Minimum Number Of Removals To Make Mountain Array

solved by	Senan
Platform	LeetCode
difficulty	Hard
# Serial	1671
tags	Binary Search
language	C++
solved on	@30/10/2024
link	https://leetcode.com/problems/minimum-number-of-removals-to-make-mountain-array/description/
Completion	☑

Intuition

We want to transform the given array into a "mountain array" (an array with elements first strictly increasing, then strictly decreasing) by removing the minimum number of elements. To achieve this, we can leverage the concept of Longest Increasing Subsequence (LIS) and Longest Decreasing Subsequence (LDS). By finding the LIS up to each index and the LDS from each index, we can determine the largest mountain array that can be formed ending and starting at any index. The minimum elements to remove will then be the difference between the array size and this largest possible mountain.

Approach

1. **Compute LIS:** Calculate the longest increasing subsequence up to each index in the array. Store these values in a vector `lis`.
2. **Compute LDS:** Reverse the array and calculate the longest increasing subsequence (LIS) again, which effectively gives the LDS for the original array. Reverse the result to align with the original indices and store in `lds`.
3. **Calculate Minimum Removals:** For each index, if `lis[i] > 1` and `lds[i] > 1`, it can form a peak in a mountain. Calculate the potential mountain length at each index and compute the minimum removals by subtracting the maximum mountain length from the total array size.

Complexity

Time Complexity:

- **LIS Calculation: O(n^2)** due to the double loop for each index to compute LIS.
- **Overall Complexity: O(n^2)** as we calculate LIS and LDS independently and then perform an **O(n)** comparison.

Space Complexity:

- **Space: O(n)** for the LIS and LDS vectors.

Code

```
class Solution {
    vector<int> LIS(vector<int> &arr) {
        vector<int> answer(arr.size(), 1);
        for (int i = 1; i < arr.size(); i++) {
            for (int j = i - 1; j >= 0; j--) {
                if (arr[i] > arr[j]) {
                    answer[i] = max(answer[i], answer[j] + 1);
                }
            }
        }
        return answer;
    }
public:
    int minimumMountainRemovals(vector<int>& nums) {
        auto lis = LIS(nums);
        reverse(nums.begin(), nums.end());
        auto lds = LIS(nums);
        reverse(nums.begin(), nums.end());

        int minRemoval = INT_MAX;
        for (int i = 1; i < nums.size() - 1; i++) {
            if (lis[i] == 1 || lds[i] == 1) continue;
            int mountainSize = nums.size() - (lis[i] + lds[i] - 1);
            minRemoval = min(minRemoval, mountainSize);
        }

        return minRemoval;
    }
};
```