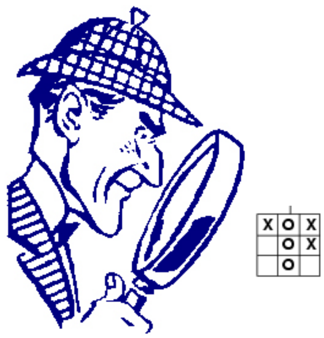
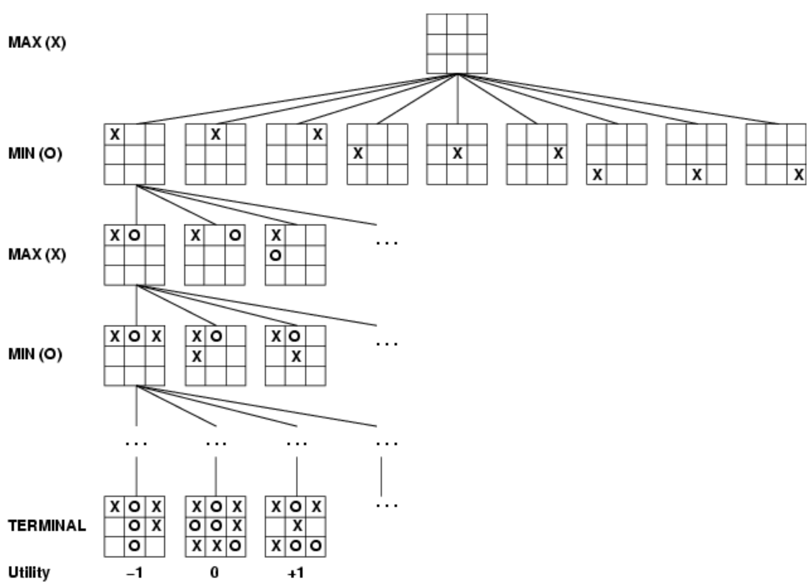
CSE 473

Chapter 6

Games and Game Tree Search



Game Tree for Tic-Tac-Toe



Optimal Strategy: Minimax Search

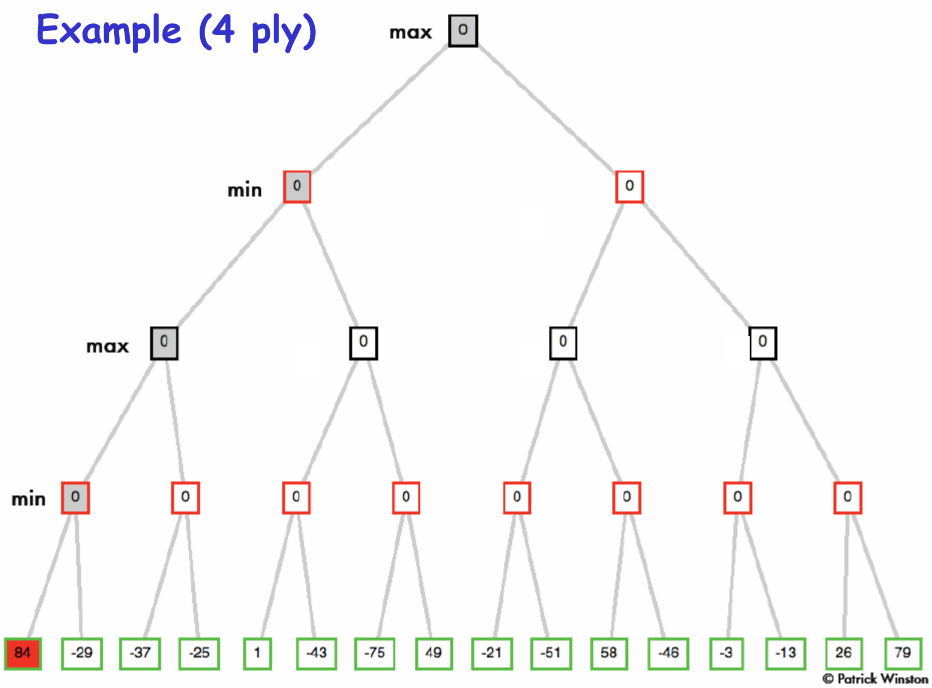
- Find the contingent *strategy* for MAX assuming an infallible MIN opponent
- Assumption: Both players play optimally!
- Given a game tree, the optimal strategy can be determined by using the *minimax* value of each node:

