Artificial Intelligence

Omid Naeej Nejad
610301189
Assignment 2

Question 1: Zero-Sum Games and the Minimax Algorithm

Solution

1. **True.** Multiplying all terminal utilities by a positive constant preserves their ordering. Since minimax depends only on comparing values, the optimal move does not change.

2. **True.** Adding a constant to all terminal utilities also preserves the relative ordering of outcomes. Thus the minimax decision remains unchanged.

3. **True.** In zero-sum games, minimax recommends the optimal move for the player to move at the root (usually the Max player), because maximizing one's utility automatically minimizes the opponent's.

4. **False.** If the game is not zero-sum, players do not share a single utility function. Minimax is no longer theoretically valid because each player optimizes a different payoff; hence minimax cannot guarantee optimal actions in general-sum games.

5. **True.** In the minimax algorithm, Max seeks to maximize his utility, while Min seeks to minimize the opponent's utility value.

Question 2: Minimax Theory

Solution

(a) What are cooperative and competitive games?

Cooperative games: Games in which players can form binding agreements and work together to maximize a shared outcome.

Competitive games: Games in which players act independently and their interests are opposed. Most classical adversarial games (chess, checkers) are competitive, especially zero-sum games where one player's gain is the other's loss.

(b) What are perfect-information and imperfect-information games?

Perfect-information games: Games in which all players always know the full state of the game and all previous actions. There is no hidden information. Examples: chess, checkers, tic-tac-toe.
Imperfect-information games: Games in which players do not have full visibility of the state or past actions. Some information is hidden, such as cards, private states, or uncertain events. Examples: poker, card games, many real-world strategic situations.

(c) Analyzing minimax as an AI algorithm

Completeness: Minimax is complete if the game tree is finite. It will eventually explore all terminal states and return a value. For infinite games, it is not complete.
Optimality: Minimax is optimal in deterministic, zero-sum, perfect-information games because it assumes the opponent plays optimally and computes the best guaranteed outcome.
Time Complexity: The time complexity is $O(b^m)$ where b is the branching factor and m is the maximum depth of the game tree. Minimax must explore the entire tree.
Space Complexity: The space complexity is $O(bm)$
for depth-first implementation, because only the current branch of recursion needs to be stored.

(d) If Min does not play optimally, how does the Minimax utility for Max compare to the true outcome?

If Min does not always choose the best move for minimizing Max's payoff, the actual utility achieved by Max in the real game will be *greater than or equal to* the value predicted by minimax.
Reason: Minimax assumes the opponent (Min) plays perfectly and chooses the move that minimizes Max's payoff. If Min plays worse than optimal, Max cannot do worse than the minimax value and may do strictly better. Thus:

$$\text{Actual payoff for Max} \geq \text{Minimax value.}$$

(e) If all Min nodes are replaced by chance nodes, how does the root Minimax value change?

Replacing Min nodes with chance nodes means the opponent no longer acts adversarially but instead behaves according to probability distributions.
The resulting value becomes an *expected value* rather than a worst-case value. Thus, the root value will generally:

$$\text{Expected value} \geq \text{Minimax value.}$$

Reason: Chance nodes compute averages, while Min nodes choose the worst case for Max. Therefore, replacing Min with chance increases (or keeps the same) the

expected utility for Max.

(f) If all Max nodes are replaced by chance nodes, how does the root Minimax value change?

A chance node takes an average over outcomes, not the maximum. Therefore, replacing Max with chance reduces (or keeps the same) the expected utility for Max.

Question 3: Applying the Minimax Algorithm

Solution

Step 1: Compute values of the Min nodes

$$B = \min\{3, 12, 8\} = 3$$

$$B \to b_1$$

$$C = \min\{2, 6, 4\} = 2$$

$$C \to c_1$$

$$D = \min\{14, 5, 2\} = 2$$

$$D \to d_3$$

Step 2: Compute the value of the Max root

$$\boxed{A = \max\{B,\, C,\, D\} = \max\{3,\, 2,\, 2\} = 3}$$

$$\boxed{A \to B}$$

Question 4: Alpha-Beta Pruning

Solution

1. **True.** Alpha–beta pruning never changes the final minimax value. It only avoids exploring branches that cannot affect the optimal decision.

2. **True.** Alpha–beta pruning reduces the number of visited nodes and the running time, but it does not alter any minimax value in the tree. The computed utilities remain exactly the same as without pruning.

3. **True.** With perfect move ordering (best children explored first), the number of leaf evaluations becomes linear in the depth of the tree:

   $$O(b^{m/2}) \to O(m) \text{ when } b = 2.$$

   More generally, perfect ordering yields the optimal improvement.

4. **True.** Along any path of the search:

   $$\alpha \text{ is non-decreasing}, \qquad \beta \text{ is non-increasing}.$$

   As the algorithm proceeds, α only increases and β only decreases.

