Midterm Examination Solutions

**Problem 1: Uninformed Search (10 points)**
Show which nodes are expanded, and the order in which they are expanded using:

a.) Breadth-first Search
A, B, C, D, E, F, G, H, I, J

b.) Depth-first Search

c.) Iterative Deepening Search (starting with depth 0, and incrementing the depth by 1 in each iteration)
A, A, B, C, D, A, B, E, F, C, G, H, I, D, J
Problem 2: Heuristic Search (18 points)
Suppose that you are given a heuristic function \( h\)-hat. Although \( h\)-hat is not admissible, it is nearly admissible, in the sense that there is some small constant \( \varepsilon \) such that:

\[
0 \leq h\text{-hat}(n) \leq h(n) + \varepsilon
\]

for every node \( n \). Prove that, with such a heuristic, the A* algorithm will find a path to the goal (assuming one exists) for which the total cost \( g \) is within \( \varepsilon \) of optimal.

Proof:
Let \( y \) be a goal node with optimal cost. We denote this optimal cost \( g(y) = g^* \).

We perform a proof by contradiction. Negating the desired result, we assume that A* has found a path to a goal node \( v \) which has total cost \( g(v) > g^* + \varepsilon \).

In order for node \( v \) to be expanded, it means that no other node in the OPEN list for A* could have a smaller f-hat value.

\[
f\text{-hat}(v) = g(v) + h\text{-hat}(v) > g^* + \varepsilon \quad \text{(since } g(v) > g^* + \varepsilon, \text{ and } h\text{-hat}(v) \geq 0)\]

First, we note that the node \( v \) cannot be the same as the node \( y \), since \( f\text{-hat}(y) < f\text{-hat}(v) \).

\[
f\text{-hat}(y) = g(y) + h\text{-hat}(y) = g^* + h\text{-hat}(y) \leq g^* + \varepsilon \quad \text{(since } h(y) = 0)\]

Now, let node \( u \) be some node on the path to the optimal goal \( y \) that has not yet been expanded, but is on the OPEN list. Note that such a node \( u \) must exist since the goal \( y \) is on some path from the root, and \( y \) has not yet been expanded (otherwise we would have found a goal node prior to \( v \)).

Realize that, \( f\text{-hat}(u) = g(u) + h\text{-hat}(u) \leq g(u) + h(u) + \varepsilon \).

Now, we note that \( g(u) + h(u) \) is simply the cost of reaching the optimal goal \( y \), going through the node \( u \). But since \( u \) lies on the path to the optimal goal \( y \), it means that the cost of reaching the optimal goal \( y \) going through node \( u \), is simply \( g^* \) the cost of reaching the optimal goal node.

Thus, \( f\text{-hat}(u) \leq g^* + \varepsilon \).

This f-hat value is lower than \( v \), which means that node \( u \) must get expanded before expanding node \( v \). However, if this were the case, and node \( u \) can represent any node on the path to \( y \), it means that all nodes on the path to \( y \) (including \( y \)) must be expanded before node \( v \). Hence we have a contradiction, so the goal node \( v \) that is found cannot have cost \( > g^* + \varepsilon \). So the goal node found must be within \( \varepsilon \) of optimal.
Problem 3: Alpha-Beta (12 points)

Consider the following game tree. The value for each leaf node in the tree is shown immediately below each node. The first player is the maximizing player.

![Game Tree]

a.) Perform alpha-beta search, and show which nodes in the search tree are pruned. Also give the “backed-up” values at each node of the game tree that is not pruned.

Pruned nodes: Q
   I (and, thus, T and U)
   K (and, thus, X and Y)

b.) What move should the first player make?

The first player should make the move that leads to node B.
Problem 4: Delayed-Reinforcement Learning (12 points)

a.) Briefly explain what the "exploration vs. exploitation trade-off" is in delayed-reinforcement learning. Explain why it arises and how occasionally taking random actions helps to address this problem.

The exploitation vs. exploration trade-off refers to the fact that an agent may learn a policy that is non-optimal, but still achieves a positive reward. As a result, the agent may choose to follow this policy and obtain the reward. This exploitation of the existing policy, however, may cause the agent to foregoing the opportunity to learn a more optimal policy that achieves a greater discounted reward. Such a policy may exist, but if the agent has not explored the space of policies sufficiently, then this better policy may not be found.

