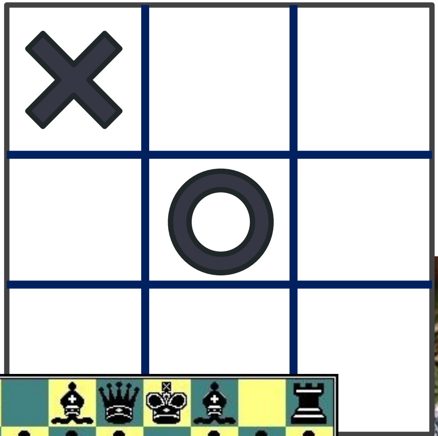
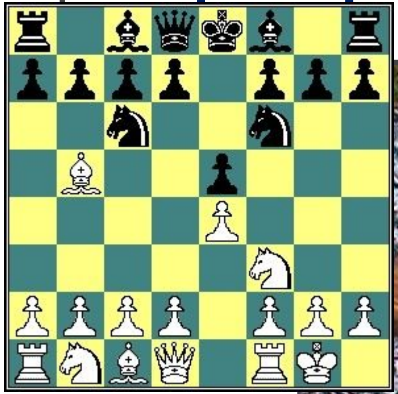
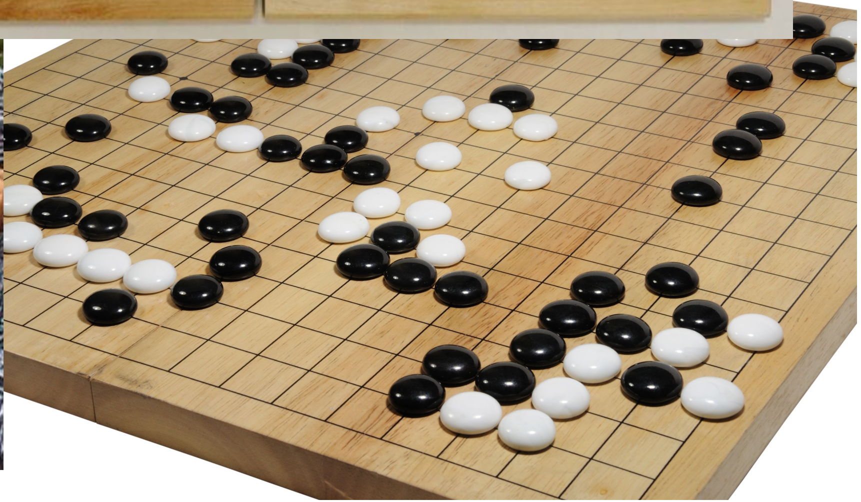
Alpha-Beta Pruning

willie
(some slides adapted form Sara Owsley Sood)

