Introduction to Games, Min Max, and α-β Pruning

Algorithms & Games

What is a (mathematical) game?

- 2 players [A(lice), B(ob)/ L(eft), R(ight)/ . . .]

What is a (mathematical) game?

- 2 players [A(lice), B(ob)/ L(eft), R(ight)/ ...]
- The players move in turns (A,B,A,B,...)

What is a (mathematical) game?

- 2 players [A(lice), B(ob)/ L(eft), R(ight)/ ...]
- The players move in turns (A,B,A,B,...)
- Both players have complete information (no hidden cards,...)

What is a (mathematical) game?

- 2 players [A(lice), B(ob)/ L(eft), R(ight)/ ...]
- The players move in turns (A,B,A,B,...)
- Both players have complete information (no hidden cards,...)
- No randomness (flipping coins, rolling dice,...)

What is a (mathematical) game?

- 2 players [A(lice), B(ob)/ L(eft), R(ight)/ ...]
- The players move in turns (A,B,A,B,...)
- Both players have complete information (no hidden cards,...)
- No randomness (flipping coins, rolling dice,...)
- A (finite) set of positions, one (or more) marked as starting position

What is a game, cont.?

- For each position, there exists a set of successors, (possibly empty)

What is a game, cont.?

- For each position, there exists a set of successors, (possibly empty)
- A players legal move: transformation from one position to a successor

What is a game, cont.?

- For each position, there exists a set of successors, (possibly empty)
- A players legal move: transformation from one position to a successor
- Normal play: the first player which can NOT move loses (the other player wins)

What is a game, cont.?

- For each position, there exists a set of successors, (possibly empty)
- A players legal move: transformation from one position to a successor
- Normal play: the first player which can NOT move loses (the other player wins)
- Every game ends after a finite number of moves

What is a game, cont.?

- For each position, there exists a set of successors, (possibly empty)
- A players legal move: transformation from one position to a successor
- Normal play: the first player which can NOT move loses (the other player wins)
- Every game ends after a finite number of moves
- No draws

Chocolate game (chomp)

Chocolate game (chomp)

Chocolate game (chomp)

Chocolate game (chomp)

Winner: player taking the last (non toxic) piece of chocolate

Chocolate game (chomp)

Chocolate game (chomp)

Chocolate game (chomp)

A

Chocolate game (chomp)

B

Chocolate game (chomp)

A

Who wins a game?

- Which player wins the game (A, B)?

Who wins a game?

- Which player wins the game (A, B)?
  - First player (starting) or second player?

Who wins a game?

- Which player wins the game (A, B)?
  - First player (starting) or second player?
  - Assume both players play optimal
    There are 'First-Player-win' and 'Second-Player win' games

Who wins a game?

- Which player wins the game (A, B)?
  - First player (starting) or second player?
  - Assume both players play optimal
    There are 'First-Player-win' and 'Second-Player win' games
- What is the optimal strategy?

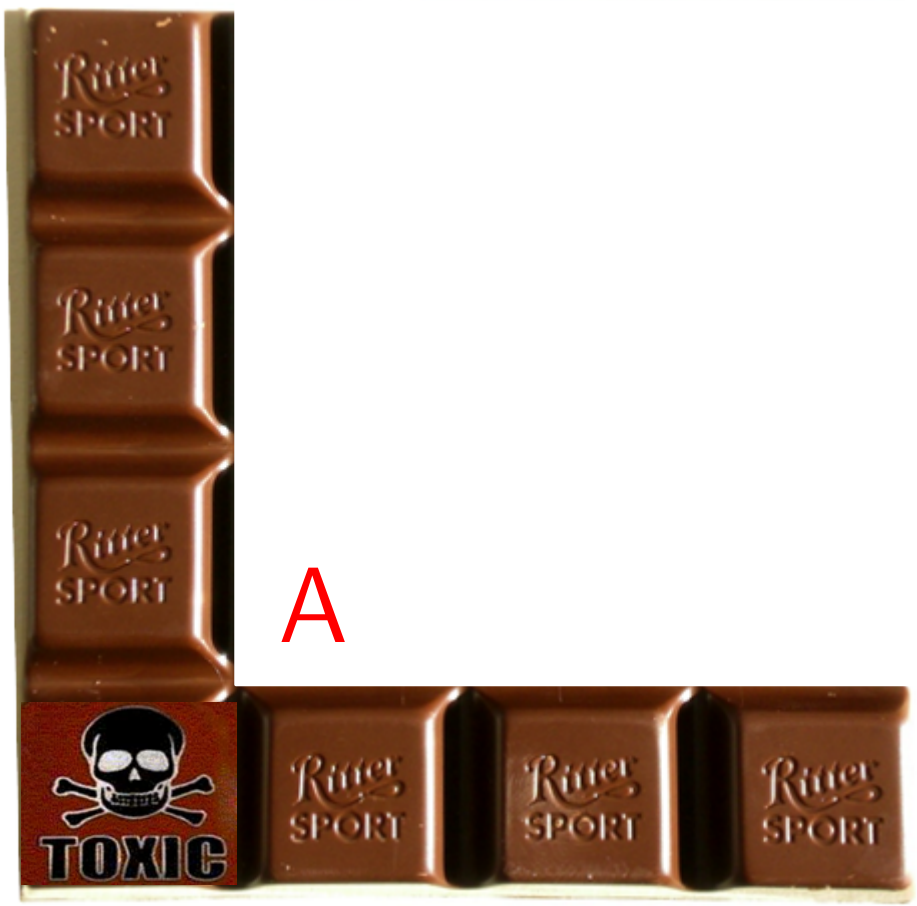
Who wins a game?

- Which player wins the game (A, B)?
  - First player (starting) or second player?
  - Assume both players play optimal
    There are 'First-Player-win' and 'Second-Player win' games
- What is the optimal strategy?

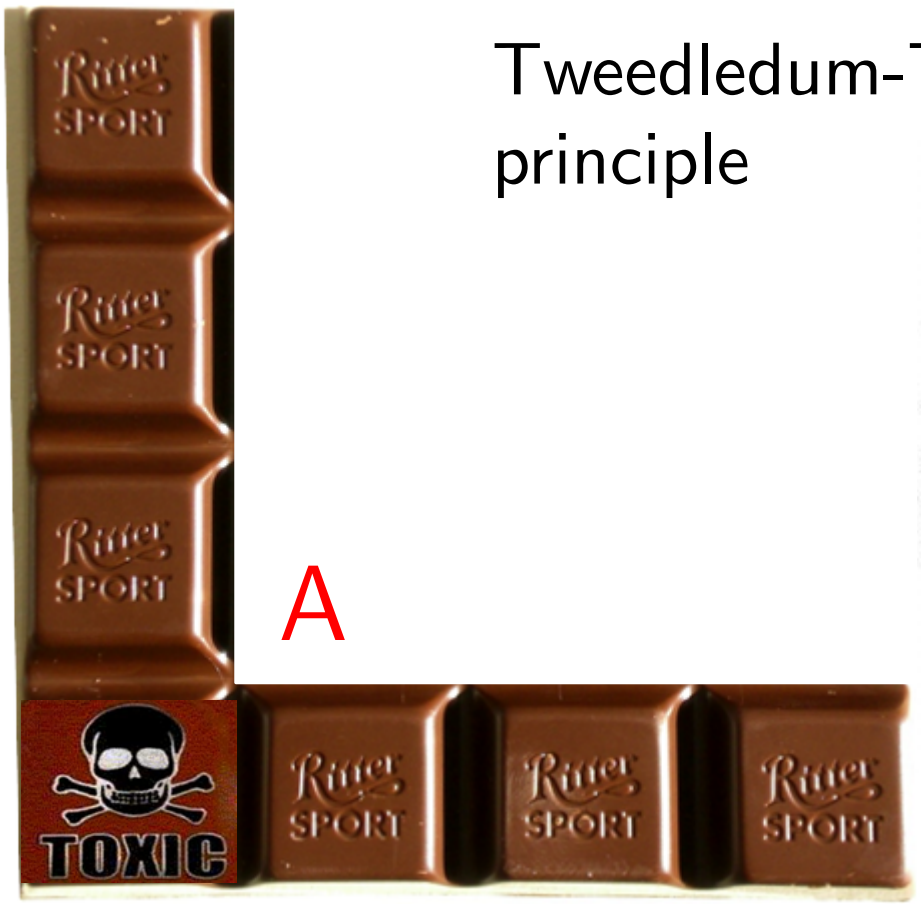
Play Chomp!

Chocolate game (again)

Chocolate game (again)



Chocolate game (again)



A

Tweedledum-Tweedledee-principle

Alice in Wonderland by Lewis Carrol

Chocolate game (and again)

Chomp is a first player win for all sizes

Chocolate game (and again)

Chomp is a first player win for all sizes

Proof by strategy stealing and contradiction!

Chocolate game (and again)

Chomp is a first player win for all sizes

Proof by strategy stealing and contradiction!

Chocolate game (and again)

Chomp is a first player win for all sizes

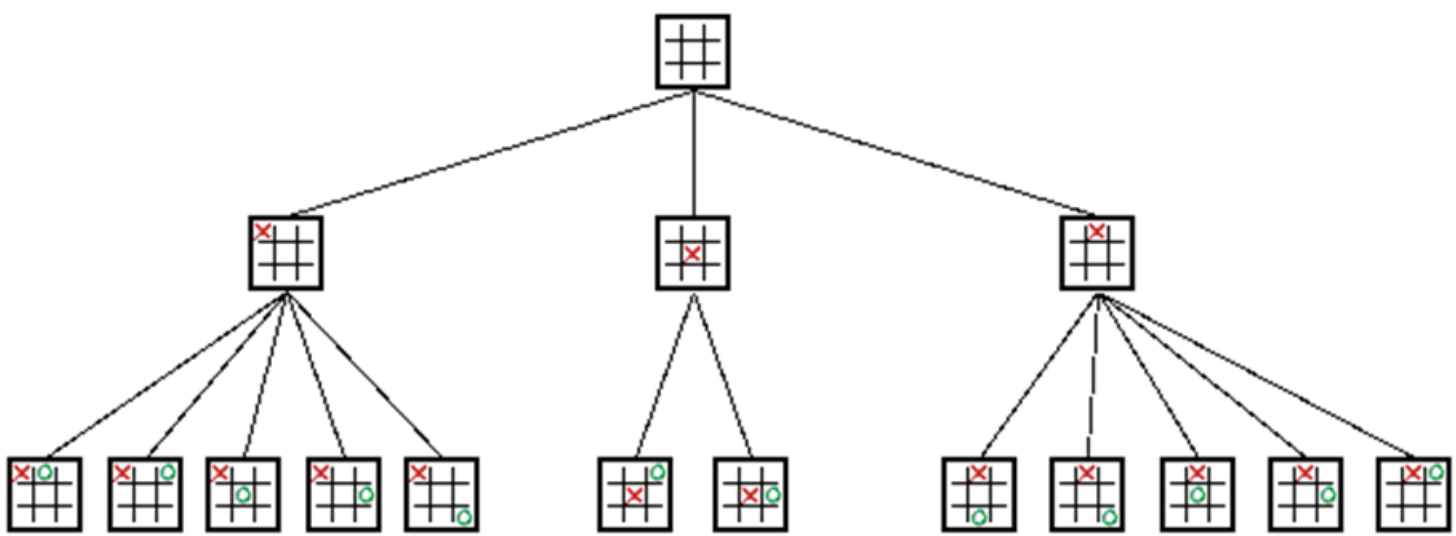
B A

Proof by strategy stealing and contradiction!

Classic Approach

Classic approach for playing 2-player games: game tree

- States are the nodes, moves are the edges of the tree
- Possible moves for a state are child nodes
- The depth of the tree is bounded (look ahead)
- Leaves of the tree are evaluated by using a heuristic



Classic Approach

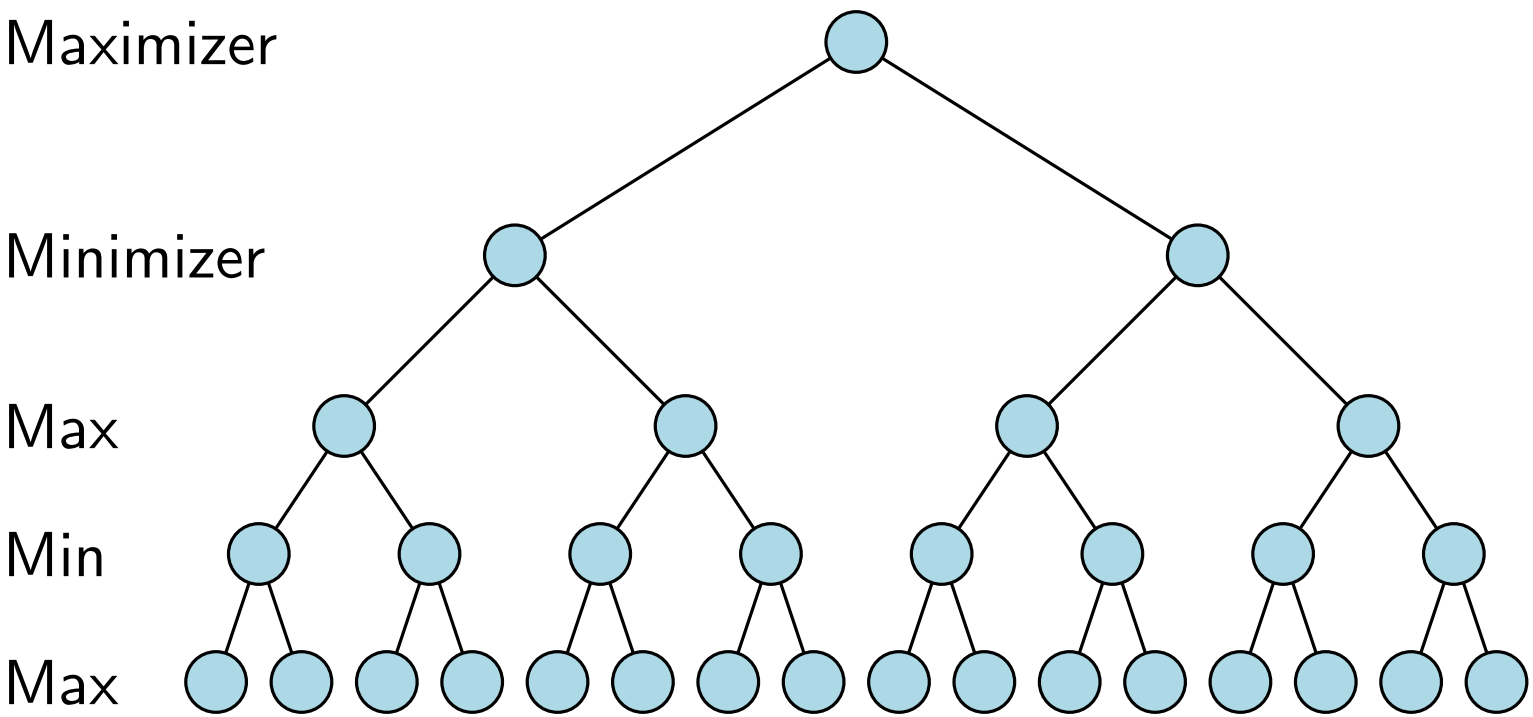
Classic approach for playing 2-player games: game tree

- States are the nodes, moves are the edges of the tree
- Possible moves for a state are child nodes
- The depth of the tree is bounded (look ahead)
- Leaves of the tree are evaluated by using a heuristic

- The value of inner nodes are the maximum (maximizer) or minimum (minimizer) of the values of all child nodes
- One player tries to maximize the score of a state, while the other tries to minimize it
- Termial states (win, loose) get extreme values, draw states neutral values

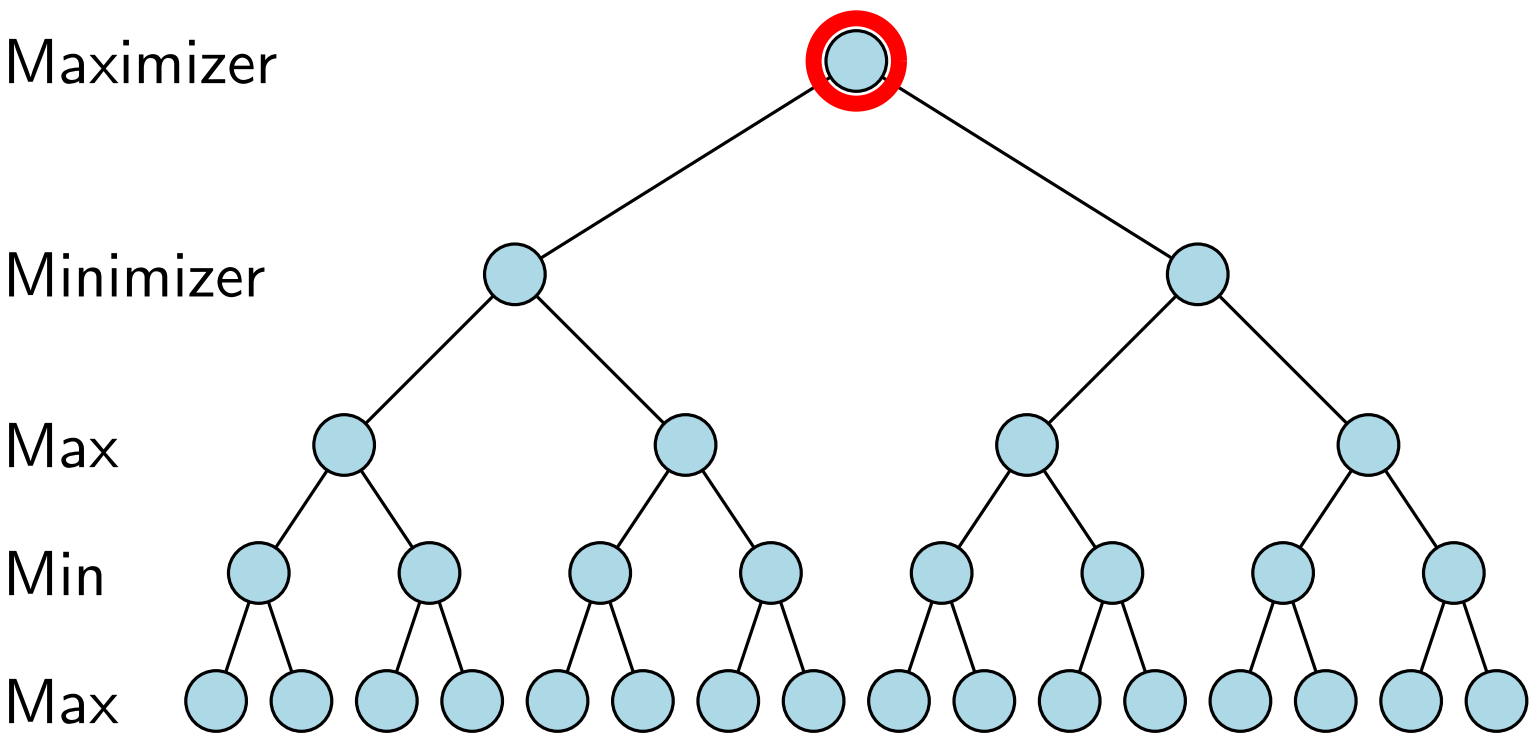
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



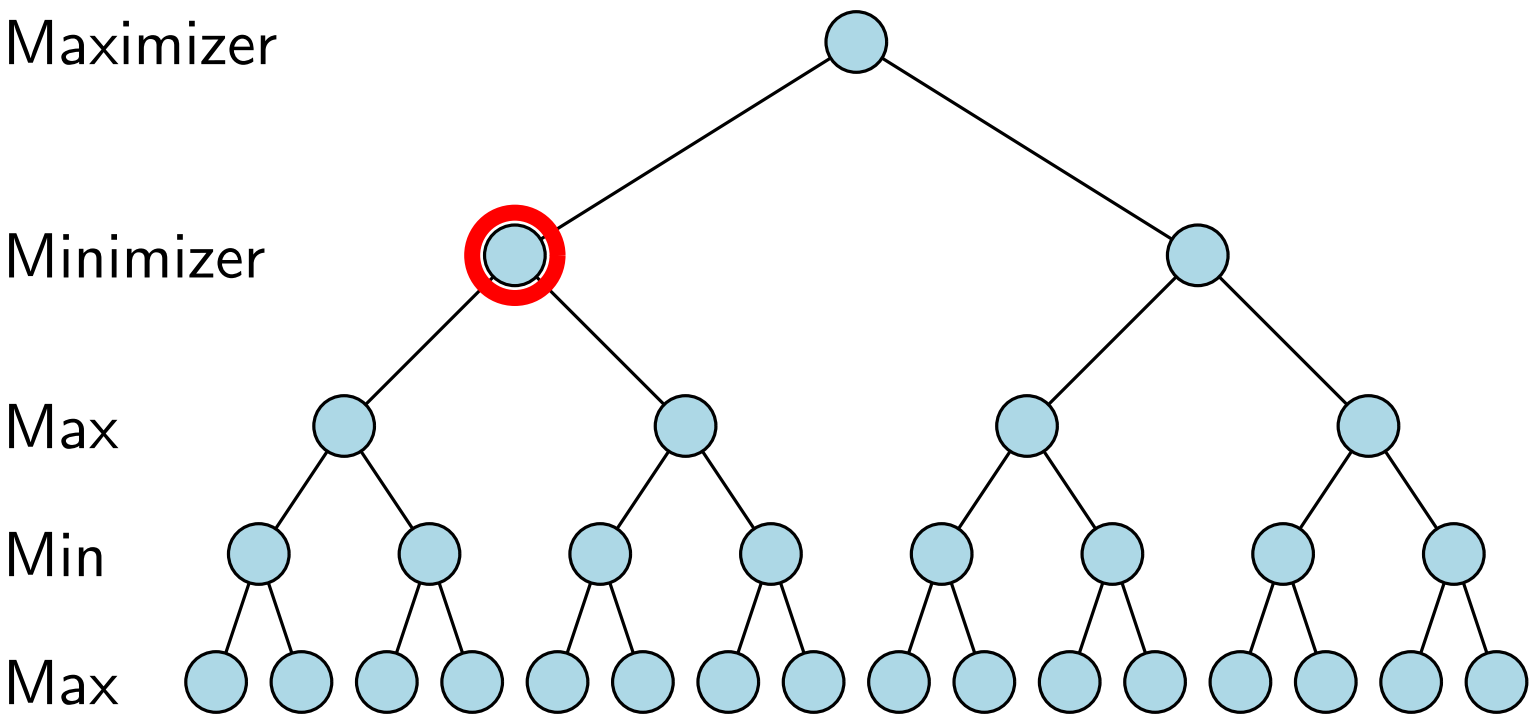
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



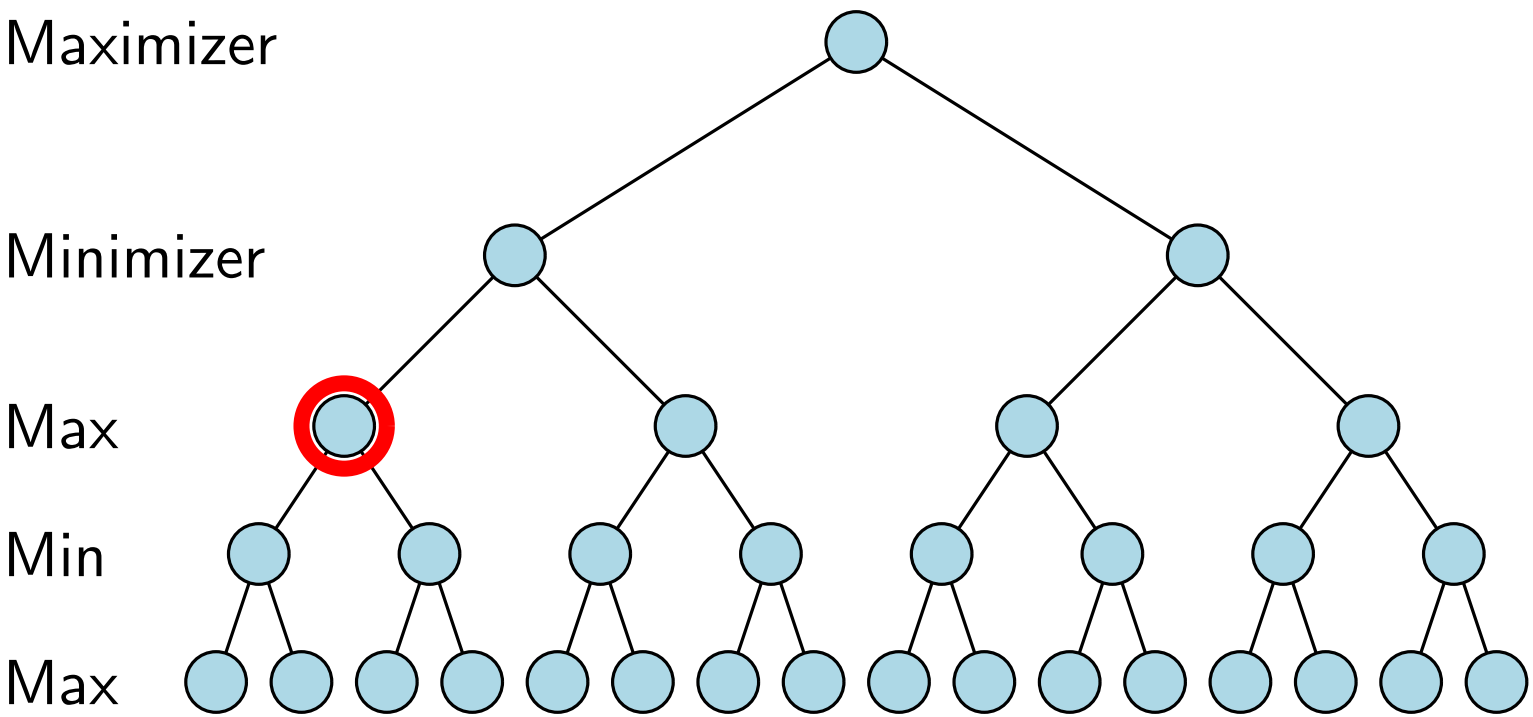
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



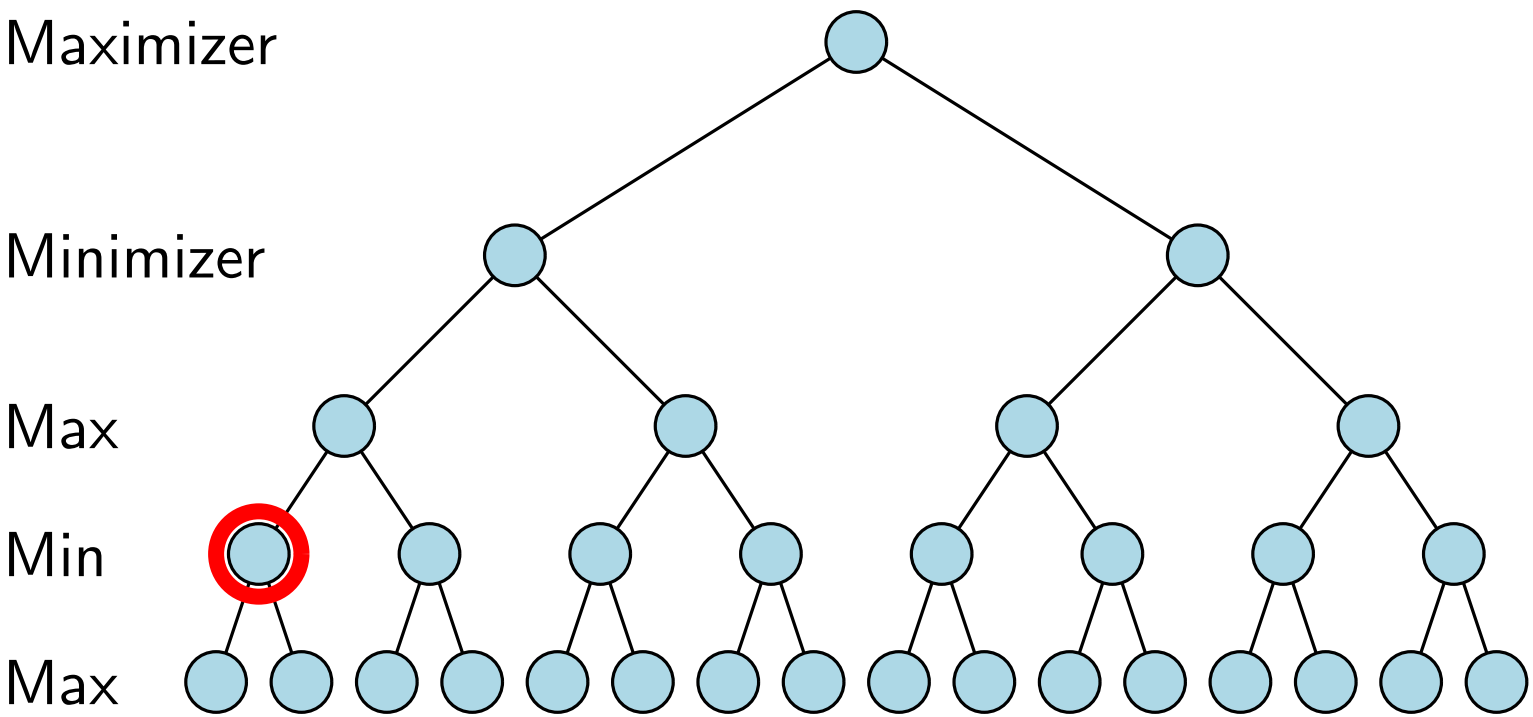
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



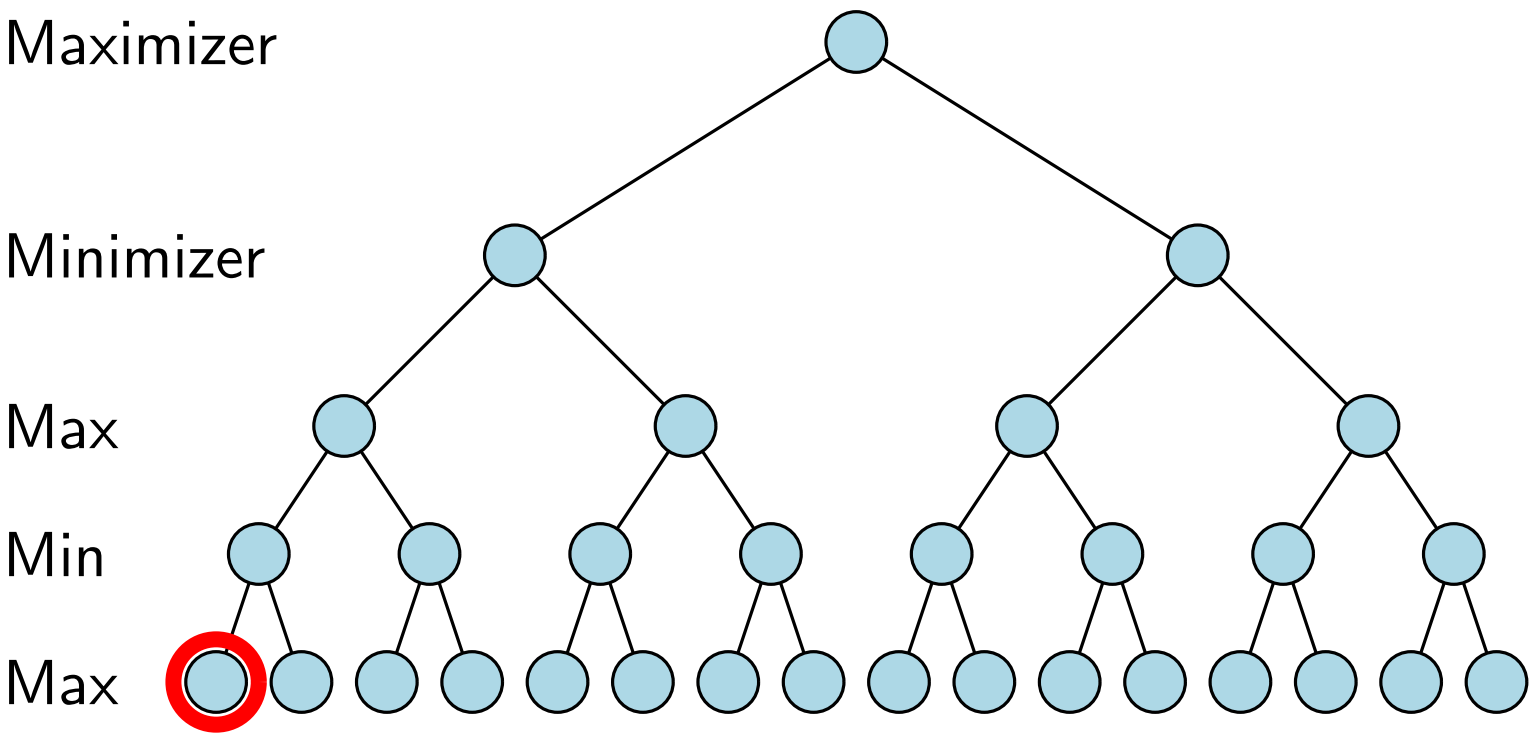
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



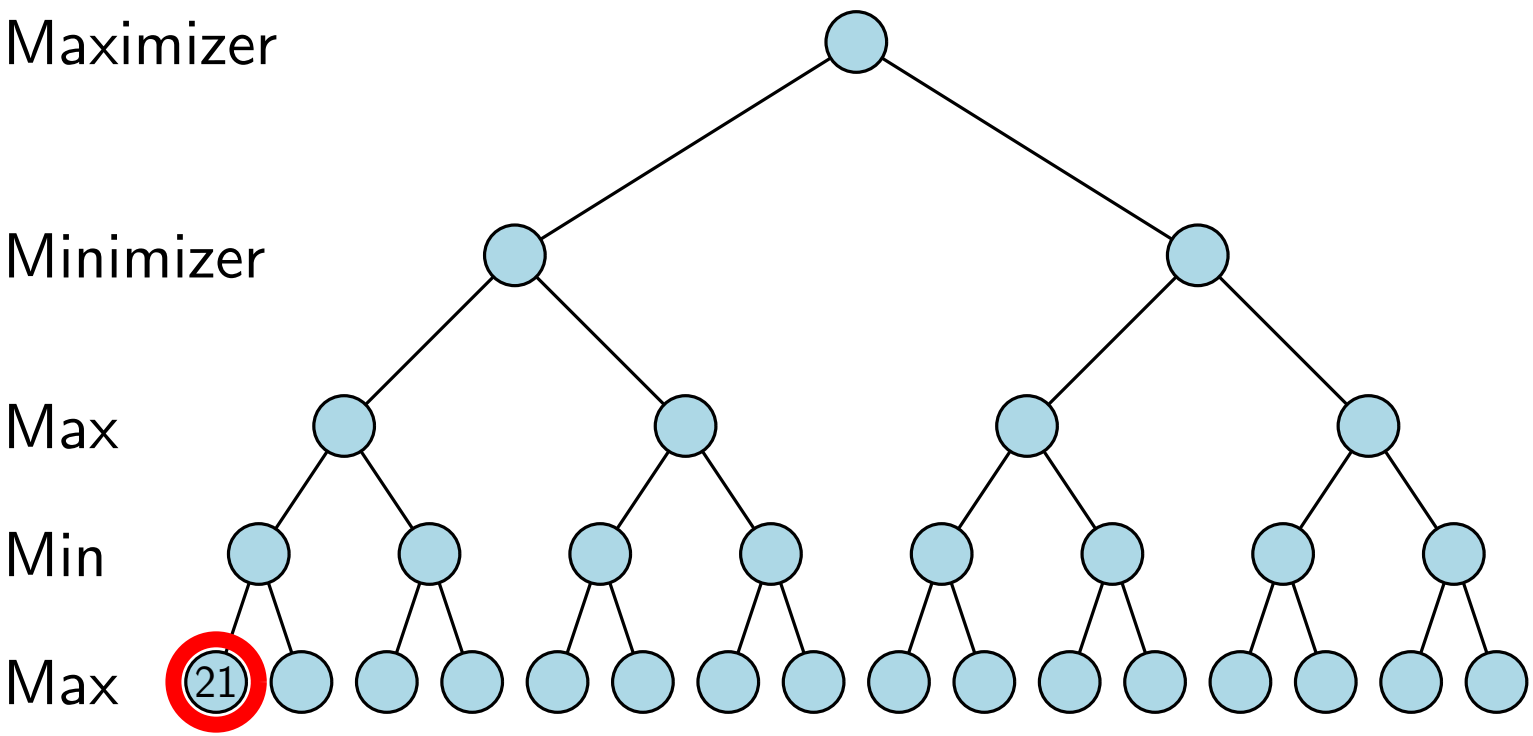
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



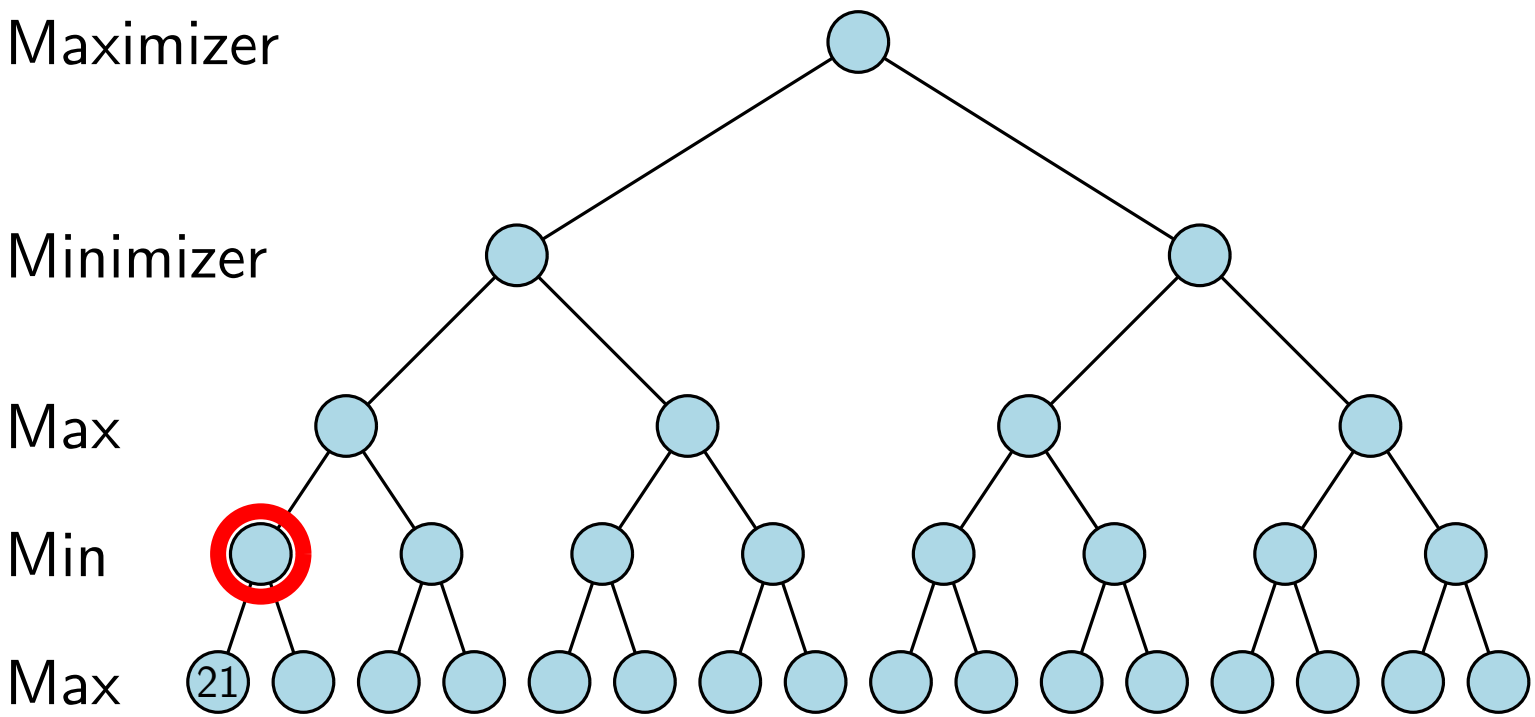
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



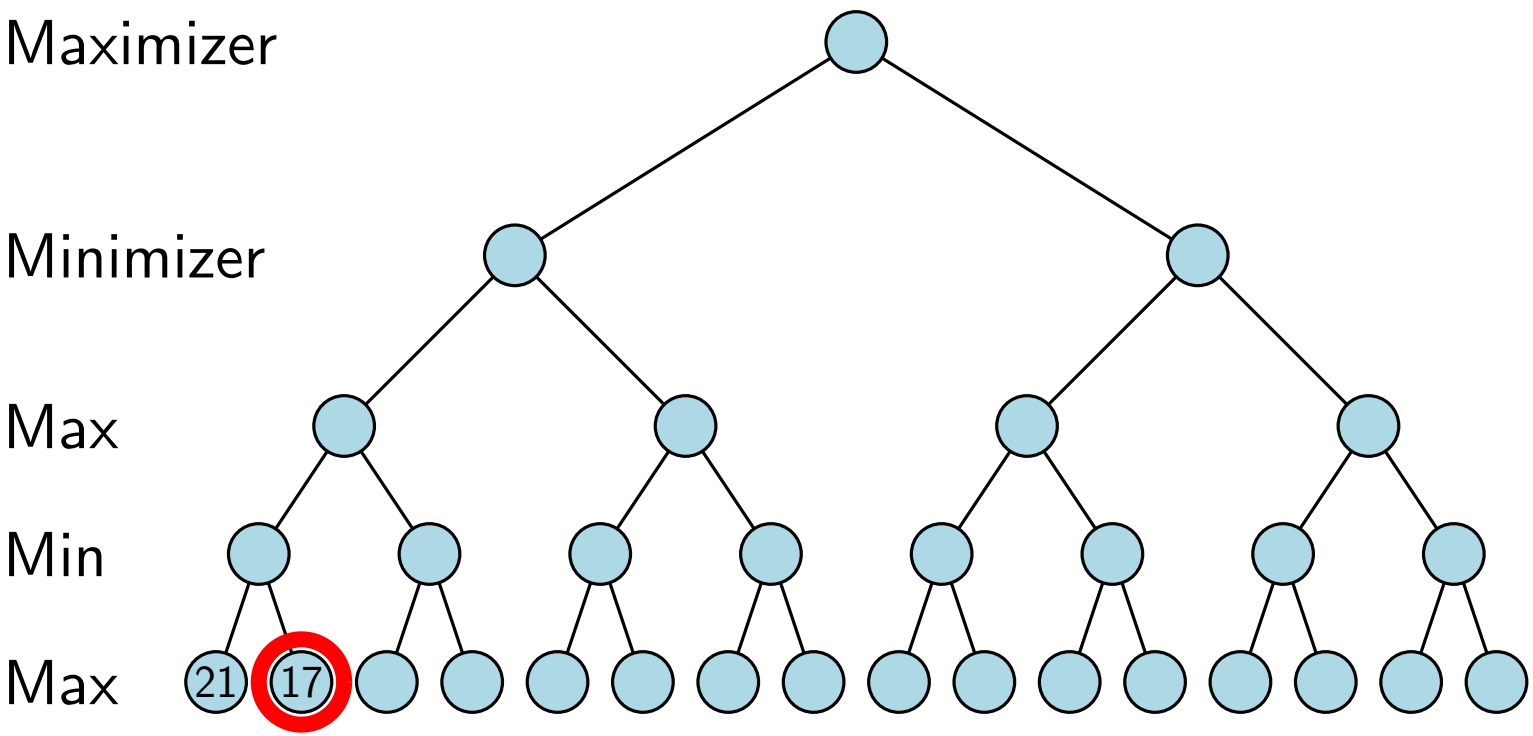
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



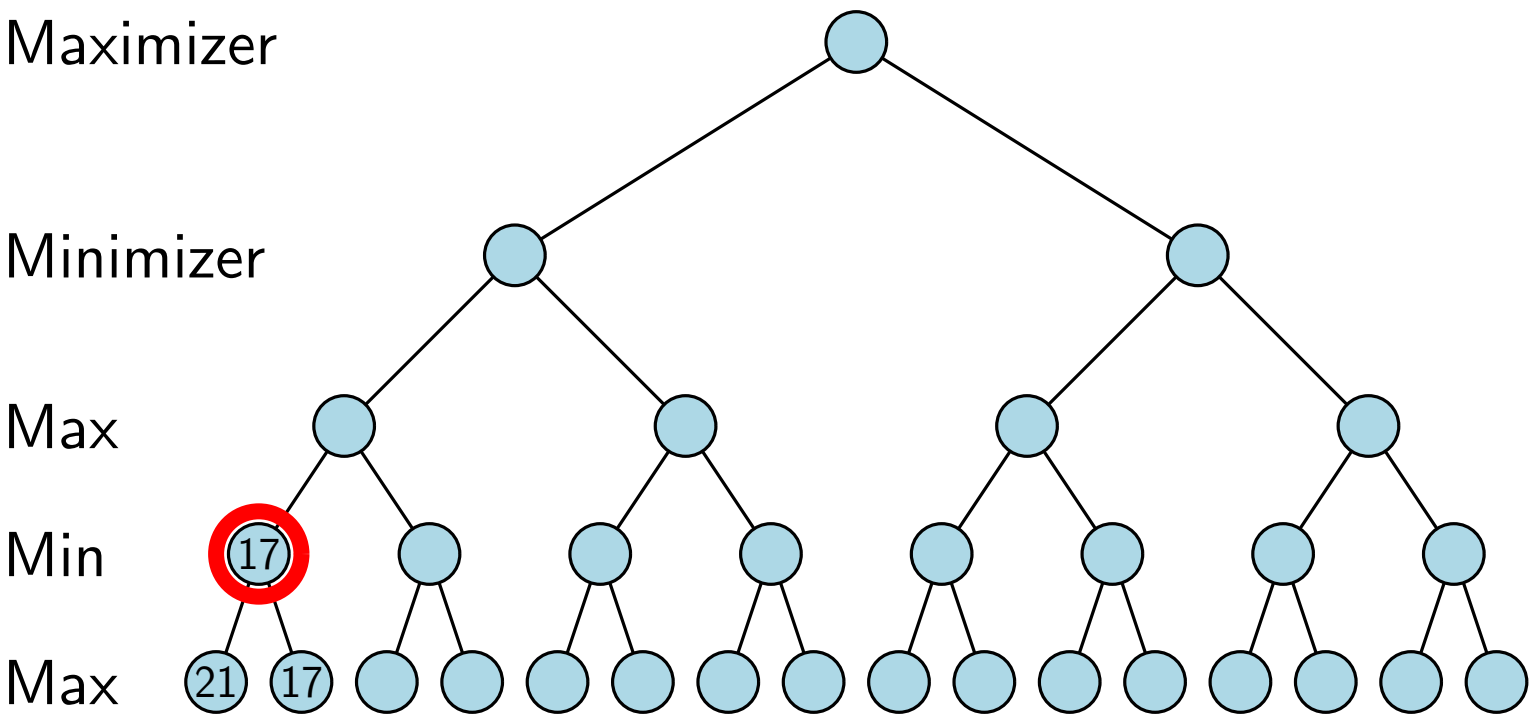
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



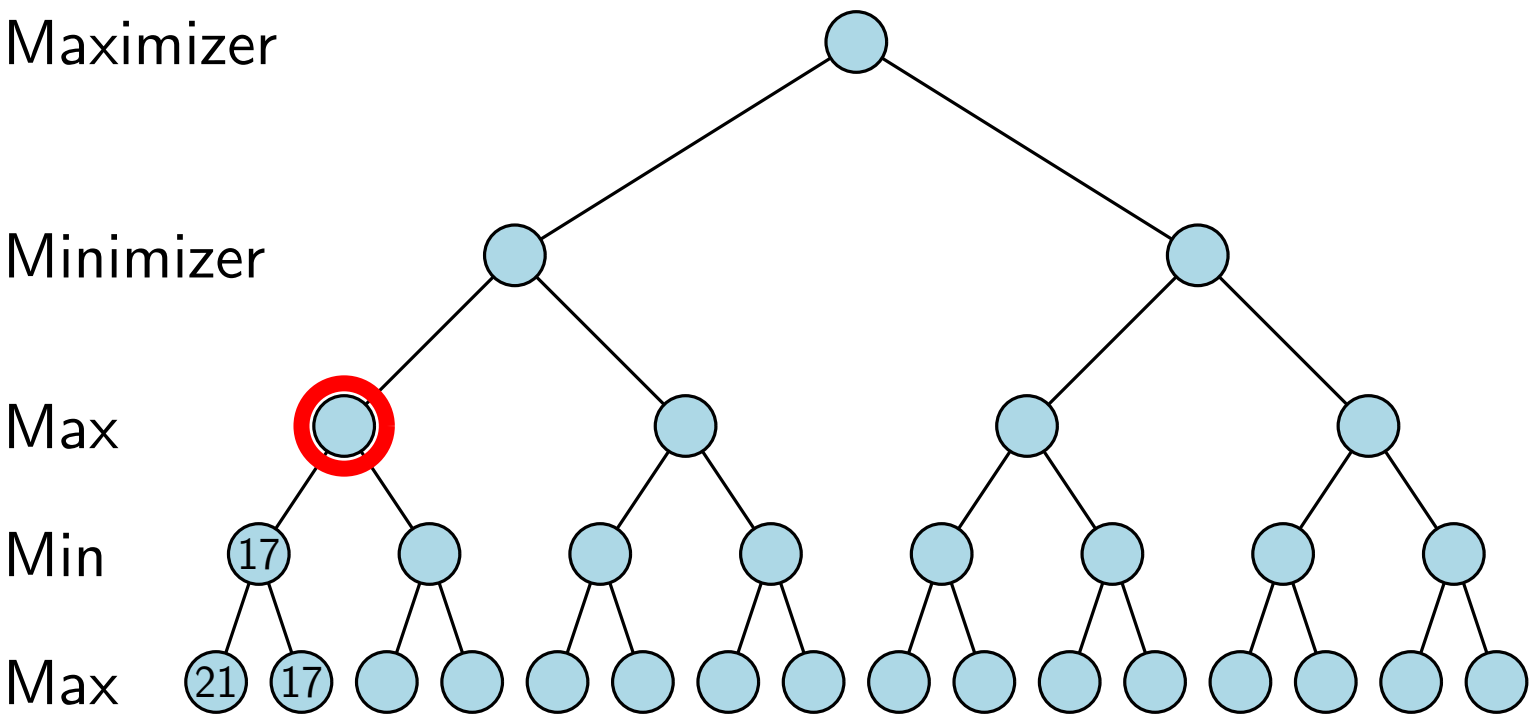
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