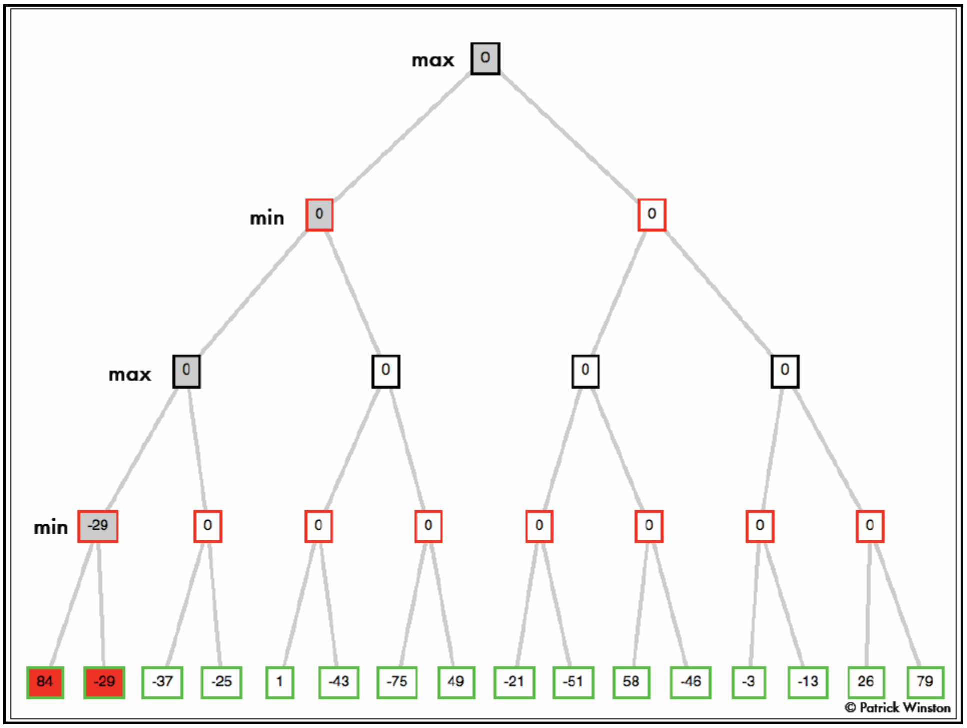
MINIMAX-VALUE(n)=

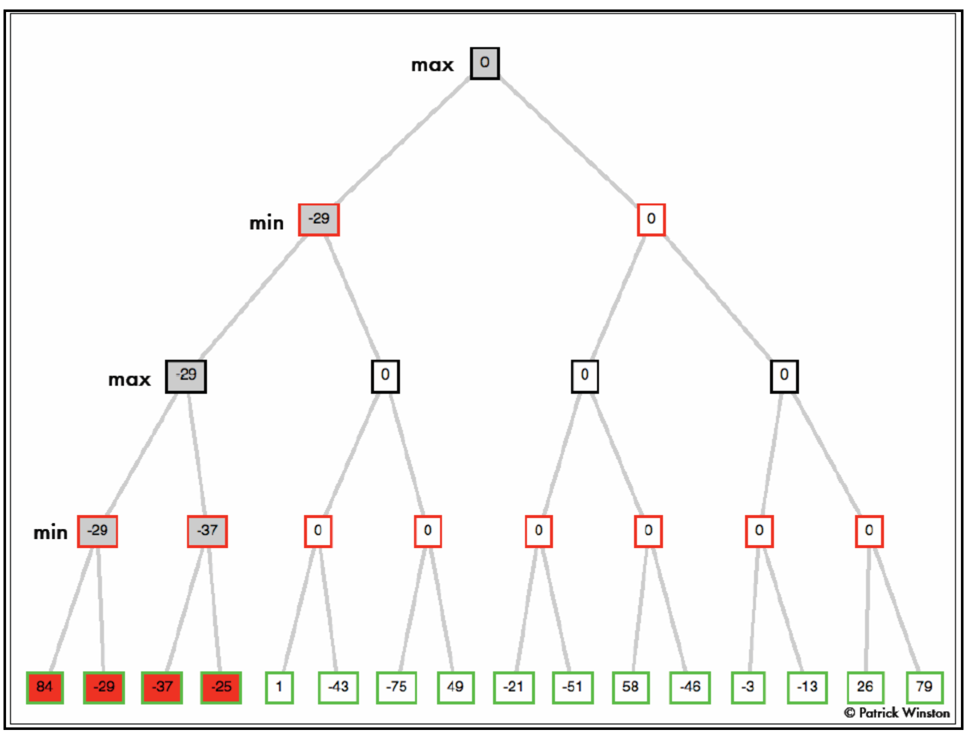
UTILITY(n)	If n is a terminal
$\max_{s \in succ(n)}$ MINIMAX-VALUE(s)	If n is a MAX node
$\min_{s \in succ(n)}$ MINIMAX-VALUE(s)	If n is a MIN node

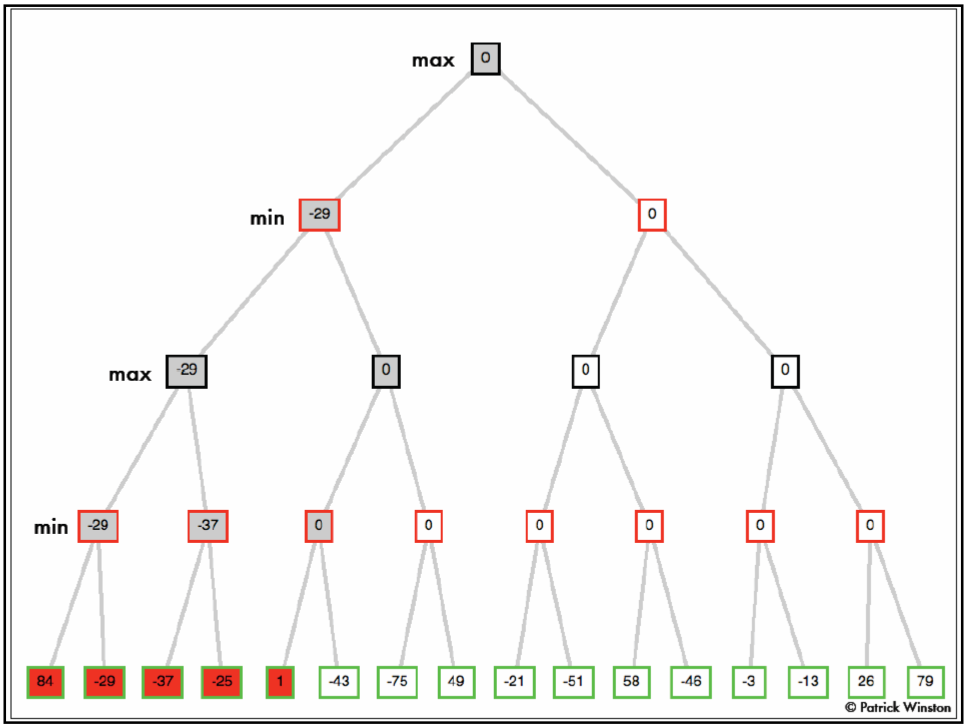
3

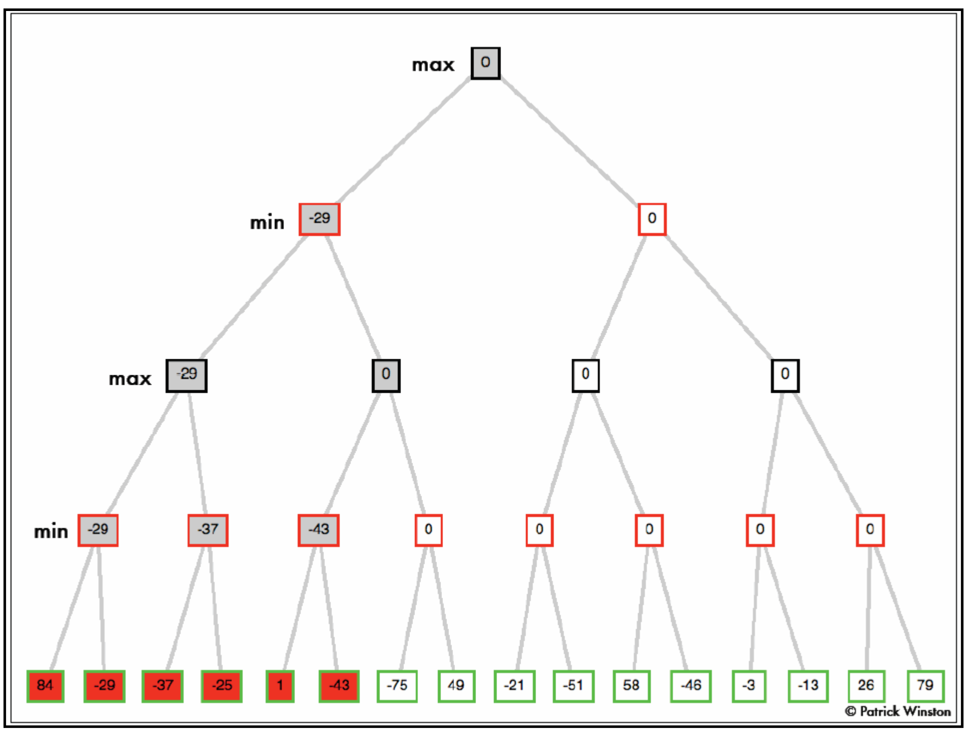
Example (4 ply)

