Step 1: Complete Problem 1 : Conceptual

a) How would you define Reinforcement Learning? How is it different from regular supervised or unsupervised learning? [2 points]

Reinforcement Learning is a way to define an environment and build an optimal solution, path, or methodology to interact with that environment. To put it in layman's terms, RL is teaching a computer the "rules of the game" and allowing the machine to find the best way to "win". The "rules" include things like variables, states, transition probabilities, constraints, and optimization functions.

Supervised or unsupervised learning both focus on either classification or regression, they produce some sort of generalized equation. In contrast, RL focuses on building a Markov Decision process. The use cases are different, and therefore the inputs are different. Supervised/unsupervised learning focus on working with given dataset or samples, and RL focuses on interacting with the environment.

b) Can you think of three possible applications of RL that were not mentioned in the lecture? For each of them, what is the environment? What is the agent? What are possible actions? What are the rewards? [3 points]

A common application is any kind of autonomous motion, like autonomous cars. The environment is the physical environment: the road, the vehicle itself, the surrounding people, trees, signage, etc. The agent is the vehicle, its engine, wheels, brakes, etc. Possible actions are changes in velocity and direction. The rewards would be arriving at the destination, with as little fuel burn as possible, and with penalties for unsafe events/actions.

An interesting application would be speech. The environment would be the rules of the language's grammar and syntax. The agent would be a conversation/sentence generator. Possible actions would be the application of words and their order in a sentence. Rewards could be generating a sentence that a human could interact with, or would answer a question asked by a human.

Another application could be food science. The environment would be the tastes and nutritional contents of the ingredients. The agent would be something like a recipe generator. The possible actions would be adding ingredients, how to cook them, and in what order. The rewards would be maximizing nutritional value while balancing caloric intake and avoiding any potentially dangerous ingredients/combinations.

c) What is the discount rate? Can the optimal policy change if you modify the discount rate? [1 points]

The discount rate, in effect, provides the agent an idea of how to balance future rewards and immediate rewards. The higher the discount rate, the more important future rewards are. The agent

will use this to decide what actions to take. The optimal policy will almost certainly change, depending on the discount rate.

d) How do you measure the performance of a Reinforcement Learning agent? [1 points]

The performance is measured by how effectively the policy meets the optimization criteria. In the case of a chess game, it would be measured by how often the policy produces a winning outcome, with possible constraints like fewest moves, or fewest pieces lost.

e) What is the credit assignment problem? When does it occur? How can you alleviate it? [2 points]

The credit assignment problem refers to the difficulty in figuring out which actions directly account for a successful outcome. This occurs most often in complicated systems where rewards are delayed. The higher the dimensionality the harder it is to assign credit to any one action. This is further complicated when the rewards are delayed. One solution is to reduce the dimensionality of the data. By eliminating some states and then comparing outcomes, the system can eventually find more "important" states in regards to the desired outcomes.

f) What is the point of using a replay memory? [1 points]

Sometimes sequential states and rewards are highly correlated, which can lead to inefficient learning – the algorithm is getting stuck in something akin to a local maxima/minima. With replay memory, we randomly sample environments at different times, and use that to train the model. This allows the model to learn the most efficient path without seeing highly correlated outcomes.

