Games: Efficiency and more

Andrea Danyluk
February 20, 2017

Announcements

- Programming Assignment 1: Search
  - Due tomorrow
- Code review sign-up
- Programming Assignment 2 posted

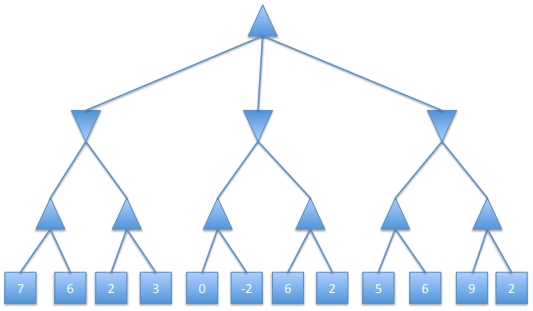
Today's Lecture

- Making minimax more efficient:
  - α-β pruning

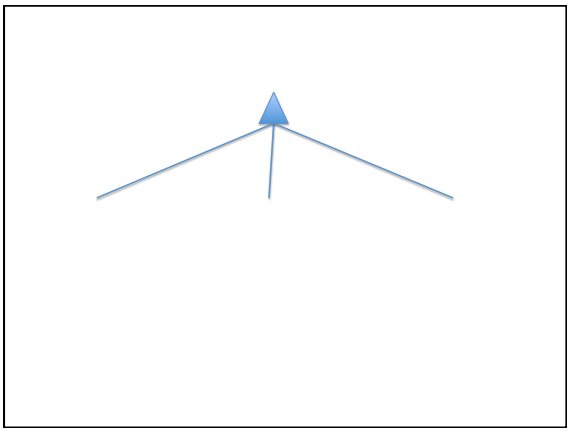
Minimax Reality

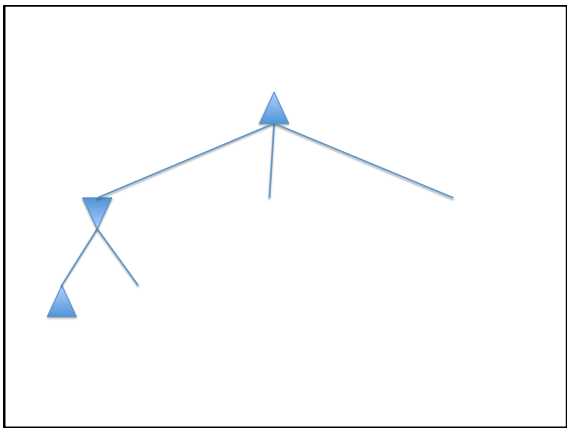
- Can rarely explore entire search space to terminal nodes.
- Choose a depth cutoff – i.e., a maximum ply
- Need an evaluation function
  - Returns an estimate of the expected utility of the game from a given position
  - Must be efficient to compute
    - Trading off plies for heuristic computation
    - More plies makes a difference
- Consider iterative deepening

