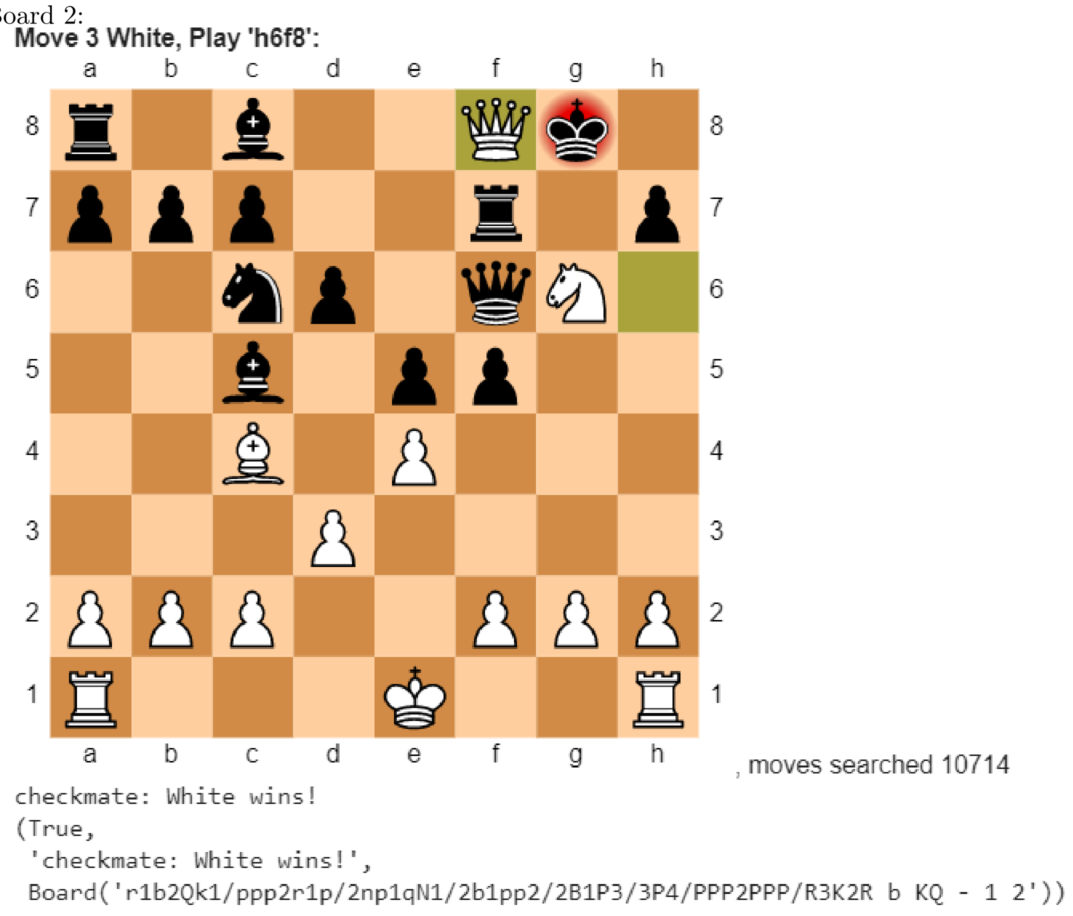
# Homework 3
# 601.464/664 Artificial Intelligence

**Jason Zhang jzhan127**

April 10, 2020

1 Open the following google colaboratory notebook. Follow all the steps specified in it. Include link to your solved notebook in your submission. Optional: implement your own chess-playing agent and we will run a small competition between agents of other students (you can work in teams).
**Answer:**
https://colab.research.google.com/drive/1VT7vXc-OR-iPnU0dz7cRQpB-D9Jbqbgp

Board 2:

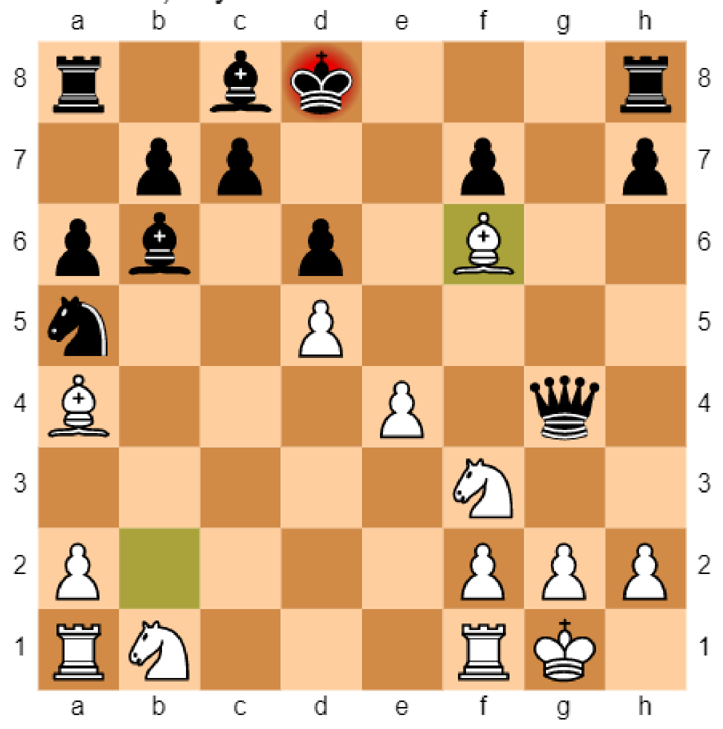


```
Move 3 White, Play 'h6f8':
```

```
, moves searched 10714
checkmate: White wins!
(True,
 'checkmate: White wins!',
 Board('r1b2Qk1/ppp2r1p/2np1qN1/2b1pp2/2B1P3/3P4/PPP2PPP/R3K2R b KQ - 1 2'))
```

Board 3:

Move 3 White, Play 'b2f6':



, moves searched 10910

```
checkmate: White wins!
(True,
 'checkmate: White wins!',
 Board('r1bk3r/1pp2p1p/pb1p1B2/n2P4/B3P1q1/5N2/P4PPP/RN3RK1 b - - 0 2'))
```

2 Why do we assume that we play against an optimal opponent in the minimax algorithm. What happens otherwise?
**Answer:**
The minimax algorithm can guarantee optimal outcome with the highest utility only if both we and our opponent plays optimally. The minimax algorithm finds the next best move by evaluating the possible utility gain in subsequent future moves down the line, this in effect requires the opponent to be the perfect adversary in order for minimax to select the best move sequence. If we have an irrational adversary, we can run into a case where minimax chooses a non optimal path since the problem with minimax is that the player only looks at its immediate successor's(opponent's) utility (which it assumes to be the optimal), and if the adversary plays irrationally, the minimax player could choose a suboptimal path.

3 What kind of node exploration is the minimax algorithm using? Depth-first or breadth-first?
**Answer:**
Minimax uses a depth-first search exploration.

4 What is the time complexity of the naive minimax algorithm? Prove it.
**Answer:**
The time complexity for naive minimax is $O(b^m)$. To prove it lets assume we have a search space of $b$, meaning $b$ possible moves per node. That means the number of searched nodes grows by a factor of $b$ (each successor node also has $b$ moves) as we go deeper down the tree. This growth happens until we reach terminal nodes, which can be arbitrarily be at some depth $m$. Thus by depth $m$ we have searched through $1 + b + b^2 + ... + b^m$ nodes, which makes our time complexity $O(b^m)$.

5 Explain why minimax algorithm with $\alpha - \beta$ pruning is more efficient than naive minimax. What is the complexity and why it depends on the ordering of the elements?

**Answer:**
$\alpha - \beta$ pruning is more efficient than naive minimax due to the fact that we reduce our search

complexity by pruning branches that we know are suboptimal paths. We essentially keep track of the best value for a player seen so far and compare it to other paths we search. If the other paths have worse utility returns then we prune that branch. The time complexity for it is $O(b^{m/2})$. It is dependent on the ordering of the elements because if we end up searching starting from the worst possible element to the best possible element, then we will get no pruning optimization, but if we try and start from the best element and work towards the worst element, we can definitely prune branches that are worse than our best element and thus saving some extra search time complexity.

