Lab 1 report

Filjo653 & Thela038

Our agent works the same in both task 1 and task 2. In every step, the agent looks for the closest unexplored tile in its internal state. When such a tile is found, the agent tries to move towards it. In this way, the agent will explore all tiles in the world. More precisely, the decision-loop consist of the following steps:

1. Check if the current tile is in the home position, if it is, then save its location for later.
2. Update our internal representation of the world, based on the new data since the last step.
3. If we are currently standing on top of dirt, perform a “suck” action, otherwise continue to step 4.
4. Search through our internal representation of the world in order to find the closest unexplored tile and return the shortest path leading from our current tile to the unexplored tile.
5. If we can’t find any path to an unexplored tile, then that means we have explored the entire world and should go back to our home tile. Since we saved the home position in step 1, we can repeat the same search but search for the shortest path to our home tile instead of to an unexplored tile.
6. From the path constructed in step 4 or 5, find the direction that we should be facing in order to follow the path.
7. If we are already facing in the direction found in step 6, perform a “move forward” action. Otherwise, continue to step 8.
8. If we are not facing in the direction we want, perform a “rotate right” action.

The data that is stored in the agent is:

- The internal representation of the world, or the “state”. This contains a map of all known tiles in the world, as well as the agents' position and direction.
- Home position. Since the already provided implementation of the world could overwrite the home position in the state, we added this variable to store the home position instead.

We chose this solution because it was a good tradeoff between being relatively efficient, and not being too hard to implement. We chose a BFS algorithm to search the state for the next tile to explore because it was the optimal algorithm to find the path to the closest unexplored tile.