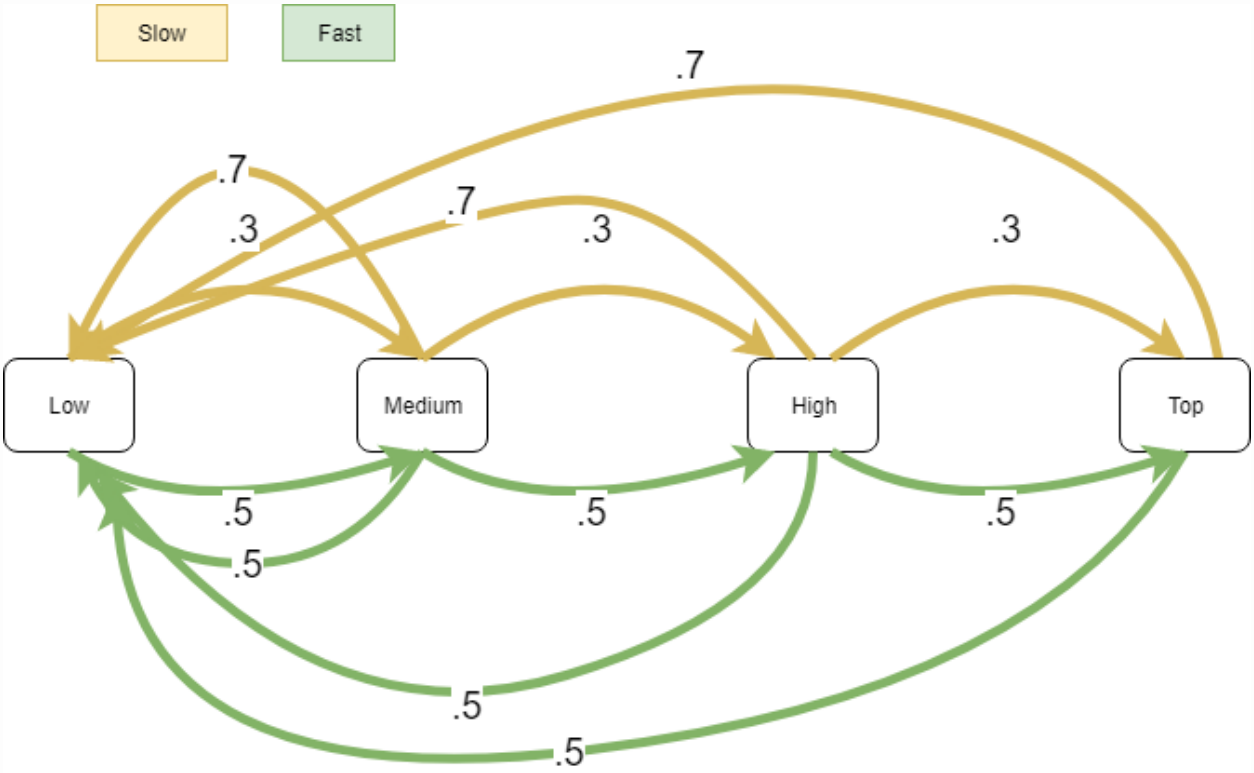
g) What is an off-policy RL algorithm? [1 points]

An off-policy algorithm does not follow the policy at every decision point. It can follow the path that will provide the highest Q-value, regardless of any restrictions imposed by the policy.

Step 2: Complete Problem 2: Markov Decision Processes (MDP)
A robot operates on a hill and uses photovoltaic cell to recharge. That robot can be in one of four states: low, medium, high and top on the hill. If it spins its wheels slowly, it climbs the hill in each time step (from low to medium , from medium to high, from high to top) with a probability of 0.3. It slides down the hill to low with a probability of 0.7. If it spin its wheels rapidly, it climbs the hill in each time step from low to medium , from medium to high, from high to top) with a probability of 0.5. It slides down the slope to low with a probability of 0.5.

Spinning its wheels slowly uses one unit of energy per time step while spinning its wheels rapidly uses two units of energy per time step. The robot is low on the hill and wants to reach the top with minimum energy usage.

a) Draw a diagram of Markov Decision Process [3 points].



The probability of spinning wheels and remaining at the *Low* state are not shown here, to keep the graph legible. But there is a .5 probability of a fast wheel turning from state *Low* and remaining at state *Low*, and a .7 probability of a slow wheel turn from state *Low* and remaining at state *Low*.

b) Solve the Markov Decision Process using undiscounted value iteration for the first 5 iterations (clearly outline the process) [5 points].

c) Describe the optimal policy [1 points].