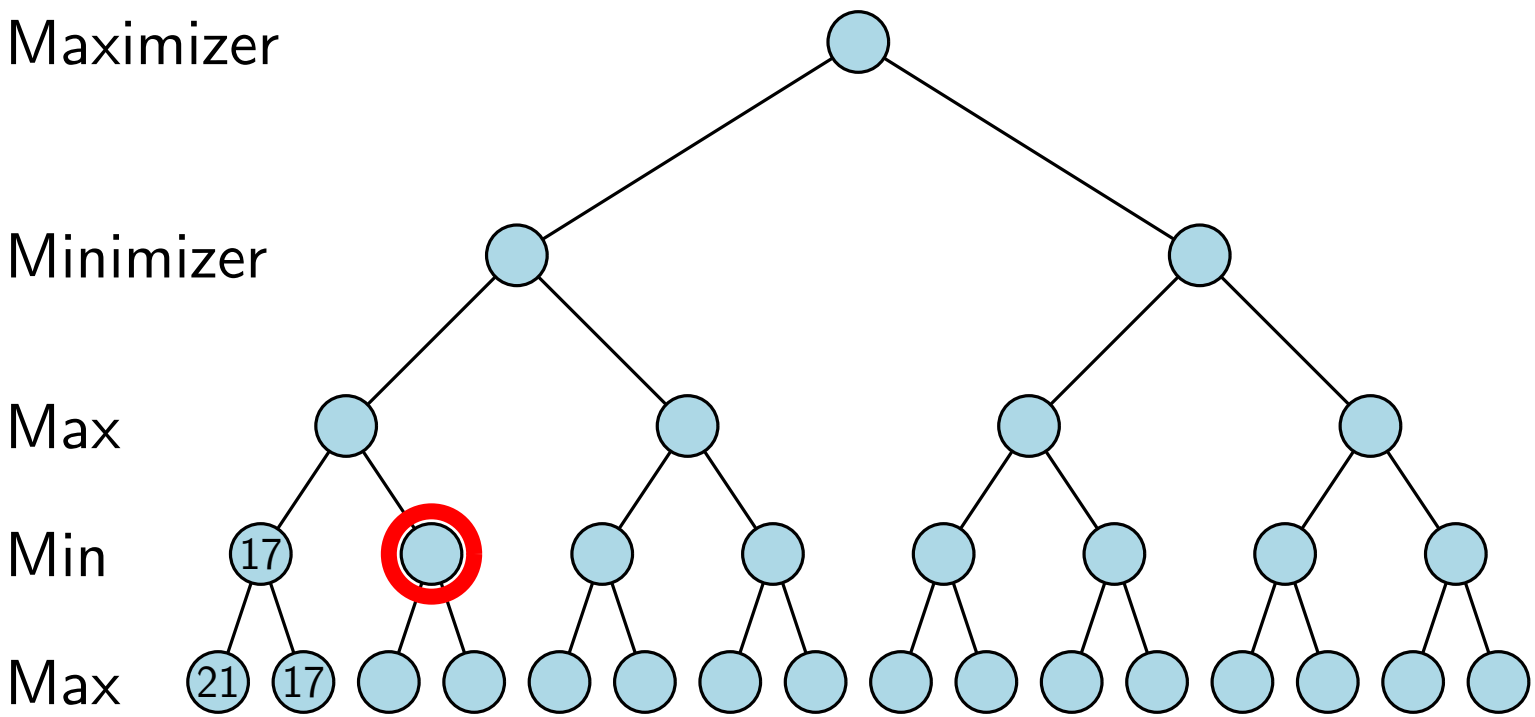
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



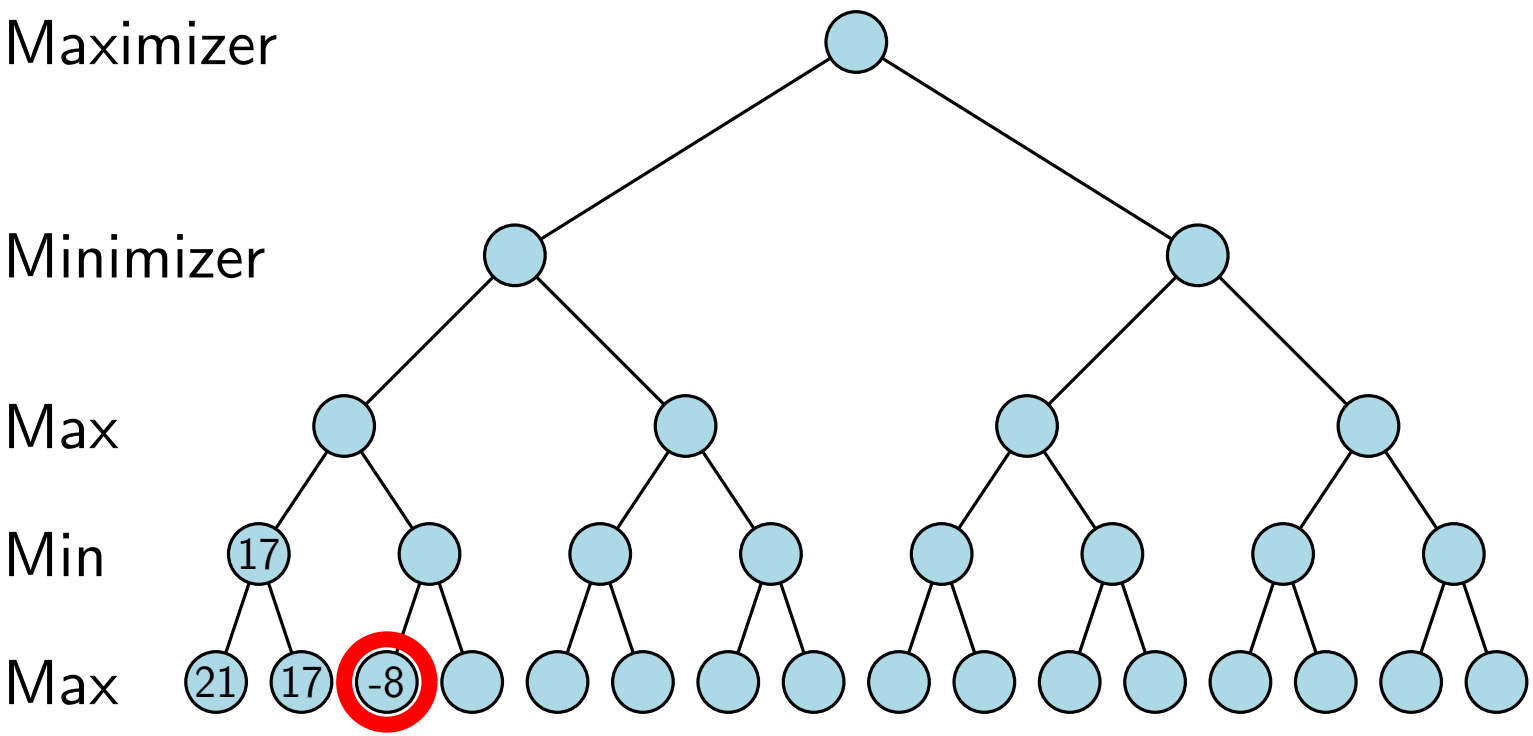
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



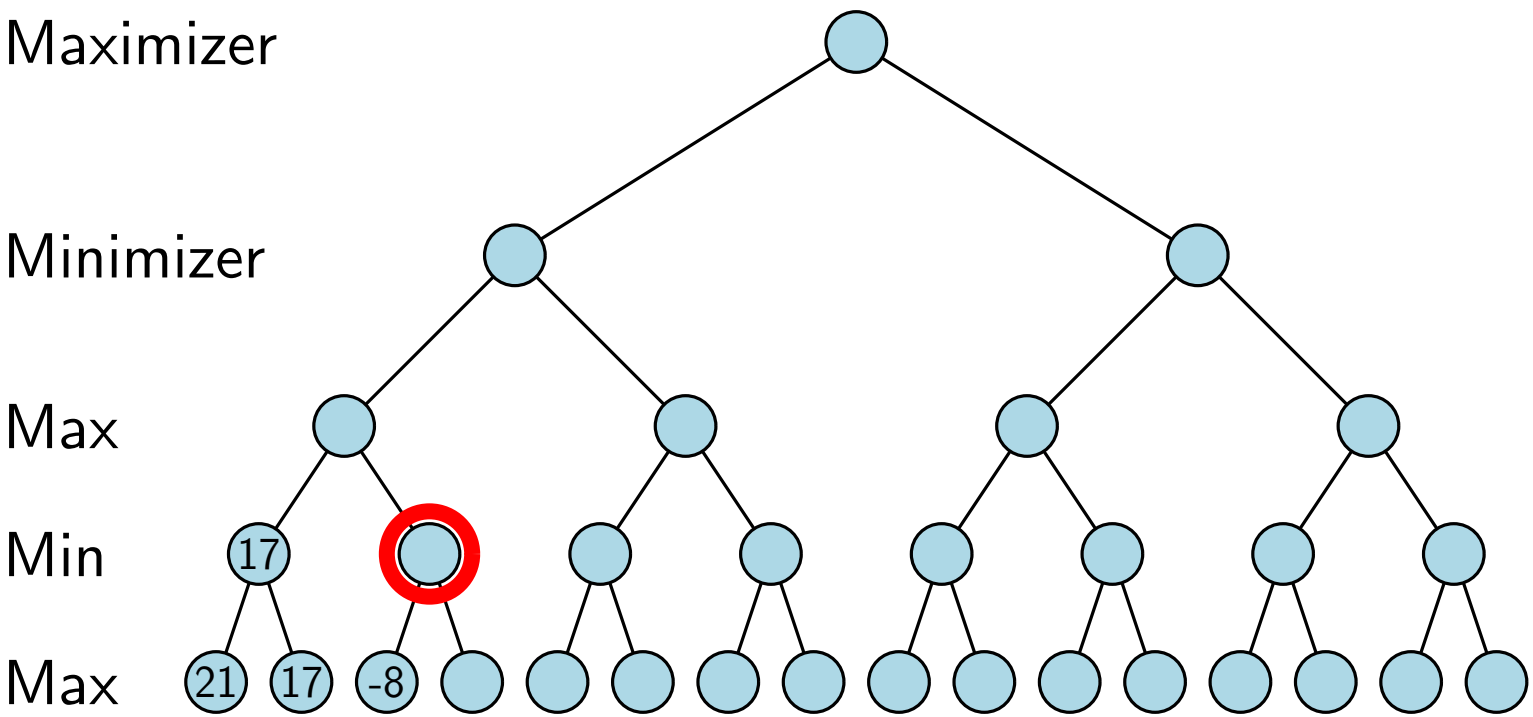
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



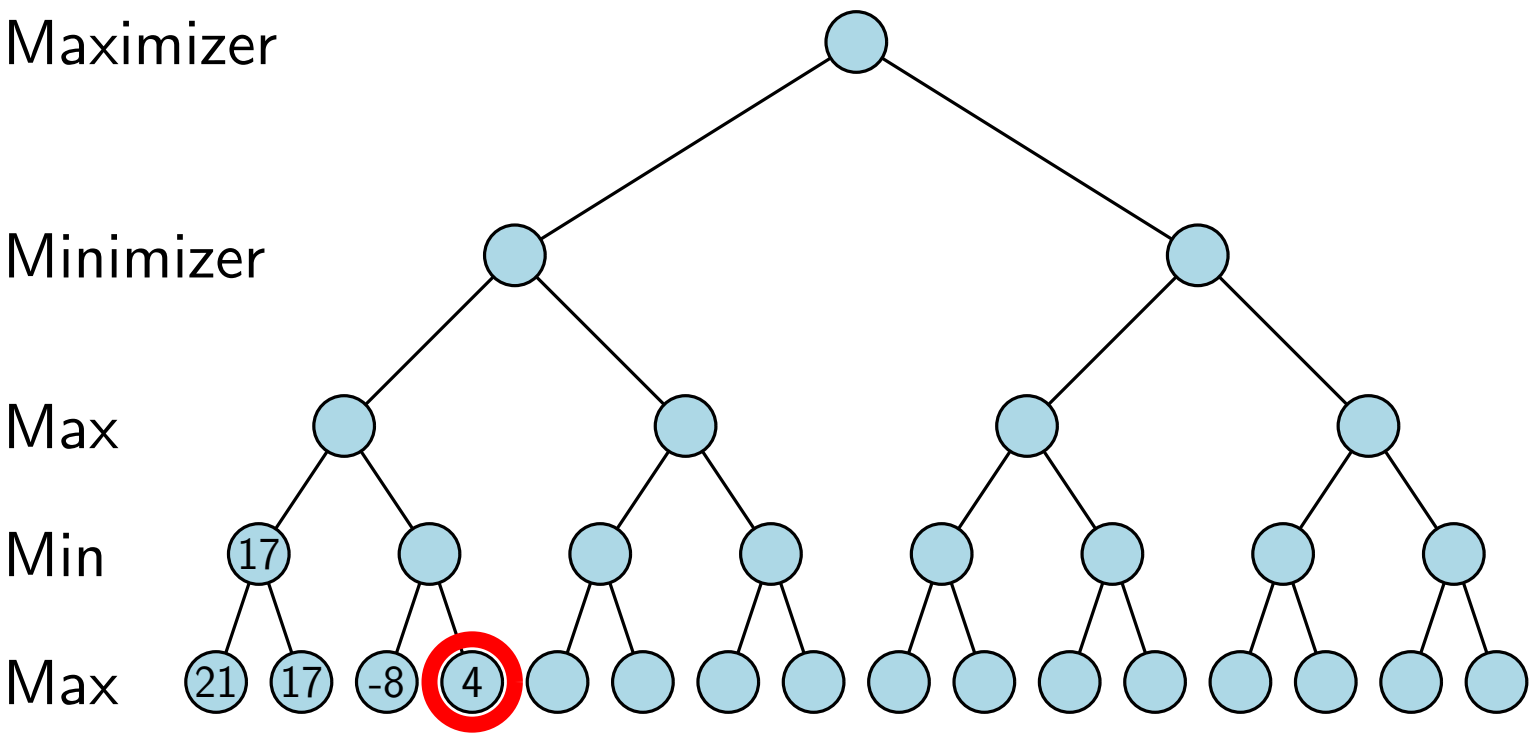
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



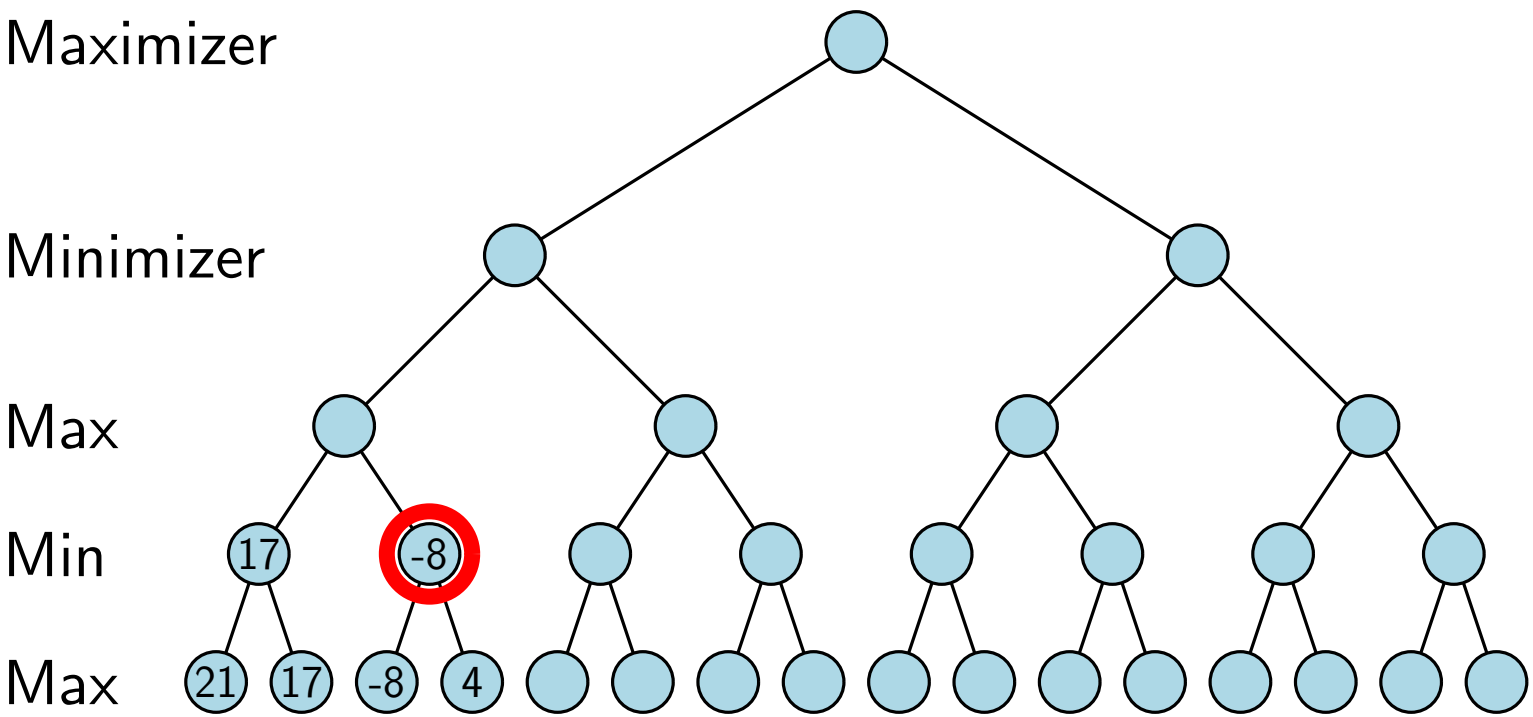
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



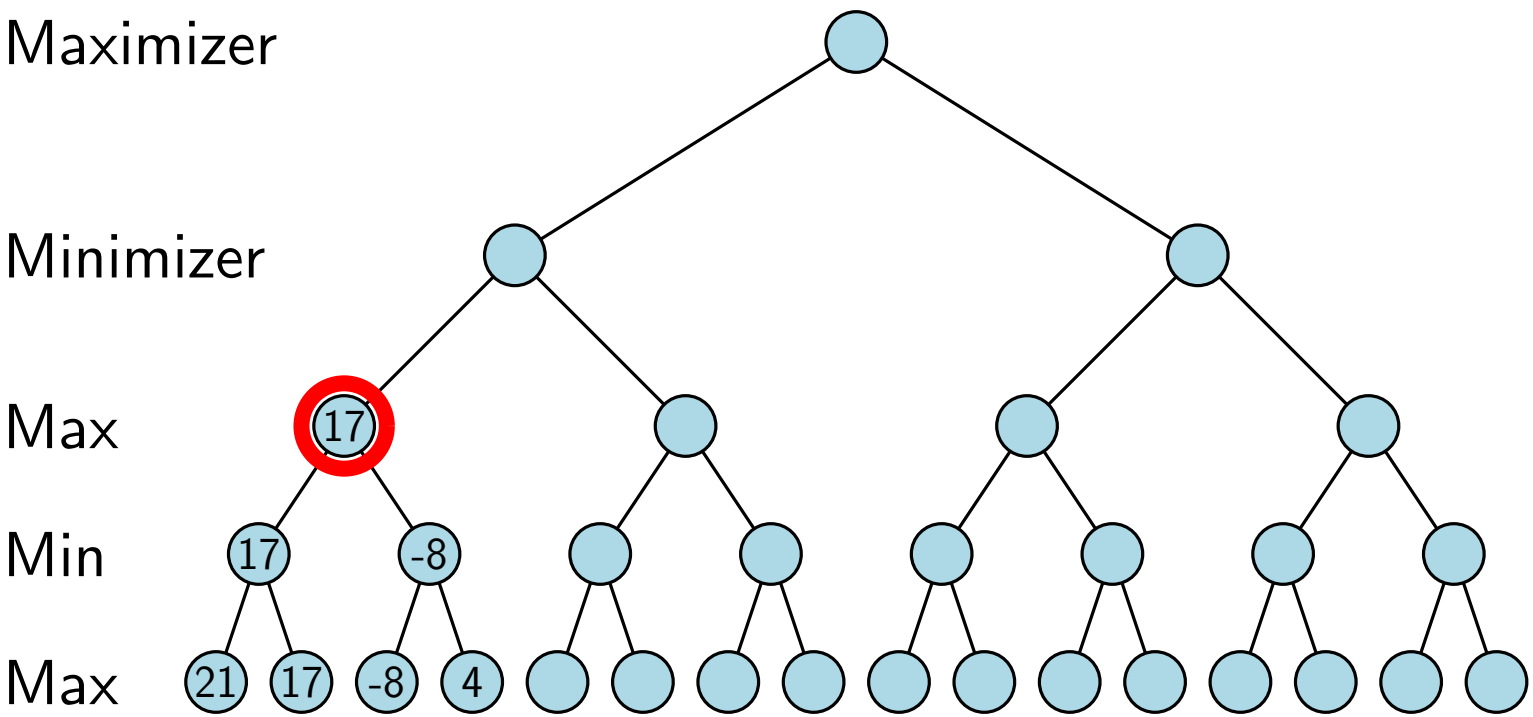
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



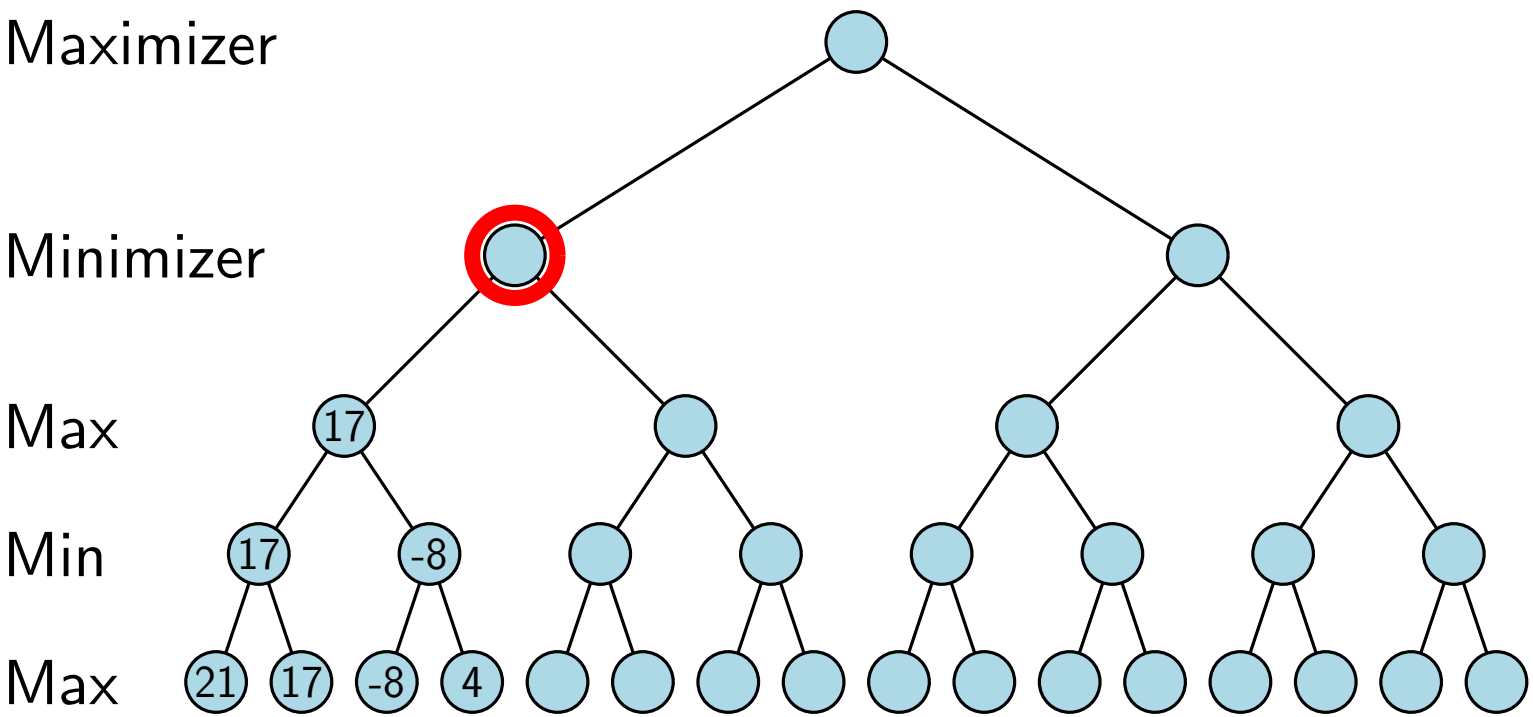
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



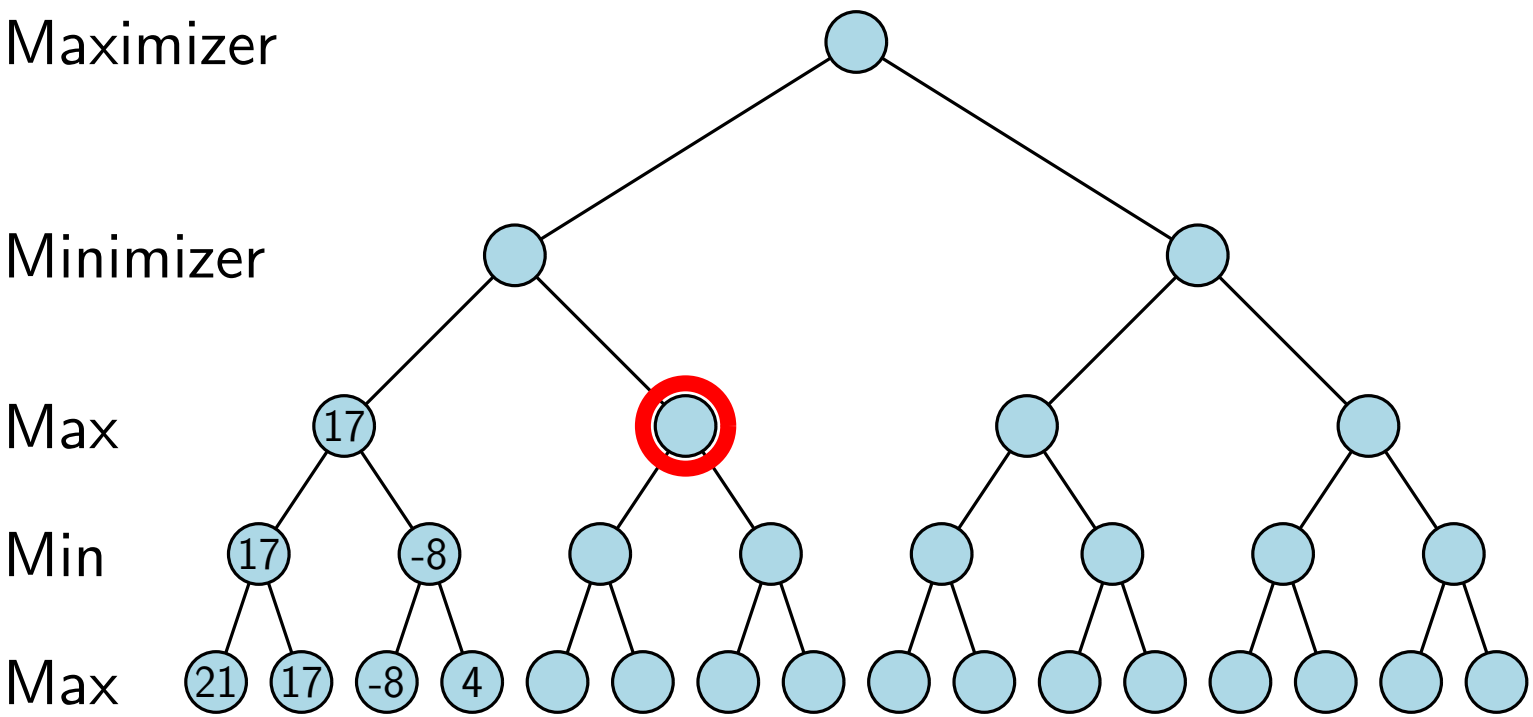
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



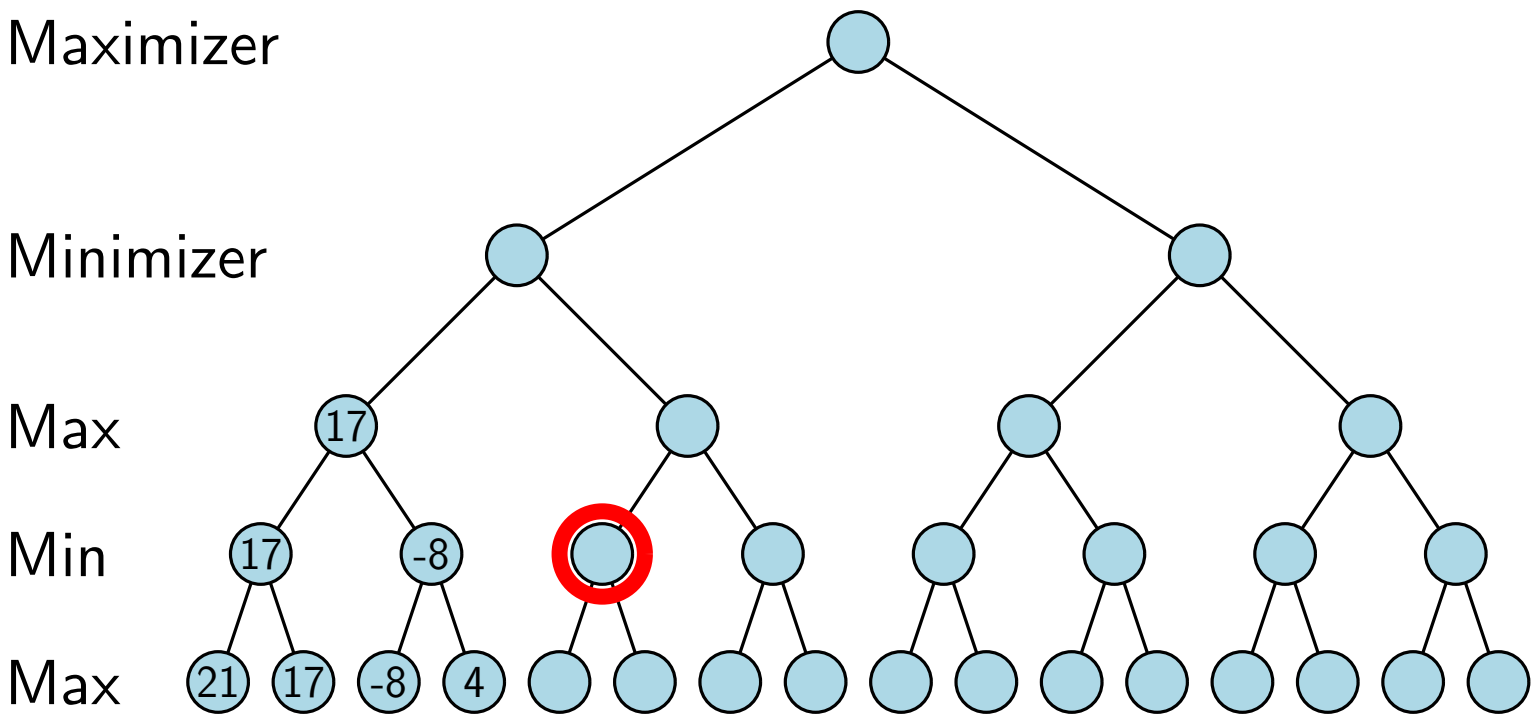
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



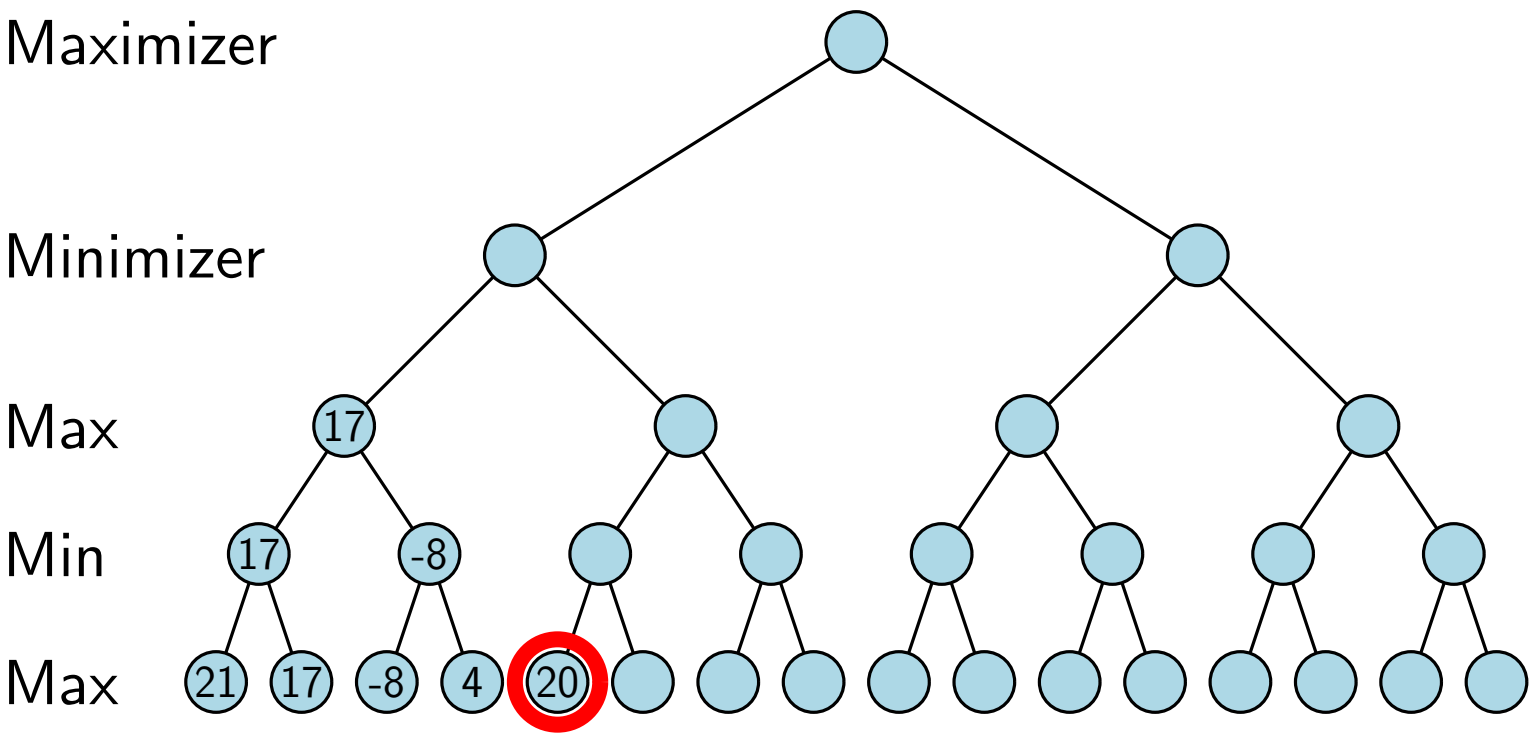
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



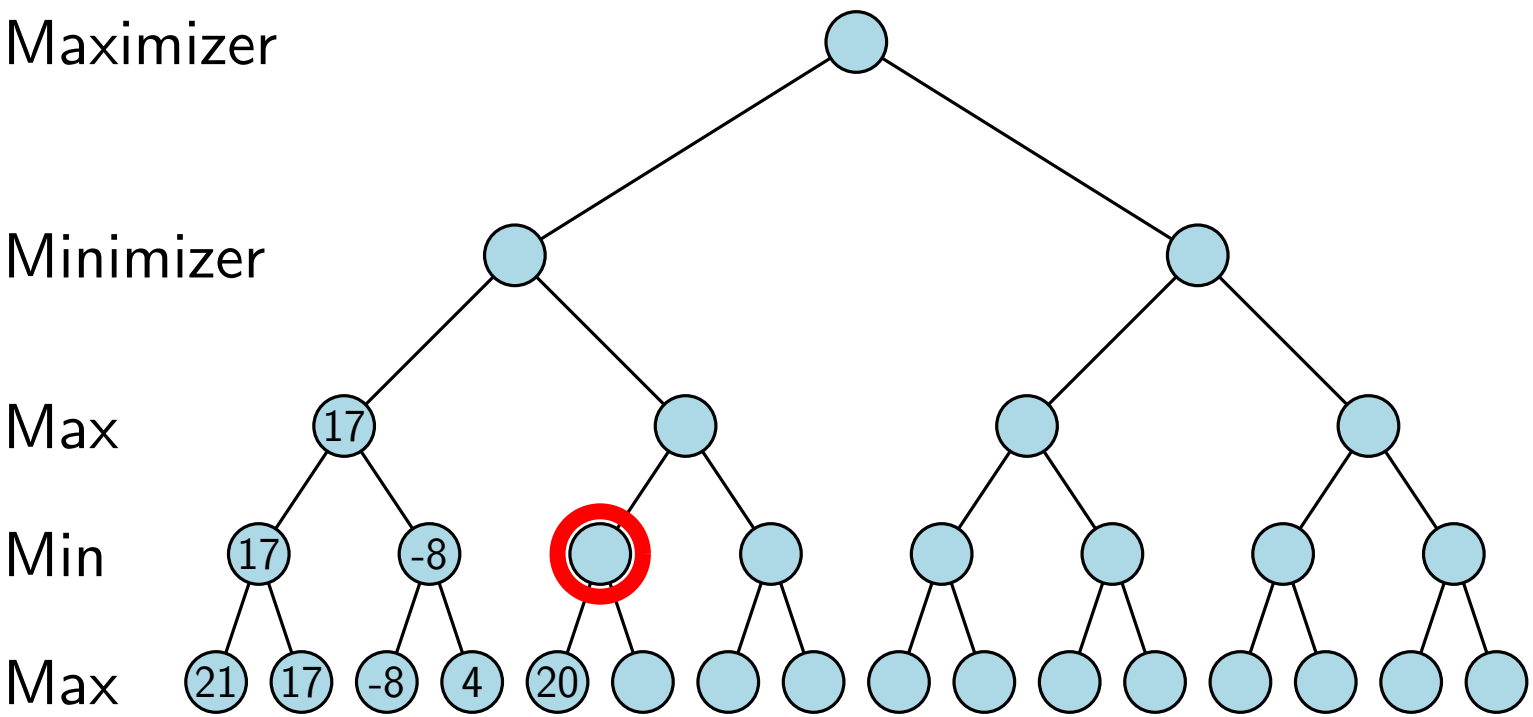
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



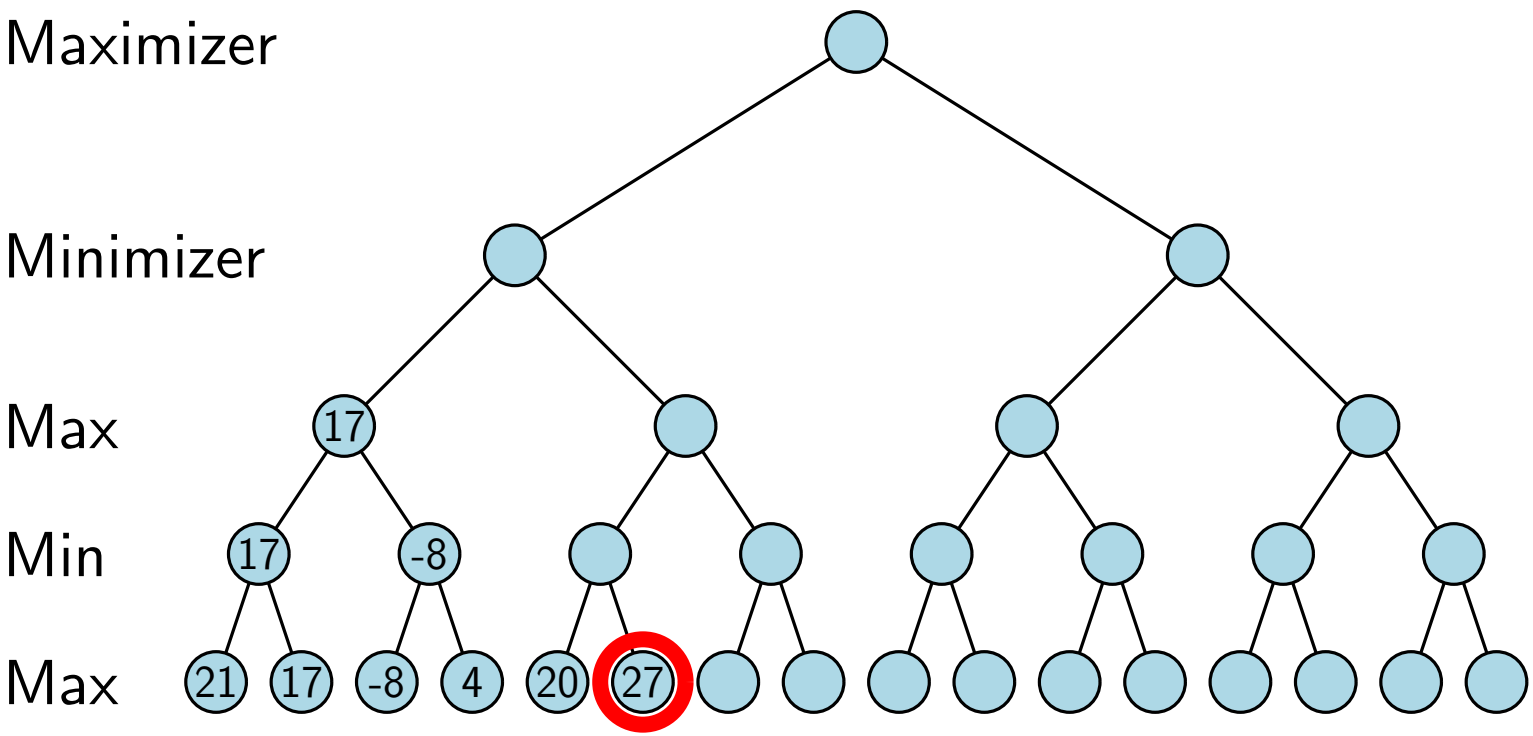
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



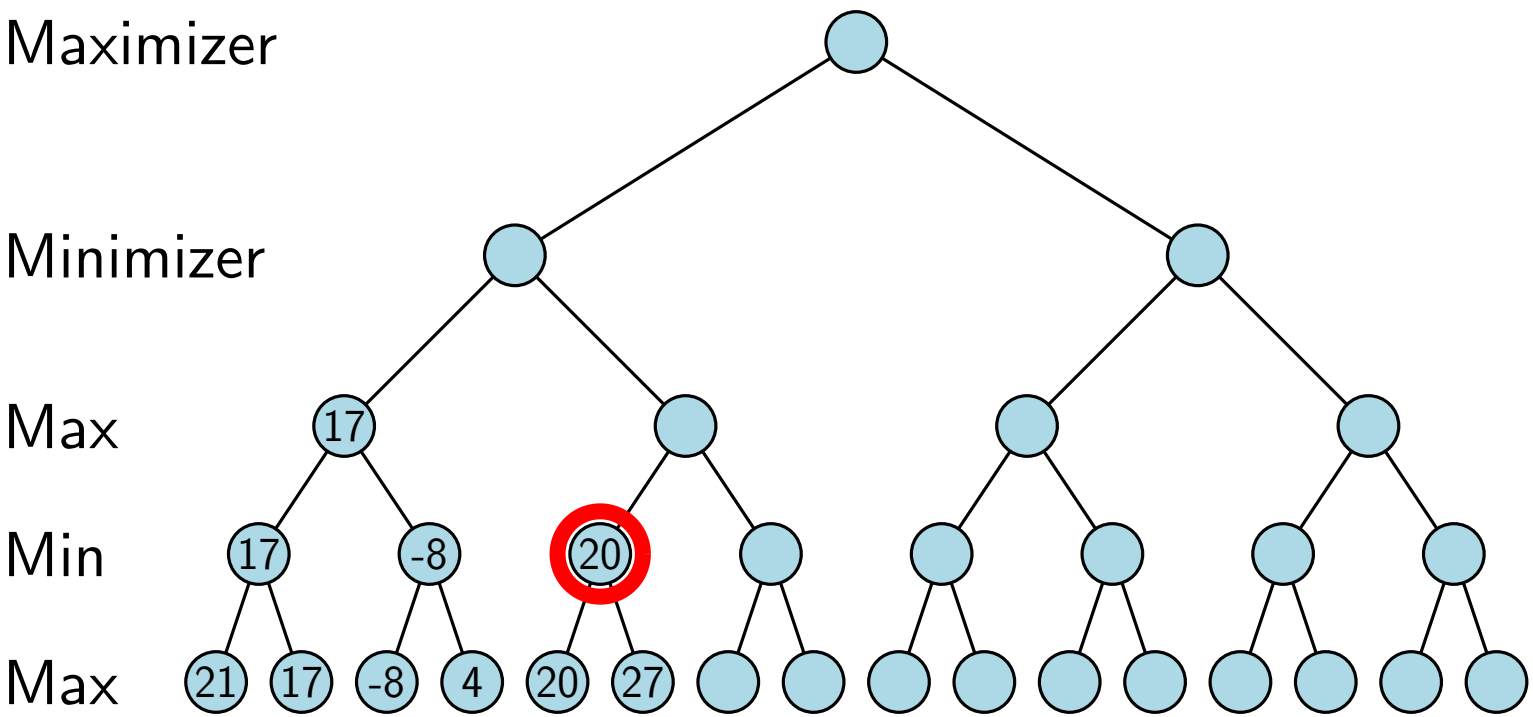
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



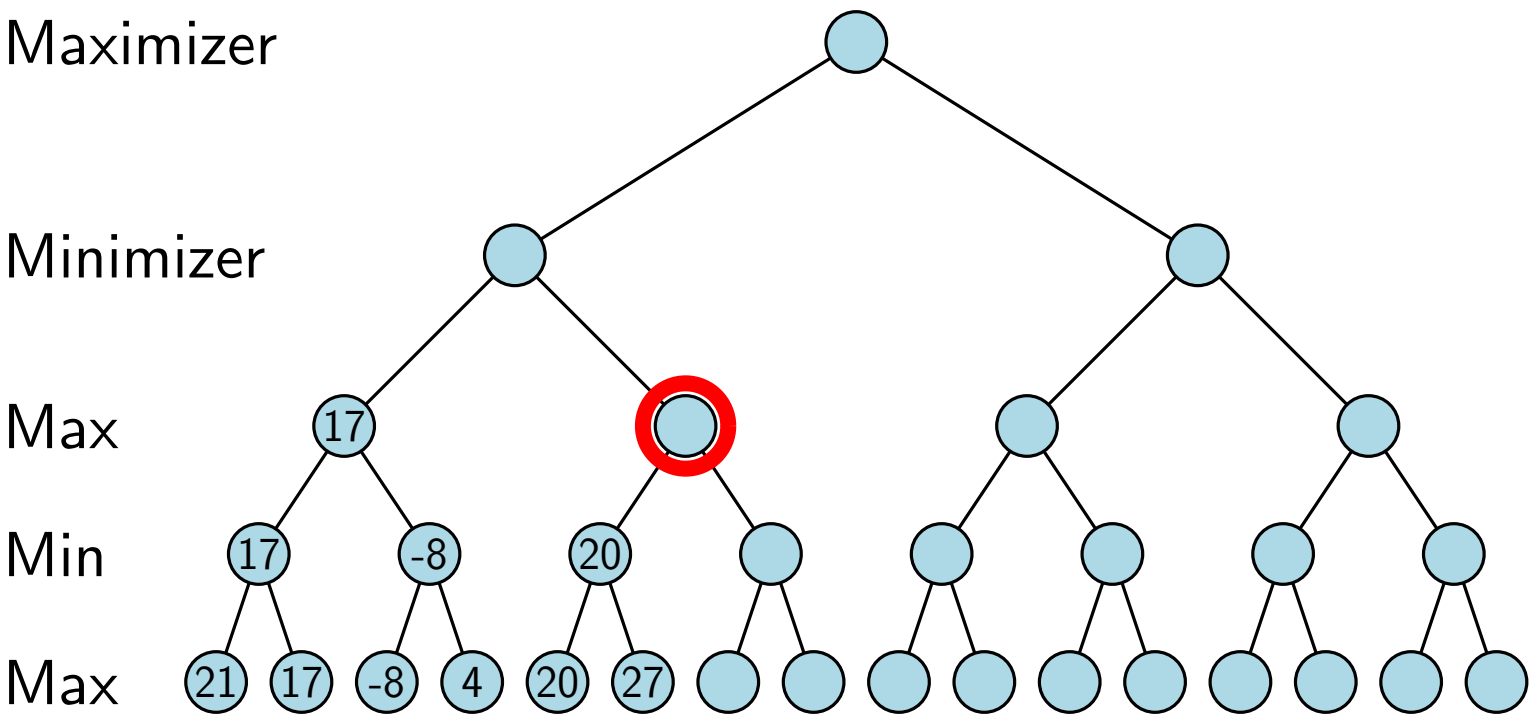
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



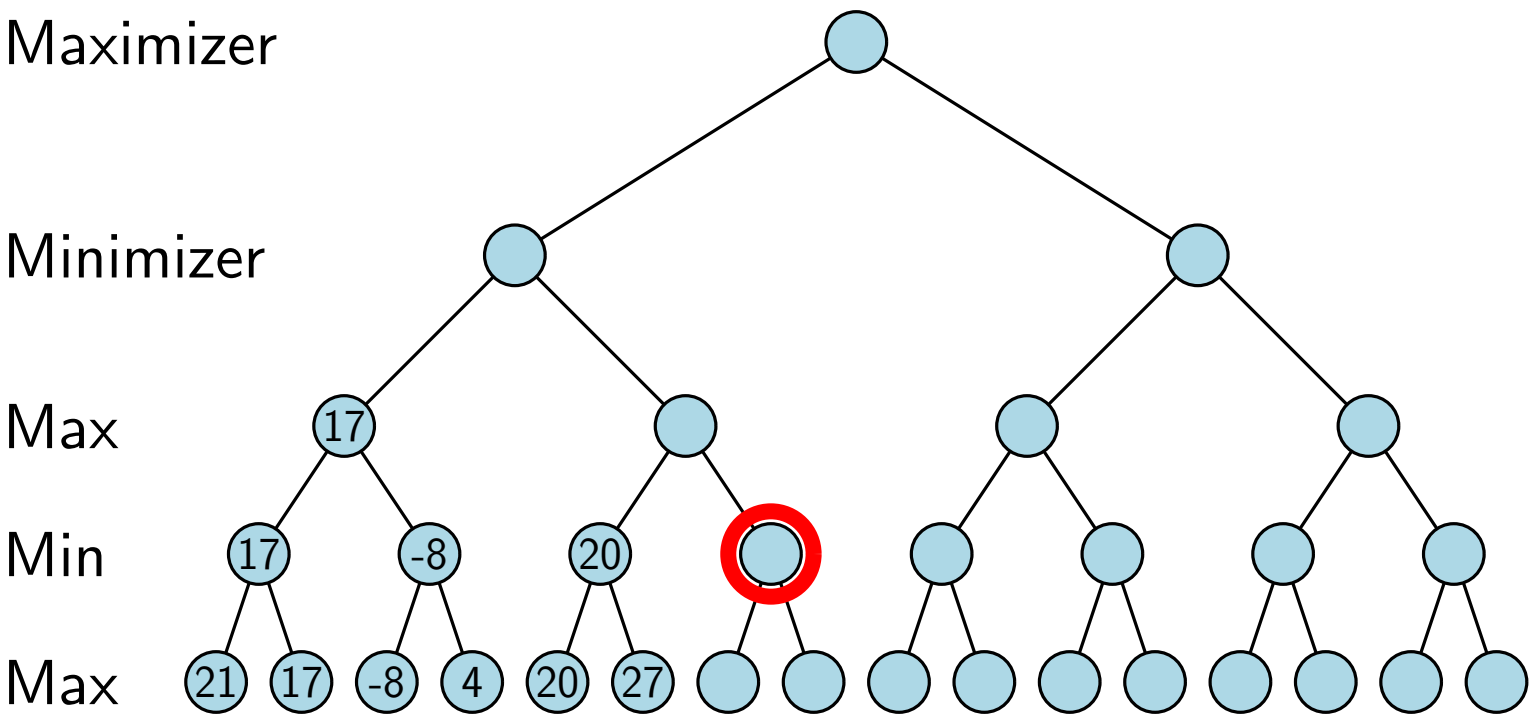
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