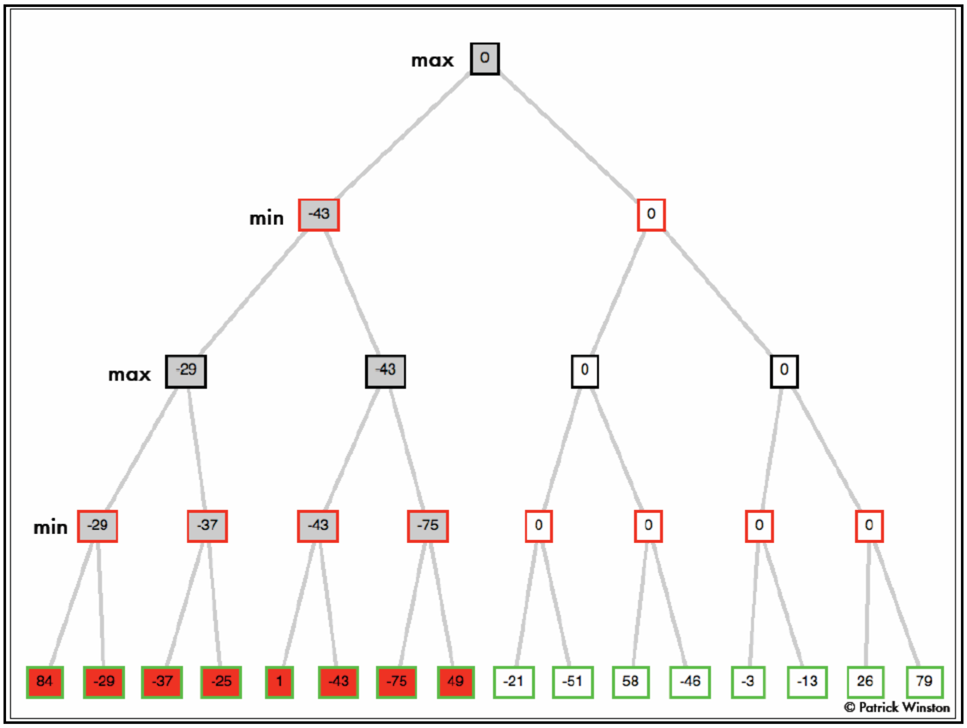


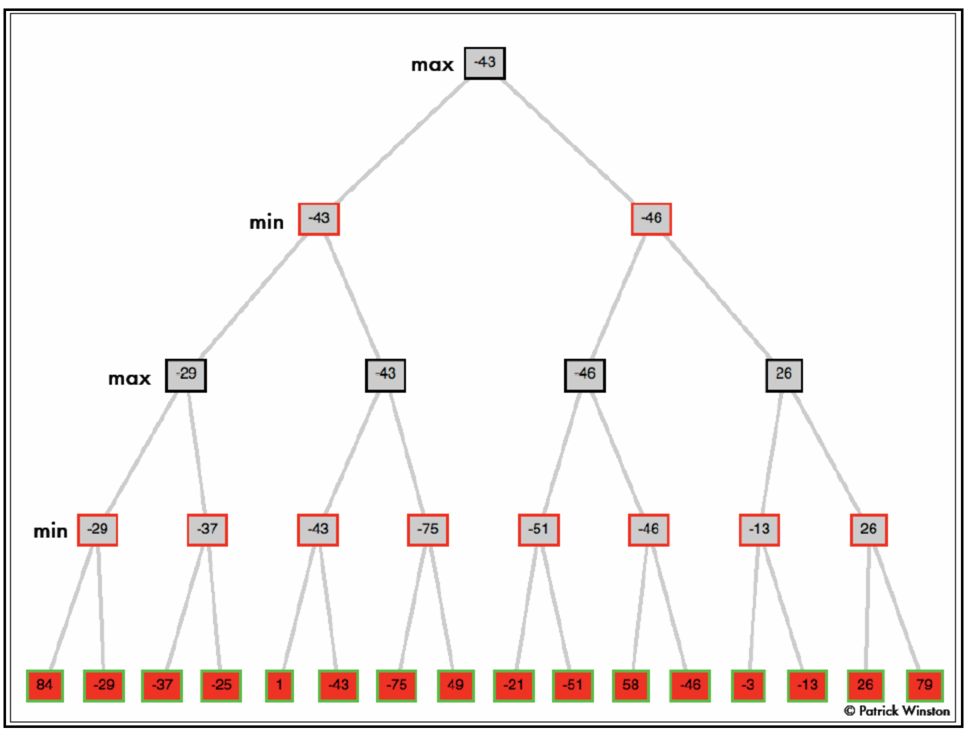

max
0
min
0
0
max
0
0
0
0
min
-29
0
0
0
0
0
0
0
84
-29
-37
-25
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79
© Patrick Winston

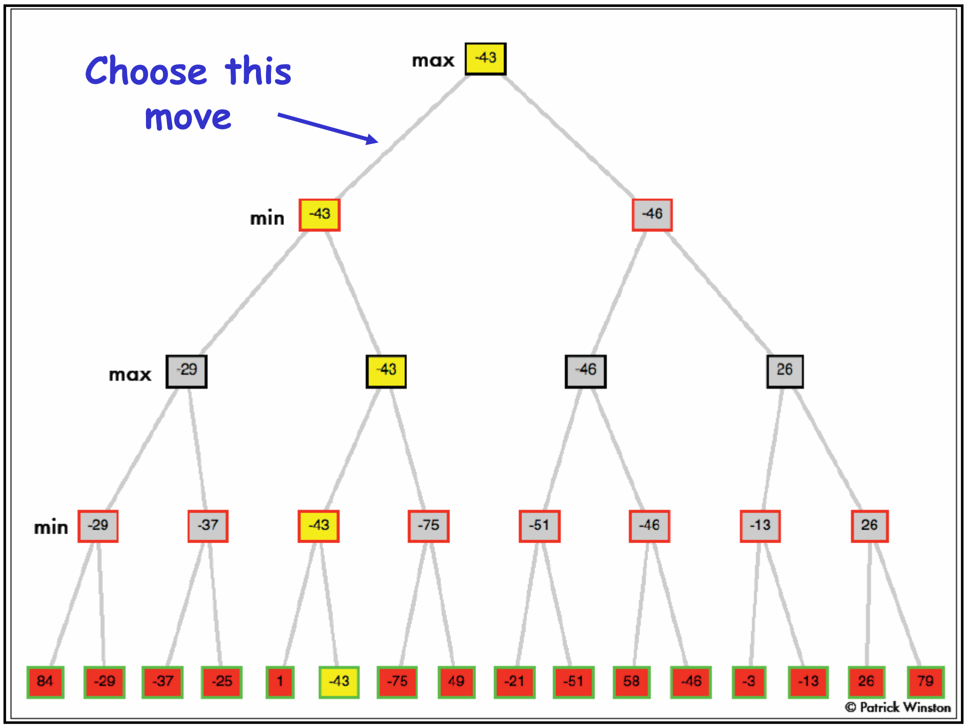

max
0
min
-29
0
max
-29
0
0
0
min
-29
-37
0
0
0
0
0
0
84
-29
-37
-25
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79
© Patrick Winston


max
0
min
-29
0
max
-29
0
0
0
min
-29
-37
0
0
0
0
0
0
84
-29
-37
-25
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79
© Patrick Winston


max
0
min
-29
0
max
-29
0
0
0
min
-29
-37
-43
0
0
0
0
0
84
-29
-37
-25
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79
© Patrick Winston


max
0
min
-43
0
max
-29
-43
0
0
min
-29
-37
-43
-75
0
0
0
0
84
-29
-37
-25
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79
© Patrick Winston


max
-43
min
-43
-46
max
-29
-43
-46
26
min
-29
-37
-43
-75
-51
-46
-13
26
84
-29
-37
-25
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79
© Patrick Winston



