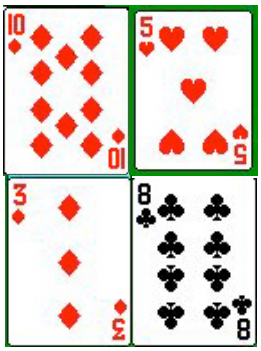
# Artificial Intelligence Tutorial 2 - Answers

1) Lets invent another trivial, but illustrative board game. Four cards are dealt face up from the pack and two players take it in turn to put them into a 2x2 grid, with player 1 going first. For example, they may end up with a grid like this:



Then they add up the numbers on each diagonal. The diagonal from top left to bottom right is called D1, the diagonal from bottom left to top right is D2. The scores at the end of the game are:

player 1 scores D1 + D2 *if both D1 and D2 are even*
player 1 scores D1-D2 if D1 *is even and D2 is odd*
player 1 scores D2-D1 if D2 *is even and D1 is odd*
player 1 scores –D1 – D2 *if both D1 and D2 are odd*

player 2 scores D1 + D2 *if both D1 and D2 are odd*
player 2 scores D1-D2 if D1 *is odd and D2 is even*
player 2 scores D2-D1 if D2 *is odd and D1 is even*
player 2 scores –D1-D2 *if both D1 and D2 are even*

Picture cards count as 10, and the person with the highest score wins the game (it's drawn if they score the same). This scoring scheme is not as difficult as it looks: if a diagonal adds up to an even number, add it to player 1's score and take it from player 2's score, if it adds up to an odd number, then add it to player 2's score and take it from player 1's score. In the above example, diagonal D1 is 10 + 8 = 18, and diagonal D2 is 3 + 5 = 8. As D1 and D2 are both even, player one gets all the points, scoring 18+8=26. Player 2 scores –18-8 = -26.

1a) If the 3 in the above grid had been a 4, what would player two have scored?

If the 3 was a 4, then D1 would still be 18, but D2 would be 4+5=9. As D1 is even and D2 is odd, player 2 scores D2-D1 = 9-18 = -9.

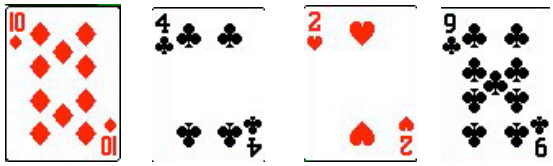
1b) Suppose we are programming an agent to play this game using a minimax approach. It is useful to identify symmetries in the game which we can use to reduce the search space. Explain why it doesn't matter where in the grid player 1 puts his/her first card and discuss how this will affect the search that our agent does when deciding how to play the first card.

The scores at the end of the game only depend on which cards are diagonally opposite to each other. That is, it doesn't matter where player one puts the first card, as the score will depend on where the other cards are put in relation to this. So, the scores depend on relative positions, not absolute positions.

This will reduce the search that an agent playing as player 1 has to perform in order to determine which card to play first and in which grid position: without loss of generality (i.e., without losing of the possibilities for scoring), we can just presume that player 1 puts the first card in the top left hand

corner. This will reduce the search by three quarters, as we don't have to worry about scenarios where player one puts the first card in the top right hand, bottom left hand or bottom right hand corners.

1c) Suppose that these cards were dealt:



and that for his first go, player 1 puts the 10 in the top left hand corner of the grid. Explain in English why, if player two acts rationally, player 1 will lose this game.

If player two acts rationally, this means that they do the right thing in a scenario, given the information about that scenario that they have. In this case, doing the right thing means choosing the right move to eventually lead to a win, or to minimise losses if a win isn't possible from that scenario.

Here, if player two put the 9 in the bottom right hand corner, the two diagonals would be D1=10+9=19 and D2=2+4=7. Hence, if player two acts rationally, player 1 is guaranteed to score 6-19 = -13, and hence lose the game.

