Name: Tasbiha Asim
Course: Artificial Intelligence
Assignment 3
Program: SE4GD

--
Part a) Modeling the MDP as an infinite horizon MDP: the agent, once he starts to cook successfully, never ends, and it remains in an absorbing state. Using the above problem description, answer the following questions:

a) Provide a concise description of the states of the MDP. How many states are in this MDP? (i.e. what is |S|).

The |S| contains 128 states. Each state is considered to be a tuple with the following configuration:
- Index 1: x coordinate
- Index 2: y coordinate
- Index 3: 0 or 1 indicating whether the agent has the beater or not
- Index 4: 0 or 1 indicating which recipe the agent is supposed to make

For example; the tuple (7,3,0,1) represents that the agent is at (7,3) cell of the grid, it does not have the egg beater and has to make pudding.

Given that the dimensions of the grid are 8 x 4 (omitting the wall at x = 5), we can conclude that the total number of states will be;

dimensions of grid x length of tuple = 32 x 4 = 128

The states of MDP are expressed in |S| as following;

|S| = S1, S2, …, S32, .., S128

b) Provide a concise description of the actions of the MDP. How many actions are in this MDP? (i.e. what is |A|).

The agent can go up, down, right and left within the grid so the actions set will contain the following;

|A| = Left, Right, Up, Down

States and Actions:
https://docs.google.com/spreadsheets/d/1PLCXY7ReABCSe6AhrkXM5f1paOO44B2GhtiijSWp0_I/edit?usp=sharing

c) What is the dimensionality of the transition function P?

The dimensionality of P is calculated by multiplying the number of states by number of actions. Therefore,
Number of states = 128
Number of actions = 4
Dimensionality = 128 x 4 = 512

d) Report the transition function P for any state s and action a in a tabular format.

Link:
https://docs.google.com/spreadsheets/d/1IofgrrLpxELQrCpbK_gceJmiVXbCr8hbAFKAQ12oemM/edit?usp=sharing

e) Describe a reward function R : S × A × S and a value of γ that will lead to an optimal policy.

The reward value for each state *s* going to next state *s'* for action *a* is reported in the excel sheet which can be accessed from the link below;

Link:
https://docs.google.com/spreadsheets/d/1lMtRhBD0jbsIAlBxV9M1etvxhiBGujXJEGITBVypbTo/edit?usp=sharing

Description:
The following principles are taken into account while assigning reward values to the states;
- The agent cannot change recipes during one episode. The recipe asked for at the start of the episode is the one that we should aim to finish by the end of the episode.
- The reward values assigned for egg beater state and goal states (scrambler or pudding state) are dependent on whether the agent has the egg beater or not. So if the agent does not have the egg beater than the reward value at (1,3) or (8,3) is 100 which is the maximum reward. Then when it acquires the egg beater, the maximum reward value i.e. 100 is awarded to the pudding maker or scrambler.
- The empty cells are rewarded values in the range of 50 - 95 based on how far they are from the maximum reward state. So if the agent has the egg beater then the rewards at empty cells are calculated based on how far the agent is from the goal state.
- When the agent acquires the egg beater, the state changes from either (1,3,0,0) to (1,3,1,0) or from (8,3,0,1) to (8,3,1,1) depending on which recipe it is supposed to make.

- If the agent is to hit a wall for some state *s* at action *a* then the reward value for that (state, action, nexts_state) is said to be -100.
- The agent is said to be at the wrong side of the grid when it is supposed to make the recipe that can be made by the tool on the other side. For example, if the agent is at the right of the grid i.e. when x>=6 but it is supposed to make scrambled eggs, then the agent is at the wrong side of the grid. Therefore, the reward values for states on the other side will get negative values.
- When the agent is at the wrong side of the grid, its goal is to reach the gate states. Thus the reward values calculated are relative to the gate state. And the gate state will have the highest value in the wrong grid.

The rewards are calculated for agent's position on the grid based on the following conditions;

- **When the agent does not have the egg beater and the recipe is scrambled eggs:**
  The states in this case are represented as; (x coordinate, y coordinate, 0, 0) where the 0 at third index means that the agent has no egg beater and 0 at fourth index means that agent is supposed to scramble eggs.
  The agent should aim to reach the egg beater and therefore, the value of reward at (1,3,0,0) is 95. The reward values for empty cells are then calculated relative to the egg beater state which means that if the agent is closer to the egg beater cell, for example if it is at (1,2,0,0) then the value for reward will be much higher i.e. 90 because then the agent only has to take one step upwards to get to the egg beater state. However, if the cell is much farther away from the egg beater state eg; at (1,1,0,0) then the reward at this cell is 55 which is comparatively lesser than 90.
  However, if the agent is at the right side of the grid (which is the wrong side since the agent is supposed to make scrambled eggs) then the cells are assigned negative rewards. When the agent is at the wrong side of the grid, the gate state will have a higher value of reward because the agent should try to get out of the wrong grid. For example in this case, if the agent is at (7,1) cell of the grid then it is at the wrong side of the grid and hence will have a negative reward value for (7,1,0,0) which is -75 in this case. If the agent is further away from the gate 2, then the reward will decrease even more, for example, at (9,1,0,0) the reward value is -85.