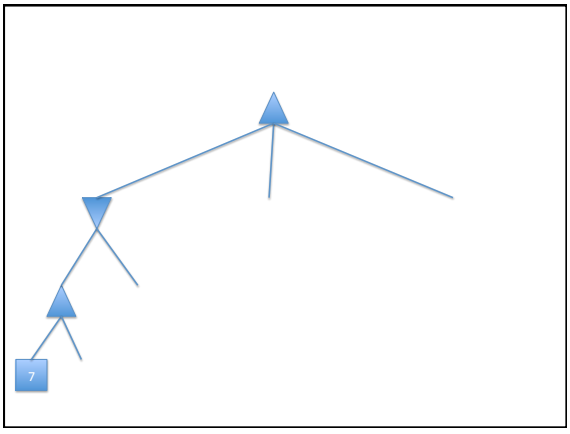
An earlier example revisited

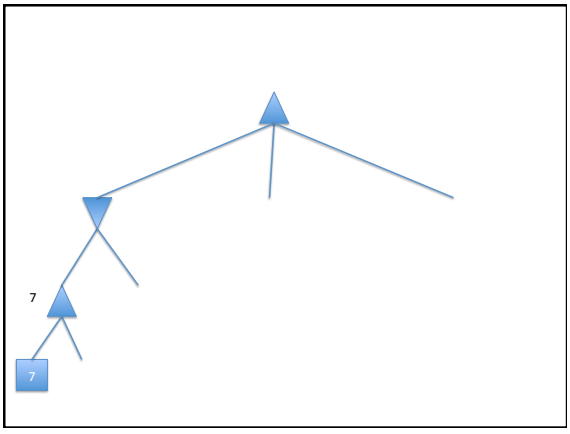


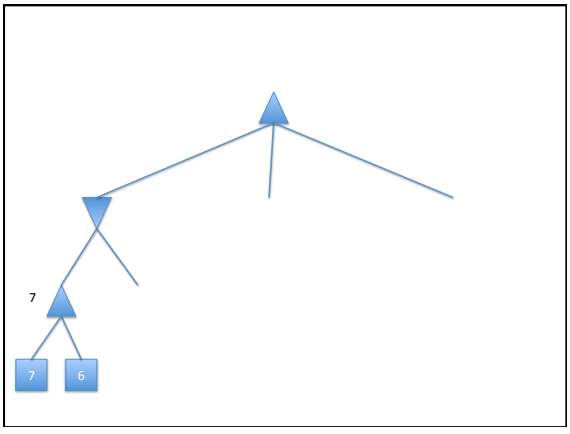
An earlier example revisited

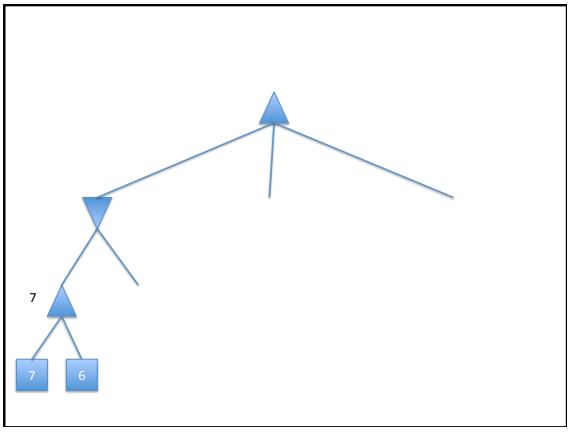


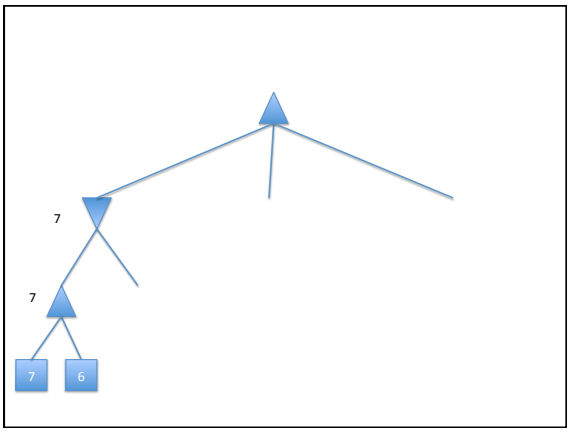


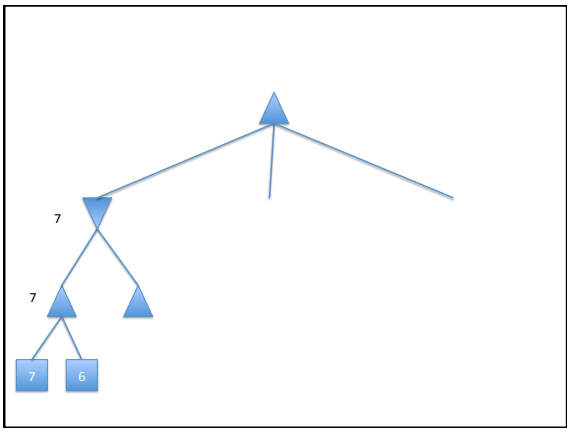

7

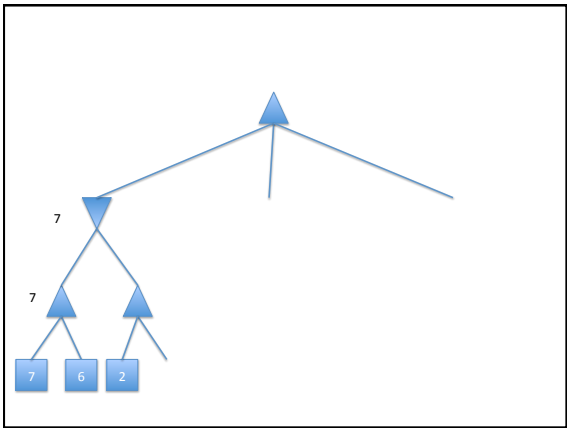

7
7

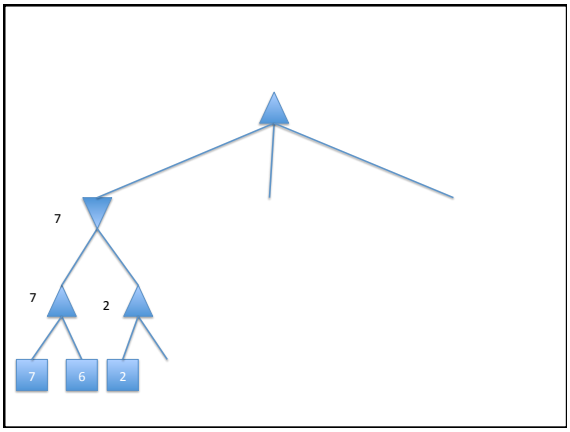

7
7
6

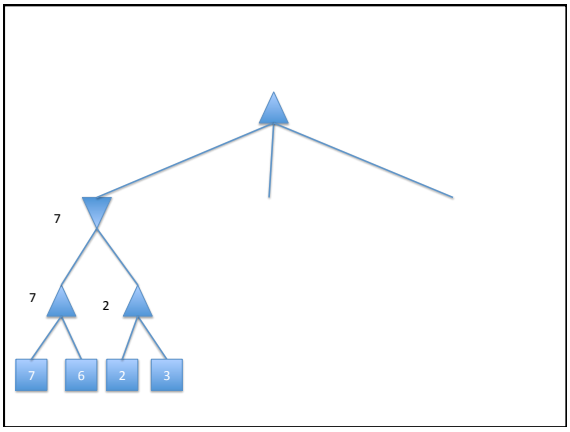

7
7
6

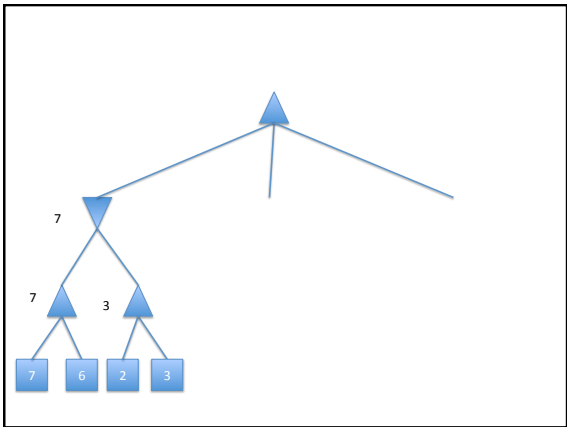

7
7
7
6

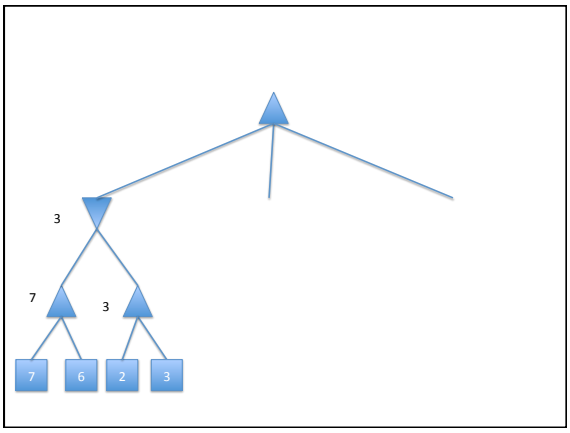

7
7
7
6

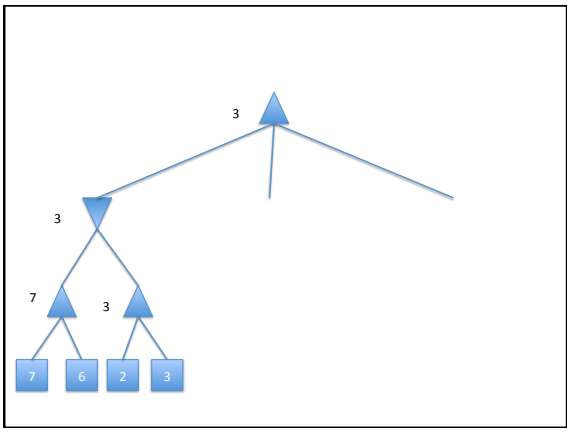

7
7
7
6
2

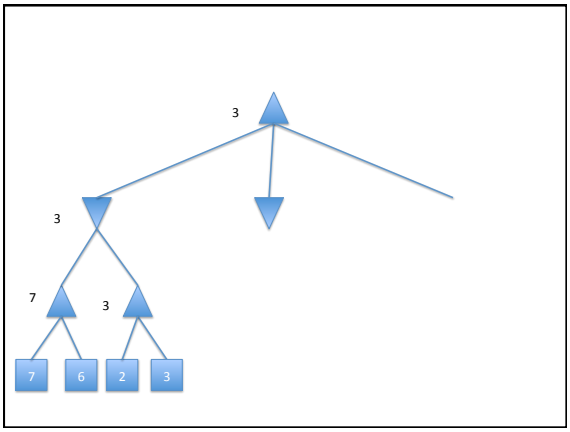

7
7
2
7
6
2

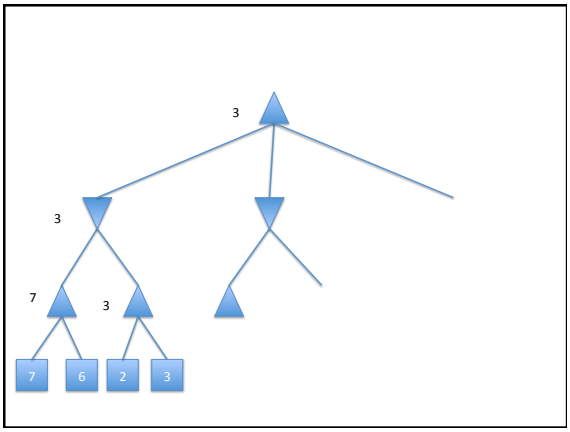

7
7
2
7
6
2
3

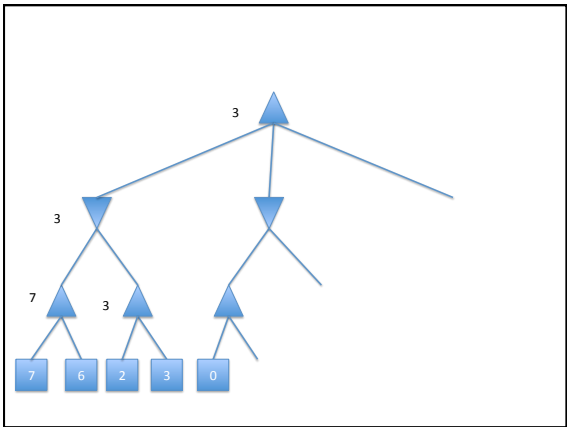

7
7
3
7
6
2
3