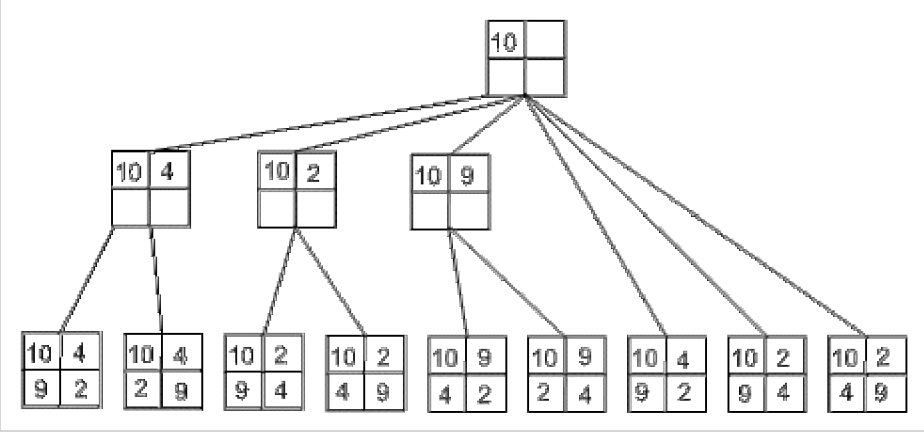
1d) Complete the minimax diagram below and use it to prove that player 1 will lose this game if they put the 10 in the top left hand corner and player two acts rationally.

[To complete this diagram in the space provided, think about why the third line says player 1 & 2, and think about symmetry in where player two puts his first card, as in question 2b.]

It's rarely possible to draw entire search trees on paper without thinking hard about symmetry. If a search space has a symmetry, then this will mean that the number of nodes to draw on the search graph is greatly reduced. To determine symmetries in minimax search problems, ask yourself whether the final score will be affected if you do something one way or another. If the final score for the board state will obviously be the same whether you do one thing or another, then you only have to do one of those things in the search. As well as enabling us to draw search trees, it's important to use symmetry when writing your search agents, because they will be able to search more quickly, which will mean they can search further and hence make better choices.

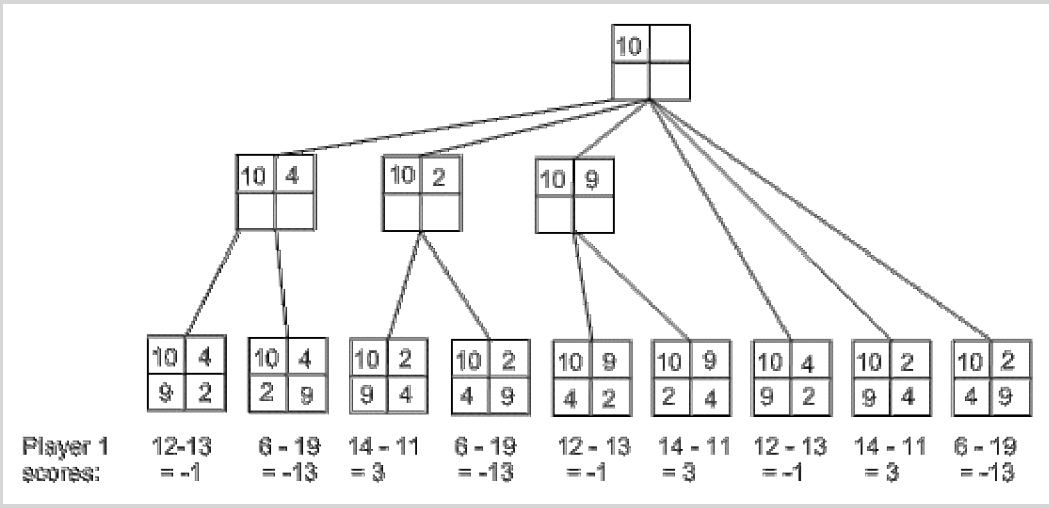
In this board game, it doesn't matter where player 1 puts his/her first card, because the scores are only dependent on the spatial relationships between the cards. So, without losing any generality, we can assume that player 1 will place the first card in the top left hand corner. Following this, the outcome of the game is only dependent on whether player two puts a card on the same diagonal as player 1's card, or not on the same diagonal. So - again, without losing any generality – we can assume that player two will put his/her card in the top right or bottom right position. If they put the card in the bottom left position, this will result in the same set of scores as in the case where they put the card in the top right position (because the only important thing is that it is not on the same diagonal as player 1's card). Finally, for player 1's second turn, it may be that they have to put it on a square with no card on the same diagonal. In this case, it doesn't matter where they put it, so they may as well put it on the top line. There will only be one place left for player two's last card, so we can fill this in straight away.

