Game Playing
Part 1 Minimax Search

Yingyu Liang

yliang@cs.wisc.edu

Computer Sciences Department

University of Wisconsin, Madison

[based on slides from A. Moore, C. Dyer, J. Skrentny, Jerry Zhu]

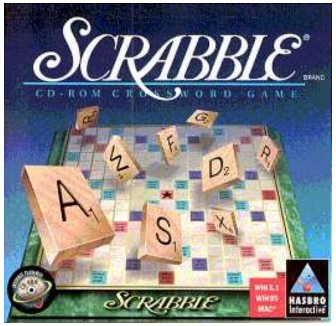
Not playing these games (not in this course) …

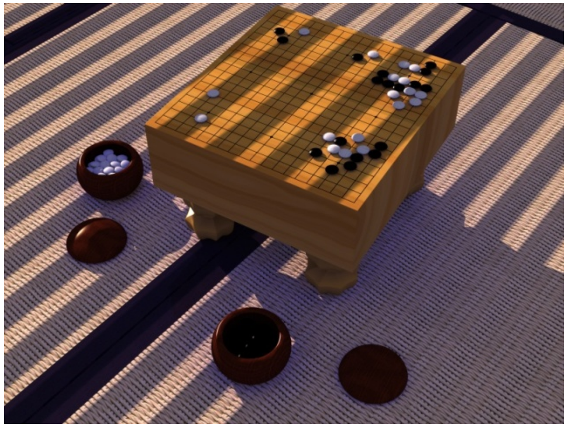
Instead, learn principles of how machines play

Overview

- Important characteristics of games
  - two-player zero-sum discrete finite deterministic game of perfect information
- Minimax search
- Alpha-beta pruning

Game Examples



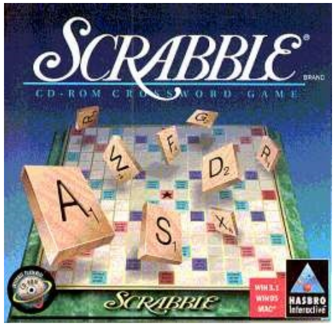


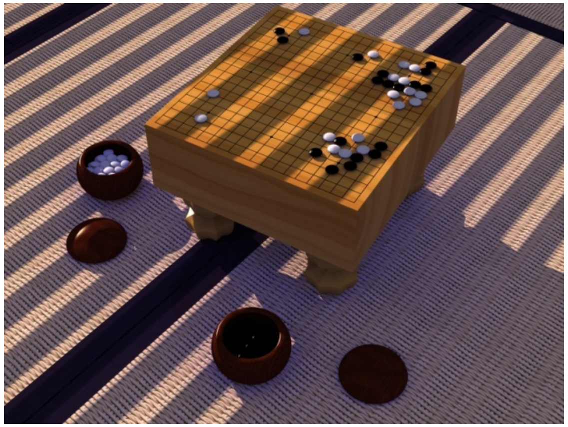
Two-player zero-sum discrete finite deterministic games of perfect information

Definitions:

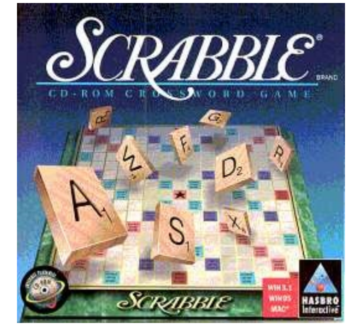
- Zero-sum: one player's gain is the other player's loss. Does not mean *fair*.
- Discrete: states and decisions have discrete values
- Finite: finite number of states and decisions
- Deterministic: no coin flips, die rolls – no chance
- Perfect information: each player can see the complete game state. No simultaneous decisions.

Which of these are: Two-player zero-sum discrete finite deterministic games of perfect information?

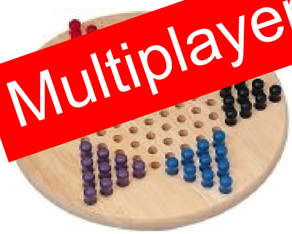




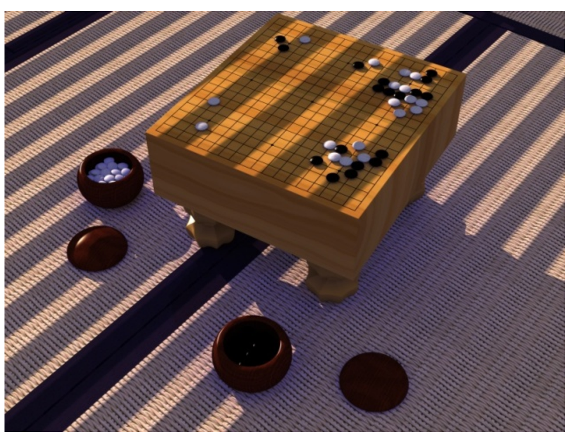
Which of these are: Two-player zero-sum discrete finite deterministic games of perfect information?



One player



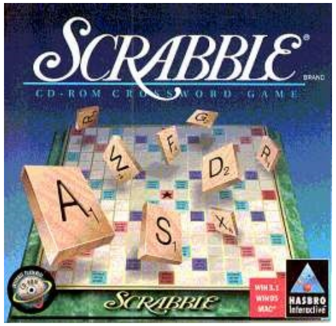
Multiplayer

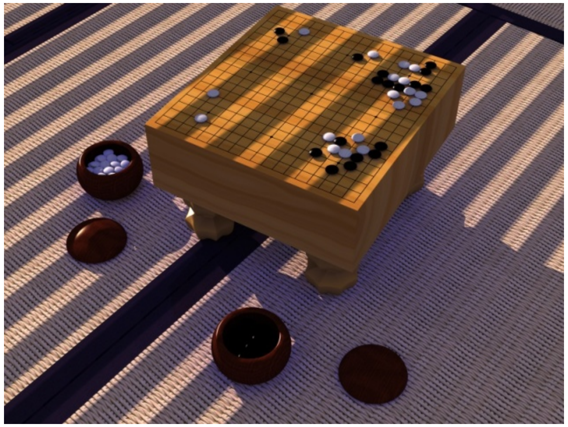


Involves Improbable Animal Behavior

Which of these are: Two-player zero-sum discrete finite deterministic games of perfect information?

Zero-sum: one player's gain is
the other player's loss.
Does not mean *fair*.

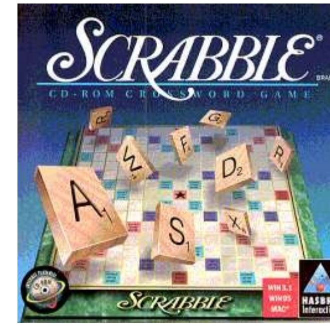


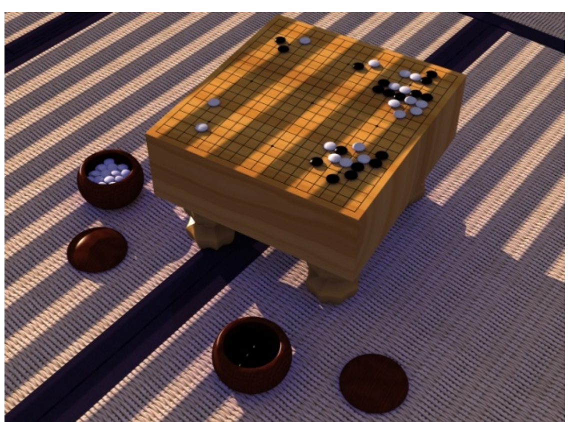


Which of these are: Two-player zero-sum discrete finite deterministic games of perfect information?

Zero-sum: one player's gain is the other player's loss. Does not mean *fair*.

Discrete: states and decisions have discrete values



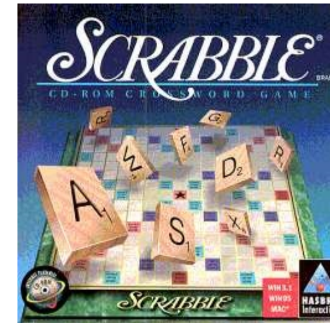


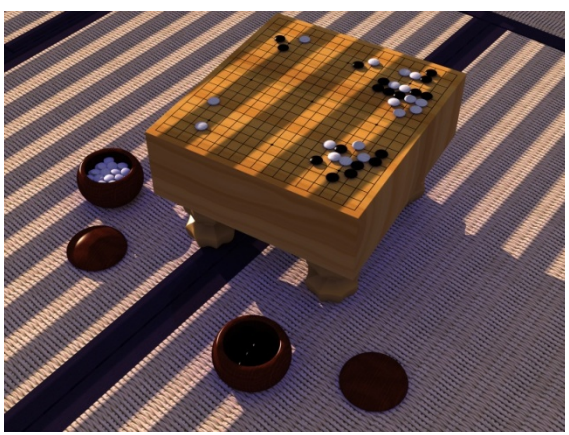
Which of these are: Two-player zero-sum discrete finite deterministic games of perfect information?

Zero-sum: one player's gain is the other player's loss. Does not mean *fair*.

Discrete: states and decisions have discrete values

Finite: finite number of states and decisions





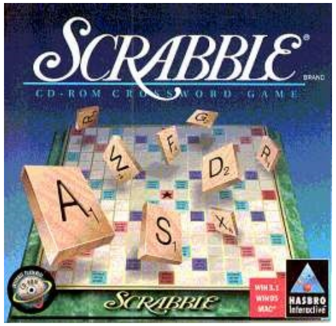
Which of these are: Two-player zero-sum discrete finite deterministic games of perfect information?