5. **False.** Alpha–beta pruning *can* prune the leftmost branch, if enough information from other branches produces a cutoff before entering it. There is no restriction forcing the leftmost branch to be fully evaluated.

6. **False.** Alpha–beta pruning *can* prune the rightmost branch as well. Pruning depends on the values of α and β, not on the physical position (left or right) of a branch.

Question 5:

Solution

(a) What do the values α and β represent? What does the interval $[\alpha, \beta]$ indicate?

α: The best value that the Max player can guarantee so far along the current path. It is the lower bound on the possible value of the node.
β: The best value that the Min player can guarantee so far. It is the upper bound on the possible value of the node.
$[\alpha, \beta]$: The range of utility values that are still possible for the subtree without causing a cutoff. If this interval becomes invalid (i.e., $\alpha \geq \beta$), pruning occurs.

(b) Does the ordering of children affect the amount of pruning? Provide an example.

Yes. The order in which children are evaluated greatly affects pruning. Exploring moves that likely produce cutoffs increases pruning.
Example: Suppose a Max node has children with values

$$10,\ 1,\ 0.$$

If we evaluate the child with value 10 first, then Min nodes downstream may quickly raise α, causing early cutoffs. But if we evaluate the children with less value first, almost no pruning happens.

(c) What are the best-case and worst-case time complexities of alpha–beta pruning?

Let b be the branching factor and m the depth.
Best case (perfect ordering):

$$O(b^{m/2})$$

Alpha–beta effectively doubles the search depth.
Worst case (bad ordering):

$$O(b^m)$$

No pruning occurs, and the algorithm behaves exactly like minimax.

(d) If the tree is traversed from right to left instead of left to right:

Root value: The minimax value does *not* change. Alpha–beta never alters the final utility.

Number of pruned nodes: The amount of pruning *may change*. A right-to-left ordering might be better or worse depending on which branches lead to earlier cutoffs. Thus, pruning depends on move ordering, not direction.

Question 6: Maximum Possible Alpha-Beta Pruning

The structure is:

$$\text{Max (root)} \rightarrow \begin{cases} \text{Min}_1 \\ \text{Min}_2 \\ \text{Min}_3 \\ \text{Min}_4 \end{cases}$$

The first Min node expands to another Max node:

$$\text{Min}_1 \rightarrow \text{Max}'_{11}, ..., \text{Max}'_{14}$$
$$...$$
$$\text{Min}_4 \rightarrow \text{Max}'_{41}, ..., \text{Max}'_{44}$$

Solution

Step 1: Count total nodes without pruning

There are:

- 4 Min nodes under the root.
- One of them expands to another Max node with 4 leaves.
- The other 3 Min nodes each have 4 leaves.

So total nodes:

$$1 \text{ (root)} \ + \ 4 \text{ (Min nodes)} \ + \ 4 * 4 \text{ (second Max)} = 21$$

Step 2: Determine maximum possible pruning

To maximize pruning, we assume:

- Move ordering is perfect.
- The first child of each decision node produces an extreme value that triggers cutoffs.

At the second Max node: Evaluating the first child may set an α value so strong that the remaining 3 children are pruned.
At each of the other Min nodes: Evaluating the first leaf may set a β value making the remaining 3 leaves unnecessary.
Thus at the leaf layer we can prune:

$$\boxed{3 \text{ leaves per Min node} \times 4 \text{ Min nodes} = 12 \text{ leaves}}$$

Question 7:

(a)

Solution

A (Max): $\alpha = -\infty,\ \beta = +\infty$.
1) Subtree B (Min)
B has a single leaf 10, so

$$B = 10.$$

Update at A:

$$\alpha = \max(-\infty, 10) = 10.$$

2) Subtree C (Min)
At C: $\alpha = 10,\ \beta = +\infty$.
2.1) Child F (Max) with leaf 15:

$$F = 15, \quad \beta_C = \min(+\infty, 15) = 15.$$

2.2) Child G (Max)
At G: $\alpha_G = 10,\ \beta_G = 15$ (inherited from C).
a) Child J with leaf 1:

$$J = 1, \quad \alpha_G = \max(10, 1) = 10.$$

b) Child K (Min) with children N=5, O=2:
At K: $\alpha_K = 10,\ \beta_K = 15$.
- Evaluate N = 5:

$$\beta_K = \min(15, 5) = 5.$$

Now $\alpha_K \geq \beta_K$, so **O is pruned**.
Thus $K = 5$ and

$$G = \max(1, 5) = 5.$$

Now at C:

$$C = \min(15, 5) = 5.$$

No more children under C, so $C = 5$.
At A:

$$\alpha = \max(10, 5) = 10.$$

3) Subtree D (Min)
At D: $\alpha = 10,\ \beta = +\infty$.
3.1) Child H (Max) with leaf 15:

