Understanding Questions

Question 4:

```
                    MAX
                 /        \
               /            \
         CHANCE          CHANCE
        0.5/   \0.5     0.5/    \0.5
        MIN    MIN       MIN     MIN
        / \    / \       / \     / \
       2   2  1   2     0   2  -1   0
```

1. Copy the figure, mark the value of all the internal nodes, and indicate the best move (for the max player) at the root with an arrow. (You don't have to state your solution on the figure as long as you present the solution clearly.)

Answer: we will first evaluate the MIN nodes: Leftmost MIN node: min(2, 2) = 2 And Second MIN node from the left: min(1, 2) = 1, Second MIN node from the right: min(0, 2) = 0 and Rightmost MIN node: min(-1, 0) = -1, now we can evaluate the CHANCE nodes based on what we got: Left CHANCE node: (0.5 * 2) + (0.5 * 1) = 1.5 and Right CHANCE node: (0.5 * 0) + (0.5 * -1) = -0.5, with that we can evaluate the MAX node: MAX node: max(1.5, -0.5) = 1.5 so we get from that, that the best move for the MAX player at the root is towards the left CHANCE node.

2. Given the values of the first six leaves, do we need to evaluate the seventh and eighth leaves? Given the values of the first seven leaves, do we need to evaluate the eighth leaf? Explain your answers

Answer: <u>Given the values of the first six leaves, do we need to evaluate the seventh and eighth leaves?</u> Yes, we need to evaluate all the CHANCE nodes in order to evaluate the MAX node but in this case we can't evaluate the right CHANCE node with only the first six leaves that are 2, 2, 1, 2, 0, and 2, with them we can only evaluate the 1st MIN Node that is 2, MIN Node 2 that is 1, and MIN Node 3 that is 0. However, to evaluate 4th MIN Node, we need the seventh leaf (the eighth leaf is not needed since its 0 and -1 is smaller than it). <u>Given the values of the first seven leaves, do we need to evaluate the eighth leaf?</u> No, Since the seventh leaf already determines the MIN Node 4 which is what we need to evaluate the right CHANCE node, that way all the CHANCE nodes are evaluted so the eighth leaf is not necessary (the explanation given for the first part of the question explains why we can evaluate the other CHANCE nodes).

3. Suppose the leaf node values are known to lie between –2 and 2 inclusive. After the first two leaves are evaluated, what is the value range for the left-hand chance node?

Answer: after evaluating the first two leaves we get that MIN Node 1 (leftmost) is min(2,2)=2, since we wanna know the value range for the left-hand chance node we will evaluate these two cases:

1. The 2nd MIN Node is 2 which is the highest number it can get (given the leaf node values range from -2 to 2) so in this case: Left CHANCE

Node=0.5×(MIN Node 1)+0.5×(MIN Node 2) and in our case we get that max CHANCE Node = $0.5\times2+0.5\times2==1+1=2$.
2. The MIN Node 2 is -2 which is the lowest number it can get (the leaf node values range from -2 to 2) so in this case: Left CHANCE Node=0.5×(MIN Node 1)+0.5×(MIN Node 2) and in our case we get that Min CHANCE Node = $0.5\times2+0.5\times(-2)=0.5\times2-1=1-1=0$.
We can conclude from that: the value range for the left-hand CHANCE node is [0,2] (lies between 0 and 2).

4. Given the assumption of (3), and assuming we keep evaluating the leaves left to right until we know what is the best action for the max player, after which leave we can stop and return the answer?
Answer: once the fifth leaf is evaluated, the best action for the MAX player becomes clear, and further evaluation isn't necessary: Once the fifth leaf (0) is evaluated, we can determine that the rightmost CHANCE node even if it gets evaluated fully, the highest it can reach is 1 (assuming best-case scenario: 2 for the last two leaves). Since the left CHANCE node is already 1.5, the MAX node will always choose the left CHANCE node. Therefore, we can stop after evaluating the fifth leaf and determine that the best action for the MAX player is to choose the left CHANCE node. This makes the final MAX value
1.5 as calculated earlier.