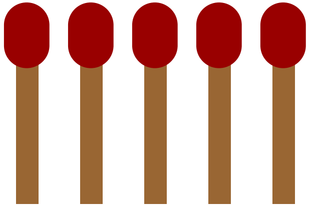
Opponent doesn’t want you to win







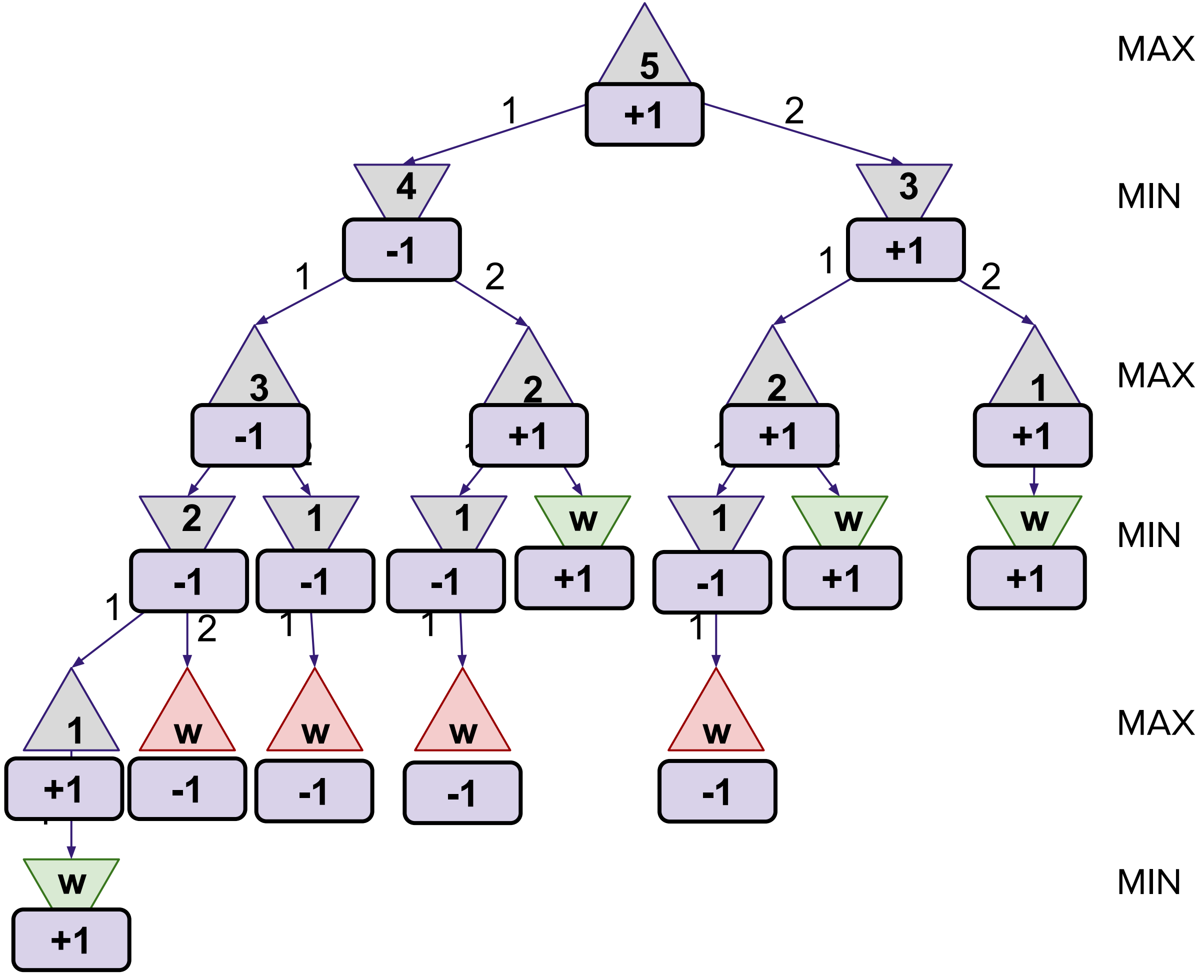
Baby Nim



Take 1 or 2 at each turn
Goal: take the last match

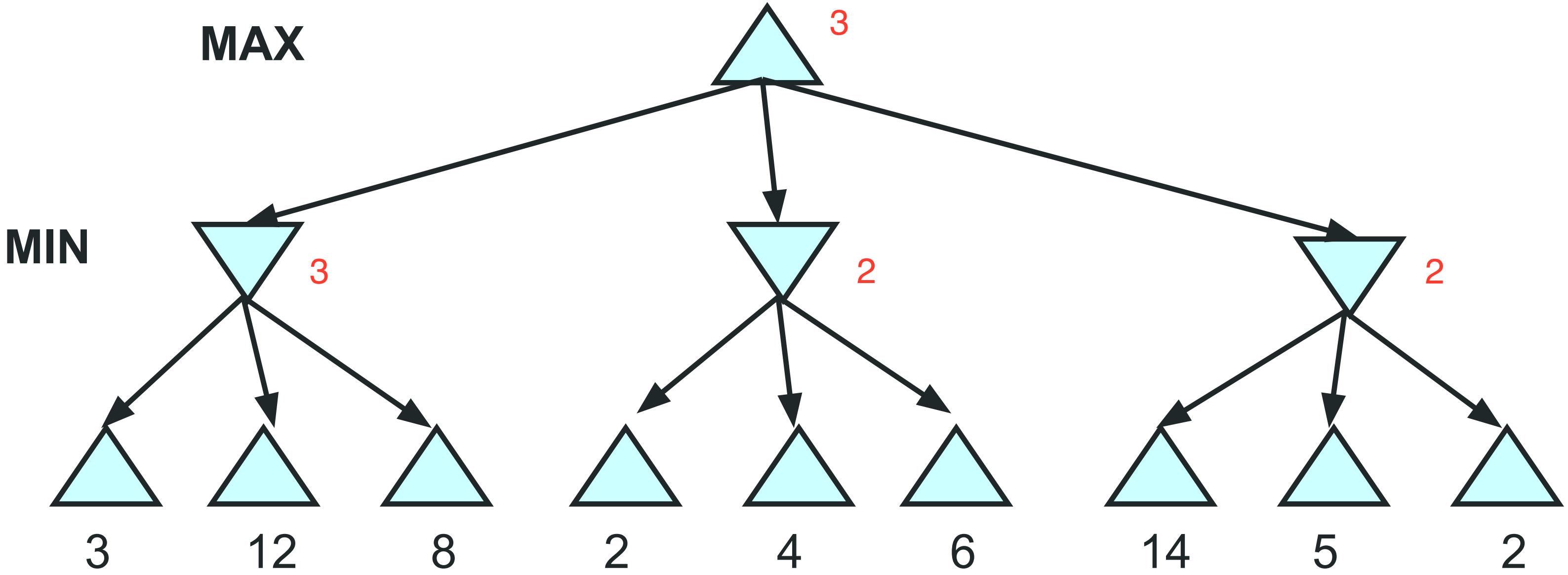
MAX Wins w = 1

MIN Wins w = -1

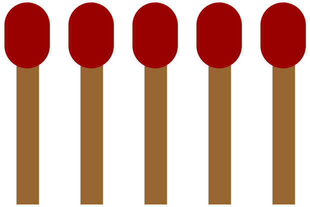


MINIMAX example 2

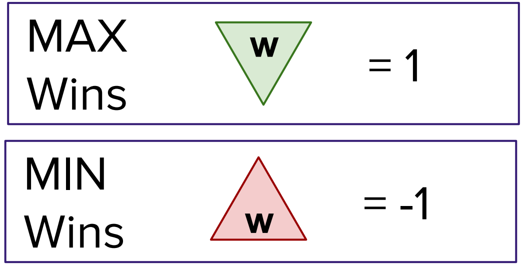
Not just -1, 0, +1

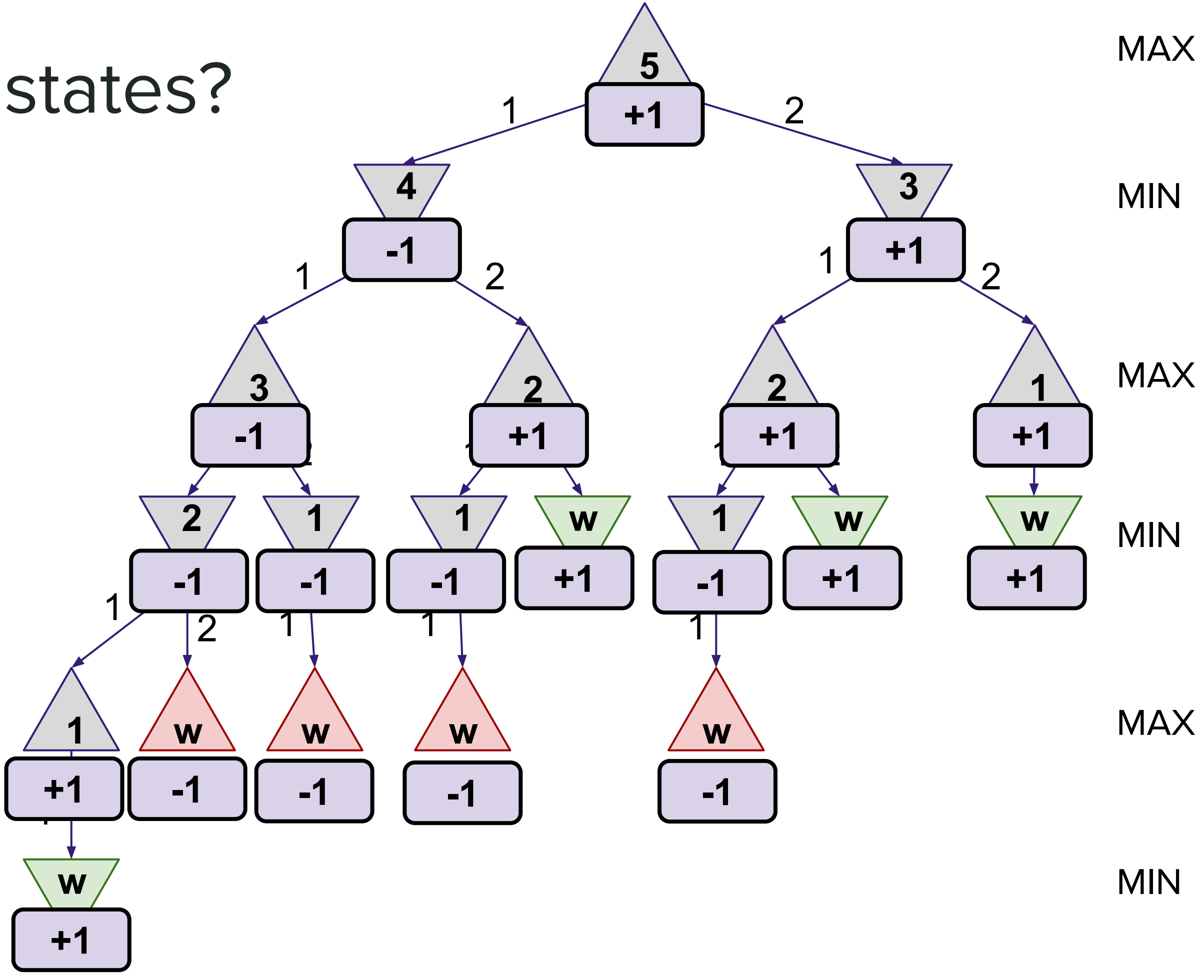


How many game states?



Take 1 or 2 at each turn
Goal: take the last match





Properties of minimax

For chess, b ≈ 35, d ≈100 (100 ply) for "reasonable" games

- exact solution completely infeasible

Is minimax reasonable for

Mancala?
B?
D?

Tic Tac Toe?
B?
D?

Connect Four?
B?
D?

Resource limits

Suppose we have 100 secs, and can explore 10^4 nodes/sec

➔ can explore 10^6 nodes per move

Standard approach (Shannon, 1950):

- **evaluation function**

  estimated desirability of position

- **cutoff test:**

  e.g., depth limit

Cutting off search

Change:

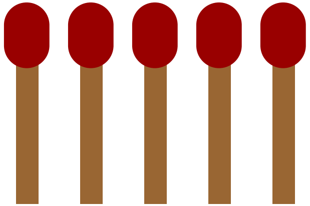
- if TERMINAL-TEST(state) then return UTILITY(state)

into

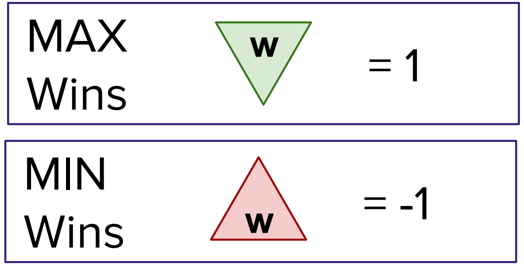
- if CUTOFF-TEST(state,depth) then return EVAL(state)

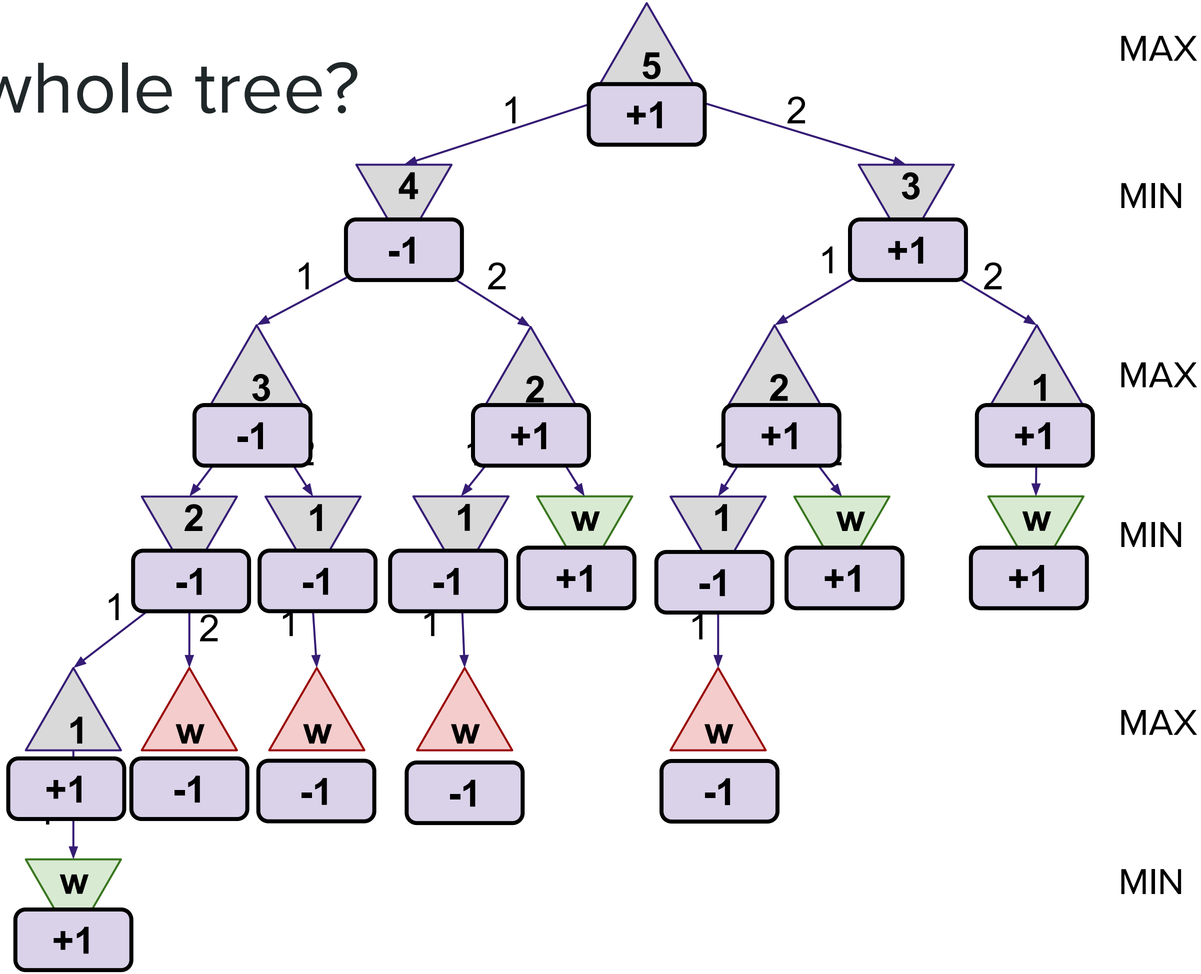
- Introduces a fixed-depth limit
  - Is selected so that the amount of time will not exceed what the rules of the game allow
- When cuttoff occurs, the evaluation is performed

Do we need the whole tree?

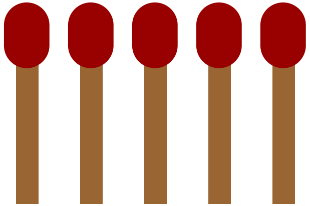


Take 1 or 2 at each turn
Goal: take the last match

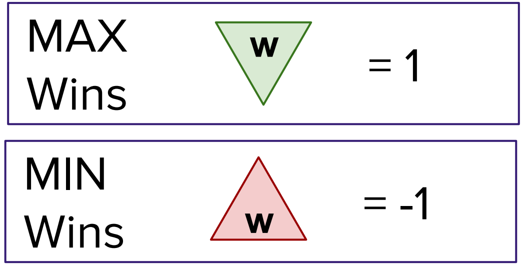


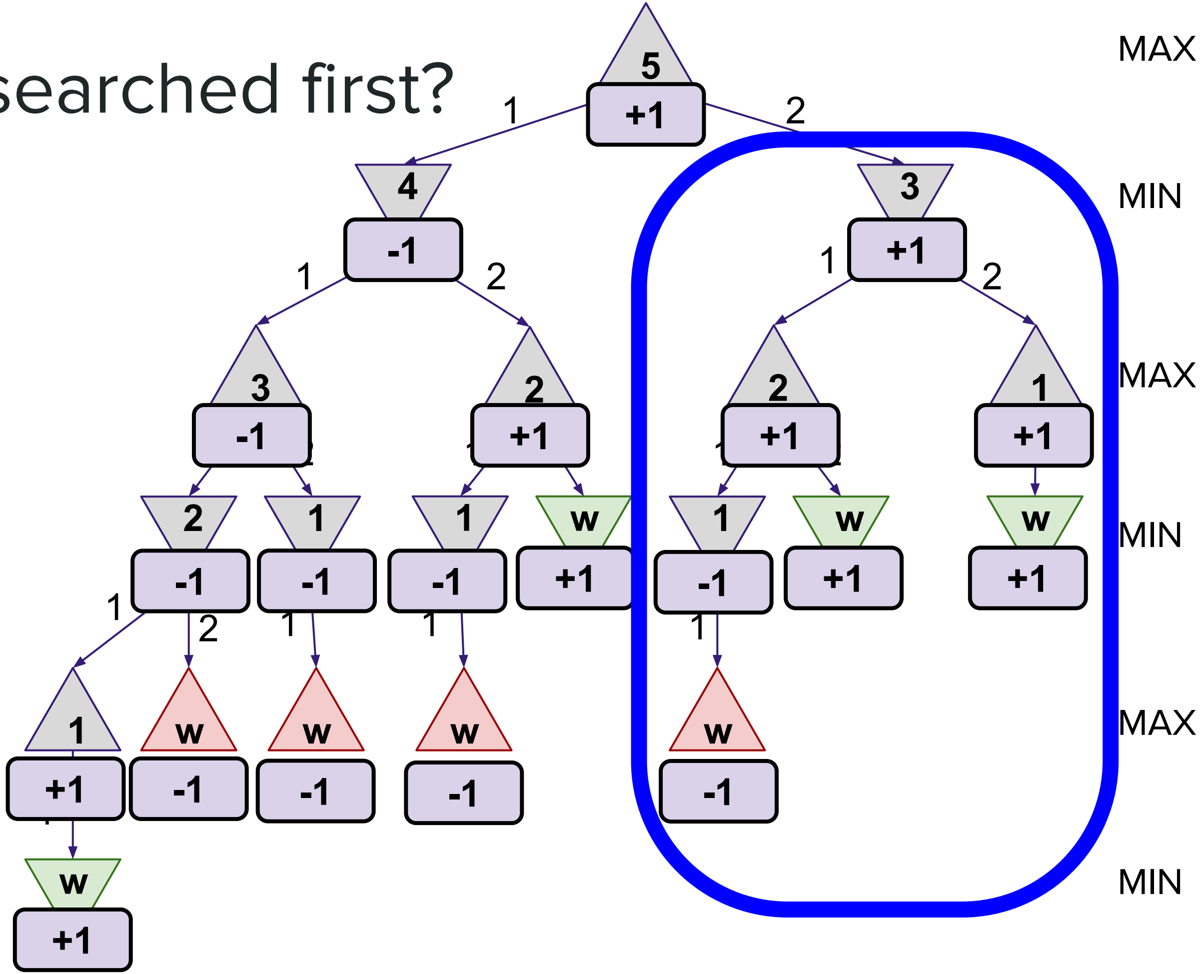


What if this path searched first?