$$H = 15, \quad \beta_D = \min(+\infty, 15) = 15.$$

3.2) Child I (Max)
At I: $\alpha_I = 10,\ \beta_I = 15$.
a) Child L with leaf 20:

$$L = 20, \quad \alpha_I = \max(10, 20) = 20.$$

Since now $\alpha_I \geq \beta_I$ $(20 \geq 15)$, **the remaining child M and its leaves P, Q are pruned**. So

$$I = 20.$$

Back at D:

$$D = \min(15, 20) = 15.$$

At A:

$$\alpha = \max(10, 15) = 15.$$

4) Subtree E (Min)
E has a single terminal value:

$$E = 20.$$

At A:

$$A = \max\{B = 10,\ C = 10,\ D = 15,\ E = 20\} = 20.$$

(b)

Solution

Root O (Max): $\alpha = -\infty,\ \beta = +\infty$.
1) Subtree M (Min)
At M: $\alpha = -\infty,\ \beta = +\infty$.
1.1) Node I (Max)
At I: $\alpha = -\infty,\ \beta = +\infty$.
- A = 6: $\alpha_I = \max(-\infty, 6) = 6$.
- B = 7: $\alpha_I = \max(6, 7) = 7$.
So $I = 7$.
At M: $\beta_M = \min(+\infty, 7) = 7$.
1.2) Node J (Max)
At J: $\alpha_J = -\infty,\ \beta_J = 7$.
- C = 8: $\alpha_J = \max(-\infty, 8) = 8$.
Now $\alpha_J \geq \beta_J$, so **D is pruned**. Thus $J = 8$.
At M (Min):

$$M = \min(7, 8) = 7$$

At root O (Max):

$$\alpha_O = \max(-\infty, 7) = 7$$

2) Subtree N (Min)
At N: $\alpha_N = 7,\ \beta_N = +\infty$
2.1) Node K (Max)
At K: $\alpha_K = 7,\ \beta_K = +\infty$
- E = 5: $\alpha_K = \max(7, 5) = 7$
- F = 6: $\alpha_K = \max(7, 6) = 7$
So $K = 7$. At N: $\beta_N = \min(+\infty, 7) = 7$
2.2) Node L (Max)
At L: $\alpha_L = 7,\ \beta_L = 7$
- G = 8: $\alpha_L = \max(7, 8) = 8$

Now $\alpha_L \geq \beta_L$, so **H is pruned** and $L = 8$
At N (Min):

$$N = \min(6, 8) = 6$$

3) Root value
At O (Max):

$$O = \max(M = 7,\ N = 6) = 6$$

(c): Best Ordering and Alpha-Beta Pruning

Solution

The minimax values of the three Min nodes are:

$$\min\{4, 3\} = 3, \quad \min\{1, 3, 2\} = 1, \quad \min\{4, 2\} = 2.$$

So with *perfect ordering*
- at the Max root we examine child a first (value 3), then c (value 2), then b (value 1);
- at each Min node we examine its children from *smallest* leaf value to largest:
- under a: order $e(3), d(4)$
- under b: order $f(1), h(2), g(3)$
- under c: order $j(2), i(4)$

Step 1: Subtree a

At root (Max): $\alpha = -\infty,\ \beta = +\infty$.
At node a (Min): $\alpha = -\infty,\ \beta = +\infty$.

- Visit e: value 3. $\beta_a = \min(+\infty, 3) = 3$; no cutoff since $\beta_a(3) \not\leq \alpha(-\infty)$.
- Visit d: value 4. $\beta_a = \min(3, 4) = 3$.

So $a = 3$. At root: $\alpha = \max(-\infty, 3) = 3$.

Step 2: Subtree c

At node c (Min): $\alpha = 3,\ \beta = +\infty$.

- Visit j: value 2. $\beta_c = \min(+\infty, 2) = 2$. Now $\beta_c \leq \alpha$ $(2 \leq 3) \Rightarrow$ cutoff.

Child i is pruned. So $c = 2$. At root: $\alpha = \max(3, 2) = 3$

Step 3: Subtree b

At node b (Min): $\alpha = 3,\ \beta = +\infty$.

- Visit f: value 1. $\beta_b = \min(+\infty, 1) = 1$. Now $\beta_b \leq \alpha$ $(1 \leq 3) \Rightarrow$ cutoff.

Children h and g are pruned. So $b = 1$.

