Problem B. Railway

Subtask 1

Loop all triples (train j from Zurich, train k from Lugano, Tunnel i) and check if the two trains crash in the tunnel. We can do this check in $\mathcal{O}(1)$ time as shown below, so this takes $\mathcal{O}(nmt)$ time in total.

A train from Zurich departing at time c_j enters the tunnel i at time $c_j + a_i$, and leaves it at time $c_j + b_i$. A train from Lugano to Zurich departing at time d_k enters tunnel i at time $d_k + (s - b_i)$ and leaves it at time $d_k + (s - a_i)$.

Since the two trains travel in opposite directions, they crash in the tunnel if they are both strictly inside the tunnel at some point in time. Thus, it suffices to check if the **real** intervals $(c_j + a_i, c_j + b_i)$ and $(d_k + (s - b_i), d_k + (s - a_i))$ intersect.

Subtask 2

Loop over tunnels, and for every tunnel, compute the union A of the intervals of time when trains from Zurich are strictly inside the tunnel, and the union B of times trains from Lugano are strictly inside it. If $A \cap B \neq \emptyset$, there is a crash in the tunnel. We can check if the intersection is empty in $\mathcal{O}(n + m)$ time, for $\mathcal{O}((n + m)t)$ total work.

Subtasks 3 and 4

Remember the condition for trains j and k to crash inside tunnel i: $(c_j + a_i, c_j + b_i)$ and $(d_k + (s - b_i), d_k + (s - a_i))$ must intersect. This holds if $c_j + a_i < d_k + (s - a_i)$ and $d_k + (s - b_i) < c_j + b_i$. Moving terms, we get $c_j - d_k < s - 2a_i$ and $s - 2b_i < c_j - d_k$. Thus, we must have $c_j - d_k \in (s - 2b_i, s - 2a_i)$.

This gives a $\mathcal{O}(nm \log t)$ solution: let T be a sorted vector of the tunnel intervals $(s - 2b_i, s - 2a_i)$. Loop all pairs (train j from Zurich, train k from Lugano), and binary search the first interval for which $s - 2b_i < c_j - d_k$, then check if $c_j - d_k < s - 2a_i$. If true, the two trains crash in the tunnel. Otherwise, the trains do not crash.

The difference between subtasks 3 and 4 is that, as long as s, c_j and d_k are all even, all trains meet at integer coordinates. This does not change the solution, but can make implementing some approaches easier. You can also guarantee that s, c_j and d_k are even by multiplying all input values by 2.