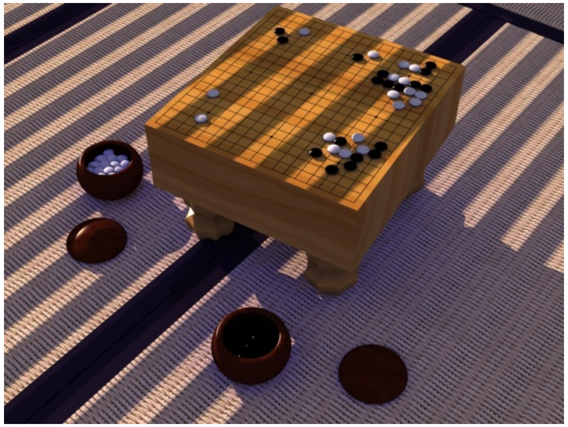
Zero-sum: one player's gain is the other player's loss. Does not mean *fair*.

Discrete: states and decisions have discrete values

Finite: finite number of states and decisions

Deterministic: no coin flips, die rolls – no chance





Which of these are: Two-player zero-sum discrete finite deterministic games of perfect information?

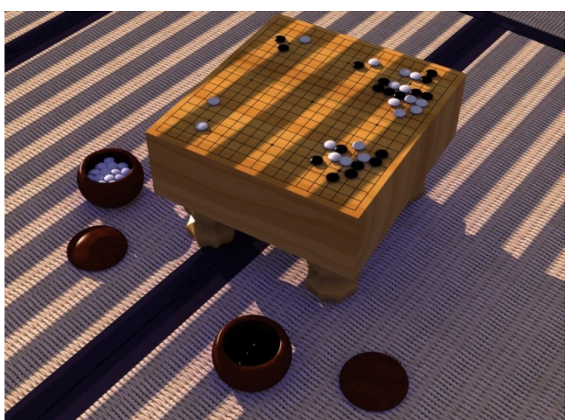
Zero-sum: one player's gain is the other player's loss. Does not mean *fair*.

Discrete: states and decisions have discrete values

Finite: finite number of states and decisions

Deterministic: no coin flips, die rolls – no chance

Perfect information: each player can see the complete game state. No simultaneous decisions.



Which of these are: Two-player zero-sum discrete finite deterministic games of perfect information?

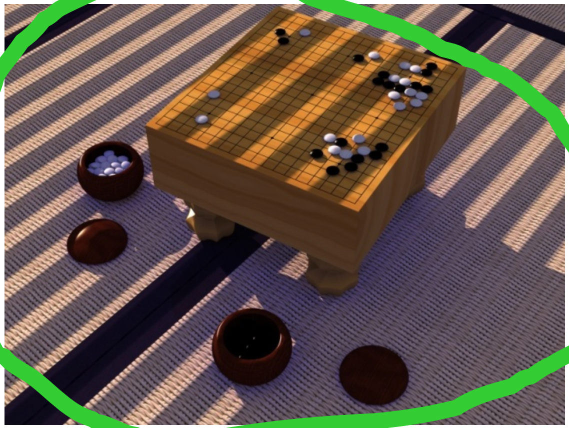
Zero-sum: one player's gain is the other player's loss. Does not mean *fair*.

Discrete: states and decisions have discrete values

Finite: finite number of states and decisions

Deterministic: no coin flips, die rolls – no chance

Perfect information: each player can see the complete game state. No simultaneous decisions.



II-Nim: Max simple game

- There are 2 piles of sticks. Each pile has 2 sticks.
- Each player takes one or more sticks from one pile.
- The player who takes the last stick loses.

(ii, ii)

II-Nim: Max simple game

- There are 2 piles of sticks. Each pile has 2 sticks.
- Each player takes one or more sticks from one pile.
- The player who takes the last stick loses.

(ii, ii)

- Two players: Max and Min
- If Max wins, the score is **+1**; otherwise **-1**
- Min's score is –Max's
- Use Max's as the score of the game

II-Nim: one trajectory of the game

(ii, ii)

II-Nim: one trajectory of the game

(ii, ii)

Max takes one stick from one pile

(i, ii)

II-Nim: one trajectory of the game

(ii, ii)

Max takes one stick from one pile

↓

(i, ii)

Min takes two sticks from the other pile

↓

(i,-)

II-Nim: one trajectory of the game

(ii, ii)

Max takes one stick from one pile

(i, ii)

Min takes two sticks from the other pile

(i,-)

Max takes the last stick

(-,-)

Max gets score **-1**

The game tree for II-Nim

Two players:

Max and Min

(ii ii) Max

who is to move at this state

Convention: score is w.r.t. the first player Max. Min's score = – Max

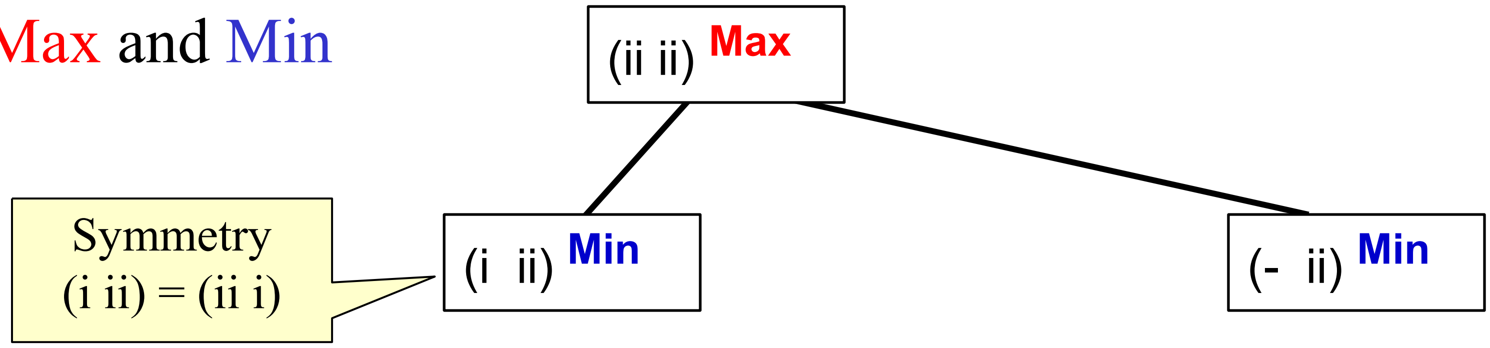
Max wants the largest score

Min wants the smallest score

The game tree for II-Nim

Two players:

Max and Min

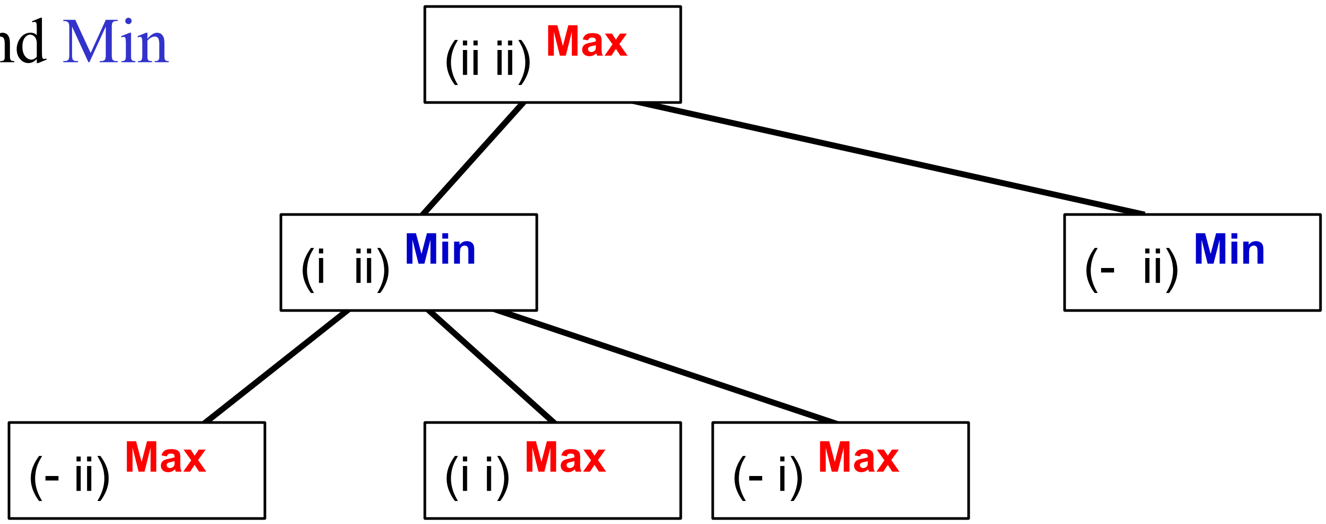


Max wants the largest score

Min wants the smallest score

The game tree for II-Nim

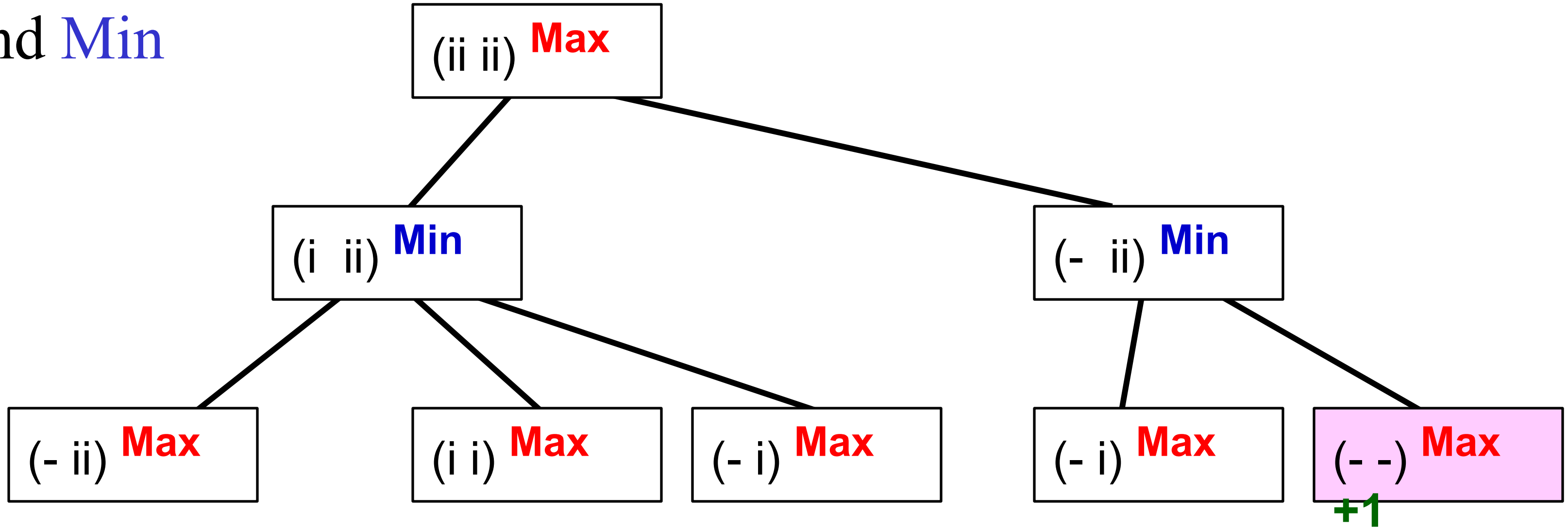
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

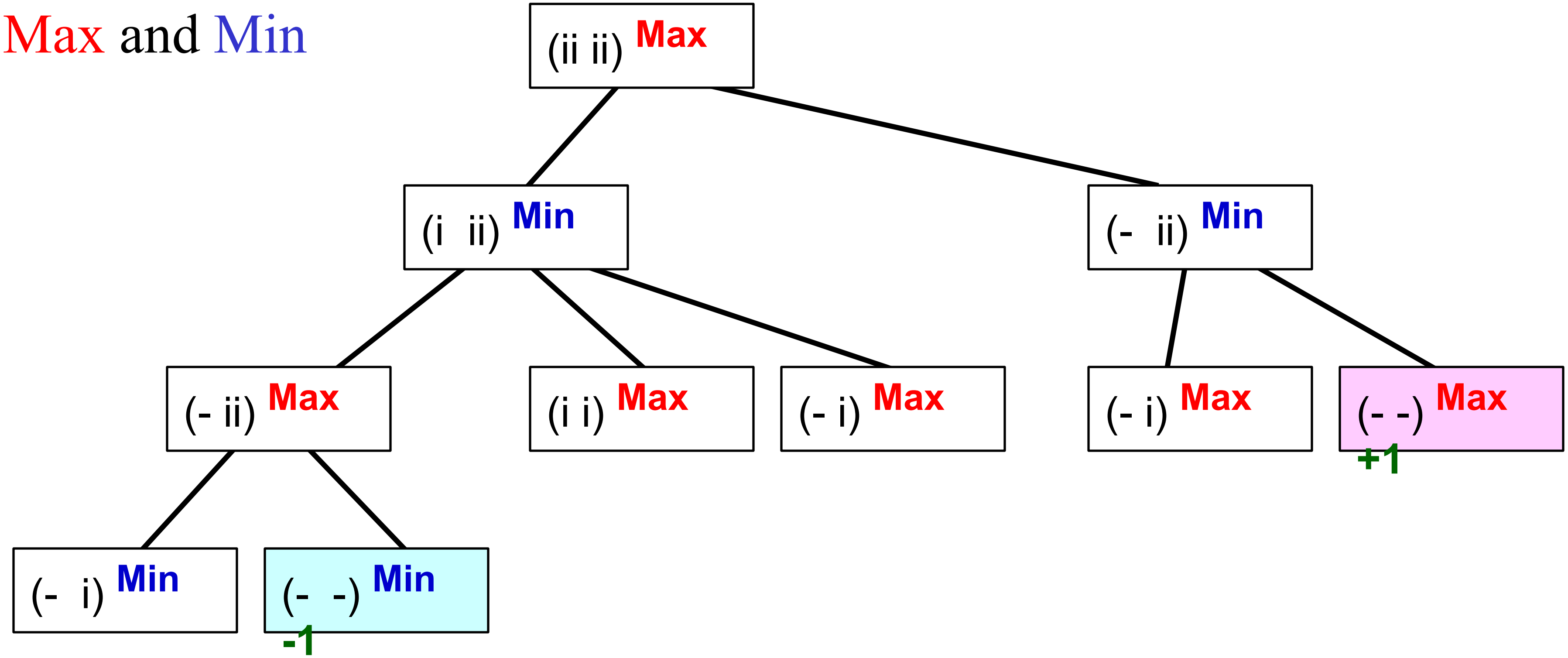
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

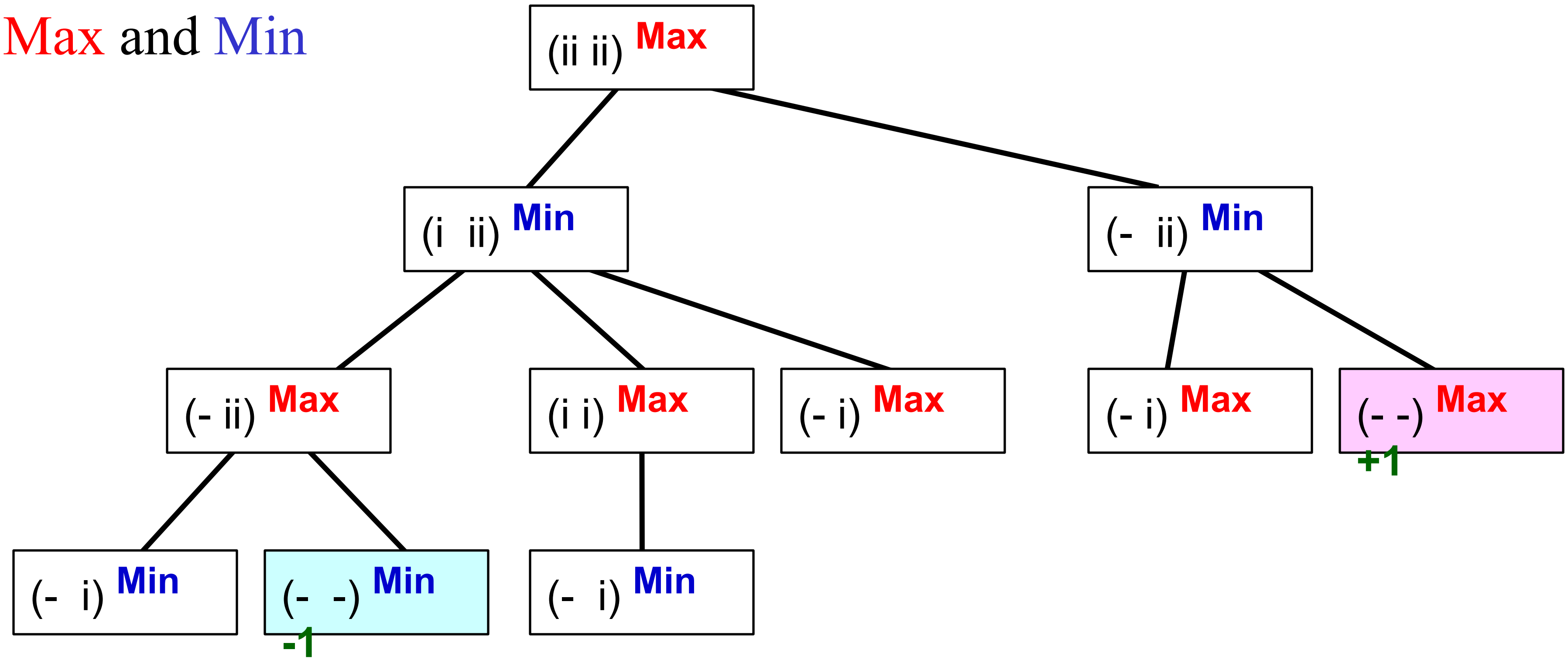
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

