Lecture 7 - Games (Minimax)

Games vs. Search Problems:

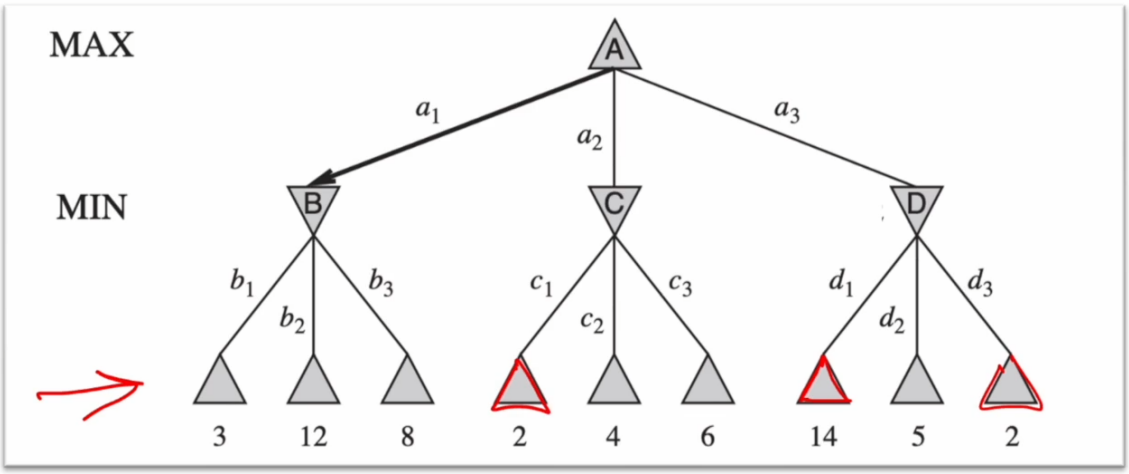
- Environment characteristics:
  - The games' environment is fully observable
  - Discrete
  - Deterministic / stochastic, meaning that there is no randomness involved.
  - We are no longer in a benign environment - it is adversarial because the opponent(s) are actively trying to beat us to win the game.
- Unpredictable opponent: Specifying a move for every possible opponent reply
- Time limits: If unlikely to find goal within a time limit, then we should approximate.

Two player zero-sum game: What is good for one opponent is equally as bad for the other. There is no win-win outcome.

The two players are referred to as the MAX and the MIN. We can formally define a game as the following:

```
initial_state() ## How the game is set up at the start
to_move(s) ## The player whose turn it is to move in a state s
actions(s) ## The set of legal moves in state s
result(s,a) ## defines the state resulting from taking an action a in state s
is_terminal(s) ## True when the game is over at state s, like goal_test
utility(s) ## Function which defines the final numerical value when the game
ends in a terminal state s
```

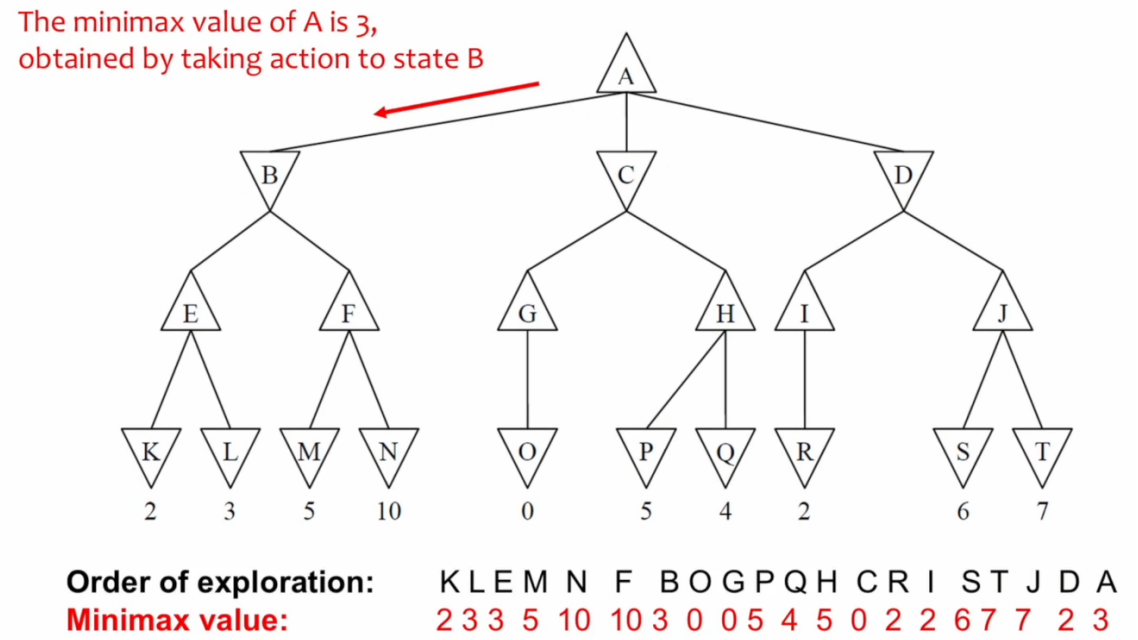
In the case of chess, the utility can either be +1 (MAX wins), −1 (MAX loses, MIN wins) or 0 (draw). The objective of MAX is to maximize the utility, and the objective of MIN is to minimize the utility. Consider the trivial game below:



The tree is one move deep (for each player), meaning that it is a 2-ply game. The objective of MAX is to go where the utility is highest (14), and MIN would be to go to 2.

If we start at the root, what choice should be made? The minimax search algorithm assigns a minimax value to every node. This is the optimal value a player can achieve at that node. What is minimax(A)? We need to first get minimax(B), minimax(C) and minimax(D).

Since those are all MIN values, we would get minimax$(B)=3$, minimax$(C)=2$ and minimax$(D)=2$. Then, we take the maximum of these three to get minimax$(A)=3$. So it would go with a_1 in this game. Recursive calling. What is the order of the nodes in which minimax values are assigned? It would be at the left-most bottom, then the parent. See below:



This is like **DFS**. It is a complete depth-first exploration of the game tree. It could be implemented with either a stack or recursive calling. This is the recursive pseudo-code:

Minimax(s):

$$\begin{cases} \text{utility}(s, \text{MAX}) & \text{if is-terminal}(s) \\ \max_{a \in \text{actions}(s)} \text{Minimax}(\text{result}(s,a)) & \text{if to-move}(s) = \max \\ \min_{a \in \text{actions}(s)} \text{Minimax}(\text{result}(s,a)) & \text{if to-move}(s) = \min \end{cases}$$

Let's look at the properties of the minimax search:

- Complete? Yes, if the tree is finite
- Optimal? Yes, against the optimally playing opponent
- Time complexity? $O(b^m)$
- Space complexity? $O(bm)$

For chess, $b=35$ and $m=100$ for a reasonable game. This is huge. We want to improve the time-comlexity. Two problems that are completed on paper.