Question 6:

success in the game
1. Theoretically, which of the algorithms is more suitable for the game 2048? Explain what each of the algorithms assumes about the opponent and which of the assumptions better suits the game.
Answer:
AlphaBetaAgent (Minimax with Alpha-Beta Pruning):
The assumption about the opponent here is that the AlphaBetaAgent assumes that the opponent (the tile generation) is adversarial, trying to minimize the player's score. It optimizes for the worst case scenario by pruning branches that do not need to be explored. It’s **Suitability for 2048:** The assumption is not well suited for 2048 since the game features random tile placements rather than an adversarial opponent. The conservative approach of AlphaBetaAgent may not be optimal for dealing with the stochastic nature of 2048.

ExpectimaxAgent:

The assumption about the opponent here is that the ExpectimaxAgent assumes that the opponent's moves are random and computes the expected utility by averaging over all possible outcomes. It models the game's randomness by considering the probabilities of different outcomes. It’s **Suitability for 2048:** This assumption is more suitable for 2048 since it aligns with the game's mechanics, where new tiles appear in random locations with known probabilities.

in comparison: The expectimax algorithm's approach to randomness makes it theoretically more suitable for 2048 than the minimax algorithm with alpha-beta pruning. It captures the probabilistic nature of the game better, leading to more strategic and potentially higher scoring moves.

2. Empirically, which of the algorithms is more successful in 2048? To answer the question, run each of the algorithms for 10 games and display the scores and the largest tiles that each of the algorithms achieves.
Answer:

Analysis:

Average Scores & Mean Highest Tile:

AlphaBetaAgent:

experiment	Game 1	Game 2	Game 3	Game 4	Game 5	Game 6	Game 7	Game 8	Game 9	Game 10
1 - scores	3260	11848	5900	6768	7160	3212	6948	6408	11880	7332
2 - scores	6720	11564	6432	7112	12308	7220	5700	7016	14312	6352
1 - highest tile	256	1024	512	512	512	256	512	512	1024	512
2 - highest tile	512	1024	512	512	1024	512	512	512	1024	512

- Mean Score: 7487
- Mean Highest Tile: 512 (appeared most of the times)

ExpectimaxAgent:

experiment	Game 1	Game 2	Game 3	Game 4	Game 5	Game 6	Game 7	Game 8	Game 9	Game 10
1 - scores	7360	12200	7700	6528	15872	16016	7600	7164	16344	14600
2 - scores	12200	5620	12176	12240	6784	11324	11744	3024	6424	5556
1 - highest tile	512	1024	512	512	1024	1024	512	512	1024	1024
2 - highest tile	1024	512	1024	1024	512	1024	1024	256	512	512

- Highest Tile Achieved: 9923
- Mean Highest Tile: 1024 (appeared most frequently)

The experimental results demonstrate the ExpectimaxAgent's superior performance in 2048 compared to the AlphaBetaAgent. Despite longer computational times per game, the ExpectimaxAgent consistently achieves higher scores and larger tile values. Across numerous trials, the ExpectimaxAgent exhibited a higher mean score and higher mean maximum tile value, providing quantitative evidence of its ability to navigate the game's stochastic nature more effectively.

3. Are the theoretical and empirical results consistent with each other? If so, explain. If not, explain how this is possible.
Answer:
The empirical results very much align with the theoretical predictions. The ExpectimaxAgent that is better at handling the game's randomness, performs better both theoretically and empirically. It achieves higher scores and larger tiles more consistently than the AlphaBetaAgent that operates under the less suitable assumption of an adversarial opponent.

4. Compare the standard deviations in the score and the largest tile of the two algorithms. Give an intuitive explanation for the difference in results
Answer:

Analysis:

I used Standard Deviation Calculator (calculatorsoup.com), to calculate the standard deviations in the score and the largest tile

AlphaBetaAgent:

- Score: 3534.8
- Highest Tile: 281.73

ExpectimaxAgent:

- Score: 4359.435
- Highest Tile: 323.8

Intuitive Explanation:

The ExpectimaxAgent excels by considering probabilities, leading to higher average scores. Its probabilistic approach allows it to occasionally find optimal paths, resulting in larger tiles. It uses expectimax decision-making, considering expected future states, which helps handle uncertainty better

The AlphaBetaAgent assumes a deterministic opponent, causing a mismatch with 2048's randomness, leading to more predictable but less optimal outcomes.

