Alpha-Beta Pruning

2/8/17

Backward Induction and Min/Max

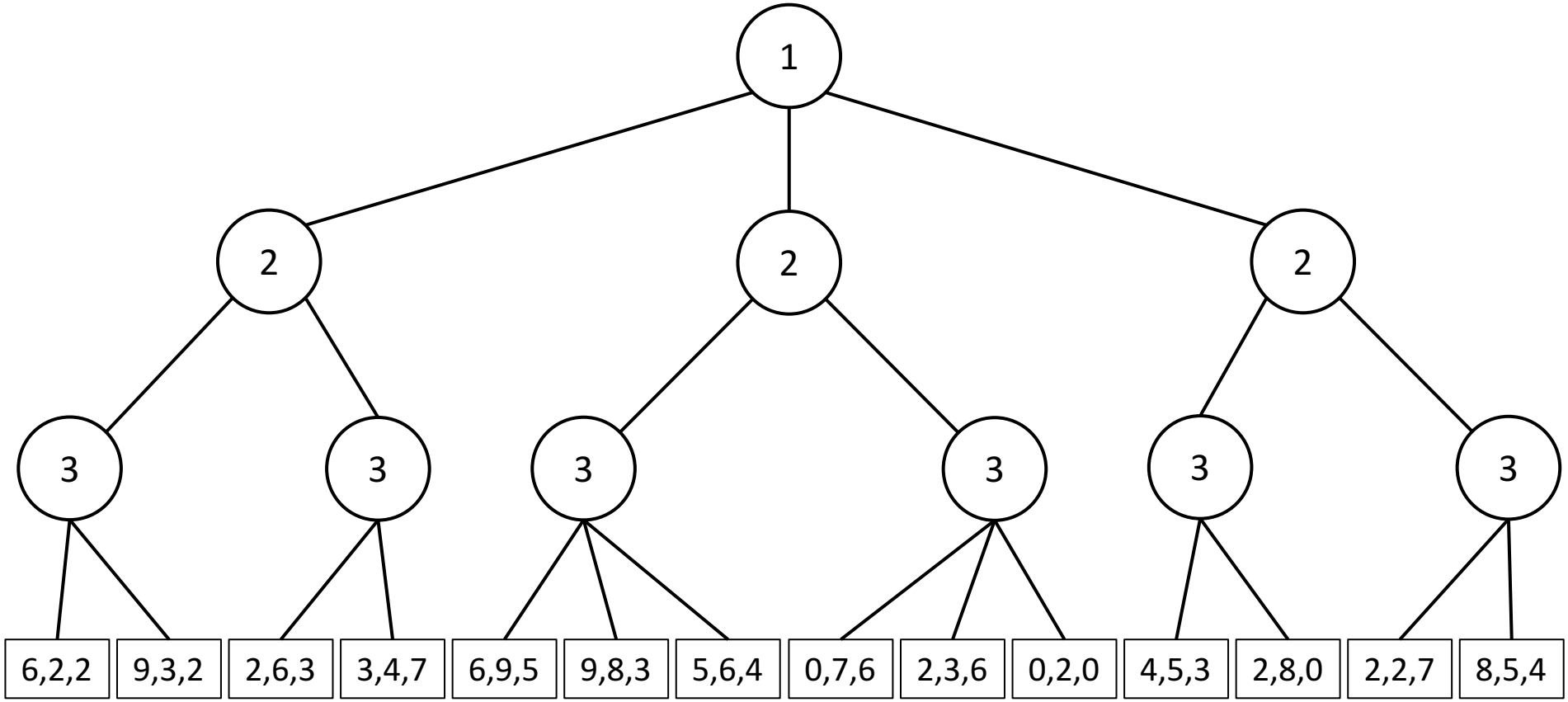
Key Idea:

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

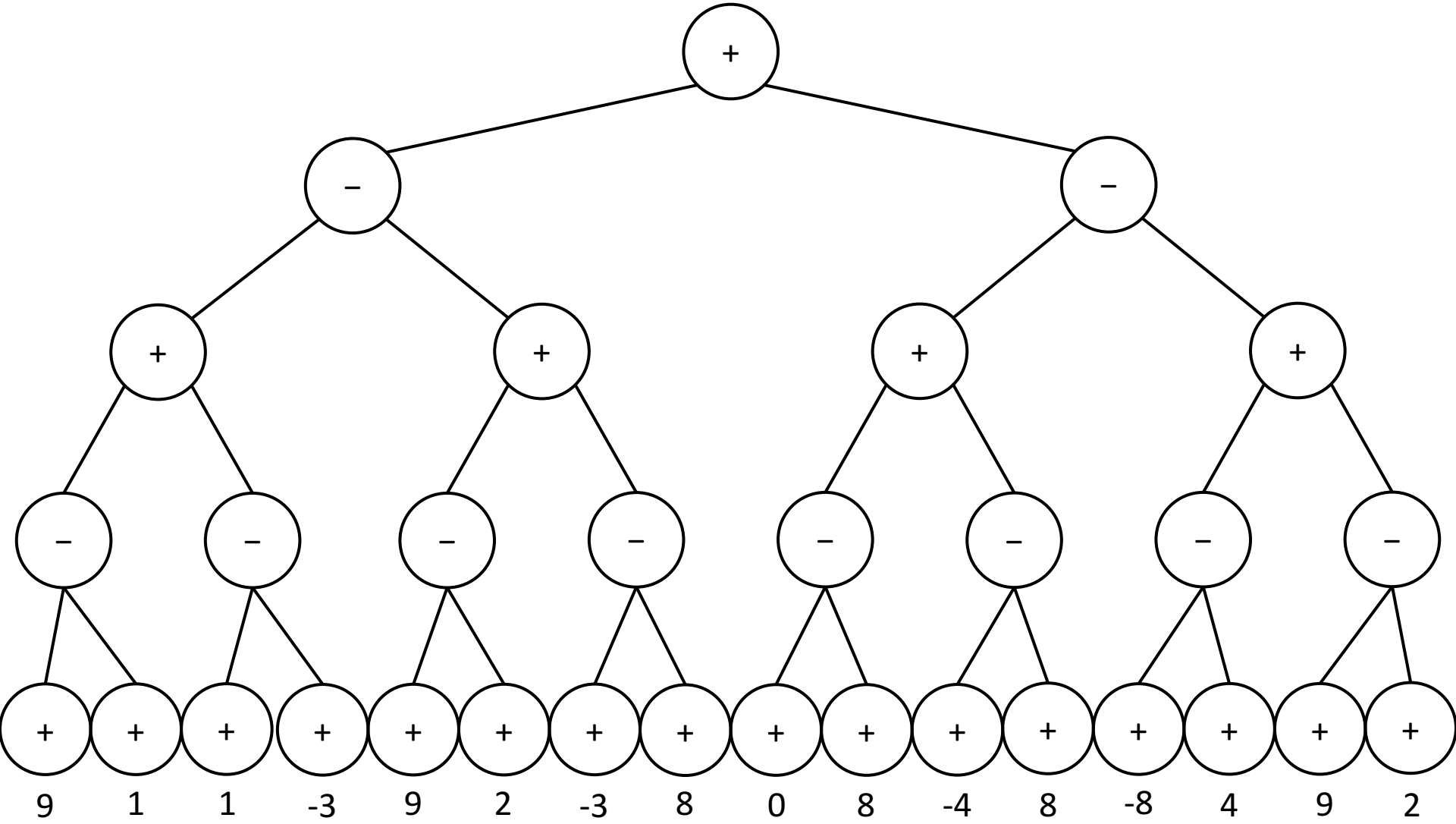
These algorithms give optimal actions if they can explore the entire game tree.

- If the tree is too large, we impose a depth bound.
- At the bound, we guess what outcome will result.

