Daily Coding Problem #76

Problem

This problem was asked by Google.

You are given an N by M 2D matrix of lowercase letters. Determine the minimum number of columns that can be removed to ensure that each row is ordered from top to bottom lexicographically. That is, the letter at each column is lexicographically later as you go down each row. It does not matter whether each row itself is ordered lexicographically.

For example, given the following table:

```
cba
daf
ghi
```

This is not ordered because of the a in the center. We can remove the second column to make it ordered:

```
ca
df
gi
```

So your function should return 1, since we only needed to remove 1 column.

As another example, given the following table:

```
abcdef
```

Your function should return 0, since the rows are already ordered (there's only one row).

As another example, given the following table:

```
zyx
wvu
```

tsr

Your function should return 3, since we would need to remove all the columns to order it.

Solution

For this question, we can look over each column, check that each one is ordered, and remove them if the column is not sorted:

```
def bad_cols(board):
    num_bad_cols = 0
    num_cols = len(board[0])
    i = 0
    while i < num_cols:
        if is_sorted_up_to(board, i):
            i += 1
            continue
        else:
            remove_col(board, i)
            num_bad_cols += 1
            num_cols -= 1

    return num_bad_cols

def remove_col(board, i):
    for row in board:
        row.pop(i)

def is_sorted_up_to(board, i):
    '''Returns whether the table is sorted in lexicographic order up to column i.'''
    return all(board[r][:i + 1] <= board[r + 1][:i + 1] for r in range(len(board) - 1))
```

Recall that we have M rows and N columns. We're iterating over each column, and checking all the rows are sorted up to that column, so this runs in O(N^2 * M) time.