1 Traffic Quest (6 points)

You are designing the AI for an automated taxi. Its route graph is made up of intersection nodes \( n \in N \) connected by roads \( r \), where traversing each road \( r \) takes time \( c(r) \) when the road is clear but time \( b(r) \) when busy with traffic. For example, the freeways may be fast when clear but slow in traffic, while surface streets might always be somewhere in between. The agent cannot know whether a road it embarks on will be clear, but it does know that if one road segment is clear (busy), the next will be clear (busy) with probability \( p \). Similarly, if a road is clear (busy), the next segment will be busy (clear) with probability \( 1 - p \). An example is show to the right.

(a) (2 points) Assume the agent knows whether it just experienced a busy road. Assume also that the traffic change probability is known to be \( 1 - p \). Formulate the problem of reaching a fixed destination node \( g \) in minimal time as an MDP. You may wish to use the notation out(\( n \)) (in(\( n \))) for the set of roads leaving (entering) a node \( n \), and end(\( r \)) (start(\( r \))) for the end node (start node) of a road \( r \). Answer for the general case, not the specific graph above.

States:
A pair of the form \((n, s)\) where \( n \in N \) is an intersection, and \( s \in \{\text{busy, clear}\} \) is the traffic state that was just experienced.

Actions:
\( \text{actions}((n, s)) = \text{out}(n) \). Any of the roads \( r \in \text{out}(n) \) leading out of the current intersection \( n \) can be chosen as an action.

Transition function:
For all \( n \in N \) and \( r \in \text{out}(n) \),

\[
\begin{align*}
T((n, \text{busy}), r, (\text{end}(r), \text{busy})) &= p; \\
T((n, \text{busy}), r, (\text{end}(r), \text{clear})) &= 1 - p; \\
T((n, \text{clear}), r, (\text{end}(r), \text{clear})) &= p; \\
T((n, \text{clear}), r, (\text{end}(r), \text{busy})) &= 1 - p.
\end{align*}
\]

Reward function:
For all \( n \in N, s \in \{\text{busy, clear} \}, r \in \text{out}(n) \),

\[
\begin{align*}
R((n, s), r, (\text{end}(r), \text{clear})) &= -c(r); \\
R((n, s), r, (\text{end}(r), \text{busy})) &= -b(r).
\end{align*}
\]

Discount:
\( \gamma = 1 \)
(b) (1 point) In a general map, how many iterations of value iteration will it take before all states have their exact values? Give the smallest bound you can and justify your answer.

There are two possible assumptions that can be made regarding cycles in the road graph, and either assumption is valid.

**Case 1 Assume no cycles in the road graph**

*Answer 1:* $|N|$, the number of nodes in the graph. This is an upper bound on the length of the longest path from start to goal.

*Answer 2:* The length of the longest path in the graph.

Both these answers are motivated by viewing value iteration as an expectimax search to a depth equal to the number of iterations. If we assume there are no cycles, the search tree itself would have maximum depth equal to the number of actions in the longest path, and searching any deeper would not change the values.

**Case 2 Assume that there can be cycles in the graph**

*Answer:* In general, the exact values will not be reached after any finite number of iterations of value iteration.

If there is a cycle in the state graph, and in particular the optimal policy with some probability $x$ results in the agent following such a cycle, then there is some probability ($x^k$) that the agent follows the cycle at least $k$ times under the optimal policy, for any $k$. Consequently, no finite number of iterations of value iteration, equivalent to a finite-depth expectimax search, can consider all possible paths that the agent might take under the optimal policy.

As an example of a case where such a policy might be optimal, suppose there is a very high probability $p = 0.99$ of the traffic state staying the same, and the road graph consist of just two nodes, start and goal. Furthermore, suppose that there is a road from start to goal that takes time 1 when clear but time 10000 when busy, but that there is also a self loop road from start back to start that takes time 1 when clear or busy. Then the optimal policy is to loop at the start until the traffic state is clear, and then take the road to the goal (and with probability 0.99, the traffic will remain clear).

Even though value iteration will never reach the exact values, it is still useful for this problem. Note that in the case of this traffic problem, although we cannot bound the length of the path followed by the agent under the optimal policy, provided that $b(r) > 0$ and $c(r) > 0$ for all roads, we can say that the agent, under the optimal policy, terminates eventually with probability 1. Thus, even though $\gamma = 1$, the values will converge (i.e. become arbitrarily close) to the exact values as more iterations of value iteration are performed. (In MDPs where $\gamma < 1$, this convergence holds even without any termination conditions.)
(c) (2 points) Imagine the taxi must reach the airport at $g$ by the flight time $f$. Now assume that the agent only cares about getting to the airport on time, i.e. on or before $f$. The agent’s utility should be +100 if it is on time and -100 if it is late. How should we alter the original MDP? State the new MDP below. Leave any unchanged aspects blank. Answer for the general case, not the specific graph above.