Using the symmetries of the problem, we can work out the search tree if player 1 put the 10 in the top left hand corner:



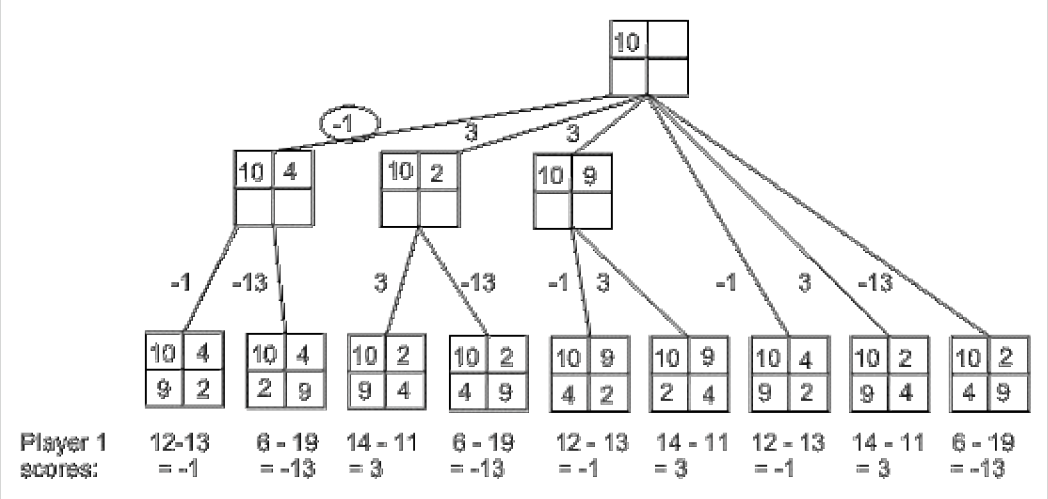

This may be simpler than you expected. This is because we have used the symmetry to really cut down on our work. For instance, if player 2 chose to put a card in the bottom left corner, then this leaves the top right and bottom left still open, into which the two remaining cards will be put. Of course, it doesn't matter which way round the two cards are put in these positions, because the diagonal will still add up to the same score. So, really, if player two puts their card in the bottom right corner, then the board is fixed in terms of the final score, so we can choose to put the smallest remaining number in the top right without worrying about losing any important end states.

We now come to the minimax part of the problem: to show that if player 1 puts a 10 in the top left hand corner then player 2 is guaranteed to win if they follow the minimax strategy. To show this, we first put the scores for player 1 beneath the final board states as follows:




Remember how the scores were calculated: if any diagonal adds up to an even number, then this is added to player 1's score and taken from player 2's, otherwise it is added to player 2's score and taken from player 1's. As there is only one odd number, then there will always be a single odd diagonal and a single even diagonal. For instance, in the leftmost end state, the diagonal from top left to bottom right was 10 + 2 = 12, so player 1 got this  (because it is even). The diagonal from top right to bottom left was 9 + 4 = 13, so player 2 got this (because it is odd). That means that, in total, player 1 scores 12 – 13 = -1 for that final end state.

