Alpha-Beta Pruning

2/8/17
Backward Induction and Min/Max

Key Idea:

• Recursively determine the outcome that results from each available action.
• Return the best outcome for the current player.

These algorithms give optimal actions if they can explore the entire game tree.

• If the tree is too large, we impose a depth bound.
• At the bound, we guess what outcome will result.
Backward Induction Exercise

Diagram of a decision tree with various outcomes at each node.
Min/Max Search Exercise

```
  +
 / \
-   -
  +   +
 /   /   /
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
 |   |   |
+   +   +
```

9 1 1 -3 9 2 -3 8 0 8 -4 8 -8 4 9 2
Pruning Illustrated
Key idea:
- Keep track of upper and lower bounds on the value
- If the bounds cross, the state being examined will never be reached under optimal play.

\[
\begin{align*}
V &= 4 \\
4 \leq V &\leq 10 \\
-10 \leq V &\leq 10 \\
\end{align*}
\]

\[
\begin{align*}
V &= 5 \\
\end{align*}
\]

\[
\begin{align*}
V &= 4 \\
\end{align*}
\]

\[
\begin{align*}
V &= -1 \\
\end{align*}
\]

\[
\begin{align*}
? &
\end{align*}
\]
function alpha_beta(state, UB, LB, depth)
    if depth limit or end of game reached:
        return value(state)
    best_val = -MAX if maximizing else MAX
    for each action available in state:
        next_state = make_move(state, action)
        val = alpha_beta(next_state, UB, LB, depth+1)
        if player is maximizer:
            if val >= UB: return val
            LB = max(val, LB)
            best_val = max(val, best_val)
        else:
            if val <= LB: return val
            UB = min(val, UB)
            best_val = min(val, best_val)
    return best_val

Difference from ordinary Min/Max
Alpha-Beta Pruning Exercise

```
+  
-   +   -
+   +    +
-1 0 2  -3 -2  
```
This game is identical to the last one, except that I’ve swapped the location (and thus the implied search order) of a few nodes.

The result: nothing gets pruned!
Discussion Questions

1. How can we organize the search to maximize the amount of work saved by pruning?

2. How can we modify alpha/beta pruning to work on non-zero-sum games?

3. How can we modify alpha/beta pruning to work on 3-(or more)-player games?