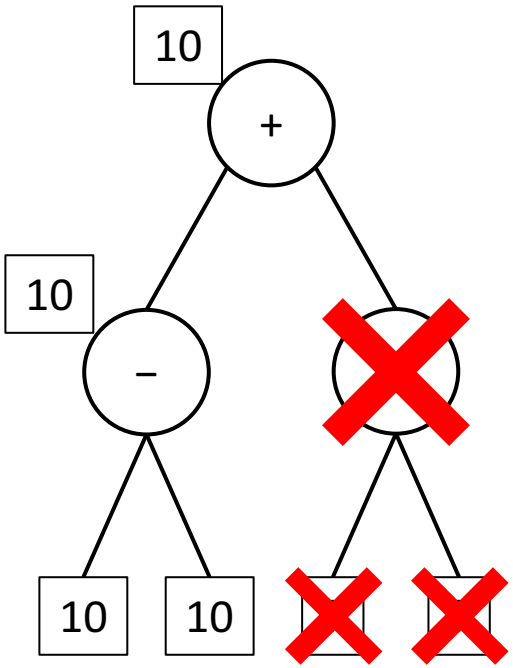
Backward Induction Exercise

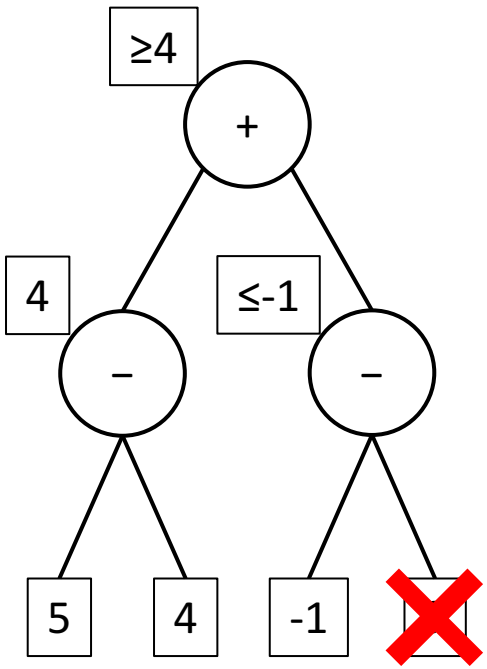


Min/Max Search Exercise



Pruning Illustrated

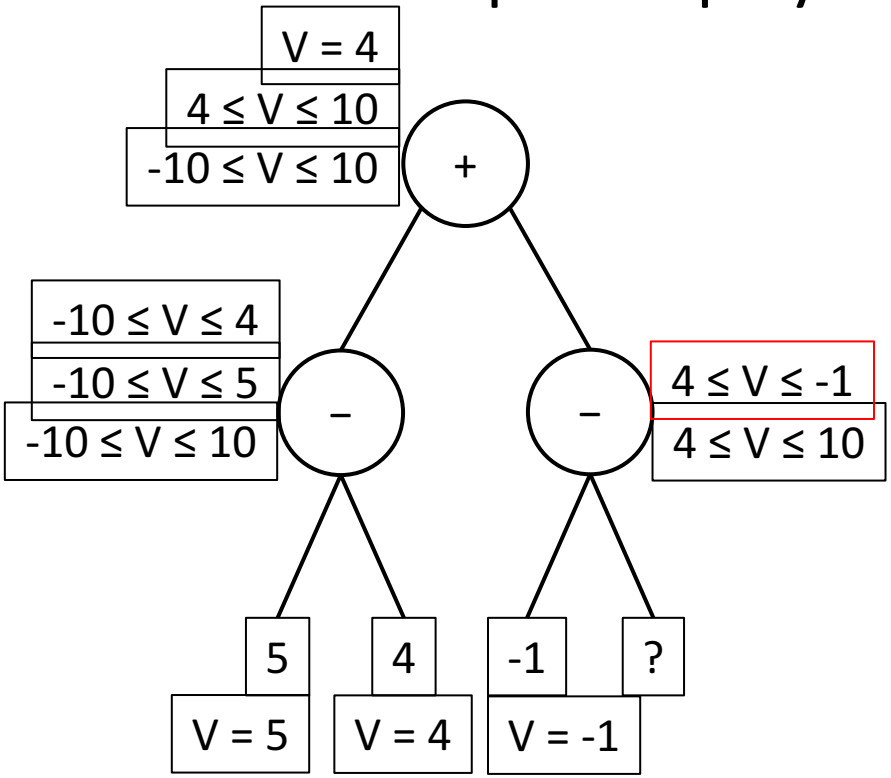




Alpha-Beta Pruning

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 Pseudocode

```
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 Example