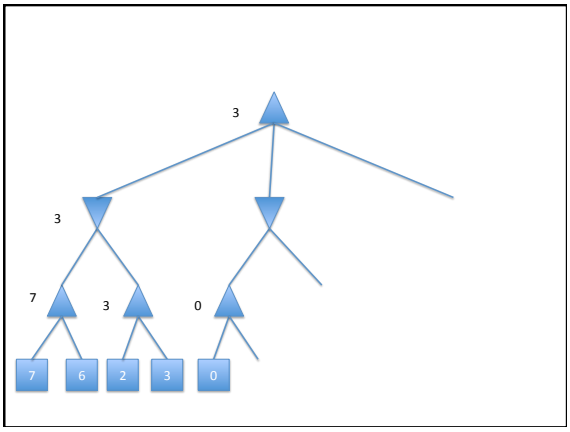

3
7
3
7
6
2
3

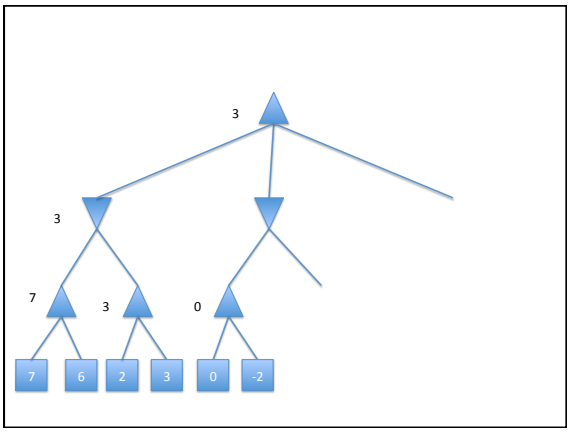

3
3
7
3
7
6
2
3

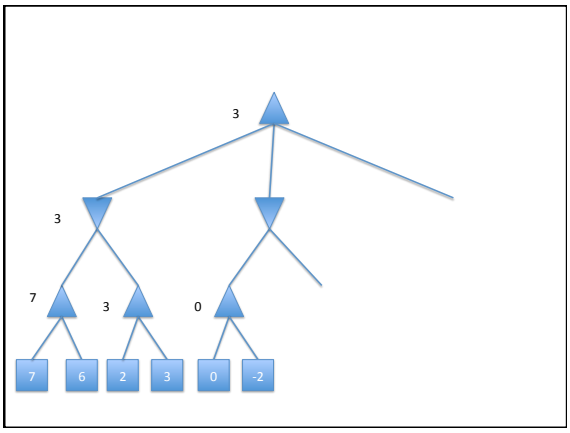

3
3
7
3
7
6
2
3

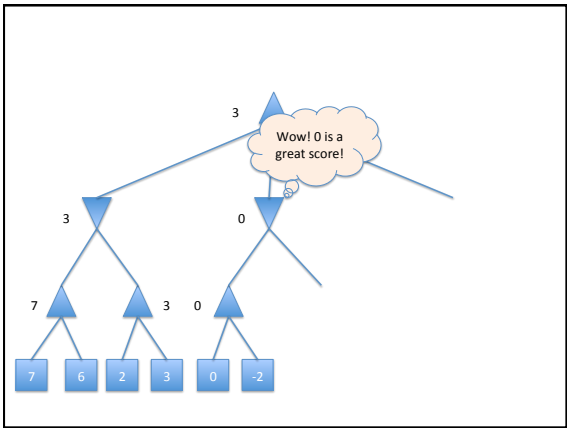

3
3
7
3
7
6
2
3

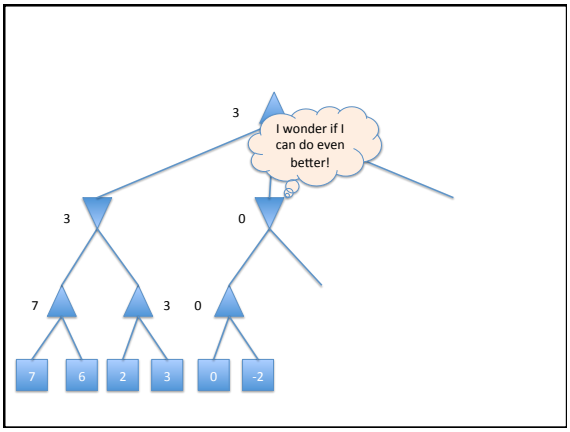

3
3
7
3
7
6
2
3
0

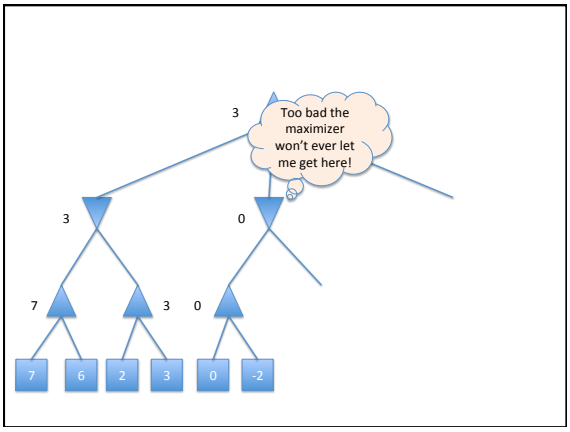

3
3
7
3
0
7
6
2
3
0

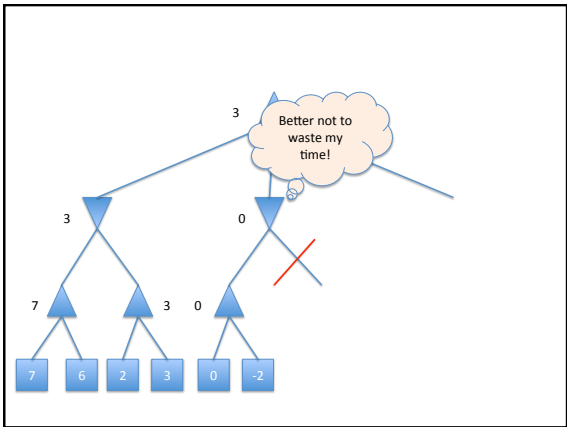

3
3
7
3
0
7
6
2
3
0
-2

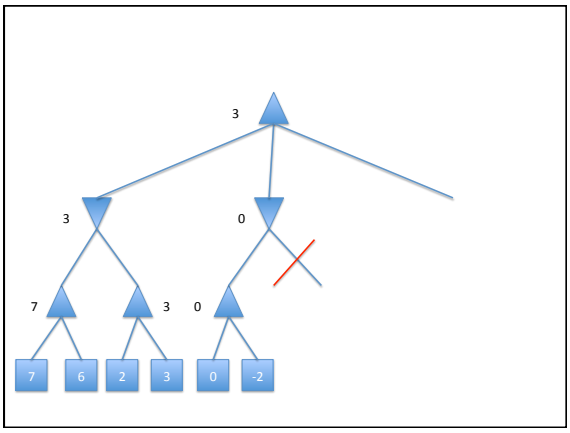

3
3
7
3
0
7
6
2
3
0
-2

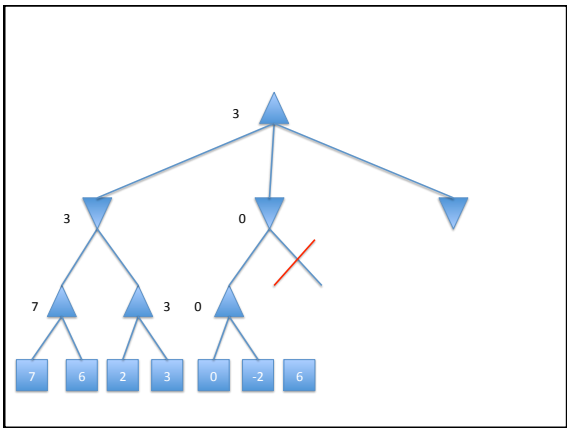

3
Wow! 0 is a great score!
3
0
7
3
0
7
6
2
3
0
-2

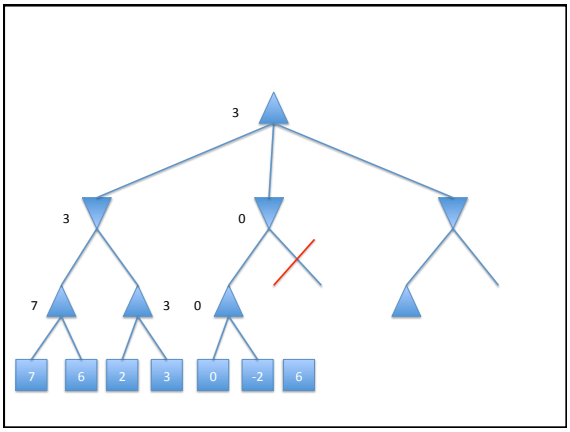

3
I wonder if I can do even better!
3
0
7
3
0
7
6
2
3
0
-2

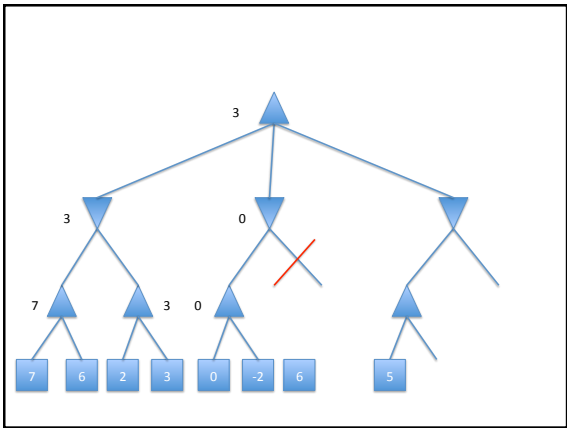

3
Too bad the maximizer won't ever let me get here!
3
0
7
3
0
7
6
2
3
0
-2

