Desired Solution

We begin by reformulating the problem. Let $pre_i = \sum_{j=1}^{i} d_j$. For each position $1 \le i \le n$, define $S(i)$ as the number of stars Ian can collect if he starts at position i. Then: $S(i) = |\{j \ge i \mid t_j \le pre_j - pre_{i-1}\}|$

There are two neat approaches to solving this problem efficiently, both based on computing all values of $S(i)$.

Full Solution 1

We modify the condition, note that $t_j \le pre_j - pre_{i-1} \iff pre_{i-1} \le pre_j - t_j$. We define $a_j = pre_j - t_j$, then the formula for $S(i)$ becomes $S(i) = |\{j \ge i \mid a_j \ge pre_{i-1}\}|$. We compute $S(i)$ by enumerating i from right to left. We maintain a multiset that stores all current values of a_j for $j \ge i$. For each i, we first insert a_i, then we compute how many elements in the multiset are $\ge pre_{i-1}$.

This yields an $O(N \log N)$ solution using either a Binary Indexed Tree (Fenwick Tree) or an ordered set (e.g., `__gnu_pbds::tree` in C++). Time Complexity: $O(N \log N)$

Full Solution 2

Using the same simplification $S(i) = |\{j \ge i \mid a_j \ge pre_{i-1}\}|$, we now observe that if there exists $j < i$ such that $pre_{j-1} \le pre_{i-1}$, then starting from i is never optimal. This means that we only need to consider starting positions i such that pre_{i-1} is the minimum in the prefix, we call these valid starting positions.

We compute $S(i)$ for these valid starting positions by sorting all a_j in non-increasing order. Then we iterate over all valid starting positions i in increasing order. We maintain a set A of indices $j \ge i$ such that $a_j \ge pre_{i-1}$. We remove any indices $j < i$ in A, since they are no longer valid. After the cleanup, $S(i) = |A|$. We can use a priority queue or a multiset to maintain and update A dynamically. Time Complexity: $O(N \log N)$

Subtask 1 $(N \le 2000)$

These subtasks are intended for brute-force solutions. For each starting point i, simulate and count the amount of stars Ian can collect. Time Complexity:$O(N^2)$

Subtask 2 (at most 20 negative d_i's)

This subtask supports Solution 2. Since at most 20 prefix sums may decrease (due to negative values), the number of valid starting positions is at most 20. By identifying all valid starting positions and simulate each in $O(N)$, we obtian a solution to pass this subtask. Time Complexity: $O(N|\{i : P_i < 0\}|)$

Subtask 3 (no positive d_i)

Since all $d_i \le 0$, pre_i is a non-increasing sequence. We can use binary search to count the number of indices which satisfy the inequality $a_j \ge pre_{i-1}$ as described in the full solutions. Time Complexity $O(N \log N)$

Subtask 4

This subtask is intended to encourage a simplified version of Solution 1. Since the number of stars collected is at most 20, we can calculate the amount of elements in the multiset that are $\ge pre_{i-1}$ by maintain the top 20 largest elements in a fixed-size array, using `std::set` and `prev()` function, or use a `priority_queue` and pop elements greater than pre_{i-1} until none remain. Time Complexity: $O(N \log N + N \times \text{ans})$ or $O(N \log N \times \text{ans})$ depending on which data structure is used