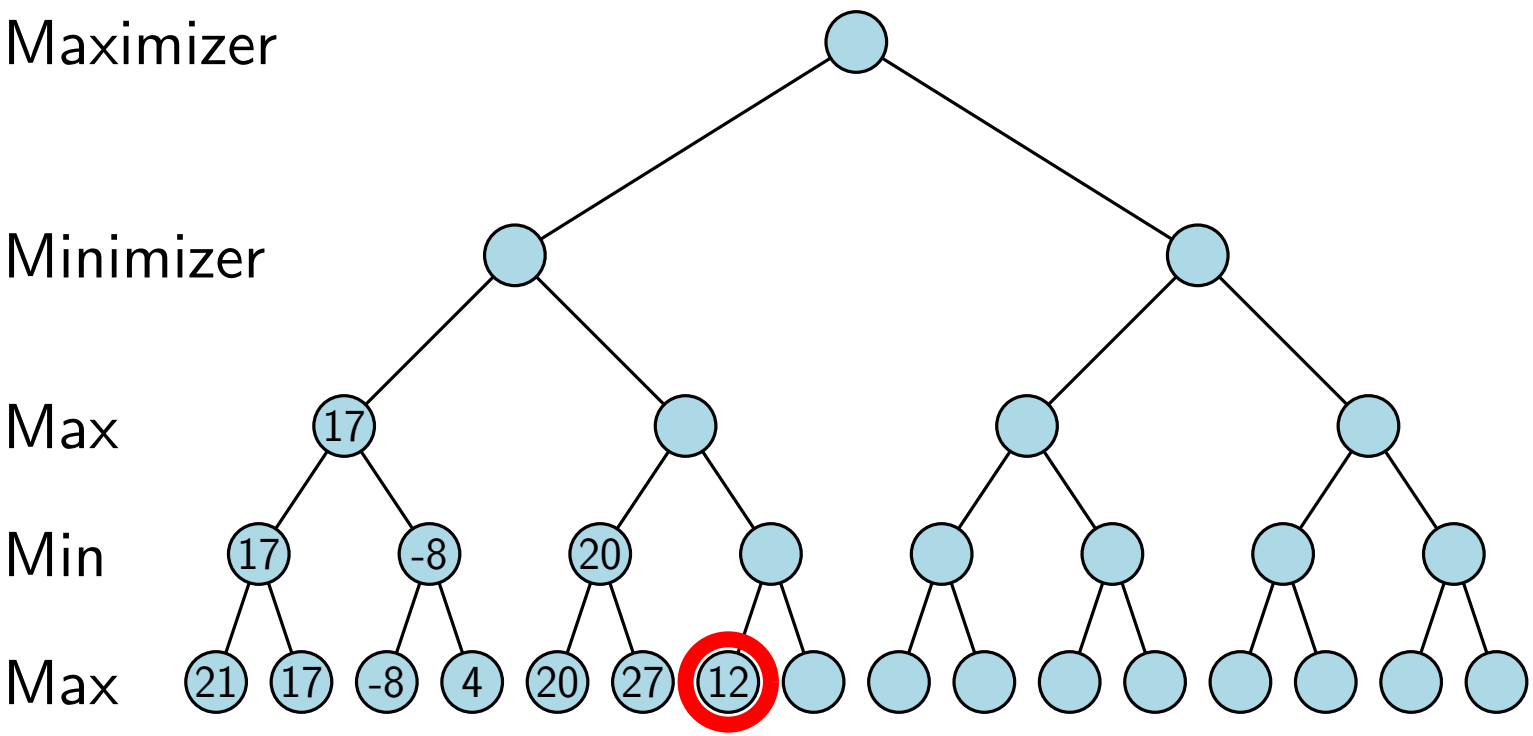
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



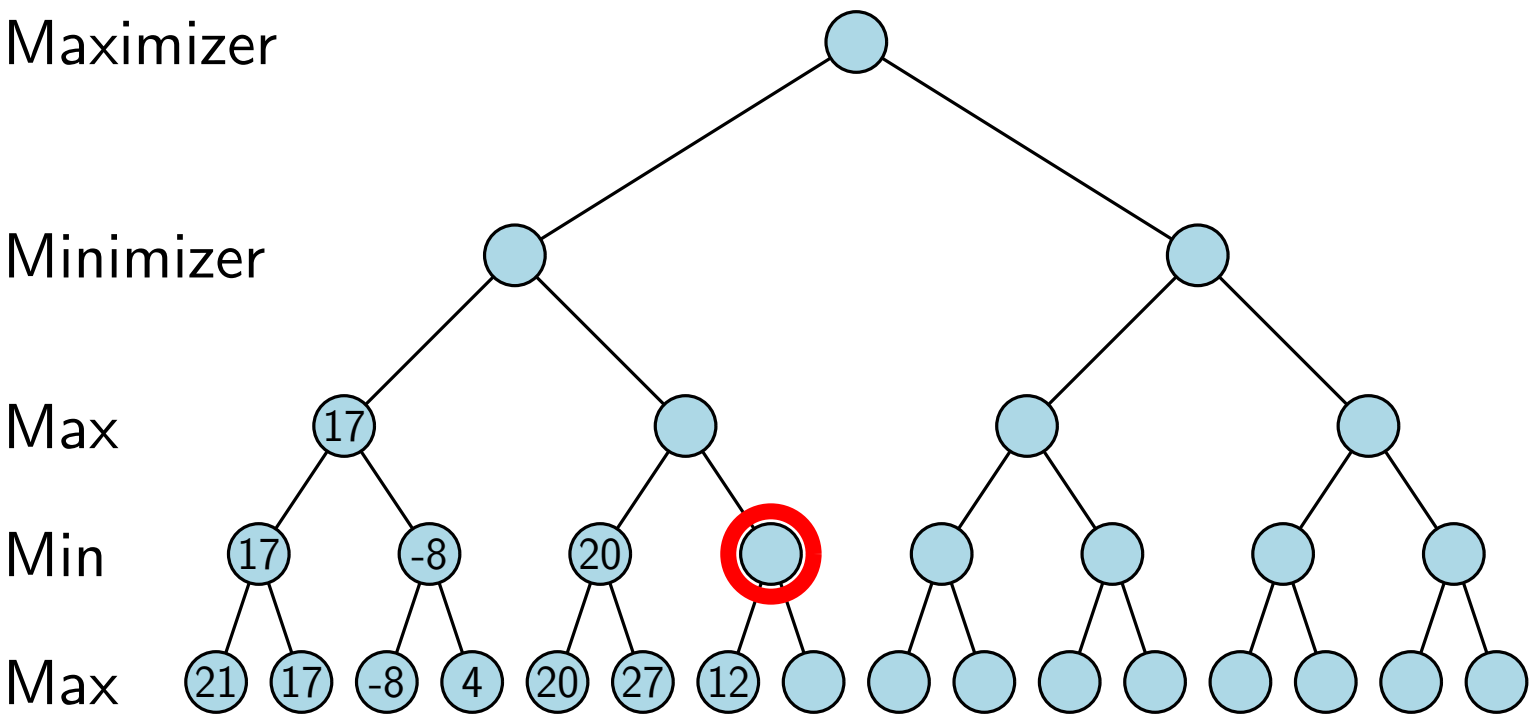
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



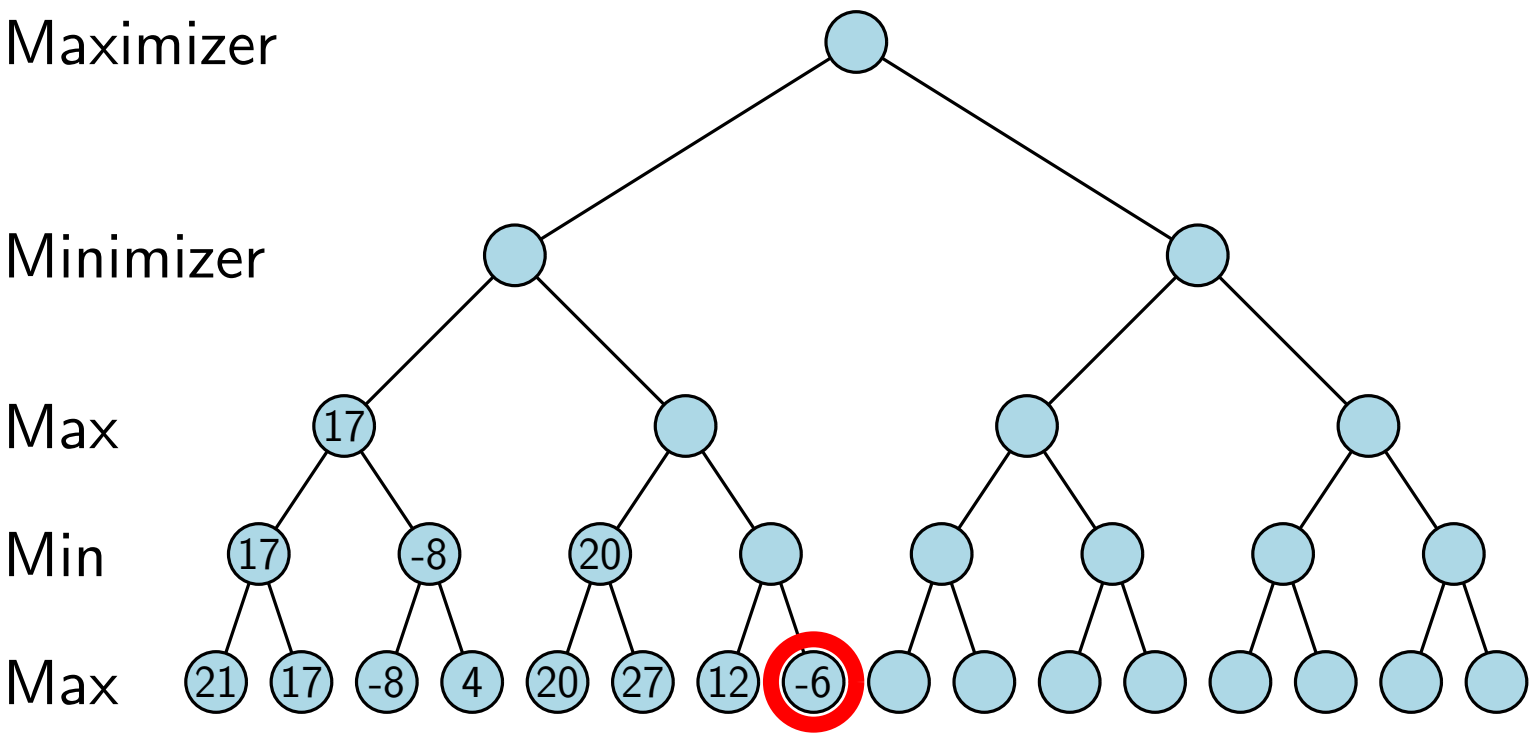
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



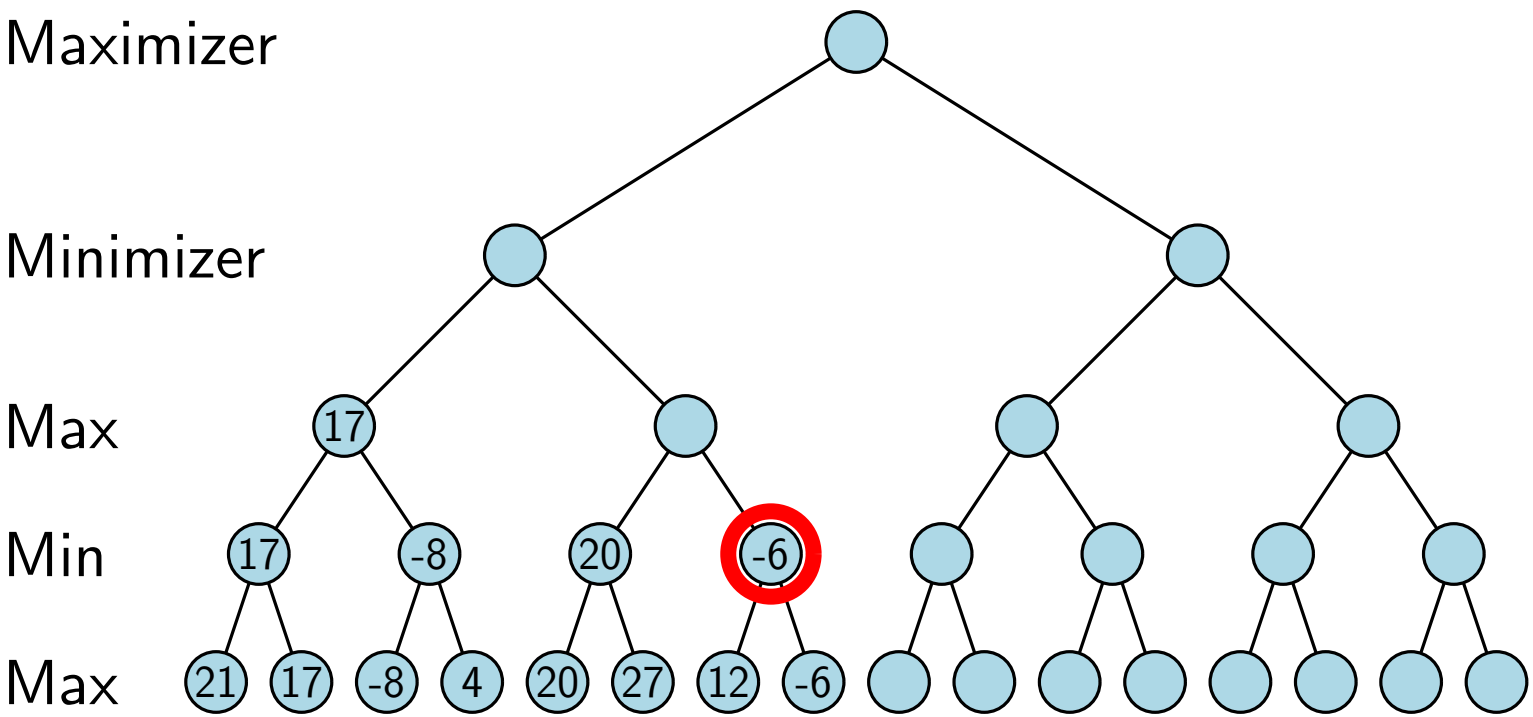
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



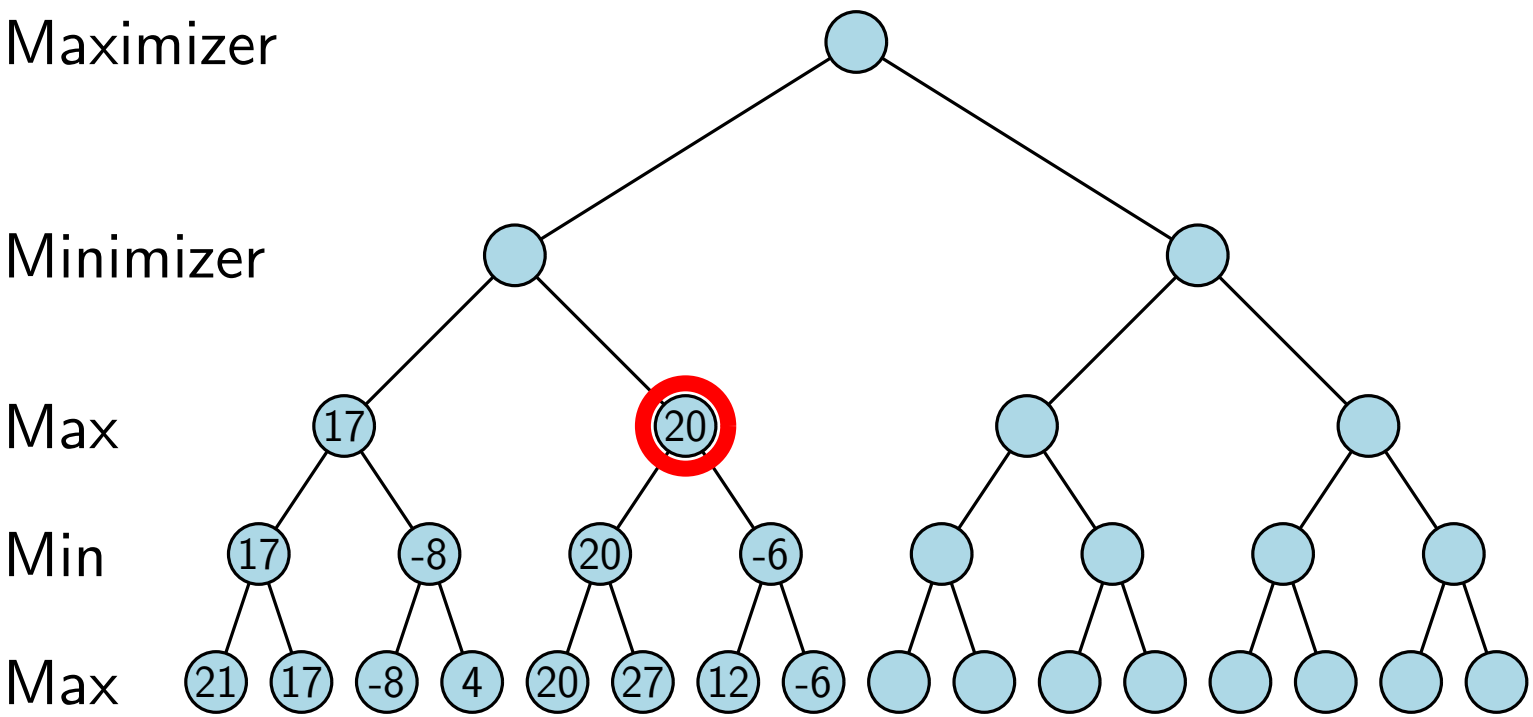
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



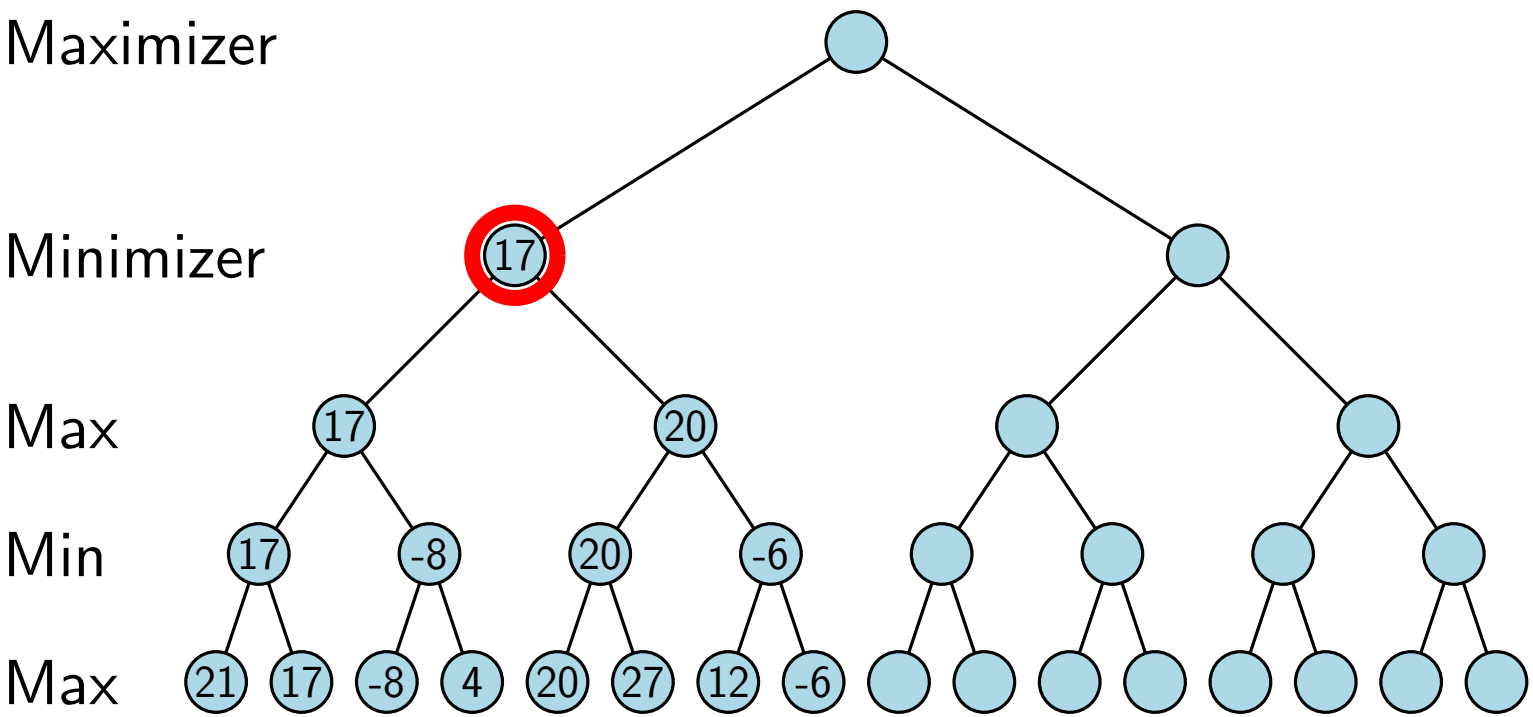
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



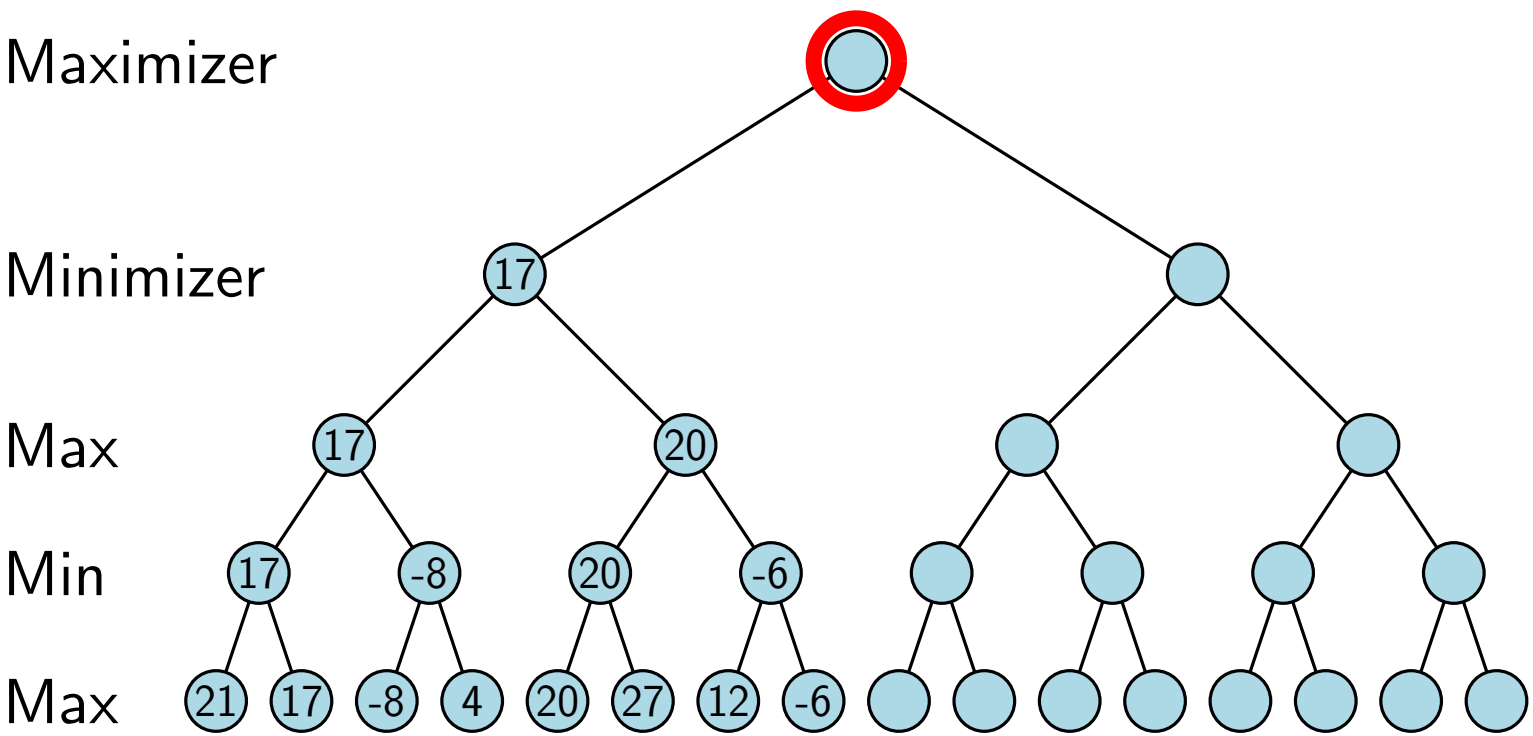
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



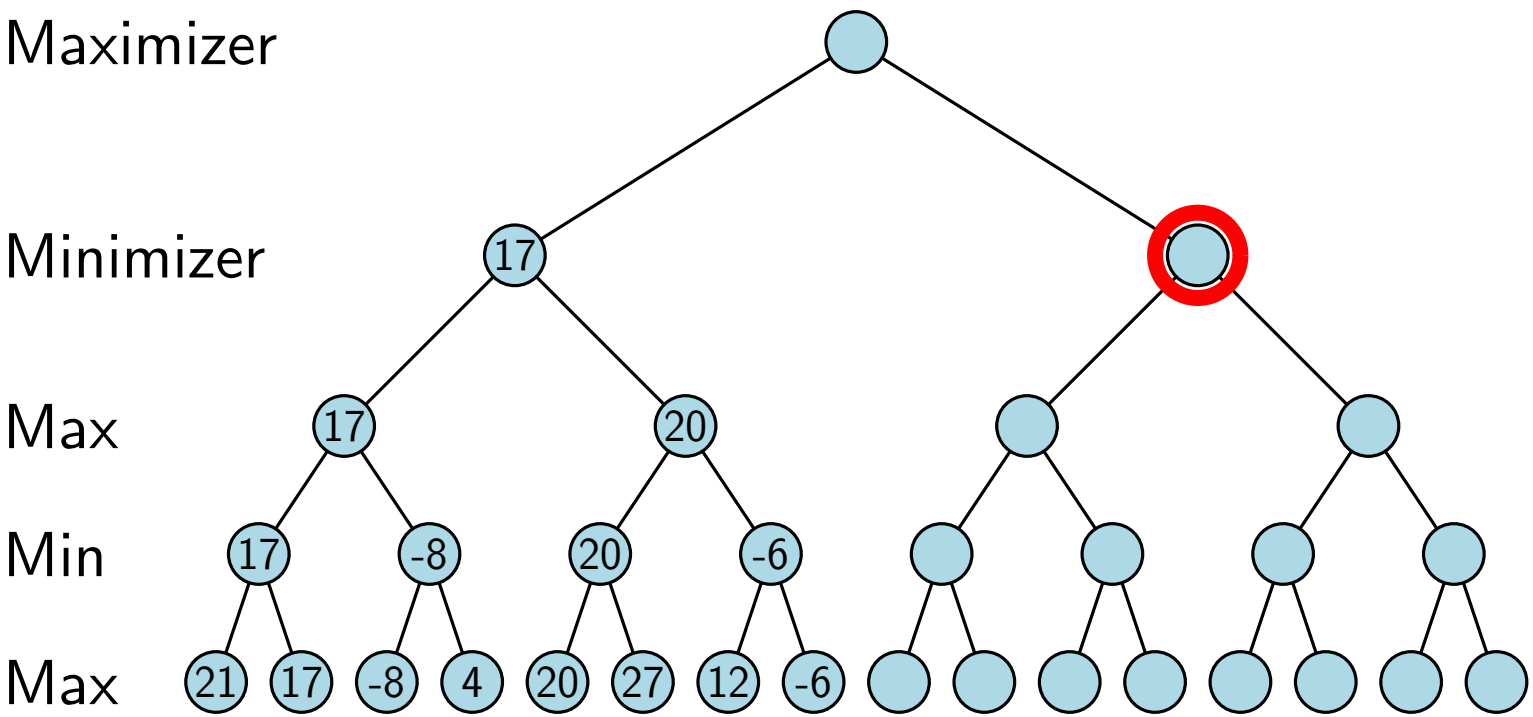
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



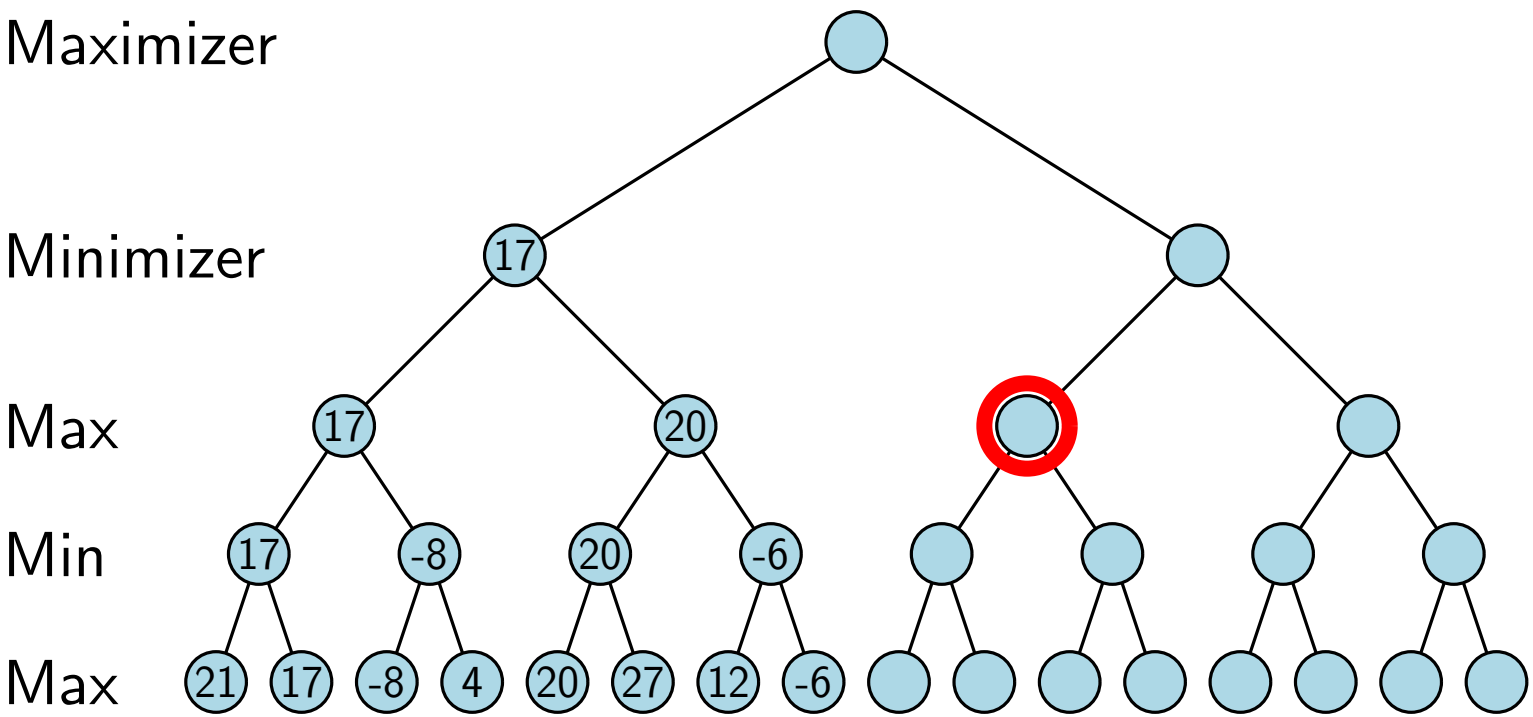
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



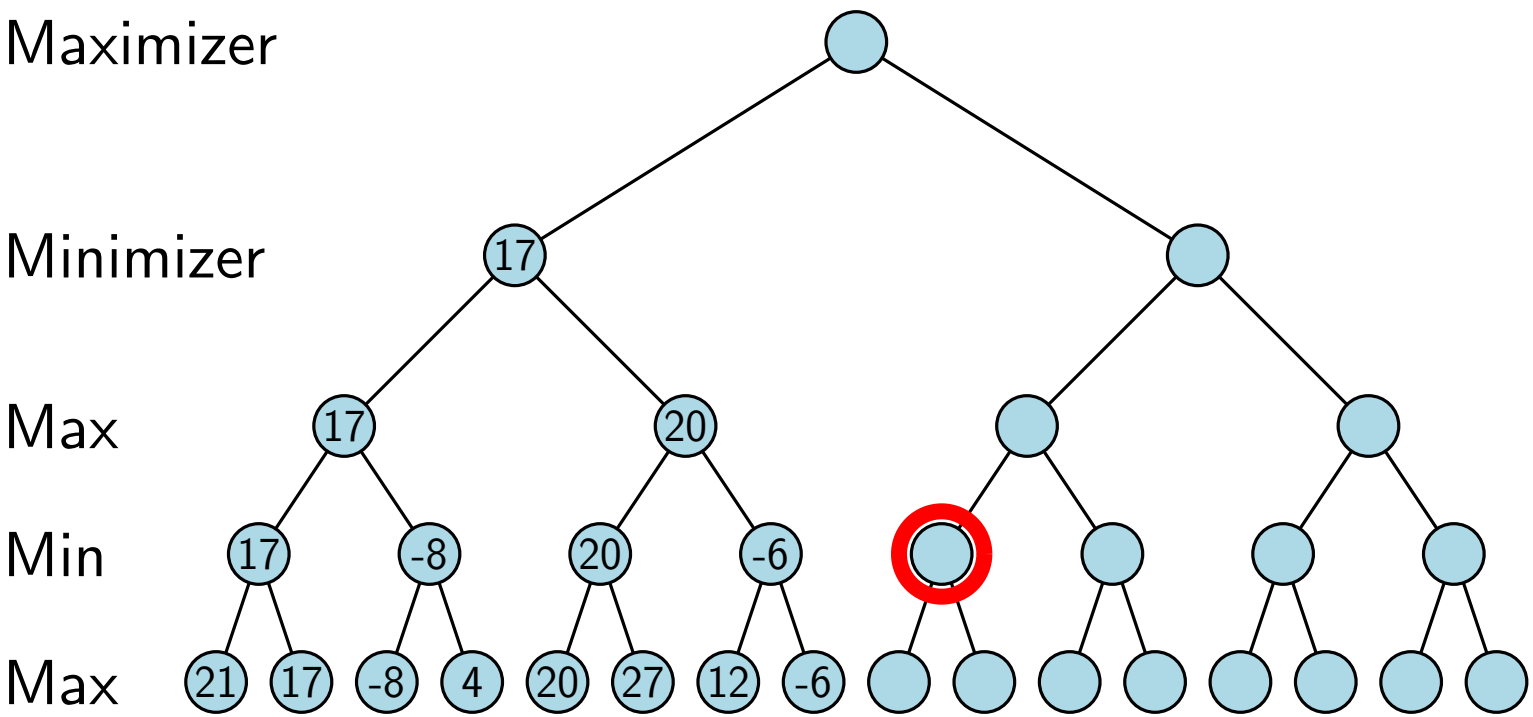
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



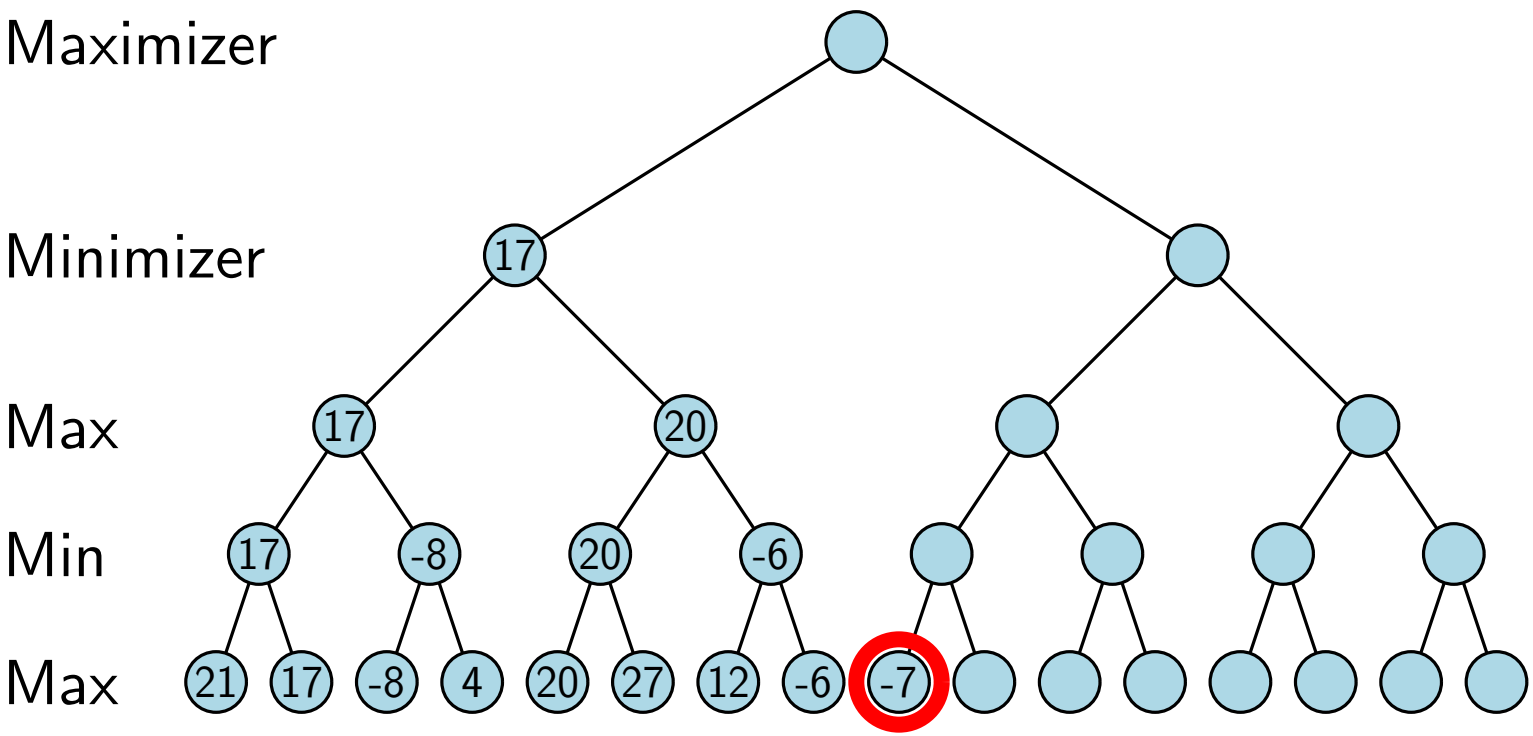
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



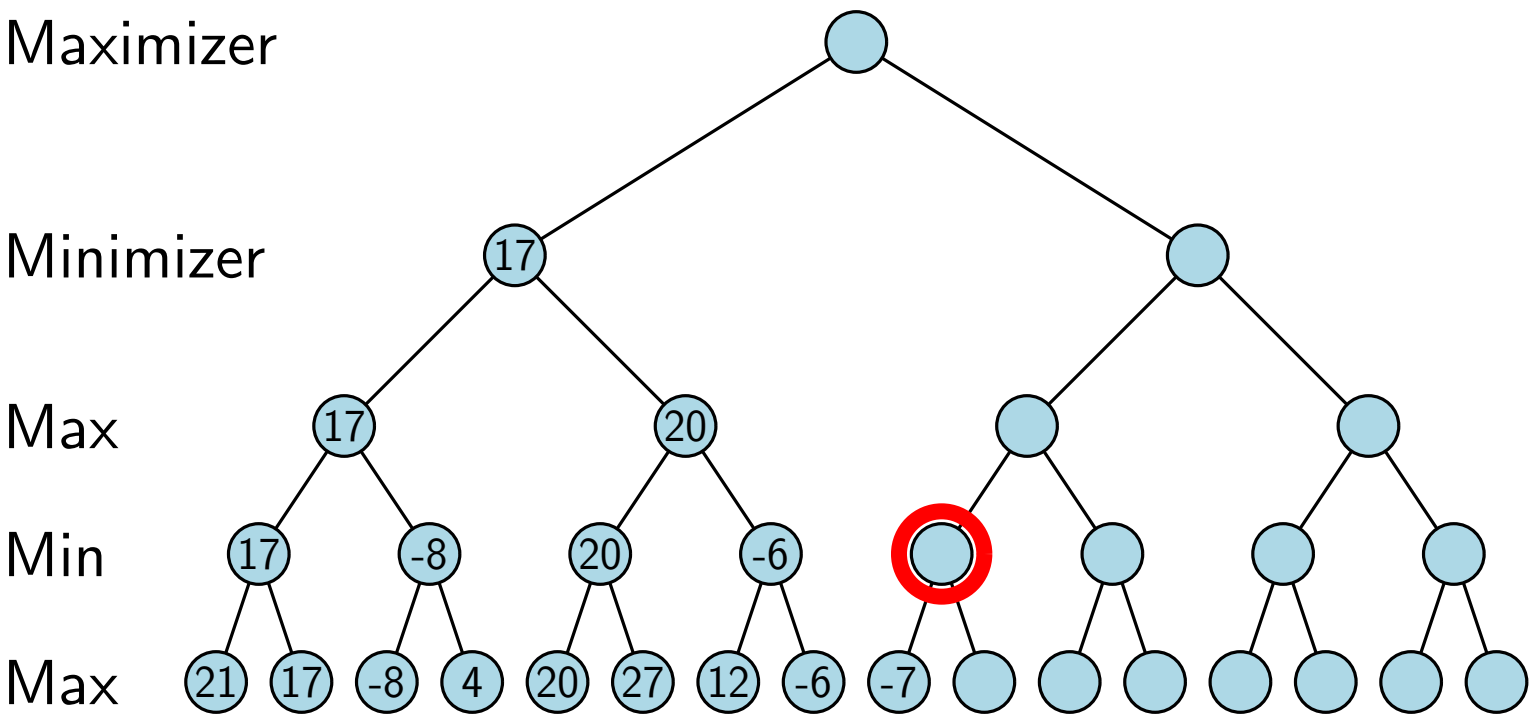
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



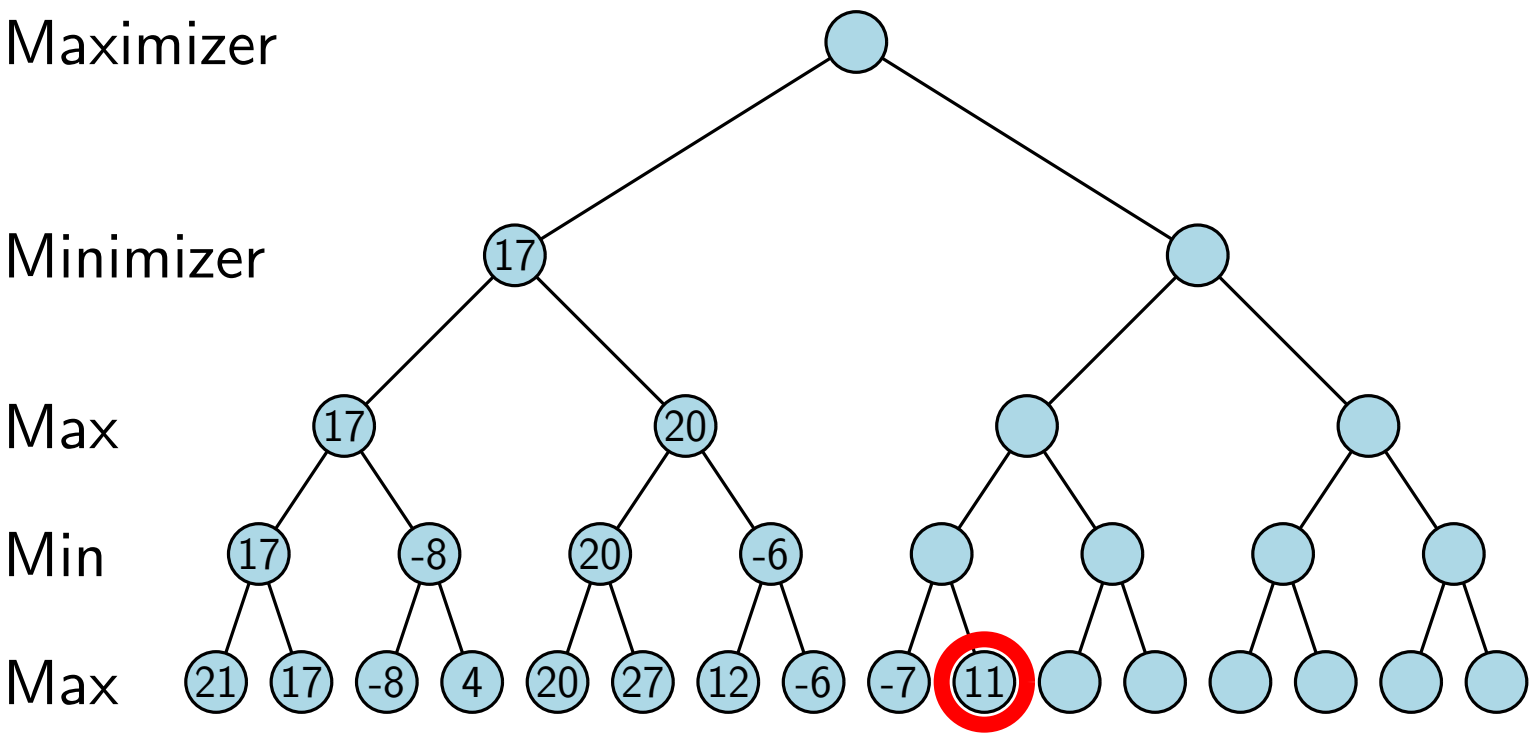
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



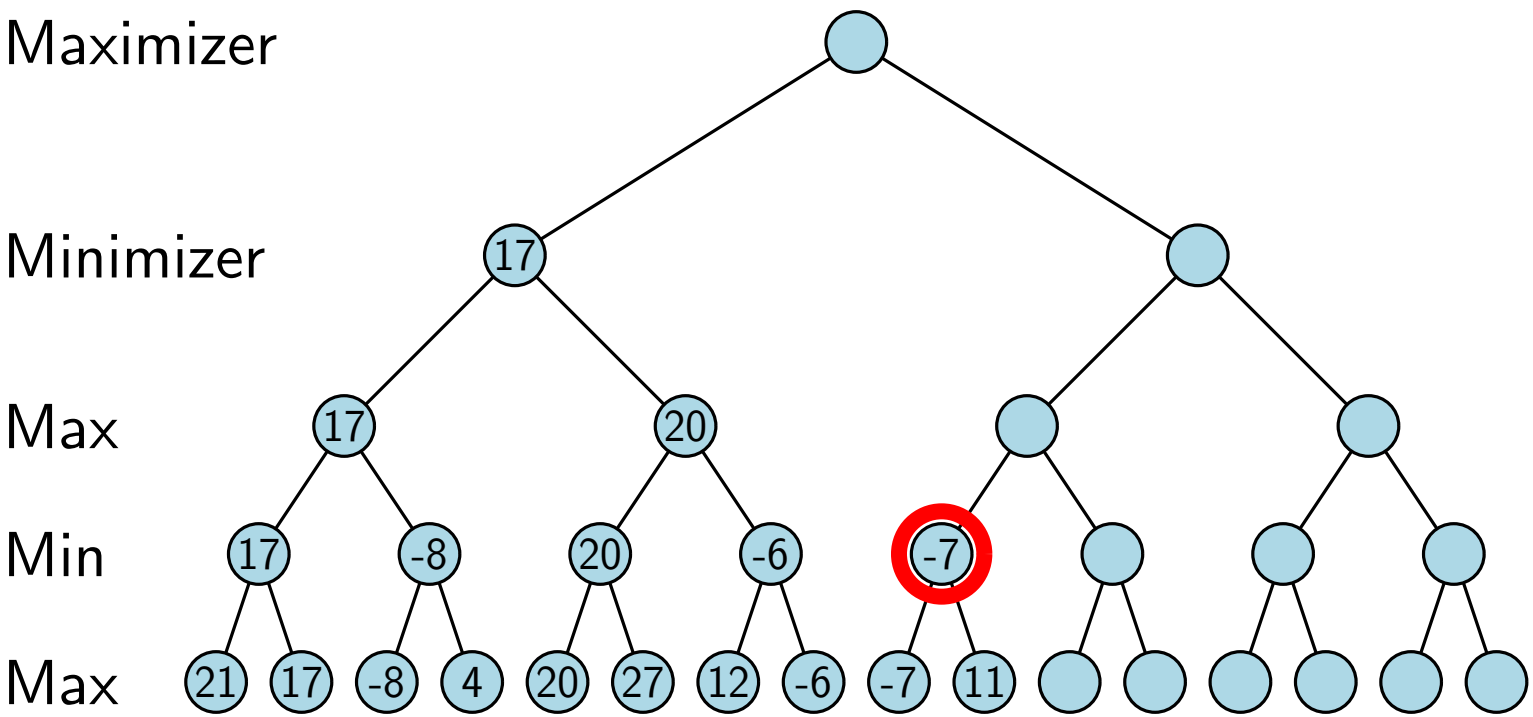
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



