CS 540 Introduction to Artificial Intelligence

Games II

University of Wisconsin-Madison

Spring 2026 Sections 1 & 2

Announcements

- **Homework**:
  - HW8 due on **Wednesday April 15th at 11:59 PM**

- Class roadmap:

Games – Part II
Introduction to Reinforcement Learning
Reinforcement Learning II

Outline

- Review/Complete Games I
- Sequential-move games
  - Game trees, minimax, search approaches
- Speeding up sequential-move game search
  - Pruning, heuristics

Games I Review

- Game Properties
- Simultaneous-Move Games
- Strictly Dominant Strategy and Dominant Strategy Equilibrium
- Nash Equilibrium
  - Pure Nash Equilibrium
  - Mixed Strategy Nash Equilibrium

Pure Nash Equilibrium may not exist

So far, pure strategy: each player picks a deterministic strategy. But:

Player 2 Player 1	*rock*	paper	*scissors*
rock	0, 0	-1, 1	1, -1
paper	1, -1	0, 0	-1, 1
scissors	-1, 1	1, -1	0, 0

Mixed Strategies

Can also randomize actions: "**mixed**"

- Player *i* assigns probabilities x_i to each action

$$x_i(a_i), \text{ where } \sum_{a_i \in A_i} x_i(a_i) = 1, x_i(a_i) \geq 0$$

- Now consider **expected rewards**

$$u_i(x_i, x_{-i}) = E_{a_i \sim x_i, a_{-i} \sim x_{-i}} u_i(a_i, a_{-i}) = \sum_{a_i} \sum_{a_{-i}} x_i(a_i) x_{-i}(a_{-i}) u_i(a_i, a_{-i})$$

Mixed Strategy Nash Equilibrium

Consider the mixed strategy $x^* = (x_1^*, \ldots, x_n^*)$

- This is a **Nash equilibrium** if

$$u_i(x_i^*, x_{-1}^*) \geq u_i(x_i, x_{-i}^*) \quad \forall x_i \in \Delta_{A_i}, \forall i \in \{1, \ldots, n\}$$

Better than doing anything else, "**best response**"

Space of probability distributions over strategies.

- Intuition: nobody can **increase expected reward** by changing only their own strategy.

Mixed Strategy Nash Equilibrium

Example: $x_1^*(\cdot) = x_2^*(\cdot) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$

Player 2 Player 1	*rock*	paper	*scissors*
rock	0, 0	-1, 1	1, -1
paper	1, -1	0, 0	-1, 1
scissors	-1, 1	1, -1	0, 0

Finding Mixed NE in 2-Player 2-action Zero-Sum Game

Example: Two Finger Morra. Show 1 or 2 fingers. The "even player" wins if the sum is even, and vice versa.

odd even	*f1*	f2
f1	2, -2	-3, 3
f2	-3, 3	4, -4

Finding Mixed NE in 2-Player 2-action Zero-Sum Game

Two Finger Morra. Two-player zero-sum game. No pure NE:

odd even	*f1*	f2
f1	2, -2	-3, 3
f2	-3, 3	4, -4

Finding Mixed NE in 2-Player 2-action Zero-Sum Game

Suppose odd's mixed strategy at NE is (*q*, 1-*q*), and even's (*p*, 1-*p*)

By definition, *p* is best response to *q*: $u_1(p,q) \geq u_1(p',q) \forall p'$.

Note $u_1(p,q) = pu_1(f_1,q) + (1-p)u_1(f_2,q)$

- Players only mix strategies if the expected payoffs are equal.
- If one strategy was better, they would never mix — they'd just pick the best!
  → $u_1(f_1,q) = u_1(f_2,q)$

- Average is no greater than components
  → $u_1(p,q) = u_1(f_1,q) = u_1(f_2,q)$

	odd / even	q: *f1*	1-q: f2
p	*f1*	2, -2	-3, 3
1-p	f2	-3, 3	4, -4

Finding Mixed NE in 2-Player 2-action Zero-Sum Game

$$u_1(f_1, q) = u_1(f_2, q)$$
$$2q + (-3)(1-q) = (-3)q + 4(1-q)$$
$$q = \frac{7}{12}$$

Similarly, $u_2(p, f_1) = u_2(p, f_2)$

$$p = \frac{7}{12}$$

At this NE, even gets -1/12, odd gets 1/12.

		q	1-q
	odd / even	*f1*	f2
p	*f1*	2, -2	-3, 3
1-p	f2	-3, 3	4, -4

Properties of Nash Equilibrium

Major result: (John Nash ’51)

- Every finite (players, actions) game has at least one Nash equilibrium
  - But not necessarily **pure** (i.e., deterministic strategy)
- Could be more than one
- Searching for Nash equilibria: computationally **hard**.
  - Exception: two-player zero-sum games (can be found with linear programming).

