TDDC17 ARTIFICIAL INTELLIGENCE

Lab 1: Intelligent Agents

Robin Andersson (roban591)
Lawrence Thanakumar Rajappa (lawra776)

September 18, 2019

1 Aim

This first lab explains how to implement the concept of intelligent agents. The agent is a Vacuum cleaner which is moving in a square grid. Each tile in that grid is assigned with a state such as clean, dirty or obstacle. The agent can perform basic actions such as moving forward, turning right, turning left and sucking the dirt. The aim of this lab is to write an algorithm to automate the cleaning process. No assumptions regarding the world dimension or the dirt and the obstacles positions can be made. The vacuum cleaner has to be completely autonomous.

2 Data Structures

Below are the data structures that are being used to solve the two tasks. The final data structure path is only used in task 2.

- Tile: A class used to hold the x and y coordinates of a tile.
- action_list: A LinkedList of Actions (left, right, forward) used by the agent to navigate to the next tile.
- path: A LinkedList that corresponds to how to get from the current tile to the destination tile.

3 Task 1

The first task is to suck dirt wherever it is present without any obstacles. This is implemented in the steps given below.

1. Specify that the first row and column consists of walls (index 0 in each direction).
2. Find the closest unknown tile using the euclidean distance formula.
3. Generate the necessary actions to get to that tile by first generating the actions for the y-direction and then the x-direction.
4. If a tile is dirty then it will be cleaned, otherwise the agent will continue moving toward the goal.
5. If the agent is facing south and hits a bump (obstacle) then we know that the south wall is found, so all indexes from that y-position are marked as walls.
6. If the agent is facing east and hits a bump (obstacle) then we know that the east wall is found, so all indexes from that x-position are marked as walls.
7. When the goal tile is reached the agent returns to step 2, if no unknown tile is found, then the agent returns to the home position by generating actions similar to step 3.
8. When the agent is back at the home position it shuts down.

This solution was chosen since we only would need to bump into a wall twice to find the limits of the world and using the euclidean distance gives the shortest route from one tile to another. However it could be more efficient to for instance search the bottom half of the world first and then search the top half since that would reduce the number of tiles that we have to traverse to reach the home position. In our case, we usually end up around the bottom right corner which gives the longest path to the home position.

4 Task 2

The second task is to also to suck dirt but with obstacles. This is implemented in the steps given below.

1. Using breadth-first search the agent finds a destination tile and generates a path to that tile consisting of all tiles that must be passed along the way.

2. Then it takes the closest tile from the path and generates the necessary actions to get to that tile.

3. If a tile is dirty then it will be cleaned, otherwise the agent will continue moving toward the goal.

4. When the goal tile is reached the agent returns to step 1, if no unknown tile is found, then the agent returns to the home position by specifying in step 1 that it wants to find the home position instead of an unknown position.

5. When the home position is reached the agent shuts down.

We chose breadth-first search since it is cost effective and because we did not have implement backtracking like depth-first search. An improvement over breadth-first search would be to use Djikstra's algorithm and add a cost for turning as well, where a simple left or right turn has a cost of 2 and a u-turn has a cost of 3.