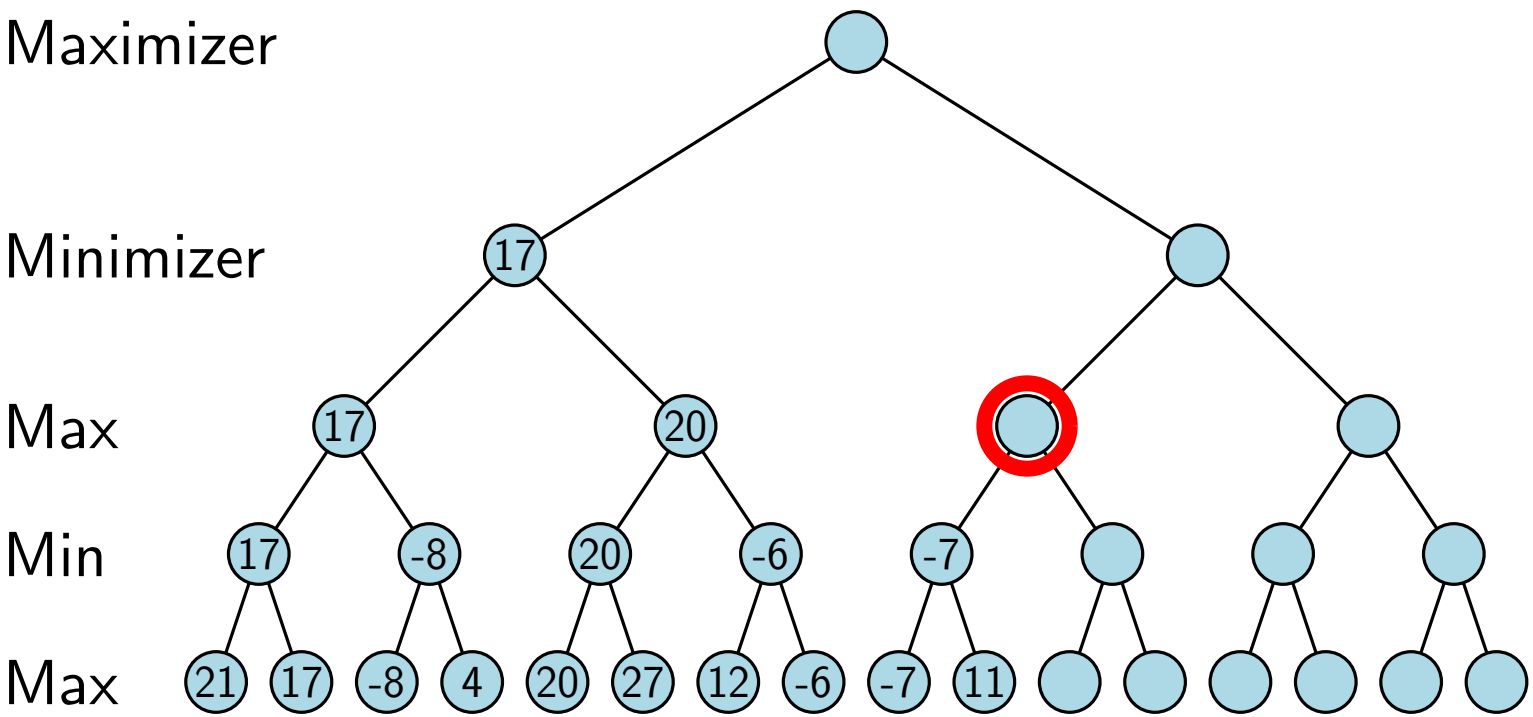
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



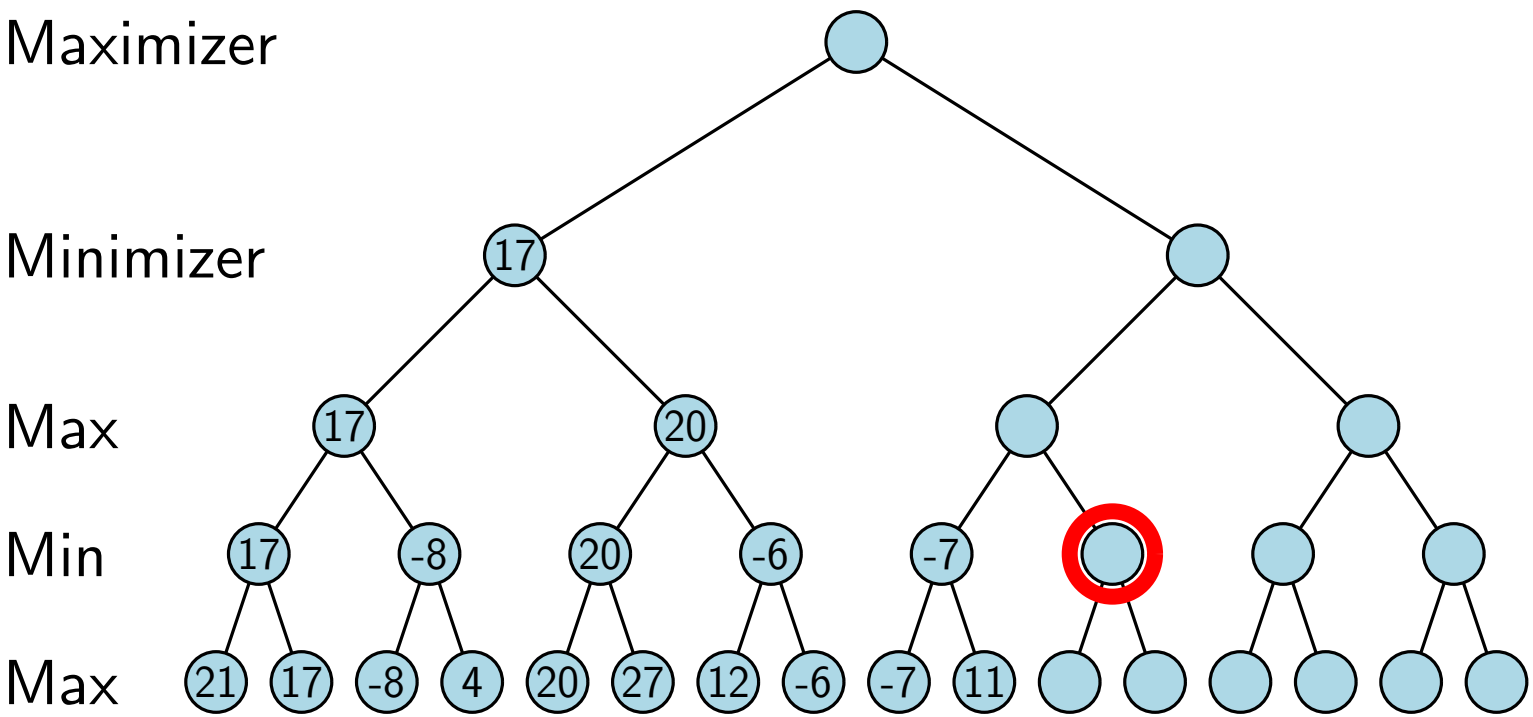
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



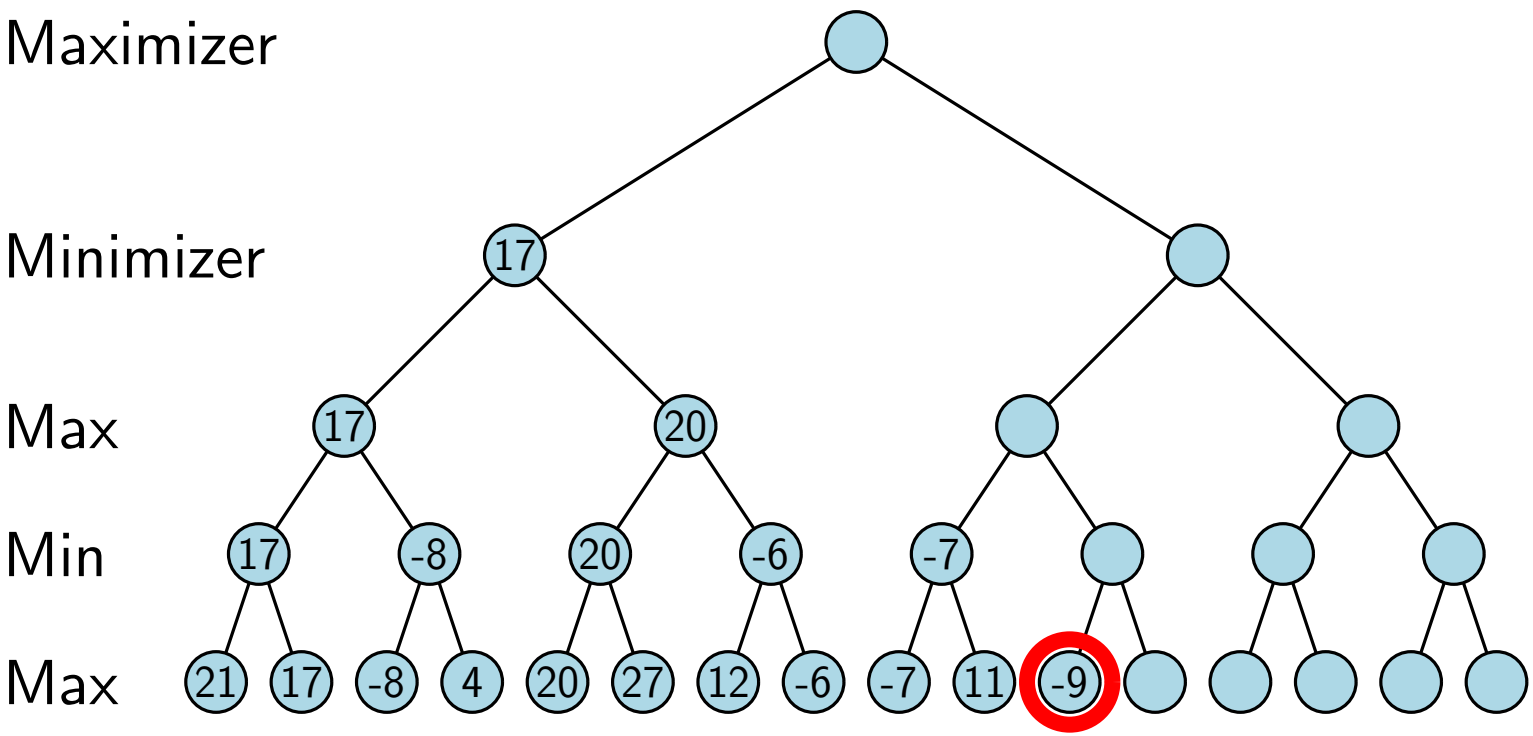
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



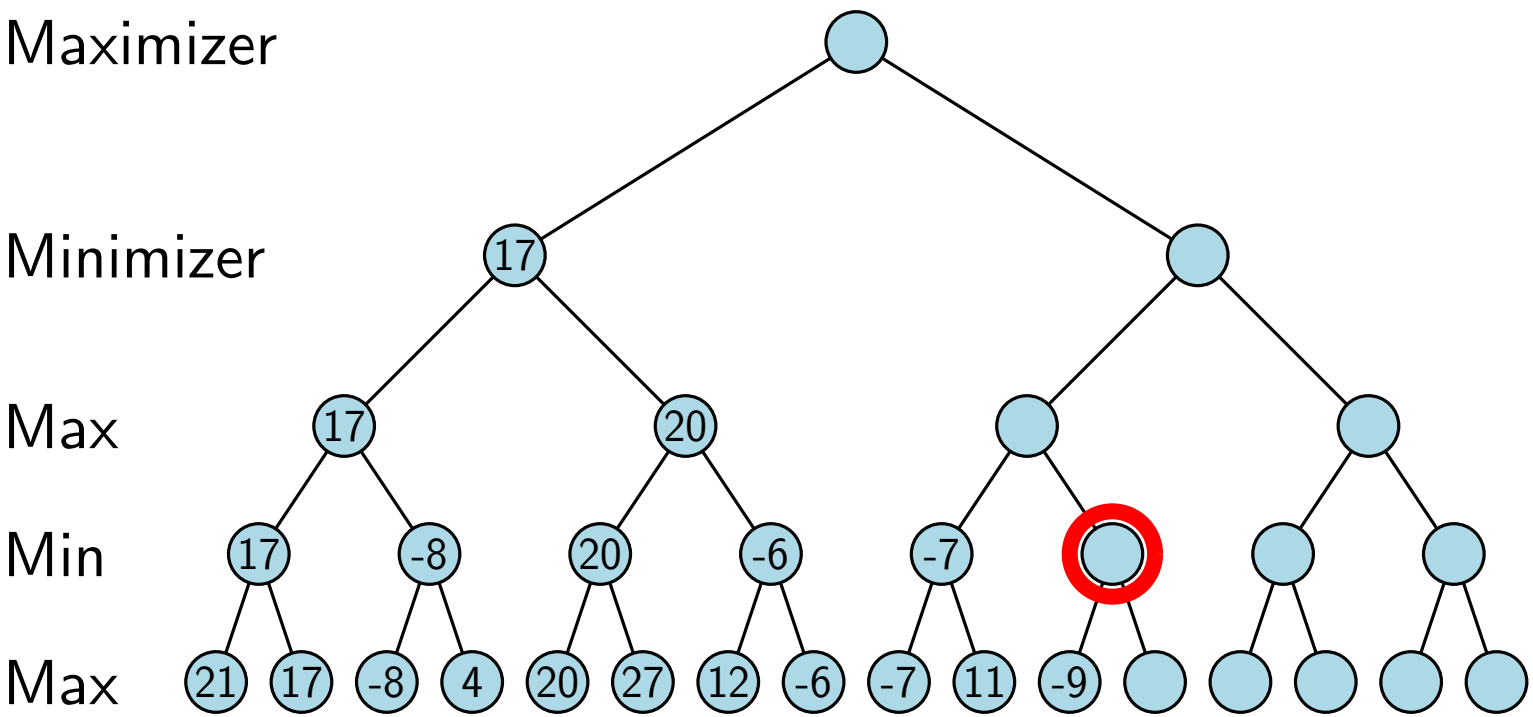
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



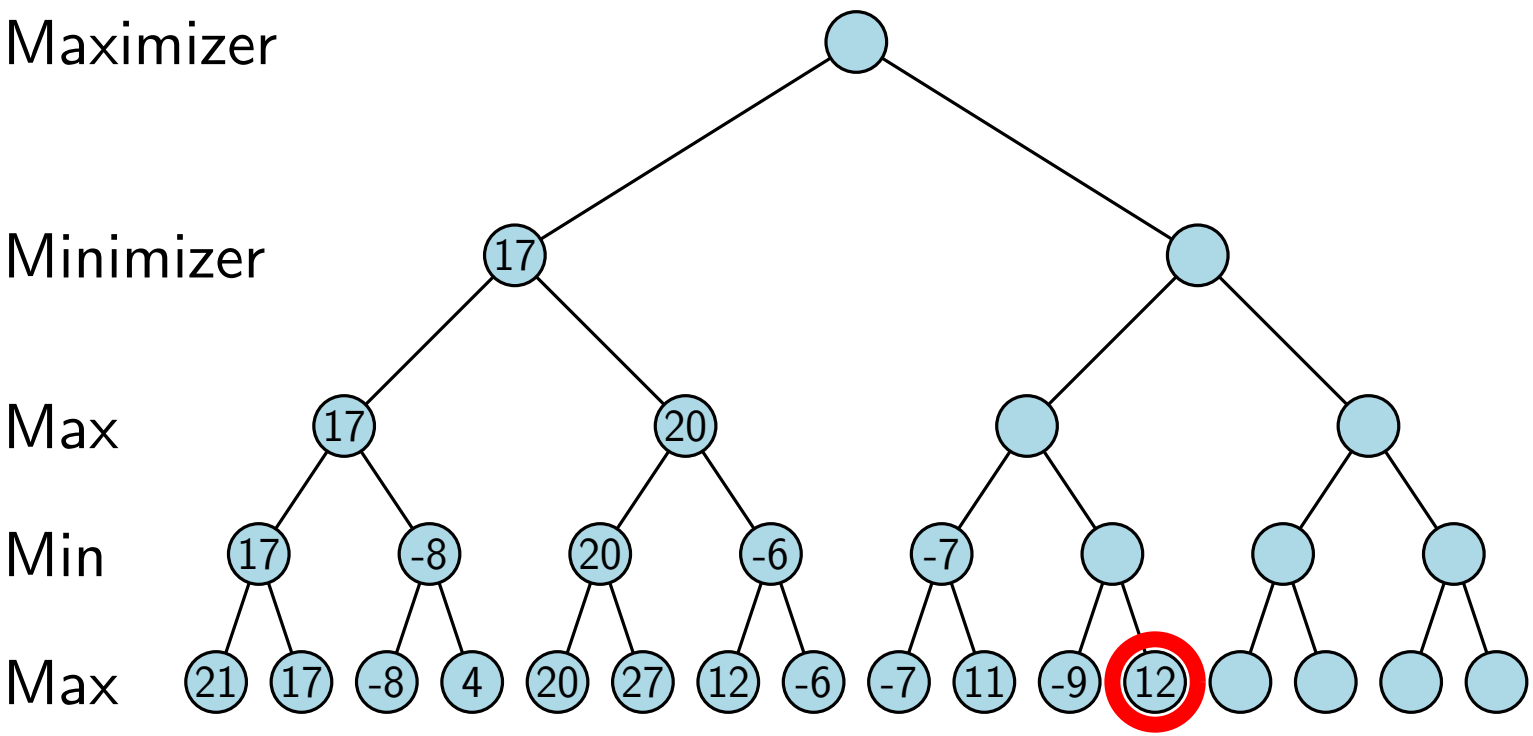
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



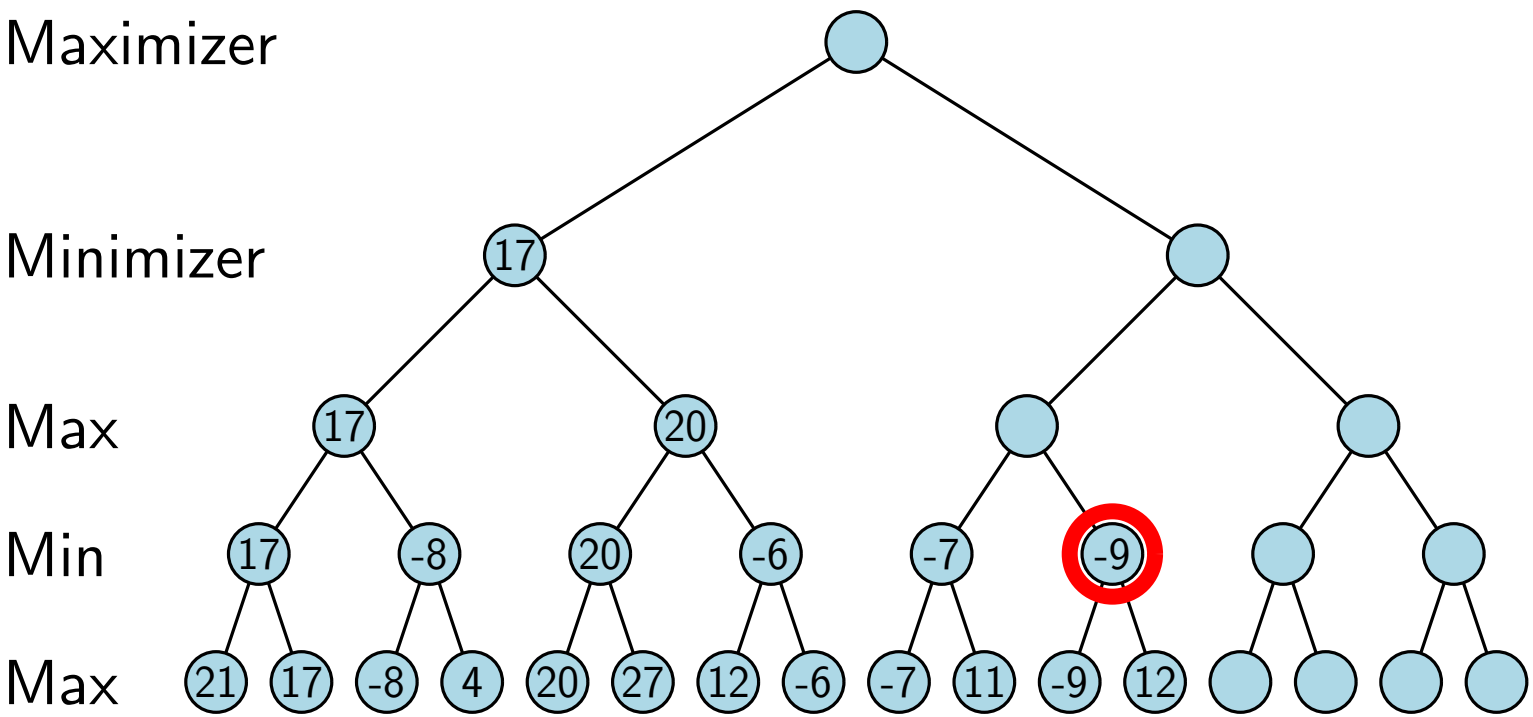
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



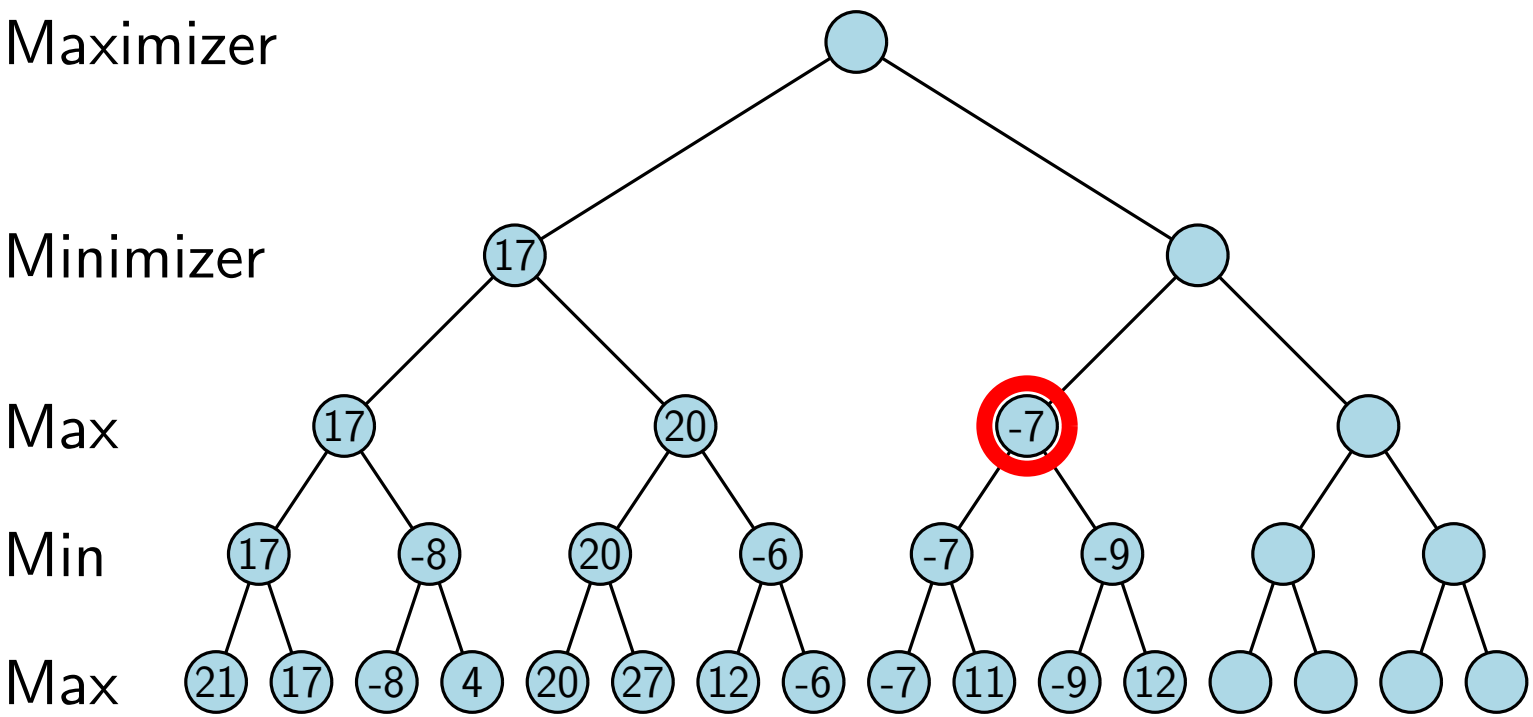
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



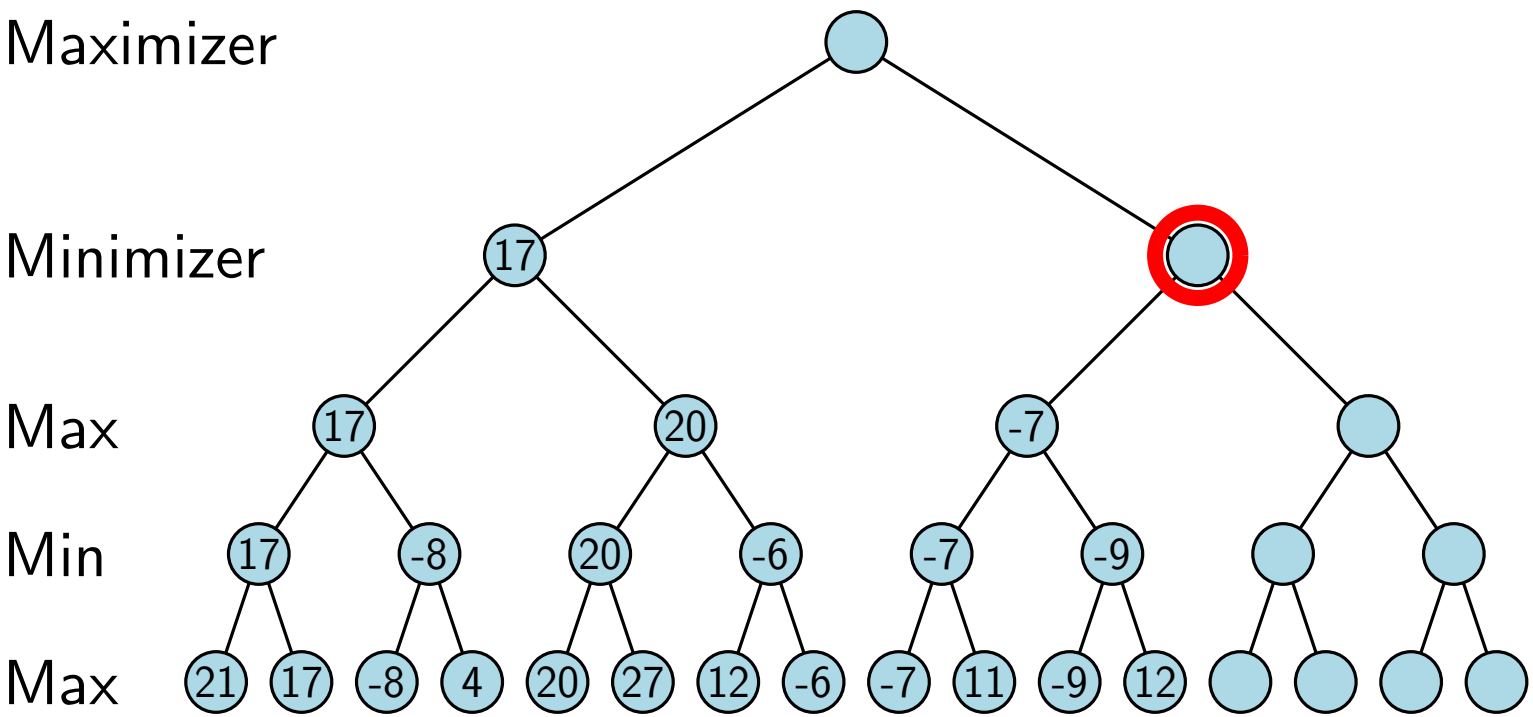
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



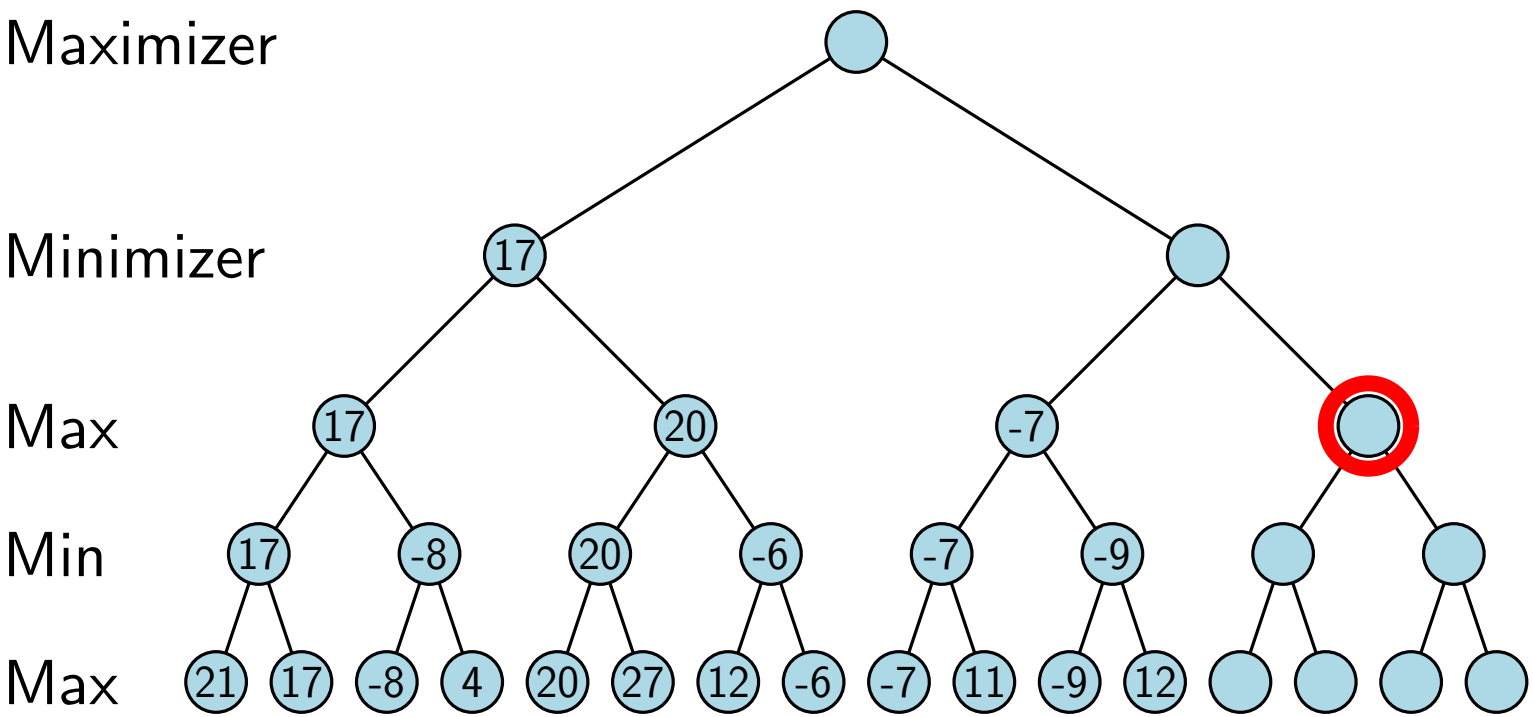
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



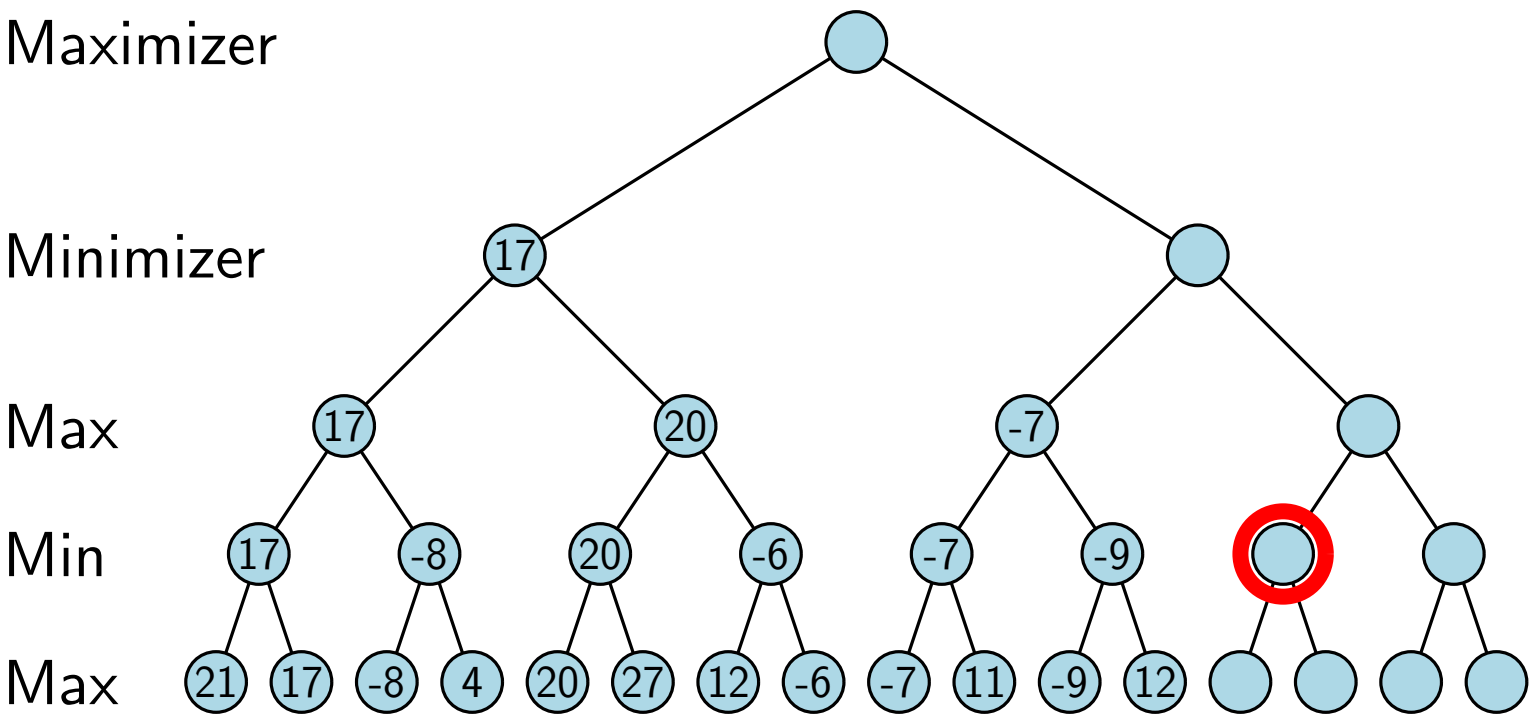
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



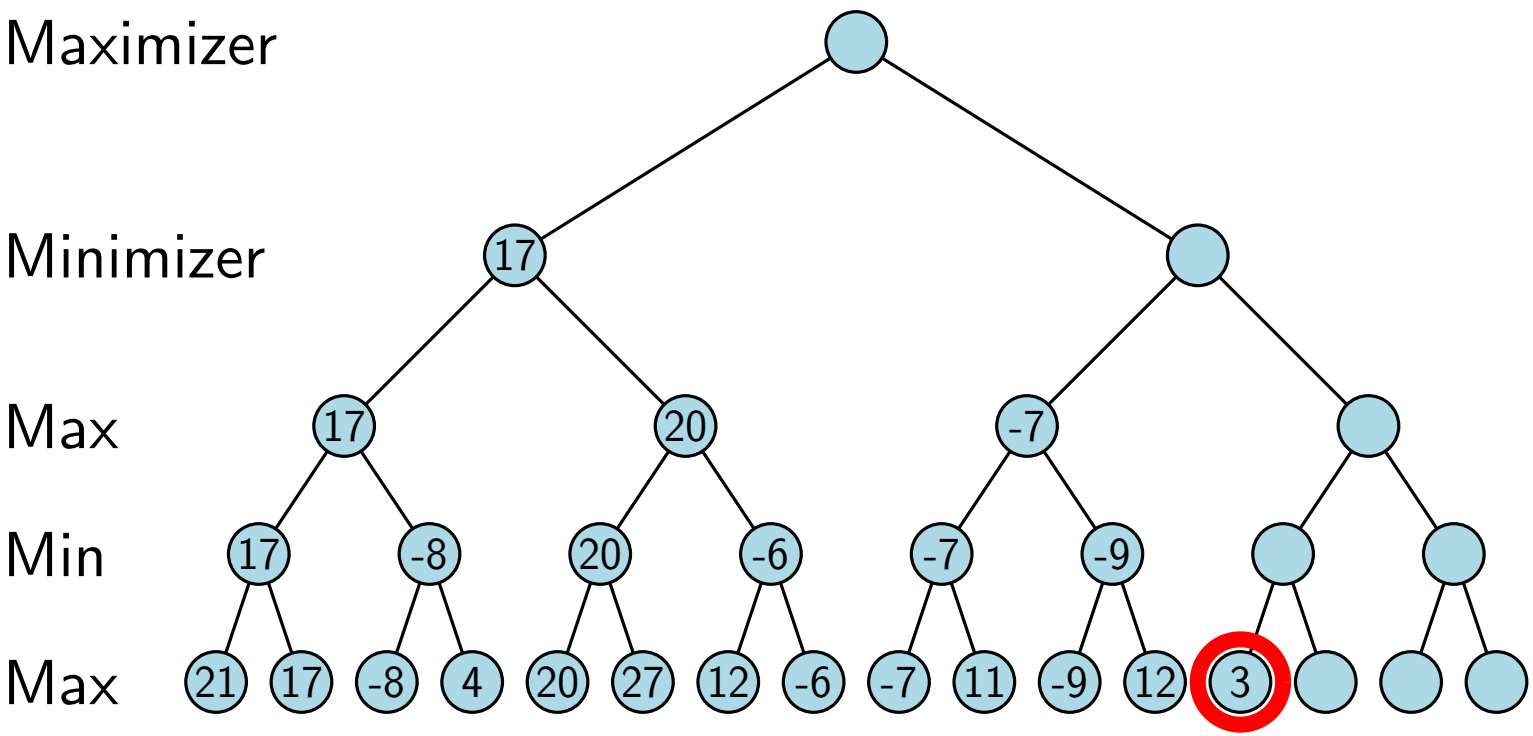
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



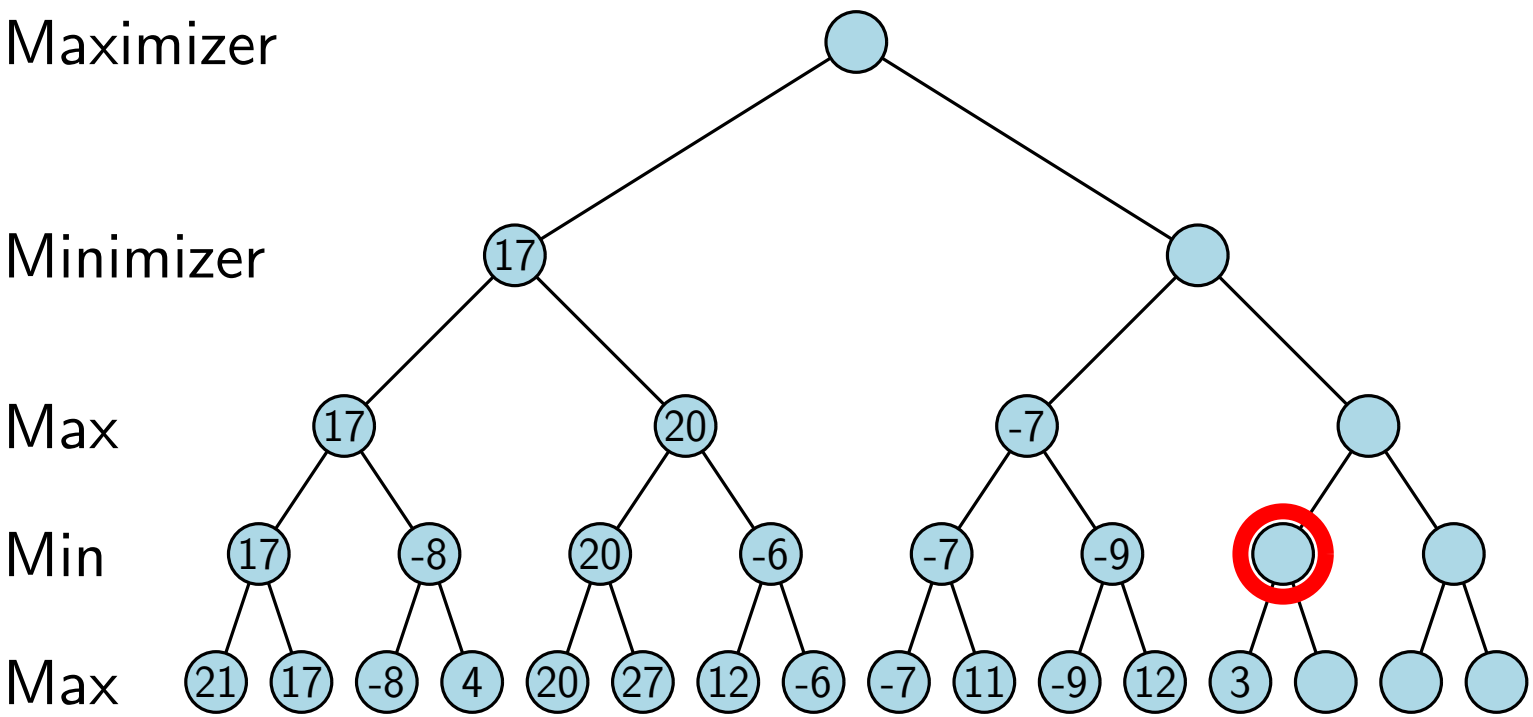
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



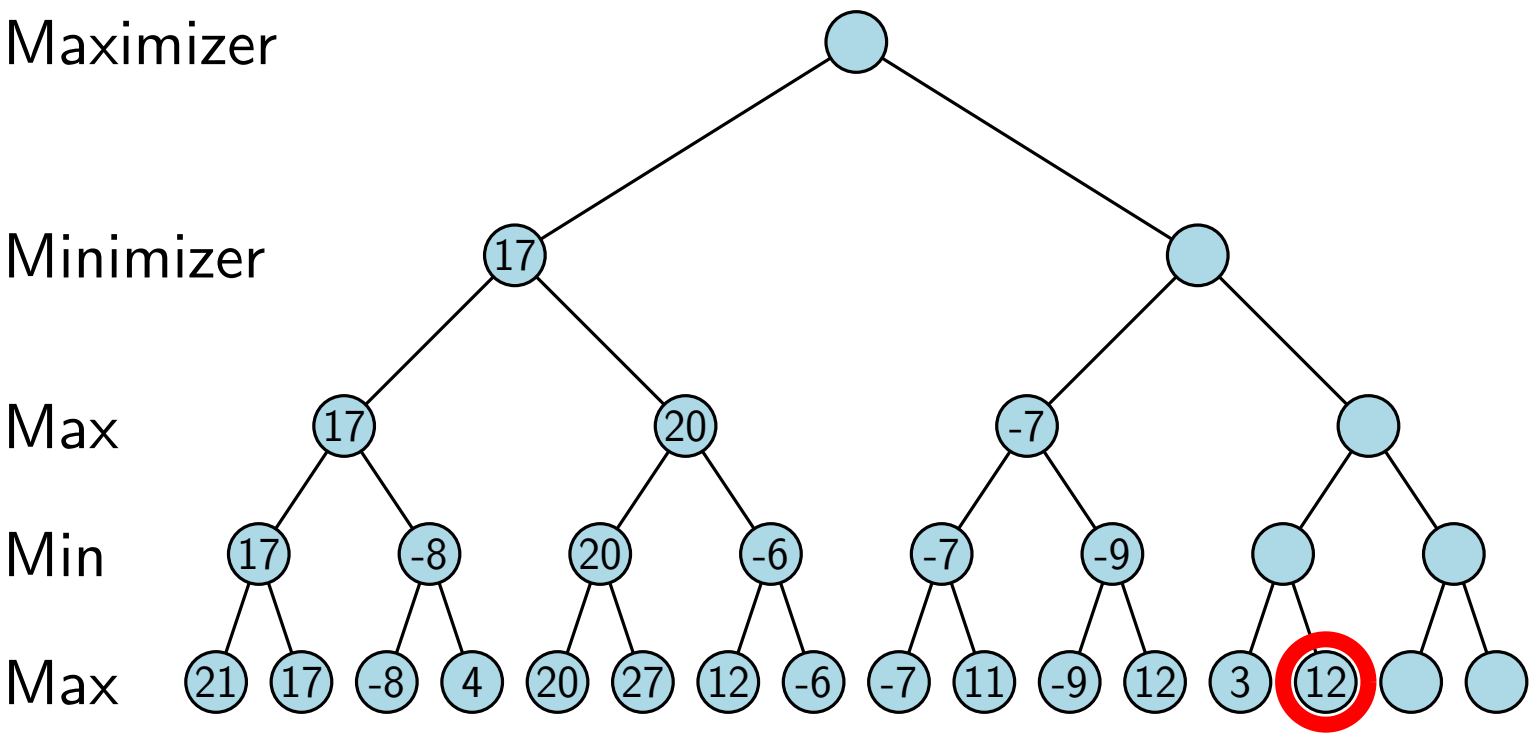
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



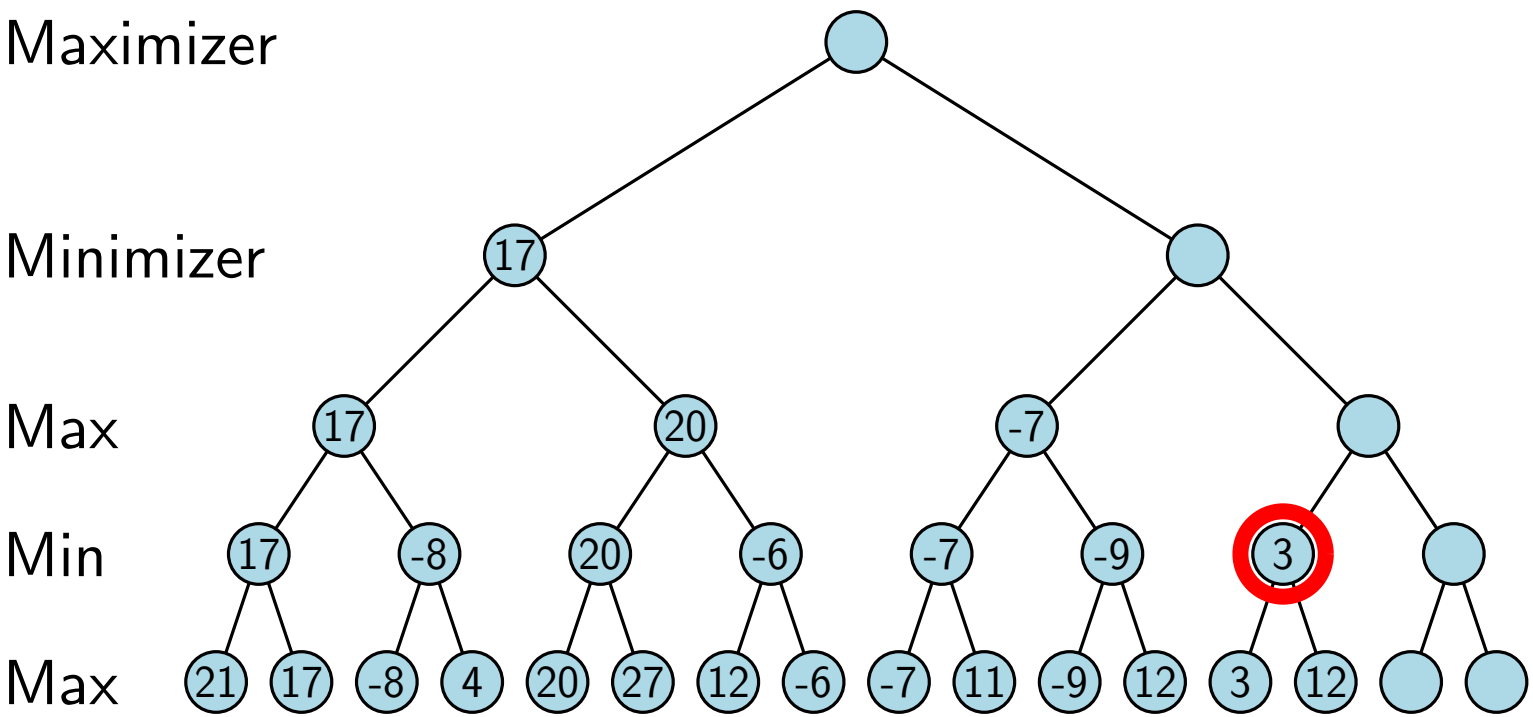
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



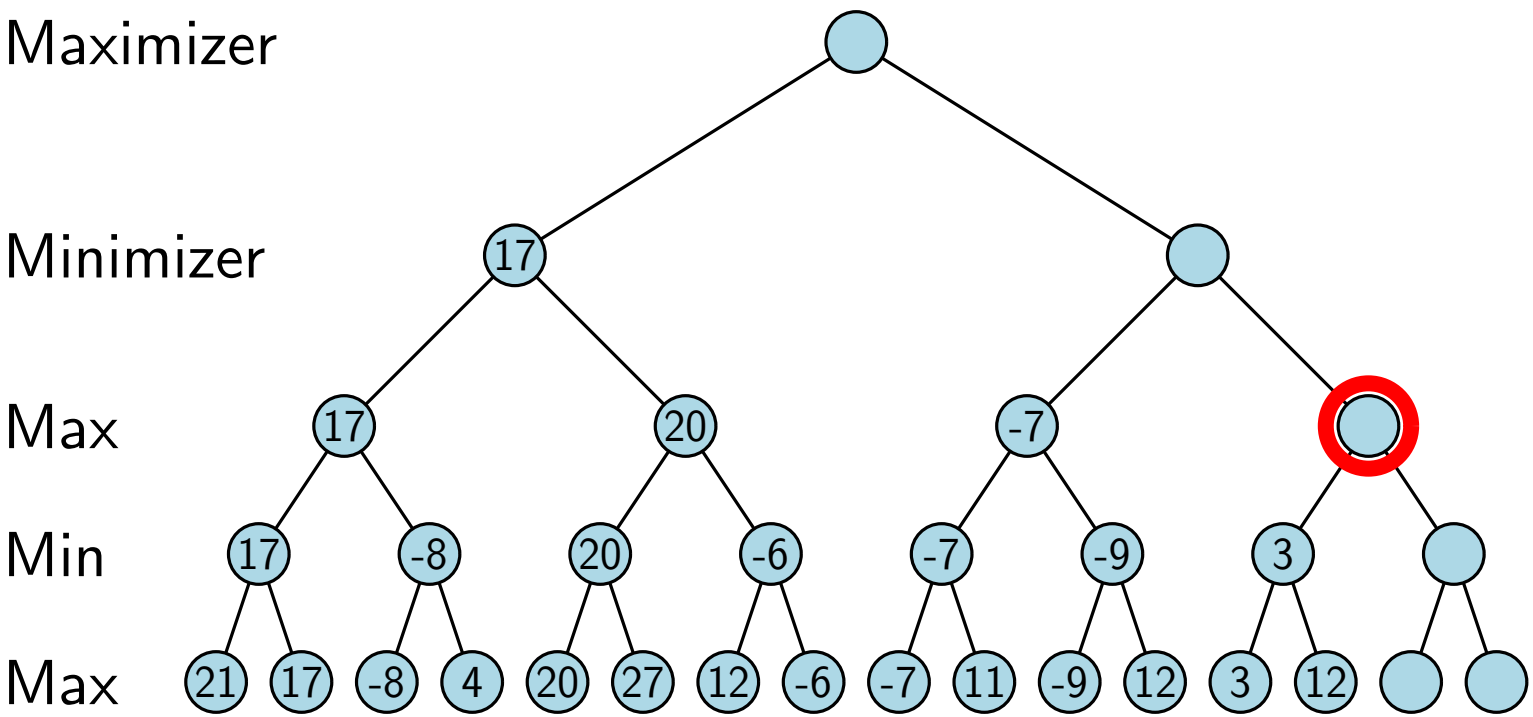
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



