example, `strs = ["abc", "bce", "cae"]` can be arranged as follows:

> abc
> bce
> cae

You want to **delete** the columns that are **not sorted lexicographically**. In the above example (**0-indexed**), columns 0 (`'a'`, `'b'`, `'c'`) and 2 (`'c'`, `'e'`, `'e'`) are sorted, while column 1 (`'b'`, `'c'`, `'a'`) is not, so you would delete column 1.

Return *the number of columns that you will delete.*

Example 1:

```
Input: strs = ["cba","daf","ghi"]
Output: 1
Explanation: The grid looks as follows:
  cba
  daf
  ghi
Columns 0 and 2 are sorted, but column 1 is not, so you only
need to delete 1 column.
```

Example 2:

```
Input: strs = ["a","b"]
Output: 0
Explanation: The grid looks as follows:
  a
  b
Column 0 is the only column and is sorted, so you will not
delete any columns.
```

Example 3:

```
Input: strs = ["zyx","wvu","tsr"]
Output: 3
Explanation: The grid looks as follows:
  zyx
  wvu
  tsr
All 3 columns are not sorted, so you will delete all 3.
```

Constraints:

- `n == strs.length`
- `1 <= n <= 100`
- `1 <= strs[i].length <= 1000`
- `strs[i]` consists of lowercase English letters.

Python:

```
class Solution:
    def minDeletionSize(self, strs: List[str]) -> int:
        res = 0

        for col in range(len(strs[0])):
            for row in range(1, len(strs)):
                if strs[row][col] < strs[row - 1][col]:
                    res += 1
                    break

        return res
```

JavaScript:

```
const minDeletionSize = strs => {
    let res = 0;

    for (let col = 0; col < strs[0].length; col++) {
```

```
        for (let row = 1; row < strs.length; row++) {
            if (strs[row][col] < strs[row - 1][col]) {
                res++;
                break;
            }
        }
    }

    return res;
};
```

Java:

```
class Solution {
    public int minDeletionSize(String[] strs) {
        // Initialize the delete count to 0
        int deleteCount = 0;
        // Get the number of rows and columns in the grid
        int n = strs.length;
        int m = strs[0].length();
        // Iterate through each column of the grid
        for (int i = 0; i < m; i++) {
            // Iterate through each element in the column
            for (int j = 1; j < n; j++) {
                // If the current element is lexicographically smaller than the previous element,
                // increment the delete count and break out of the loop
                if (strs[j].charAt(i) < strs[j - 1].charAt(i)) {
                    deleteCount++;
                    break;
                }
            }
        }
        // Return the delete count
        return deleteCount;
    }
}
```