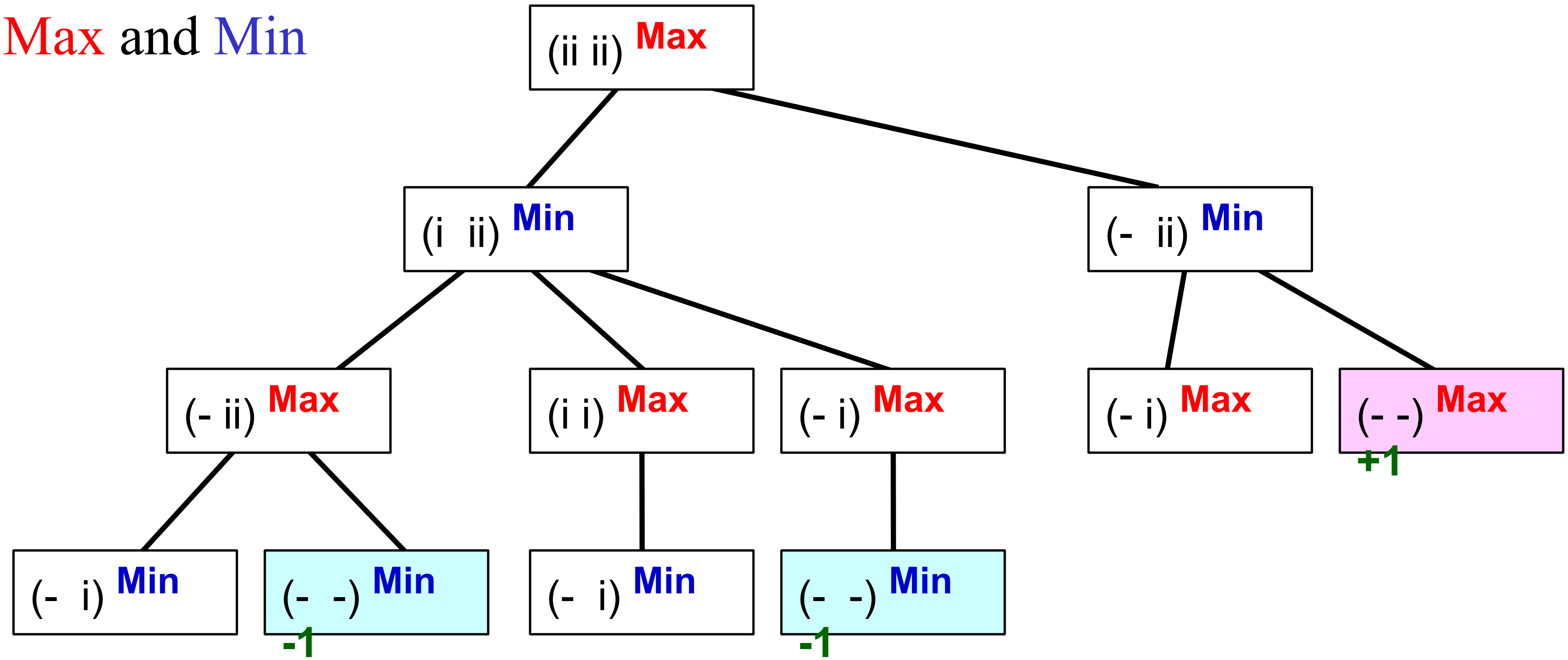
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

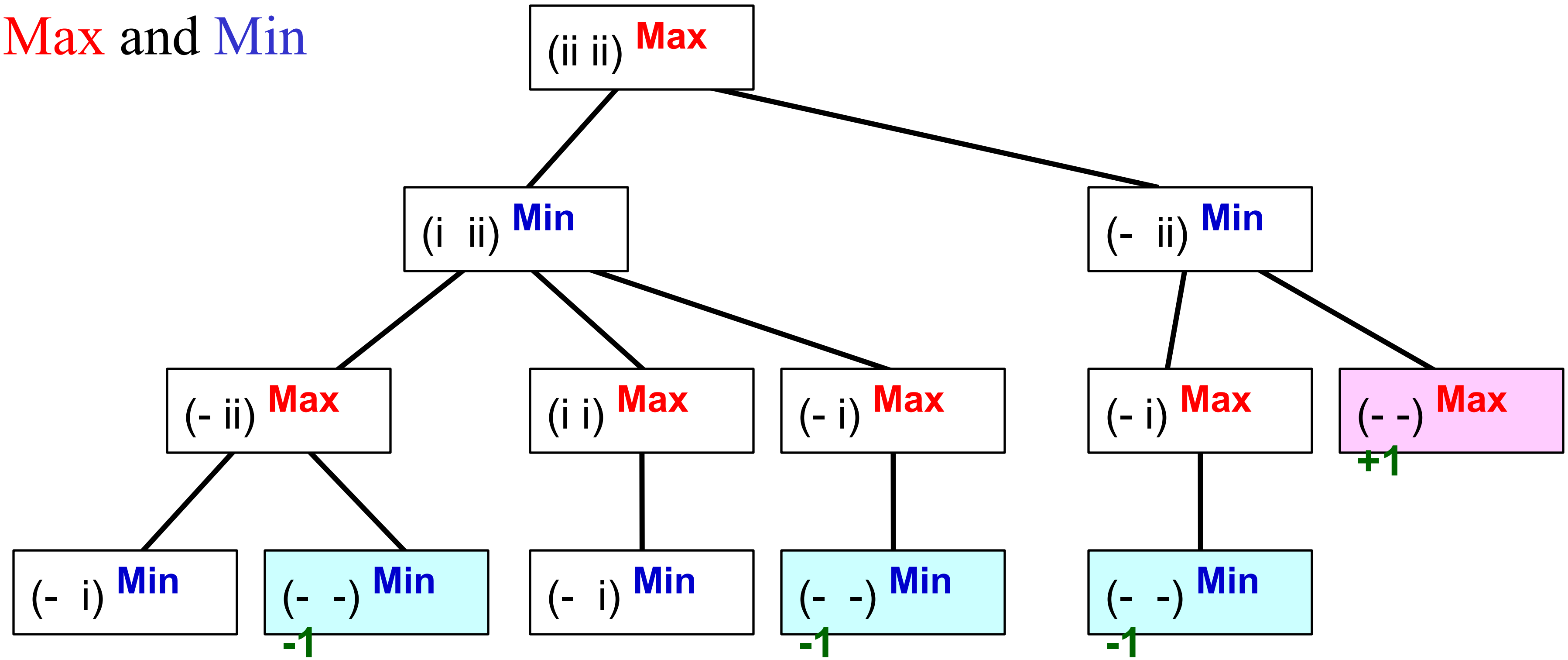
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

Two players:
Max and Min

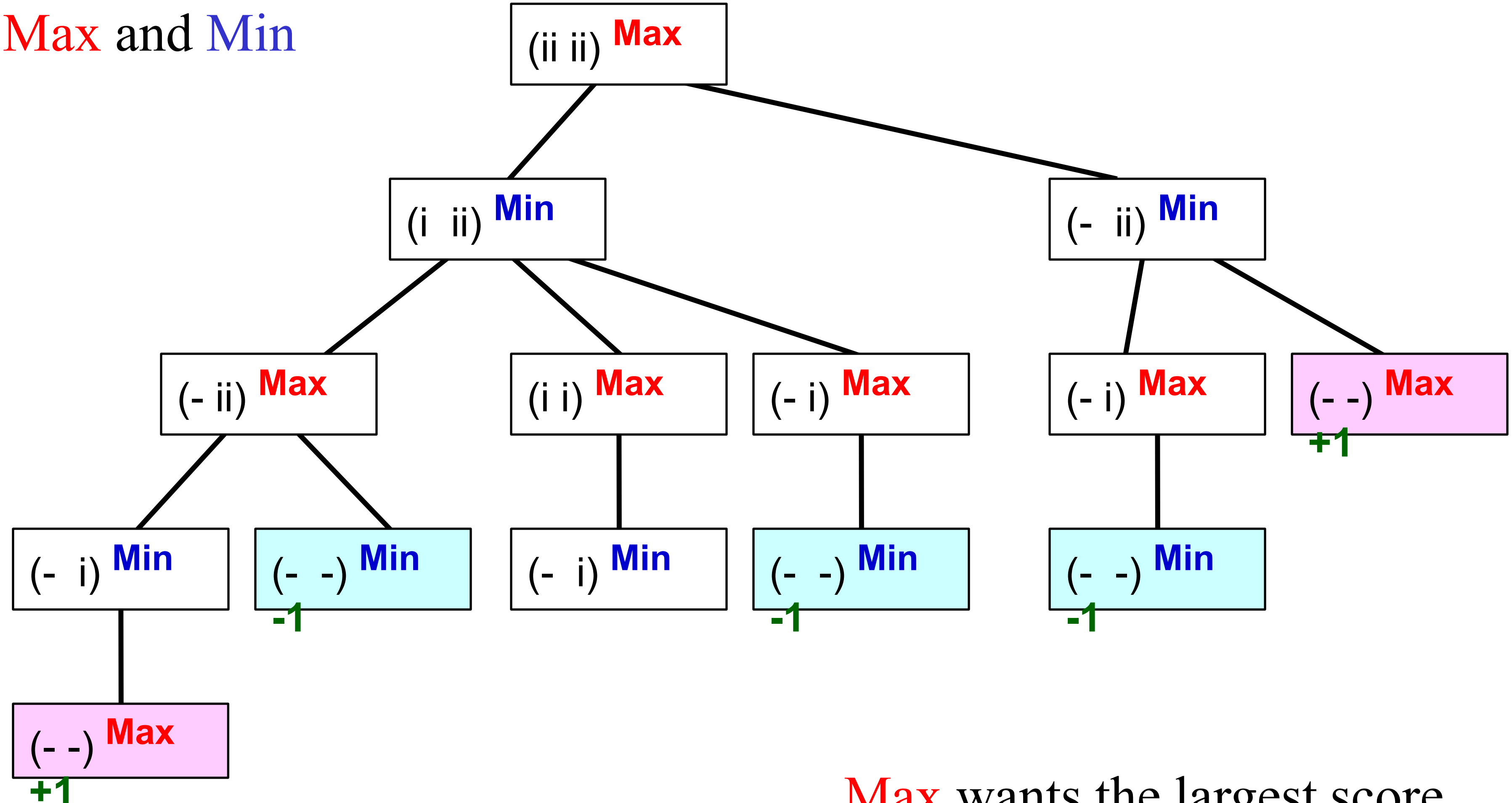


Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

Two players:

Max and Min

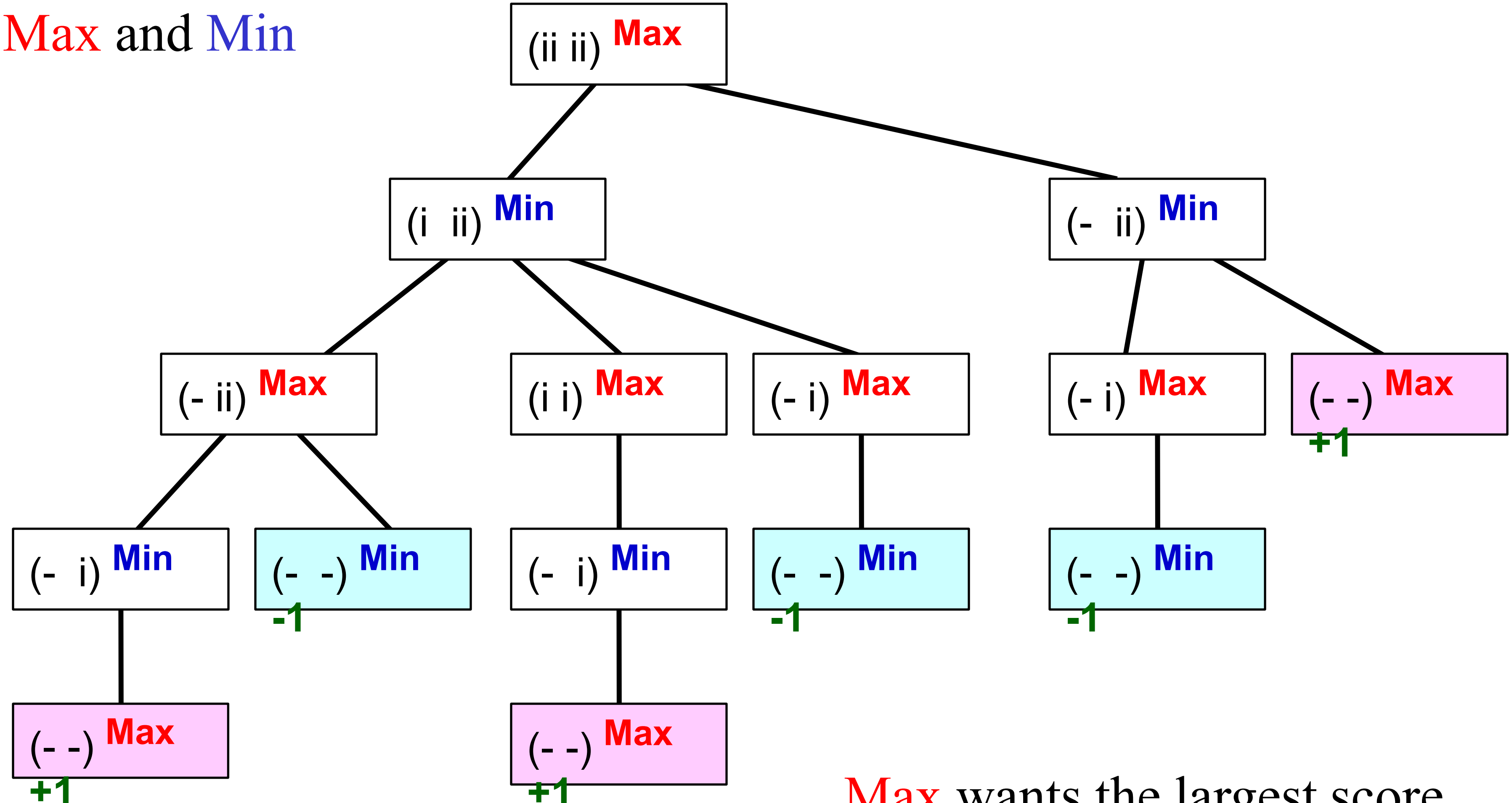


Max wants the largest score

Min wants the smallest score

The game tree for II-Nim

Two players:
Max and Min



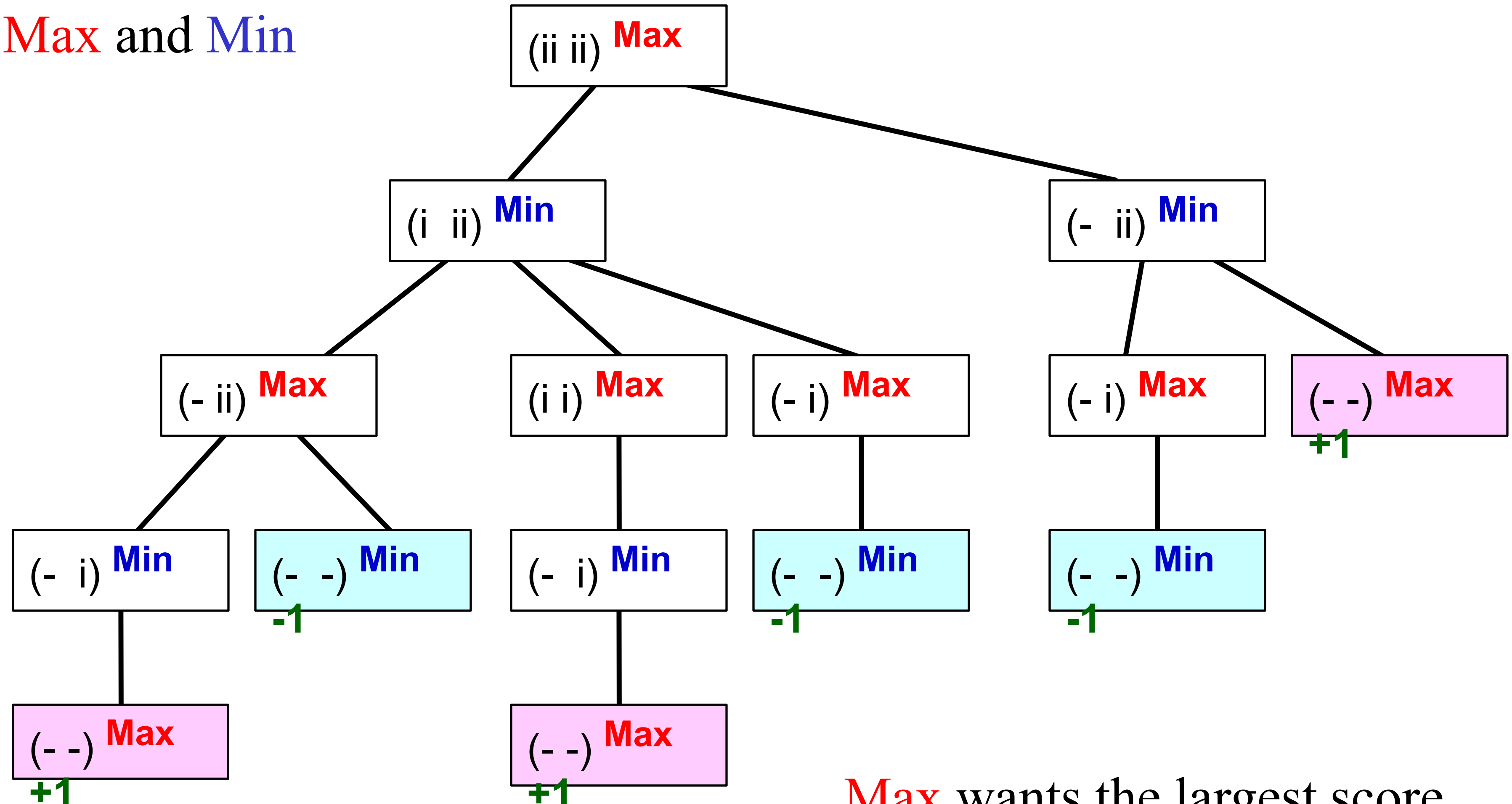
Max wants the largest score
Min wants the smallest score

Game theoretic value

- Game theoretic value (a.k.a. minimax value) of a node = the score of the terminal node that will be reached if both players play optimally.