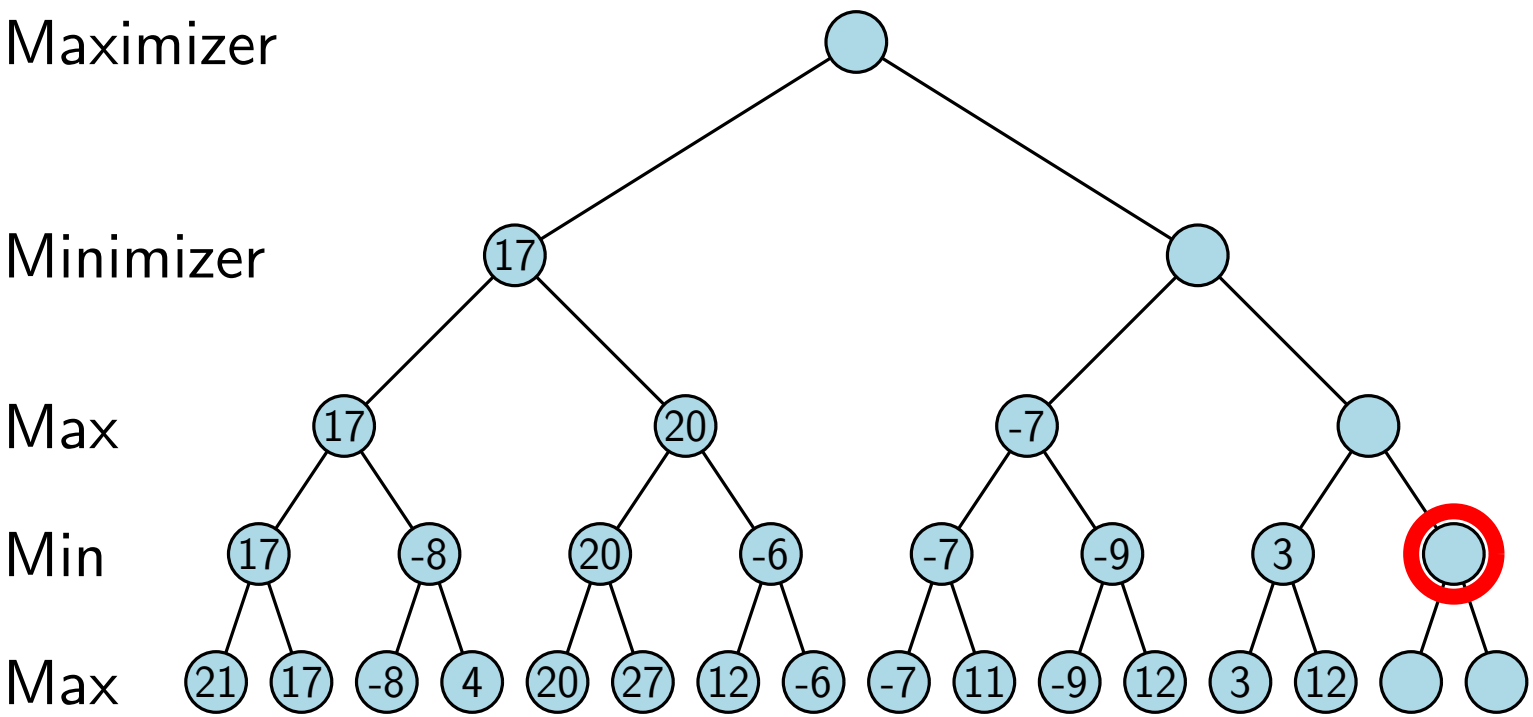
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



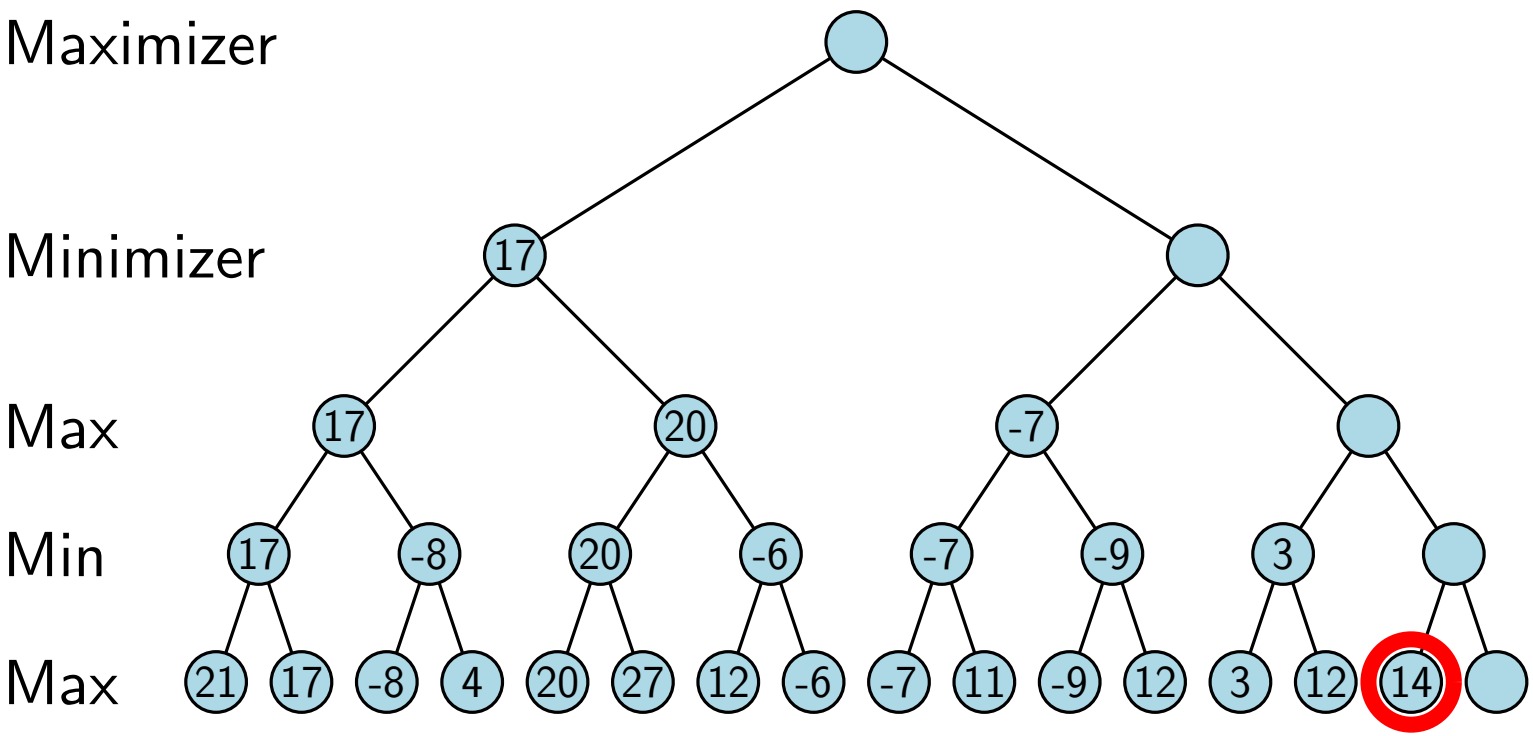
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



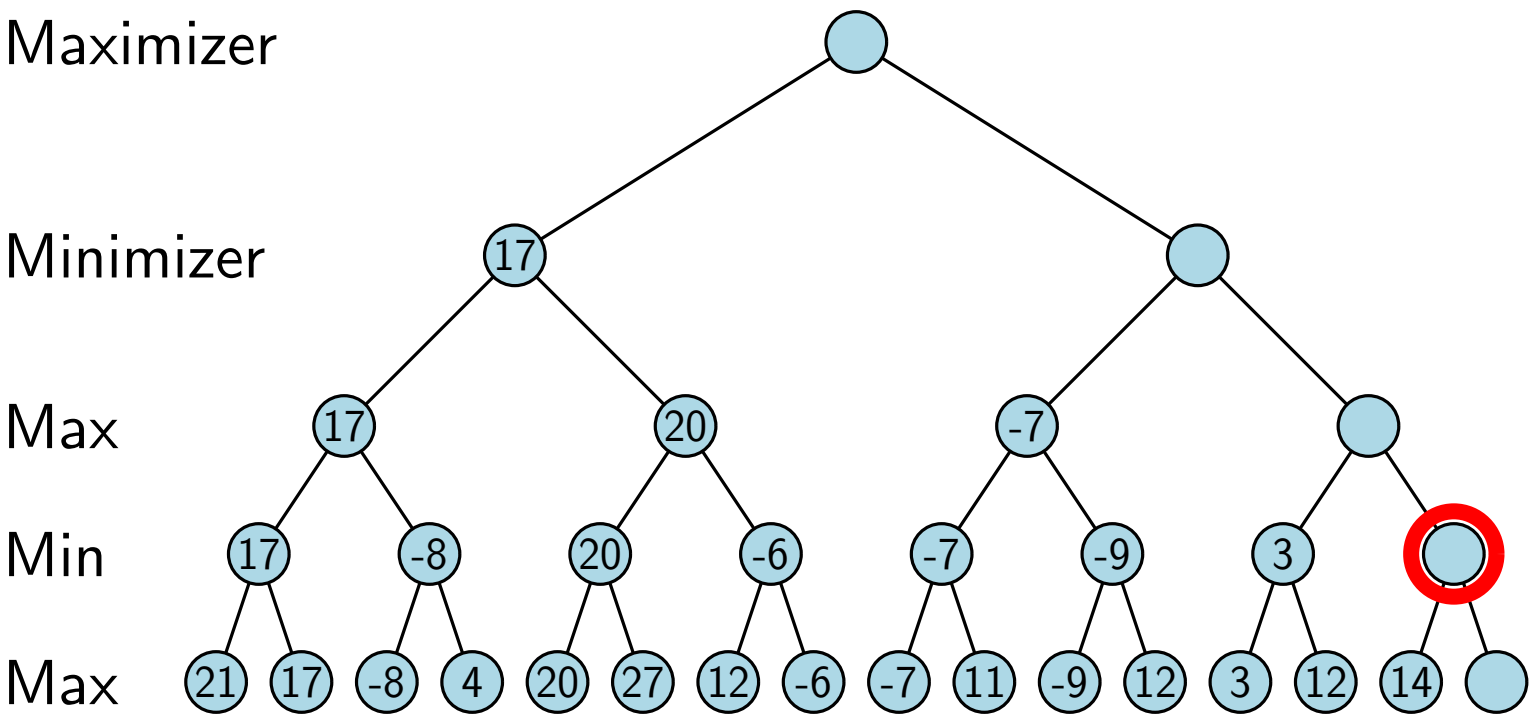
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



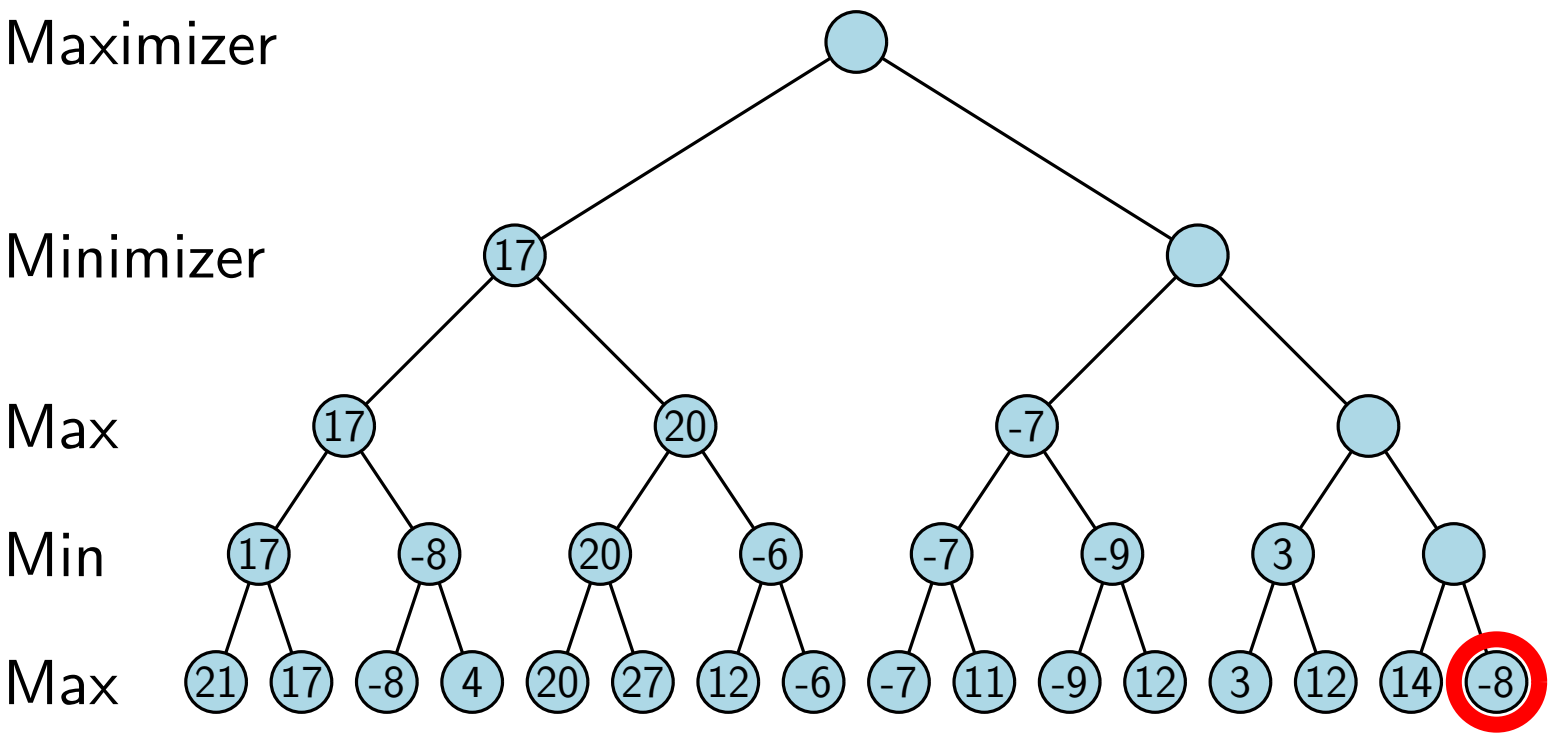
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



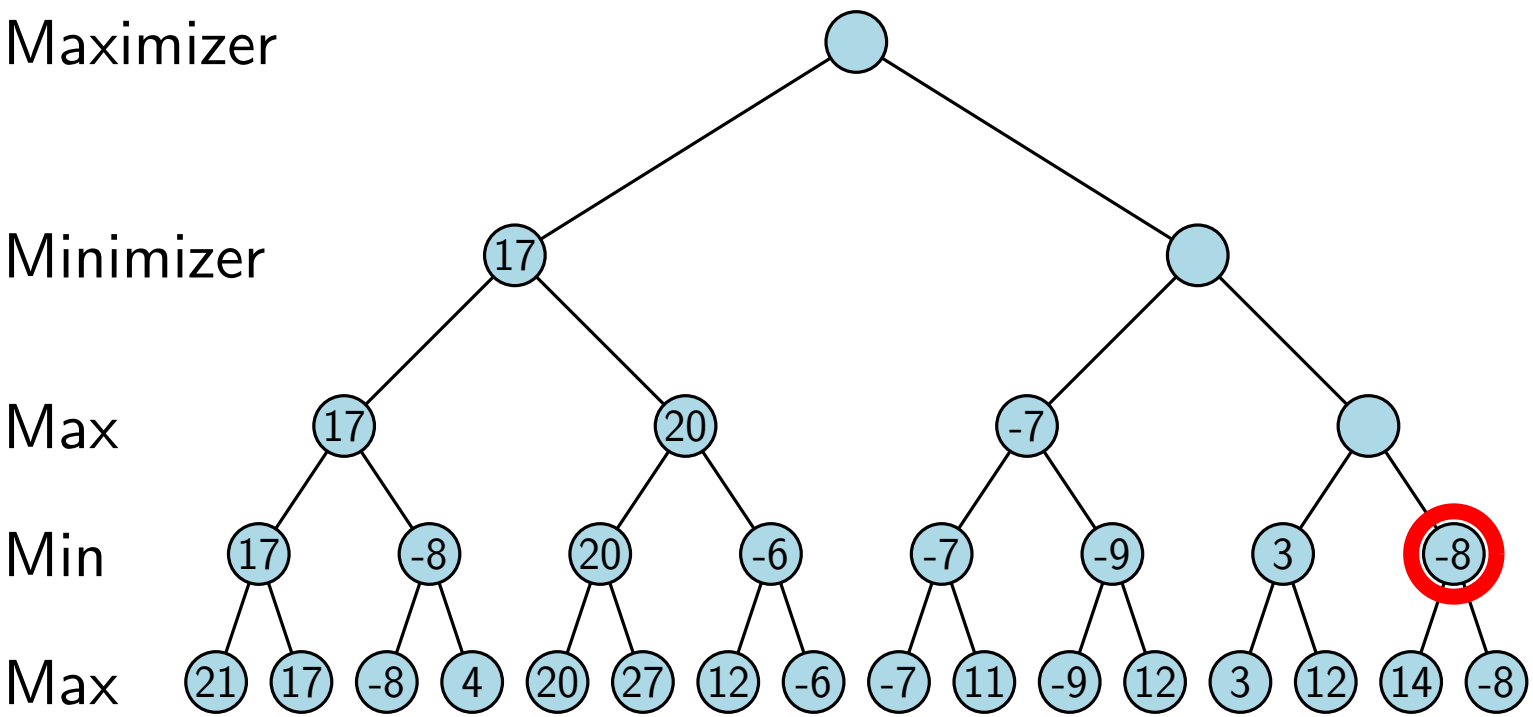
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



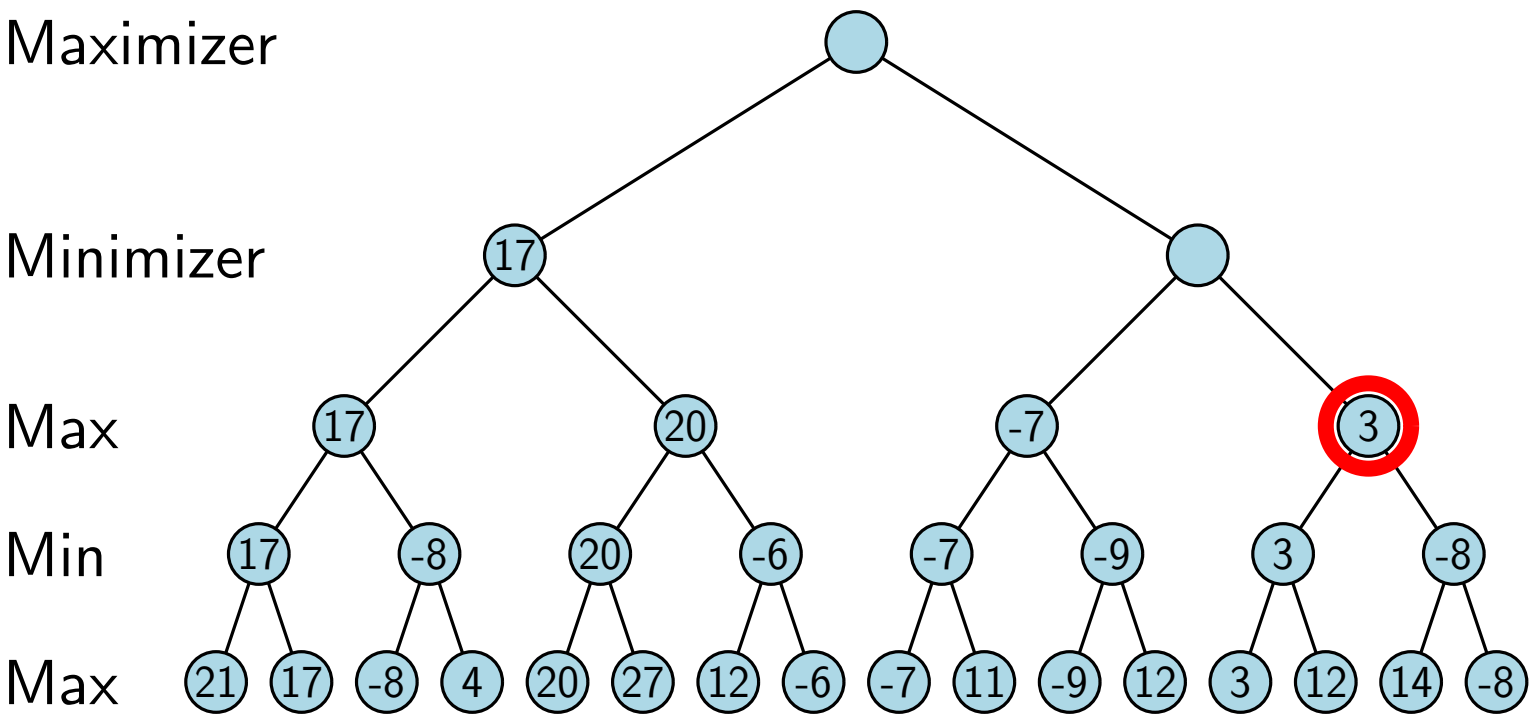
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



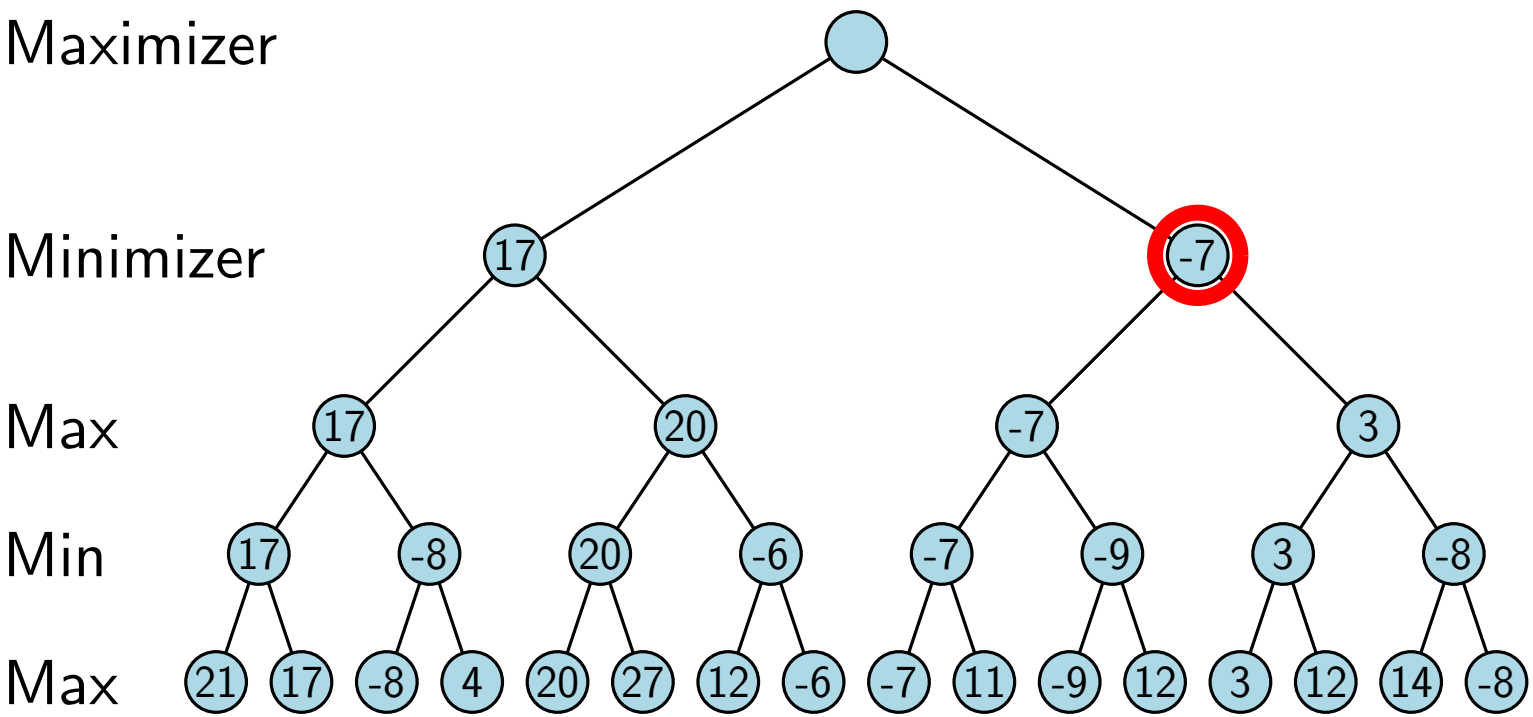
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



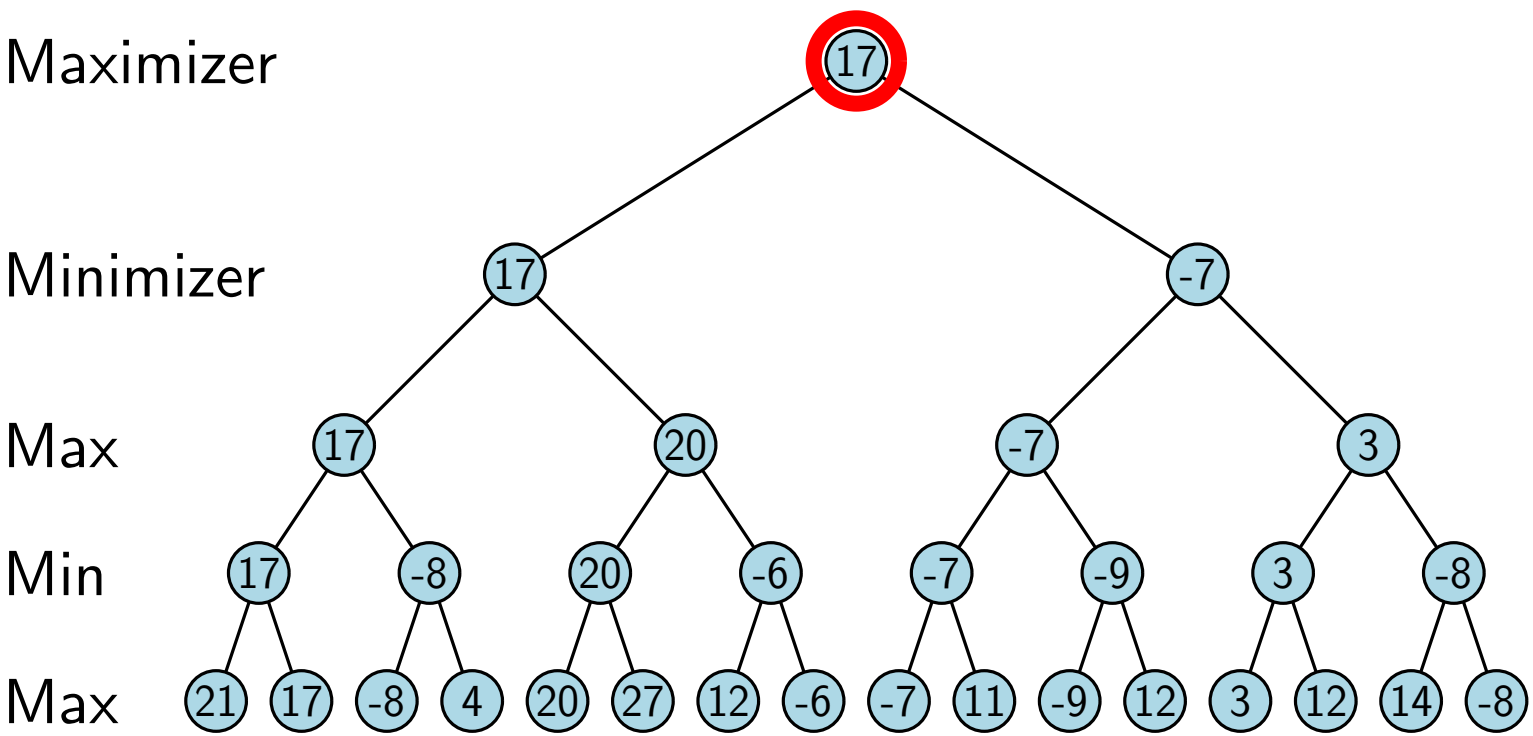
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



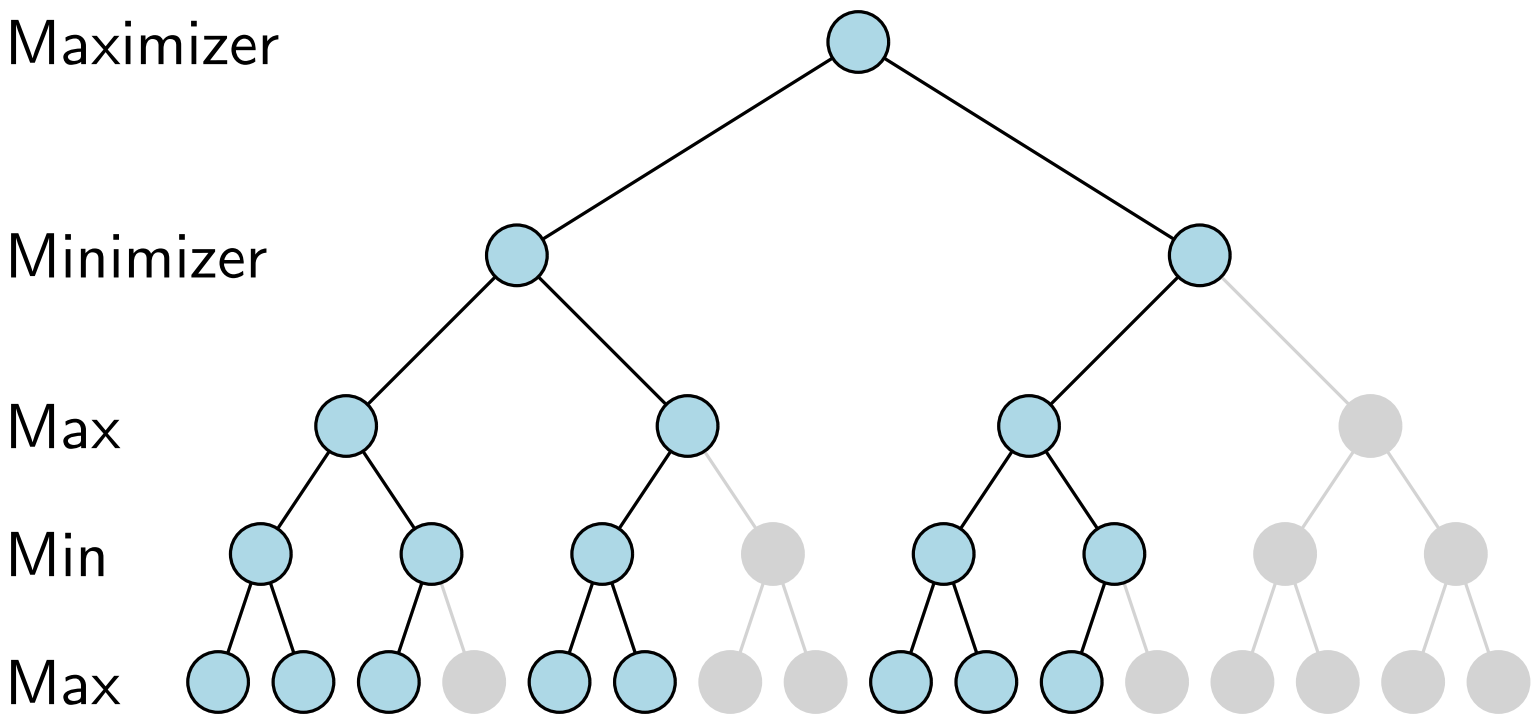
Game Tree: Min Max Algorithm

In a 2-player game usually player A is the maximizer and player B is the minimizer of the state evaluation.



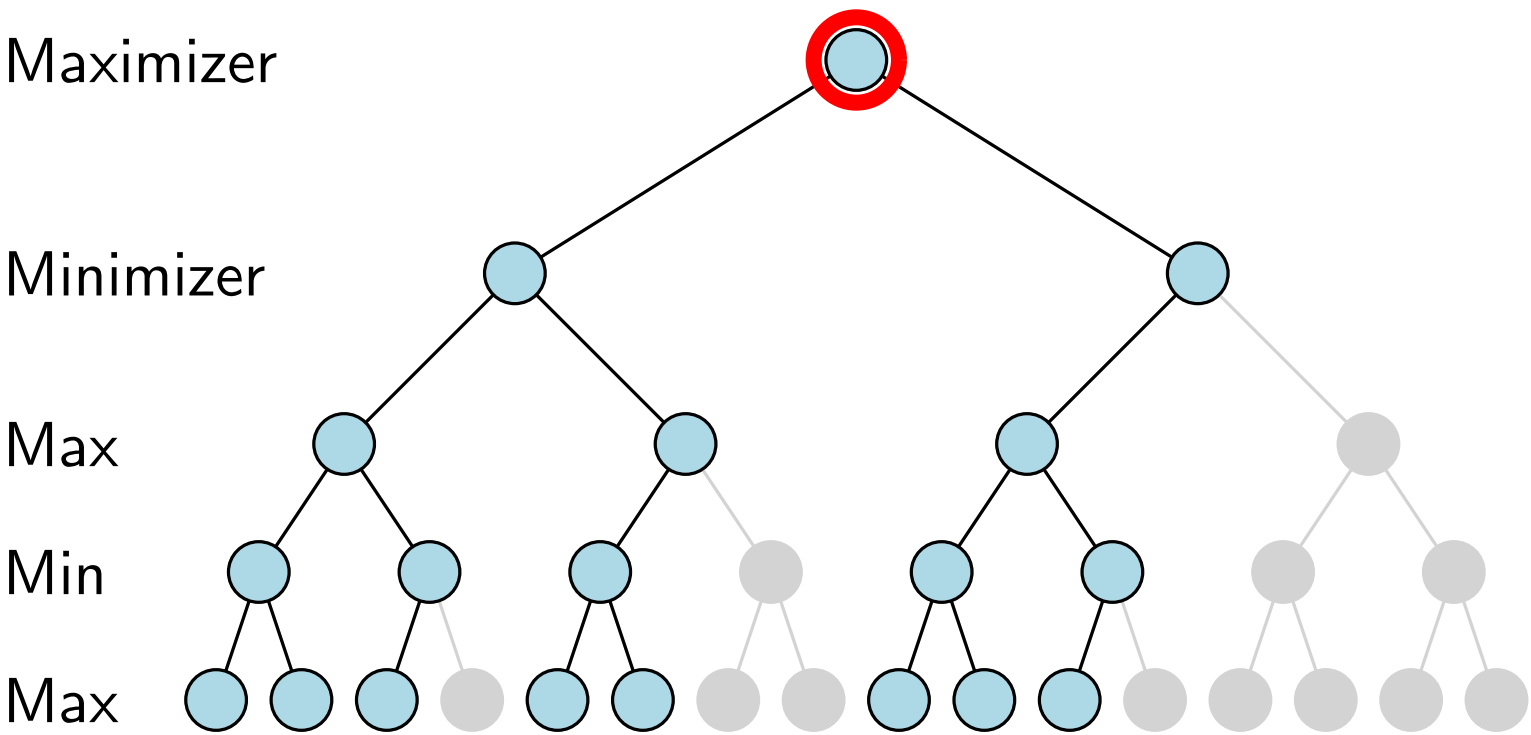
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



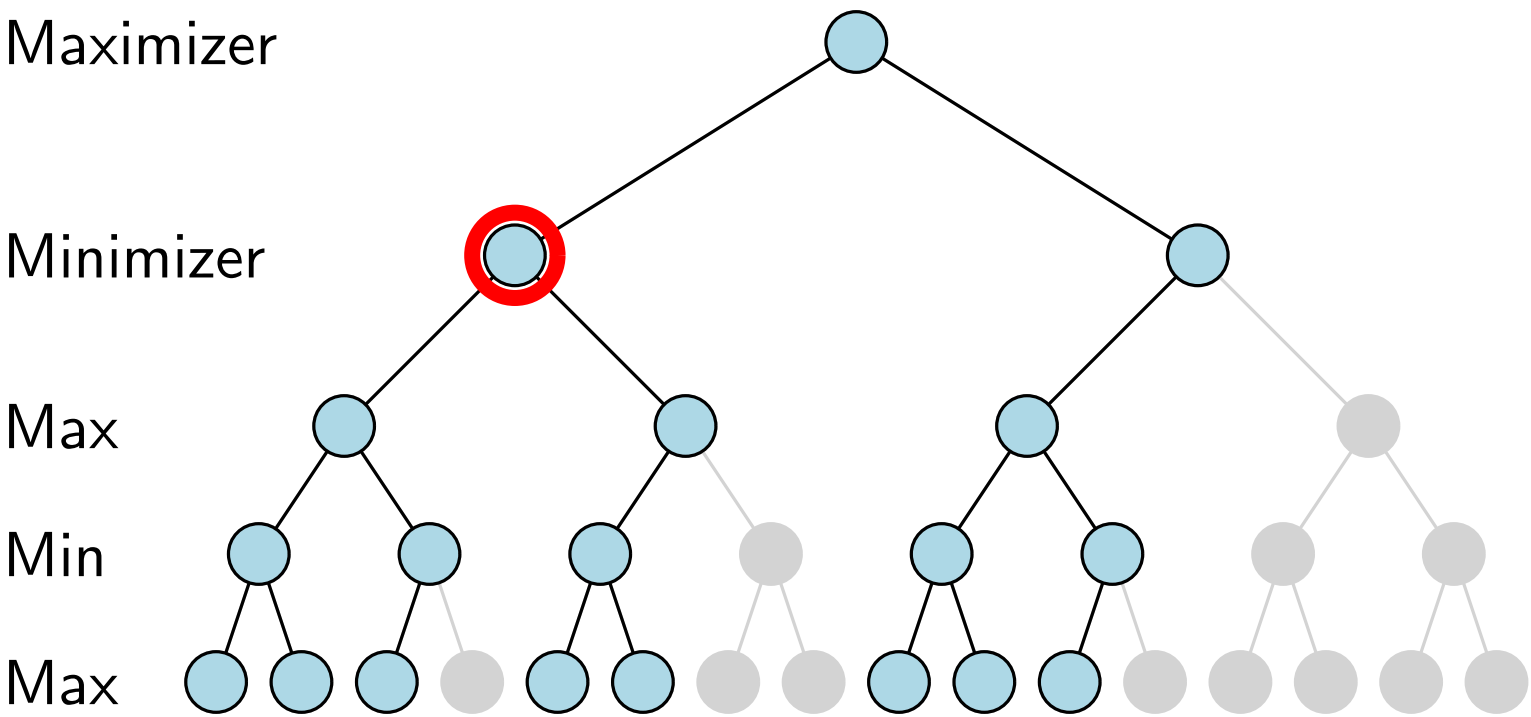
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



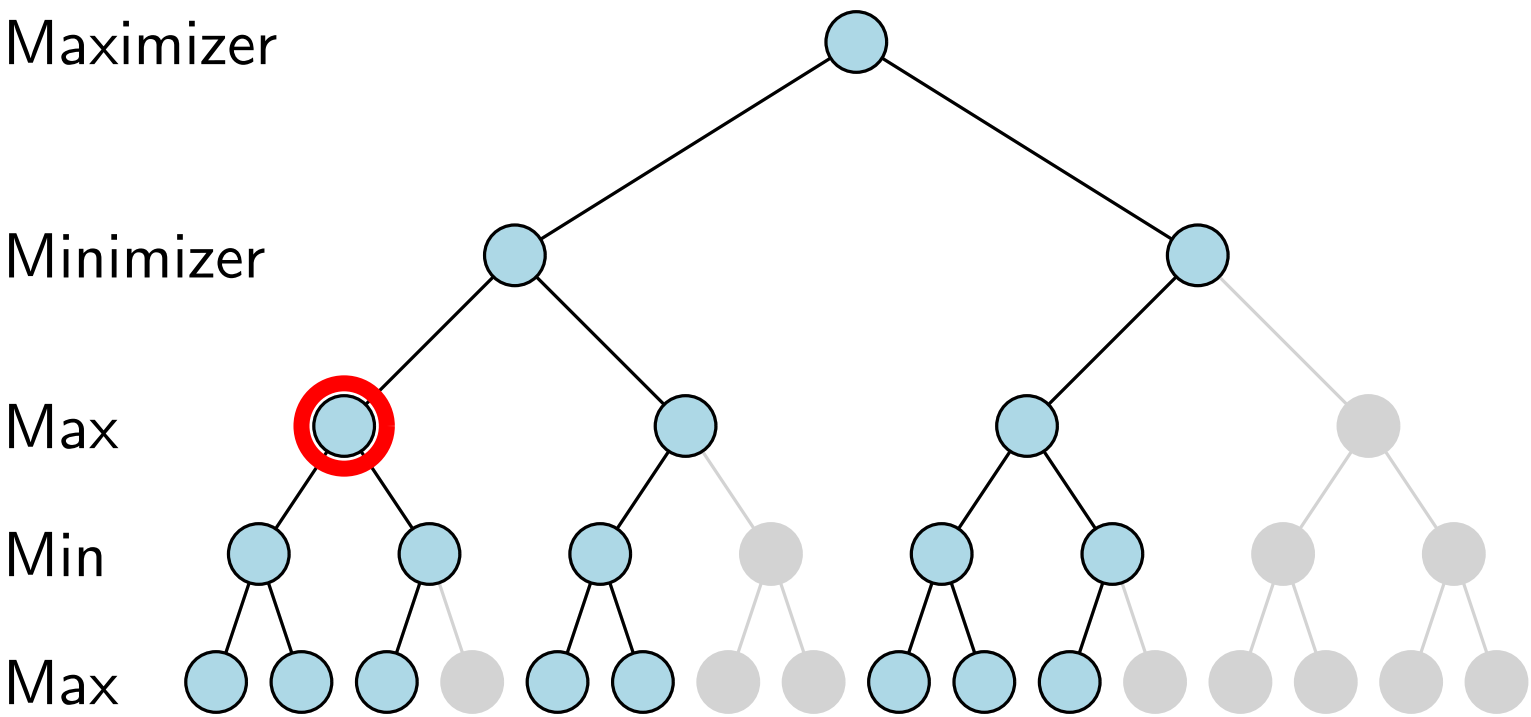
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



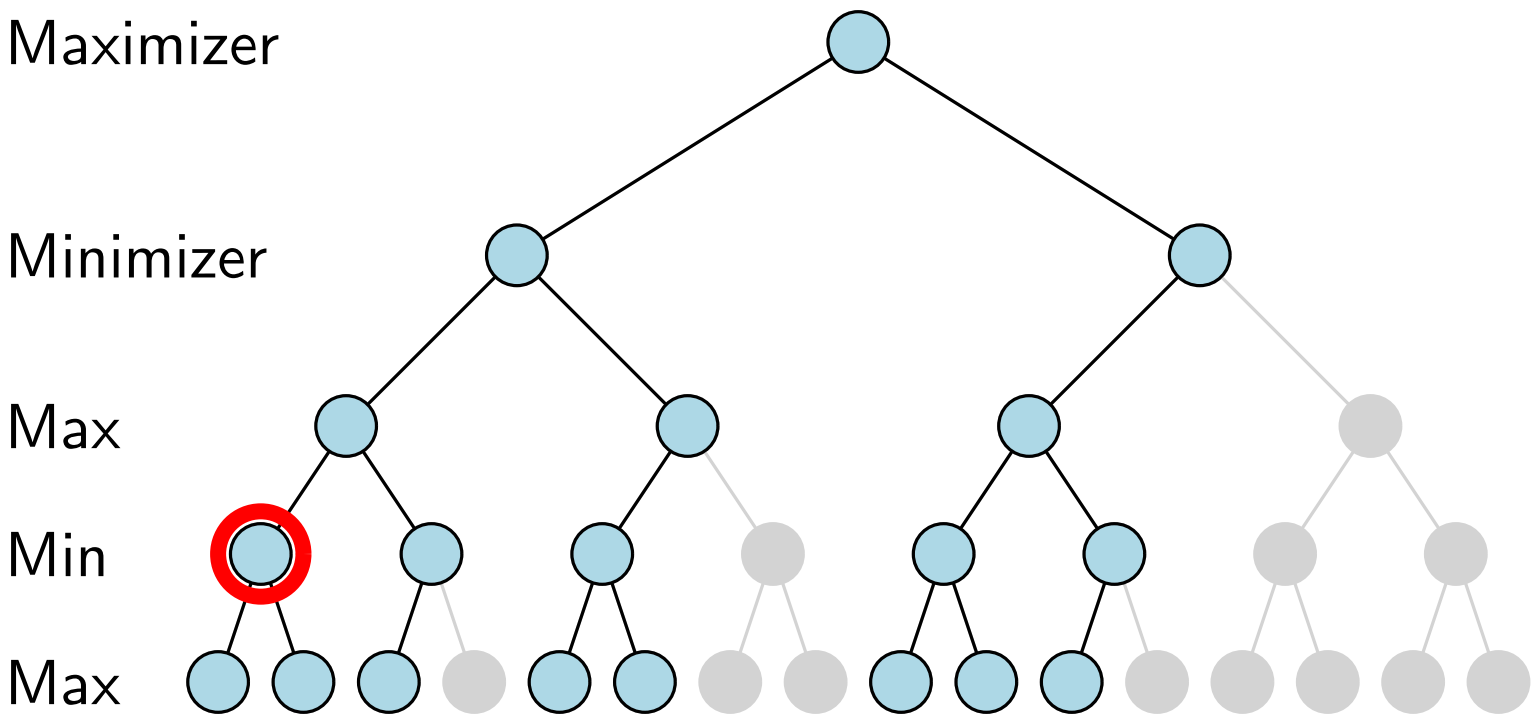
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



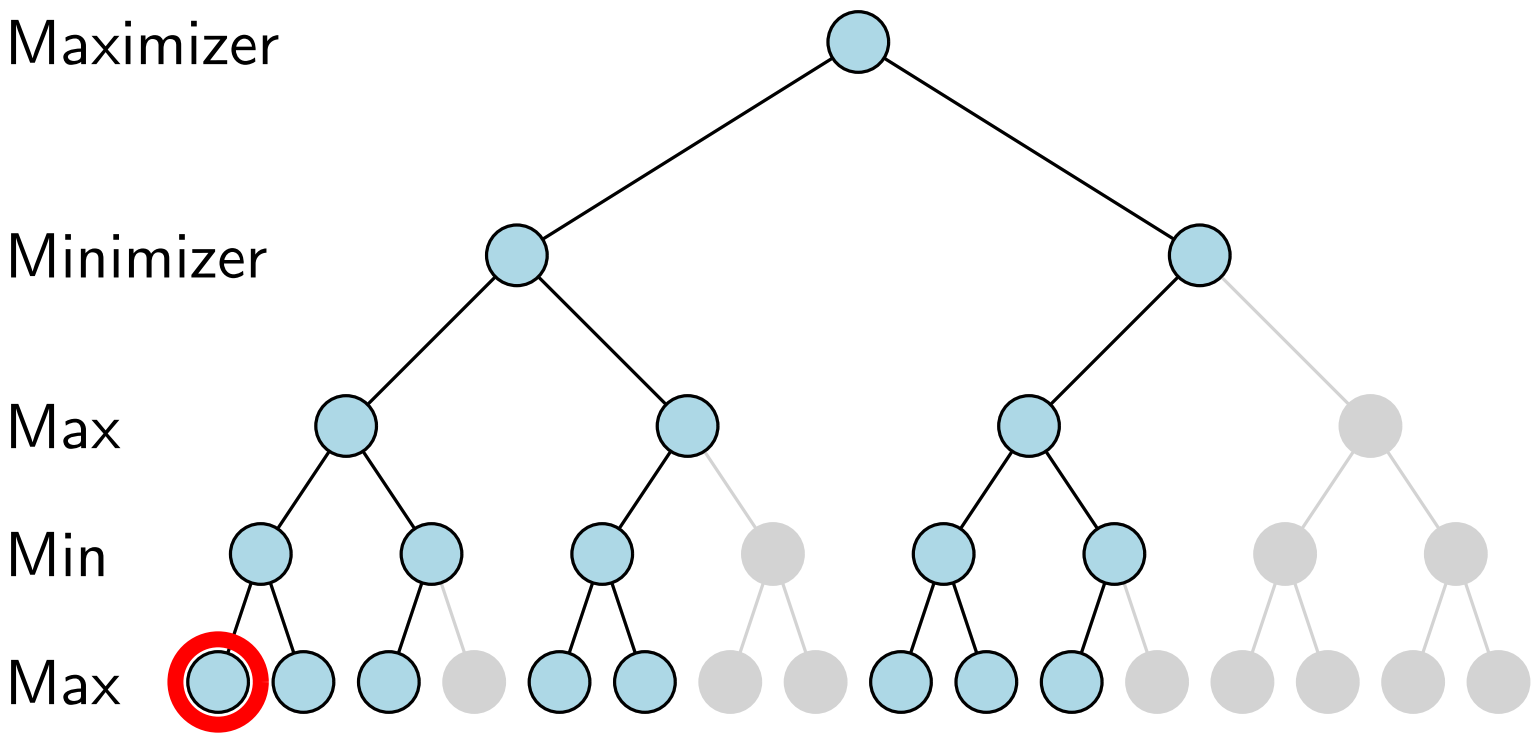
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