We now propagate these scores to the top of the diagram as follows:




Whenever there is a choice for which number to put at the next stage up the diagram, we must appeal to the minimax principle: that player 1 will choose in order to maximise their score, and player 2 will choose in order to minimise player 1's score. It is important to know who is making the choices at which stages in this diagram. If we look at the –1 which has been circled, then we should ask ourselves: will this number be –1 or –13 (the two possibilities from beneath the board state). The –1 comes if player 1 puts a 2 in the bottom right hand corner, and the –13 comes if player 1 puts a 2 in the bottom left hand corner. So, player 1 has the choice here. Hence we move the –1 to the top, as we can assume that player 1 will choose to score –1 rather than –13.

We can now look at player 2's first move: if they play 4 in the top right corner, player 1 can get at most –1. If they play 2 in top right, player 1 can get 3. If they put a 9 in the top right corner, then player one can put a four in the bottom right and score 3. If they put a 2 in the bottom right corner, then player 1 can get –1. If they play a 4 in the bottom right, then player 1 can get 3. If they put a 9 in the bottom right hand corner, player 1 will score –13. So which will they choose? Obviously, it will be the 9 in the bottom right corner. So, we've used minimax reasoning to show that player 2 will make sure player 1 scores –13 if player 1 puts a 10 in the top left corner.

Note that, if we do this for the alternatives to player 1's first choice (e.g., putting a 2 in the top left corner), then after the calculations, it turns out that it is always possible for player 2 to make player 1 score a negative number. Hence, player 1 cannot win in this situation.

2) In a general game of the type described above, if player 1 puts the highest even card into the top left hand corner, then player 2 should put the highest odd card into the bottom right hand corner (if there is an odd card).

2a) If we programmed an agent with this rule, explain why this would enable an agent playing as player two to act rationally when putting the second card into the grid.

As before, the agent will act rationally if they choose a move which will eventually lead to them winning with maximum profit, or minimising their losses. In this case, the rule would force the agent to make an odd-scoring diagonal with the highest score possible, as both the highest scoring cards have been chosen, and an odd plus an even is an odd number. As this will get added to their score, it is a good rule to use (it doesn't guarantee a win, but does guarantee minimal losses if they don't win.

2b) We can represent this rule in Prolog as follows:

```
put_card(A, bottomright) :-
      is_card(X),
      player_one_put(X, topleft),
      \+ (is_card(Y), is_even(Y), is_bigger(Y,X)),
```

```
      is_even(X),
      is_card(A),
      \+ is_even(A),
      \+ (is_card(Z), \+ is_even(Z), is_bigger(Z,A)).
```

A set of rules like this may enable our agent to become an expert system and play the game without having to perform a search.

What would be the rule if Player one put the biggest odd number in the bottom left corner? Write down the representation of this in Prolog.

In this case, we would want our agent to put the biggest even numbered card in the top right, for the same reasons as before. The Prolog representation of this rule is very similar to the previous one:

```
put_card(A, topright) :-
      is_card(X),
      player_one_put(X, bottomleft),
      \+ (is_card(Y), \+ is_even(Y), is_bigger(Y,X)),
      \+ is_even(X),
      is_card(A),
      is_even(A),
      \+ (is_card(Z), is_even(Z), is_bigger(Z,A)).
```

2c) In a game where two even and two odd cards have been dealt, how many rules would an agent playing the second move need in order to cover all eventualities?

We count the number of rules required in terms of how many possibilities player one has for playing the first card. However, we can reduce this number by making the rules more intelligent. Firstly, there are four different cards in the general case, and four different grid positions for player one to choose from, hence 16 possible first moves for player 1. So, in the most stupid case, our agent would need 16 rules. However, if we describe the cards as the "highest odd", "lowest odd", "highest even" and "lowest even" cards, and defined a predicate able to calculate the diagonal opposite of a position, we could replace the 16 rules by just four:

If player 1 puts the highest even card in position X, then put the highest odd card in position Y, where Y is diagonally opposite to X.
If player 1 puts the highest odd card in position X, then put the highest even card in position Y, where Y is diagonally opposite to X.
If player 1 puts the lowest even card in position X, then put the lowest odd card in position Y, where Y is diagonally opposite to X.
If player 1 puts the lowest odd card in position X, then put the lowest even card in position Y, where Y is diagonally opposite to X.