The game tree for II-Nim

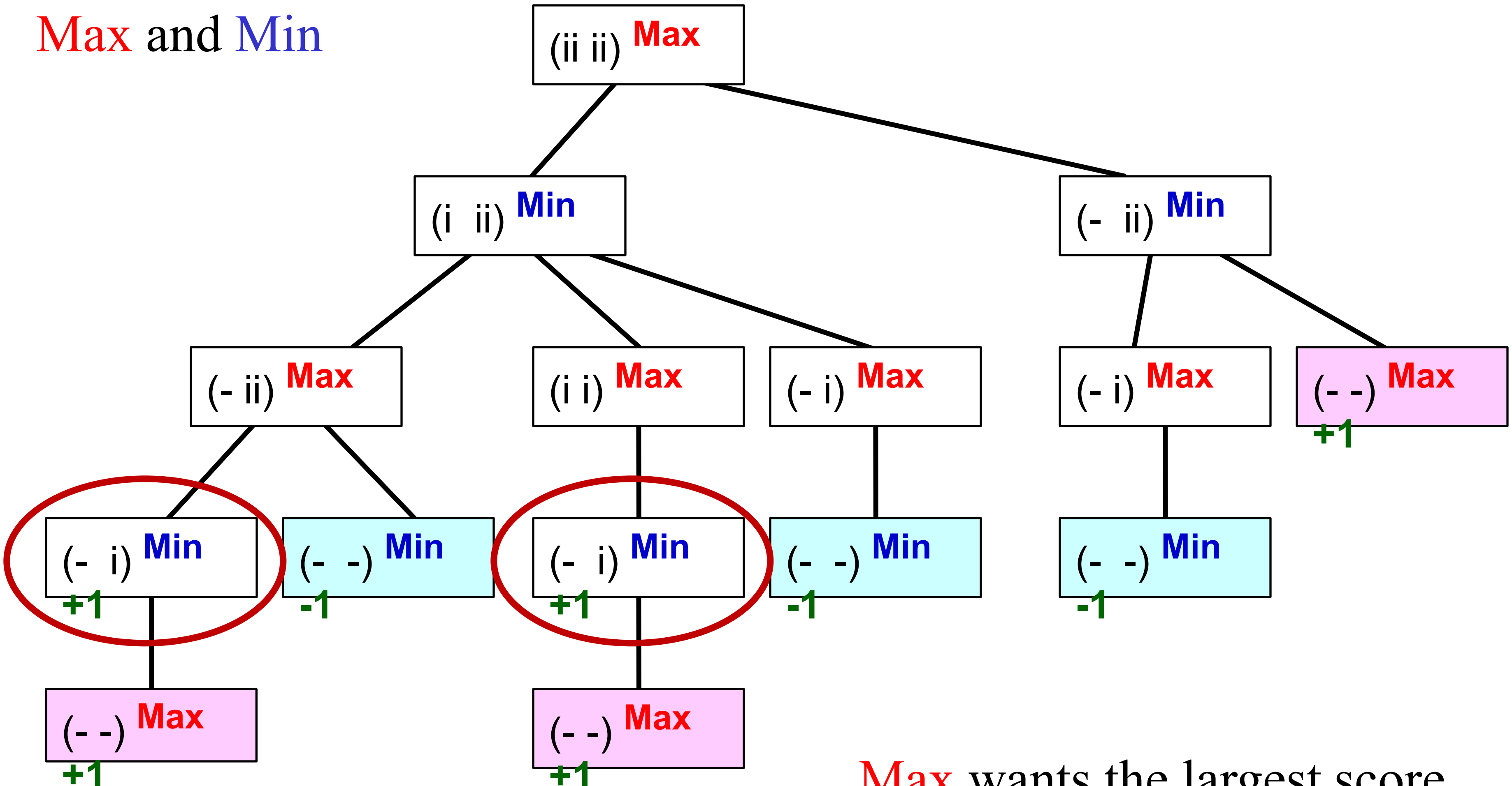
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

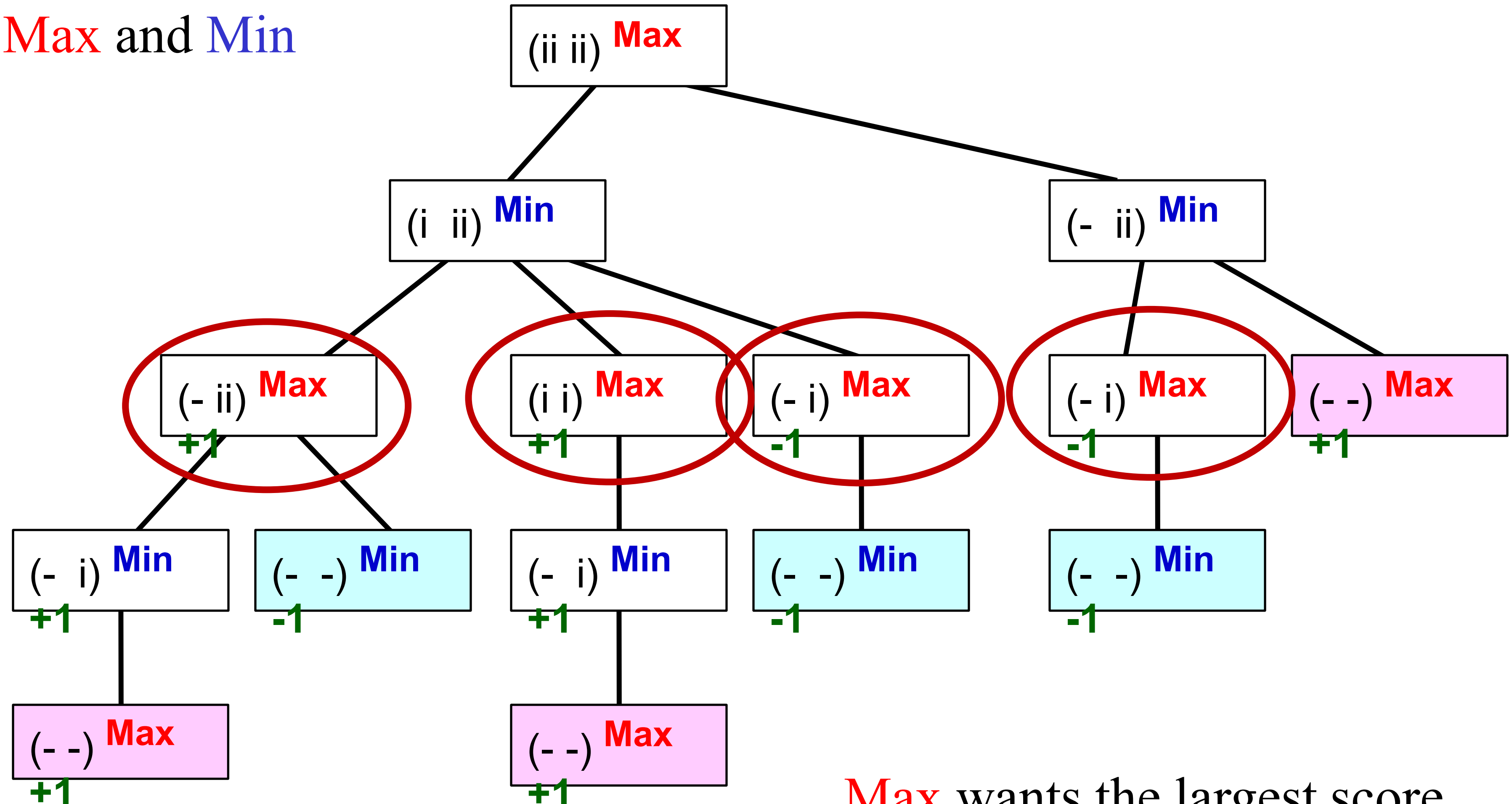
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

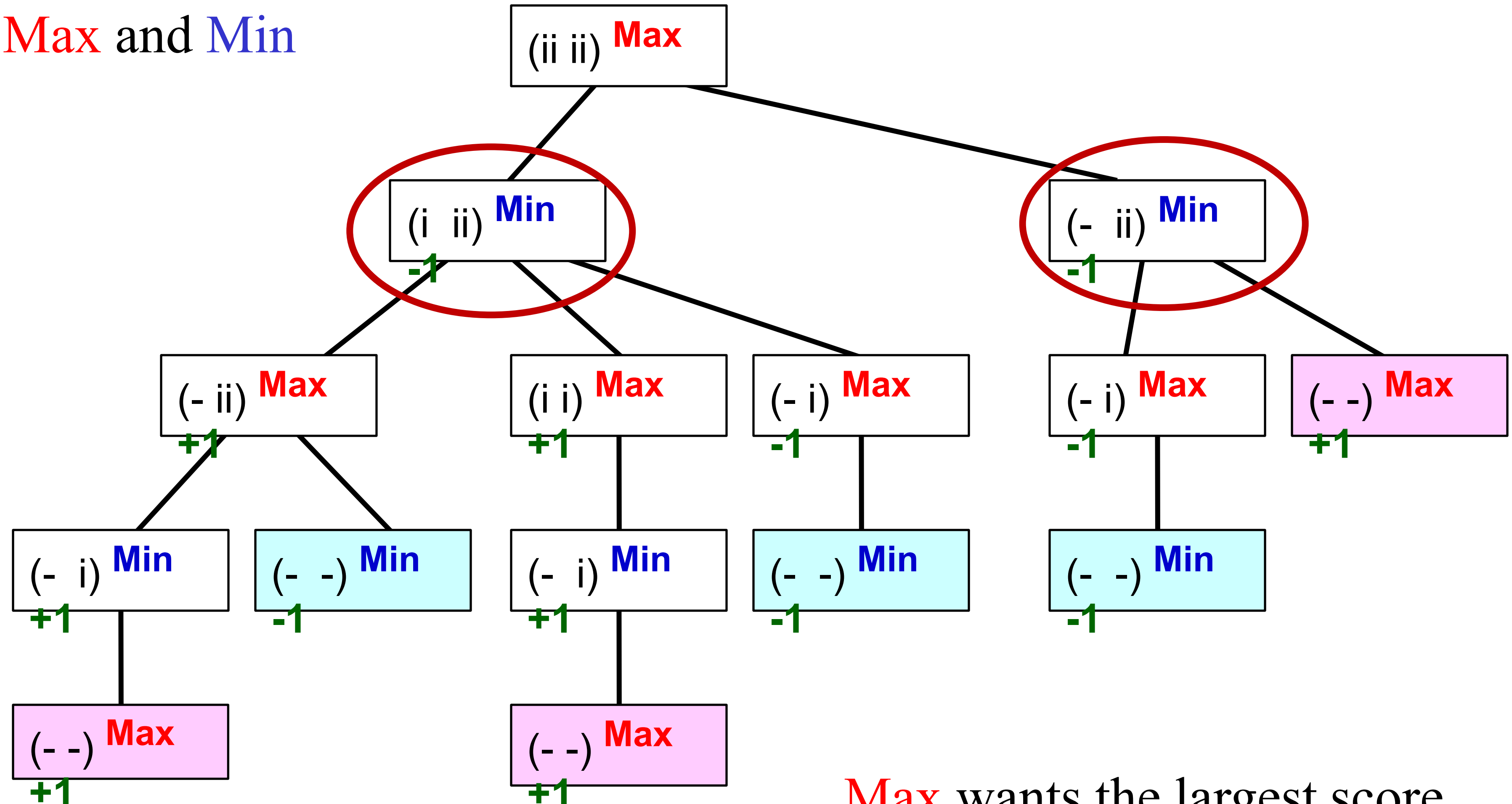
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