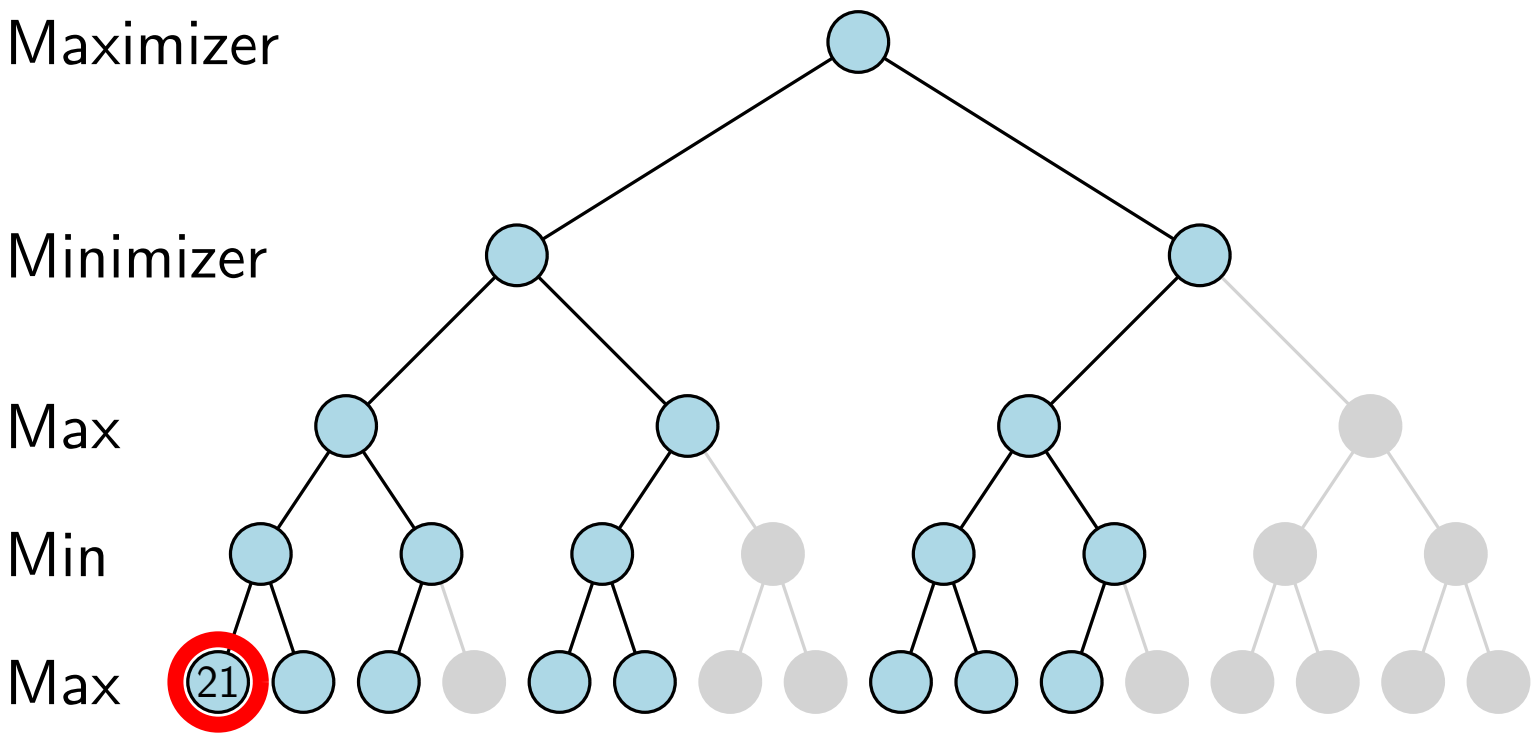
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



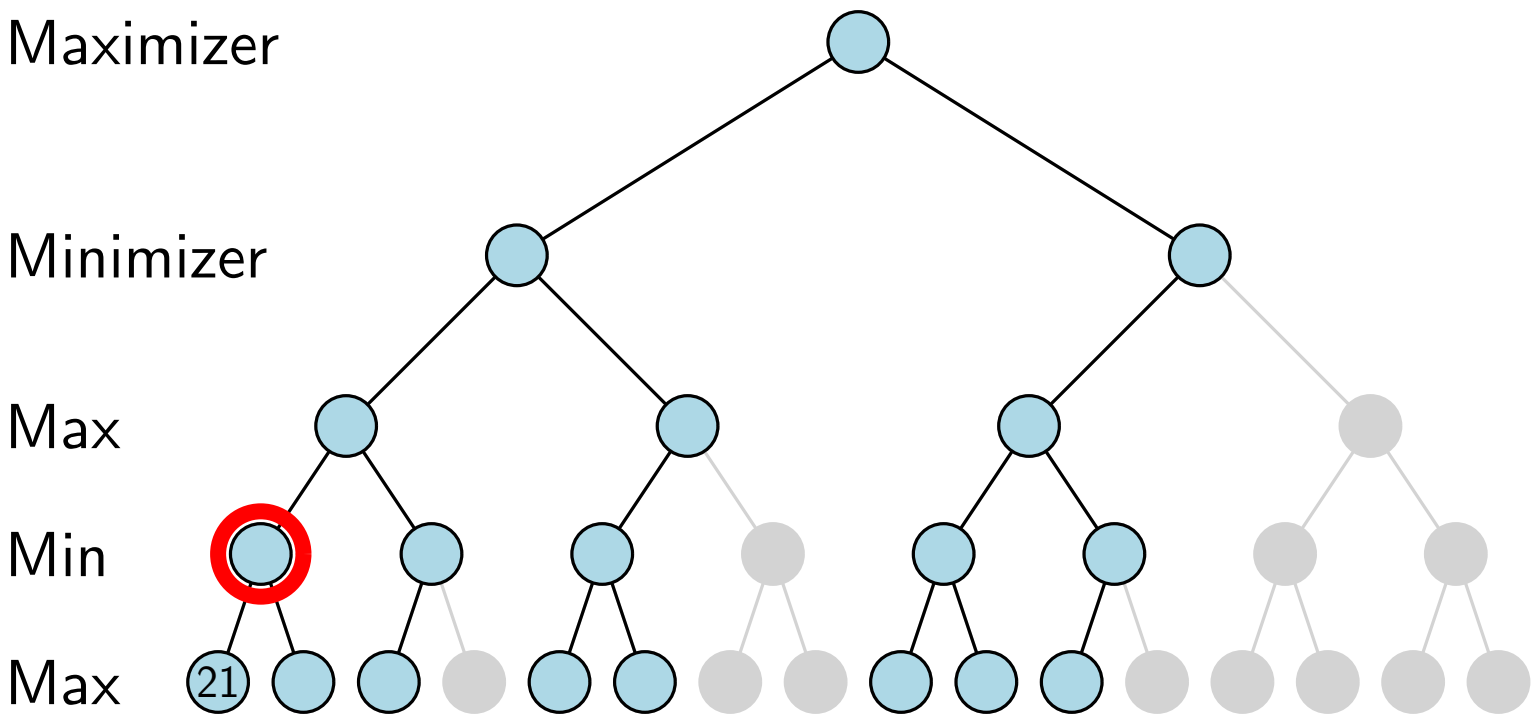
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



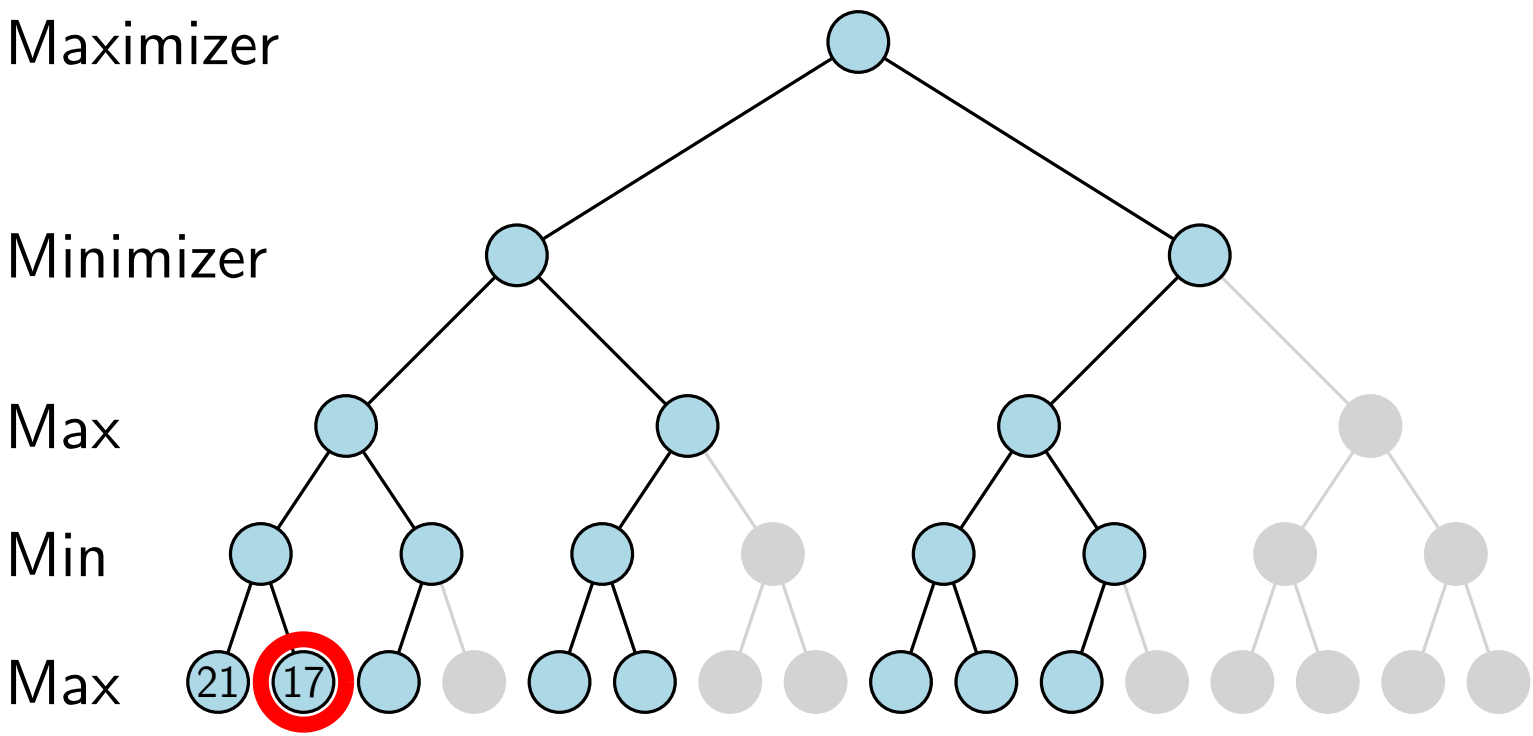
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



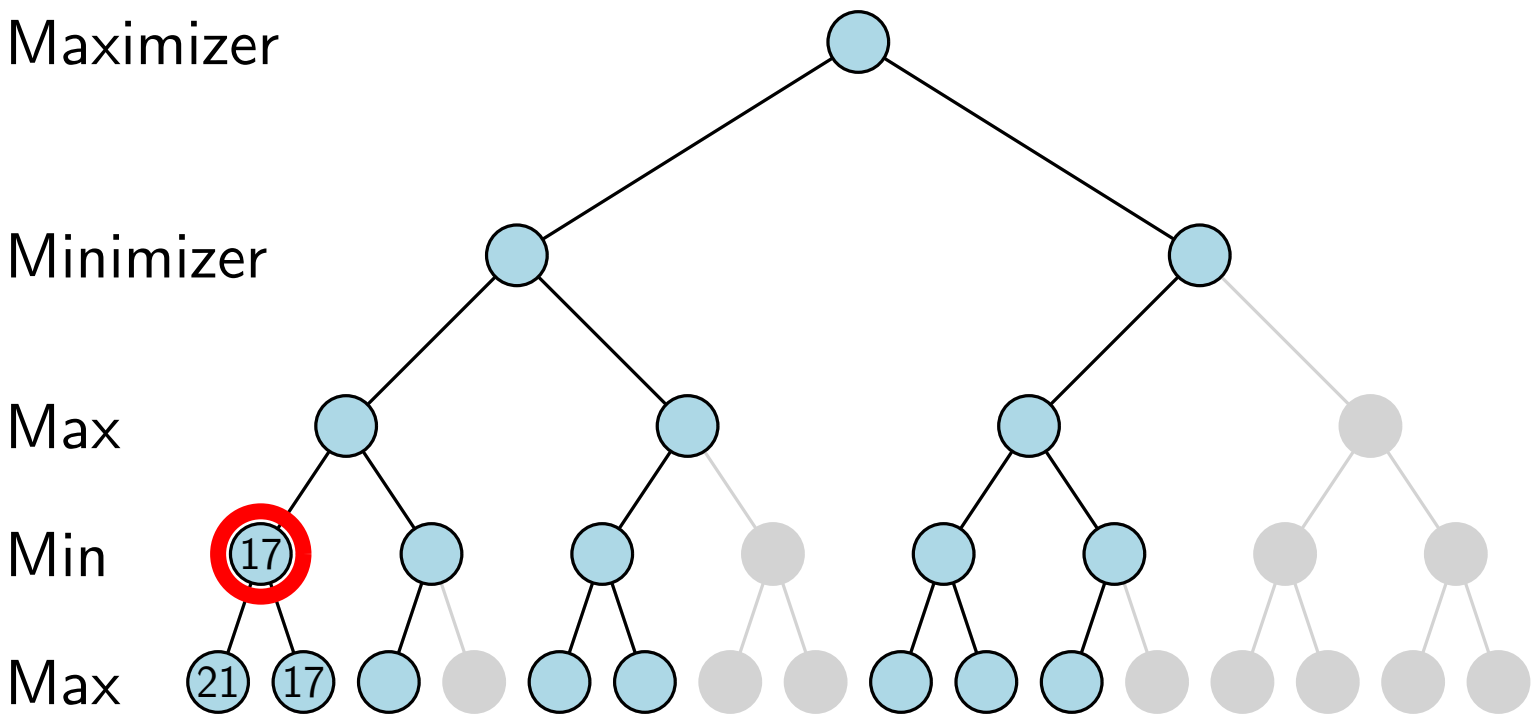
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



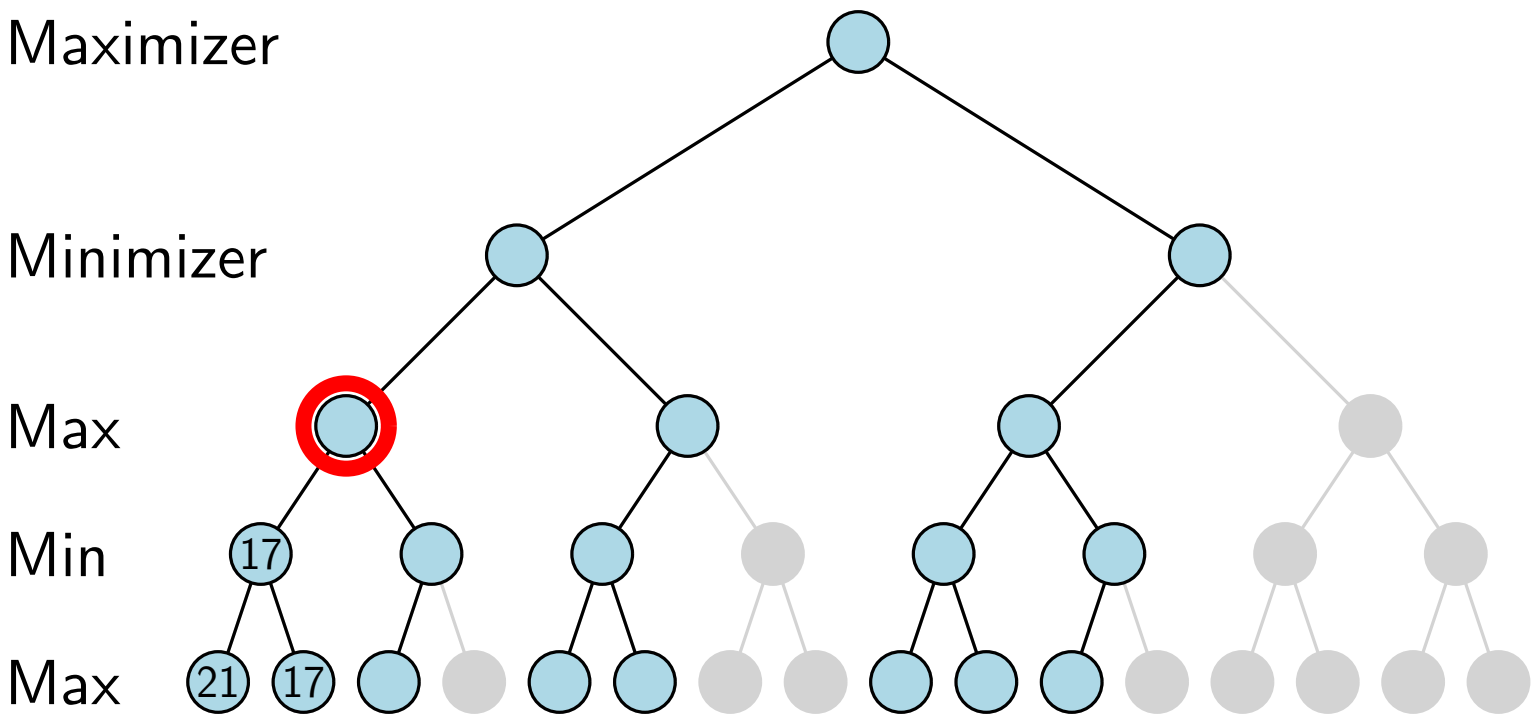
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



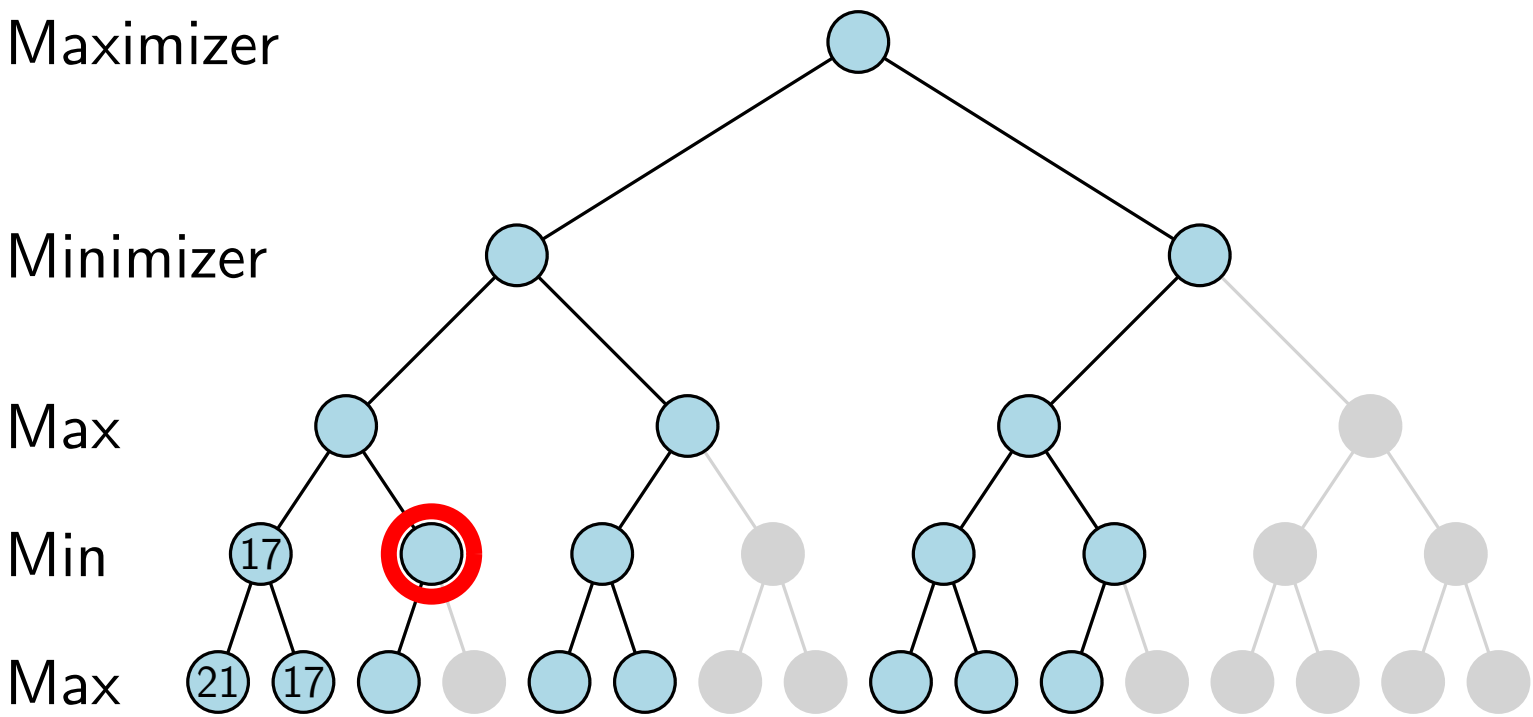
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



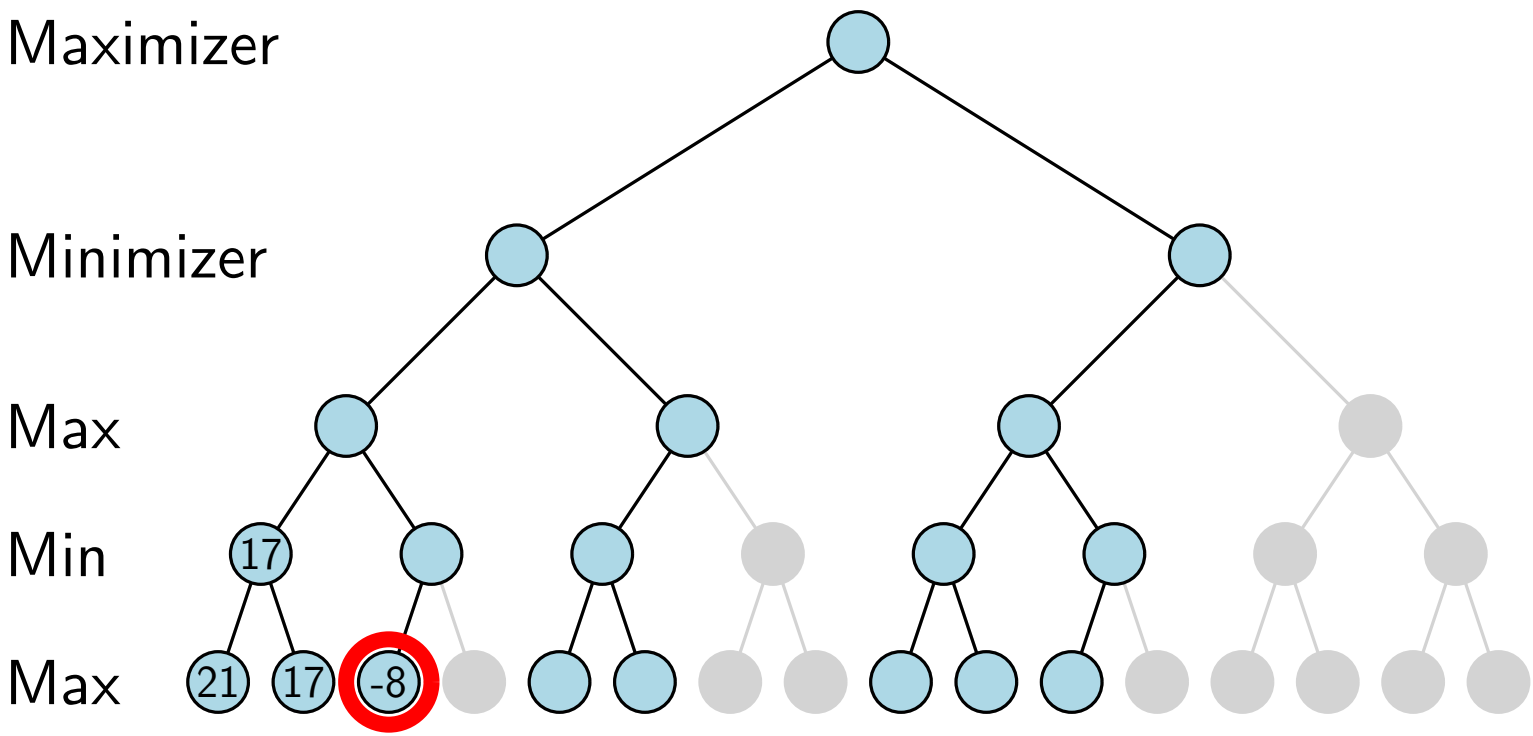
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



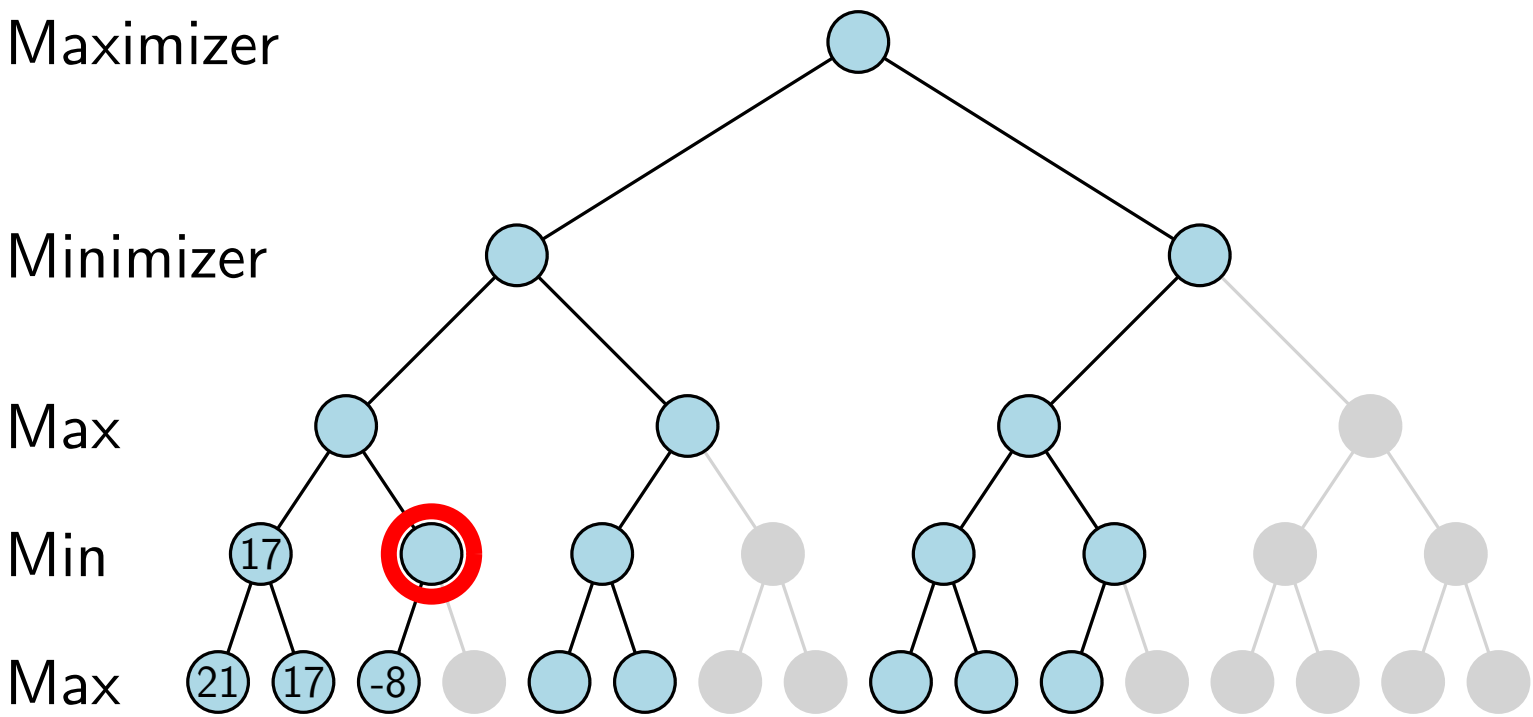
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



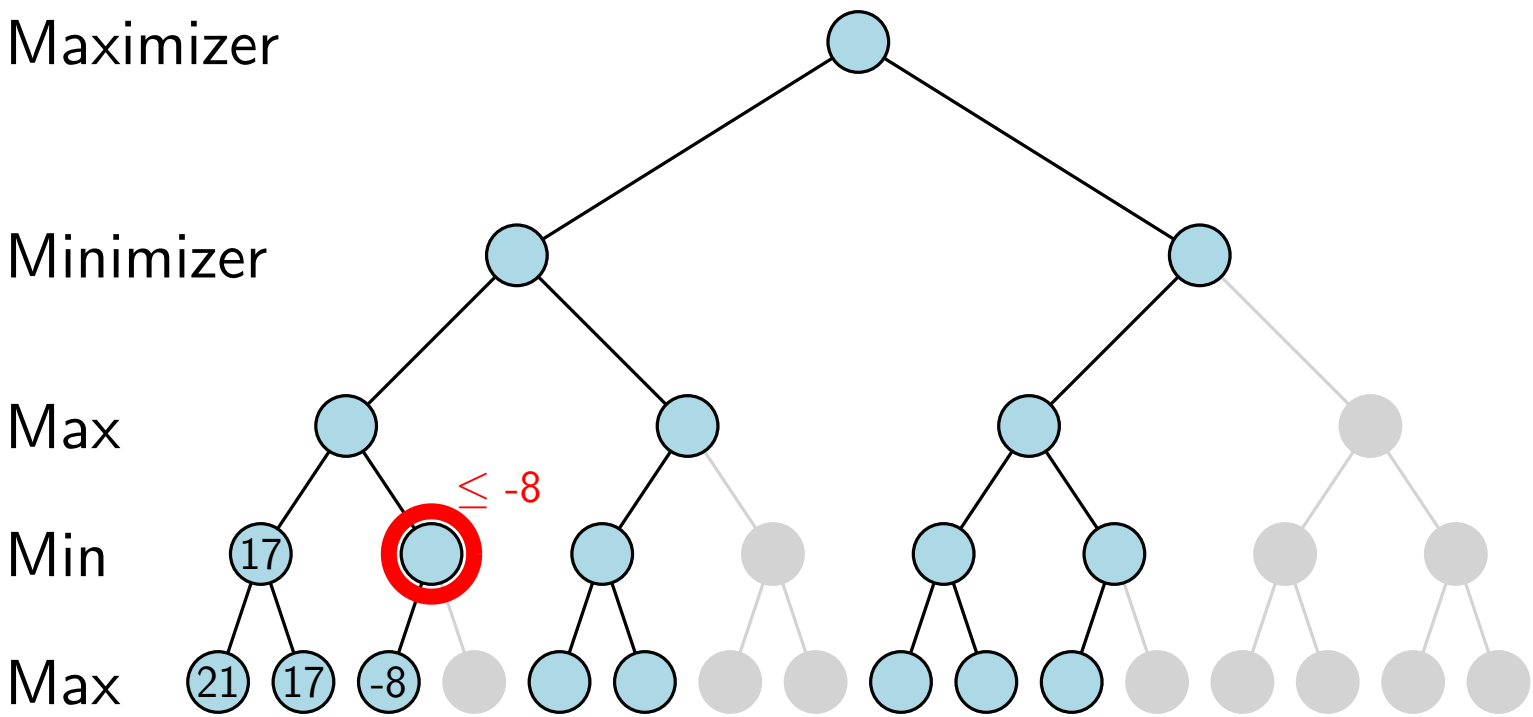
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



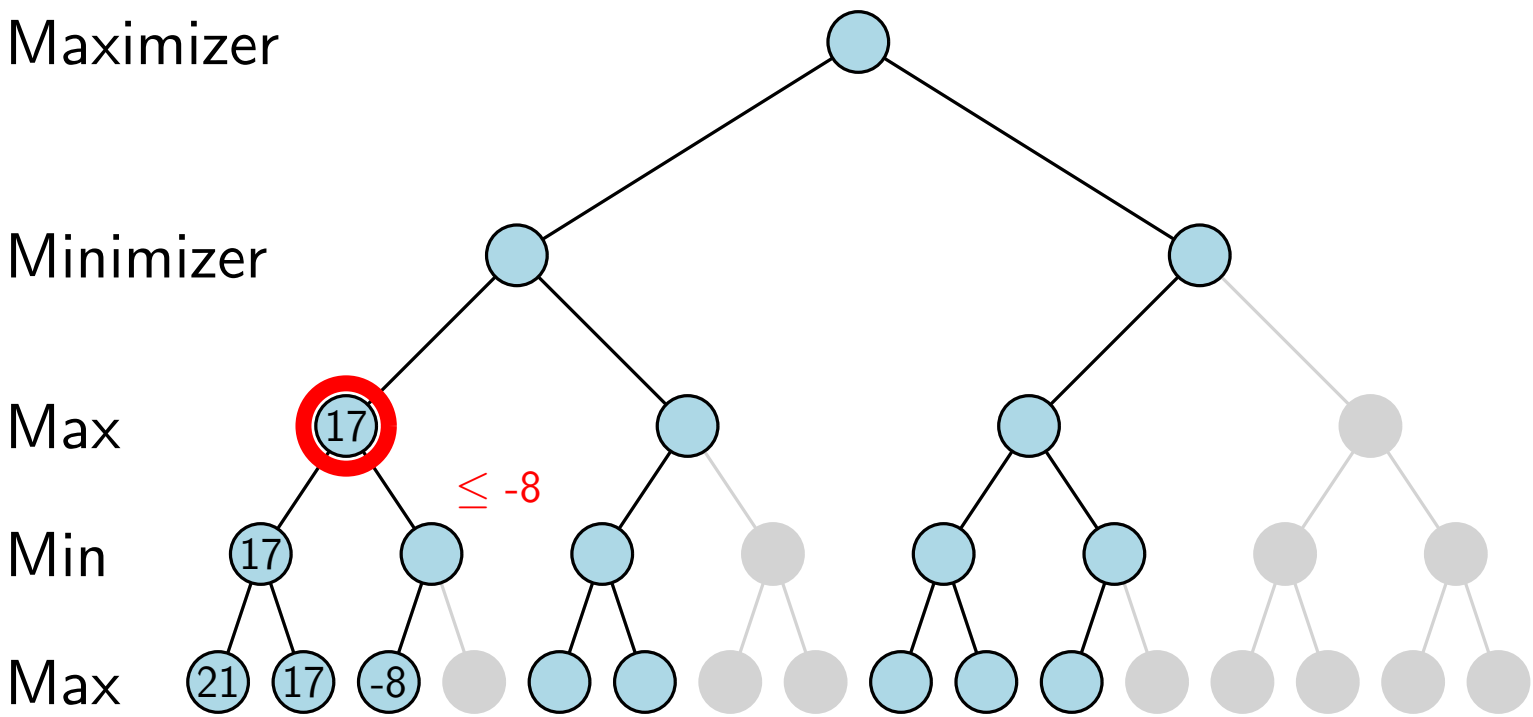
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



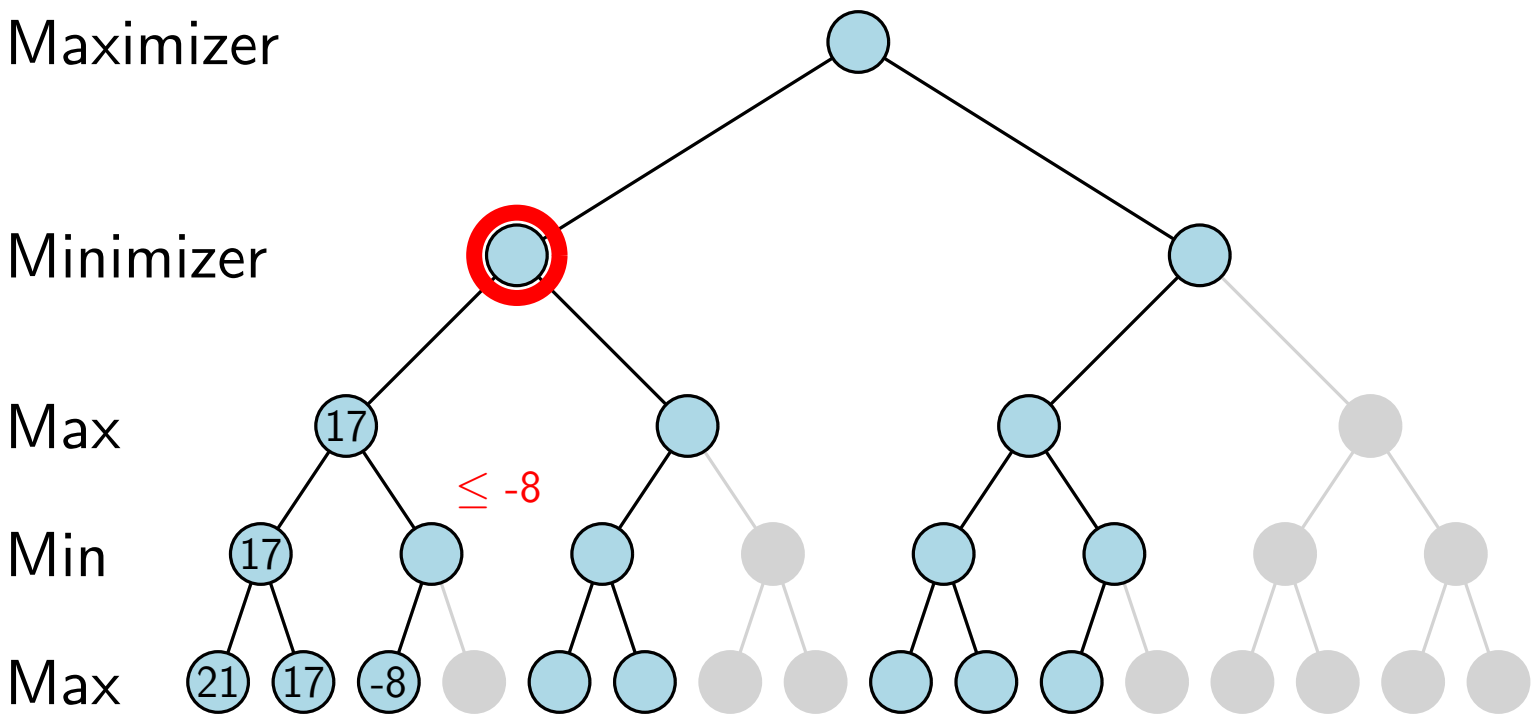
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



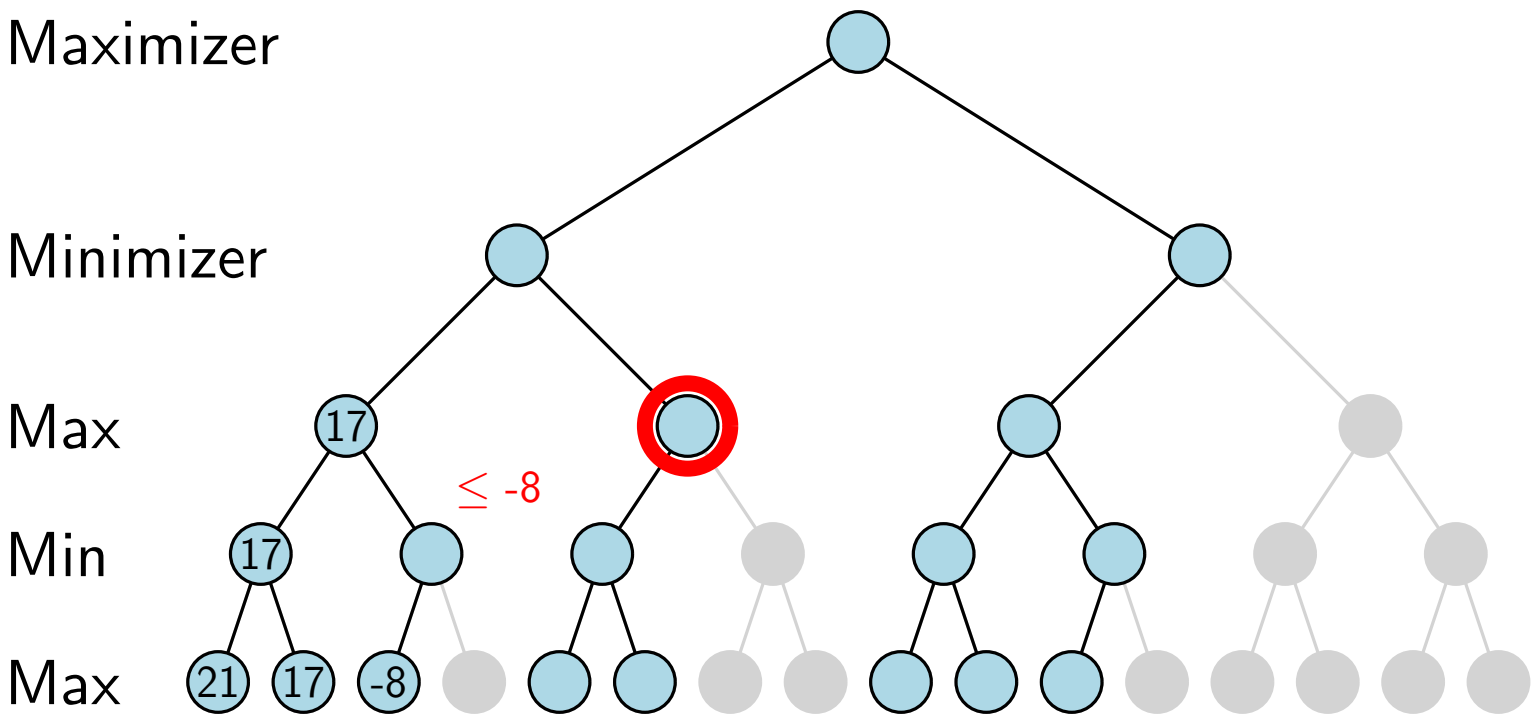
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



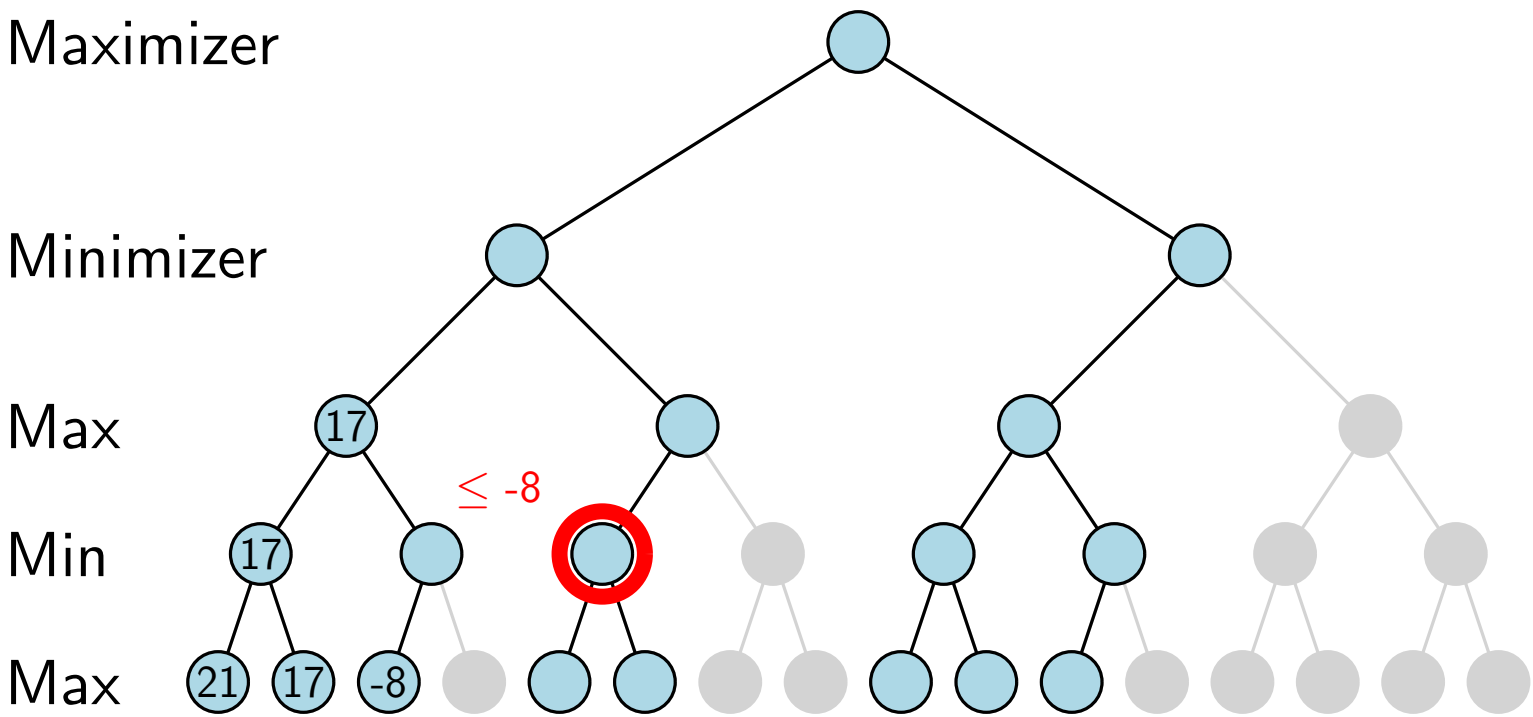
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



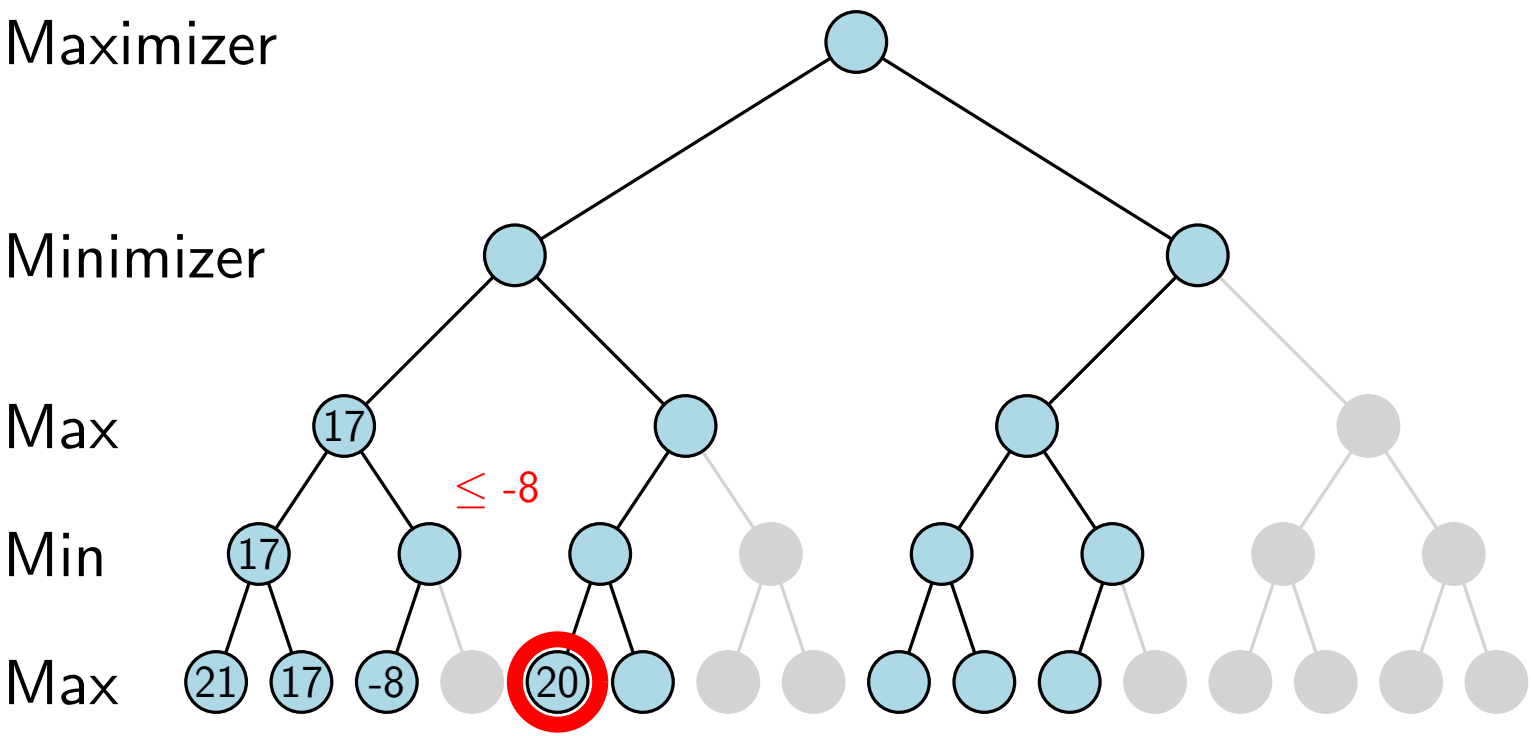
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



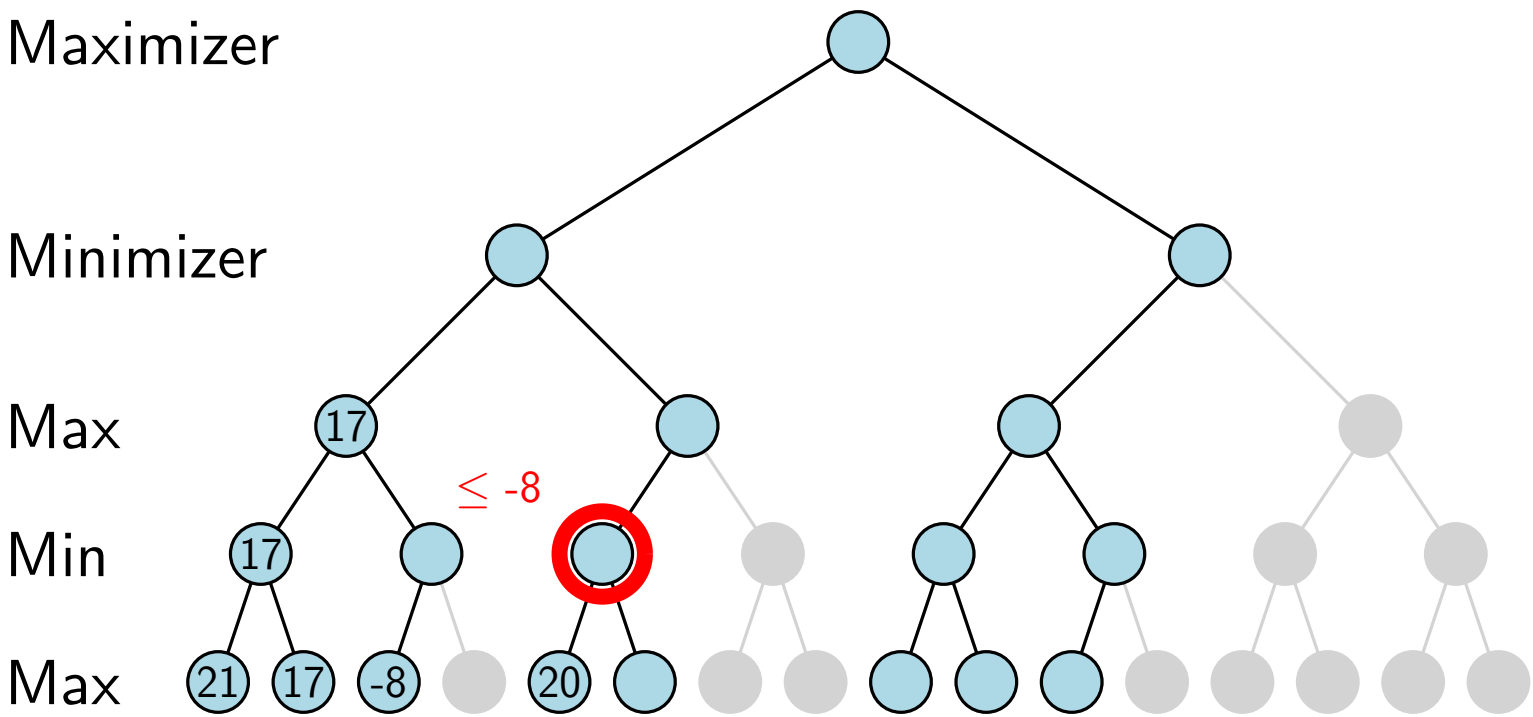
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



