CS 3600 Midterm (section A)

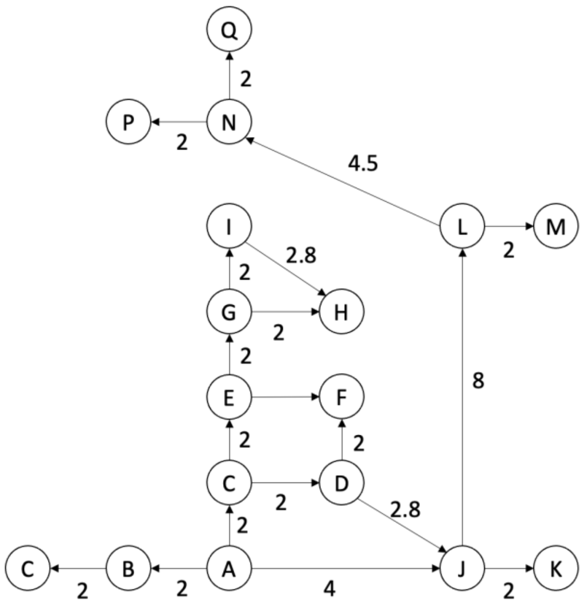
We expect this exam to take ~1.5 hours, however you may use your time in any way you see fit. The exam is open notes, open book, and open lecture videos. The course collaboration policy and late day policy are in effect.

You may edit the file to insert your answers or print, write by hand, and scan back in. You may add extra white space between questions if you need it.

Submit your final version as a PDF to Gradescope.

List any collaborations here: Shavon Edih

Question 1. Consider the following layout of your secret underground lair built into the side of a mountain. The circles are rooms and the edges are corridors. The numbers are the distance in decameters. You have cleverly built a robot to deliver your meals. Unfortunately, because of the geology of the mountain all the corridors are sloped and the robot can only traverse the corridors going downhill (minions must carry the robot back uphill when it has finished making its delivery). The direction of the arcs in the graph indicate the downhill direction. The robot always starts in room A (the kitchen).



1.a (1 pt.) Compute the average number of successors: ___1.12________________

1.b (1 pt.) Compute the average number of predecessors. The predecessors are found by reversing the directionality of the arcs: ____1.12_______________

1.c (1 pt.) Suppose you are in your science lab in room P when you get hungry. The agent could run the breadth-first search *forward* from the kitchen to the delivery destination, or run breadth-first search *backward* by reversing the directionality of the arcs and starting from the destination and working back to the kitchen.

Should the search be conducted forward or backward? _____backwards_____________

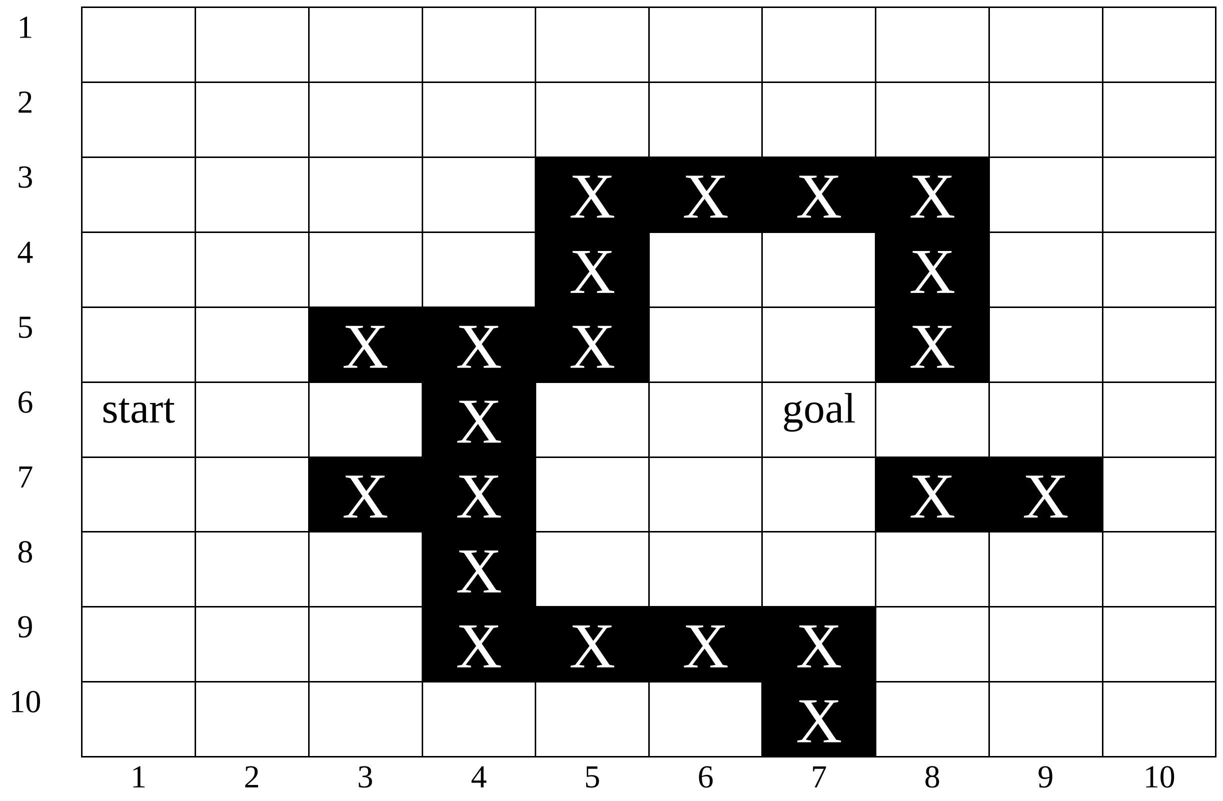
Explain why:

When conducting a bfs, forward takes around 15 moves while backwards only takes 4 moves: P,N,L,J,A.

