LeetCode 1526. 形成目标数组的子数组最少增加次数

Feijoa_Li

一、 题目描述

给定一个整数数组 `target` 和一个初始数组 `initial`，`initial` 数组与 `target` 数组维度相同且初始所有元素为0。每次操作可以选择任意子数组，并将子数组中每个元素增加1。返回从 `initial` 得到 `target` 的最少操作次数。

二、 解题思路

本题可以使用动态规划的思想解决。$dp[i]$ 表示将前 $i+1$ 个元素从目标值减少到 0 所需的最少操作次数。注意，从目标数组减少到 0 的操作次数与从 0 增加到目标数组的操作次数相同。

初始状态：

$$dp[0] = target[0]$$

状态转移方程： 对于 $i \geq 1$，考虑 $target[i]$ 与 $target[i-1]$ 的关系：

- 如果 $target[i] \leq target[i-1]$，则在减少前 i 个元素的过程中,对于 $target[i-1]$ 必然执行了 $target[i-1]$ 次操作(因为 $target[i-1] == 0$)

  因此我们大可将每一步包含着 $target[i-1]$ 的操作再往右延伸一格,影响到 $target[i]$ 上.

  所以此时:

  $$dp[i] = dp[i-1]$$

- 如果 $target[i] > target[i-1]$ 我们依旧用上面的思考方式,但此时，$target[i-1]$ 的操作无法使 $target[i]$ 减少到 0 ，还需要额外增加 $target[i] - target[i-1]$ 次操作.

  所以此时:

  $$dp[i] = dp[i-1] + (target[i] - target[i-1])$$

因此，整合两个式子得出状态转移方程为：

$$dp[i] = dp[i-1] + \max(0, target[i] - target[i-1])$$

所以最终答案为 $dp[n-1]$，其中 n 是数组长度。
由于 $dp[i]$ 只依赖于 $dp[i-1]$，我们可以使用一个变量来滚动计算，节省空间。

三、 代码实现

```
int cnt = target[0]; // 初始化 dp[0]
for (int i = 1; i < target.size(); i++) {
    // 状态转移: dp[i] = dp[i-1] + max(0, target[i] - target[i-1])
    cnt += max(0, target[i] - target[i - 1]);
}
return cnt;
```

四、 复杂度分析

- **时间复杂度:** $O(n)$，其中 n 是数组长度。只需遍历数组一次。
- **空间复杂度:** $O(1)$，仅使用常数额外空间。