States:
Augment the state with time remaining before the flight. The state is now $(n, s, t)$ where $n$ and $s$ are as before, and $t \in T = \{0, 1, 2, \ldots, f\} \cup \{\text{END}\}$, where END indicates that $t < 0$, meaning that the taxi is late.

Actions:
Same as before.

Transition function:
The time goes down by however long $r$ took. For all $n \in N$, $t \in T$, and $r \in \text{out}(n)$,
\[
T((n, \text{busy}, t), r, (\text{end}(r), \text{busy}, t - b(r))) = p;
\]
\[
T((n, \text{busy}, t), r, (\text{end}(r), \text{clear}, t - c(r))) = 1 - p;
\]
\[
T((n, \text{clear}, t), r, (\text{end}(r), \text{clear}, t - c(r))) = p;
\]
\[
T((n, \text{clear}, t), r, (\text{end}(r), \text{busy}, t - b(r))) = 1 - p.
\]

Reward function:
\[
R((n, s, t), r, (\text{end}(r), s', \text{END})) = -100.
\]
\[
R((n, s, t), r, (g, s', t)) = +100 \text{ for all } t \geq 0.
\]
\[
R((n, s, t), r, (\text{end}(r), s', t')) = 0 \text{ otherwise.}
\]

Discount: same.

(d) (1 point) MDPs which are time-sensitive often give rise to what are called non-stationary policies, where the actions depend on the time remaining. Give an example from the example above where the optimal action depends on the time remaining. Clearly state the values, actions, times, etc. involved.

Suppose we start at $x$, need to reach $g$, and the initial traffic state is clear. If there is time $t = 4$ remaining, the route through $y$ is optimal, as it guarantees reaching the goal in 4 time steps, while with the other two paths there is some probability of taking more than 4 time steps. In the case that there is only time $t = 3$ remaining, though, the road straight to $g$ is optimal, as it provides a probability $p$ chance of arriving on time; in contrast, the route through $y$ is guaranteed to result in a late arrival, and the route through $z$ provides only a probability $p^2$ chance of arriving on time.
2 Reward Shaping (9 points)

In an MDP, the instantaneous rewards $R$ from a state are only part of the value $V$ of the state, which includes future rewards as well. States may have high outgoing rewards but low value, or vice versa, which can impede learning in a reinforcement learning setting. One common technique is called reward shaping, where an agent’s instantaneous rewards are modified to encourage faster learning, usually by “forwarding” future rewards. For example, a taxi might get rewards for each mile closer it gets to its destination.

Consider a fixed MDP $M$ with states $S$, actions $A$, transitions $T$, discount $\gamma$, and rewards $R$. An example is shown below:

In the example, the discount is 1.0 and the rewards are all at the end of the game.

Your answers to the next parts should be in the format of brief tables.

(a) (1 point) Compute the value of each state under the policy which always chooses action $x$.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>5.5</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>8</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) (1 point) Compute the optimal value of each state.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>7</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>8</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
</tr>
</tbody>
</table>

(c) (1 point) Compute the optimal q-value of each q-state.

<table>
<thead>
<tr>
<th>q-state</th>
<th>q-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a, x)</td>
<td>6</td>
</tr>
<tr>
<td>(a, y)</td>
<td>7</td>
</tr>
<tr>
<td>(b, x)</td>
<td>3</td>
</tr>
<tr>
<td>(b, y)</td>
<td>4</td>
</tr>
<tr>
<td>(c, x)</td>
<td>8</td>
</tr>
<tr>
<td>(e, null)</td>
<td>0</td>
</tr>
<tr>
<td>(f, null)</td>
<td>0</td>
</tr>
</tbody>
</table>

(d) (1 point) What is the optimal policy?
(e) (1 point) If we were q-learning and observed the episodes \([a, x, 0, b, y, 6, e]\) and \([a, y, 0, c, x, 8, f]\), what would the final q-values be? Assume a learning rate of 0.5.

\[ \begin{array}{c|c}
(a, x) & 0 \\
(a, y) & 0 \\
(b, x) & 0 \\
(b, y) & 3 \\
(c, x) & 4 \\
(e, null) & 0 \\
(f, null) & 0 \\
\end{array} \]

Given any MDP, we can compute an equivalent MDP called the \textit{pushed} MDP \(M_P\) by changing the rewards as follows. We define \(R_P(s, a, s') = Q^*(s, a) - \gamma V^*(s')\), where all values are from the original MDP \(M\). Of course, computing the pushed MDP is in general no easier than solving the original MDP.