Occasionally taking random actions helps address this problem, for two reasons. First, since the agent is exploiting its current policy most of the time, it is still obtaining positive rewards in general. However, by taking occasional random actions, the agent will explored parts of its world that it would otherwise have not ventured into if it was rigidly exploiting its current policy. Hence, the agent can balance exploiting getting rewards with its current policy, while also occasionally exploring other parts of its world to try to learn better policies.

b.) You are given a robot simulator (like the one you wrote in the first programming assignment). In this simulation, the robot is using delayed-reinforcement learning with $0 < \beta < 1$, and takes a random action (as opposed to the action that maximizes expected reward) with probability $P = c$, where $c$ is a very small (much less than 0.1) positive constant. If the robot simulation runs for many (possibly infinite) number of time steps, is it guaranteed to eventually find an optimal policy? Informally, explain why or why not.

Yes, the simulation is guaranteed to eventually find an optimal policy. The reason for this is that there are two criteria for guaranteeing finding an optimal policy. The first is that the reinforcement learning be done using $0 < \beta < 1$. This is given. The second is that every state in the world be visited infinitely often. Since the simulation contains an element of randomness and may proceed for possibly infinite time steps, every state in the world will be visited a potentially infinite number of times. This theoretically guarantees that an optimal policy will be converged on by the agent. In practice, an optimal policy is generally found after a large number of simulation steps when a small element of randomness is used.
Problem 5: Propositional Logic (15 points)
Use resolution to perform the following proof:

Given: \((P \land Q \land R) \supset S\)
- \(\lnot Q \supset \lnot R\)
- \(\lnot S \supset P\)
- \(R\)

Prove: \(S\)

1. \(\{\lnot P, \lnot Q, \lnot R, S\}\)  Given
2. \(\{Q, \lnot R\}\)  Given
3. \(\{S, P\}\)  Given
4. \(\{R\}\)  Given
5. \(\{\lnot S\}\)  Negation of goal
6. \(\{\lnot P, \lnot R, S\}\)  Resolve 1 and 2 on Q
7. \(\{\lnot R, S\}\)  Resolve 3 and 6 on P
8. \(\{S\}\)  Resolve 4 and 7 on R
9. \(\{\}\)  Resolve 5 and 8 on S
Problem 6: Predicate Logic (23 points)
a.) Convert the following English sentences into first-order predicate logic.
   - Angelica is a mouse.
   - Bob is a friend of all dogs.
   - Cecil is a cat.
   - David is a dog.
   - Angelica is friends with all of Bob’s friends who are not cats.
   - Dogs are not cats.

   Mouse(Angelica)
   (\forall x) (\text{Dog}(x) \Rightarrow \text{Friend}(Bob, x))
   \text{Cat}(Cecil)
   \text{Dog}(David)
   (\forall x) ((\text{Friend}(Bob, x) \land \neg \text{Cat}(x)) \Rightarrow \text{Friend}(\text{Angelica}, x))
   (\forall x) (\text{Dog}(x) \Rightarrow \neg \text{Cat}(x))

b.) Convert the logical sentences from part (a) into clausal form.

   1. \{\text{Mouse(Angelica)}\}
   2. \{\neg \text{Dog}(x1), \text{Friend}(Bob, x1)\}
   3. \{\text{Cat(Cecil)}\}
   4. \{\text{Dog(David)}\}
   5. \{\neg \text{Friend}(Bob, x2), \text{Cat}(x2), \text{Friend}(\text{Angelica}, x2)\}
   6. \{\neg \text{Dog}(x3), \neg \text{Cat}(x3)\}

c.) Use resolution to prove: Angelica is friends with someone.

   7. \{\neg \text{Friend}(\text{Angelica}, x4)\} \quad \text{Negating the goal}
   8. \{\neg \text{Friend}(Bob, x5), \text{Cat}(x5)\} \quad \text{Resolving 5 and 7 on Friend(\text{Angelica}, x4)}
      \quad \text{with } s = \{x2/x4\}
   9. \{\neg \text{Dog}(x6), \text{Cat}(x6)\} \quad \text{Resolving 2 and 8 on Friend(Bob, x5)}
      \quad \text{with } s = \{x1/x5\}
   10. \{\neg \text{Dog}(x7)\} \quad \text{Resolving 6 and 9 on Cat(x6)}
       \quad \text{with } s = \{x3/x6\}
   11. \{\} \quad \text{Resolving 4 and 10 on Dog(David)}
       \quad \text{with } s = \{x7/David\}