Take 1 or 2 at each turn
Goal: take the last match





Alpha-Beta Pruning

Pruning: eliminate parts of the tree from consideration

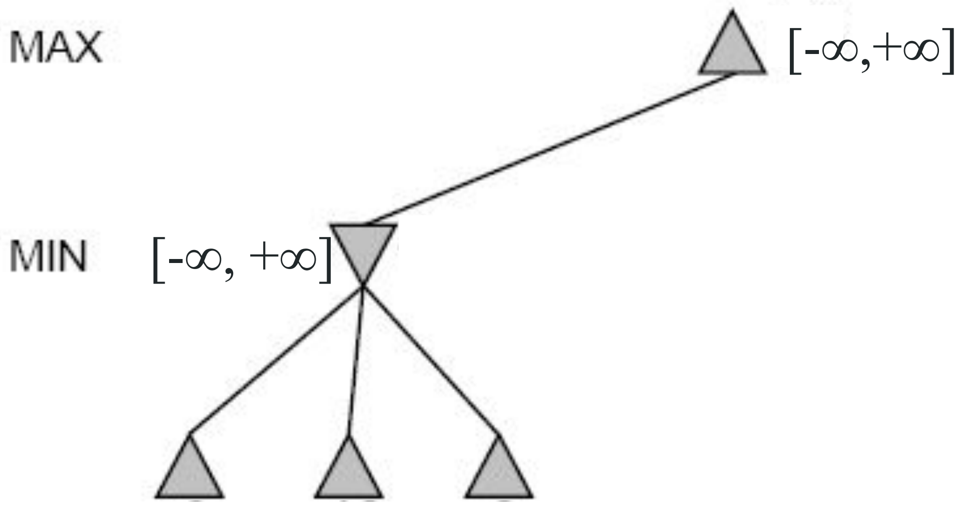
Alpha-Beta pruning: prunes away branches that can't possibly influence the final decision

Consider a node n

- If a player has a better choice m (at a parent or further up), then n will never be reached
- So, once we know enough about n by looking at some successors, then we can prune it.