(f) (2 point) Prove that the optimal Q-values in \(M_P\) will be the same as for \(M\) (in general, not this graph in specific). You may assume that there is an ordering on states \(s\) such that transitions can only result in states later in the ordering. \textit{Hint}: Begin with the Bellman equations which define \(Q^*\) in \(M_P\) and look for opportunities to apply definitions of other quantities.

\textbf{Base case:} For the final state \(s_f\) in the ordering, we have \(Q^*_M(s_f, \text{null}) = Q^*_M(s_f, \text{null}) = 0\), meaning that the proposition is true for the last state in the ordering.

\textbf{Inductive step:} Assume that the proposition is true for all states following \(s\) in the ordering. For state \(s\) itself,

\[ Q^*_M(s, a) = \sum_{s'} T(s, a, s')(Q^*_M(s, a) - \gamma \max_{a'} Q^*_M(s', a')) \]

(1)

\[ = \sum_{s'} T(s, a, s')(Q^*(s, a) - \gamma \max_{a'} Q^*(s', a') + \gamma \max_{a'} Q^*_M(s', a')) \]

(2)

\[ = \sum_{s'} T(s, a, s')Q^*(s, a) \quad (\text{by inductive assumption}) \]

(3)

\[ = Q^*(s, a) \quad (\text{because } Q^*(s, a) \text{ does not depend on } s'). \]

(4)

\[ (5) \]

\textbf{Proof:} From the base case, we have that the proposition is true for the final state in the ordering. From the inductive step, we have that if the proposition is true for some state in the ordering, then the proposition is true for any state immediately previous to that state in the ordering. Thus, by induction, the proposition is true for all state-action pairs.

(g) (1 point) If we were q-learning and observed the same transitions as in (e) but in the pushed MDP, what would the modified sequence of rewards be?

\[ \begin{array}{c|c}
R_{M_P}(a, x, b) & 2 \\
R_{M_P}(b, y, e) & 4 \\
R_{M_P}(a, y, c) & -1 \\
R_{M_P}(c, x, f) & 8 \\
\end{array} \]
Learning from a pushed MDP can be faster than learning from the original MDP, though to really improve learning it is often necessary to alter rewards in ways which change the optimal Q-values. In particular, we can sometimes “shape” rewards by giving the agent immediate points for doing something we believe to be progressing in a good way, such as making progress toward a subgoal. That is, we can modify the rewards by “front-loading” them so that moving to high value states gives large immediate rewards rather than simply access to large future rewards.

(h) (1 point) In general, if we are doing reinforcement learning, why would we prefer an agent to get front-loaded instantaneous rewards which are good indicators of total rewards as opposed to getting all rewards at the end of the game? Give an answer which holds even if the discount is 1.

Front-loaded instantaneous rewards provide information about the relative values of actions, and therefore what the optimal policy is, without needing to wait for rewards from later states to propagate back.
### Conformant Search, The Sequel (4 points)

Consider again an agent in a maze-like grid, as shown to the right. Initially, the agent might be in any location \( x \) (including the exit \( e \)). The agent can move in any direction (N, S, E, W). Moving into a wall is a legal action, but does not change the agent’s actual position. Formally, let \( \text{post}(\ell, d) \) be the possibly unchanged location resulting from choosing direction \( d \) from location \( \ell \). Similarly, let \( \text{pre}(\ell, d) \) be the possibly empty set of locations from which choosing direction \( d \) would result in location \( \ell \). The agent is still trying to reach a designated exit location \( e \) where it can be rescued.

Imagine now that the agent’s movement actions may fail, causing it to stay in place with probability \( f \). In this case, the agent can never be completely sure where it is, no matter what clever actions it takes. Imagine the agent also has a new action \( a = Z \) which signals for pick-up at (it hopes) the exit. The agent can only use this action once, at which point the game ends. The utility for using \( Z \) if the agent is actually at the exit location is 0, but -1000 elsewhere.

(a) (1 point) Assuming that the agent currently believes itself to be in each location \( \ell \) with probability \( P(\ell) \), what is the agent’s expected utility from choosing \( Z \)? Your answer should be a simplified expression for full credit.

\[
-1000 \times (1 - P(e))
\]

Imagine that the agent receives a reward of -1 for each movement action taken, and wishes to find a plan which maximizes its expected utility. Assume there is no discounting, so the agent seeks to maximize the undiscounted expected sum of rewards. Note that despite the underlying uncertainty, this problem can be viewed as a deterministic state space search over the space of belief distributions. Unlike in W1, this variant as posed below does not have a finite search space.

