

Best Runner

First, let's analyze the subgroups where the main idea from the general solution is not used:

Subgroup 2: The first runner will always win: he must move one lane to the left after each lane (as long as this is possible). This way, he will run the most lanes, as he will have run on the shortest lanes.

Subgroup 3: This can be solved using dynamic programming. $dp[i][j]$ (where i is the current lane of the runner, and j is the remaining time) equals the maximum number of lanes that can be run starting from such a situation.

Subgroup 5: The runner who starts on the lane with the minimum length always wins, as he can run the maximum number of times on the shortest lane.

For the other subgroups, we need to understand what the optimal route of the runner will look like. Let's say he starts on lane i .

- He must finish on the shortest lane he has been on (let's denote it as j). If he stayed on the shortest lane and ran on it until the end of the time, he would have run at least as many lanes as in any other scenario.
- He must also transition from i to j as quickly as possible, and then only run on j . This way, he will run the maximum number of lanes among all scenarios.

Let's say we have a runner starting on lane i and ending on lane j . We define the number of lanes he will run. Without loss of generality, let $i < j$. Then he will spend $a_i + a_{i+1} + \dots + a_{j-1}$ time to reach lane j , and then he will run on j for the remaining time. Thus, he will run $j - i + \lfloor \frac{T - (a_i + a_{i+1} + \dots + a_{j-1})}{a_j} \rfloor$ lanes (if the time from lane i to lane j does not exceed T). If we maintain prefix sums of the array a , we can compute the number of lanes in $O(1)$.

This is enough to solve **subgroup 1**: for each starting position, iterate through all ending positions and find the maximum number of lanes.

Subgroup 4: here we need to iterate only through those lanes that are strictly shorter than all lanes between the starting lane and it as ending positions. Since the lengths of the lanes do not exceed 20, for each starting position, we will consider no more than 20 ending lanes to the left and right. To quickly find the nearest lane to the left (or right) that is shorter than the current one, we can precompute them in advance.

Subgroup 6: we just need to apply a small trick. We will instead find for each ending position j the runner who will run the maximum number of lanes and finish on lane j . It can be shown that this will either be the nearest runner to the left of the lane, or the nearest runner to the right (or a runner starting on this lane, if such exists) — other runners will take more time to reach j (if j is the optimal ending position). Thus, we need to iterate through no more than 2 starting lanes for each ending lane.