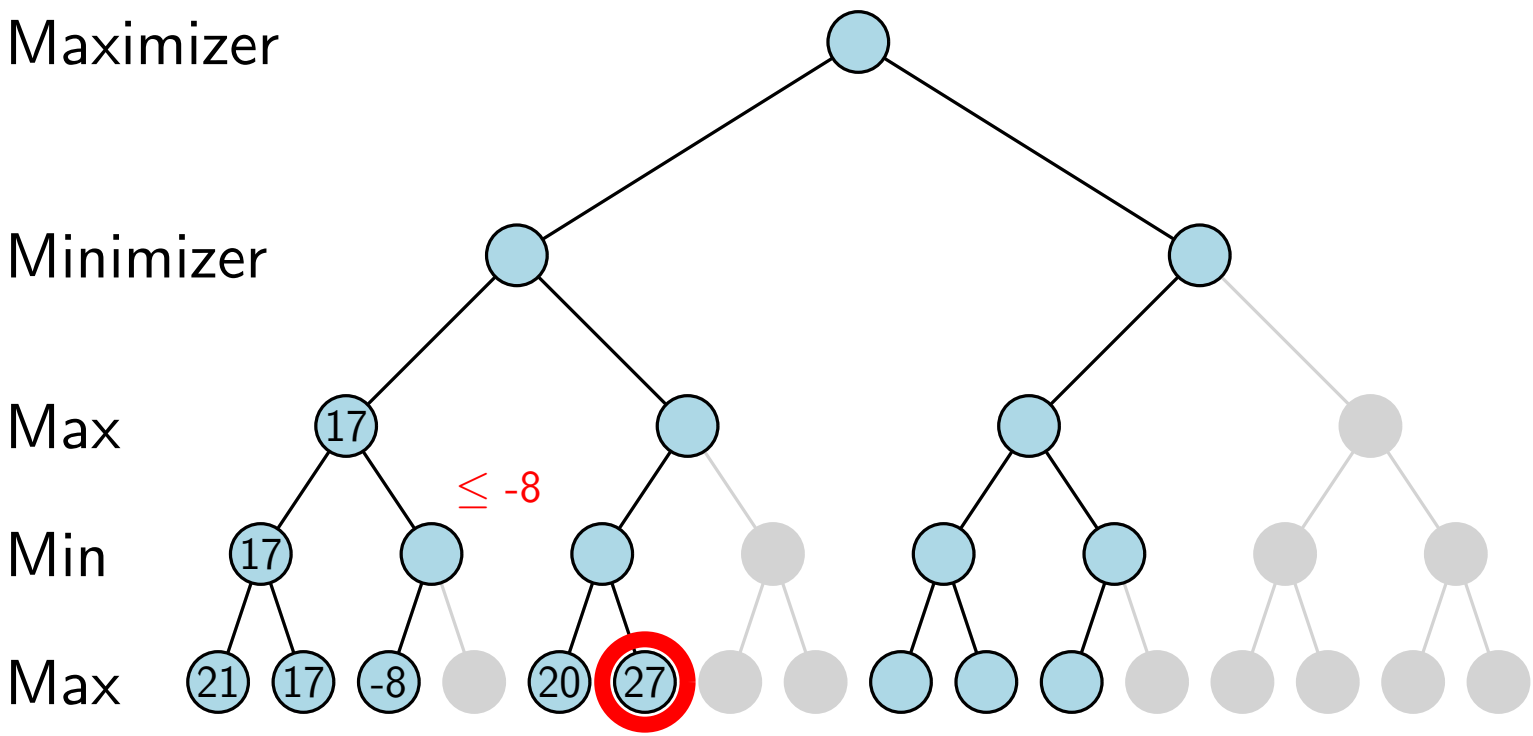
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



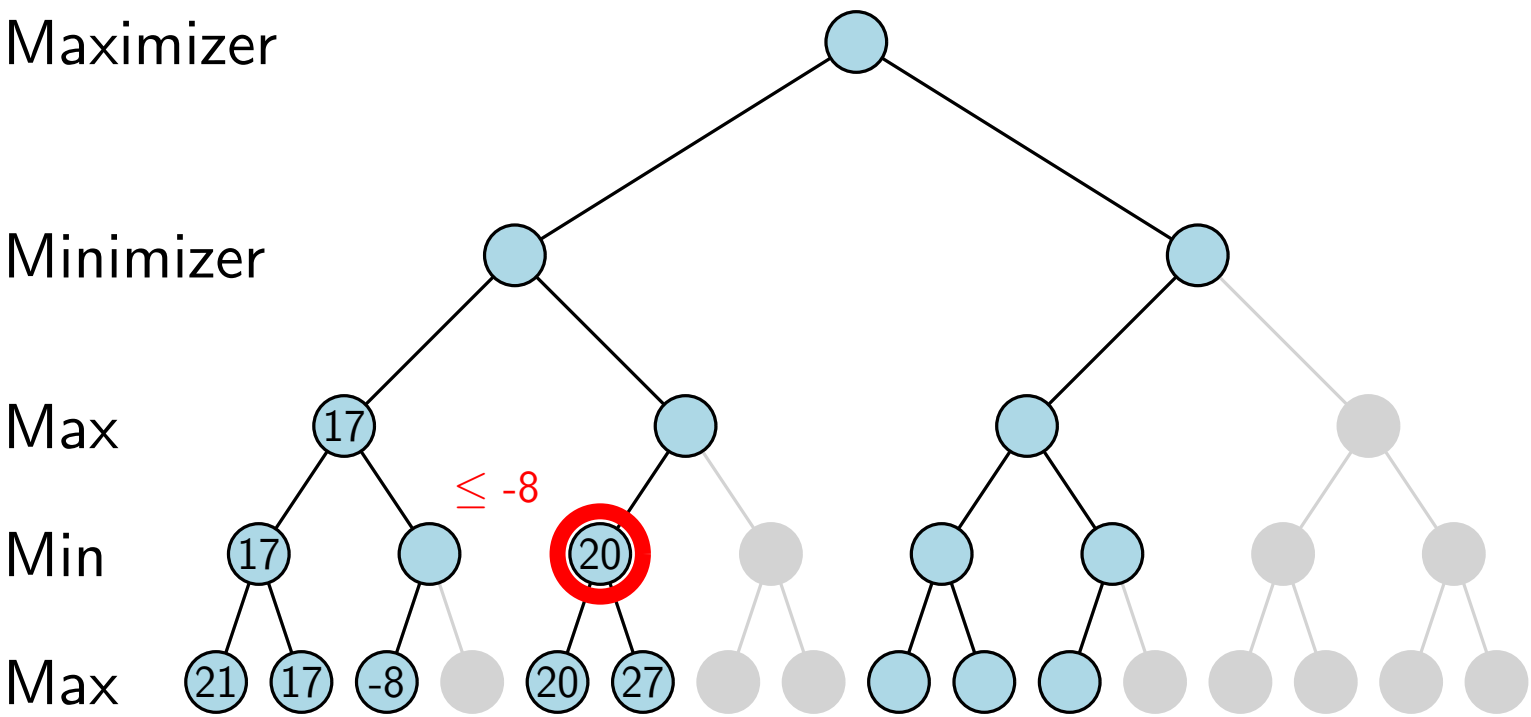
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



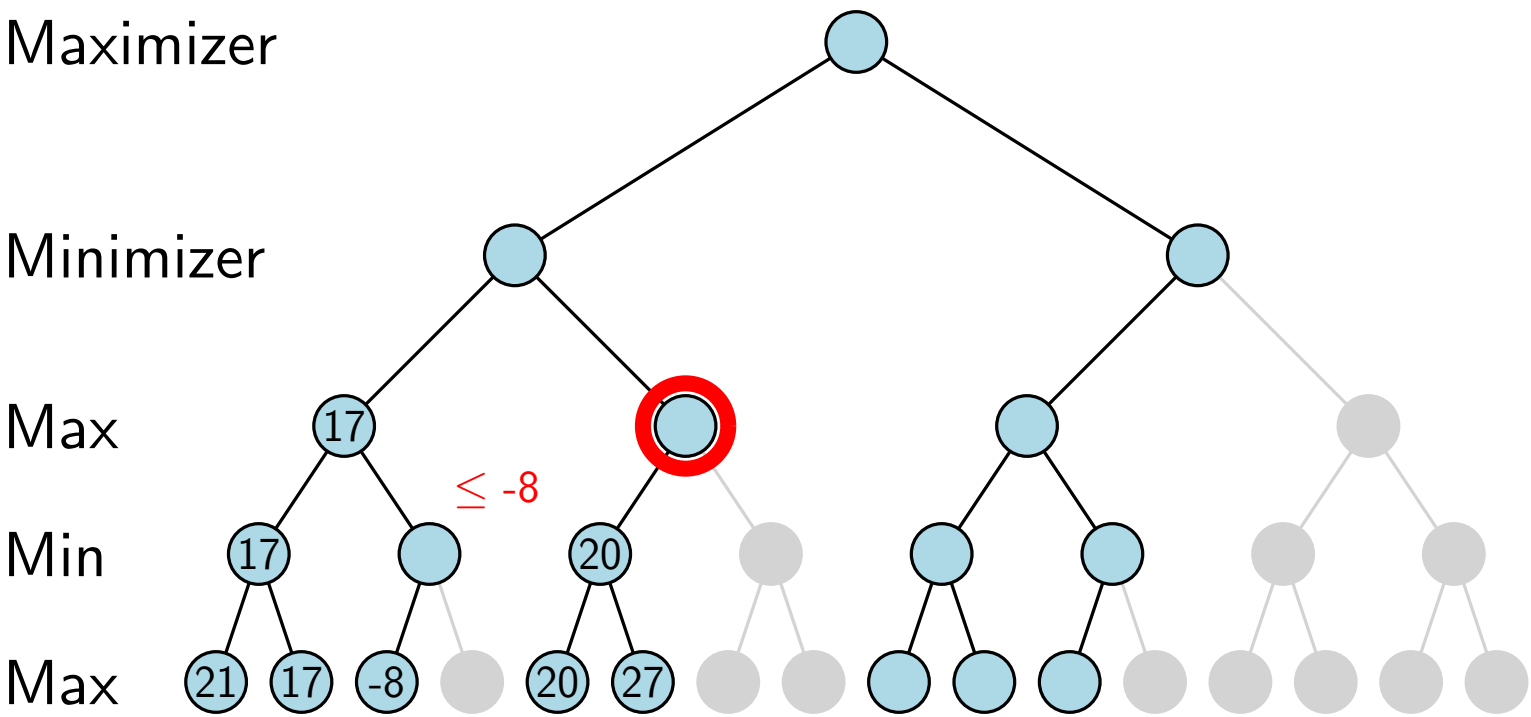
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



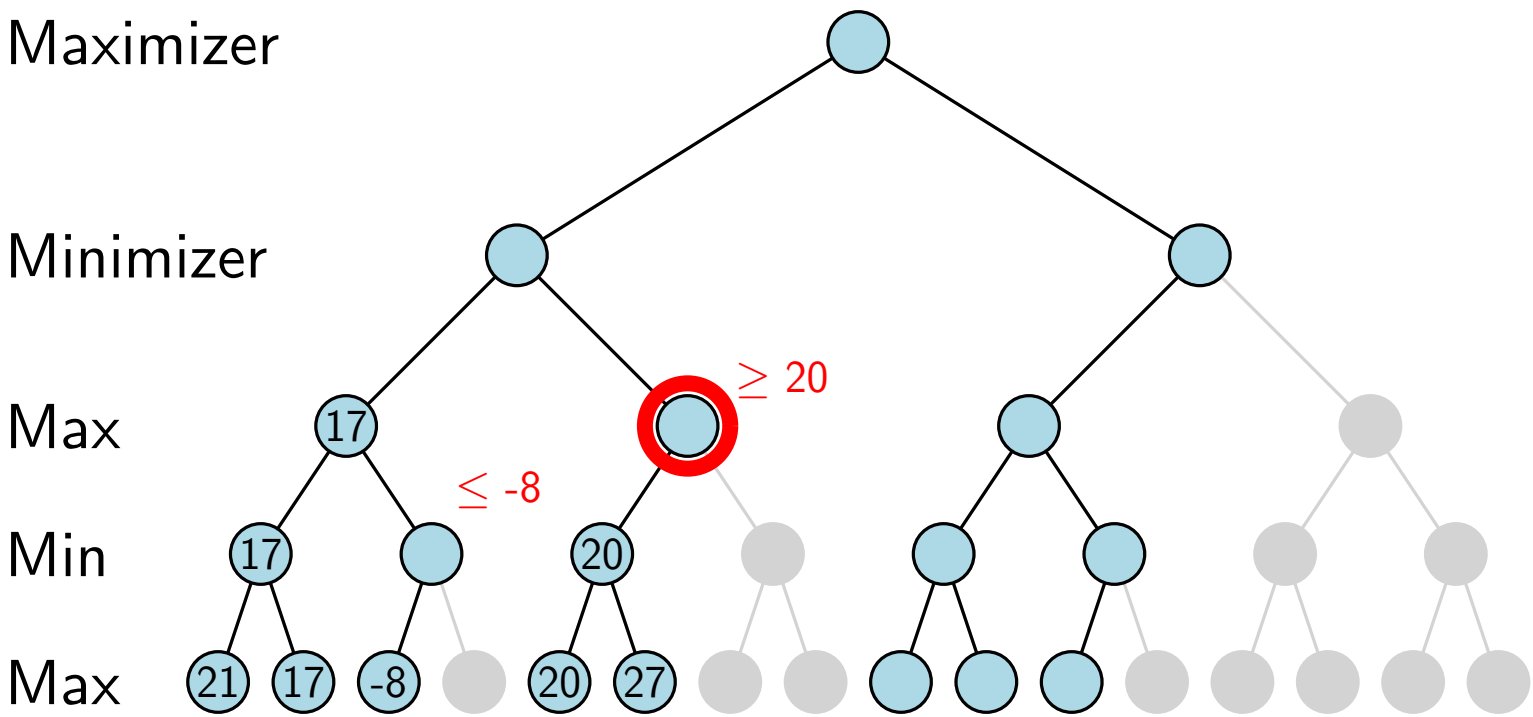
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



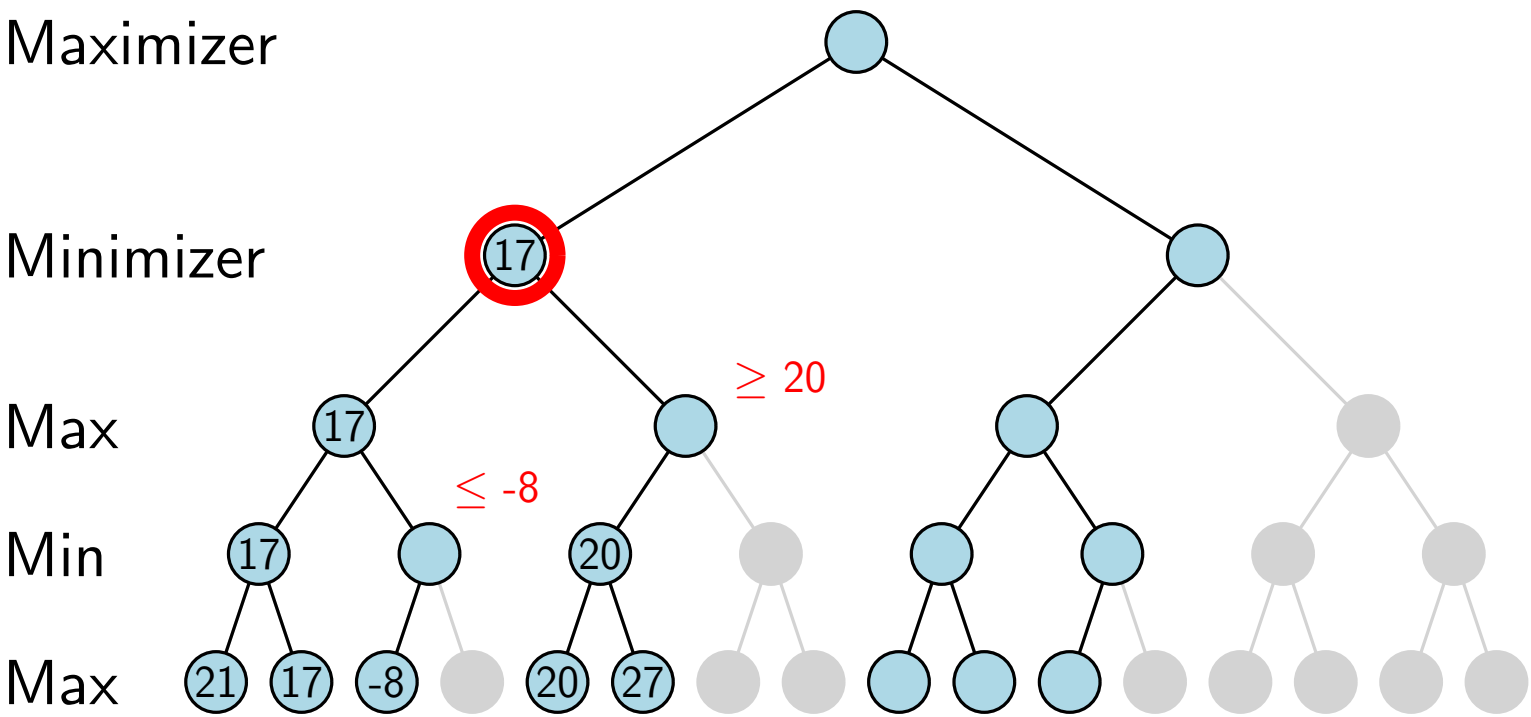
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



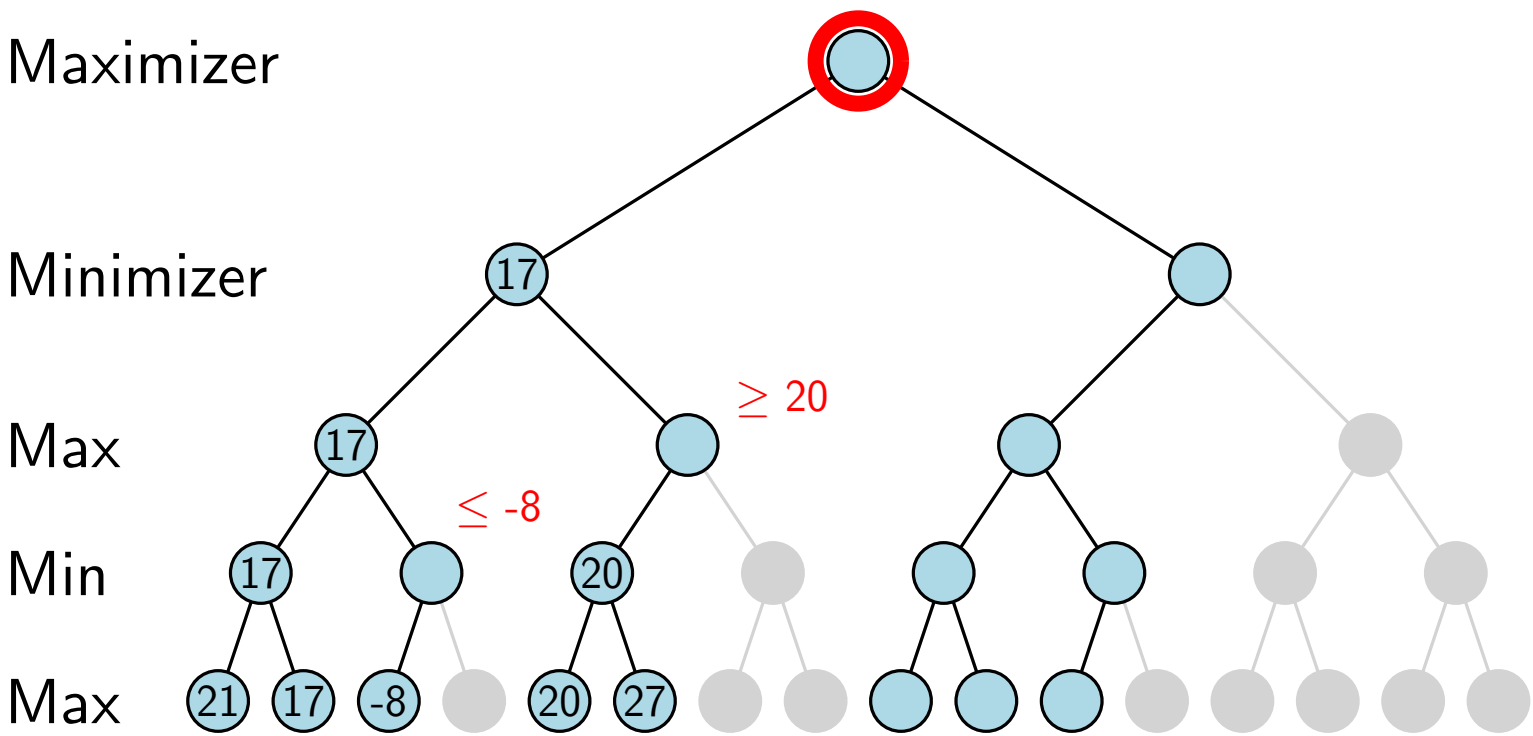
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



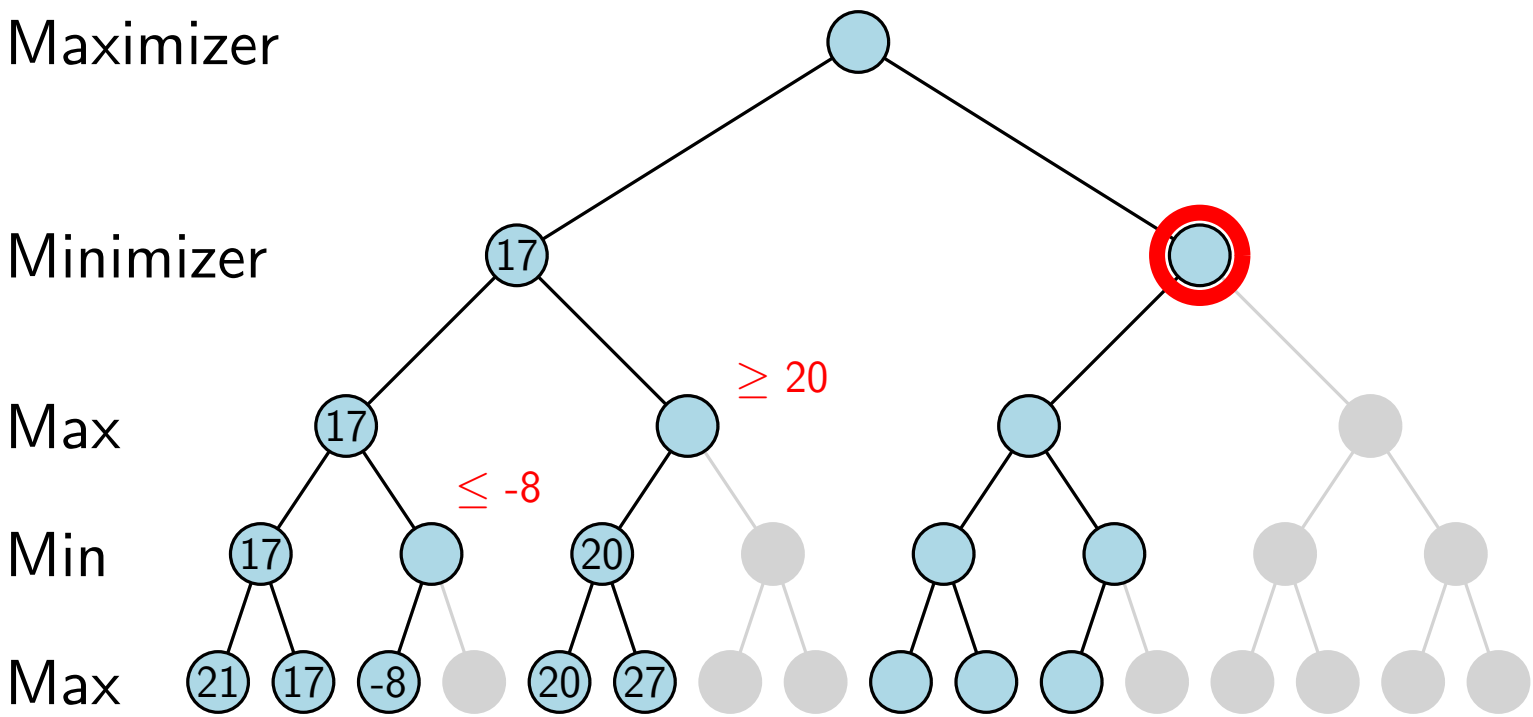
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



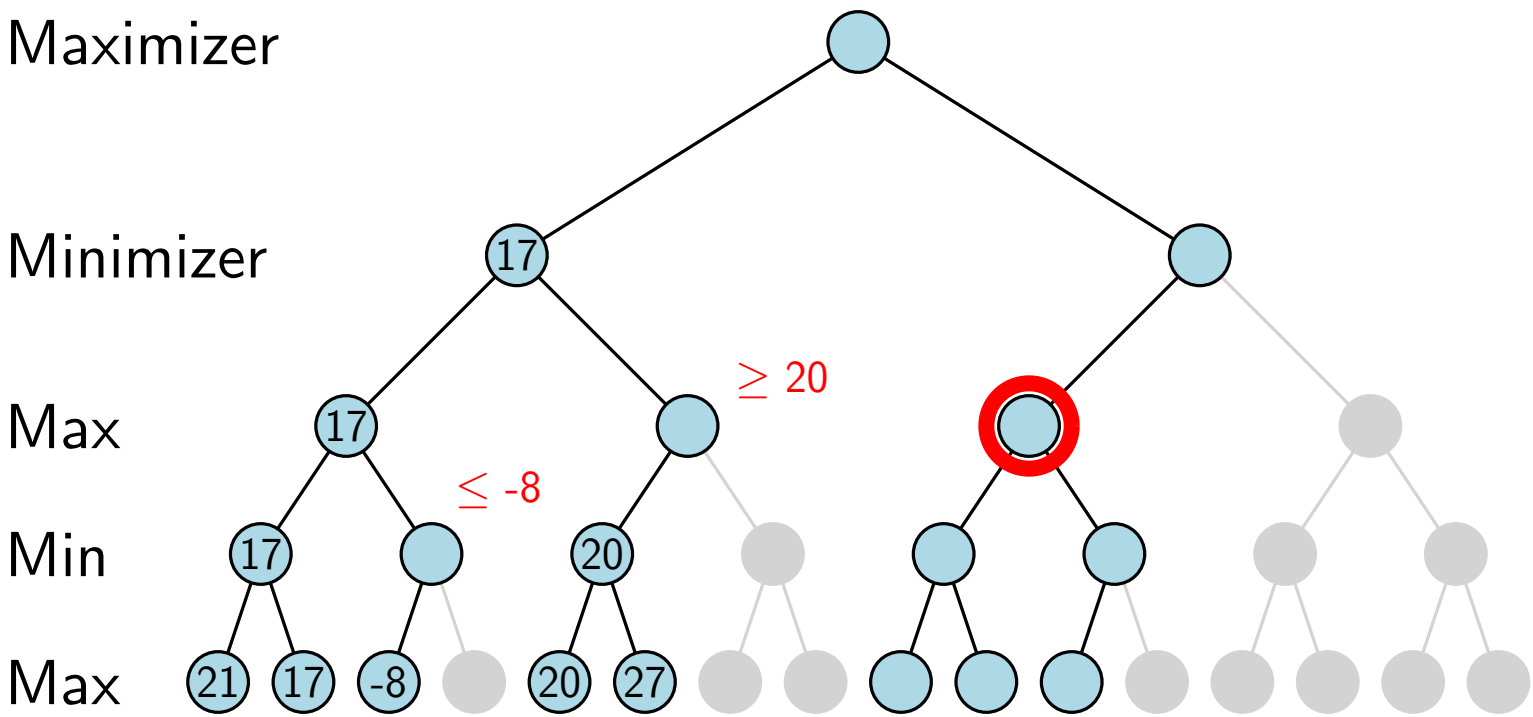
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



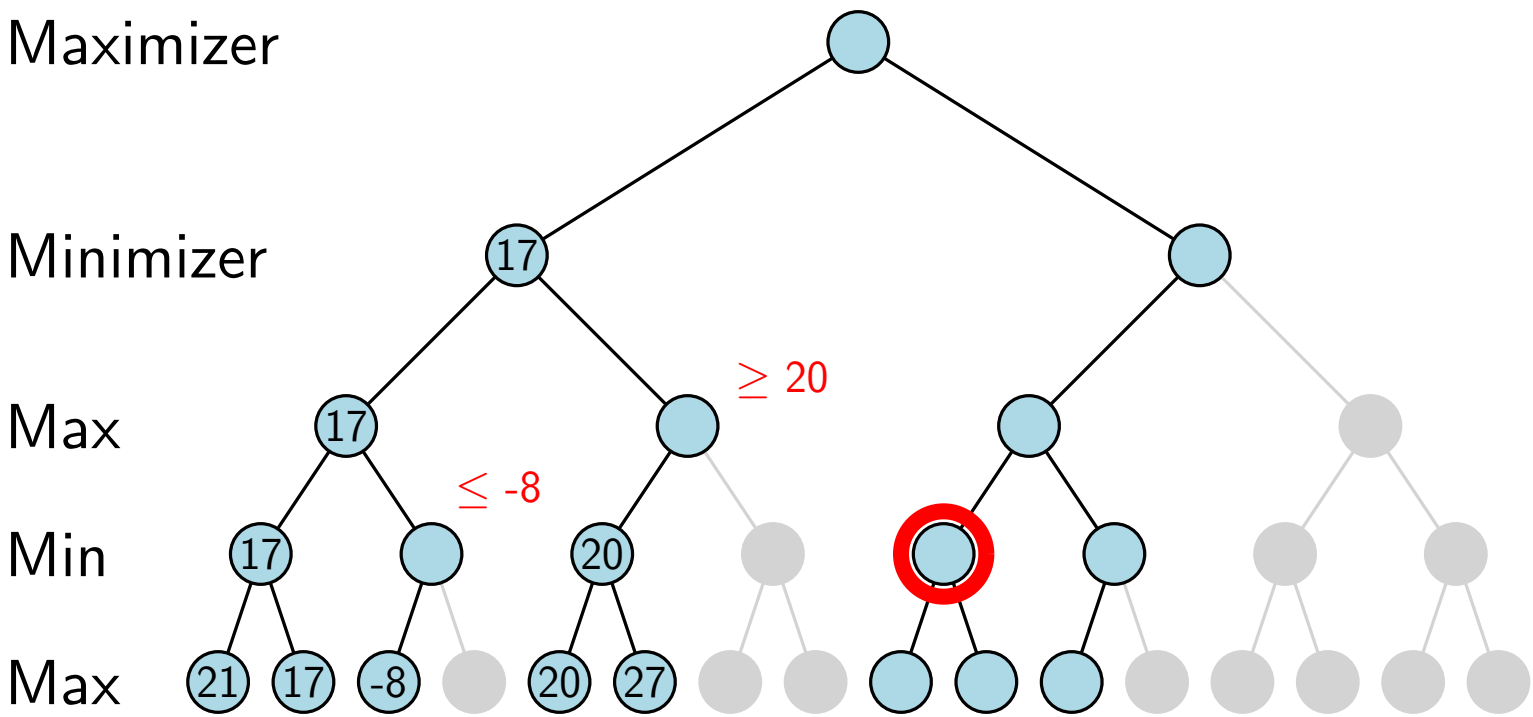
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



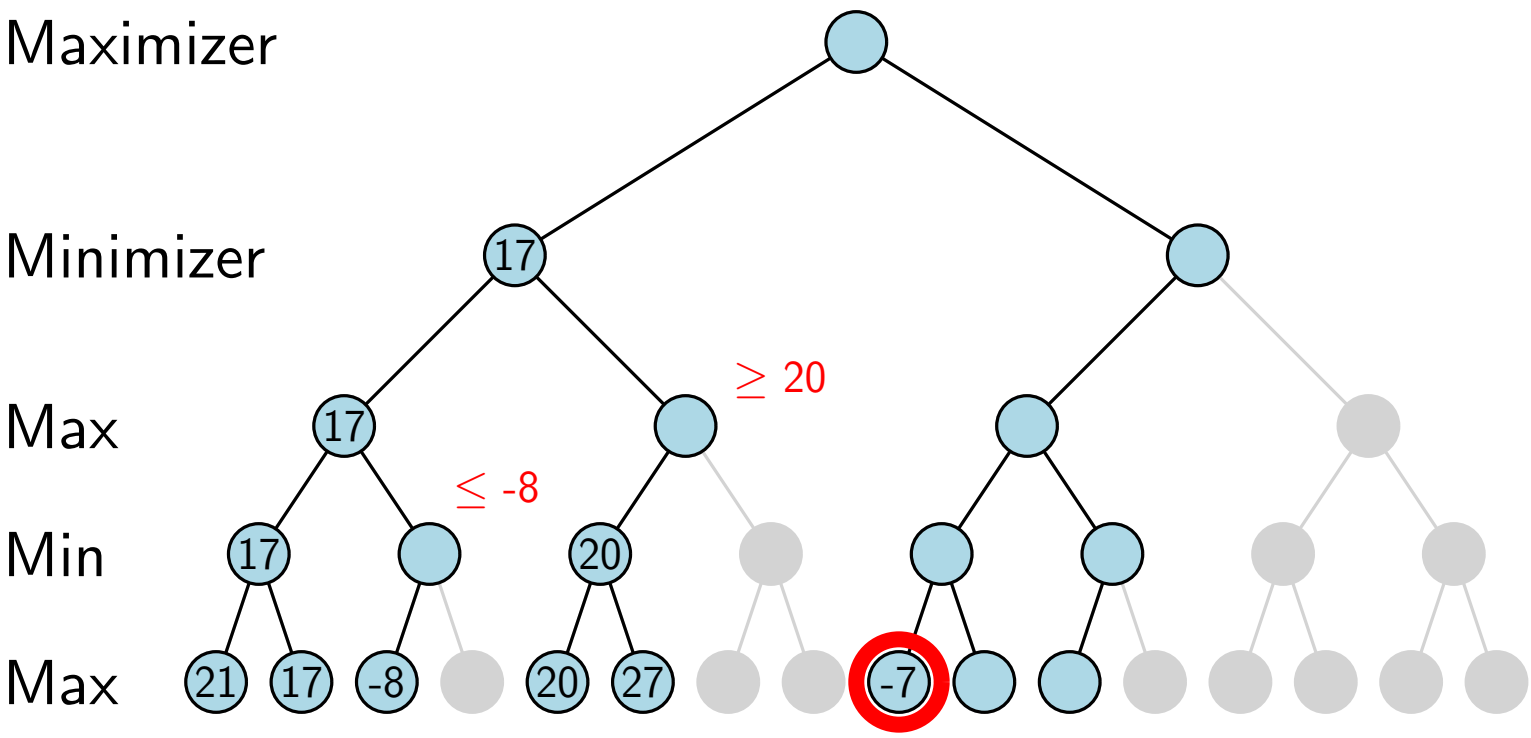
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



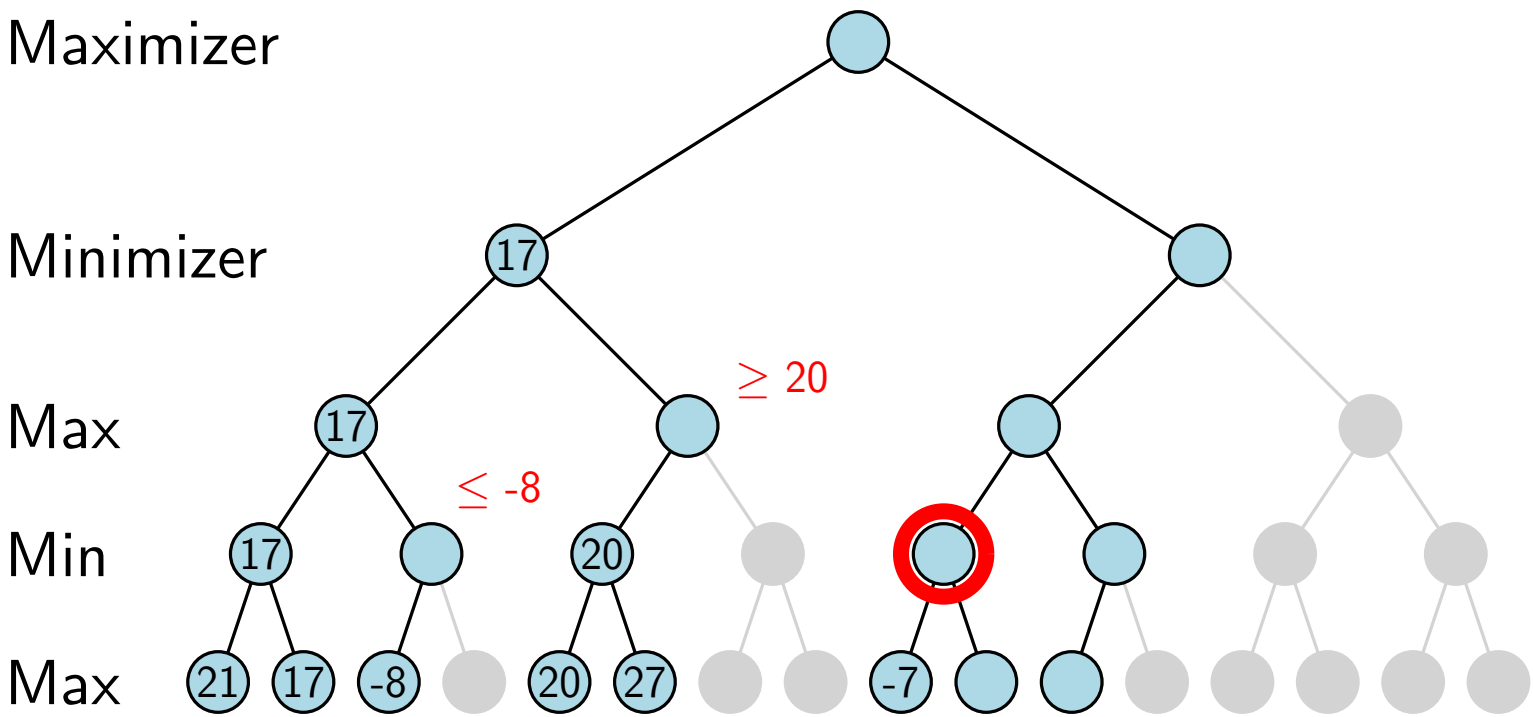
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



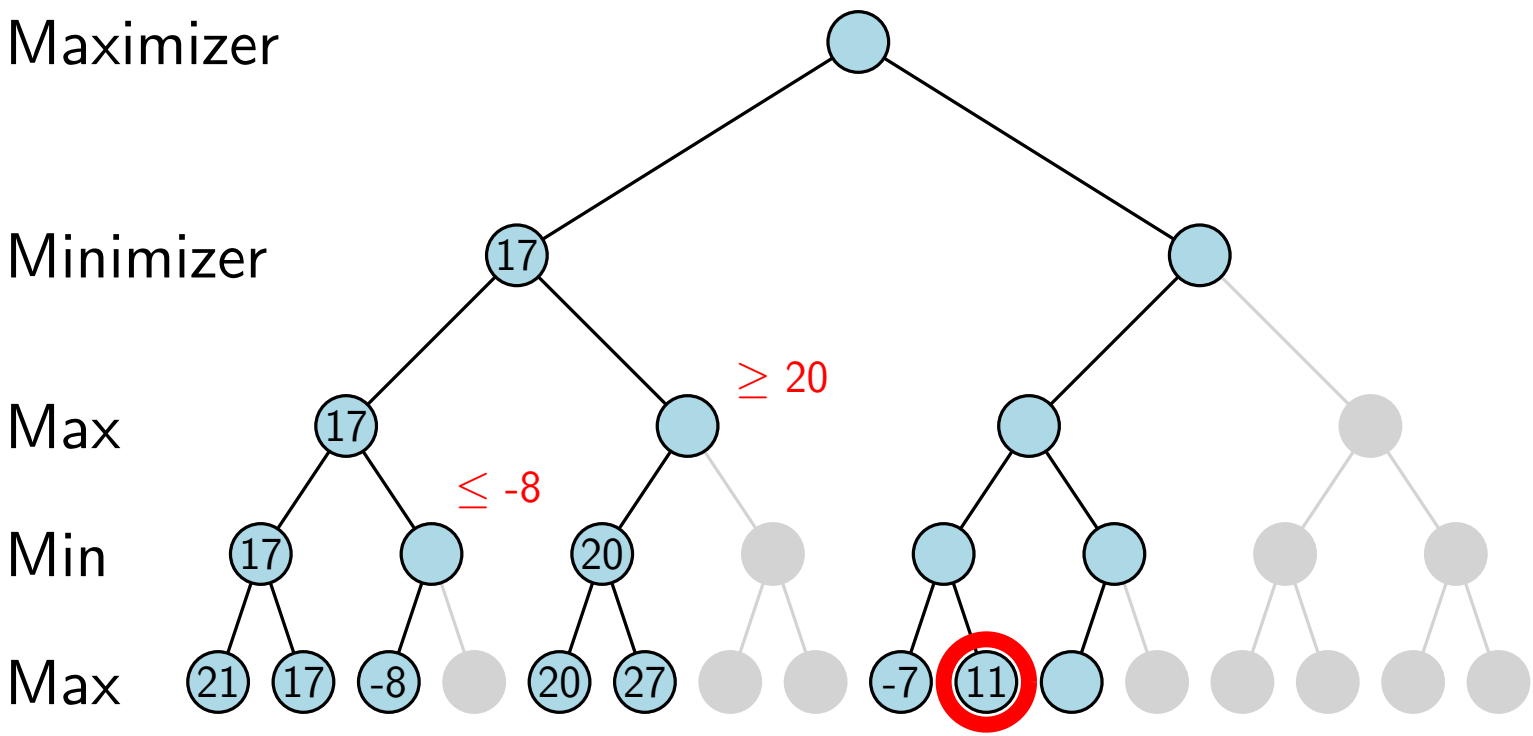
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



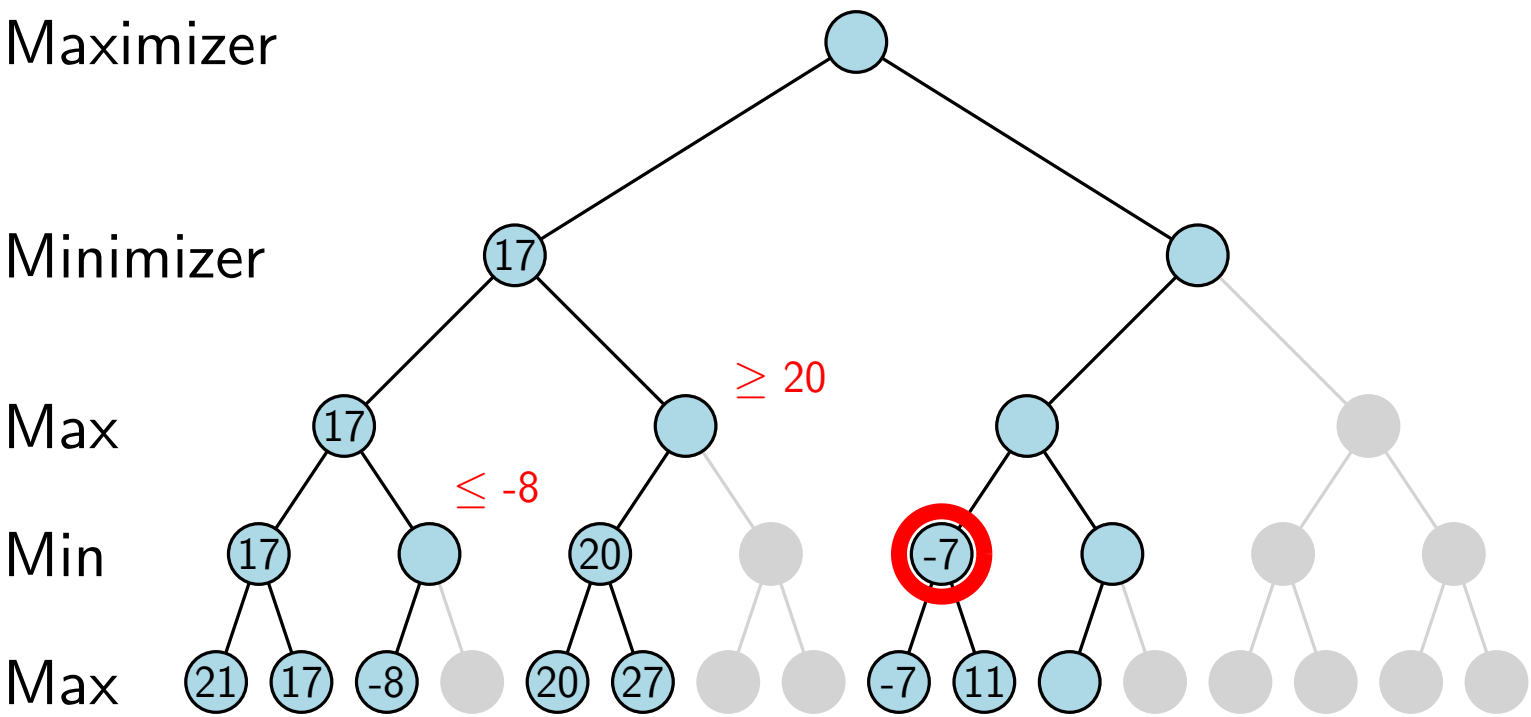
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



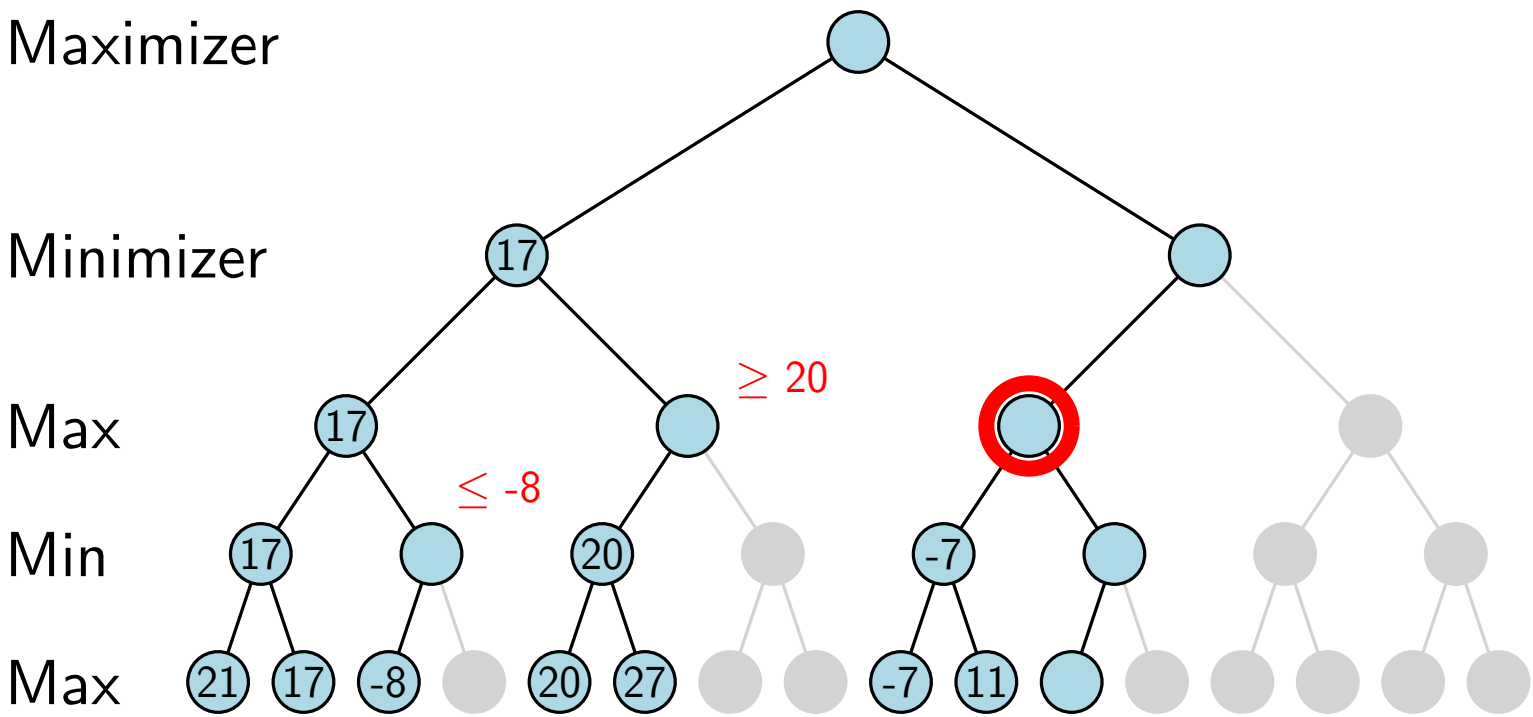
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



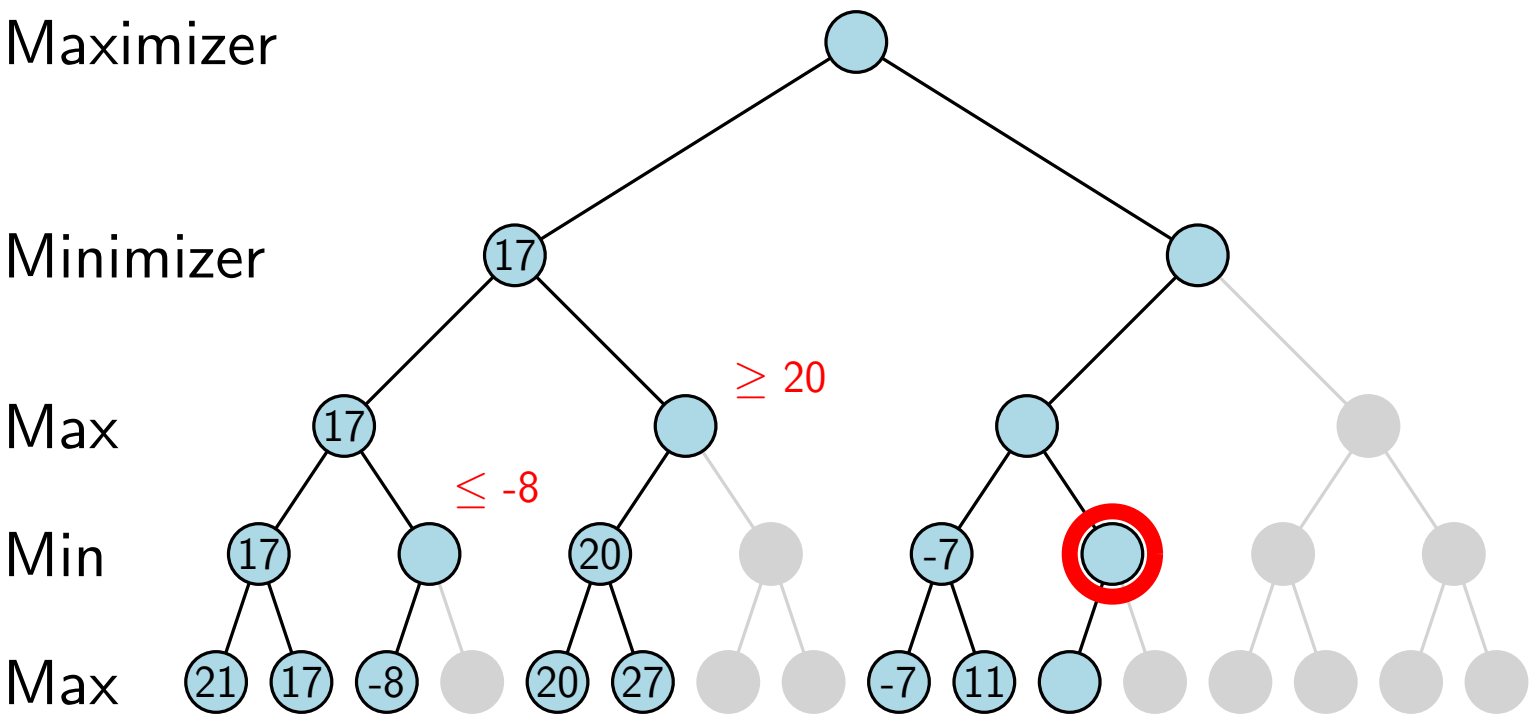
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