6 Why did we introduce EVAL function instead of UTILITY for some games? Explain what is a good EVAL function for chess and how it affects the minimax algorithm.

**Answer:**
We introduce EVAL function instead of UTILITY for some games due to resource limits. For a game like chess where there is a large search space and a time limit as well, it is infeasible to have a UTILITY in which we calculate all possibilities and return a utility for the terminal states of the game. The EVAL function instead is introduced to estimate the desirability of a position at a certain point in the game. This helps algorithms reduce search depth (bc of memory and time constraints) and also estimate the quality of their position at the point the algorithm decides to cut off searching. A good EVAL function for chess is a linear weighted sum of features: $Eval(s) = w_1 f_1(s) + w_2 f_2(s) + ... + w_n f_n(s)$ where $w$ and $f(s)$ could be the value of each of the pieces and $f(s)$ is the difference between the number of your pieces and the number of your opponents pieces for each piece. How it affects the Minimax algorithm is that now with Cutoff-Test and EVAL, instead of searching to the terminal states, minimax will search up to some specified depth and run a Cutoff-Test where we evaluate the positions at the depth of where we cut off using EVAL to which minimax will use the computed quality from EVAL to make a decision.

7 What is a Horizon effect and Quiescence?
**Answer:**
The Horizon effect is the problem in which we are unable to see what adverse scenarios lie beyond our search level. An example would be the possibility of only delaying adverse moves without making any forward progress IE putting opponents repeatedly in check but not checkmate. This is due to the fact that with Cutoff-Test and EVAL, we have a limited search depth, so we don't know what will happen past that depth.

Quiescence is the idea of trying to further explore unstable board positions to see if game changing moves lie ahead. Using Quiescence search can slightly alleviate the Horizon effect by doing a deeper search in unstable nodes until we reach a nodes of stable state, this is to avoid possible blunders of a fixed depth search. Stability is defined with respect to the fluctuations in evaluated game state values going down some decision path.

8 Under what kind of transformation the behaviour of minimax algorithm is preserved in case of a game with no chance nodes? In case of a game with chance nodes?
**Answer:**
In a game with no chance nodes any monotonic transformation(order preserving transformation) preserves the behavior of minimax.

In a game with chance nodes a positive linear transformation preserves the behavior of minimax.

9 A and B are both true.
**Answer:**
P = A is true
Q = B is true
$P \wedge Q$

10 If A is true, then B must be true as well.
**Answer:**
P = A is true
Q = B is true
$P \implies Q$

11 If a student studies for a test, they will do well on it. We can also tell that if a student did well on a test, then they must have studied for it.
**Answer:**
P = A Student studies for a test
Q = A Student does well on a test

$P \iff Q$

12 If a student is completely dry and it is raining outside, it is because they have an umbrella or a hoodie and it is not raining heavily
**Answer:**
P = A student is completely dry
Q = It is raining outside
R = A student has an umbrella
S = A student has a hoodie
T = It is not raining heavily

$P \wedge Q \implies (R \vee S) \wedge T$

13 Simplify and translate the following propositional logic sentence into English: $A \vee (A \wedge B) \iff \neg(A \wedge B \wedge C)$
**Answer:**
$A \vee (A \wedge B) \iff \neg(A \wedge B \wedge C)$ simplifies to $A \iff \neg A \vee \neg B \vee \neg C$ (Using truth tables and DeMorgan's Law). Translation: A is true if and only if A is not true or B is not true or C is not true.

14 Is the following sentence valid? $A \vee B$
**Answer:**
It is not valid, a counter example would be A is False B is False. In this case $A \vee B$ evaluates to False.

15 Is the following sentence satisfiable? $A \implies B$
**Answer:**
Yes, we can consider the case A = true, B = true.
In this case $A \implies B$ is true.

16 Is the following sentence unsatisfiable? $(A \wedge (B \vee C)) \wedge ((A \wedge B) \vee (A \wedge C))$
**Answer:**
No, it is satisfiable. We can examine the case in which A = True, B = True, and C = True. By plugging these in we can get the expression: $(true \wedge (true \vee true)) \wedge ((true \wedge true) \vee (true \wedge true))$. This expression clearly evaluates to true.