Laboratory 1: Intelligent Agents

TASK 3: Explanation

a) First thoughts & failures:

At first, we have tried to implement several ideas that we had. One of those was to make the agent go forward until it bumps into a wall and then to turn right or left, a bit like the reactive agent. Even if it was working to a certain extent, it was almost as efficient as the random agent.

We although decided to push the idea, by looking into the tiles in front and at the sides of the agent. If the state of the tile was unknown, then we would visit the tile. This algorithm showed much better results than the first, but we were stuck. Without adding obstacles, it could suck all the dirt, but it could not go home.

b) Choice of the Depth-First Search Algorithm:

To overcome our difficulties, we thought that our algorithm looked a lot like the Depth-First Algorithm. We just had to implement a mechanism for the queue. We thought about using a *stack* for keeping a track of each direction the agent is moving forward to.

However, a single stack was not enough as it only guaranteed us to come back to our starting position and we want to finish at home. We then introduced a second stack, that would start to be filled out once we visited once the home, hence we would know the path to come back to it.

c) Implementation of the Depth-First Search Algorithm:

The algorithm works in the following manner:

1. The state of the tile in front of the agent is looked, if it is unknown, the agent goes to this tile and the direction of the agent is added to the first stack.
   If the tile in front of the agent is a wall, in case of a bump, the last direction will be removed from the stack as the agent did not effectively move forward.
2. If the state of the tile in front of the agent is not unknown, hence the agent has already been to this tile, the side tiles will be analyzed. If at least one of them has not been visited, then the agent turns in that direction.
3. If the front and both sides tiles have been visited, then the agent turns back to the last direction in his stack, which is removed once the agent has the same direction. Then, it analyzes if one of the surrounding tiles has not been visited. If so, it goes there.
4. When the agent goes to the home tile, the agent will start to fill a second stack, which purpose is to guide the agent home when it has returned at his starting point, therefore when his first stack is empty.

d) Improvements and alternative algorithms:

Depending on when the agent finds his home and his starting point, the algorithm is more or less effective. Nevertheless, it always sucks all the dust and goes back to the home within the authorized number of iterations. For improvements, the mechanism to return to the home is sometimes not very efficient. But this one has the advantage to always work and being arguably efficient.

As for other algorithms, we have thought about the Breadth-First Search, which is similar in complexity, but we chose quite naturally, as explained, to the DFS.