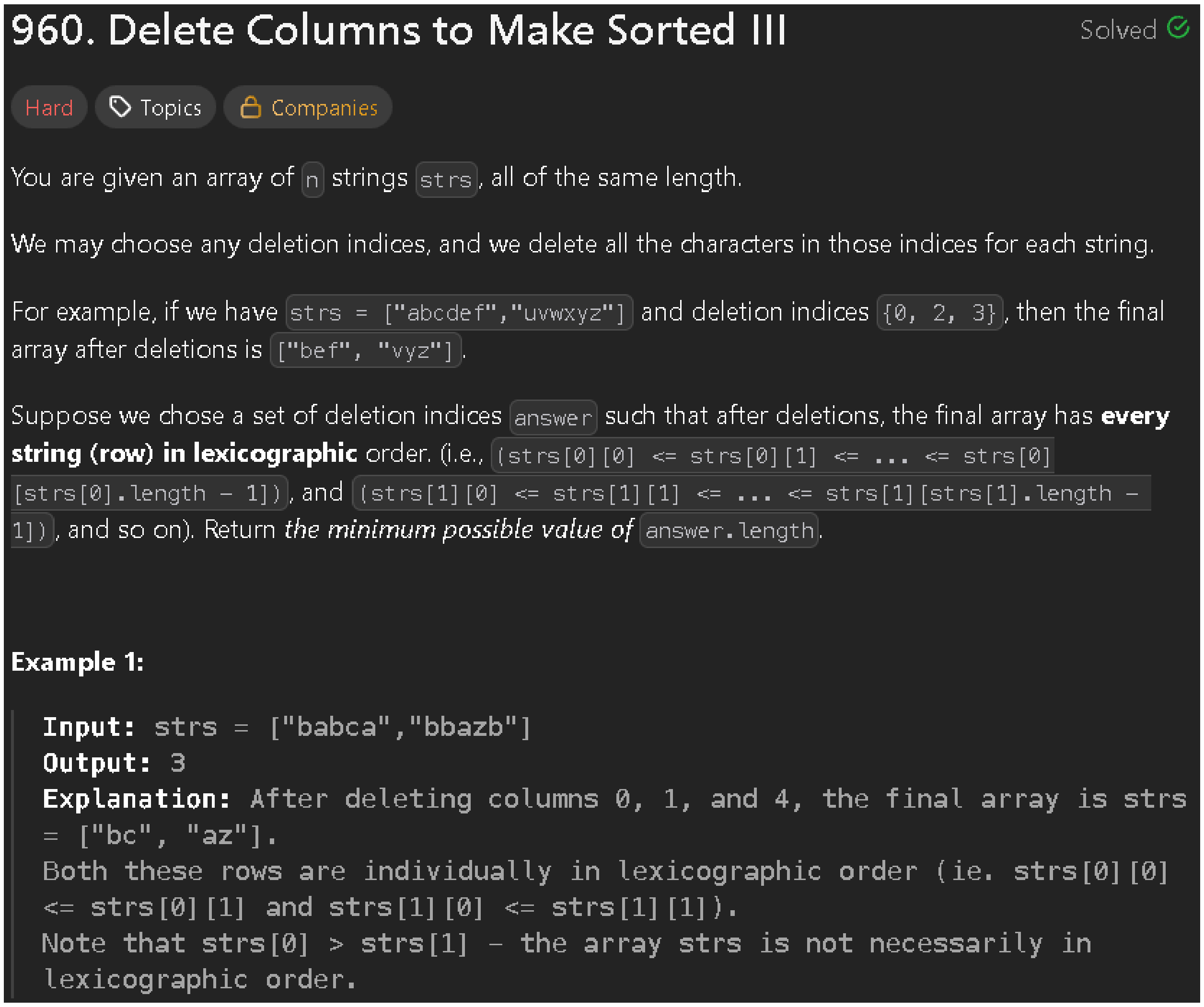


960. Delete Columns to Make Sorted III

Solved ✓

Hard   Topics   Companies

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

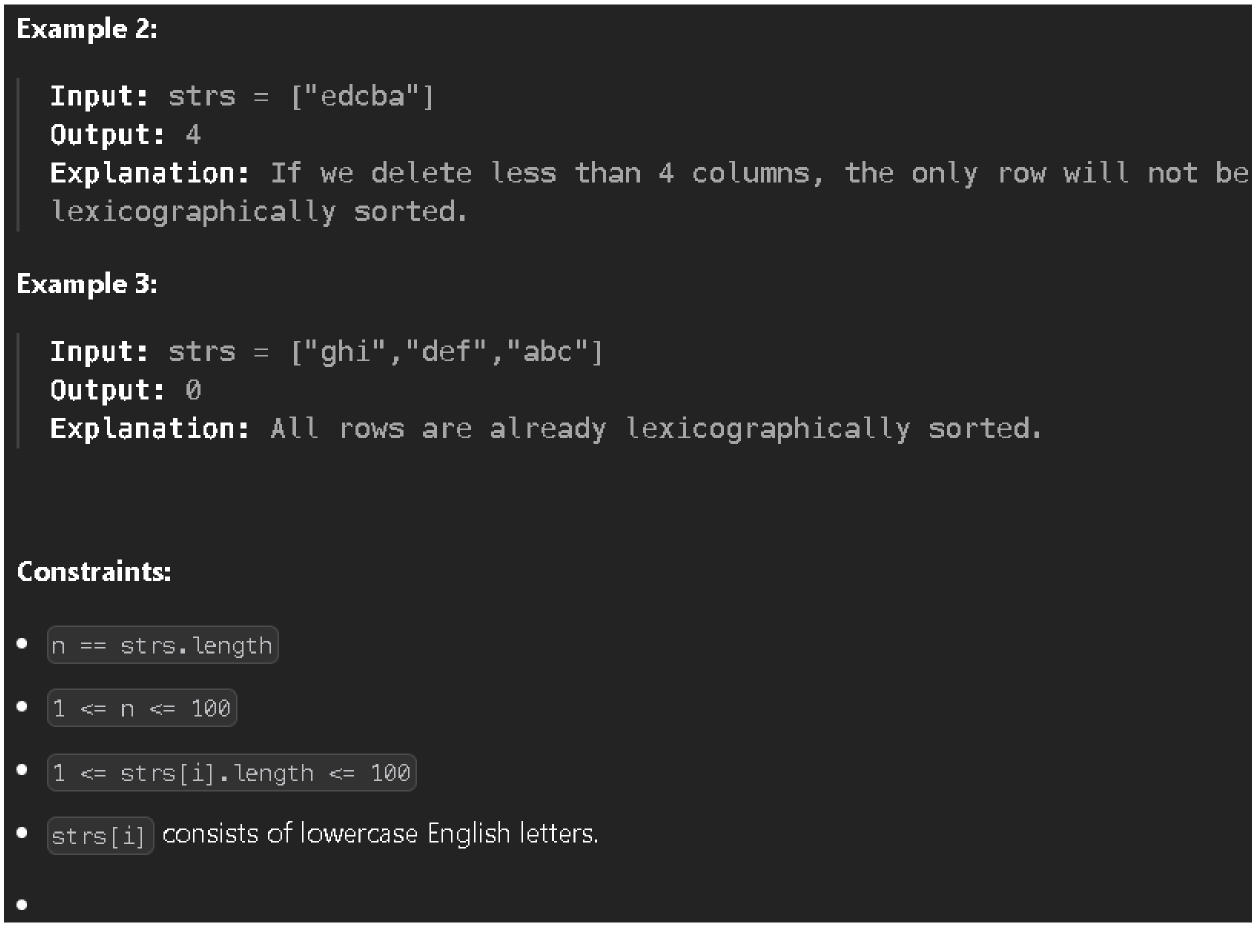
We may choose any deletion indices, and we delete all the characters in those indices for each string.

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.
-

Python:

```
from typing import List

class Solution:
    def minDeletionSize(self, strs: List[str]) -> int:
        n = len(strs[0])
        m = len(strs)
        dp = [1] * n

        for i in range(1, n):
            for j in range(i):
                ok = True
                for r in range(m):
                    if strs[r][j] > strs[r][i]:
                        ok = False
                        break
                if ok:
                    dp[i] = max(dp[i], dp[j] + 1)
        mx = 0
        for v in dp:
            mx = max(mx, v)
        return n - mx
```

JavaScript:

```
/**
 * @param {string[]} strs
 * @return {number}
 */
var minDeletionSize = function(strs) {
  const n = strs[0].length;
  const m = strs.length;
  const dp = new Array(n).fill(1);

  for (let i = 1; i < n; i++) {
    for (let j = 0; j < i; j++) {
      let ok = true;
      for (let r = 0; r < m; r++) {
        if (strs[r].charCodeAt(j) > strs[r].charCodeAt(i)) { ok = false; break; }
      }
      if (ok) dp[i] = Math.max(dp[i], dp[j] + 1);
    }
  }

  let mx = 0;
  for (const v of dp) mx = Math.max(mx, v);
  return n - mx;
};
```

Java:

```
class Solution {
    public int minDeletionSize(String[] strs) {
        int n = strs[0].length(), m = strs.length;
        int[] dp = new int[n];
        Arrays.fill(dp, 1);
        for (int i = 1; i < n; i++) {
            for (int j = 0; j < i; j++) {
                if (isValid(strs, j, i)) {
                    dp[i] = Math.max(dp[i], dp[j] + 1);
                }
            }
        }
        int max = 0;
        for (int val : dp) max = Math.max(max, val);
        return n - max;
    }

    private boolean isValid(String[] strs, int j, int i) {
        for (String s : strs) {
            if (s.charAt(j) > s.charAt(i)) return false;
        }
        return true;
    }
}
```