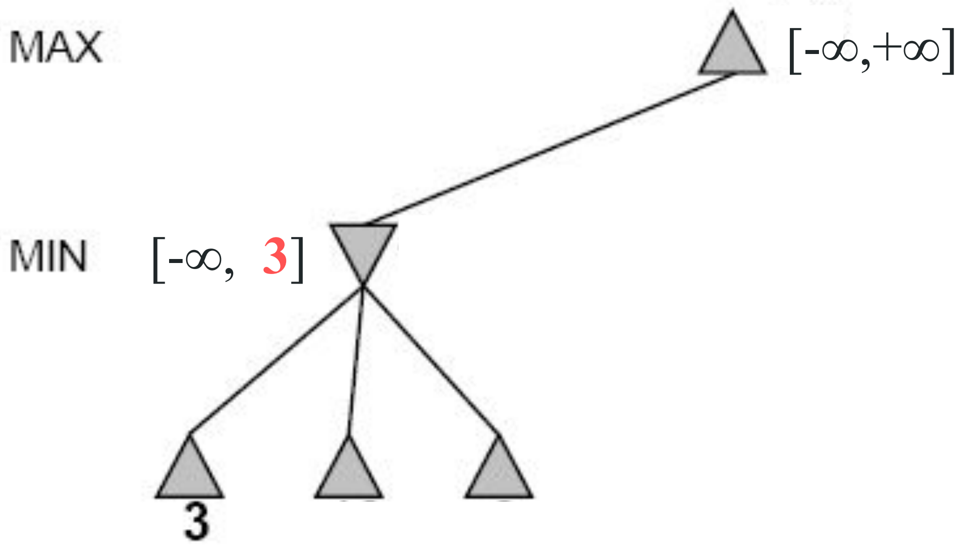
Alpha-Beta Example

Do DF-search until first leaf



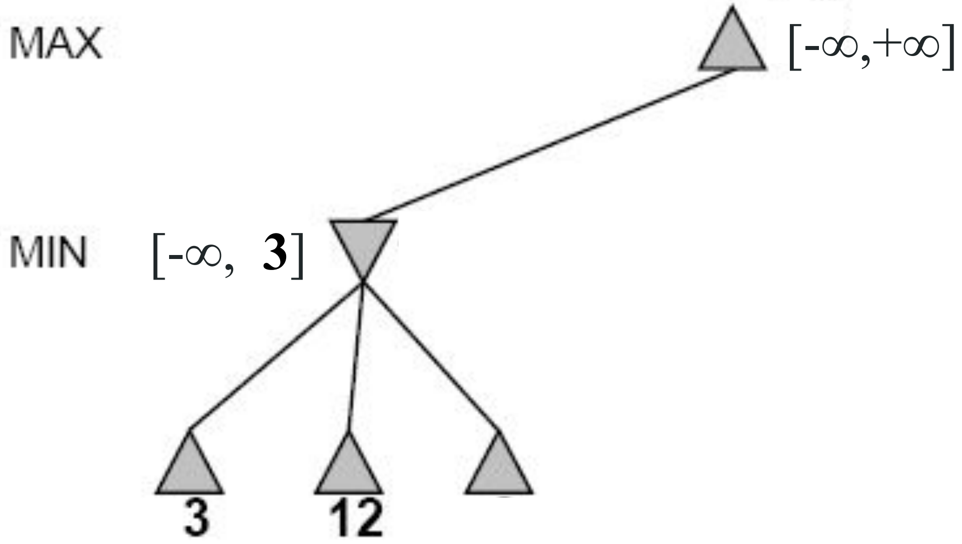
Alpha-Beta Example

At worst, MIN will choose 3



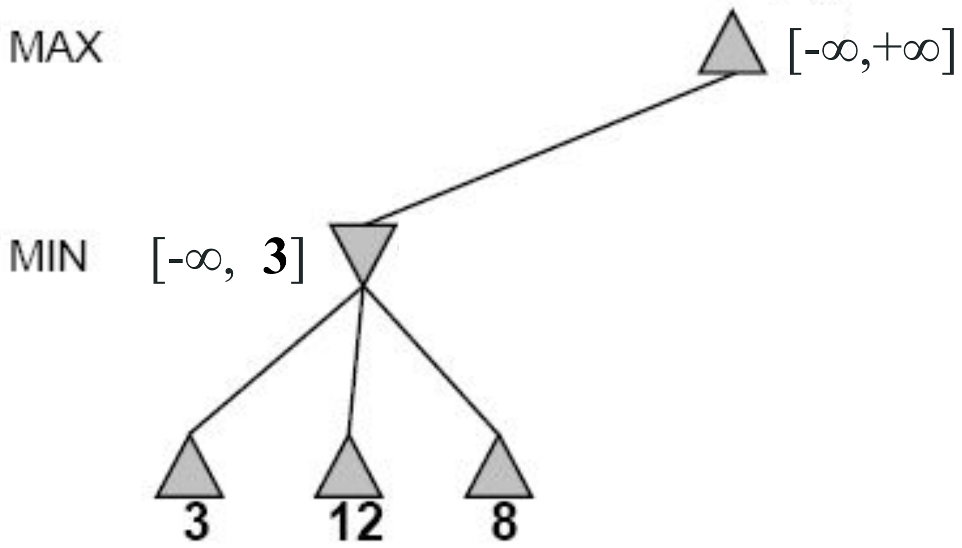
Alpha-Beta Example

MIN ignores 12



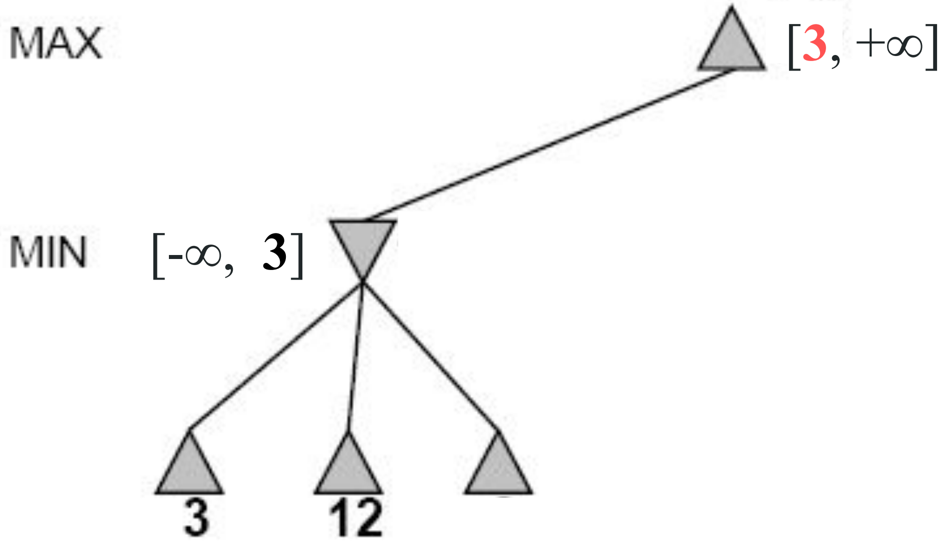
Alpha-Beta Example

MIN chooses 3



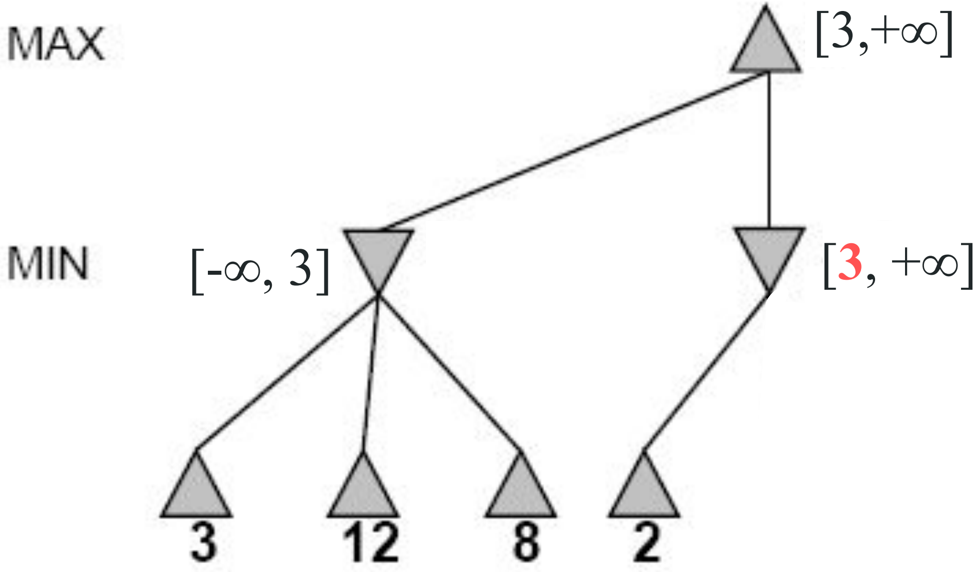
Alpha-Beta Example

MAX will choose 3 or more



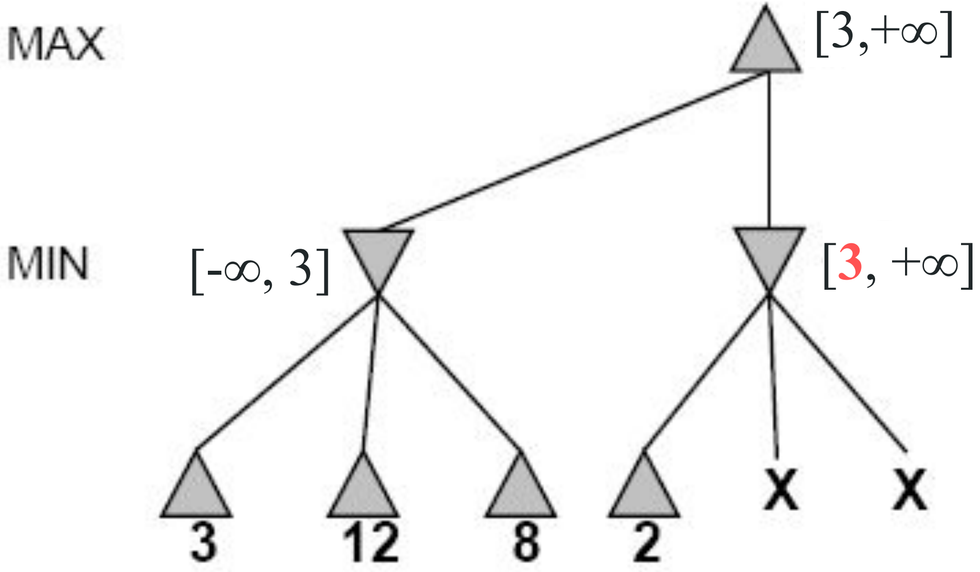
Alpha-Beta Example

2 is worse than the least option of 3



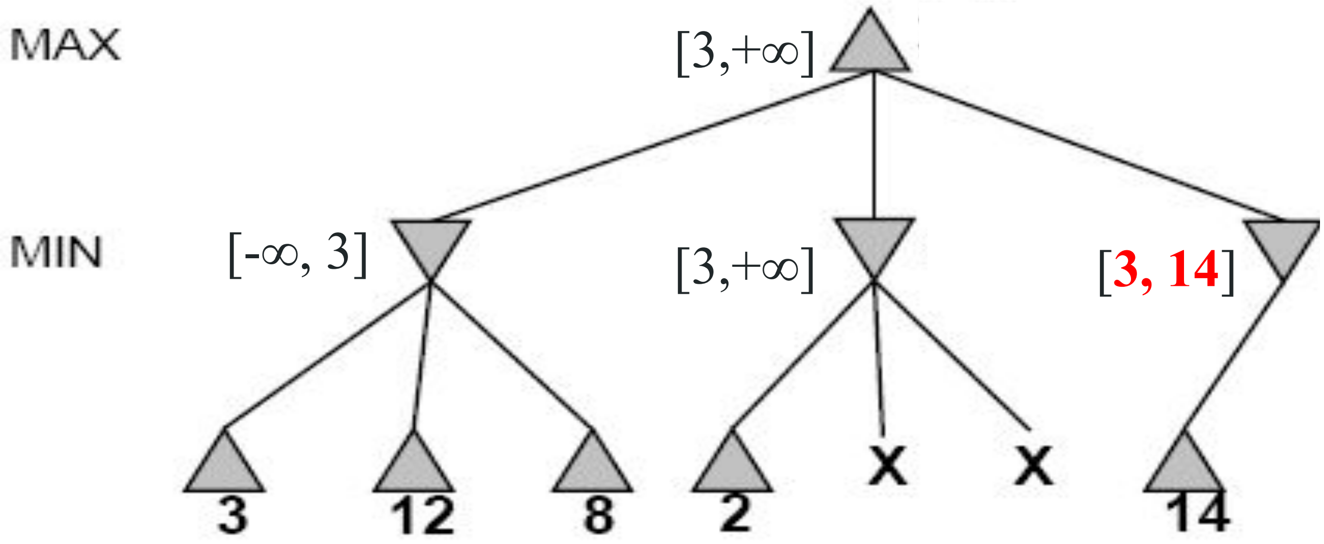
Alpha-Beta Example

Prune rest of branch



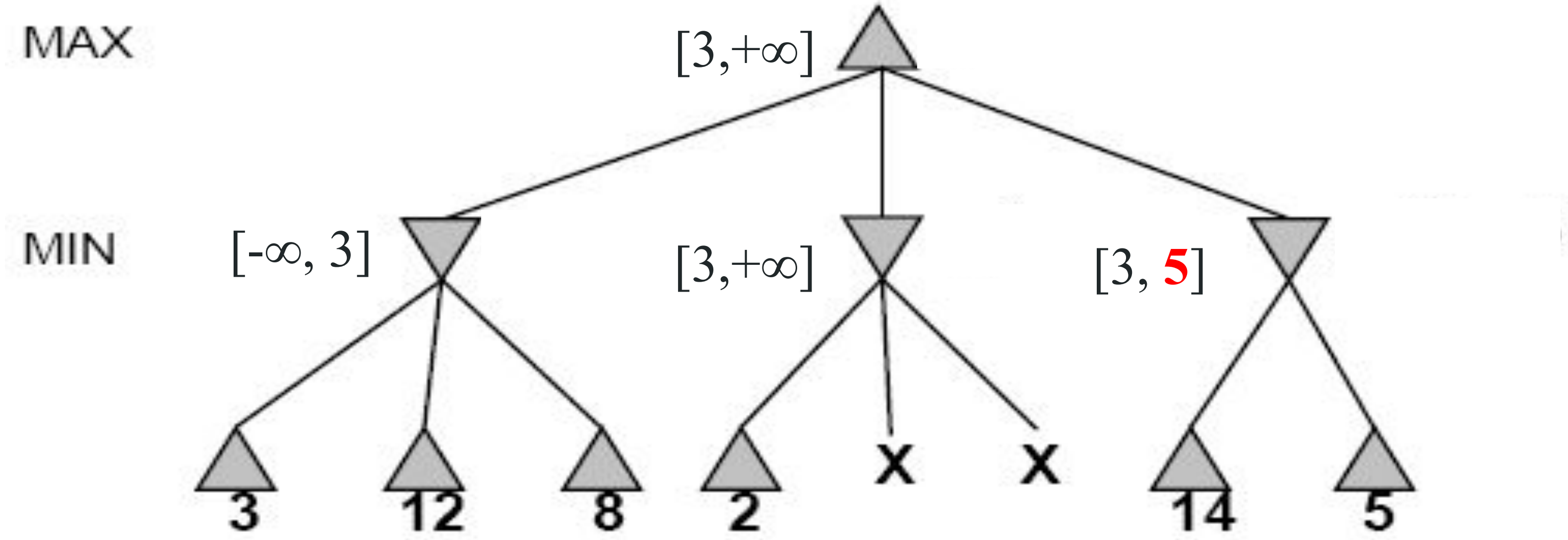
Alpha-Beta Example

14 is a better option for MAX



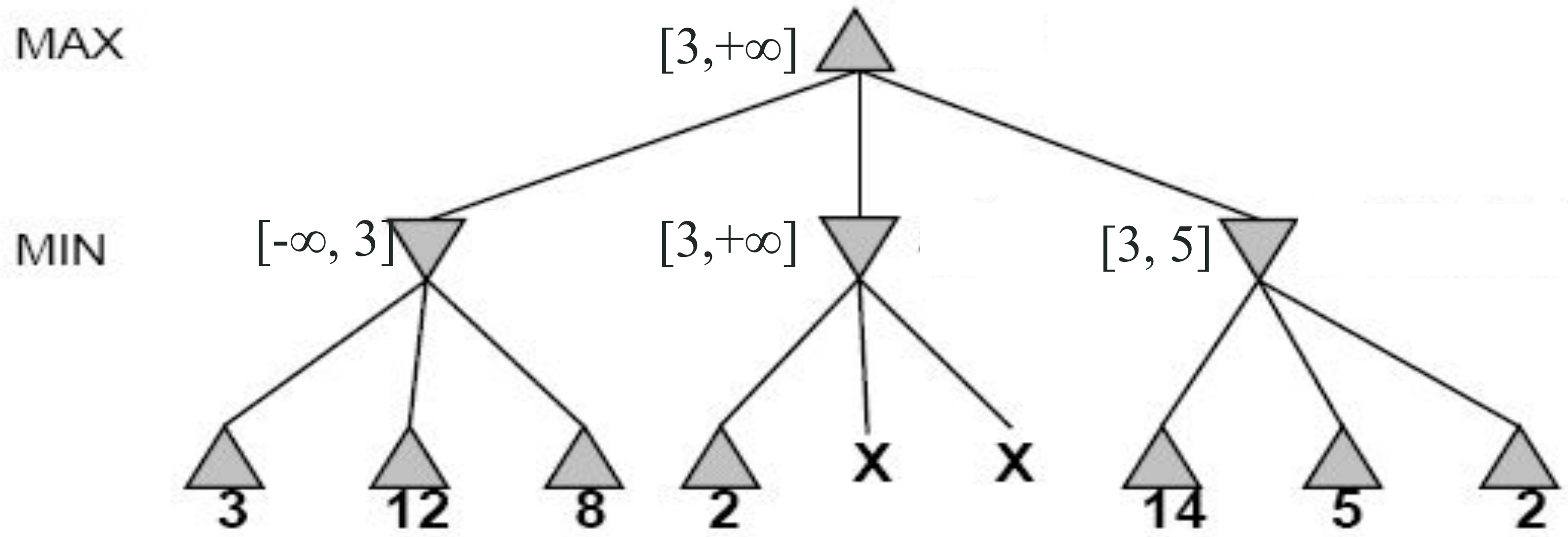
Alpha-Beta Example

5 is an even better option



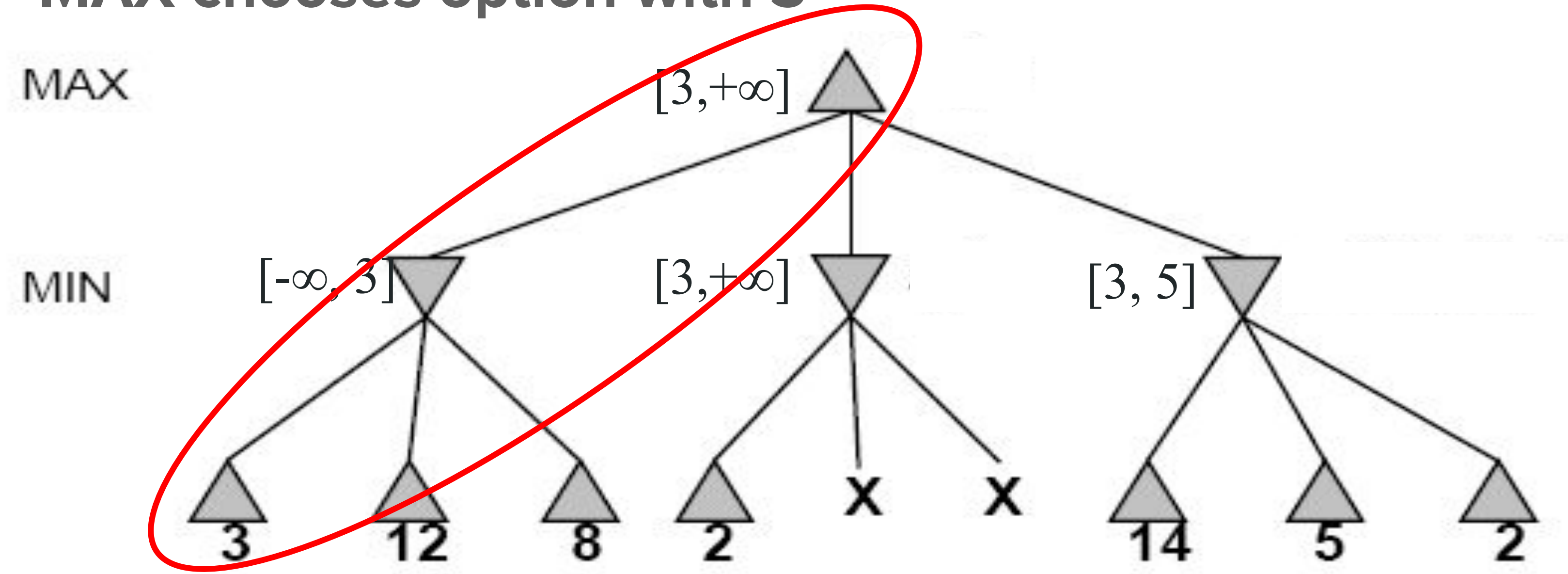
Alpha-Beta Example

2 is less than best option



Alpha-Beta Example

MAX chooses option with 3