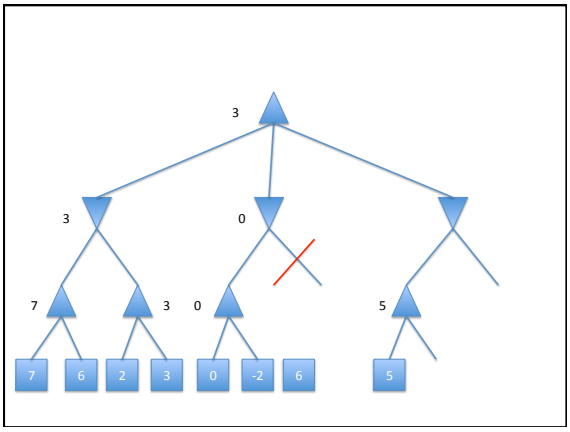

3
Better not to waste my time!
3
0
7
3
0
7
6
2
3
0
-2

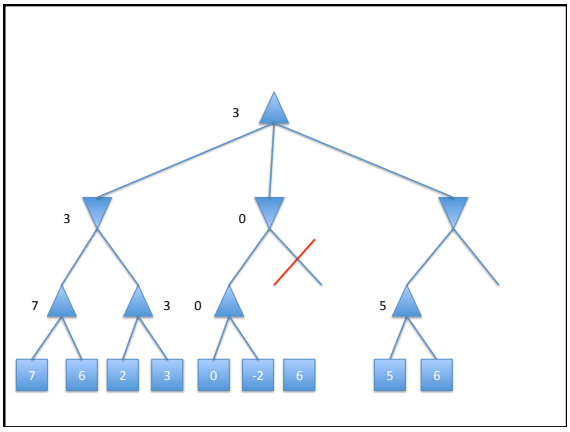

3
3
0
7
3
0
7
6
2
3
0
-2

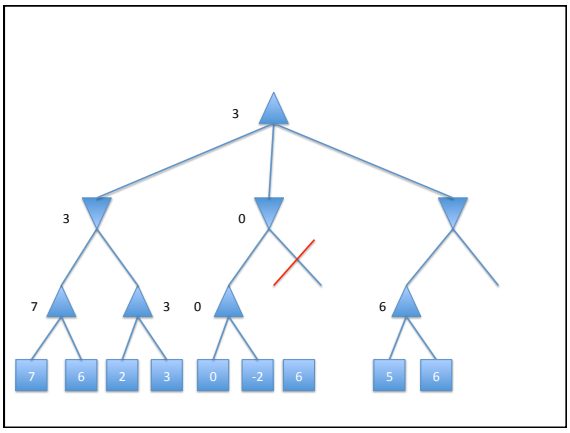

3
3
0
7
3
0
7
6
2
3
0
-2
6

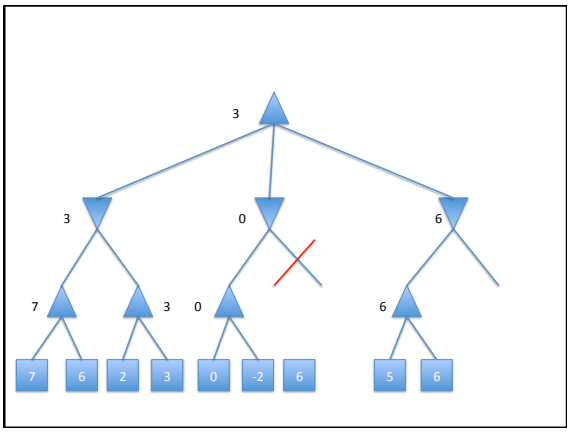

3
3
0
7
3
0
7
6
2
3
0
-2
6

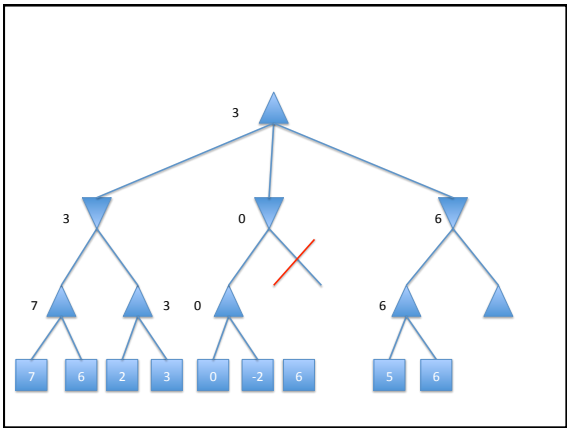

3
3
0
7
3
0
7
6
2
3
0
-2
6
5

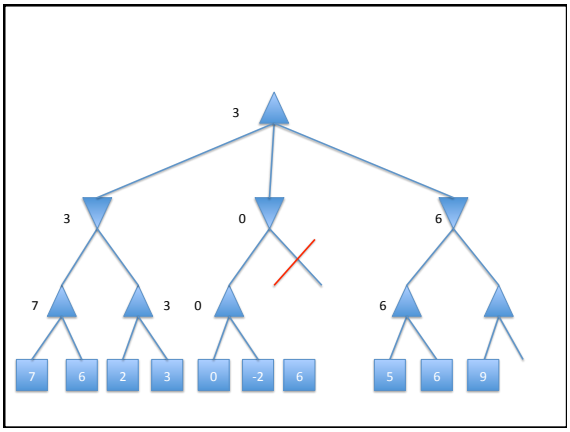

3
3
0
7
3
0
5
7
6
2
3
0
-2
6
5

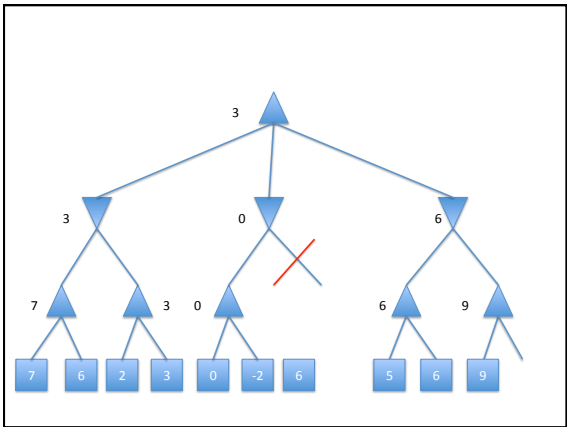

3
3
0
7
3
0
5
7
6
2
3
0
-2
6
5
6

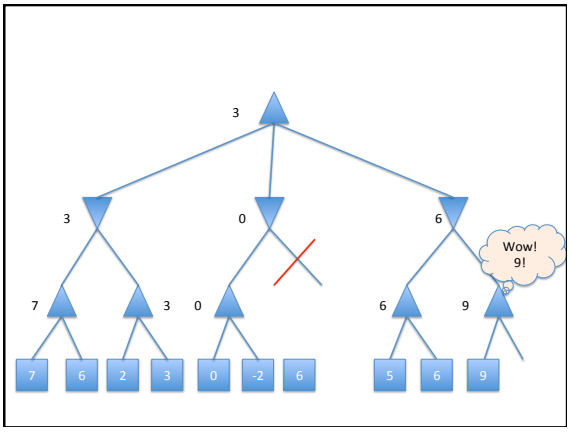

3
3
0
7
3
0
6
7
6
2
3
0
-2
6
5
6

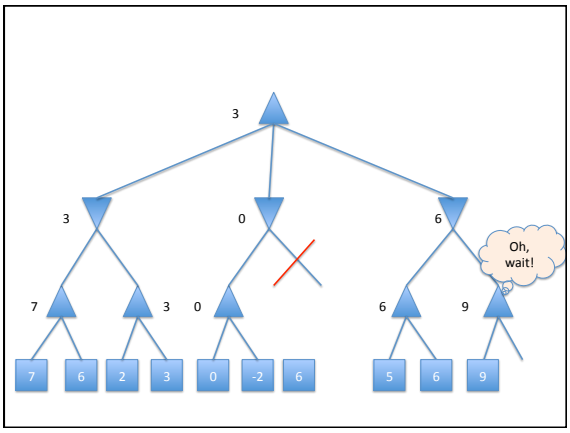

3
3
0
6
7
3
0
6
7
6
2
3
0
-2
6
5
6

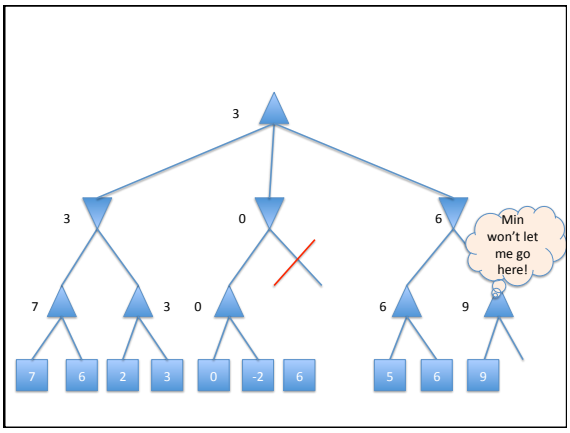

3
3
0
6
7
3
0
6
7
6
2
3
0
-2
6
5
6

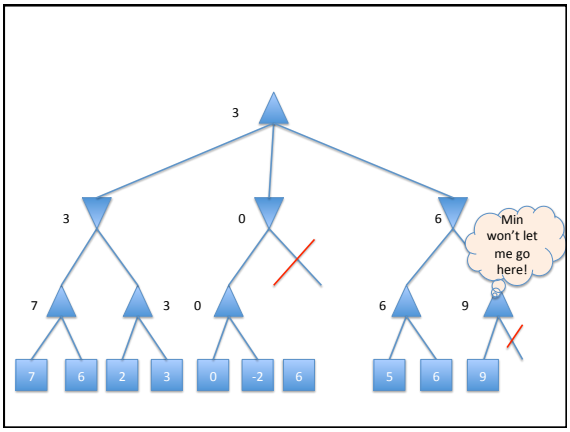

3
3
0
6
7
3
0
6
7
6
2
3