Minimax Algorithm

function MINIMAX-DECISION(*state*) **returns** *an action*

  $v \leftarrow$ MAX-VALUE(*state*)
  return the *action* in SUCCESSORS(*state*) with value v

---

function MAX-VALUE(*state*) **returns** *a utility value*

  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s))
  return v

---

function MIN-VALUE(*state*) **returns** *a utility value*

  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow \infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(s))
  return v

12

Properties of minimax

- Complete? Yes (if tree is finite)
- Optimal? Yes (against an optimal opponent)
- Time complexity? $O(b^m)$
- Space complexity? $O(bm)$ (depth-first exploration)

13

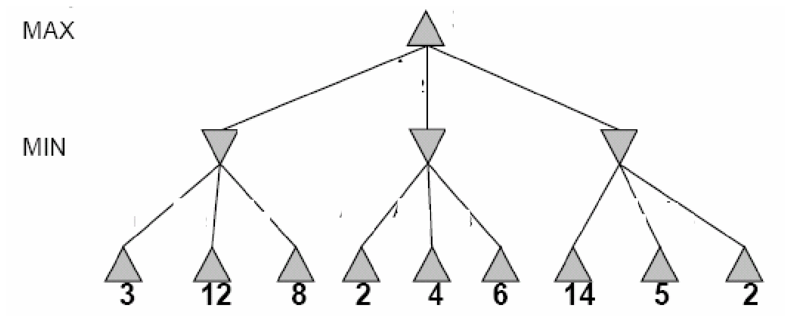
Good enough?

- Chess:
  - branching factor $b \approx 35$
  - game length $m \approx 100$
  - search space $b^m \approx 35^{100} \approx 10^{154}$
- The Universe:
  - number of atoms $\approx 10^{78}$
  - age $\approx 10^{21}$ milliseconds

Can we search more efficiently?