Final result

Root (Max) chooses

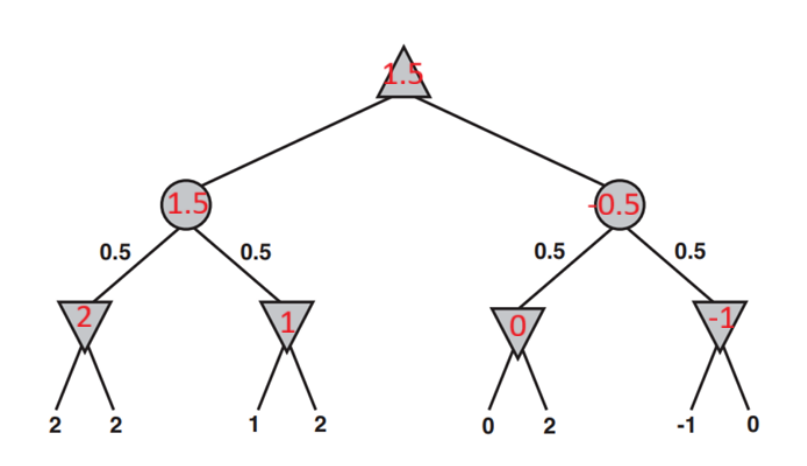
$$\max\{a = 3,\ c = 2,\ b = 1\} = 3,$$

so the minimax value of the root is 3, and the optimal move is to go to subtree a.

Question 8: Expectiminimax with Chance Nodes

Solution

(a)



Best move at the root: choose the *left* branch.

(b)

After evaluating the first six leaves:
- Left subtree value = 1.5 (complete).
- Middle subtree value = 0 (complete).

Since the leaves' value is not restricted, we must evaluate leaf 7 or leaf 8 to determine the Min value under that branch. The strategy won't be changed if we known leaf 7 as well.

(c)

$$V \in 0.5[-2, 2] + 0.5 \cdot 2 = [0, 2]$$

(d)

Skippable leaves under assumption (c): the right subtree leaves 7 and 8.
The root's first subtree should traverse entirely which finally set a threshold (= 1.5): Max will prefer any branch whose expected value could exceed 1.5.
After investigating leaves 5 and 6, the first child's value of second chance node specified to 0. Thus, hopefully it can be 1 which can not exceed the threshold, so the 7th and 8th leaves will be pruned:

$$0.5 \cdot 0 + 0.5 \cdot 2 = 1 < 1.5$$

Question 9:

Solution

We consider the two-player zero-sum game where each player, on their turn, chooses between coin A and coin B. The probabilities are:

$$P_A(H) = 0.75, \quad P_A(T) = 0.25, \qquad P_B(H) = 0.90, \quad P_B(T) = 0.10.$$

Subtree X

$$V_X(A) = 0.75 \cdot 8 + 0.25 \cdot 9 = 8.25$$
$$V_X(B) = 0.9 \cdot 7 + 0.1 \cdot 3 = 6.6$$

Thus,

$$V_X = \min\{8.25,\ 6.6\} = 6.6$$

Subtree Y

$$V_Y(A) = 0.75 \cdot 4 + 0.25 \cdot 2 = 3.5$$
$$V_Y(B) = 0.9 \cdot 0 + 0.1 \cdot 3 = 0.3$$

Thus,

$$V_Y = \min\{3.5,\ 0.3\} = 0.3$$

Subtree Z

$$V_Z(A) = 0.75 \cdot 7 + 0.25 \cdot 5 = 6.5$$
$$V_Z(B) = 0.9 \cdot 9 + 0.1 \cdot 7 = 8.8$$

Thus,

$$V_Z = \min\{6.5,\ 8.8\} = 6.5$$

Subtree U

$$V_U(A) = 0.75 \cdot 1 + 0.25 \cdot 6 = 2.25$$
$$V_U(B) = 0.9 \cdot 8 + 0.1 \cdot 0 = 7.2$$

Thus,

$$V_U = \min\{2.25,\ 7.2\} = 2.25$$

Root (Max)

$$V(S)_A = 0.75 \cdot V_X + 0.25 \cdot V_Y = 0.75 \cdot 6.6 + 0.25 \cdot 0.3 = 5.025$$

$$V(S)_B = 0.9 \cdot V_Z + 0.1 \cdot V_U = 0.9 \cdot 6.5 + 0.1 \cdot 2.25 = 6.075$$

Since $V(S)_B > V(S)_A$,

$$\boxed{\text{coin } B \text{ is the optimal choice}}$$

Question 10:

Solution

The two Min nodes under A have leaf values:

$$\min(8, 7) = 7, \qquad \min(-4, -3) = -4$$

Thus

$$V(A) = 0.5 \cdot 7 + 0.5 \cdot (-4) = 1.5$$

The first Min node under B is

$$\min(8, -9) = -9$$

We know that

$$x \in [-10,\ 10]$$

$$V(B) \leq -4.5 + 0.5 \cdot 10 = 0.5$$

So,

$$V(B) < V(A) \quad \text{for every possible value of } x$$

Since the Max player chooses the larger of $V(A)$ and $V(B)$, the branch B can *never* be selected. Therefore:

$$\boxed{\text{Leaf } O \text{ does not need to be evaluated.}}$$