0
-2
6
5
6
9

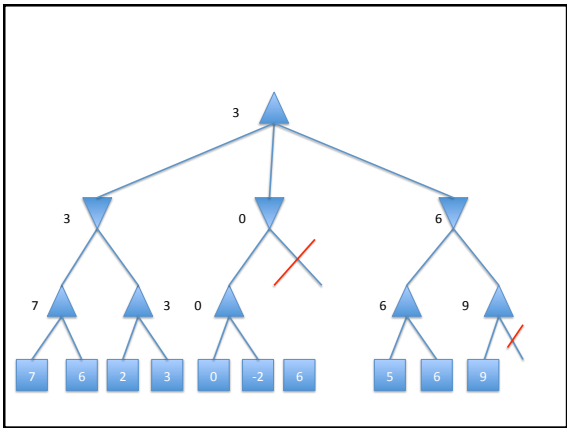

3
3
0
6
7
3
0
6
9
7
6
2
3
0
-2
6
5
6
9

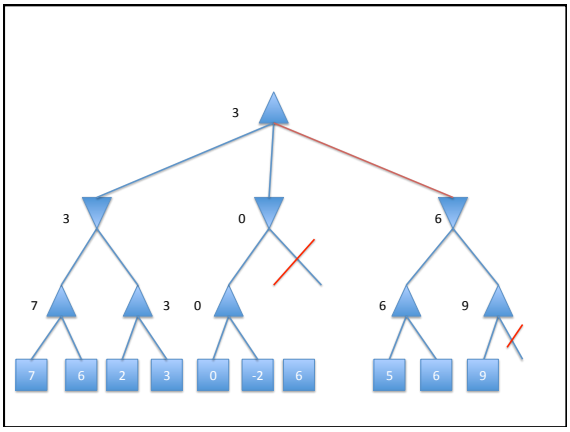

3
3
0
6
Wow!
9!
7
3
0
6
9
7
6
2
3
0
-2
6
5
6
9











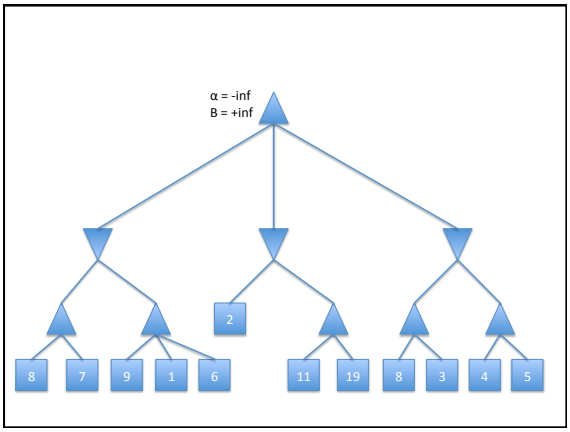
α-β Pruning

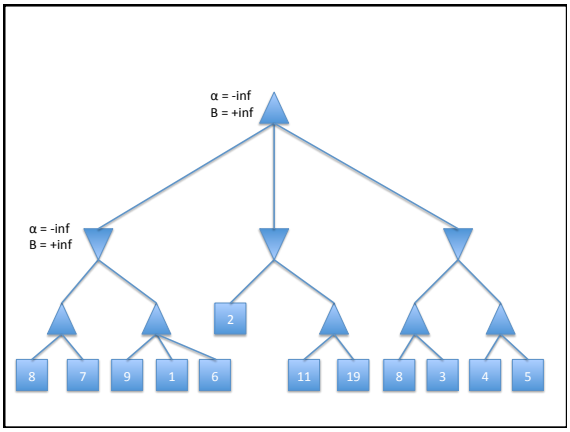
- If something looks too good to be true, it probably is.
- One example of the class of branch and bound algorithms with two bounds
  - α : the value of the best choice for Max
  - β : the value of the best choice for min

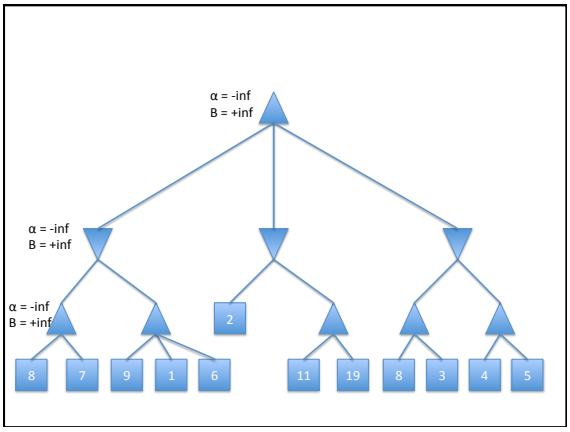
α-β Pruning

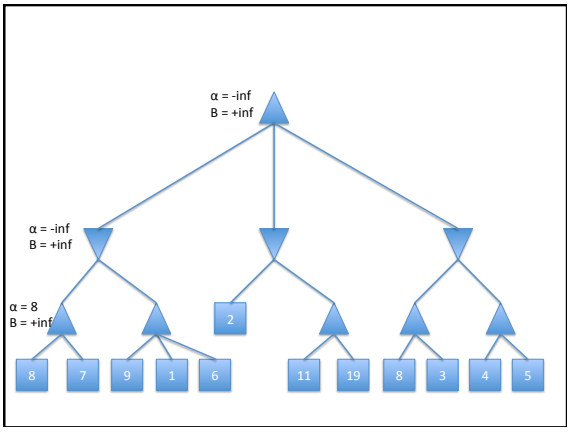
- Given these two bounds
  - α : the value of the best choice for Max
  - β : the value of the best choice for min
- Basic idea of the algorithm
  - On a minimizing level, if you find a value < α, cut the search