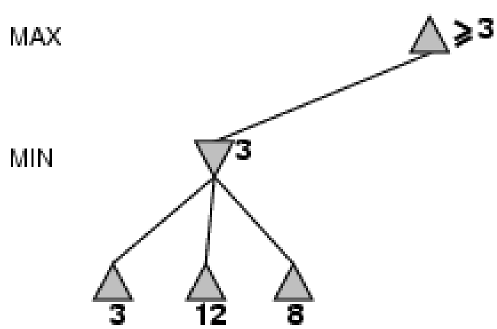
14

Two-Ply Game Tree



15

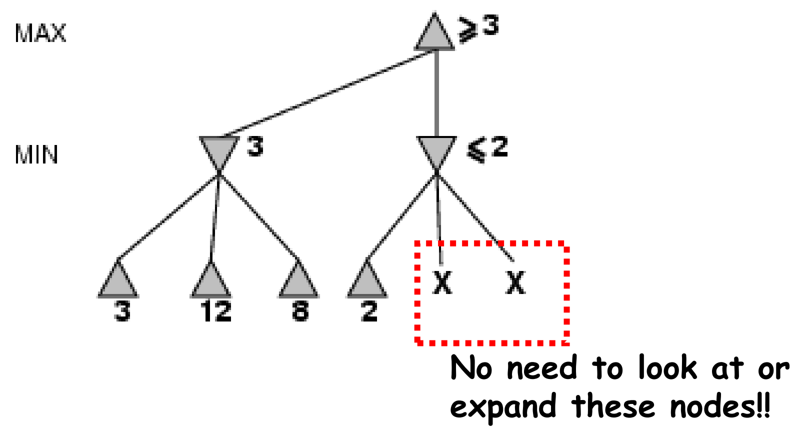
Pruning trees



Minimax algorithm explores depth-first

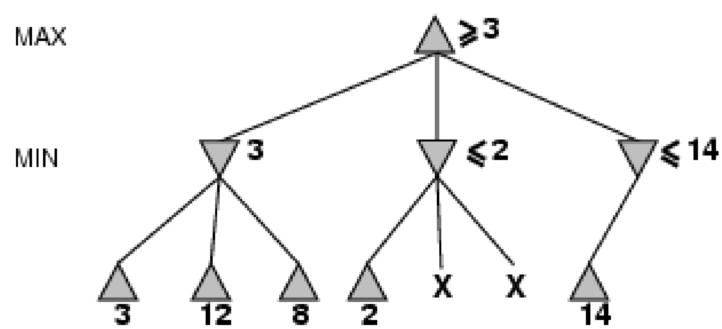
16

Pruning trees



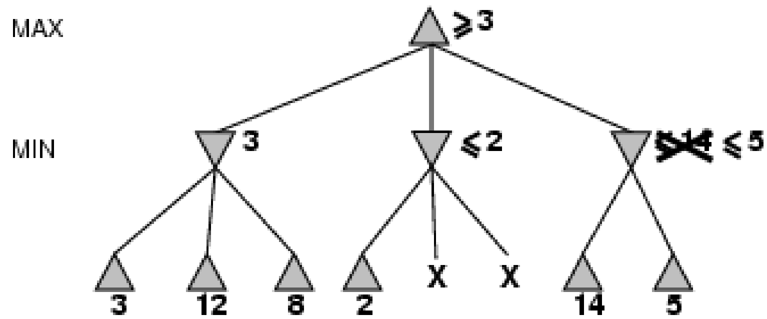
17

Pruning trees



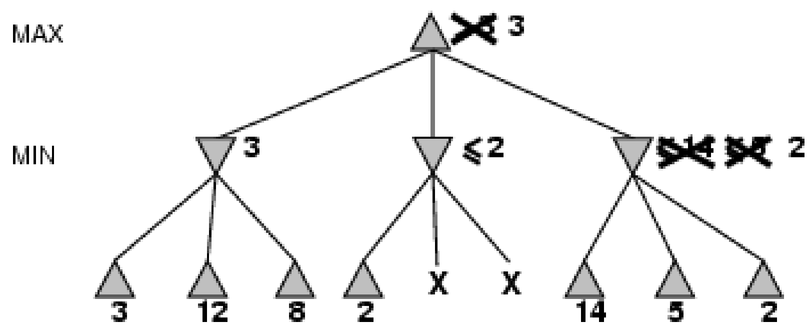
18

Pruning trees

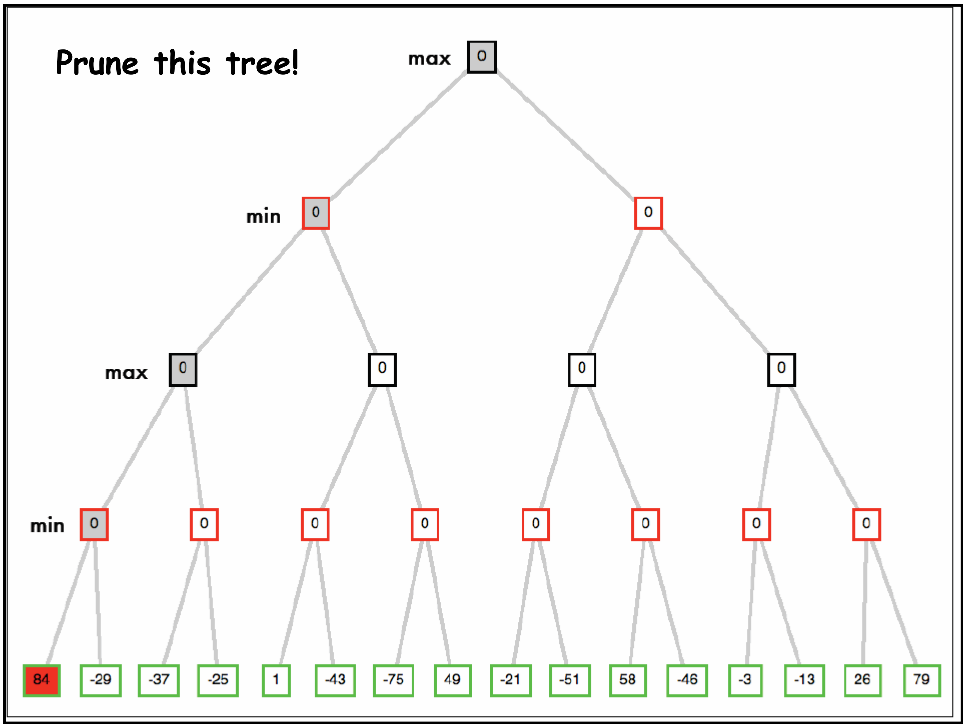


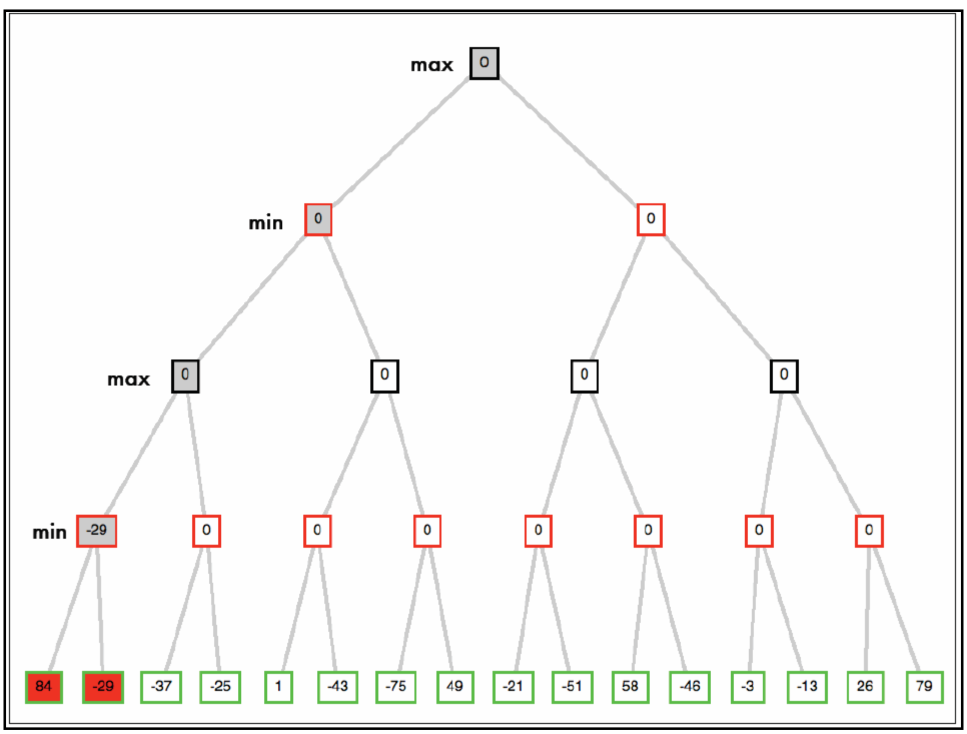
19