```
function ALPHA-BETA-SEARCH(state) returns an action
  inputs: state, current state in game

  v ← MAX-VALUE(state, −∞, +∞)
  return the action in SUCCESSORS(state) with value v
```

```
function MAX-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for a, s in SUCCESSORS(state) do
    v ← MAX(v, MIN-VALUE(s, α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v
```

```
function MIN-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s, α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

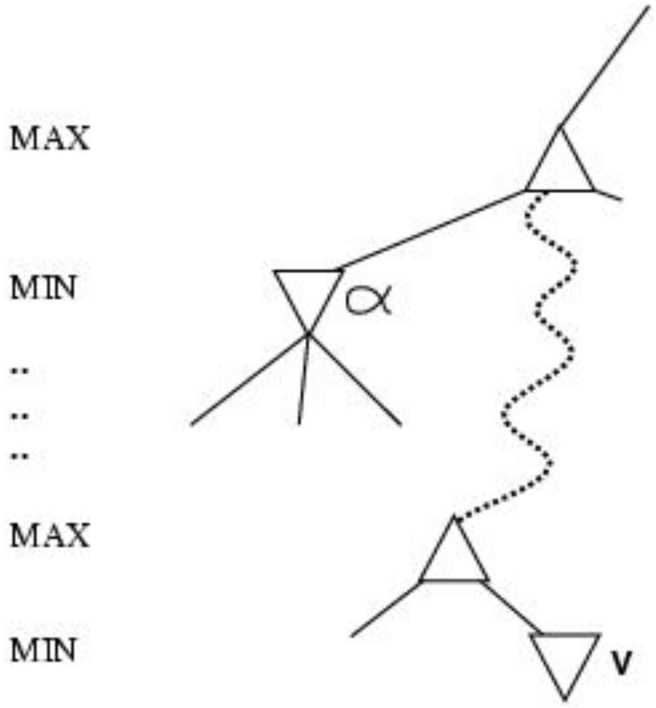
Why is it called α-β?

α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for *max*

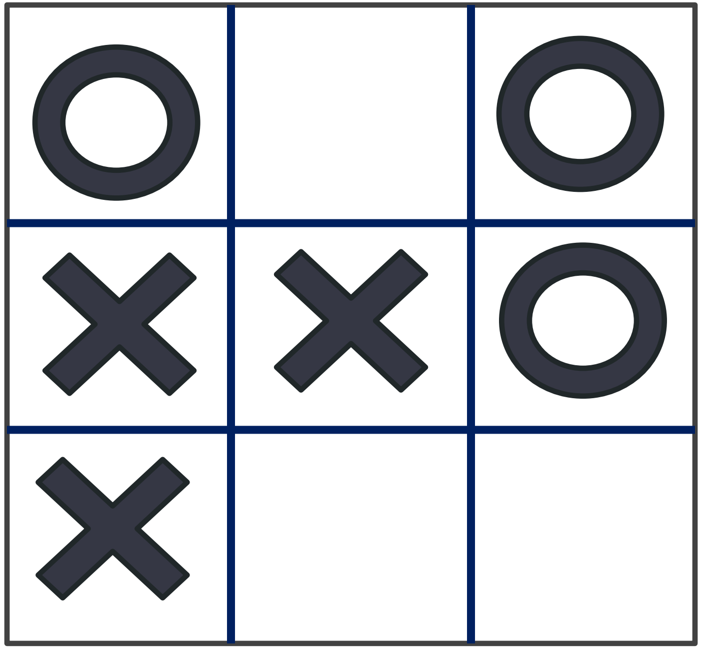
If *v* is worse than α, *max* will avoid it

➡ prune that branch

Define β similarly for *min*



Try it out

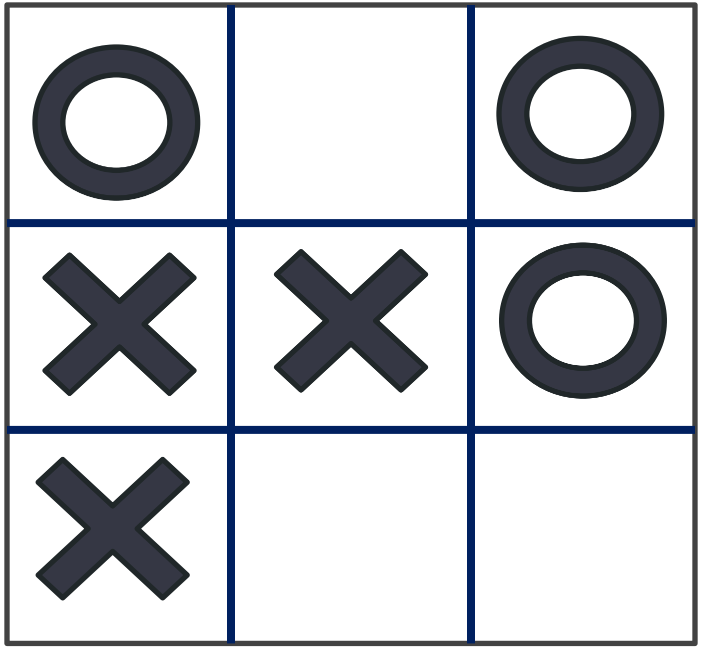


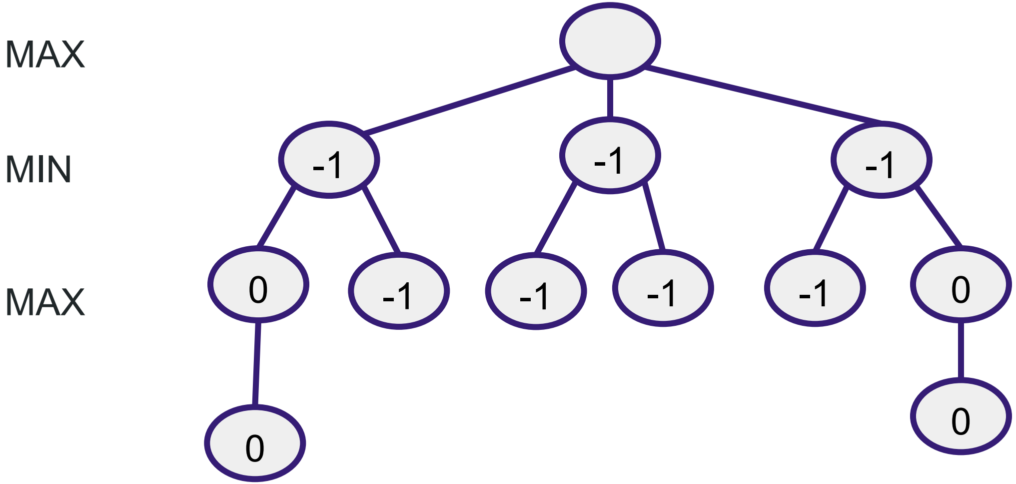
It is currently X’s turn

Use Alpha-Beta to determine what move X should make?