The AlphaBetaAgent's minimax with alpha-beta pruning is less suited for 2048's probabilistic nature, reducing outcome variation as it doesn't adapt well to randomness.

Alternative games

1. Imagine an alternative game "2048 Boom" where after every move of the player there is a one in a billion chance of a "boom" where the game stops and the player finishes with a score of 0. How will the change affect the performance of the algorithms? In this section only, assume an exact expectimax algorithm that takes into account the probabilities of the moves.

Answer:

- **Expectimax:** This algorithm would consider this tiny probability in its calculations. Since the probability is extremely low (one in a billion), it would have a negligible effect on the overall strategy. The expectimax player would still focus on maximizing the expected score, knowing that the likelihood of the "boom" event is practically zero.
- **Minimax:** The "boom" event, being extremely rare, would not significantly influence the minimax algorithm's strategy. Minimax would not alter its behavior to account for this low probability event, as it does not handle probabilities well and focuses on worst case scenarios, which in this case would still involve deterministic tile merges rather than random events.

2. In our implementation the expectimax player does not calculate the exact expectation but assumes that all outcomes have equal probability. Use this fact to create a new simple game where the minimax player outperforms this not-exactly-expectimax player.

Answer:

This game must have a scenario where equal probability assumption leads to poor decision making. With that we create a game called Weighted Dice Roll where the player rolls a weighted die multiple times to accumulate the highest possible score. Each face of the die has a different probability of appearing (the not exactly expectimax player incorrectly assumes that all faces are equally likely)

Game Rules:

1. The player rolls a 6 sided die up to three times.
2. The faces of the die have the following values: 1, 2, 3, 4, 5, and 6.
3. The probabilities of each face appearing are different and known:
   - 1: 10%
   - 2: 10%
   - 3: 10%
   - 4: 10%
   - 5: 10%
   - 6: 50%
4. The player must decide whether to roll again or keep the current score after each roll.
5. The player's total score is the sum of the values from each roll.

Minimax Outperforms Expectimax: because of the weighted probabilities the Minimax can be designed to take into account the actual probabilities, avoiding low probability outcomes effectively, while the Expectimax, assuming equal probabilities, fails to leverage the higher likelihood of rolling a 6, leading to suboptimal decision making.

Example:
(The scenario is that the player decides whether to roll again based on the current score and probabilities.)

Minimax Strategy:

1. First Roll:
   - High chance to roll a 6 (50%).
   - Choose to roll with an expected value:

$0.1\times1+0.1\times2+0.1\times3+0.1\times4+0.1\times5+0.5\times6=0.1(1+2+3+4+5)+0.5\times6=4.5$

   - Based on high expected value, decide to roll.

2. Second Roll (assuming 4 or lower in the first roll):
   - Similar decision, high chance of improving the score.
   - Continue rolling if below target threshold.

3. Third Roll (if first two rolls were low):
   - Last chance to maximize score.

Not exactly Expectimax Strategy:

1. First Roll:
   - Assumes equal probability: (1+2+3+4+5+6)\6 = 3.5 This assumption leads to a conservative strategy. May choose to keep a low score, not realizing the high probability of rolling a 6.

2. Subsequent Rolls:
   - Same equal probability assumption leads to similar conservative choices.

In conclusion the Minimax understands the weighted probabilities, optimally deciding when to roll again. Likely achieves higher scores by leveraging the higher chance of rolling a 6. An Example Score: approximately 12-18 on average.
While the Expectimax assumes equal probability, leading to overly cautious decisions.likely achieves lower scores due to incorrect assumptions. an Example Score: approximately 7-10 on average.
So as we can see the actual probabilities allow minimax to make better informed decisions, while the expectimax player's equal probability assumption leads to suboptimal outcomes, making minimax outperform it.