KI Project B

Martijn Dekker 6013368
Egor Dmitriev 6100120
Cody Bloemhard 6231888

December 5, 2018

1 Exercise 1

1.a ex1-a

The game state holds all the data that represents the game in the current time frame. bla bla bla.

1.b ex1-b

The agent state holds all data that describes an agent in the current time frame. bla bla bla.

1.c ex1-c

- A: II, Stack: Pile of dishes
- B: III, Queue: Roller coaster waiting line
- C: I, PriorityQueue: Emergency room waiting line

2 Exercise 2

2.a

3 Exercise 3

3.a ex3-a

That means that if there exist a path to the goal you are searching for, it will return that goal.

3.b ex3-b

It is complete because the algorithm will keep searching till it found the path to the goal location or till there are no more possible paths to stroll, which haven't been visited before. This would happen when the stack is empty and that can only occur if it has tried every possible path.

3.c ex3-c

It will not always be the least cost solution. This is because it tries out a path and continues it till it ends, which can lead to the goal location. This first path it finds does not have to be the path with the least cost. And because it stops after finding a path it doesn't always find the least cost solution.

3.d ex3-d

Yes, the order is as we expected. Pacman does not visit all the explored squares on his way to the goal. This is because the first path he explores may or may not be a dead end. If this is the case then he will not go there, because his objective is to reach the goal.

4 Exercise 4

4.a ex4-a

It is complete because the algorithm will keep searching till it found the path to the goal location or till there are no more possible paths to stroll, which haven't been visited before.

4.b ex4-b

Yes, because every step pacman can take is equal in cost the algorithm needs to find the path with the least steps. And because it looks into every path of depth k before looking into paths of depth $>$ k it will always find the least cost solution.

4.c ex4-c

Yes, it returns a solution, because the algorithm was implemented so well that it does not matter what problem it is used on.

5 Exercise 5

5.a ex5-a

For the first agent: it's intended behaviour is to find the shortest path to the goal, because there is no reason not to cause there are no obstacles, therefore every path costs the same. It is achieved by sorting the currently visible paths based on their cost so far, with the cheapest sorted at the front. That way the algoritm will constantly pick the cheapest path to further explore. And because every path costs the same it basically works like breadth first search. For the second agent: it's intended behaviour is to take the east route because that path has some food on it. The food makes for a cheaper path than a normal path. It is achieved almost the same way as the first agent, only now not every path costs the same. The path with the food costs less, so that path will be explored first instead of the other paths. For the third agent: it's intended behaviour is to take the most west path, because the other paths contain ghosts which drive

up the cost of a path. This is because the paths with the ghosts have a high chance of death. It is achieved the same way as with the other agents, only now the paths with the ghosts have a high cost and thus they will be sorted at the end. This way the algorithm explores the path without ghosts first.

When the algorithm finds a path to the goal it stops looking because it chose the cheapest path at all time, therefore the first path must be the cheapest.

5.b ex5-b

6 Exercise 6

7 Exercise 7

7.a

Both files contain a class which can be extended to implement an agent. Both of these classes implement a different base class / interface.
The one in file *valueIterationAgents.py* is meant to be implemented for agents which estimate the Q-values and values for an environment using a Markov Decision Process, which is done before acting. The other in file *qlearningAgents.py* is meant to be implemented for agents that estimate Q-values from policies rather than from a model.

7.b

- In reinforcement learning transition probabilities are unknown. Which is one of the reasons why MDP can't be used in such cases.
- In reinforcement learning transition rewards are unknown. These are learned with experience and can differ every iteration.

7.c

GridWorld implementation differs in how terminal state and rewards is handled. In contrary to slides the rewards are set per grid cell basis. Which means a cell can give a reward and not a be a terminal state (although only action possible in a cell with a reward is a terminal state).
Terminal states do not belong to any cell, nor do they give any rewards. They are implemented as actions. These are only to indicate when the iteration should break. In the slides they behave as a cell, but a final one.

7.d

"A terminal state is a state that once reached causes all further action of the agent to cease."
Source: `http://burlap.cs.brown.edu/tutorials/bd/p1.html`.

Both slides and implementation use this definition. SInce no actions can be taken once a terminal state is reached.