Jay Trivedi

1. Chapter 2, Ex. 2.3 (18 points, 2 points for each sub-question)

For each of the following assertions, say whether it is true or false and support your answer with examples or counterexamples where appropriate.

a. **An agent that senses only partial information about the state cannot be perfectly rational.**
   False. An agent is called perfectly rational if it maximizes performance measure based on percept. So, even if only partial information can be sensed but that leads to maximize performance measure, the agent is still rational.

b. **There exist task environments in which no pure reflex agent can behave rationally.**
   True. A pure reflex agent does action based just on current percept. Hence if an environment requires the agent to also consider previous percepts, for making decisions, the agent won't behave rationally. For instance, a maze solver agent would often get stuck in an infinite loop if it doesn't keep track of the paths it has already explored.

c. **There exists a task environment in which every agent is rational.**
   True. An agent is called rational if it maximizes performance measure based on percept. So, in a task environment if all the actions result in same performance measure every agent would get same maximization and hence would be rational. For instance, in case of vacuum cleaner agent if there is just one location and performance is measured by if it has dirt or not any agent with dirt sensor will give perfect answer each time.

d. **The input to an agent program is the same as the input to the agent function.**
   False. An agent program just takes current percept as an input while the agent function takes the whole percept sequence as an input.

e. **Every agent function is implementable by some program/machine combination.**
   False. For an agent function to be implemented it needs both program and architecture. A software program can be implemented if it tractable i.e. the time required to complete it does not increase exponentially with the number of inputs, but if that is not the case the agent function can't be implemented by any program/machine combination. Another example is to try solving halting problem with Turing machine, as a paper by Alan Turing proves the problem is undecidable over Turing machine

f. **Suppose an agent selects its action uniformly at random from the set of possible actions. There exists a deterministic task environment in which this agent is rational.**
   True. As the environment is deterministic the next state can be fully determined by its current state and action, thus if we have an environment with single state doing a random action will still get us back to the same state making agent rational.

g. **It is possible for a given agent to be perfectly rational in two distinct task environments.**
   True. Rationality of an agent depends on the performance measure determined given task

environments, thus, even if two task environments are different but the agent maximizes performance in both it is perfectly rational. For instance, in a maze search if we add a locked down region where it is not possible for agent reach, no matter how many such regions we add the agent will be perfectly rational in all environments given it was rational before.

h. Every agent is rational in an unobservable environment.
False. An agent can become rational in an unobservable environment by using built in knowledge to maximize performance measures but still there will be some actions which will not to do that. For instance, a simple agent (with no sensor) who just keeps on moving in one direction in a maze is not rational.

i. A perfectly rational poker-playing agent never loses.
False. A perfectly rational playing agent will try to maximize payoff but will still lose if other agent has better cards. In case if we make the same agent play with itself one version would lose.

2. Chapter 2, Ex. 2.9 (22 points)

Notes: - Implementation should be in Python - Output the sequence of actions for the environment and initial location described in Figure 2

Implement a simple reflex agent for the vacuum environment in Exercise 2.8. Run the environment with this agent for all possible initial dirt configurations and agent locations. Record the performance score for each configuration and the overall average score.

The accompanied code gives two options to user one where he can define an environment himself and other where default environment of Fig. 2 from book is taken. The agent action is defined based only on the default environment and performance is measure by the number of cleaned square at each time step. The performance is averaged out over 1000 steps and printed.

3. Chapter 3, Ex. 3.2 (20 points, 5 pts for each subquestion)

Notes: - Each answer should be explained

Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

a. Formulate this problem. How large is the state space?

Formulating a problem requires following 6 things:

States: There can be infinitely many states for the robot, as its position is continuous variable and not discrete. Also, the state space will depend on the size of maze as well as the obstacles it encounters during solving.

Initial State: It is very well defined as robot is facing north and it is in the center of the maze

Actions: The robot as 4 actions at a given time; namely changing direction in any 3 of the other direction except the one facing or continue for distance d in the same direction.

Transition model: Depending on the state of the robot and the action it takes the model will define its new state. So, in the start if the robot moves for distance d in the same direction the transition model will give the state of robot facing north and distance d from the center.

Goal test: This test will check if the robot has come out of the maze.

Path Cost: Assuming every rotation costs r and moving distance d costs p, the path cost = no. of rotation * r+ no. of times it moved d * p

b. In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?

States: As the turn is possible only at the intersection of corridors the robot position no longer remains continuous. At every intersection there are 4 possibilities for robot to go to next intersection, thus if x is the number of intersection 4x is the number of states possible.

Initial State: It is same as the question above, center of the maze facing north

Actions: Same as above the robot has 4 actions at a given time i.e. change to any of 3 directions or continue till next intersection in the same direction

Transition Model: Move to the next intersection in the facing direction or change the direction

Goal Test: It checks whether the robot has come out of the maze.

Path Cost: Assuming every rotation costs r and moving to next intersection costs p, the path cost = no. of rotation * r+ no. of intersections crossed * p.

c. From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now?

States: As the turn is possible only at the turning point and assuming the turning point allows to change direction to a specific direction only, the number of states is equal to the number of turning points (t) plus 4 as at the start we can choose a direction (i.e. t+4). And as orientation gets defined by the turn, we no longer need to keep track of orientation of robot.

State: It is same as the question above, center of the maze facing north

Actions: Only one action is possible at every state except at start where 4 actions are possible

Transition Model: Move to the next turning point

Goal Test: It checks whether the robot has come out of the maze.

Path Cost: Assuming every rotation costs r and moving next turning point costs p, the path cost = no. of rotation * r+ no. of turning points * p.

d. In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

i. We assume the robot is in flat 2D world
ii. The robot can move only in one of the 4 directions
iii. The robot moves accurately
iv. Robot doesn't go out of power
v. The environment doesn't change

4. Chapter 3, Ex. 3.3 (20 points, 5 pts for each subquestion) Notes: - Each answer should be explained

Suppose two friends live in different cities on a map, such as the Romania map shown in Figure 3.2. On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance d (i, j) between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

a. Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)

States: All possible city pairs (i,j)

Initial State: City pair (i,j) where two friends are

Actions: Friend in city i going to adjacent city x and friend in city j going to adjacent city y

Transition Model: Travelling given distance d transits one guy from city i to x or city j to y.

Goal Test: Both are at the same city at the same time.

Path Cost: Cost for one step is max (distance (i, x), distance (j, y))

b. Let D (i, j) be the straight-line distance between cities i and j. Which of the following heuristic functions are admissible? (i) D (i, j); (ii) 2 · D (i, j); (iii) D (i, j)/2.

A heuristic is admissible if it never over estimates the cost to reach the goal. The true distance between the cities is D(i,j) and as the distance decrease twice in one step as both of them are moving it function D(i,j) and 2. D(i,j) overestimates the cost while D(i,j)/2 does not. Hence only D(i,j)/2 is admissible.

c. Are there completely connected maps for which no solution exists?

The map where there are just two cities connected by a road both friends will keep on swapping cities and will never meet for every other case fully connected map has a solution.

d. Are there maps in which all solutions require one friend to visit the same city twice?

There no maps where all solutions require one friend to visit the same city twice. Some maps may require some solutions in which one friend has to visit a city twice but there exists a pair of city with solution that doesn’t require one friend to visit the same city twice.