Daniel McDonough (dmcdonough)
Artificial Intelligence (CS 534)
Assignment 1

1) Chapter 2, Ex. 2.3

a) False. An agent that senses only partial information about the state **can** be perfectly rational. This is because an agent is defined perfectly rational when it makes the best moves possible given any known amount of the environment.
b) True. There exist task environments in which no pure reflex agent can behave rationally is true do to the fact that any task that involves memory will cause a reflex agent to fail because reflex agents do not hold percept history.
c) True. A task environment in which every agent is rational would be one where all actions result in the same outcome.
d) False. The input to an agent program is not the same as the input to the agent function. An agent function takes the percept history whereas an agent program takes the current percept
e) False. Not every agent function is implementable by some program/machine combination. An example would be an implementation of the halting problem, where an agent function would be responsible for determining other agent function outcomes given a history. Now if a function agent that included such a function agent as dictated prior but also provided the inverse of any result. Now if that agent function was acted upon itself if would always get the result incorrect. Therefore the original function agent is impossible to implement.
f) True. 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. Only in the case in which there is only one option.
g) True. It is possible for a given agent to be perfectly rational in two distinct task environments. In the case in which a discrete and continuous task environment can both be done, because the only point of matter is time of computation.
h) False. Not every agent is rational in an unobservable environment. For example in the case in which not all options

in the decision tree are “visible” the rational option may not be available to the agent.

i) False. A perfectly rational poker-playing agent will lose, due to the luck based nature of the game.

2.9)

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.)

2.9 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. (See Readme Examples)

3) 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?

The initial state (1), actions (4 turns + N distance), and transition model (RESULT(s, a)) implicitly define the state space. Where the transition model returns the state that results from doing action a in state s. If maze has N^2 big then there are $4*N^2$ states that the robot can be in.

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?

The initial state (1), actions (N distance + 4(J T-intersections) + 4(K 4-way intersections), and transition model (RESULT(s, a)) implicitly define the state space. Where the transition model returns the state that results from doing action a in state s. If maze has N^2 big the robot will only go in one direction until hitting a wall. We then have

to count the J T-intersections and multiply by the number of actions (4, assuming you can still go forward into a wall) and same with K 4-way intersections but with 4 actions. But we must not double count the selection of going forward at the intersections because N^2 area is still counting for potentially 2 actions (forward and backward). So we must add back the intersection of the two "probabilities". Therefore we have $((2*(N^2-(J+K)) + (4(J+K))$.

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?

We no longer need to keep track of the robots orientation due to the coupling of the actions. Therefore the state is always GO or 1 node in the state space, even if there is no guarantee for the goal condition to be reached.

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.

3 initial simplifications we made were:

- Robot can only move in 4 directions
- Robot can keep track of distance
- Robot can sense walls

4) 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.)

This search problem can be classified as a subset of a bidirectional search problem due to the fact that two ends are known. By having 2 breadth first searches (BFS) going at once then getting the optimal path is much faster than running a single BFS. However if the two are running parallel to meet in the middle (MIM), **BFS is not always the optimal path.** This is because of two truths; the total heuristic time for MIM is always smaller than BFS no matter what, and the min(BFS) path does not always equal the same min(MIM) route if the number of paths is odd (aka one travel path is not done in parallel). Therefore we can define P as some set of paths, X as some path in that set. X_N as the number of Nodes in the path and X_K as the total heuristic of path X. We can also have A_k and B_k which dictate the heuristic for any given connection. So we can classify the optimal path for friend A and B to be the some path X where if X_N is even take BFS(P) where the min(any(A_k) – any(B_k)) are done each step. If X_N is odd, then we can simply take the min(sum(X_k/2)) again making sure that min(any(A_k) – any(B_k)) are done each step to optimize parallelism.

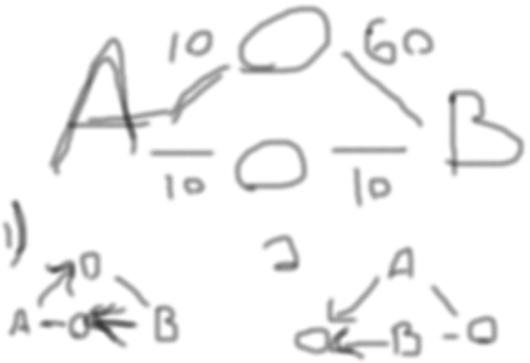
b) Let D(i, j) be the straight-line distance between cities i and j. Which of the following heuristic functions are admissible?
   I. D(i, j); NO
   II. 2 · D(i, j); NO
   III. D(i, j)/2. Yes, because movement is done in parallel.

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

   Assuming that a friend does not always have to move, there is no map without a solution. Otherwise, any map of 2 nodes would have no solution.

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

   Assuming that a friend does not always have to move, there is no map without a friend requiring to revisit a city. Otherwise, there is a possible map in which a friend must revisit a city to have the optimal path, but not otherwise an Example below shoes that friend A must revisit their initial city if they cannot wait.


A
10
O
60
B
10
O
10
1)
O
A
B
2
A
B
O