UTRKA

We process event in order of arrival and maintain the state of the driver that the event concerns.

One complete state of a driver consists of:

· the number of checkpoints the driver has passed

· which of the D events is the last legal event about the driver

At the start of the race all of the drivers' states are (0, -1).

Now, for each event of the form "driver X has passed checkpoint Y":

· if Y is the checkpoint we expect driver X to pass next:

  increase the number of checkpoints the driver has passed

  update the driver's last event

All that remains now is to sort the drivers according to their states. Driver A is in front of driver B iff:

· he has passed more checkpoints

· they've passed the same number of checkpoints, but driver A passed his last checkpoint earlier than driver B passed his

LABUDOVI

We simulate the melting of ice each day and check when the swans can meet. To do this efficiently, we use two breadth first searches (BFS) which run alternately, one of which melts ice and another one which explores squares the first swan can reach.

The overall time complexity is $O(R \cdot S)$.

NAGIBNI

For each vertex V, we can calculate the optimal solution for the subtree rooted in V for both of these cases:

· there is a line connecting vertex V to its parent

· there is no such line

For the first case, the line connecting V to its parent can either end in V or pass through V.

If the line ends in V then all of V's children aren't connected to V and we can use dynamic programming to calculate the entire cost.

If the line passes through V then exactly one of V's children is connected to V. We are best off picking a vertex for which the difference between the solutions for the two cases is maximal i.e. from which we can gain most.

For the second case, a line can start in V and end in one of its children's subtrees, or start in one of its children's subtrees, pass through V and end in another of its children's subtrees.

If the line starts in V, we need to pick a subtree in which the line will end, that is, we pick one subtree which will be connected to its parent (vertex V). Again, this will be the vertex for which the difference between the solutions for the two cases is maximal.

If the line passes through V, we need to pick two vertices that will be connected to V. These two are of course, the two with the maximal difference as before.

The naive implementation of this algorithm takes $O(N^2)$ time, but carefully implemented it can even be linear in the number of vertices.