Martin Gustafsson (margu424) & Axel Gard (axega554)

TDDC17 lab1 report

Our agent is based on the concept of depth first search and once we had it working for task 1 it was also working for task 2. As we used depth first search we used a stack to store all the previous moves we had made in order to be able to backtrack our moves. But in order to backtrack our moves we need to move backwards, which is not possible given the actions available to the agent. We therefore needed a way to turn around easily and we decided to use a queue for this that would store moves and execute them in concession. We used this queue to store the moves “turn right”, “turn right” and “move forward” as the way to backtrack a “move forward” move previously executed. But this resulted in the agent facing the wrong way so we used a boolean variable to know if the agent was facing backwards or not. We also stored another boolean variable “allCleaned” so that the agent would know when all squares were cleaned.

The main decision loop of the agent is that for as long as there is an unknown square in front of it then we move forward. If the square in front is already cleared or is a wall we instead look at the square to our right, if that is unknown we then turn right. If both the square in front and the square to the right are cleared we check the square to the left and if that is unknown we turn left. And if all three squares are cleared we backtrack one move as we have reached a dead end in terms of tiles to clean.

In order to backtrack we use our stack of moves to look at our last move and if that move was “turn right” the move “turn left” was made and if it was “turn left” the move “turn right” was made instead. If the previous move made was ”move forwards” we would first look if the agent was facing backwards because if it was we could just do the move “move forwards”, if the agent however was not facing backwards we used our queue of moves to execute the moves “turn right”, “turn right” and “move forward” in order so that we would then be facing backwards and have backtracked one step.

Being backwards came with some problems to the way we handled turning right and left when we saw an unknown square. We needed to make a special case for when we were backwards so that if we saw an empty square on the right we would turn right but add the move “turn left” to our stack as that would be the move executed if we were not facing backwards. And in the same way if we saw an empty square on the left and we were backwards we would turn left but add “turn right” to the stack of moves. In the case that we were backwards and found the square in front unknown we needed to pop the previous move of the stack and if it was a “turn left” we added “turn right” to the stack and if it was “turn right” we added “turn left” to the stack, “move forward” is not possible as a previous move as that would mean the square in front is already cleared.

When the stack of moves is empty we have cleaned all squares and we set “allCleaned” to true. We then go through our world view and set all squares that are cleared to unknown and use our depth first search again to search all squares again. But this time with a new condition that if we find our home square we shut down as we have cleaned all squares and are back home.

There are of course alternative solutions to the vacuum agent problem. We started out exploring some of these alternatives. Of course there are more than these solutions that could have been used but these were the once that we explored. The first agent we built for task 1 was to always take a right turn

when the agent hits a wall or an already cleaned tile. This solution however had problems with different starting locations. An example would be that if the agent did not start in a corner the agent would miss some tiles.

The second implementation was that the agent would walk to the top right corner and from this corner it would then move in a row by row pattern to ensure that we had cleaned all the dust. However we realized that this approach would not be sufficient for task 2.
We decided to go back to the first build of the agent however we changed the rules in order to handle obstacles and a bad starting location. The agent would now try to keep a wall to it's left and look to see if the obstacle had a way to pass by it's left side. This agent was also able to backtrack in the sense that if it was stuck with three walls surrounding it then the agent was allowed to move to a square already cleaned.

The problem with this agent was that it would end up cleaning itself into a corner and spinarund in this corner having no way to go and leaving some times half the map uncleaned. So we add more tiles for it to "backtrack". This would improve the performance however the agent would just end up in bigger 2x2 circles then just spinning. So we changed the agent to be able to "backtrack" for so long as it could only see cleaned and wall tiles. This would sometimes solve task 2 however most of the time the agent would in the end just walk around the edge of the map and never look in the middle. This is when we built the final agent using depth first search and utilizing proper backtracking methods and this method could always solve task 2.

To find home as previously mentioned we just keep using the same depth first search algorithm. However a more effective version would have been to use the A* algorithm to find home however this felt unnecessary as that would only make the code more complex and with marginal improvements in performance. We could have used breadth first search instead of depth for the whole agent however this would have meant that the agent would need to make a lot of extra moves and turns. So we believe that the agent is more effective utilizing a depth first search algorithm.