1.d (1 pts.) Is it always more efficient to run the search in the direction you specified in 1.c regardless of what the robot's destination is? Explain why.

Yes, it is more efficient to run backwards because in the graph specified, A has 3 successors which is the most out of all the nodes present. So, when conducting a forward bfs we would have to search through more nodes than conducting it backwards. Therefore, it backwards is more efficient.

Question 2. Consider the following grid world. The agent can move up, down, left, and right. There is a cost of 1 for every action. The agent starts in the designated cell and must reach the cell marked “goal”. The goal is a terminal state and also has a reward value of 100.



2.a. (1 pt.) If the transition function is deterministic, one would use A* with a heuristic h(state) = manhattan_distance(state, goal). What is the length of the optimal path from the start to the goal?

__18________

2.b. (1 pt.) How many states must be visited by A* using the above heuristic? Hint: the maximum f-value of any state that can be part of the solution path is 18.

____50______

2.c. (2 pts.) Suppose we didn’t have the transition function (and thus didn’t know if it was deterministic or not) and thus used Q-learning instead. We refer to the fact that there is only a single state, the “goal”, that provides a reward as a “sparse” reward function. With an empty Q-table and with most states returning zero reward, it can take quite a while for the agent to find the goal randomly and to start propagating q-values to neighboring states. If we had a “dense” reward function, then every state would provide a reward and the agent would learn much faster.

A perfect dense reward function would be one in which each state gave out reward proportional to its true utility (i.e. imagine we had a transition function and used value iteration until convergence then copied the utility values into the reward function).

Given that we cannot know the true state utilities ahead of time, perhaps we can instead compute in O(n) time (where n is the number of states) a dense reward function that approximates the true utility values for each state. That is, we would need to create a reward function by performing a O(1) operation on each state. Describe in a few sentences how to compute such a reward function.

Hint: think about how admissible heuristics are created.

To create a dense reward function for Q-learning when the transition function is unknown, use a heuristic based on distance. Start with empty rewards and estimate the distance of each state to the goal. Assign rewards inversely proportional to these distances: closer states get lower rewards, while farther ones get higher. This encourages exploration and faster learning without knowing true utility values or the transition function.

2.d. (1 pt.) Suppose the q-learning agent is using an epsilon-greedy exploration strategy using the dense rewards you computed above. Is it a problem if the reward values for states might be misleading? For example, the state at (x=3, y=6) might end up with a reward that is higher than the rewards given at (x=3, y=5) or (x=2, y=5) or (x=2, y=6). That is, will the agent get stuck and never learn a policy that reaches the goal? Why or why not?

The agent will not get permanently stuck or fail to learn a policy to reach the goal. This is because the epsilon-greedy strategy ensures a balance between exploration and exploitation. The agent continues to explore states while also exploiting the actions with higher estimated Q-values. Over time, the agent updates its Q-values based on its experiences, and the Q-learning process allows it to adapt to the true utility values. Even if initial reward values are misleading, the exploration mechanism of epsilon-greedy encourages continued learning and adaptation, preventing the agent from getting stuck in the policy.

Question 3: For each of the following scenarios indicate whether it would be best to use A*, value-iteration, tabular Q-learning, or deep Q-learning. Justify your answer for each.

a. (1 pt). An MDP has thousands of states, four actions per state, the transition function is known, and there are several states that give reward.

   Technique: _____Value Iteration___________

   Justification: Value Iteration is ideal for this scenario since known transitions, and there are several rewarding states.

b. (1 pt.) An MDP has billions of states, 10 actions per state, the transition function is unknown, and there are many states that give reward.

   Technique: ______deep Q-learning__________

   Justification: Deep Q-learning is the preferred option for environments with billions of states, unknown transition functions, and many rewarding states because of its big state space and only Q-learning can do unknown transition functions.

c. (1 pt.) An MDP has tens of thousands of states, each state has different actions associated with it (though never more than five), the transition function is deterministic, and there is only one state that gives reward which is also terminal.

   Technique: ______A*_________

   Justification: A* is the appropriate choice in this scenario because it focuses on finding the shortest path from a start state to a goal state in a deterministic environment. In this case, there is only one state that provides a reward, which is also a terminal state.

d. (1 pt.) An MDP has thousands of states, and each state has 700^4 possible actions, the transition function is deterministic, and there are several states that give reward.

   Technique: ____Value Iteration____________

   Justification: Value Iteration is the best choice when dealing with a scenario with deterministic transition functions, and multiple states that provide rewards. Additionally, value iteration can handle large spaces.

e. (1 pt.) An MDP has thousands of states, ten actions per state, the transition function is not known, there and there are several states that give reward.

   Technique: _______Tabular Q-learning_________

Justification: Tabular Q-learning is the best for this scenario since transition state is not known and several states give rewards and additionally because it has a big state space which tabular can do.

f. (1 pt.) An MDP has thousands of states, four actions per state, the transition function is unknown, there are many states that give reward, and the state space is partially-observable.

Technique: _______Deep Q-Learning_________

Justification: Deep Q-learning is a valuable choice for scenarios with thousands of states, four actions per state, unknown transition functions, many states providing rewards, and partially observable states which deep Q-learning can use the neural network.