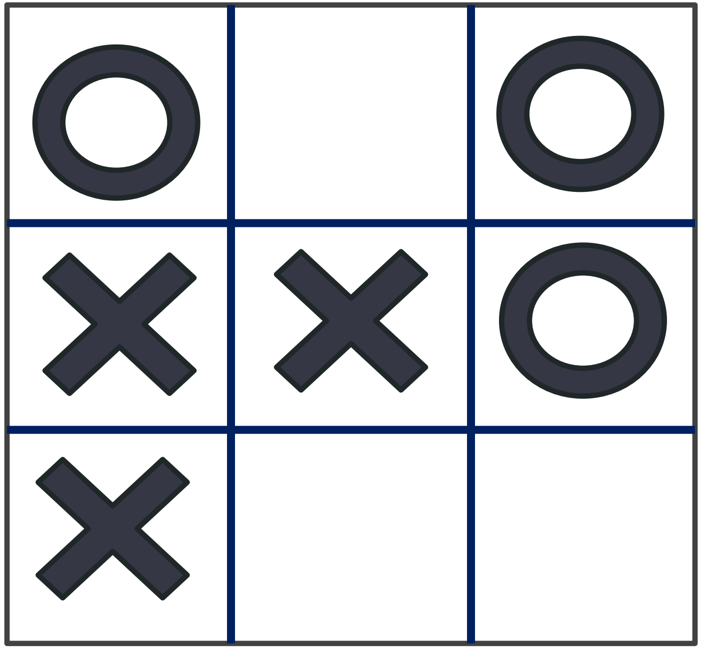
Who will win the game?

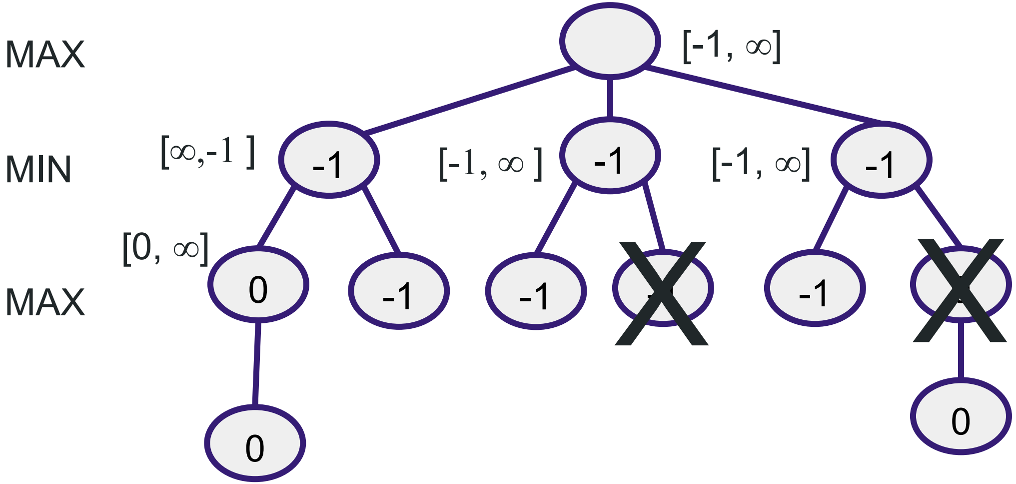
MINIMAX





Alpha-Beta





Properties of α-β

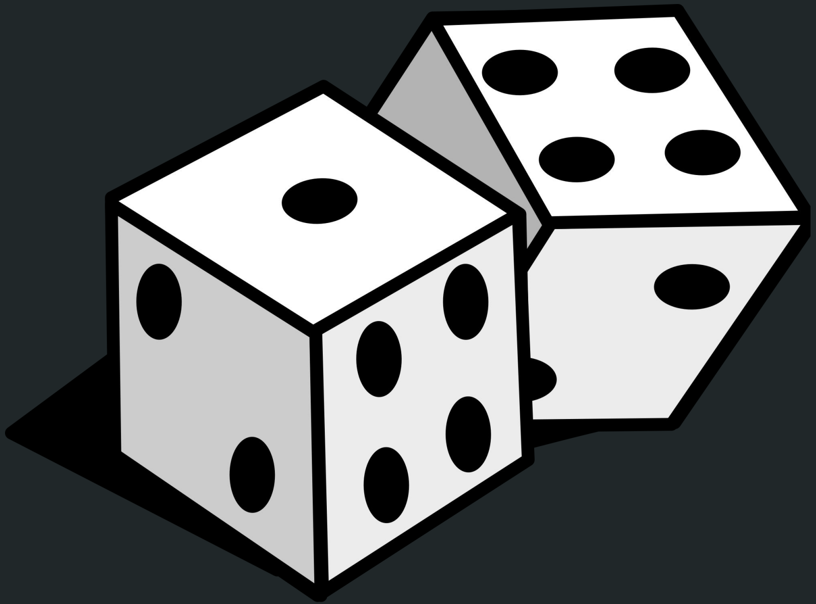
Pruning does not affect final result

However, effectiveness of pruning affected by...?

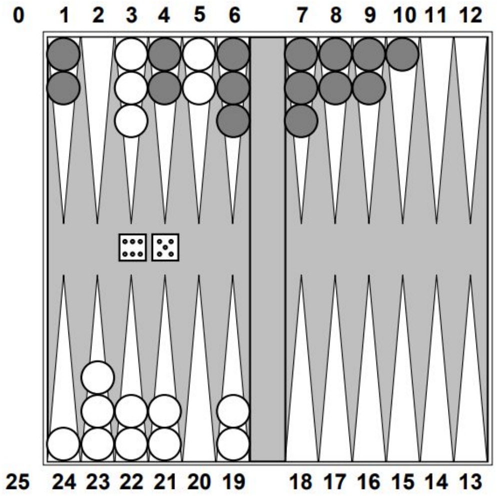
What impact can it have on running time?

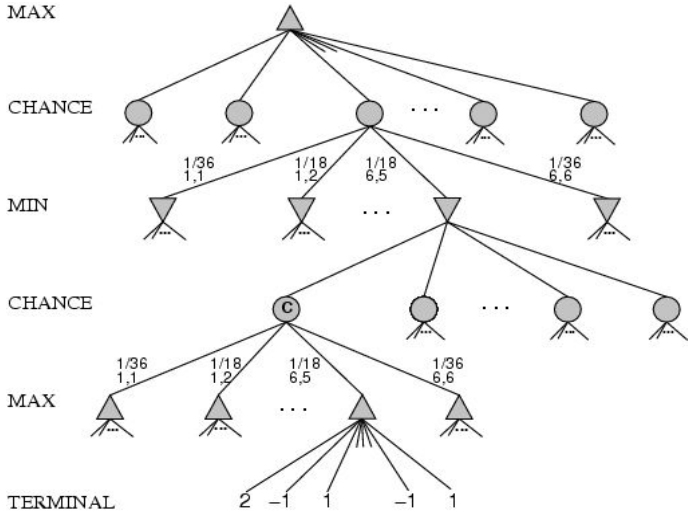
Good enough for Mancala?

What if things were more random?



Games that include chance

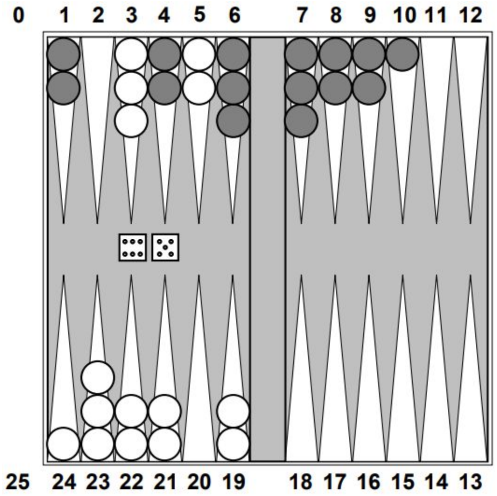


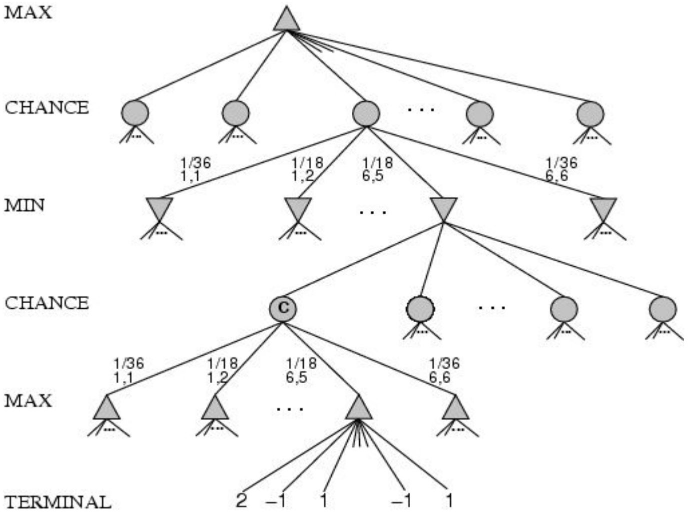


White’s turn, After rolling a 5 and a 6

Possible moves (5-10,5-11), (5-11,19-24),(5-10,10-16) and (5-11,11-16)

Games that include chance

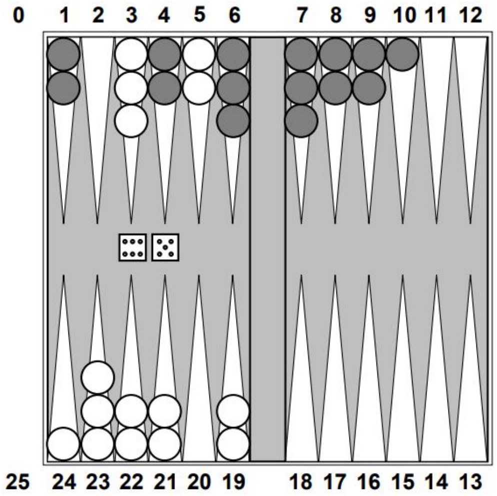


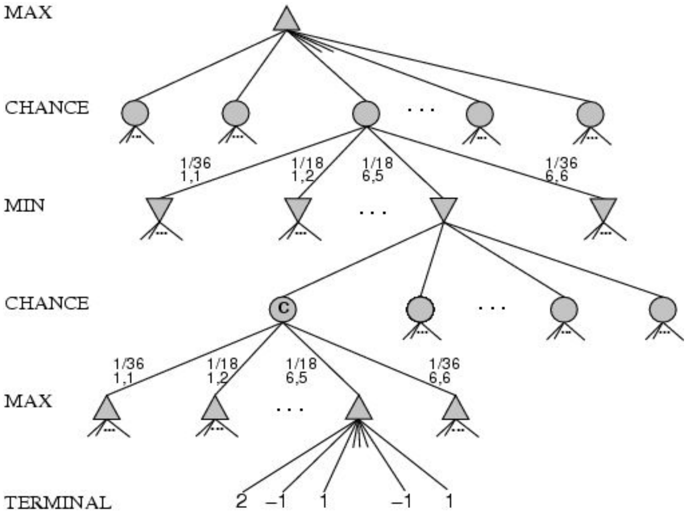


Possible moves (5-10,5-11), (5-11,19-24),(5-10,10-16) and (5-11,11-16)

[1,1] - [6,6] chance 1/36, all other chance 1/18