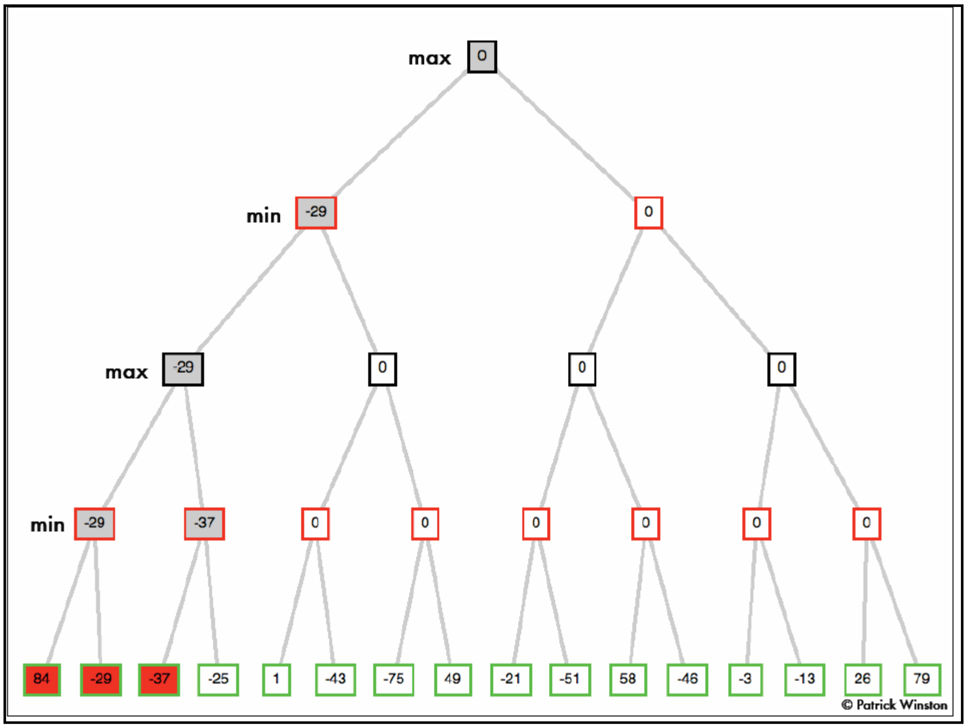
Pruning trees

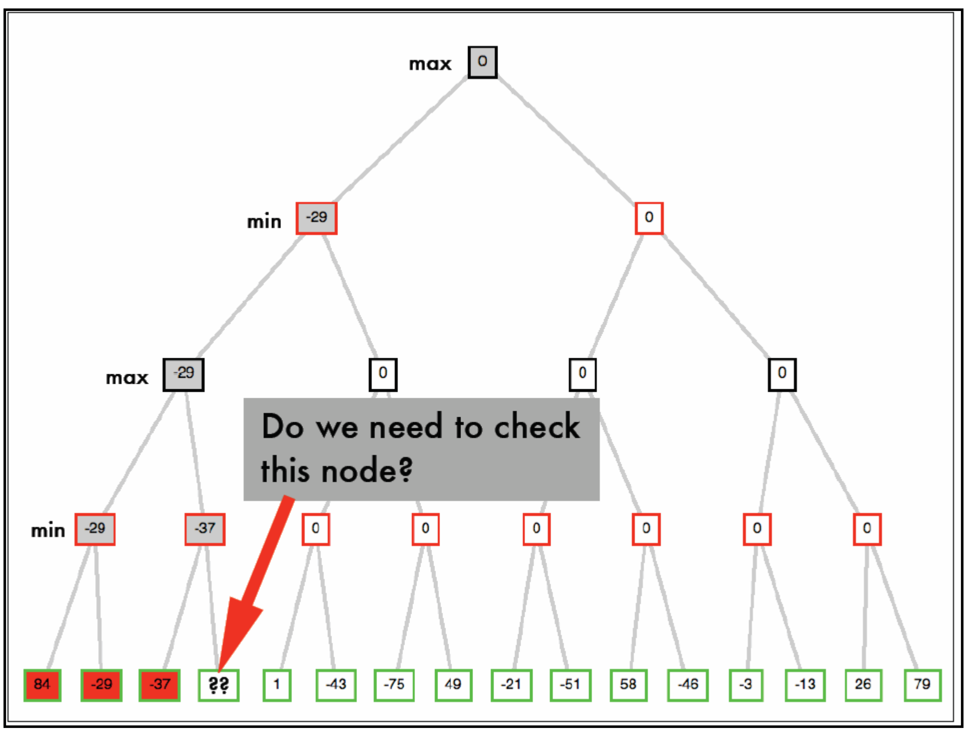


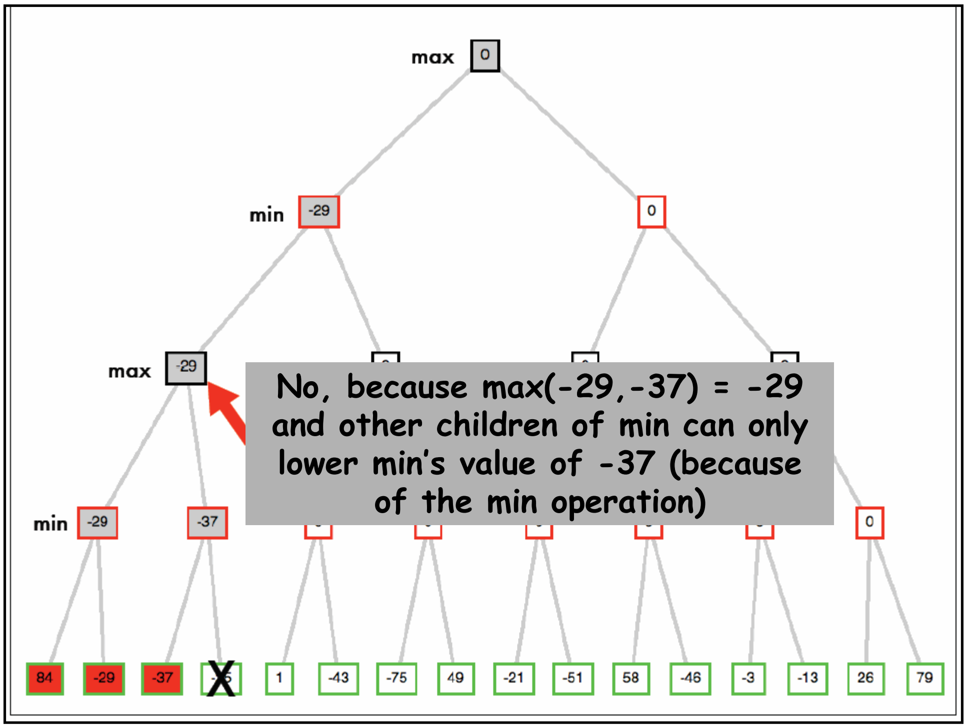
20

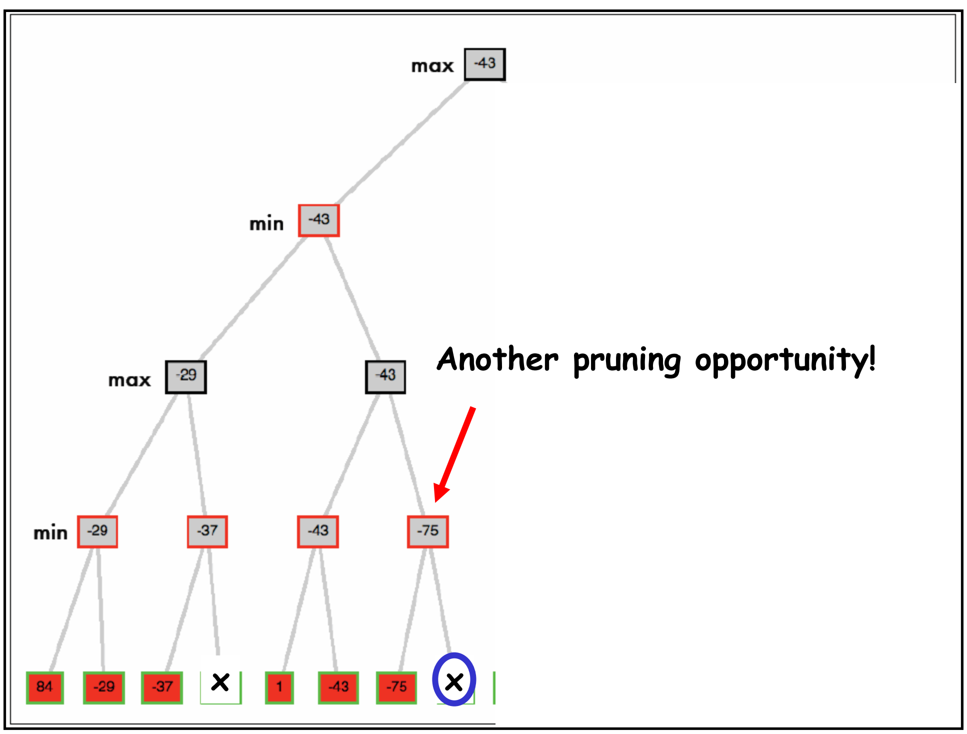

Prune this tree!
max
min
max
min
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
84
-29
-37
-25
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79


max
min
max
min
0
0
0
0
0
0
0
-29
0
0
0
0
0
0
0
84
-29
-37
-25
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79