  - On a maximizing level, if you find a value > β, cut the search

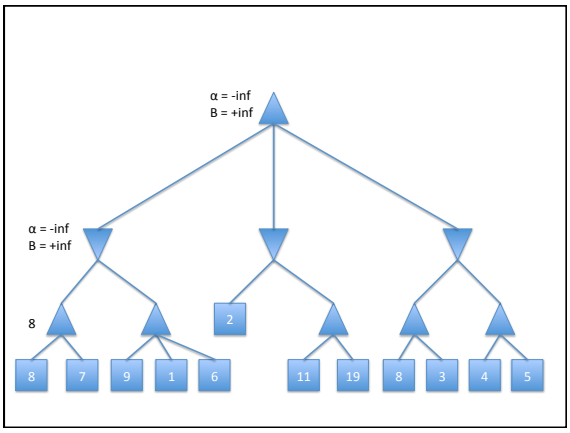
```
function αβSEARCH(state) returns an action a
   v = MAX-VALUE(state, -infinity, +infinity)
   return action a in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value v
   if TERMINAL-TEST(state) then return UTILITY(state)
   v = -infinity
   for each a in ACTIONS(state) do
      v = MAX(v, MIN-VALUE(RESULT(state, a), α, β))
      if v ≥ β then return v
      α = MAX(α, v)
   return v

function MIN-VALUE(state, α, β) returns a utility value v
   if TERMINAL-TEST(state) then return UTILITY(state)
   v = infinity
   for each a in ACTIONS(state) do
      v = MIN(v, MAX-VALUE(RESULT(state, a), α, β))
      if v ≤ α then return v
      β = MIN(β, v)
   return v
```

