5709. Maximum Ascending Subarray Sum

My Submissions (/contest/weekly-contest-233/problems/maximum-ascending-subarray-sum/submissions/) Back to Contest (/contest/weekly-contest-233/)

Given an array of positive integers `nums`, return the *maximum possible sum of an **ascending** subarray in* `nums`.

A subarray is defined as a contiguous sequence of numbers in an array.

A subarray `[numsl, numsl+1, ..., numsr-1, numsr]` is **ascending** if for all `i` where `l <= i < r`, $nums_i < nums_{i+1}$. Note that a subarray of size `1` is **ascending**.

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Easy

Example 1:

```
Input: nums = [10,20,30,5,10,50]
Output: 65
Explanation: [5,10,50] is the ascending subarray with the maximum sum of 65.
```

Example 2:

```
Input: nums = [10,20,30,40,50]
Output: 150
Explanation: [10,20,30,40,50] is the ascending subarray with the maximum sum of 150.
```

Example 3:

```
Input: nums = [12,17,15,13,10,11,12]
Output: 33
Explanation: [10,11,12] is the ascending subarray with the maximum sum of 33.
```

Example 4:

```
Input: nums = [100,10,1]
Output: 100
```

Constraints:

- `1 <= nums.length <= 100`
- `1 <= nums[i] <= 100`

JavaScript

```
const mx = Math.max;
const maxAscendingSum = (a) => {
    let n = a.length;
    let res = 0;
    for (let i = 0; i < n; i++) {
        for (let j = i; j < n; j++) {
            let sub = a.slice(i, j + 1);
            if (isAscending(sub)) {
                let sum = sub.reduce((x, y) => x + y);
                res = mx(res, sum);
            }
        }
    }
    return res;
};


const isAscending = (arr) => {
    return arr.every((x, i) => {
        return i === 0 || x > arr[i - 1];
    });
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/470386196/) | More Details > (/submissions/detail/470386196/)

Share your acceptance!