Artificial Intelligence Project 1

In this project we are asked to solve exercises 2.8 and 2.11 from the textbook (Rusell, S. and Norwig. Artificial Intelligence: A Modern Approach. Prentice Hall. Third Edition, 2009.). What each problem asks, and relevant information is provided for each. A requirements.txt is included with the project. All dependencies must be installed using pip for the program to execute correctly.

Problem 2.8 asks the following:

2.8 Implement a performance-measuring environment simulator for the vacuum-cleaner world depicted in Figure 2.2 and specified on page 38. Your implementation should be modular so that the sensors, actuators, and environment characteristics (size, shape, dirt placement, etc.) can be changed easily. (*Note:* for some choices of programming language and operating system there are already implementations in the online code repository.)

Some relevant information referenced in the exercise specification includes:

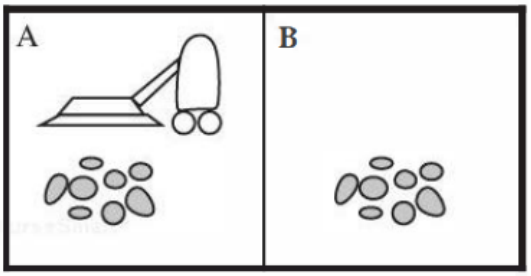


Figure 2.2 A vacuum-cleaner world with just two locations.

Percept sequence	Action
[*A*, *Clean*]	*Right*
[*A*, *Dirty*]	*Suck*
[*B*, *Clean*]	*Left*
[*B*, *Dirty*]	*Suck*
[*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Consider the simple vacuum-cleaner agent that cleans a square if it is dirty and moves to the other square if not; this is the agent function tabulated in Figure 2.3. Is this a rational agent? That depends! First, we need to say what the performance measure is, what is known about the environment, and what sensors and actuators the agent has. Let us assume the following:

- The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
- The "geography" of the environment is known *a priori* (Figure 2.2) but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The *Left* and *Right* actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
- The only available actions are *Left*, *Right*, and *Suck*.
- The agent correctly perceives its location and whether that location contains dirt.

We claim that *under these circumstances* the agent is indeed rational; its expected performance is at least as high as any other agent's. Exercise 2.2 asks you to prove this.

One can see easily that the same agent would be irrational under different circumstances. For example, once all the dirt is cleaned up, the agent will oscillate needlessly back and forth; if the performance measure includes a penalty of one point for each movement left or right, the agent will fare poorly. A better agent for this case would do nothing once it is sure that all the squares are clean. If clean squares can become dirty again, the agent should occasionally check and re-clean them if needed. If the geography of the environment is unknown, the agent will need to explore it rather than stick to squares *A* and *B*. Exercise 2.2 asks you to design agents for these cases.

From the description found in page 38, I found two different performance measures:

- Performance measure 1: Only the cleanliness of the environment is considered. Implemented in exercise2_8.py by adding 1 performance point for each Clean square on each step,
- Performance measure 2: The number of times the vacuum moves should result in a penalty in performance. In exercise2_8.py this is implemented by subtracting a performance point each time the agent moves. However, a point is still awarded for each clean square in the environment.

I coded three cases in exercise2_8.py:

- Case 1: Performance measure 1 is used to measure performance of the agent described in page 38.
- Case 2: Performance measure 2 is used with the same performance agent described in page 38.
- Case 3: Performance measure 3 is used with an agent code found in the online repository. The agent acts based on a percept sequence table. The maximum length of percept

sequence included is three. Therefore, the agent would move at most 3 times and would still achieve a state in which all squares are clean.

The three cases were simulated using the code in *exercise2_8*.py and yielded the following results. The simulation runs 1000 time steps in the environment.

n	Case 1	Case 2	Case 3
0	1996	1000	1997
1	2000	998	1995
2	1998	999	1998
3	2000	998	1998
4	2000	1000	1997
5	1998	1000	1998
6	1996	1000	1999
7	1998	998	1995
8	2000	1000	1998
9	1996	998	1995

As can be observed from the results, when using a simple agent – which keeps moving regardless of whether or not the environment is clean – performance measure 1 yields favorable results. The same agent's performance significantly decreases (by about half) when performance measure 2 is used. However, when a more "rational agent" is used with performance measure 2, the results are again favorable.