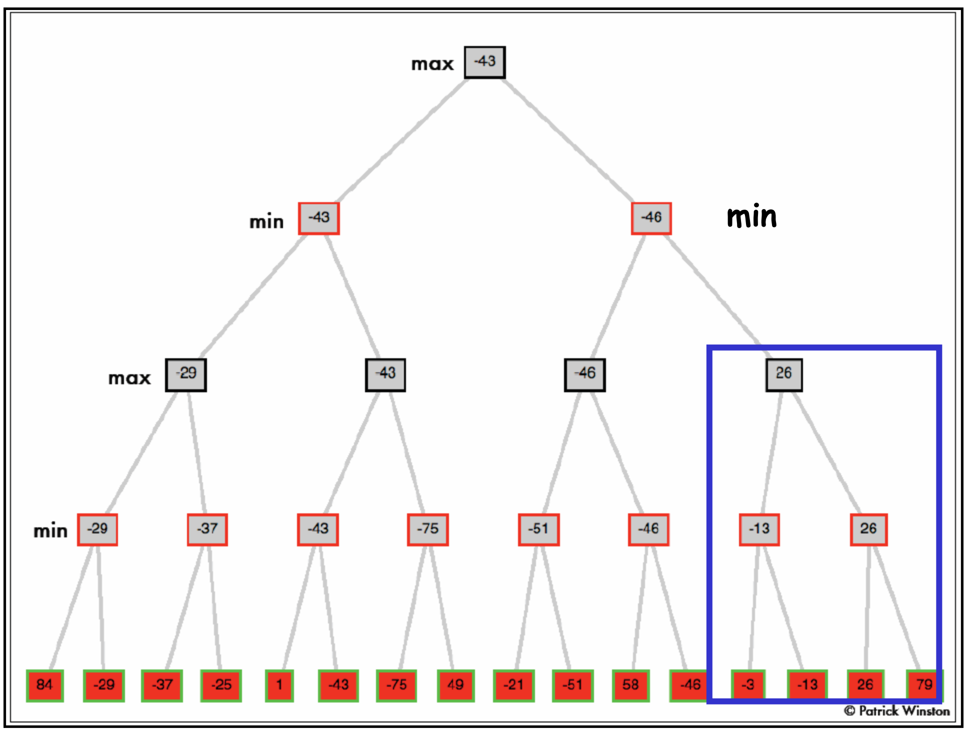

max
0
min
-29
0
max
-29
0
0
0
min
-29
-37
0
0
0
0
0
0
84
-29
-37
-25
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79
© Patrick Winston

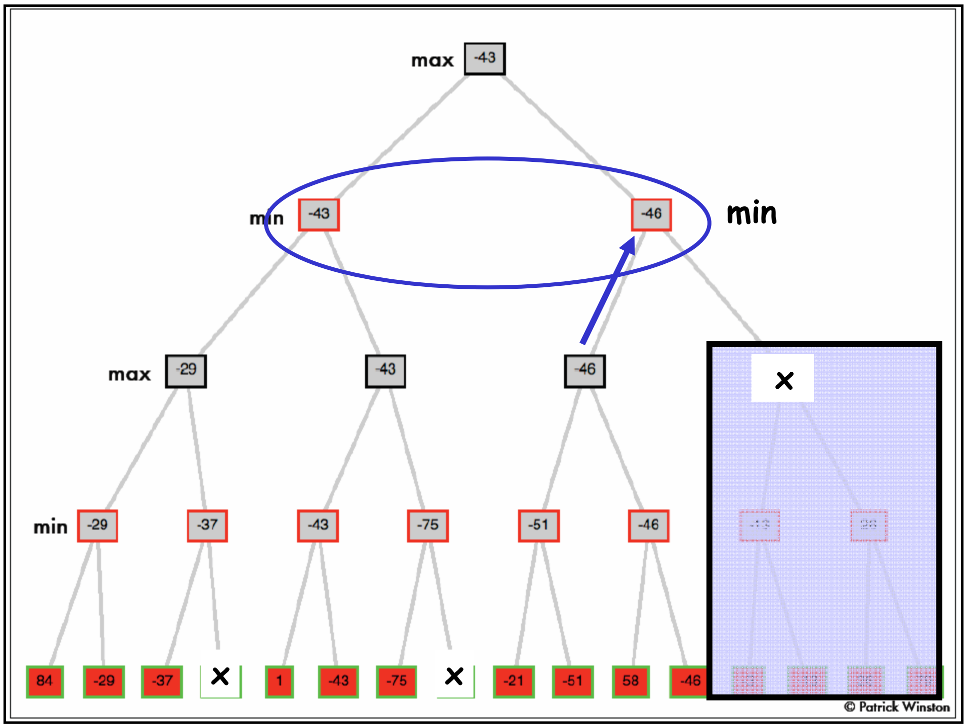

max
0
min
-29
0
max
-29
0
0
0
Do we need to check this node?
min
-29
-37
0
0
0
0
0
0
84
-29
-37
??
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79


max
min
max
min
No, because max(-29,-37) = -29 and other children of min can only lower min's value of -37 (because of the min operation)
84
-29
-37
X
1
-43
-75
49
-21
-51
58
-46
-3
-13
26
79


max
-43
min
-43
max
-29
-43
Another pruning opportunity!
min
-29
-37
-43
-75
84
-29
-37
1
-43
-75

Pruning can eliminate entire subtrees!



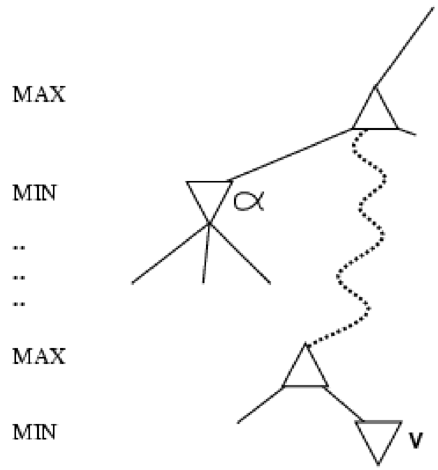


This form of tree pruning is known as alpha-beta pruning

alpha = the highest (best) value for MAX along path
beta = the lowest (best) value for MIN along path

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*



The α-β algorithm

(minimax with four lines of added code)

function ALPHA-BETA-SEARCH(*state*) **returns** *an action*
  inputs: *state*, current state in game
  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  return the *action* in SUCCESSORS(*state*) with value v

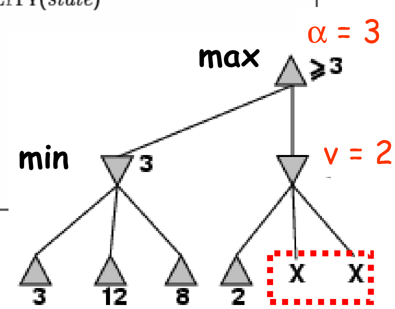
---

function MAX-VALUE($state, \alpha, \beta$) **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*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s, α, β))
New { **if** $v \geq \beta$ **then return** v → Pruning
    $\alpha \leftarrow$ MAX(α, v) }
  return v

The α-β algorithm (cont.)

