A Light Inconvenience (light)

by Fabian Gundlach

Subtask 1. Only one call to *leave* per testcase

For this subtask, it will be convenient to reindex the performers, numbering them from *right to left*, starting at 1. (Any call to *join* or *leave* will then result in an index shift). Moreover, we will always only remember the sequence $1 = f_1 < f_2 < \cdots$ of performers whose torches are on fire.

Let's think about this problem greedily: what condition do we need to put on the f_i's to be able to survive the next act?

- We can *always* handle the next call to *join* with $t_a = p_a$: we can simply shift everything to the right by p_a (i.e. keeping the previous f_i's), adding further fires on the left if desired.
- We can handle a single call to *leave* by announcing $t_a = p_a$ if and only if $f_i \leq 2f_{i-1} + 1$ *and* the leftmost torch is on fire: the first condition is necessary since having all performers starting at f_i leave will have $p_a = f_i$, and we have to light the torch of performer $f_i + 1$; the second condition is necessary to handle all but the leftmost performer leaving.

  On the other hand, it is not hard to check that this also sufficient to handle any call to *leave*: if we just announce p_a, this is enough to light the torch of the rightmost performer remaining on stage.

This suggests taking $f_i = 2^{i-1}$ (plus the leftmost performer) before the first call to *leave*, and then e.g. extinguishing all torches except the rightmost one. For any call to *join* after that, we can simply shift everything to the right again without adding further flaming torches. This will never have more than $\log_2 N + 1$ torches on fire at the same time, which easily fits into the limit specified in the statement.

Subtask 2. $N \leq 700$

Let us index the torches as in the task statement again. It suffices to announce $f_1 = 1, f_2 = 6, f_2 = 11, \dots, f_k = n$ (where n is the number of performers currently in line) at the end of any act; this will in fact only require $t_a = 4$ for any *leave* or $t_a = 4p_a$ for *join*. The maximum number of flaming torches is $\frac{1}{5}N + 1$.

Subtask 3. $N \leq 5\,000$

The previous solution does not make use of the fact that we are allowed to announce large numbers whenever p_a is sufficiently large, giving us much more freedom.

Intuitively, we should exploit this by having much more fires near the right end of the line. Consider the following straegy: fix a number K and split the performers into blocks of length K, starting on the left. We will always guarantee that the leftmost performer in each block (i.e. performers $1, K + 1, 2K + 1, \dots$) holds a flaming torch. Moreover, every torch in the last block should be on fire.

Again any call to *join* would be trivial to handle. If we have a call to *leave*, however, we would run into a problem to uphold our invariant when all $0 < k \leq K$ performers of the last block leave, as we will need $t_a = K - 1$ to fill up the last block. To avoid this issue, we will always hold some additional torches in the penultimate block: namely, we will have the leftmost ℓ torches on fire, where ℓ is the number of torches missing in the last block. As a result:

- If $j < k$ performers leave, we just announce $t_a = j$ to add fires to the penultimate block.
- If $k \leq j < K$ performers leave, announcing $t_a = j$ is now enough to fill up the new final block (and it is never a problem to add the fires on the left of the new penultimate block).

- If at least K performers leave, we are free to do whatever we want in any block anyhow.

Again, the invariant is also trivial to uphold in *join*.

The maximum number of torches on fires is $N/K + K$, so taking $K = 75$ suffices to solve this subtask.

Subtask 4. $N \le 25\,000$

This is exactly as in the previous subtask, except that we now exploit that we are allowed to announce up to $t_a = 5p_a$, which allows us to increase the distance between the torches by a factor of 5.

Subtask 5. $N \le 100\,000$

The solution to this subtask interpolates between the solution to Subtask 3 presented above and the first full solution discussed below to achieve $t_a = p_a$ with at most $O(\sqrt[3]{N})$ fires.

Subtask 6. $N \le 500\,000$

Again, this can be solved by spreading out the fires from the previous solution by a factor of 5.

Subtask 7. No further constraints.

There are at least two different approaches that yield full score:

First full solution. Having blocks of a fixed size will of course only ever get us that far, so let us combine this with the idea of having the distances between the fires grow exponentially as we walk from right to left. A first attempt to implement this strategy might be to always have for any k a flaming torch at the largest multiple of 2^k that is $\le n$. Unfortunately, this does not work for similar reasons as in the third subtask: when n itself is a power of 2, then reducing n by 1 will still incur a large cost. Fortunately, also the solution to this problem is similar (although the analysis is somewhat harder this time): we enforce flaming torches at the *two* largest multiples of 2^k for each k, and up to one additional torch in the penultimate block of size 2^k to light the third largest multiple of 2^{k-1} if this becomes necessary during *leave*.

Second full solution. Let us analyze the solution to the very first subtask more closely, for which we again flip our indexing. We have seen that we can handle a single call to *leave* if and only if $f_i \le 2f_{i-1} + 1$ for all i and the leftmost torch is on fire. Moreover, we can trivially maintain this invariant for any call to *join*.

The crucial observation now is that we can also uphold this invariant for any call to *leave* by announcing p_a and then selecting fires as follows: assume we have already selected new torches $1 = g_1 < g_2 < \dots < g_i$ satisfying the above; to select the next torch, we have to show that one of the torches $g_i + 1, \dots, 2g_i + 1$ is on fire. This is obviously true if one of these torches is the leftmost one or if $g_i < p_a$. In the remaining case, consider the smallest j with $f_{j+1} - 2p_a > 2g_i + 1$ (note that $f_{j+1} - 2p_a$ is the rightmost torch lit by the performer that was previously at position f_{j+1}). Then $f_j - 2p_a < 2g_i + 1$ by minimality, but also

$$f_j - p_a \ge \frac{f_{j+1} - 1}{2} - p_a > g_i.$$

Thus, at least one of the torches $f_j - 2p_a, \dots, f_j - p_a$ (all of which are currently on fire) lies in $\{g_i+1, \dots, 2g_i+1\}$ as claimed.

We now have to be slightly clever in selecting the g_i's in order to ensure that our program runs in time and that there aren't too many torches on fire.

- One way to do so is to always pick the largest g_{i+1} possible, which guarantees that $g_{i+2} > 2g_i + 1$ and hence ensures that we only ever have at most $2 \log_2 N + \epsilon$ torches on fire.
- More generally, you can come up with any reasonably time-efficient way to construct the g_i's and then 'sparsify' your result using the previous observation. For example, one uniform way that avoids the above case distinction is to start with the values $f_j - p_a$ plus the rightmost performer and then double each of them until you would run past one of the previously constructed elements.

Partial solutions. While we only described the full solutions above, there are various ways to obtain partial scores on the last subtask. One way to do so is to use larger bases in the above construction, i.e. only guaranteeing that $f_i \le \alpha f_{i-1} + 1$ for some $\alpha > 1$. This becomes relevant if one uses less clever ways to sparsify the fires, which would otherwise result in too many torches being on fire.

In addition, there are several ad-hoc constructions.