Games that include chance





[1,1] - [6,6] chance 1/36, all other chance 1/18

Can not calculate definite minimax value, only expected value

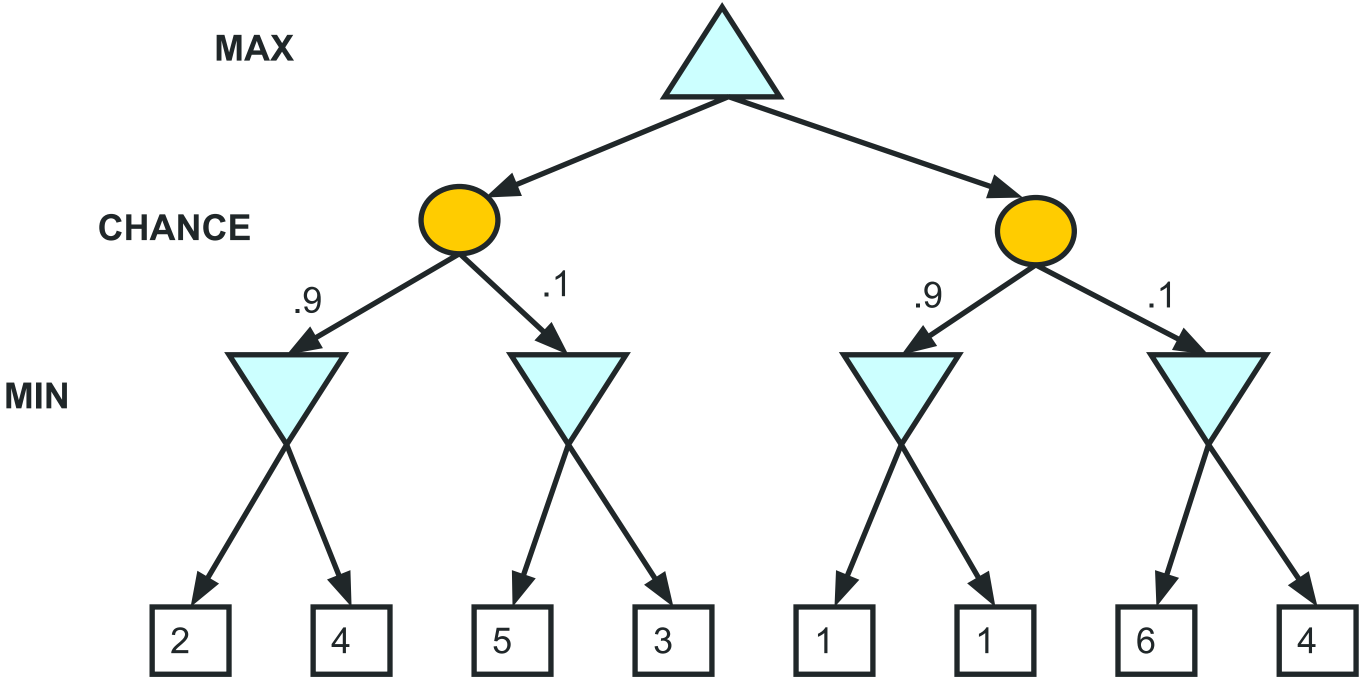
Expecti minimax value

EXPECTI-MINIMAX-VALUE(n)=

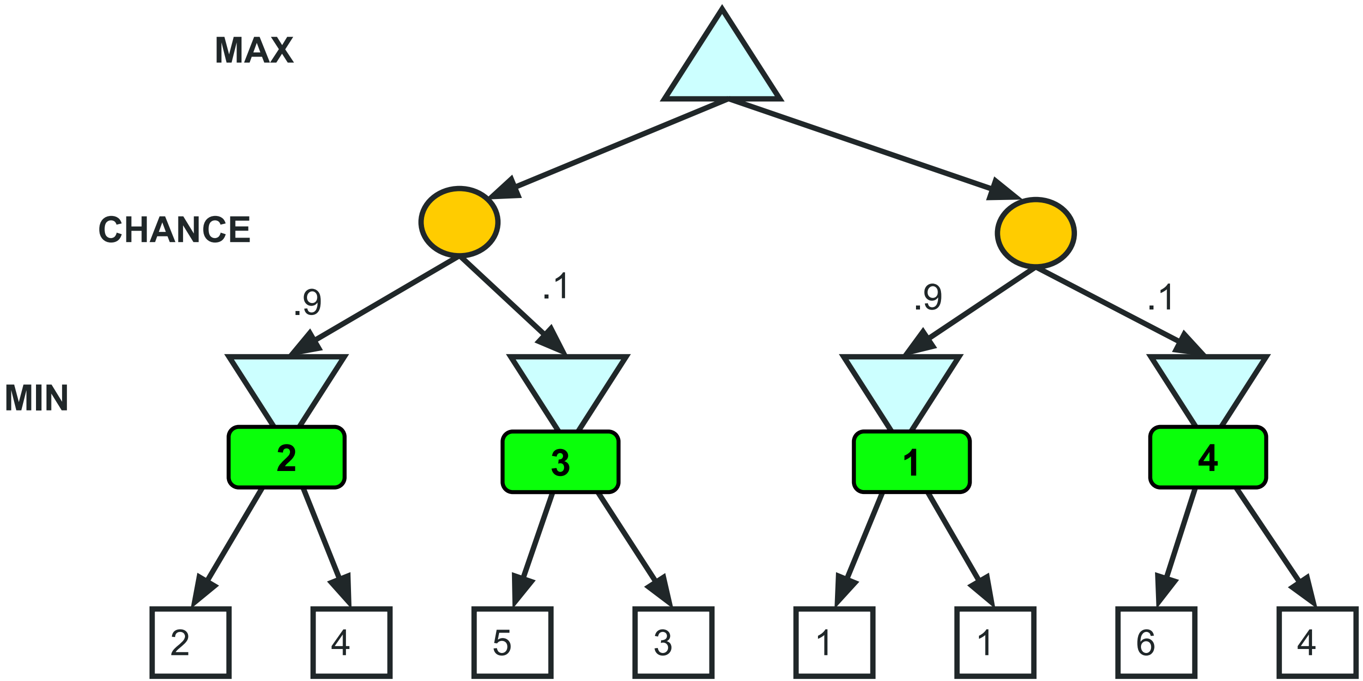
UTILITY(n)	If n is a terminal
$\max_{s \in successors(n)}$ MINIMAX-VALUE(s)	If n is a max node
$\min_{s \in successors(n)}$ MINIMAX-VALUE(s)	If n is a min node
$\sum_{s \in successors(n)} P(s)$. EXPECTIMINIMAX(s)	If n is a chance node

These equations can be backed-up recursively all the way to the root of the game tree.

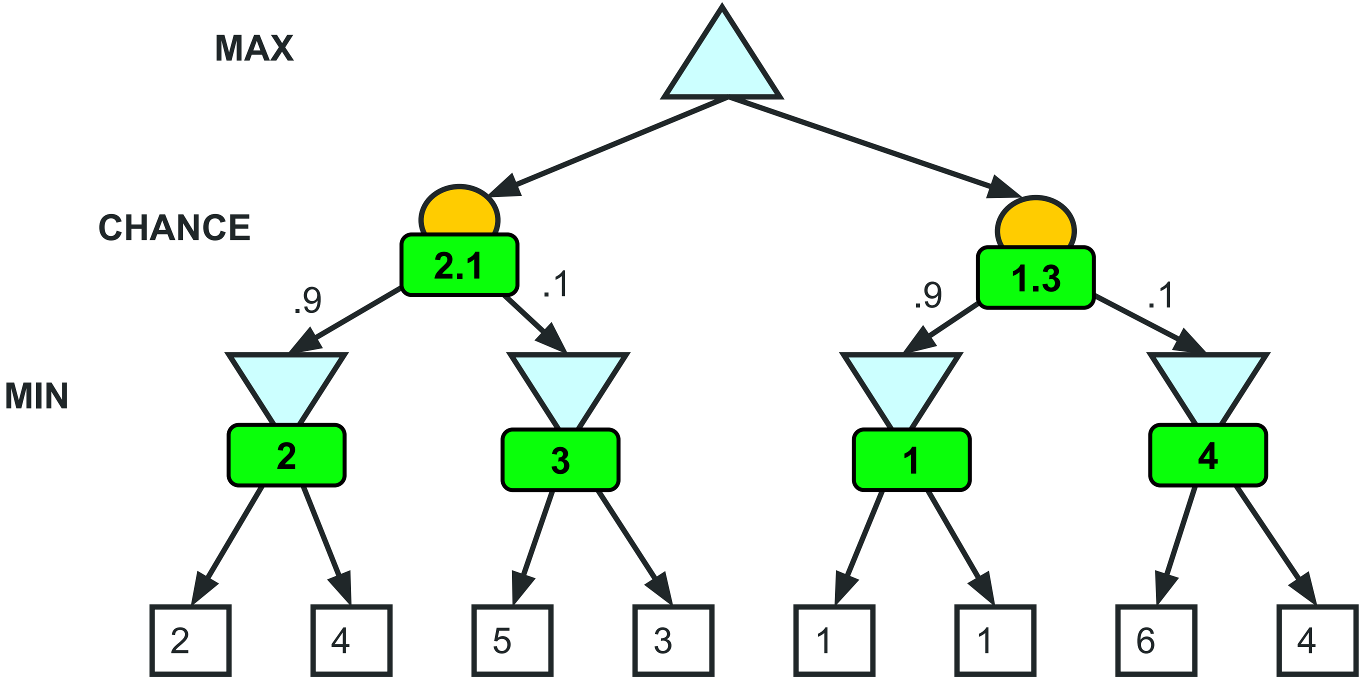
EXPECTIMINIMAX example



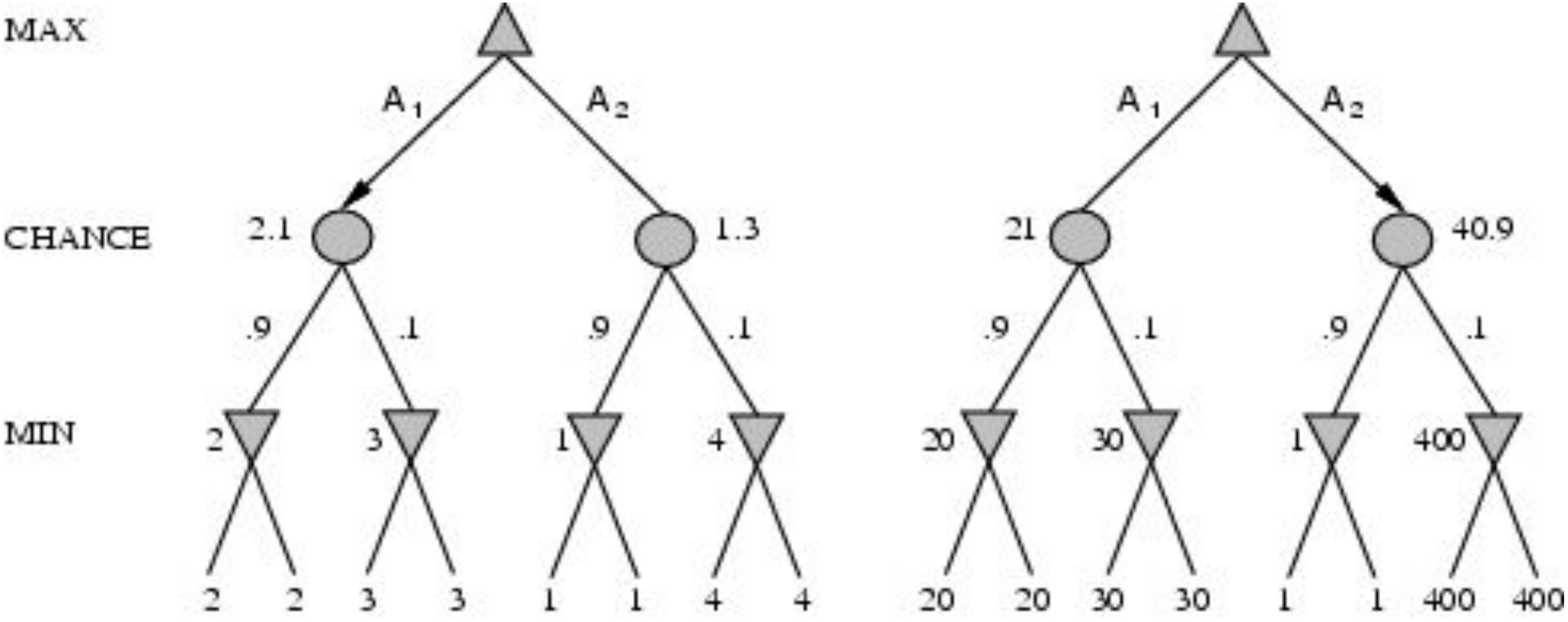
EXPECTIMINIMAX example



EXPECTIMINIMAX example

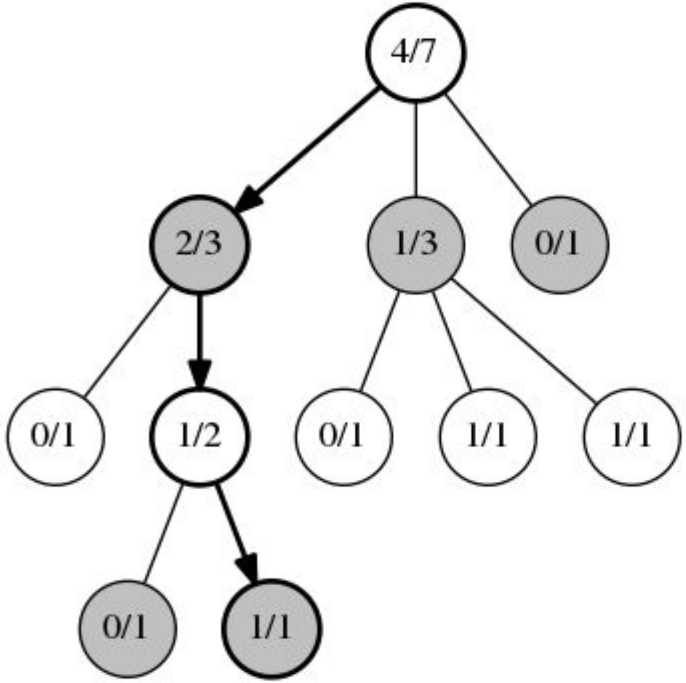


Position evaluation with chance nodes



Monte Carlo Tree Search

1. Selection
2. Expansion
3. Simulation
4. Update



