COS30019 - Introduction to Artificial Intelligence
Tutorial Problems Week 2

Task 1:

Let us examine the rationality of various vacuum-cleaner agent functions.

A. Show that the simple vacuum-cleaner agent function described in the lecture (under the given assumptions) is indeed rational.
B. Describe a rational agent function for the modified performance measure that deducts one point for each movement. Does the corresponding agent program require internal state?
C. Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn?

Task 2: Develop a PEAS (performance measure, environment, actuators, sensors) description of the task environment for:

a) Robot soccer player
b) Internet book-shopping agent

Task 3: For each agent type above, characterize the properties of the task environment and select a suitable agent design.

Task 4: Referring to the utility-based agents described in the lecture, both the performance measure and the utility function measure how well an agent is doing. Explain the difference between the two.

Programming Task – Vacuum-cleaning agent:

For this task, you are required to program a vacuum-cleaner agent. The agent operates in a 2D environment and has the sensors to detect its location and the status of the cell it is in (`DIRTY`/`CLEAN`). The agent's next action can be obtained by calling the public method `next()`. In this week, you have two programming tasks:

P1. Create an agent program, specifically the `next()` method, by taking as input a sequence of percepts including the current percepts and returning the next action the agent would execute. The behavior of this agent can be as simple as randomly choosing one action among the set of all possible actions (`LEFT, RIGHT, UP, DOWN, SUCK, NO-OP`).

P2. Given an agent program (could be the one you created in **P1**), write a main program that takes in as input a floor map, an agent's initial location, and an integer representing the lifetime of the agent (the number of time steps the agent has to operate). The program then returns the agent's performance score for that particular problem.

The floor map will be a text file having the following format:

```
[1,1] status [1,2] status [1,3] status ..... [1,N] status
[2,1] status [2,2] status [2,3] status ..... [2,N] status
......
[M,1] status [M,2] status [M,3] status ....[M,N] status
```

where status = {dirty, clean, nil}

if status = nil, there is no room in that cell (i.e., agent cannot enter that cell)

The agent’s location: [x,y]

You can now experiment with different agent designs and test their performance ☺