Grid race (tier 2)

Your task is to write a bot that will participate in a multiplayer car racing environment.

Environment description

The environment models a racetrack that is discretised in a (rectangular) grid, consisting of *cells*. Some of the cells are marked as goal cells, and the task of the agents is to reach the goal cells as soon as possible. The game is played by one or more more agents, and they start from one of the start positions. The winner is the one who reaches one of the goal positions with the lowest number of steps (or, equivalently, in the lowest number of iterations; the distance travelled by the agent is not relevant). Agents take steps one after the other in a fixed order.

There are cells marked as "wall" cells, which are impenetrable (and there is a penalty for trying to move there, see below).

The agents have velocity, and can only accelerate or decelerate by a small amount. Formally, the acceleration is added to the velocity of the agent, and that will be its new velocity. The new velocity vector is then added to its position, resulting in its new position:

$$v_{t+1} = v_t + a_t \tag{1}$$
$$x_{t+1} = x_t + v_{t+1}, \tag{2}$$

where x, v and a are the position, velocity and acceleration vectors, respectively ($x, v, a \in \mathbb{R}^2$), and

$$a_t = \left(a_t^{(r)}, a_t^{(c)}\right), \quad a_t^{(r)}, a_t^{(c)} \in -1, 0, 1, \tag{3}$$

that is, acceleration can be at most 1 in either direction. At the start of the race, velocity is zero.

If an agent takes a step that would move it onto a wall cell, or outside the map, or to a position occupied by another agent, it is blocked for five rounds, after which it is restarted from its last valid position with zero speed. The agent innocent in a collision continues without any penalty.

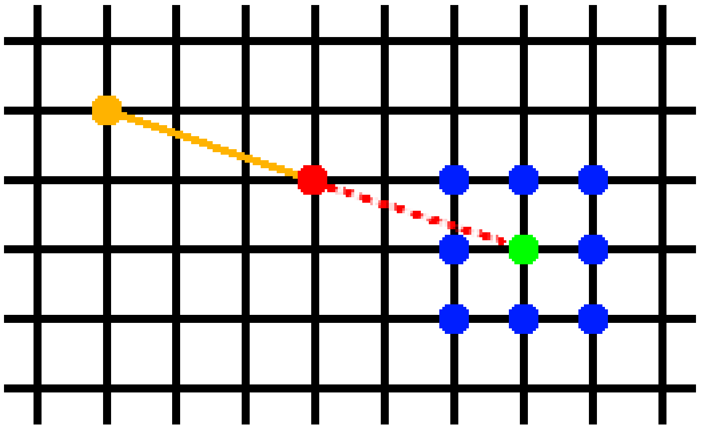

Figure 1: Visualisation of agent step. The orange and red dots mark the previous and the current positions respectively. The agent can move to the green dot or to one of its neighbors, marked by blue dots.

A crucial difference from the previous tier is that the agent only sees a small area around it. More specifically, there is a visibility radius $R \in \mathbb{N}^+$, and on its turn, the agent observes only cells that are at most R distance from its location (measured in Euclidian distance). All other cells in the observation will be marked as not visible.

To go with this, the visualisation script can now visualise this limited visibility (as a fog over the rest of the map). To use this, pass `--visibility_radius <R>` to the script, where `<R>` is the visibility radius value. The fog can then be enabled or disabled by pressing the "F" key.

Communication protocol

After starting, the bot should read the global environment parameters from the standard input. The first line contains the height H and the width W of the map, the number of players N and the visibility radius R, in that order, separated by spaces.

On its turn, the bot receives the current observation on its standard input. The first line is four integers separated by spaces: the location and the velocity of the bot, respectively (both are pairs of numbers: row and column coordinates, in that order). The next N lines contain 2-2 integers separated by spaces: the (row and column) coordinates for all the players.[1] The order of the players are the same throughout the task.

The last $2R + 1$ lines of an observation represents the area of the map currently visible to the agent. Each line contains $2R + 1$ integers, separated by spaces; these represent the cells, as follows:

[1]Note that since no two players can be on the same cell, you can identify your own line by comparing the locations to your own location.

- 0: empty cell,
- -1: wall cell (everything outside the map is considered to be wall),
- 1: start cell,
- 3: cell is not visible,
- 100: goal cell.

On its turn, the bot must output two integers separated by a space (and terminated by an end of line): the acceleration in the row and column directions, respectively. The acceleration values must be one of -1, 0 or 1.

At the end of the task, the bot receives the string “~~~END~~~” instead of an observation. Upon receiving this line, the bot should exit gracefully.