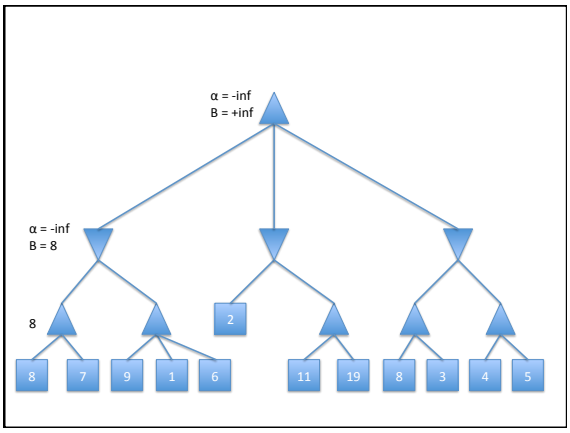


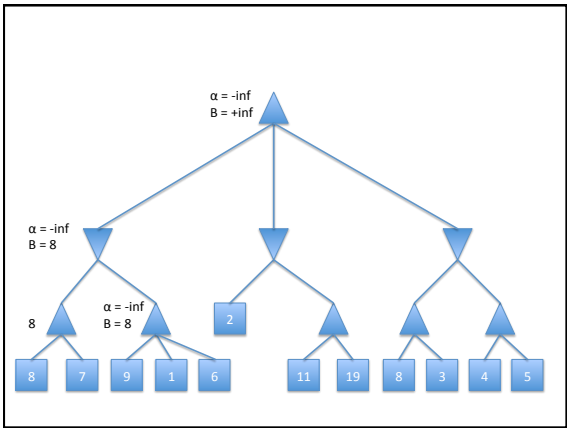


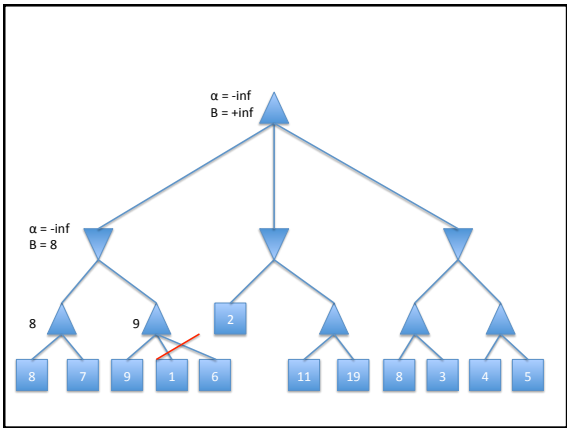


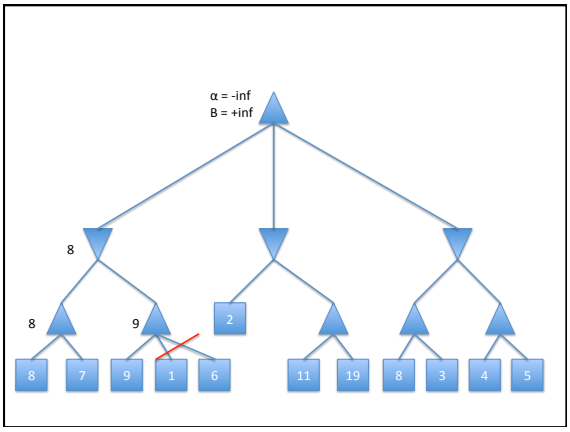


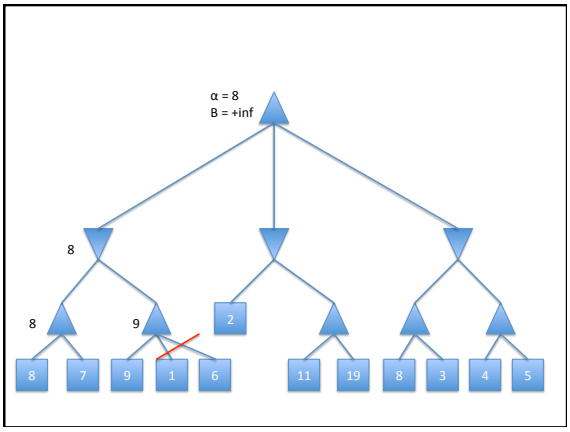

α = -inf
B = +inf
α = -inf
B = +inf
8
2
8 7 9 1 6 11 19 8 3 4 5

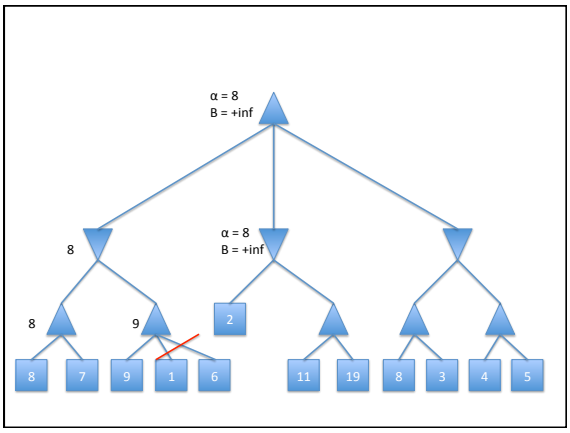

α = -inf
B = +inf
α = -inf
B = 8
8
2
8 7 9 1 6 11 19 8 3 4 5

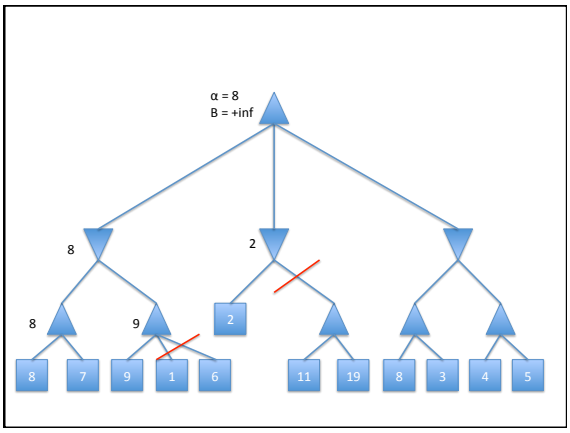

α = -inf
B = +inf
α = -inf
B = 8
8
α = -inf
B = 8
2
8 7 9 1 6 11 19 8 3 4 5

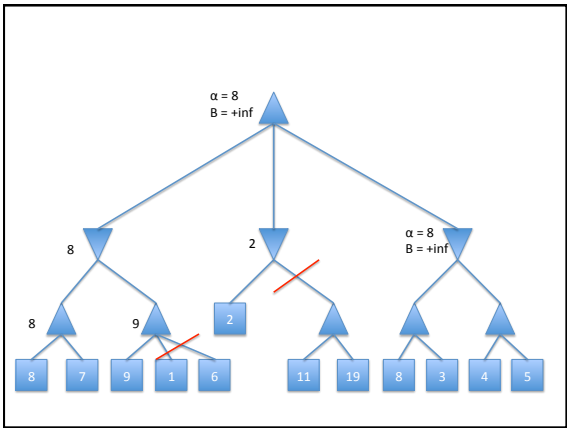

α = -inf
B = +inf
α = -inf
B = 8
8
9
2
8 7 9 1 6 11 19 8 3 4 5

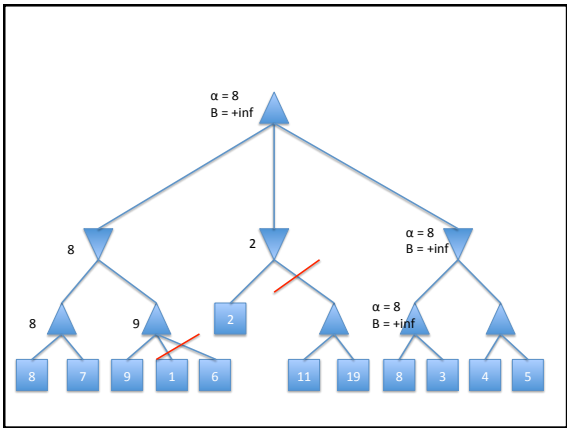

α = -inf
B = +inf
8
8
9
2
8 7 9 1 6 11 19 8 3 4 5

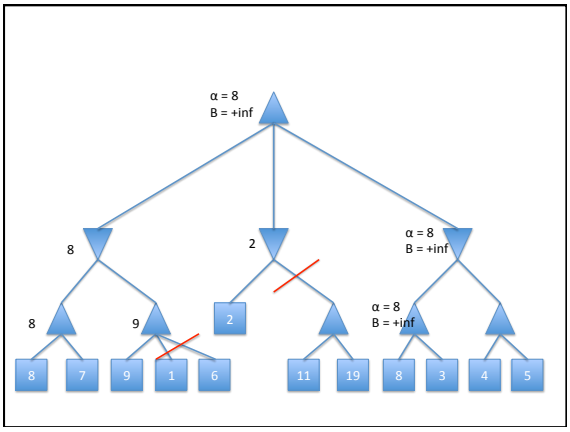

α = 8
B = +inf
8
8
9
2
8 7 9 1 6 11 19 8 3 4 5

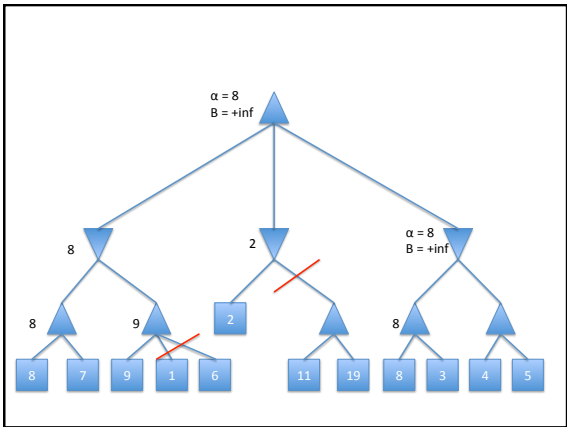

α = 8
B = +inf
8
α = 8
B = +inf
8
9
2
8
7
9
1
6
11
19
8
3
4
5

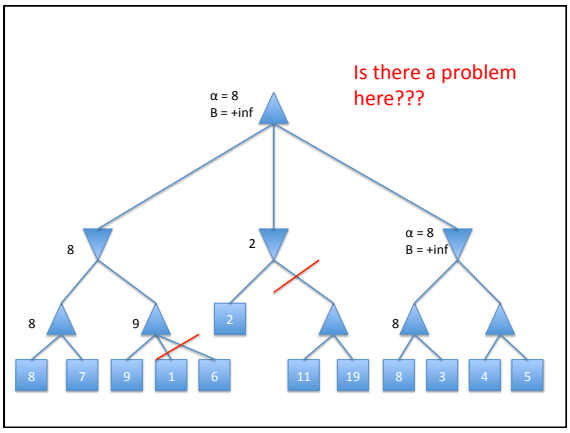

α = 8
B = +inf
8
2
8
9
2
8
7
9
1
6
11
19
8
3
4
5


α = 8
B = +inf
8
2
α = 8
B = +inf
8
9
2
8
7
9
1
6
11
19
8
3
4
5


α = 8
B = +inf
8
2
α = 8
B = +inf
8
9
2
α = 8
B = +inf
8
7
9
1
6
11
19
8
3
4
5

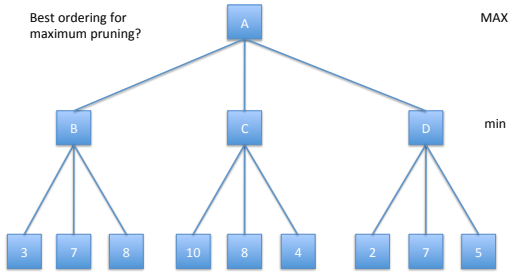

α = 8
B = +inf
8
2
α = 8
B = +inf
8
9
2
α = 8
B = +inf
8
7
9
1
6
11
19
8
3
4
5


α = 8
B = +inf
8
2
α = 8
B = +inf
8
9
2
8
8
7
9
1
6
11
19
8
3
4
5




```
function αβSEARCH(state) returns an action a
   v = MAX-VALUE(state, -infinity, +infinity)
   return action a in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value v