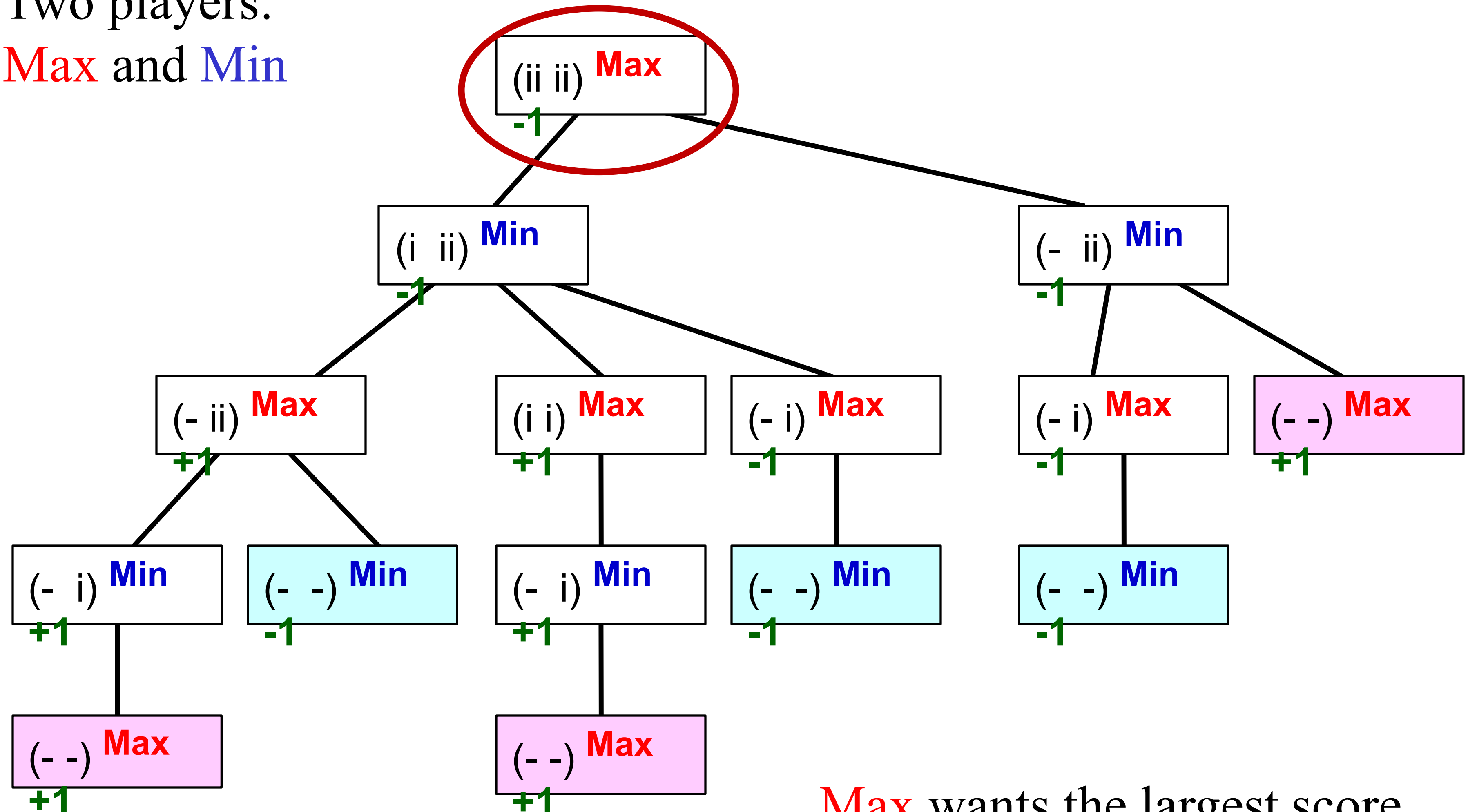
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

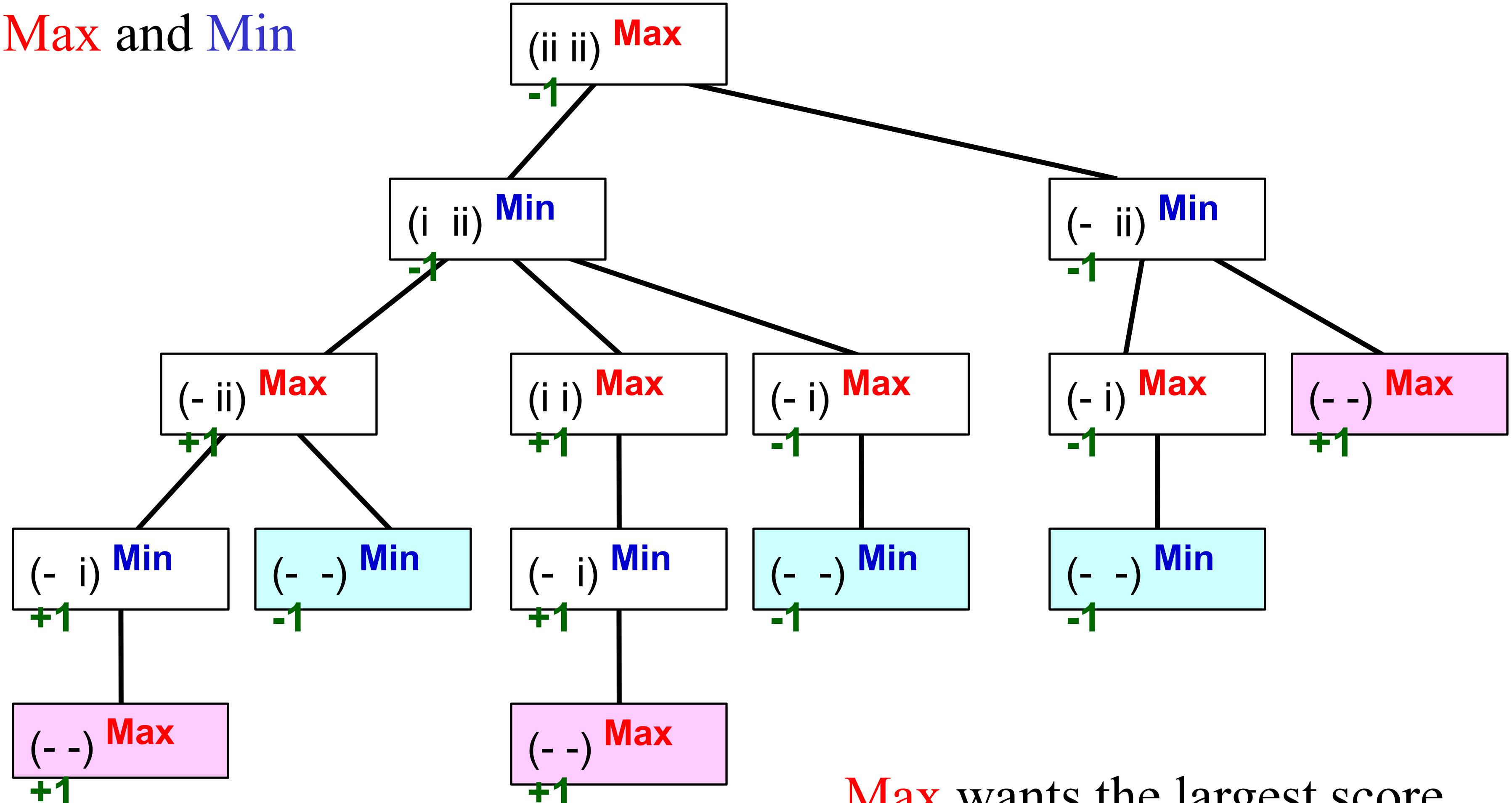
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

