

Grid race (tier 2)

Your task is to write a bot that will participate in a multiagent 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 one of the goal cells in as few steps as possible. One or more agents can participate in a race, and initially they are placed in one of the start positions. The winner is the one who reaches one of the goal positions in 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, which remains the same throughout the race.

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

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

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

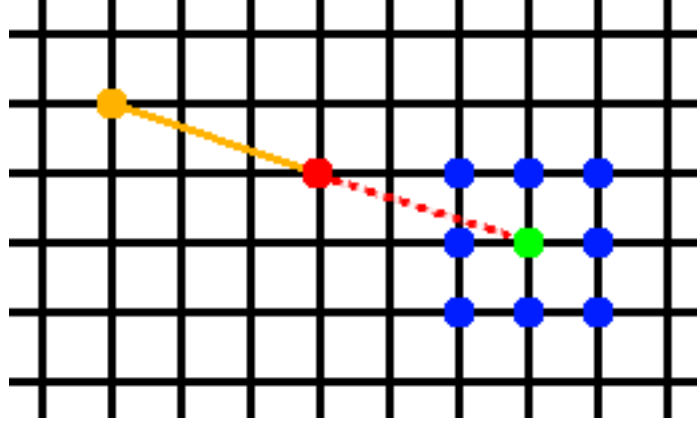


Figure 1: Visualisation of a step. The orange and the red dots mark the previous and the current positions, respectively. The agent can move to the green dot or to one of its neighbors, marked by blue dots.

restarted from its last valid position with zero speed. The agent innocent in a collision continues without any penalty.

A crucial difference from the previous tier is that the agent only sees a limited 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 a distance of at most R 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 the argument `--visibility_radius <R>` to the script, where `<R>` is the visibility radius value. The fog can then be disabled and re-enabled 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 agents 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 contains four integers separated by spaces: the location and the velocity of the agent, 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 row and column indices start from 0.

¹Note that since no two agents can be on the same cell, agents can identify their own line by comparing its own location to the list of locations received.

The last $2R + 1$ lines of an observation represent 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:

- 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 race, the bot receives the string “`~~~END~~~`” instead of an observation. Upon receiving this line, the bot should exit gracefully.

Scoring

If the agent reaches one of the goal states within the turn limit, its score is the turn index it does so (that is, because turns are indexed from 0 , the number of steps to reach the goal state minus one).

Otherwise, if the agent did not reach the goal, ran into a runtime error or even failed to connect, its score is the turn limit plus one.