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*)
  $v \leftarrow +\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
    if $v \leq \alpha$ **then return** v → Pruning
    $\beta \leftarrow$ MIN(β, v)
  return v



33

Properties of α-β

- Pruning does not affect final result
- Good move ordering improves effectiveness of pruning
- With "perfect ordering," time complexity = $O(b^{m/2})$
  allows us to search deeper – doubles depth of search
- A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)

34

Good enough?

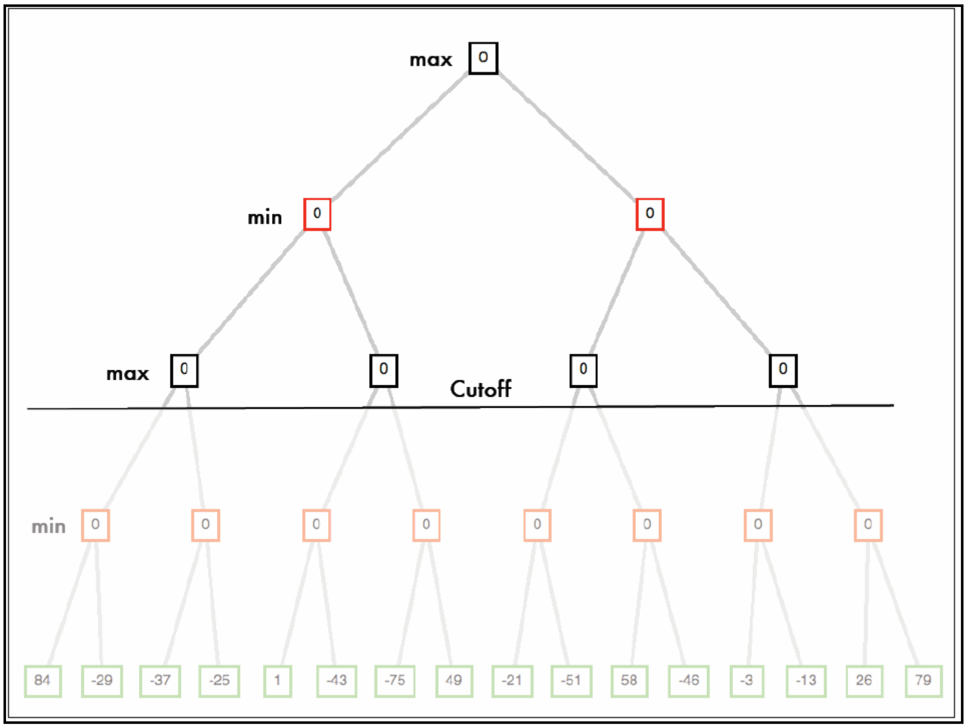
- Chess:
  - branching factor $b \approx 35$
  - game length $m \approx 100$
  - α-β search space $b^{m/2} \approx 35^{50} \approx 10^{77}$
- The Universe:
  - number of atoms $\approx 10^{78}$
  - age $\approx 10^{21}$ milliseconds

35

Can we do better?

- Strategies:
  - search to a fixed depth (cut off search)
  - iterative deepening (most common)
  - stop only at 'quiescent' nodes
    - Unlikely to change wildly in value in near future

36



Evaluation Function

- When search space is too large, create game tree up to a certain depth only.
- Art is to estimate utilities of positions that are not terminal states.
- Example of simple evaluation criteria in chess:
  - Material worth: pawn=1, knight =3, rook=5, queen=9.
  - Other: king safety, good pawn structure
  - Rule of thumb: 3-point advantage = certain victory

eval(s) =

```
w1 * material(s) +
w2 * mobility(s) +
w3 * king safety(s) +
w4 * center control(s) + ...
```

38

Cutting off search

- Does it work in practice?
  If $b^m = 10^6$ and $b=35 \Rightarrow m=4$

- 4-ply lookahead is a hopeless chess player!
  - 4-ply ≈ human novice
  - 8-ply ≈ typical PC, human master
  - 12-ply ≈ Deep Blue, Kasparov

39

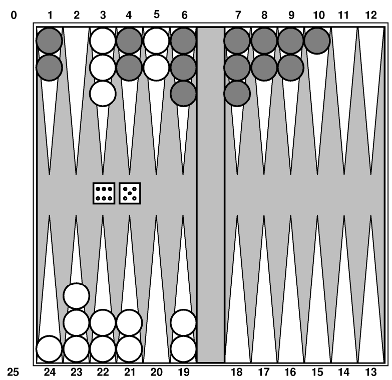
Transposition Tables

- Game trees contain repeated states
- In chess, e.g., the game tree may have 35^{100} nodes, but there are only 10^{40} different board positions
- Similar to *closed list* in graph-search, maintain a transposition table

  Got its name from the fact that the same state is reached by a transposition of moves.

40

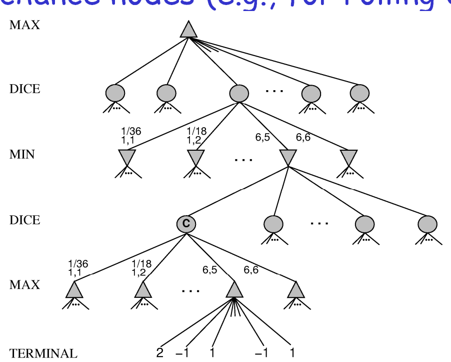
Games that Include an Element of Chance



White has just rolled 6-5 and has 4 legal moves.

Game Tree for Games with an Element of Chance

- In addition to MIN- and MAX nodes, we need chance nodes (e.g., for rolling dice).



Expectiminimax Algorithm:
For chance nodes, compute expected value over successors

- Search costs increase: Instead of $O(b^d)$, we get $O((bn)^d)$, where n is the number of chance outcomes.

Imperfect Information

- E.g. card games, where opponents' initial cards are unknown
- Idea: For all deals consistent with what you can see
  - compute the minimax value of available actions for each of possible deals
  - compute the expected value over all deals

Game Playing in Practice

- Checkers: Chinook ended 40 year reign of human world champion Marion Tinsley in 1994; used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions (!)
- Chess: Deep Blue defeated human world champion Gary Kasparov in a 6 game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation functions, and undisclosed methods for extending some lines of search up to 40 ply
- Othello: human champions refuse to play against computers because software is too good
- Go: human champions refuse to play against computers because software is too bad

Summary of Game Tree Search

- Basic idea: minimax -- too slow for most games
- Alpha-Beta pruning can increase max depth by factor up to 2
- Limited depth search may be necessary
- Static evaluation functions necessary for limited depth search and help alpha-beta
- Opening game and End game databases can help
- Computers can beat humans in some games (checkers, chess, othello) but not in others (Go)

45