Exercise 2.11 asks the following:

2.11 Consider a modified version of the vacuum environment in Exercise 2.8, in which the geography of the environment—its extent, boundaries, and obstacles—is unknown, as is the initial dirt configuration. (The agent can go *Up* and *Down* as well as *Left* and *Right*.)

a. Can a simple reflex agent be perfectly rational for this environment? Explain.

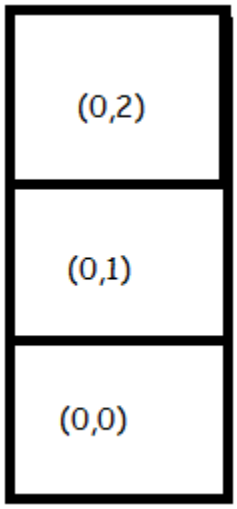
b. Can a simple reflex agent with a *randomized* agent function outperform a simple reflex agent? Design such an agent and measure its performance on several environments.

c. Can you design an environment in which your randomized agent will perform poorly? Show your results.

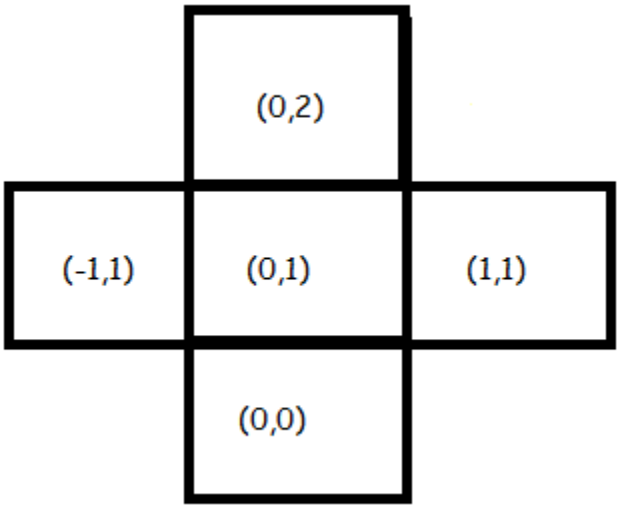
d. Can a reflex agent with state outperform a simple reflex agent? Design such an agent and measure its performance on several environments. Can you design a rational agent of this type?

For this exercise, I suggest the following environments:

- Environment A



- Environment B



a. Can a simple reflective agent be perfectly rational for this environment? Explain.

   The answer to this question really depends on the boundaries, layout of the environment, reflexes the agent has, and initial position of the agent.

   For example, if a simple reflex agent if coded to “Suck” if current square is dirty. If the current square is “Clean” it tries to move “Up”, “Right”, “Down”, or “Left” in that order. As long as there is no obstacle or boundary impeding its movement, it moves.

The described agent might perform well in environment A if it starts at position (0, 0). However, it would find itself in an endless loop going from (0,1) to (0,2) and from (0,2) to (0,1). If the number of times the agent visits each square is considered in the performance measure, it would perform poorly.

The same agent would do an even worse job in environment B since it would never visit squares (-1,1) and (1,1).

With these examples in mind, I consider that, while the agent could perform well in very specific cases, in most, it would perform poorly. Therefore, in my opinion a simple reflective agent would not be *perfectly* rational for this environment.

b. Can a simple reflex agent with a *randomized* agent function outperform a simple reflex agent? Design such an agent and measure its performance on several environments.

Yes, a randomized agent may outperform a simple reflex agent. For example, in environment B, the agent will successfully visit all squares in the environment where, the simple reflex agent might miss it.

In exercise2_11.py I implement a random agent for Environment A and Environment B. The performance measure adds 10 for "sucking dirt" and subtracts 1 for each move. The environment is tested with 20 steps.

n	environment a	environment b
0	15	13
1	7	-3
2	11	14
3	-7	4
4	-7	-1
5	11	-4
6	2	24
7	14	18
8	5	2
9	13	24
10	4	10
11	10	6
12	24	12
13	-10	5
14	16	22
15	8	33

c. Can you design an environment in which your randomized agent will perform poorly? Show your results.

   I believe in a long environment where it is only possible to move horizontally or vertically, the random agent may not perform as well when compared to a simple reflex agent.
   I am attempting to finish the implementation before the deadline. If the results are not available, I was unable to finish on time.

d. Can a reflex agent with state outperform a simple reflex agent? Design such an agent and measure its performance on several environments. Can you design a rational agent of this type?

   I believe a reflex agent with state can outperform a simple reflex agent. It is possible I am unable to finish on time. If I don't, for now, I leave this hypothesis.