(b) (2 points) Formally state this problem as a single-agent deterministic search problem (not an MDP), where the states are probability distributions over grid locations. The set of states and the start state are specified for you. Complete the definition of the successor function, and fill in the definitions of the action costs and goal test.

**States:** Distributions \( P(L) \) over locations, plus a terminal state \( done \).

**Start state:** The uniform distribution.

**Successor function:** \( \text{Succ}(P(L), Z) = done \). \( \text{Succ}(P(L), d) = P'(L) \) where

\[
P'(\ell) = fP(\ell) + (1 - f) \sum_{\ell' \in \text{pre}(\ell, d)} P(\ell').
\]

Note that \( \text{pre}(\ell, d) \) contains up to two elements, \( \ell \) itself if there is a wall in direction \( d \), and the space in the direction opposite \( d \) from \( \ell \) if that space is not a wall.
Cost of actions:

\[ \text{cost}(d) = 1 \]

\[ \text{cost}(Z) = 1000 \cdot (1 - P(e)) \]

Goal test:

\[ \text{state} = \text{done}? \]

(c) (1 point) Value iteration works perfectly well for deterministic search problems. However, why are standard deterministic search methods such as \( \text{A}^* \) better choices than value iteration for solving this problem?

Because the state space is infinite, it is not actually possible to loop through all of the states as is required by value iteration. In contrast, the optimal solution is at a finite depth in the search tree, and so optimal deterministic search methods will find the optimal solution in finite time.
4 Ghostly Deduction (6 points)

A set of ghosts can be intelligent \((I = i)\) or dumb \((I = d)\), and fast \((S = f)\) or slow \((S = s)\). They sometimes win \((W = w)\) and sometimes lose \((W = \ell)\). Sets of intelligent ghosts are uncommon, but more likely to win. Fast ghosts are also uncommon and also more likely to win. Each game gets an random scenario. The various possibilities are enumerated in the table to the right.

\[
\begin{array}{ccc|c}
I & F & W & P \\
\hline
i & f & w & 0.04 \\
i & f & \ell & 0.01 \\
i & s & w & 0.13 \\
i & s & \ell & 0.07 \\
d & f & w & 0.10 \\
d & f & \ell & 0.15 \\
d & s & w & 0.08 \\
d & s & \ell & 0.42 \\
\end{array}
\]

It is ok to leave answers as fractions.

(a) (1 point) What is the marginal probability that a ghost is fast? 

0.3

(b) (1 point) What is the joint probability that a ghost is fast and wins? 

0.14

(c) (1 point) What is the conditional probability that a ghost is fast given that it wins? 

\[
\frac{0.14}{0.35} = 0.4 
\]

(d) (1 point) What is the conditional probability that a ghost is fast given that it wins and is intelligent? 

\[
\frac{0.14}{0.35} = 0.235 
\]

This change in belief is called explaining away, where discovering the truth of one explanation decreases the belief in competing explanations.

Imagine that the ghost team has two individual ghosts. One or both of these ghosts may be brave \(B = b\), as opposed to cowardly \(B = c\). You know the distribution of the variables for each ghost are as shown to the right.

\[
\begin{array}{ccc|c}
B_1 & B_2 & P \\
\hline
b & b & 0.5-x \\
b & c & x \\
c & b & x \\
c & c & 0.5-x \\
\end{array}
\]

Here, \(x \in [0, 0.5]\).

(e) (1 point) State the value or values of \(x\) for which \(B_1\) and \(B_2\) are independent, or state that there are no such values. Briefly justify your statements with equality expressions or proofs.

\(B_1\) and \(B_2\) are independent if, and only if, \(P(B_1 = b)P(B_2 = b) = P(B_1 = b, B_2 = b)\). Since \(P(B_1 = b) = 0.5\) and \(P(B_2 = b) = 0.5\) (irrespective of \(x\)), independence holds if, and only if, \(P(B_1 = b, B_2 = b) = 0.25 = 0.5 - x\). It follows that independence holds if, and only if, \(x = 0.25\).

(f) (1 point) State the value or values of \(x\) for which \(P(B_1) = P(B_2)\), or state that there are no such values. Briefly justify your statements with equality expressions or proofs.

This is true for \(x \in [0, 0.5]\):

\[
p(B_1 = b) = p(B_2 = b) = 0.5 \quad \text{and} \quad p(B_1 = c) = p(B_2 = c) = 0.5
\]