If we think more about this problem, as there are two odds and two evens dealt, all player two needs to do is to look at the card player 1 has put on the grid: if they put an odd on the grid, then player two should put an even diagonally opposite this. If they put an even on the grid, then player two should put an odd diagonally opposite this. This can be expressed as a single rule: put the opposite parity into the opposite diagonal of the card that player one has played.

3a) Use a truth table to show that, in propositional logic, $(P \rightarrow Q) \leftrightarrow (\neg Q \rightarrow \neg P)$ is valid.
3b) For a sentence with n variables and m connectives (counting identical connectives multiple times), what can you say about the dimensions of its truth table?

3a) We will use a truth table to show that the sentence is true in all models, i.e. for all values of P and Q. There are four cases to consider: when P is true and Q is true, when P is true and Q is false, when P is false and Q is true and when P is false and Q is false. We will have to evaluate the truth of four subexpressions along the way along the way, , using the definitions of the connectives $\neg$, $\rightarrow$ and $\leftrightarrow$ . The truth table looks like this:

| P | Q | $\neg$Q | $\neg$P | P$\rightarrow$ Q | $\neg$Q $\rightarrow$ $\neg$P | …$\leftrightarrow$… |
|---|---|---|---|---|---|---|
| true | true | false | false | true | true | true |
| true | false | true | false | false | false | true |
| false | true | false | true | True | true | true |
| false | false | true | true | True | true | true |

We see that the sentences $P \rightarrow Q$ and $\neg Q \rightarrow \neg P$ are both true unless P is true and Q is false, and so the sentence $(P \rightarrow Q) \leftrightarrow (\neg Q \rightarrow \neg P)$ is always true.

3b) Such a truth table would have $2^n$ rows and **at most** (n + m) columns. There is a row for each assignment of true and false to the n variables $P_1,\ldots,P_n$. A single variable $P_1$ has 2 assignments, and each time we add a variable $P_k$ we need to consider all the assignments for $P_1,\ldots,P_{k-1}$ twice: once with $P_k$ true and once with $P_k$ false. So rows(k) = 2.rows(k-1), and hence rows(n) = 2n.

We have a column for each of the n variables. Each connective represents a subexpression of the sentence (including one for the sentence itself) and if we have a separate columns for each one then we have m columns that evaluate these expressions. So the table has (n + m) columns. However, in practice the number may be less because a) a subexpressions may be repeated, and we only need a single column to evaluate all its occurrences and b) we might not need to explicitly write down every subexpression in order to work out the truth of sentence.

4a) Translate the following sentences into first-order logic, using a function to represent mother:

(i) All dogs are mammals
(ii) Fido is a dog
(iii) Fido's mother is a mammal
(iv) All mammals have a mother who is a mammal

4b) Retranslate (i) to (iv) into first-order logic without using functions.

4c) Write your answers to part (a) as Prolog style Horn clauses, if possible.

4a) The straightforward translation is:

(i) $\forall$ X (dog(X) $\rightarrow$ mammal(X)).
(ii) dog(fido).
(iii) mammal(mother(fido)).
(iv) $\forall$ X (mammal(X) $\rightarrow$ $\exists$Y (Y = mother(X) $\wedge$ mammal(Y)).

4b) (i) and (ii) remain the same. We can rewrite (iii) as $\exists$X (mother_of(X, fido) $\wedge$ mammal(X)) and (iv) as $\forall$ X (mammal(X) $\rightarrow$ $\exists$Y (mother_of(Y, X) $\wedge$ mammal(Y)).

4c) For part (a) the sentences (i) to (iii) are Horn clauses, a conjunction of literals implying a single literal in the head of the clause. To write them in Prolog, we simply drop the universal quantification, turn the implications around and write the implication sign as :-. Don't forget the full stop at the end. The answers are:

(i) mammal(X) :- dog(X).
(ii) dog(fido).
(iii) mammal(mother(fido)).

Sentence (iv) is not a Horn-clause, so we can't write it directly in Prolog.