When the agent goes to egg beater, the state changes from (1,2,0,0) to (1,3,1,0) and then the rewards will be calculated as follows;

- **When the agent has the egg beater and the recipe is scrambled eggs:**
  The states in this case are represented as; (x coordinate, y coordinate, 1, 0) where the 1 at third index means that the agent has the egg beater and 0 at fourth index means that agent is supposed to scramble eggs.
  The goal state in this case is the fry pan and therefore, the value of reward at (1,4,0,0) is 100. The reward values for empty cells are then calculated relative to the goal state which

means that if the agent is closer to the goal cell. For example; if it is at (2,4,0,0) then the value for reward will be much higher i.e. 90 because then the agent only has to take one step left to get to the goal state (1,4,0,0). However, if the cell is much farther away from the goal state eg; at (1,1,0,0) then the reward at this cell is 55 which is comparatively lesser than 90.
However, if the agent is at the right side of the grid (which is the wrong side since the agent is supposed to make scrambled eggs) then the cells are assigned negative rewards. When the agent is at the wrong side of the grid, the gate state (6,1) will have a higher value of reward (-70). For example in this case, if the agent is at (7,1) cell of the grid then it is at the wrong side of the grid and hence will have a negative reward value for (7,1,1,0) which is -75 in this case. If the agent is further away from gate 2, then the reward will decrease even more, for example, at (9,1,1,0) the reward value is -85.

- **When the agent does not have the egg beater and the recipe is pudding:**
  The states in this case are represented as; (x coordinate, y coordinate, 0, 1) where the 0 at third index means that the agent does not have the egg beater and 1 at fourth index means that agent is supposed to make pudding.
  The highest reward in this case will be at egg beater and therefore, the value of reward at (8,3,0,1) is 100. The reward values for empty cells are then calculated relative to the egg beater state. For example; if it is at (8,2,0,1) then the value for reward will be much higher i.e. 95 because then the agent only has to take one step left to get to the egg beater state (8,3,0,1). However, if the cell is much farther away from the egg beater state eg; at (6,2,0,1) then the reward at this cell is 65 which is comparatively lesser than 90.
  However, if the agent is at the left side of the grid (which is the wrong side since the agent is supposed to make pudding) then the cells are assigned negative rewards. When the agent is at the wrong side of the grid, the gate state will have a higher value of reward. For example in this case, if the agent is at (4,2) cell of the grid then it is at the wrong side of the grid and hence will have a negative reward value for (4,2,0,1) which is -70 in this case. If the agent is further away from gate 1, then the reward will decrease even more, for example, at (1,4,0,1) the reward value is -97.

When the agent goes to egg beater, the state changes from (1,2,0,1) to (1,3,1,1) and then the rewards will be calculated as follows;

- **When the agent does not have the egg beater and the recipe is pudding:**
  The states in this case are represented as; (x coordinate, y coordinate, 1, 1) where the 1 at third index means that the agent has the egg beater and 1 at fourth index means that agent is supposed to make pudding.
  The goal state in this case is the pudding maker and therefore, the value of reward at (8,4,1,1) is 100. The reward values for empty cells are then calculated relative to the goal

state which means that if the agent is closer to the goal cell. For example; if it is at (9,4,1,1) then the value for reward will be much higher i.e. 90 because then the agent only has to take one step left to get to the goal state (9,3,1,1). However, if the cell is much farther away from the goal state eg; at (6,2,1,1) then the reward at this cell is 65 which is comparatively lesser than 90.
However, if the agent is at the left side of the grid (which is the wrong side since the agent is supposed to make pudding) then the cells are assigned negative rewards. When the agent is at the wrong side of the grid, the gate state will have a higher value of reward. For example in this case, if the agent is at (4,2) cell of the grid then it is at the wrong side of the grid and hence will have a negative reward value for (4,2,1,1) which is -70 in this case. If the agent is further away from gate 1, then the reward will decrease even more, for example, at (1,4,1,1) the reward value is -97.

A value of γ that will lead to an optimal policy: close to 0 because this is not a model-free scenario and the rewards are strictly defined. As a result, the agent does not need to explore long-term rewards.

f) Does $\gamma \in (0, 1)$ affect the optimal policy in this case? Explain why.

High discount value i.e. closer to 1 means that the agent will focus on long-term rewards and a smaller discount factor value means that the agent will focus more on immediate rewards.