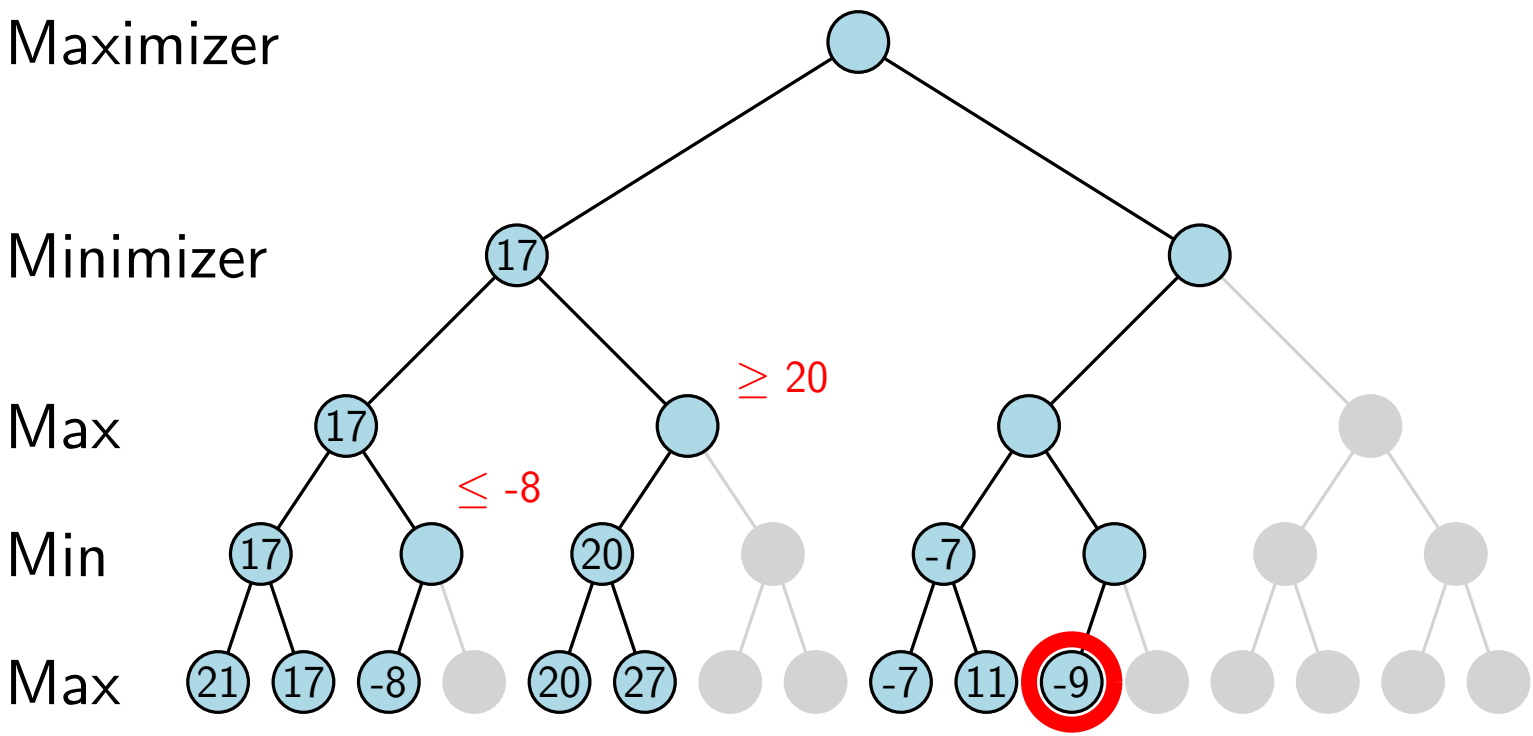
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



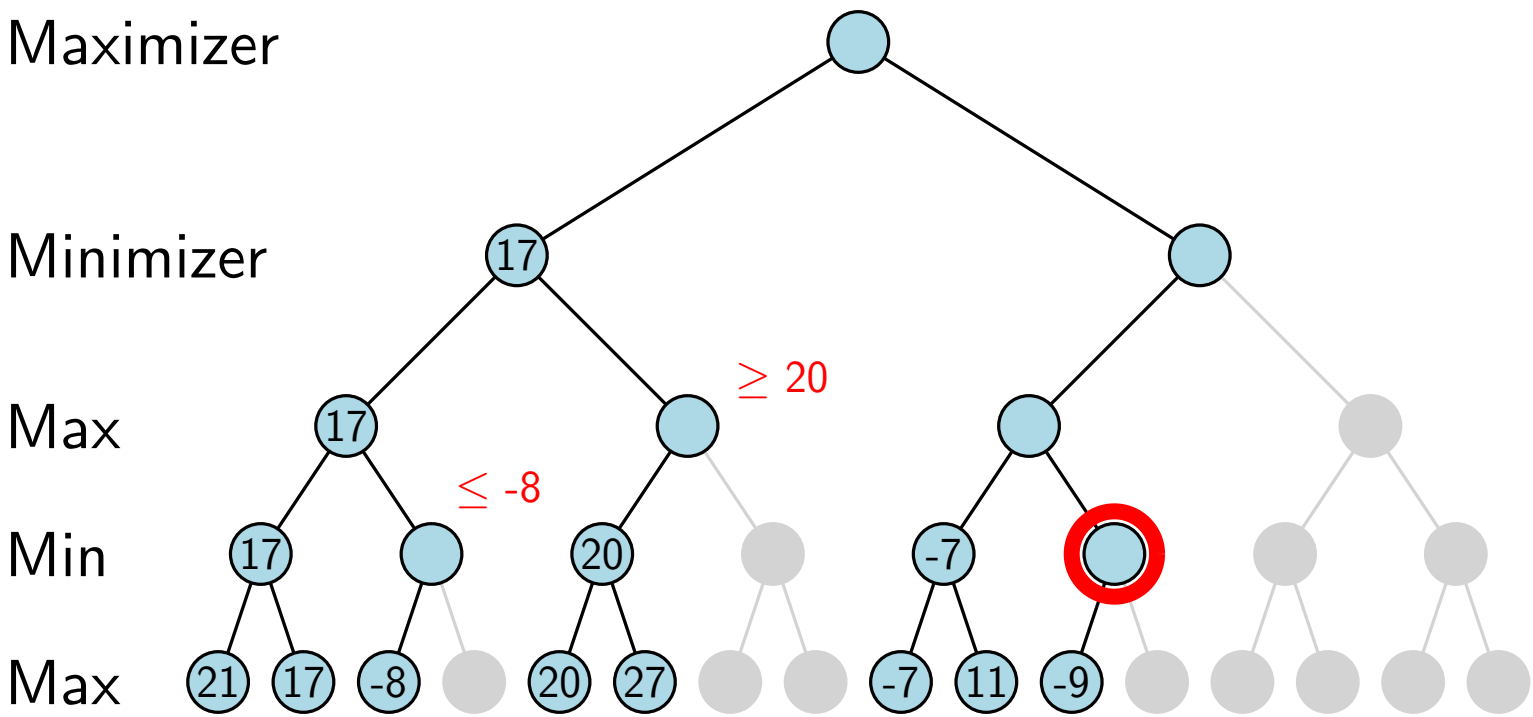
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



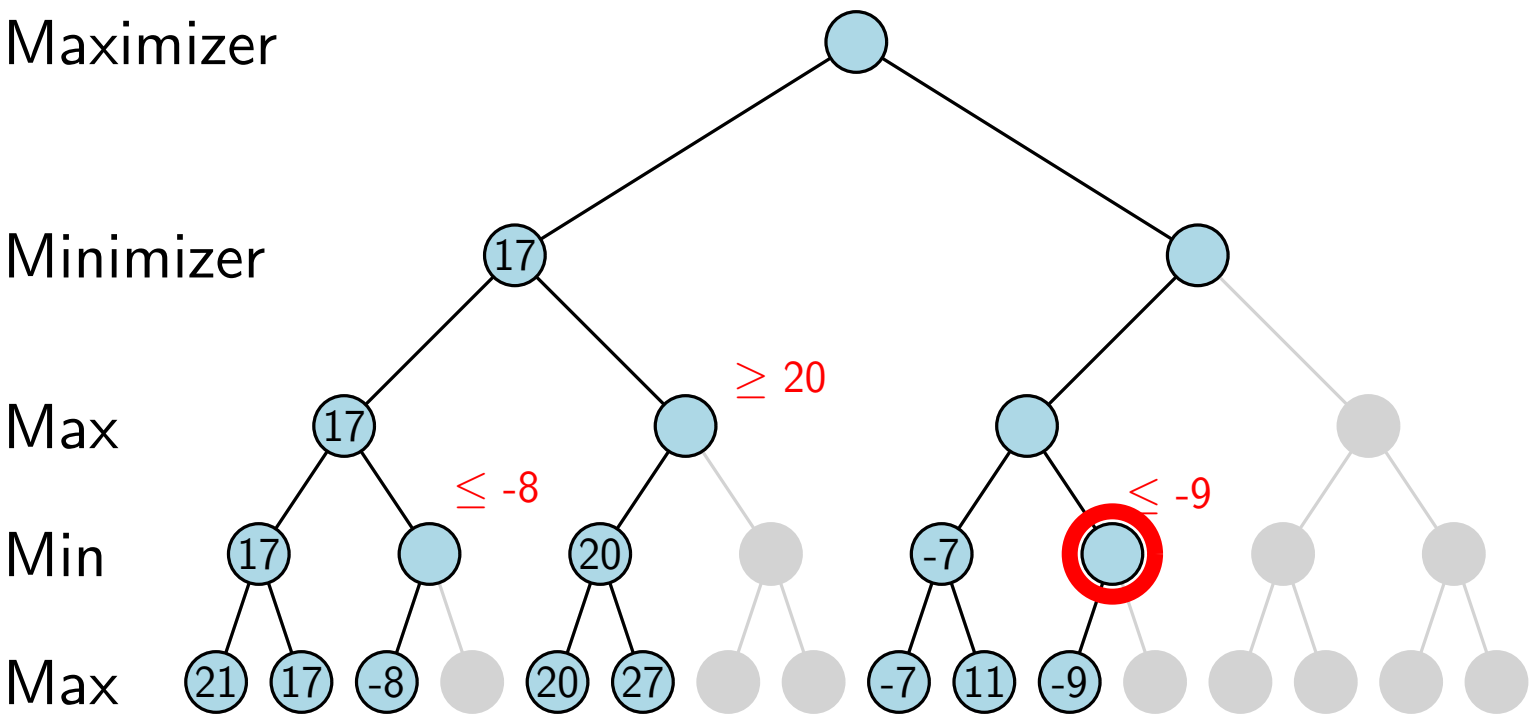
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



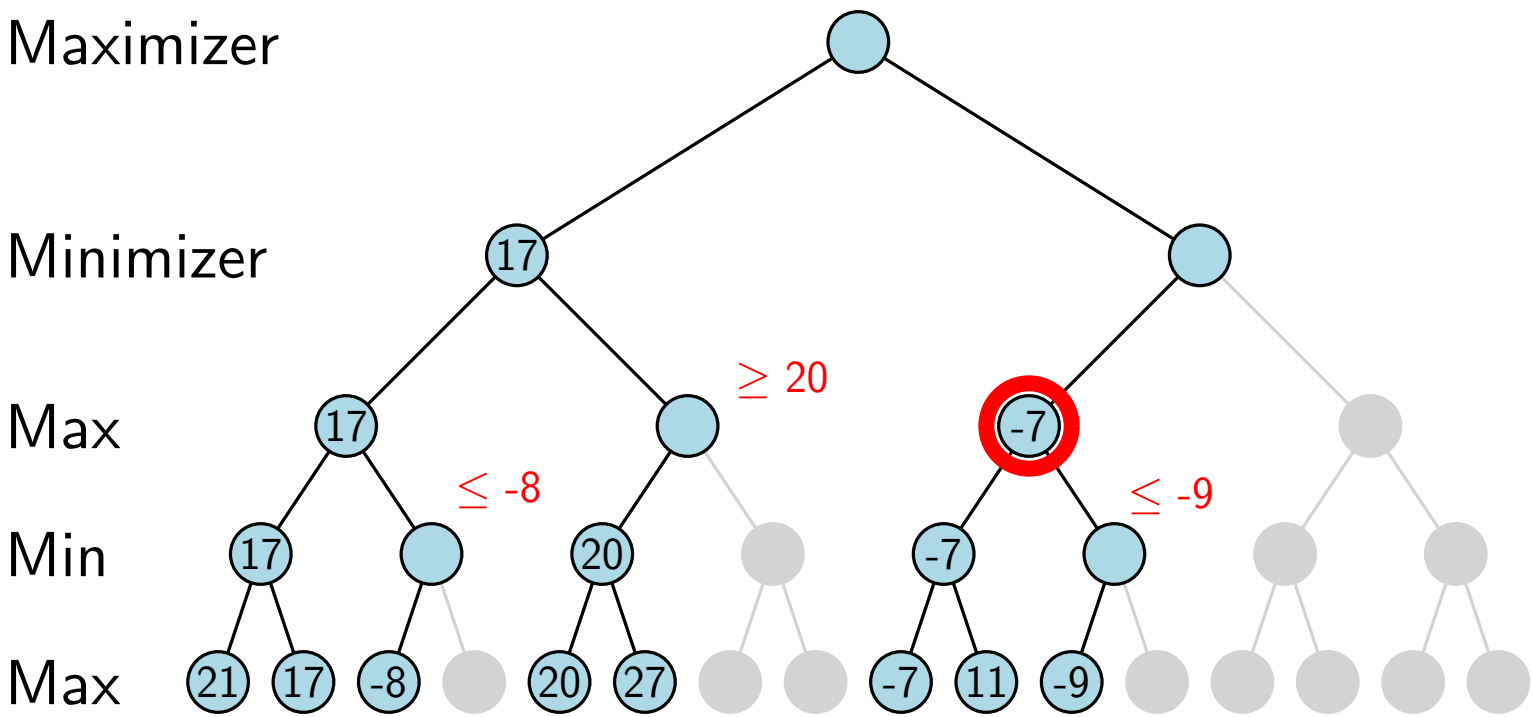
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



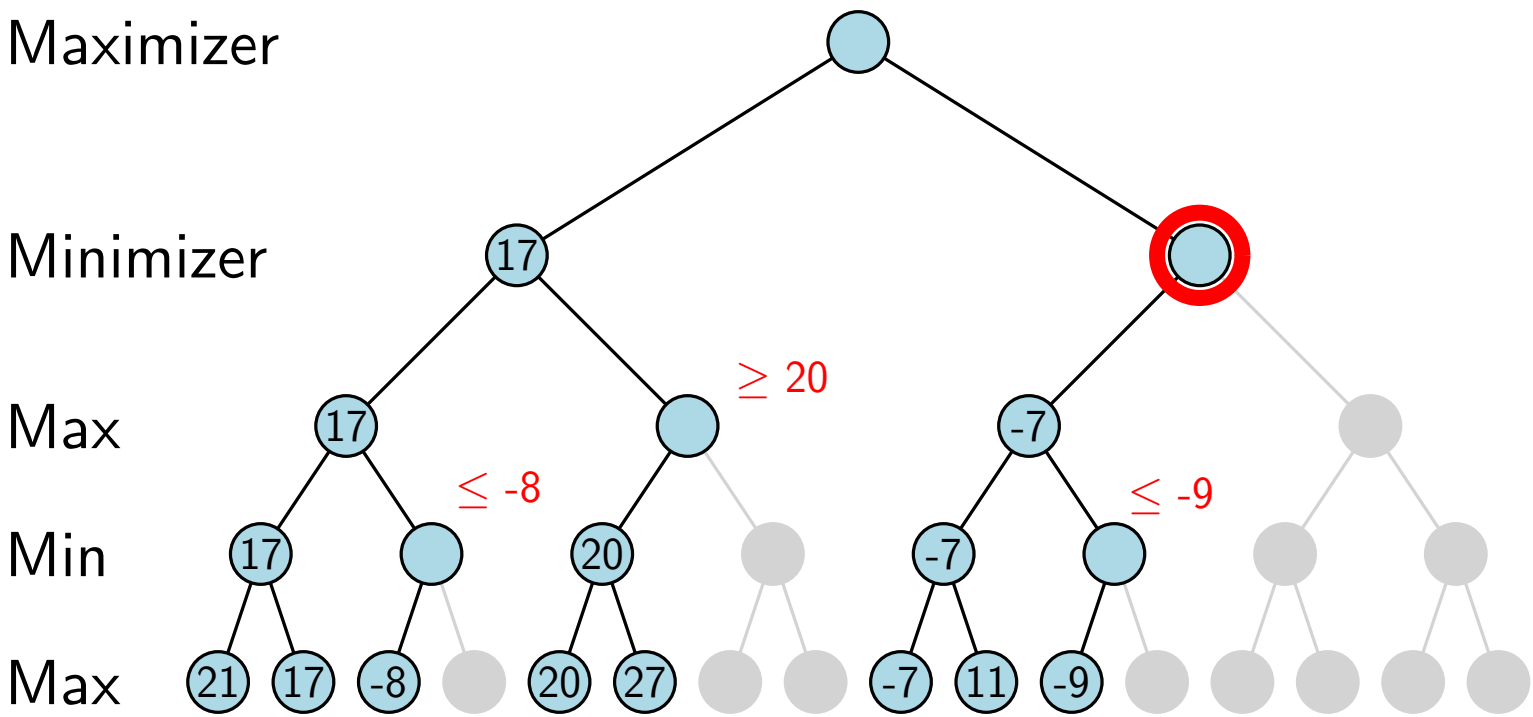
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



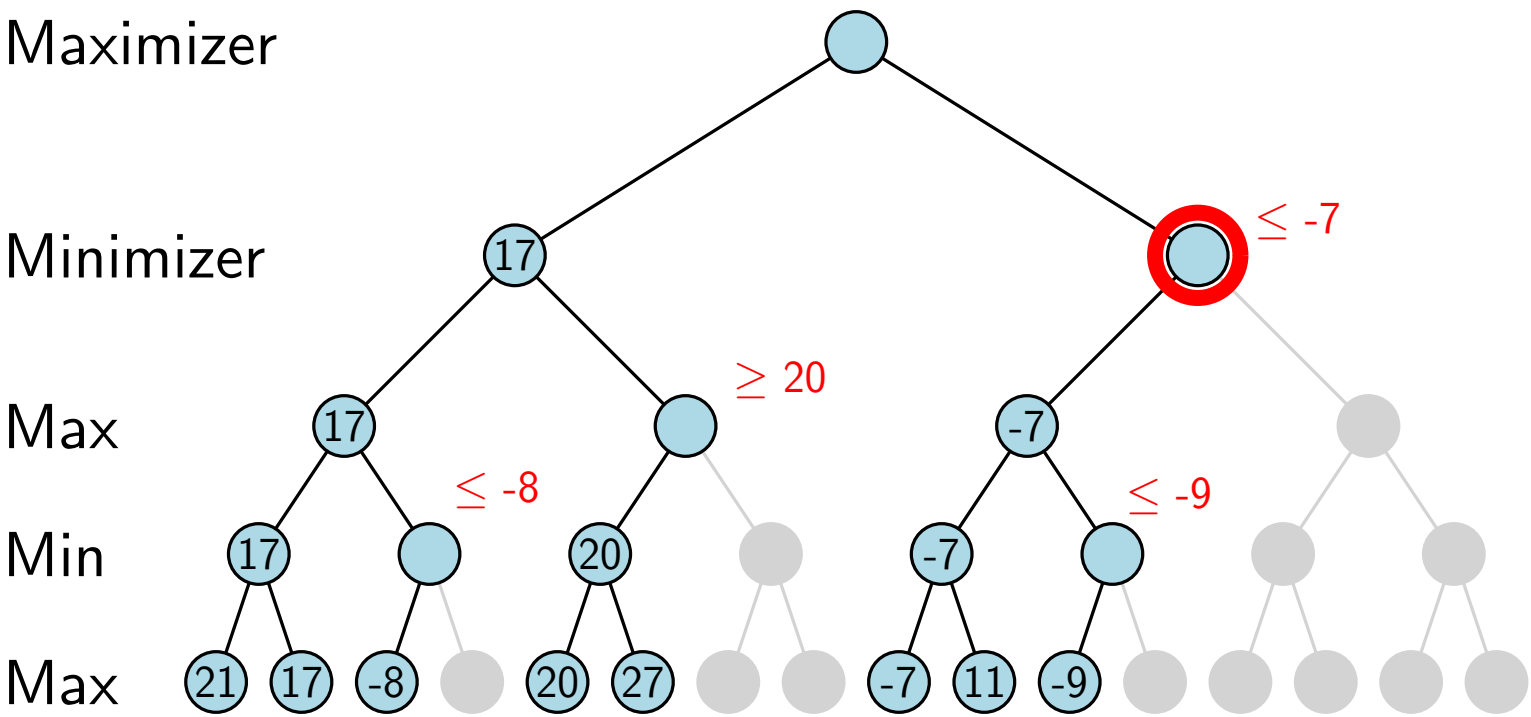
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



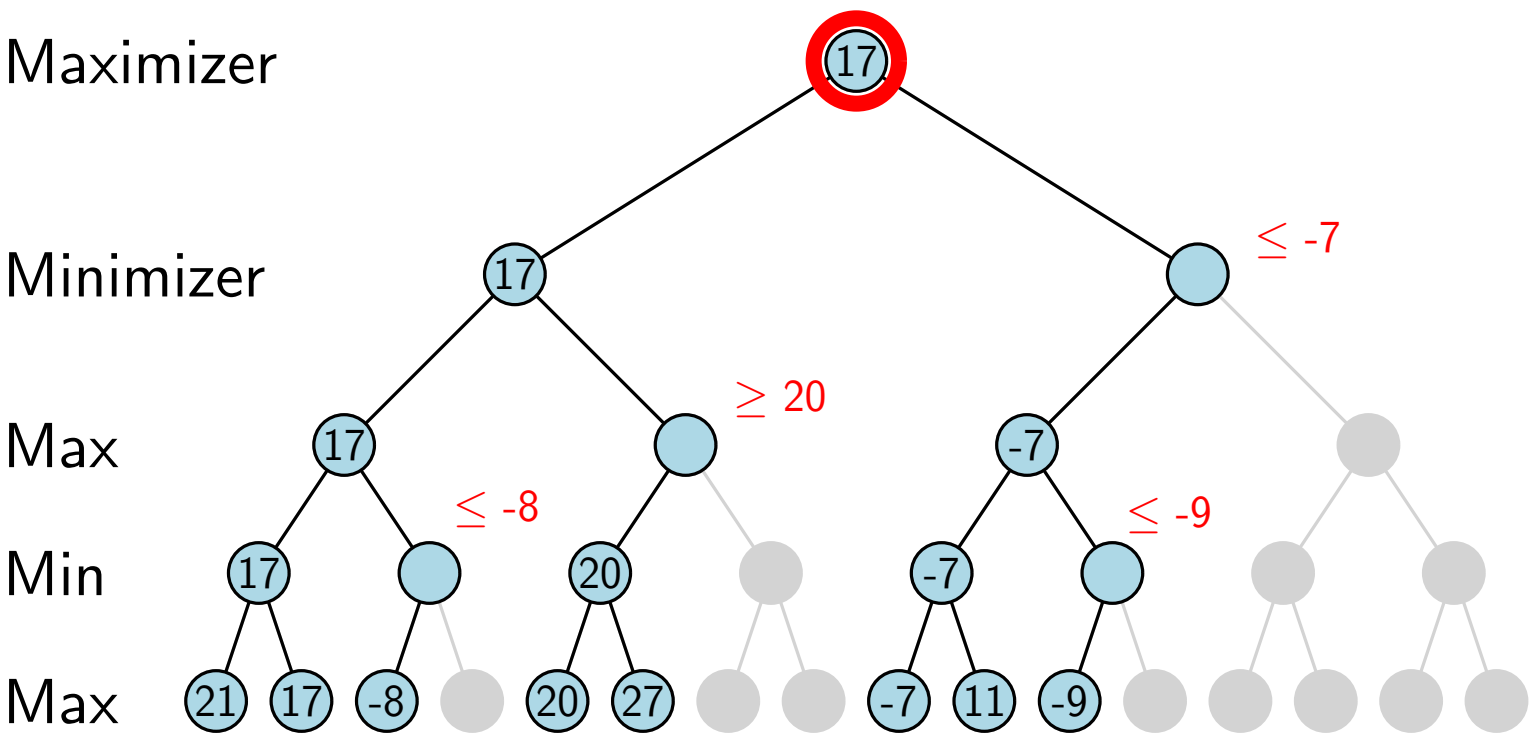
Game Tree: Min Max with α-β Pruning

α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



Game Tree: Min Max with α-β Pruning

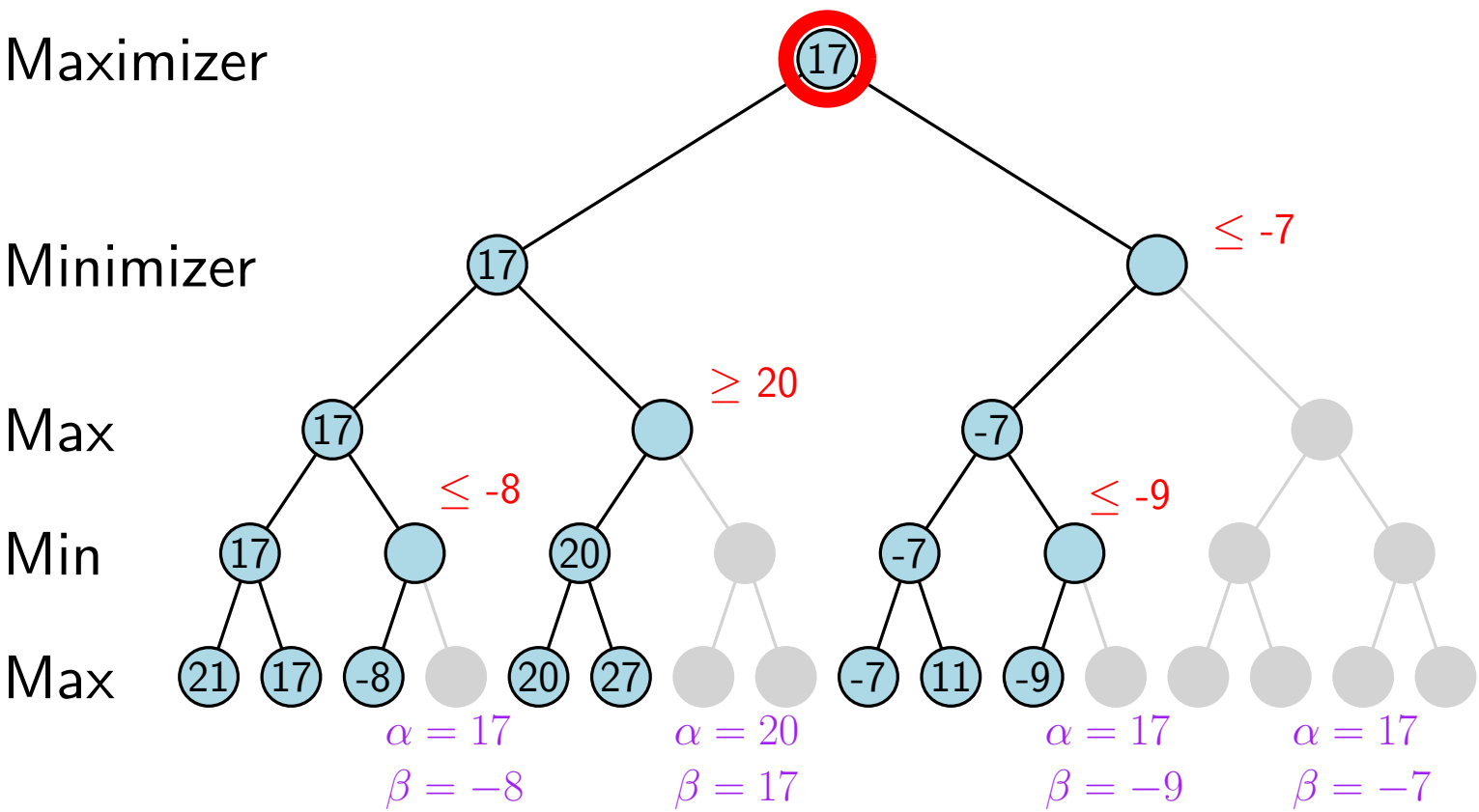
α-β pruning reduces the number of states to be visited, with exactly the same outcome. That is, although not visiting all nodes it does not miss any relevant information.



Game Tree: Min Max with α-β Pruning

α: Lower bound for maximizer: no need to consider states with scores below α.

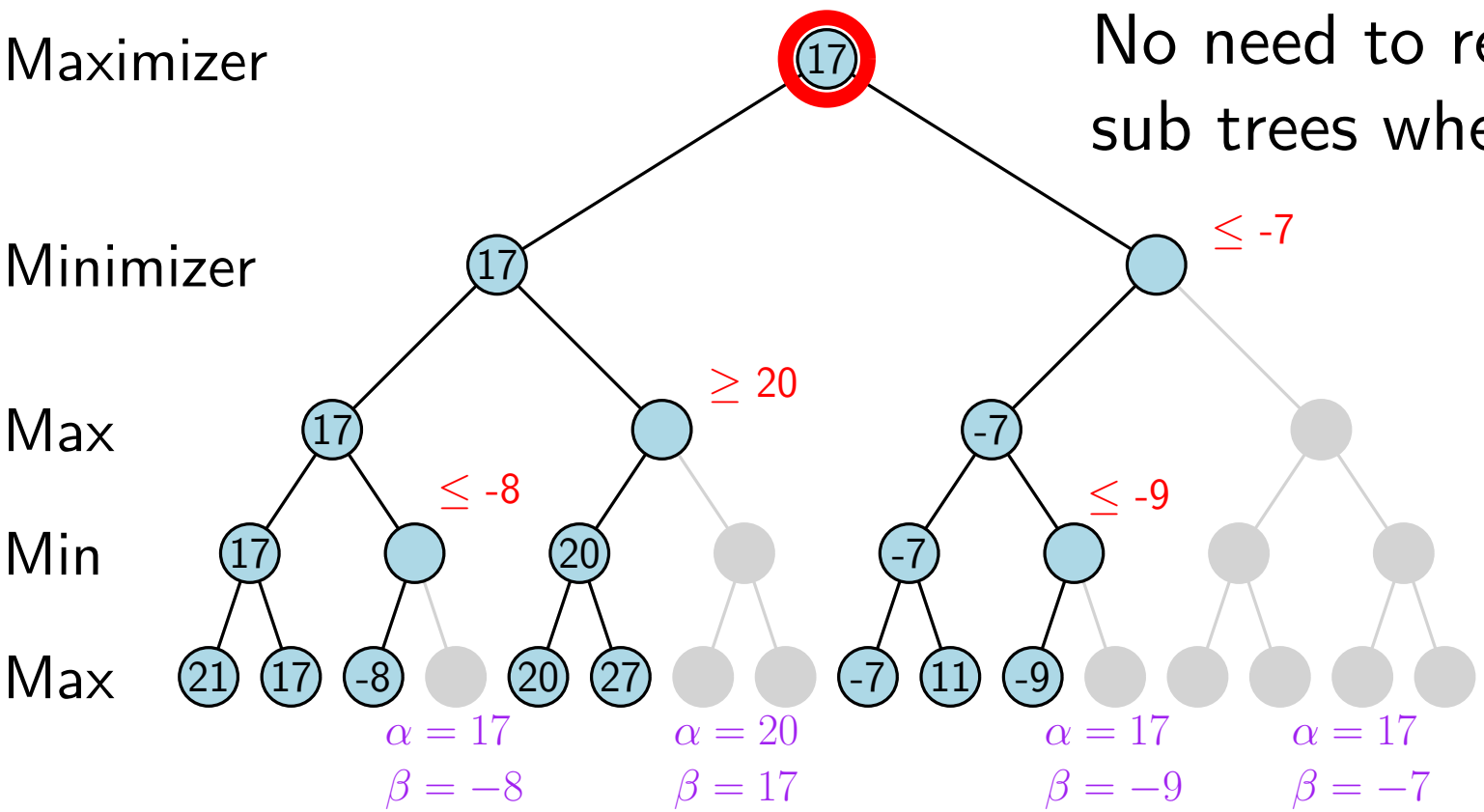
β: Upper bound for minimizer: no need to consider states with scores above β.



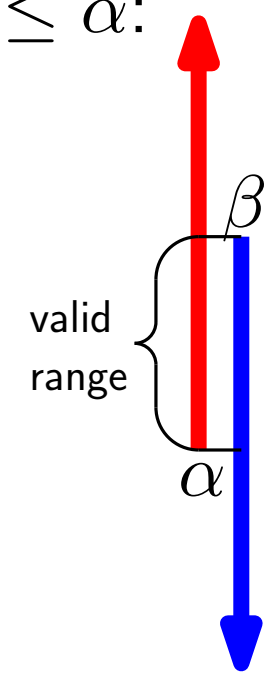
Game Tree: Min Max with α-β Pruning

α: Lower bound for maximizer: no need to consider states with scores below α.

β: Upper bound for minimizer: no need to consider states with scores above β.



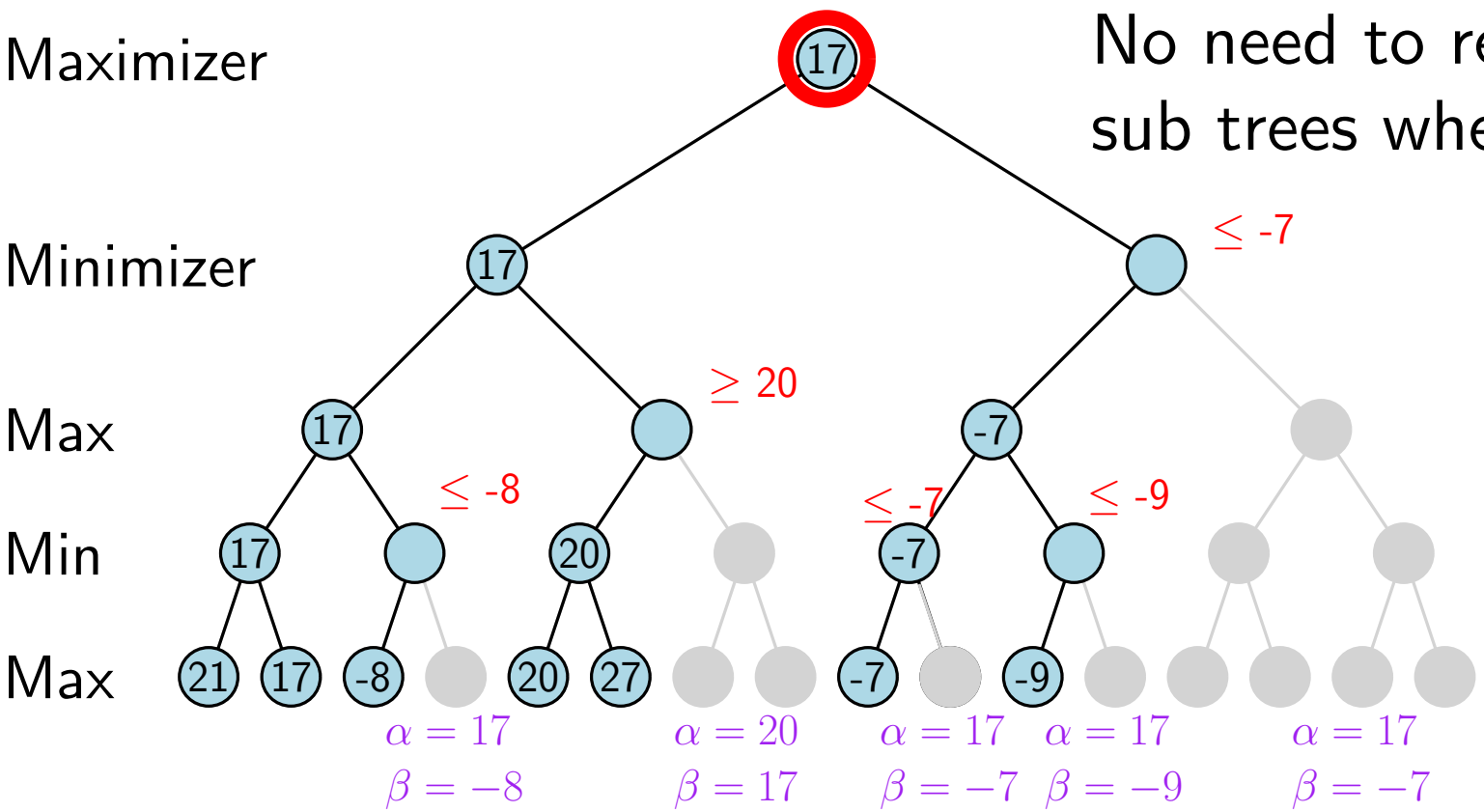
No need to recurse into sub trees when $\beta \leq \alpha$:



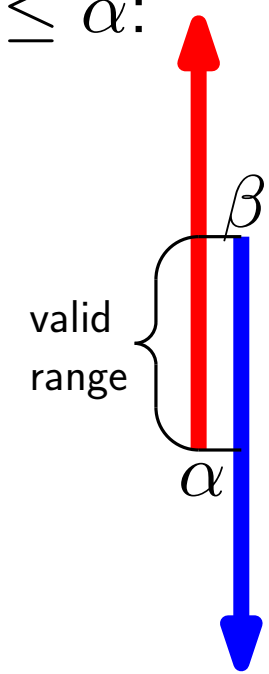
Game Tree: Min Max with α-β Pruning

α: Lower bound for maximizer: no need to consider states with scores below α.

β: Upper bound for minimizer: no need to consider states with scores above β.



No need to recurse into sub trees when $\beta \leq \alpha$:



Pseudo code for Min Max with α-β Pruning

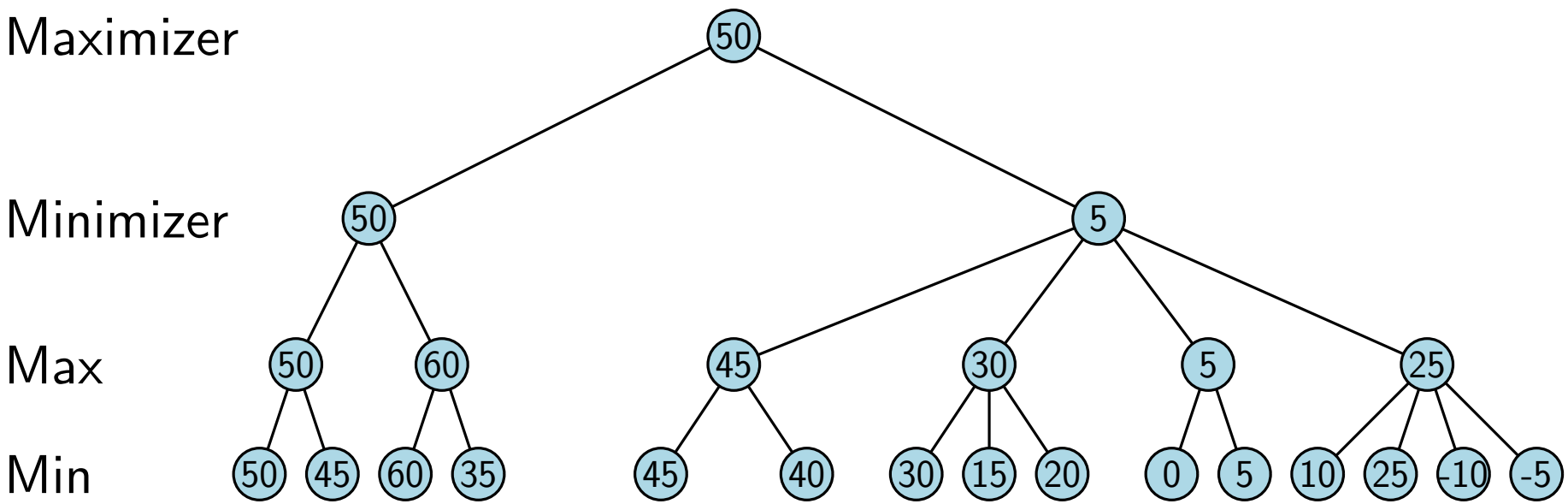
```
evaluate (node, alpha, beta)
    if node is a leaf
       return (heuristic value of node)
    if node is a minimizing node
       for each child of node
          beta = min (beta, evaluate (child, alpha, beta))
          if beta <= alpha
             return (alpha)
       return (beta)
    if node is a maximizing node
       for each child of node
          alpha = max (alpha, evaluate (child, alpha, beta))
          if beta <= alpha
             return (beta)
       return (alpha)
evaluate(root,−∞,∞);
```

- α: Lower bound for maximizer: no need to consider states with scores below α.
- β: Upper bound for minimizer: no need to consider states with scores above β.
- No need to recurse into sub trees when $\beta \leq \alpha$:

Higher degree trees: α-β pruning

- Game trees usual have higher degree

Example 2

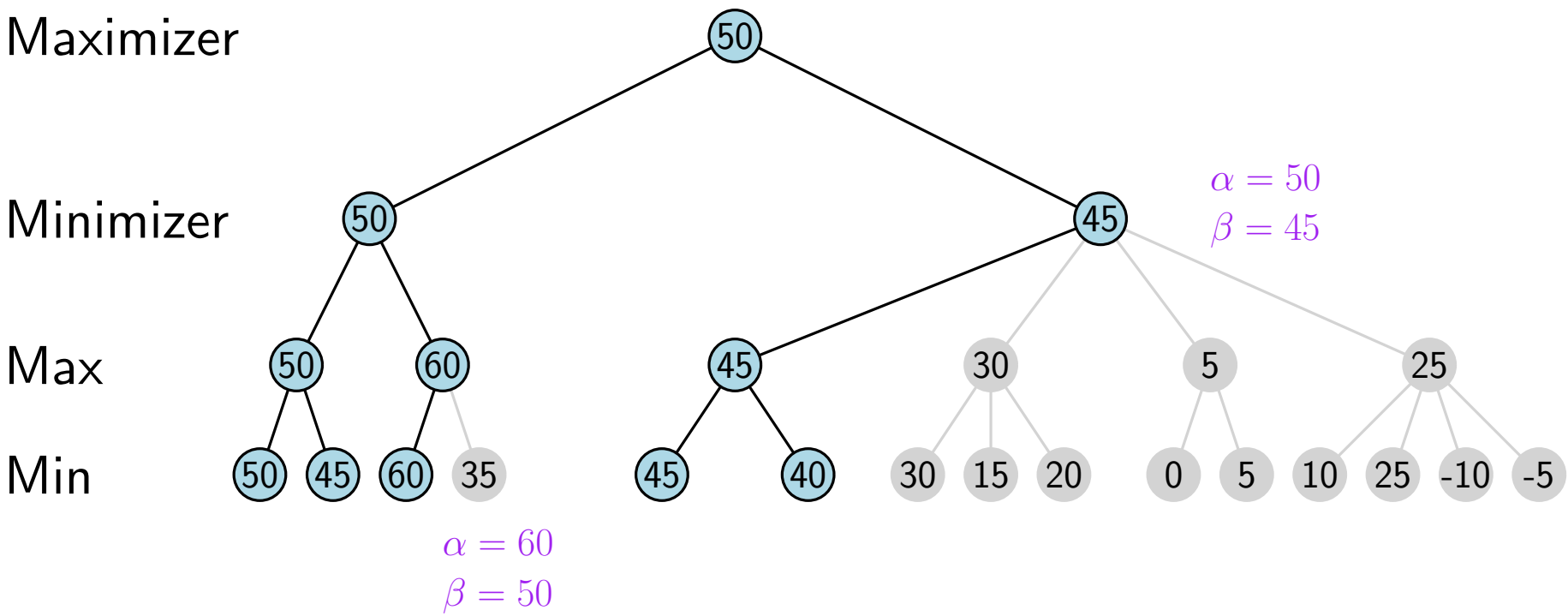


How much can we save with α-β pruning?

Higher degree trees: α-β pruning

- Game trees usual have higher degree

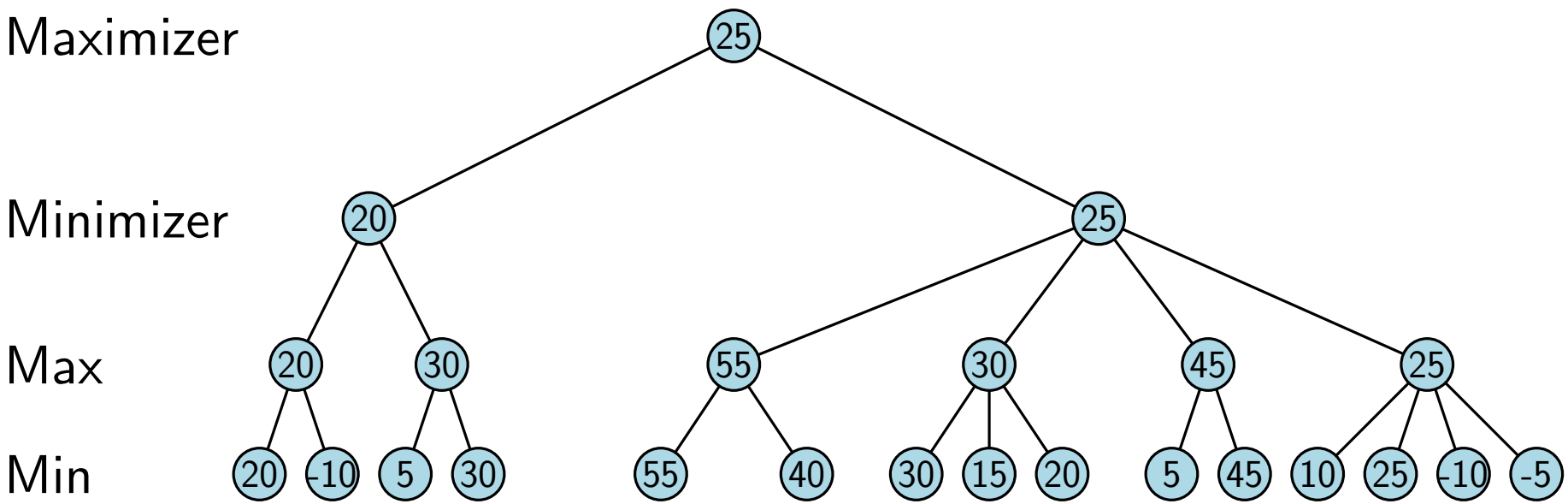
Example 2



Higher degree trees: α-β pruning

- Game trees usual have higher degree

Example 3

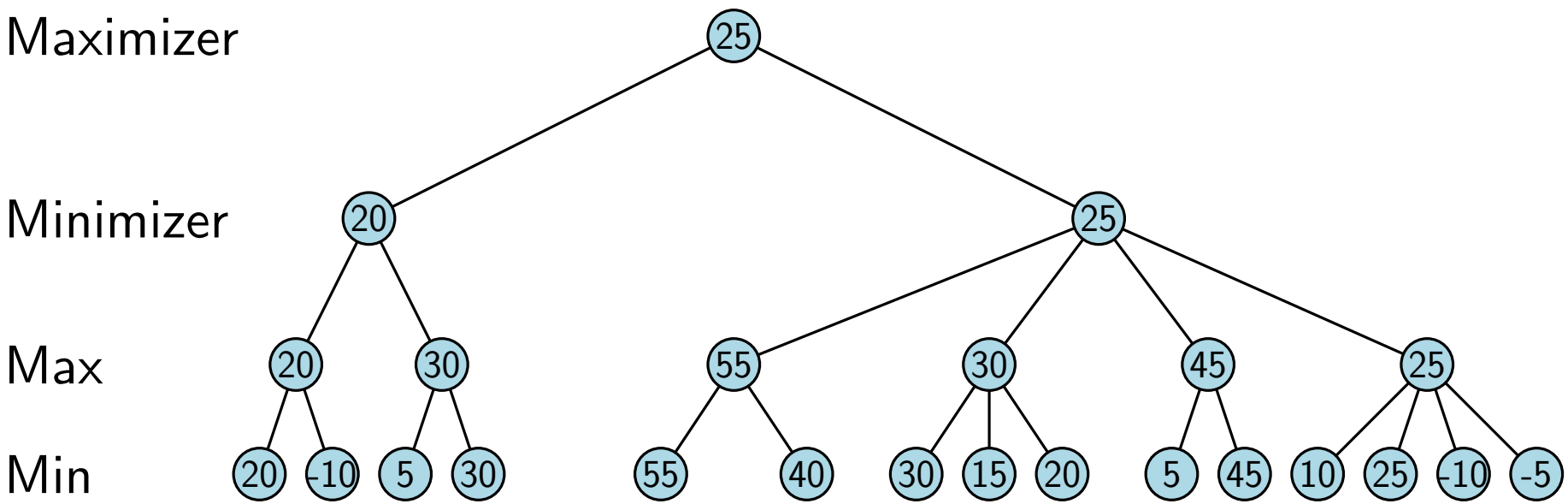


How much can we save with α-β pruning?

Higher degree trees: α-β pruning

- Game trees usual have higher degree

Example 3



How much can we save with α-β pruning?
Sometimes nothing!

Min Max and α-β Pruning

- The impact of α-β pruning depends on the order in which the states are considered
- Consider potentially best states first (large values for maximizer, small (large negativ) for minimizer)
- This is called **pre-sorted α-β pruning**

Min Max and α-β Pruning

- The impact of α-β pruning depends on the order in which the states are considered
- Consider potentially best states first (large values for maximizer, small (large negativ) for minimizer)
- This is called **pre-sorted α-β pruning**

- Min Max and α-β pruning are well established
- There exist excelent online courses and videos
- Ready to use code is freely available

Min Max and α-β Pruning

- The impact of α-β pruning depends on the order in which the states are considered
- Consider potentially best states first (large values for maximizer, small (large negativ) for minimizer)
- This is called **pre-sorted α-β pruning**

- Min Max and α-β pruning are well established
- There exist excelent online courses and videos
- Ready to use code is freely available

```
return ( end )
```