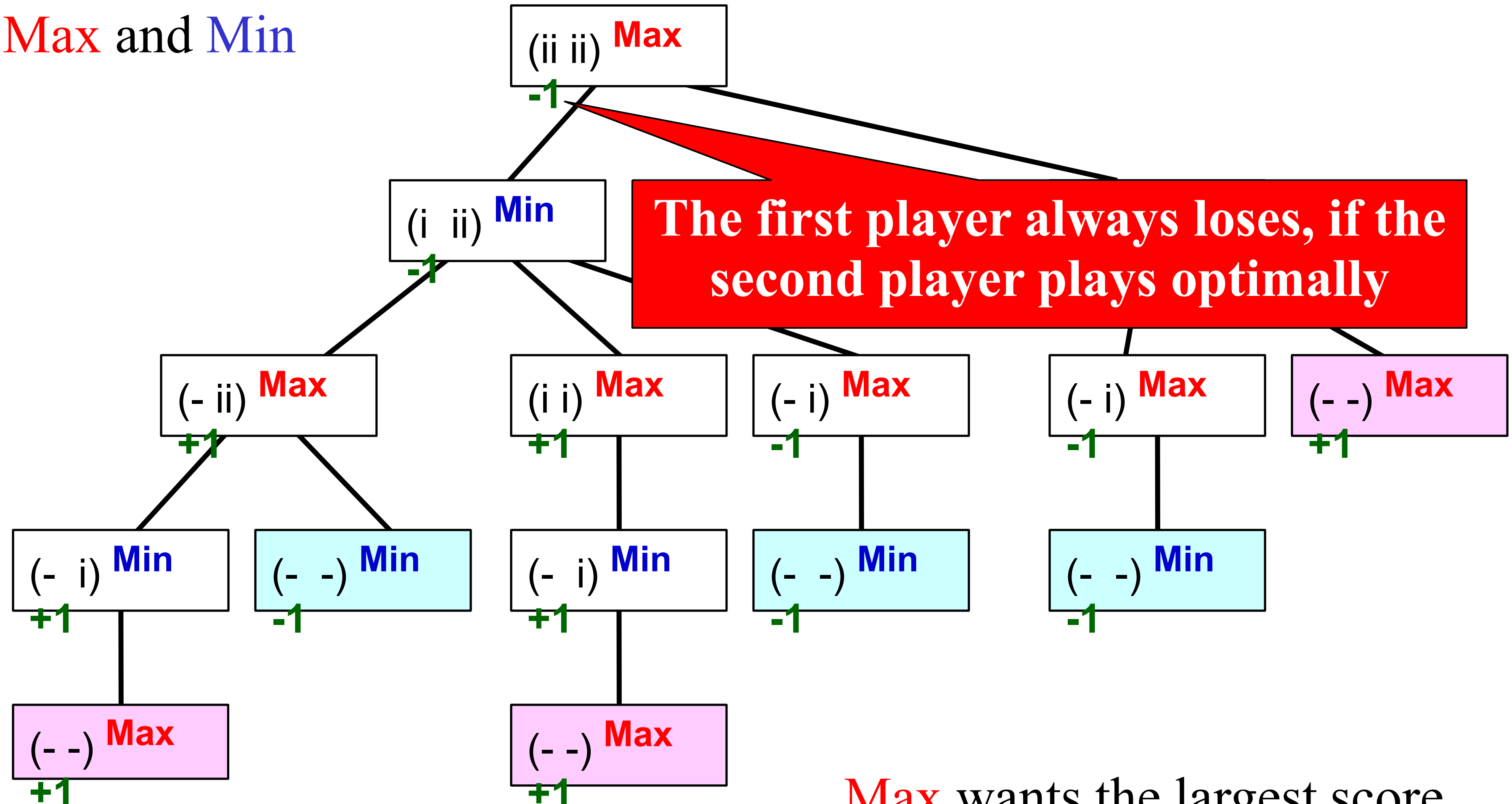
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

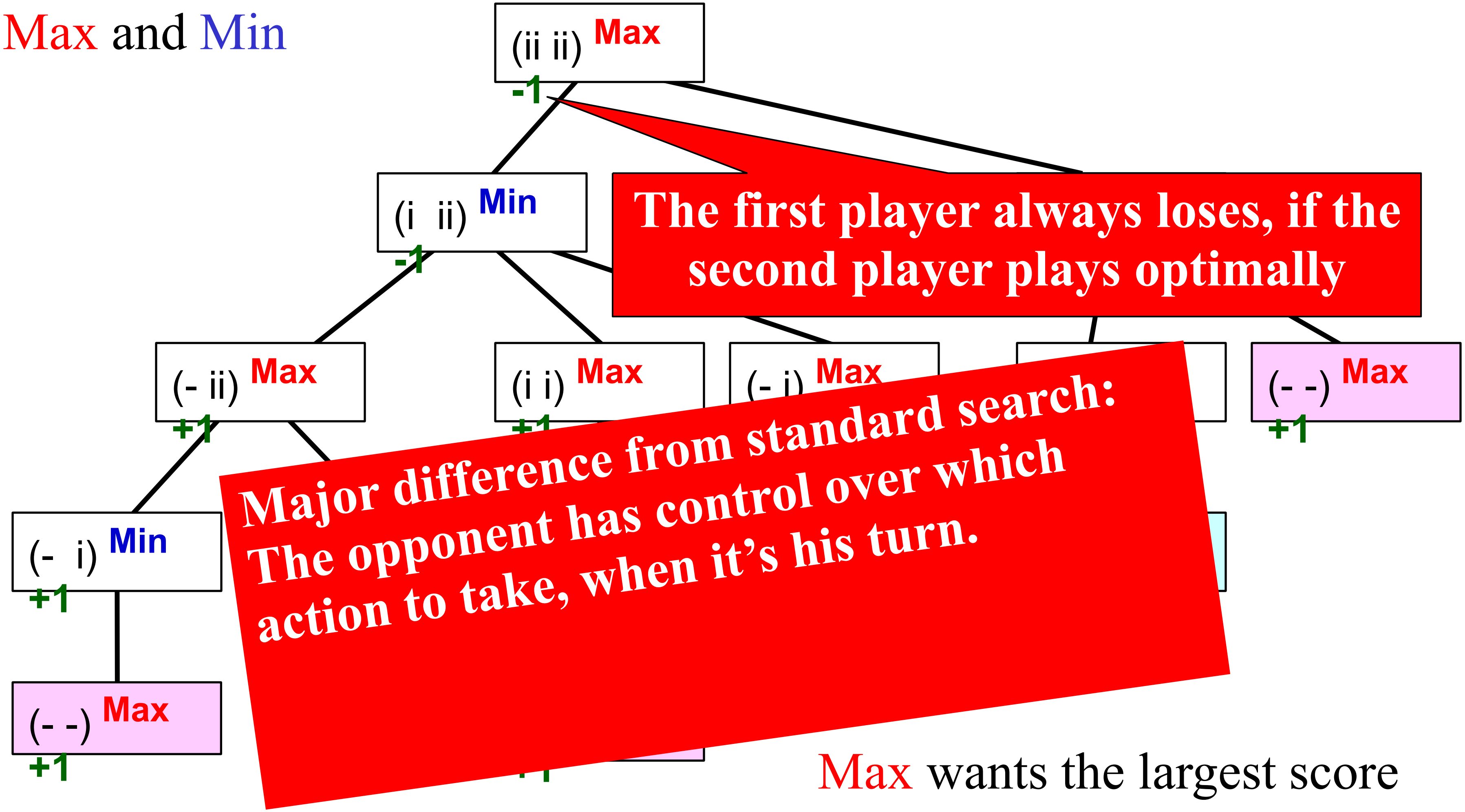
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

The game tree for II-Nim

Two players:
Max and Min



The first player always loses, if the second player plays optimally

Major difference from standard search: The opponent has control over which action to take, when it's his turn.

Max wants the largest score
Min wants the smallest score

Game theoretic value

- Game theoretic value (a.k.a. minimax value) of a node = the score of the terminal node that will be reached if both players play optimally.
- = The numbers we filled in.
- Computed bottom up
  - In Max's turn, take the max of the children (Max will pick that maximizing action)
  - In Min's turn, take the min of the children (Min will pick that minimizing action)
- Implemented as a modified version of DFS: minimax algorithm

Minimax algorithm

```
function Max-Value(s)
inputs:
    s: current state in game, Max about to play
output: best-score (for Max) available from s

    if ( s is a terminal state )
    then return ( terminal value of s )
    else
            α := – ∞
            for each s' in Succ(s)
                α := max( α , Min-value(s'))
    return α
```

```
function Min-Value(s)
output: best-score (for Min) available from s

    if ( s is a terminal state )
    then return ( terminal value of s)
    else
            β := ∞
            for each s' in Succs(s)
                β := min( β , Max-value(s'))
    return β
```

- Time complexity?
- Space complexity?

Minimax algorithm

```
function Max-Value(s)
inputs:
    s: current state in game, Max about to play
output: best-score (for Max) available from s

    if ( s is a terminal state )
    then return ( terminal value of s )
    else
            α := – ∞
            for each s' in Succ(s)
                α := max( α , Min-value(s'))
    return α
```

```
function Min-Value(s)
output: best-score (for Min) available from s

    if ( s is a terminal state )
    then return ( terminal value of s)
    else
            β := ∞
            for each s' in Succs(s)
                β := min( β , Max-value(s'))
    return β
```

- Time complexity?
  $O(b^m)$ ← bad
- Space complexity?
  $O(bm)$