Since we have a strong reward table, our agent will possess a good understanding of which actions it is supposed to take therefore a higher discount factor will not have a significant impact. The optimal value can be set to close to 0 such as 0.1 or 0.2.

It makes more sense to use a high discount factor for Q-learning problems where the environment is not known to the agent and it is a model-free scenario.

After assigning appropriate rewards, the scenario is no longer model-free and hence, no need to use a high discount value.

g) How many possible policies are there? (All policies, not just optimal policies.)

The number of possible policies is defined as;
number of actions ^ number of states

In our case;
number of actions $|A| = 4$
number of states $|S| = 128$

Possible policies = $|A|^{|S|} = \mathbf{4^{128}}$

h) Now, considering the problem as a model-free scenario, provide a program (written in python) able to compute the optimal policy for this world considering the pudding eggs scenario solely. Draw the computed policy in the grid by putting in each cell the optimal action. If multiple actions are possible, include the probability of each arrow. There may be multiple optimal policies, pick one to show it. Note that the model is not available for computation but must be encoded to be used as the "real-world" environment.

The pudding eggs scenario is considered. The states are represented by a tuple of 3. Where the first 2 indices are x,y coordinate and the third index holds True/False value to indicate whether the agent has the egg beater or not.

Rewards:

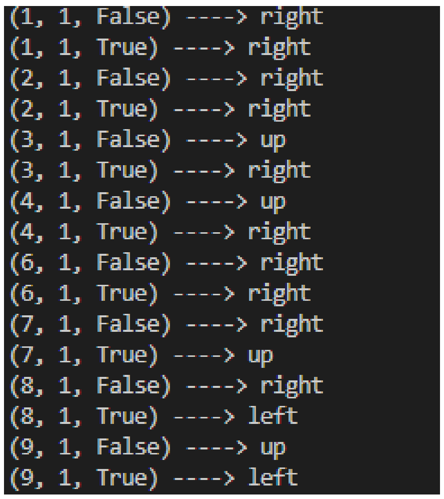
- The agent can pick egg beaters from either (1,3) cell or (8,3). The reward for which is said to be 500.
- The reward for reaching the goal state (pudding maker) is 1000 only if the agent has the egg beater otherwise it is considered as a regular empty cell.
- If it hits the wall, it suffers a penalty of -1000.

Parameters:

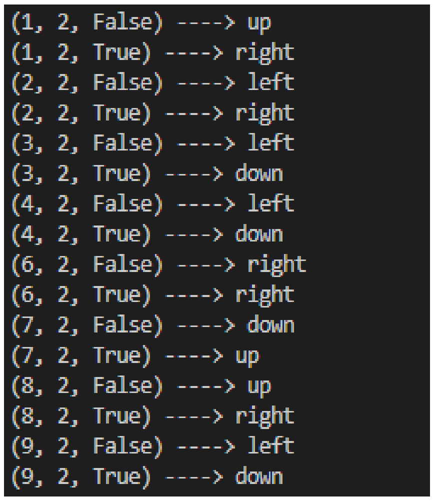
- The value for learning rate is set at 1 but it decreases continually after every 100 episodes. This is because the agent is learning and should now be able to compute next moves using the q_table.
- The discount factor is high i.e. 0.9 because this is a model-free scenario and the optimal policy will depend on future rewards.

Code:
The code for the exercise is attached as chef.py with assignment submission.

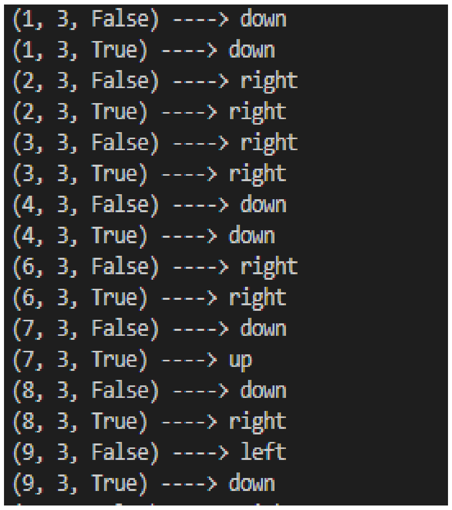
Output:
We compute a q_table for each state action pair. The optimal policy is then picked as the action with the highest computer values.
Following is the output received:



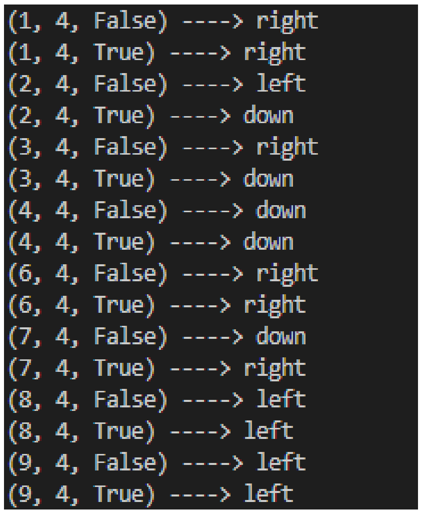
```
(1, 1, False) ----> right
(1, 1, True) ----> right
(2, 1, False) ----> right
(2, 1, True) ----> right
(3, 1, False) ----> up
(3, 1, True) ----> right
(4, 1, False) ----> up
(4, 1, True) ----> right
(6, 1, False) ----> right
(6, 1, True) ----> right
(7, 1, False) ----> right
(7, 1, True) ----> up
(8, 1, False) ----> right
(8, 1, True) ----> left
(9, 1, False) ----> up
(9, 1, True) ----> left
```



```
(1, 2, False) ----> up
(1, 2, True) ----> right
(2, 2, False) ----> left
(2, 2, True) ----> right
(3, 2, False) ----> left
(3, 2, True) ----> down
(4, 2, False) ----> left
(4, 2, True) ----> down
(6, 2, False) ----> right
(6, 2, True) ----> right
(7, 2, False) ----> down
(7, 2, True) ----> up
(8, 2, False) ----> up
(8, 2, True) ----> right
(9, 2, False) ----> left
(9, 2, True) ----> down
```



```
(1, 3, False) ----> down
(1, 3, True) ----> down
(2, 3, False) ----> right
(2, 3, True) ----> right
(3, 3, False) ----> right
(3, 3, True) ----> right
(4, 3, False) ----> down
(4, 3, True) ----> down
(6, 3, False) ----> right
(6, 3, True) ----> right
(7, 3, False) ----> down
(7, 3, True) ----> up
(8, 3, False) ----> down
(8, 3, True) ----> right
(9, 3, False) ----> left
(9, 3, True) ----> down
```



```
(1, 4, False) ----> right
(1, 4, True) ----> right
(2, 4, False) ----> left
(2, 4, True) ----> down
(3, 4, False) ----> right
(3, 4, True) ----> down
(4, 4, False) ----> down
(4, 4, True) ----> down
(6, 4, False) ----> right
(6, 4, True) ----> right
(7, 4, False) ----> down
(7, 4, True) ----> right
(8, 4, False) ----> left
(8, 4, True) ----> left
(9, 4, False) ----> left
(9, 4, True) ----> left
```

Example as explanation: from the output above, it is evident that the optimal policy for (6,1) is right whether it has the egg beater or not. And the optimal policy for (7,1) is right when it has to reach out for an egg beater but if it has the egg beater, the optimal policy is going up towards the goal state of the pudding maker.

i) Is the computed policy deterministic or stochastic?

The computed policy is a stochastic one because the agent has different probabilities of going from one state to another. The probability is distributed among actions that the agent can take from one state and in our case, the agent could go up, down, left, right from one cell to another.

The actions in our code are being picked randomly therefore, at the end, the policies calculated are stochastic ones.

A deterministic policy is a policy that always returns the same action for a given state. In other words, a deterministic policy is a function that maps states to actions without any randomness.

j) Is there any advantage to having a stochastic policy? Explain.

The stochastic policies allow the agent more flexibility and enable exploration of the environment. By introducing the element of randomness, agents get the opportunity to explore results of different action and state pairs as a result the true reward function is learnt.

In the deterministic policy, the agent is informed strictly where it is supposed to go from one state to another which can be harder and redundant to implement. For stochastic policy, the actions are chosen randomly so the need for defining the next states strictly is eliminated.

Part b) Now consider that your agent, because of his tiredness, might goes in the wrong direction. Then each action has a 60% chance of going in the chosen direction and 40% chance of going perpendicular to the right of the direction chosen. Accordingly, with these new settings, answer the following questions:

a) Report the transition function P for any state s and action a ∈ A.

The function below is a probability distribution that describes the probability of reaching state s(t+1) or s(t+1 perpendicular) from state s(t) at action(t) where t is the times value.

$$P\left(s_{t+1} \text{ or } s_{perp\ t+1} \middle| s_t, a_t\right)$$

b) Does the optimal policy change compared to Part a? Justify your answer.

The optimal policy calculated at the end will remain the same. This is because the agent moves to the correct direction 60 percent of the time and after a certain number of iterations, it will learn which policy is the most optimal. The only difference is that the time taken by an agent to arrive at the optimal policy will be longer i.e. it will take more number of episodes to figure out which policy is the best. The best policy for each state will end up acquiring the highest value in q_table anyway.

c) Will the value of the optimal policy change? Explain how.

The value of the optimal policy will change because the q_table will now be updated by different values at each iteration as compared to the ones obtained for the correct one. The agent could be going to the wrong direction 40 percent of the time, as a result, the q_table will be updated for that wrong action in that instance instead of the correct one.