   if TERMINAL-TEST(state) then return UTILITY(state)
   v = -infinity
   for each a in ACTIONS(state) do
      v = MAX(v, MIN-VALUE(RESULT(state, a), α, β))
      if v ≥ β then return v
      α = MAX(α, v)
   return v

function MIN-VALUE(state, α, β) returns a utility value v
   if TERMINAL-TEST(state) then return UTILITY(state)
   v = infinity
   for each a in ACTIONS(state) do
      v = MIN(v, MAX-VALUE(RESULT(state, a), α, β))
      if v ≤ α then return v
      β = MIN(β, v)
   return v
```

Node Order Matters



Move Ordering

- Can we order moves in such a way that α-β will prune more rather than less?
- Chess?
- Connect 4?
- Don't worry about this for Pacman assignment

Properties of α-β

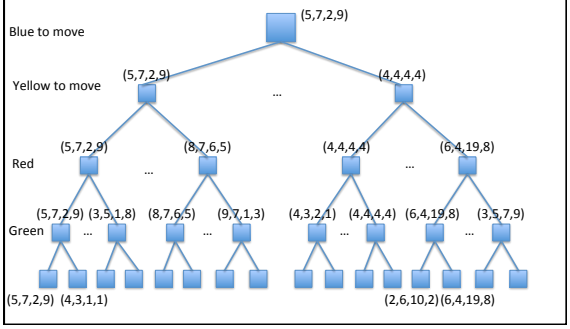
- Pruning does not affect the final minimax *value* at the root; but be careful about comparisons to guarantee best *action* is selected
- Good move ordering improves effectiveness of pruning
- If search depth is d, what is the time complexity of minimax?
  - $O(b^d)$
- With perfect pruning, can get down to $O(b^{d/2})$
  - Doubles solvable depth

[Adapted from Russell]

Games with > 2 players

- Up to 4 players
- Players try to place all 21 of their pieces
- Hope to block opponents from placing their pieces

A Four-Player Game



Multi-Player Games

- Evaluation function returns a vector of utilities
- Each player chooses the move that maximizes its utility.

- Is Pacman with 2 ghosts a multi-player game?