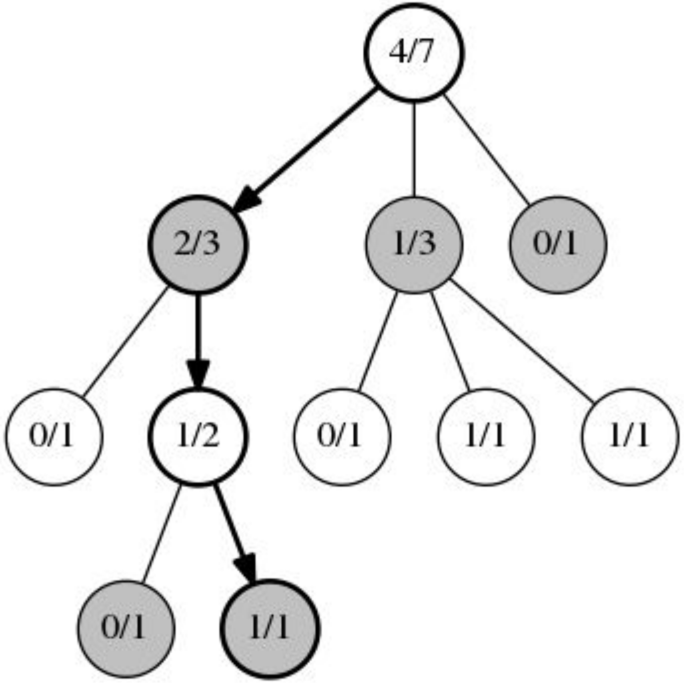
Selection

Game tree records statistics

Based on statistics, choose move

Continue until not all children have statistics

Node to be expanded

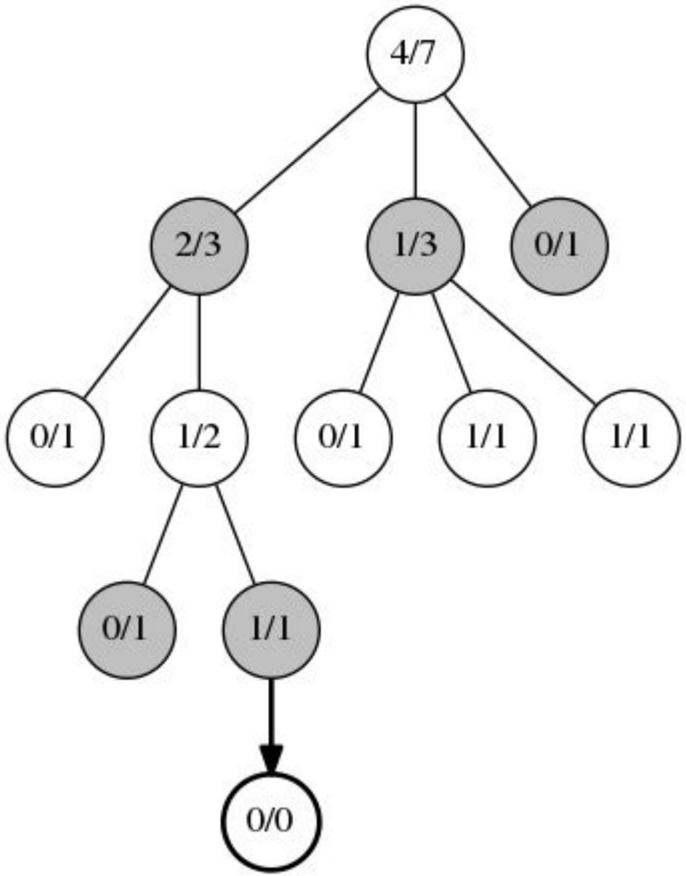


Expansion

Randomly choose a child

Create a new record (0/0)

From this child, simulate

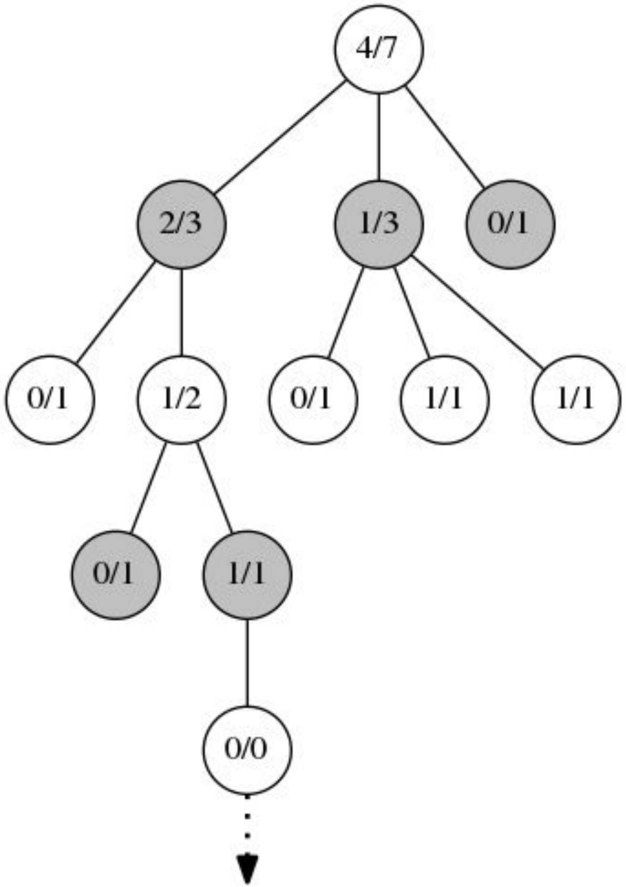


Simulation

Monte Carlo simulation

- Purely random
- Light playout
- Heavy playout

Play to end to determine game outcome

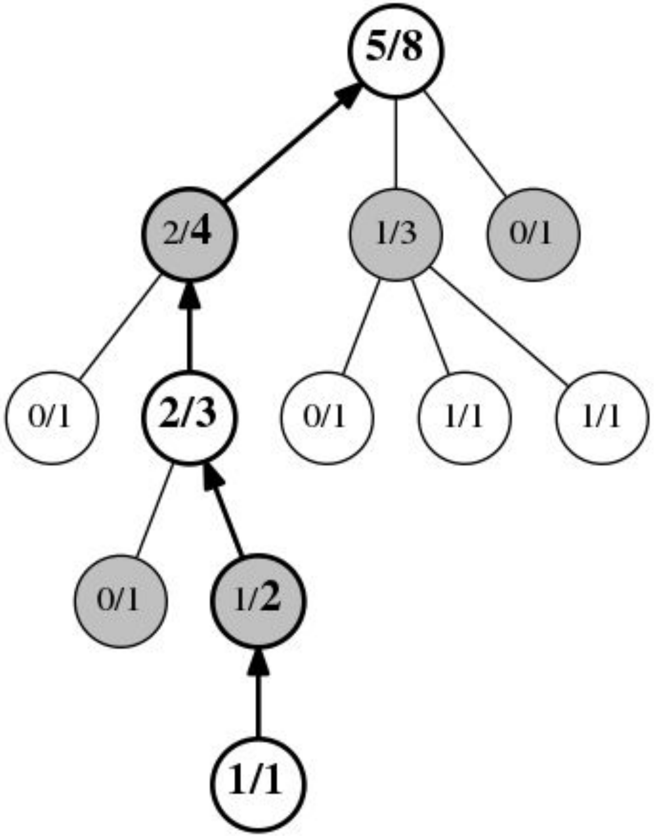


Update

Update all records in the path

Play count incremented

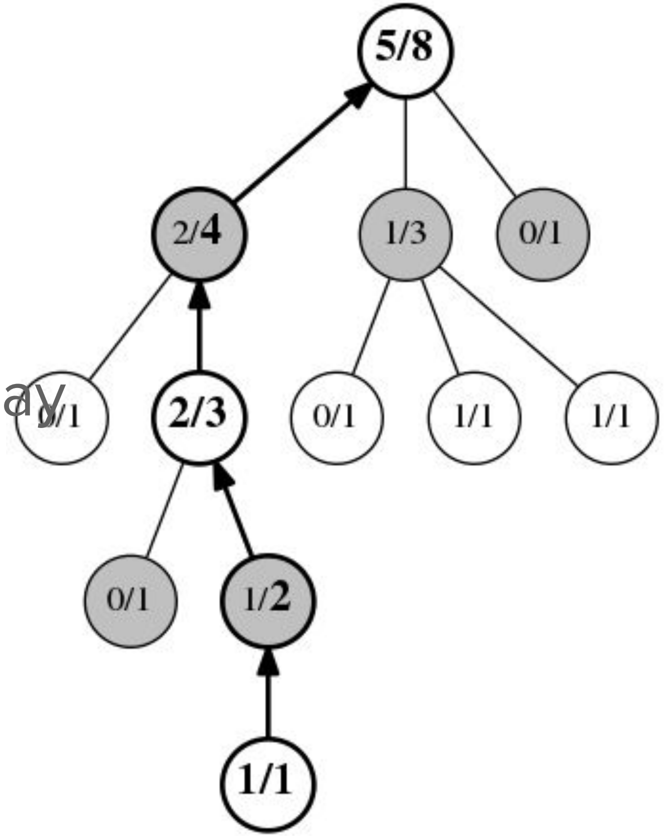
Each matching winner, win count incremented



Repeat

Continue for allotted time or some other end condition

Converges to actual optimal play



Which to select?

Just focus on the most promising path?

What if you happen to neglect a more promising path?

Upper Confidence Bound

$$\bar{x}_i \pm \sqrt{\frac{2 \ln n}{n_i}}$$