Maximilian Bragazzi Ihrén maxbr431@student.liu.se

TDDC17 – Lab 1 Explanation

Task 1

General idea

The general idea I had for this task was this:

1. Get the agent back to its home position
2. "Snake" (in this case, move right-down one row-left-down one row-right, etc) the entire area.
3. Get the agent back home again, and end the simulation.

And in each step the agent is looking for and, if it is there, sucking up dirt

The initial problem I then ran into was, of course, how to get the agent back home. It wasn't a very difficult problem though, seeing as I just had to tell it to go north until it hit the upper wall, then west until it hit home. I separated this into the function "goHome()".

Once home, the agent turns east and starts moving forward until it bumps into something, at which point it turns to the south, moves forward one, then turns in the opposite direction of the direction it was just heading (which would be west the first time). This became the function "searchWorld()".

After "searchWorld()" reported it was done, I could just simply call on "goHome()" again, until the agent was at home, at which point I'd tell the simulation it was done.

Stored data

The data I stored in the agent for this task was fairly simple:

- A goal variable, to keep track of what the agent currently was trying to do (plus a few "final int"-variables to give name to those goals).
- Which direction the agent is supposed to be heading in
- An array of actions to be performed

The last point being how I solved the movement system. If the agent is, for example, standing east and has to walk south, I push an action to turn right and one to move forward onto the array. Only if the array is empty do I call on the function that is currently supposed to be called ("goHome()" or "searchWorld()"), which in turn pushes more things onto the array I execute.

Things to be done better

There are of course always things we can do better. Instead of going home, I could have started "searchWorld()" right away, and only gotten the unknown squares I potentially missed at the end.

However, for such a simple function, I think I didn't overcomplicate things and left it fairly simple.

Maximilian Bragazzi Ihrén maxbr431@student.liu.se

Task 2

General idea

The general idea stayed the same as from task 1. However, since task 2 introduced obstacles, I faced a whole new set of problems.

I decided early to follow the old labyrinth solving "formula" of always holding one hand on a wall in order to get around an object. This lead to the functions "getAroundObstacle()" and "setGetAroundObstacle()". "setGetAroundObstacle()", as it sounds, only initiates all the variables "getAroundObstacle()", so all the "real" magic happens in the latter function.

"getAroundObstacle()" functions in such a way that it finds the next "free" point (anything that isn't a wall) beyond the obstacle right in front of the agent, and tries to move to it using the "labyrinth formula" mentioned above. It first checks if the agent have moved from its original position, and if the agent has, it checks whether or not it has found an "exit location" (which would be the point the agent is trying to get to). After that, it checks if the agent bumped into something, and if it did it checks whether or not the point its trying to get to is unavailable (by being surrounded by walls, or being a wall itself), at which point it moves on to the next "free" point. Otherwise, it just follows the standard "labyrinth formula".

Another big issue was also that the agent can't know if the obstacle it bumped into when "snaking" was the outer edge, or just a block in the way. Therefore the agent, after coming home the first time, runs around the perimeter of the map to find the outer edges, which then are stored in variables.

After being able to avoid obstacles, and knowing exactly how far the agent has to go, it can easily traverse the map.

Stored data

The data I stored in the agent for this task ended up a bit more:

- The things from task 1
- A variable to know which row the agent is currently trying to clean
- The max X and Y coordinates the agent can reach on the map
- The max and min Y coordinates the agent can reach on the current row of the map
- Which direction the agent is facing while trying to get to the current row (east or west)
- The point the agent is trying to get to
- The point the agent originated from when starting "getAroundObstacle()"
- Which "hand" to hold against the wall (left or right)
- Whether or not the agent is trying to get around an obstacle at the moment
- Which direction the agent is trying to get around the obstacle to

Things to be done better

Well, you can do small things to improve the "algorithm", such as always checking if a row is clean before you start cleaning it, but honestly the entire algorithm is fairly overcomplex, and should probably be replaced with normal pathfinding.