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

6,2,2
9,3,2
2,6,3
3,4,7
6,9,5
9,8,3
5,6,4
0,7,6
2,3,6
0,2,0
4,5,3
2,8,0
2,2,7
8,5,4
Min/Max Search Exercise
Pruning Illustrated

Diagram:

Left:
- Root: +
  - Left: -
    - Left: 10
    - Right: 10
  - Right: 10

Right:
- Root: +
  - Left: -
    - Left: 4
    - Right: 5
  - Right: -
    - Left: 4
    - Right: -1
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.

**Alpha-Beta Pruning**

```
-10 ≤ V ≤ 10
4 ≤ V ≤ 10

V = 4
4 ≤ V ≤ 10
-10 ≤ V ≤ 10

-10 ≤ V ≤ 4
-10 ≤ V ≤ 5
-10 ≤ V ≤ 10

V = 5
V = 4
V = -1

4 ≤ V ≤ -1
4 ≤ V ≤ 10
```
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
Alpha-Beta Pruning Example