Break & Quiz

Q 1.1: Which of the following is **false?**

(i) Rock/paper/scissors has a dominant pure strategy
(ii) There is a Nash equilibrium for rock/paper/scissors

A. Neither

B. (i) but not (ii)

C. (ii) but not (i)

D. Both

Break & Quiz

Q 1.1: Which of the following is **false**?

(i) Rock/paper/scissors has a dominant pure strategy

(ii) There is a Nash equilibrium for rock/paper/scissors

A. Neither

B. (i) but not (ii)

C. (ii) but not (i)

D. Both

Break & Quiz

Q 1.1: Which of the following is **false**?

(i) Rock/paper/scissors has a dominant pure strategy

(ii) There is a Nash equilibrium for rock/paper/scissors

A. Neither

B. **(i) but not (ii)** (i) is indeed false: easy to check that there's no deterministic dominant strategy; (ii) is true: there is a mixed strategy Nash equilibrium

C. (ii) but not (i)

D. Both

Break & Quiz

Q 1.2: Which of the following statements is true about Nash equilibrium?

A. Mixed Strategy Nash Equilibrium all players choose their actions deterministically.
B. Every finite (players, actions) game has at least one pure Nash equilibrium.
C. A pure Nash Equilibrium is also always a Dominant Strategy Equilibrium.
D. A Dominant Strategy Equilibrium is also always a pure Nash Equilibrium.

Break & Quiz

Q 1.2: Which of the following statements is true about Nash equilibrium?

A. Mixed Strategy Nash Equilibrium all players choose their actions deterministically.

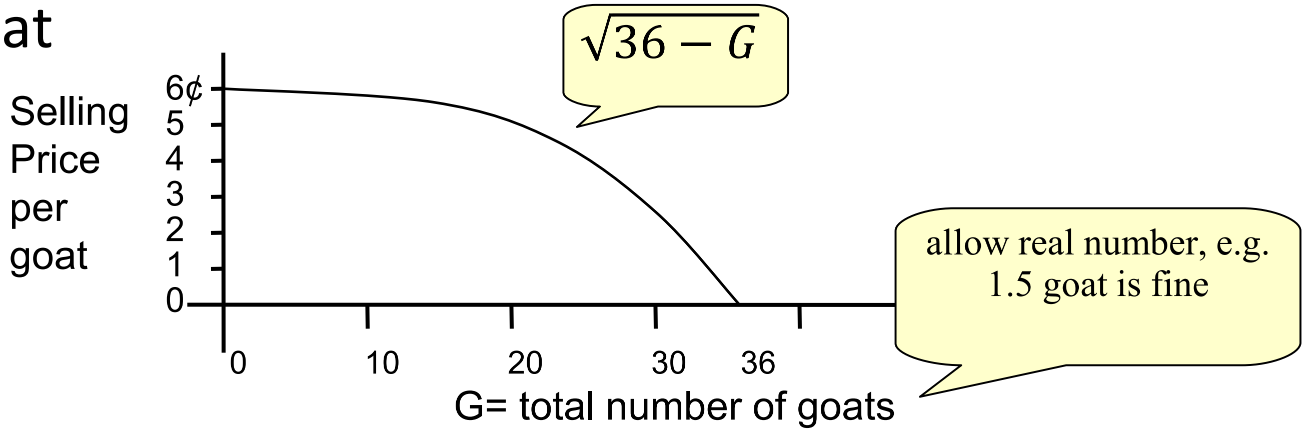
B. Every finite (players, actions) game has at least one pure Nash equilibrium.

C. A pure Nash Equilibrium is also always a Dominant Strategy Equilibrium.

D. A Dominant Strategy Equilibrium is also always a pure Nash Equilibrium.

Pure NE in an Infinite game: The tragedy of the Commons

- Price per goat



- How many goats should one (out of *n*) rational farmer graze?
- How much would the farmer earn?

Continuous Action Game

- Each farmer has infinite number of strategies $g_i \in [0,36]$
- The value for farmer *i*, when the *n* farmers play at $(g_1, g_2, \ldots, g_n)$ is

$$u_i(g_1, g_2, \ldots, g_n) = g_i \sqrt{36 - \sum_{j \in [n]} g_j}$$

- Assume a pure Nash equilibrium exists.
- Assume (by apparent symmetry) the NE is $(g^*, g^*, \ldots, g^*)$.

Finding g*

- $u_i(g_1, g_2, \dots, g_n) = g_i\sqrt{36 - \sum_j g_j}$
- g* is the best response to others (g*,…, g*)

$$g^* = argmax_{g_i \in [0,36]} u_i(g^*, \dots, g_i, \dots, g^*)$$

i-th argument

$$= argmax_{g_i} g_i \sqrt{36 - (n-1)g^* - g_i}$$

Finding g*

$$g^* = argmax_{g_i} g_i \sqrt{36 - (n-1)g^* - g_i}$$

- Taking derivative w.r.t. g_i, setting to 0:

$$g^* = \frac{72 - 2(n-1)g^*}{3}$$

$$g^* = \frac{72}{2n+1}$$

So what?

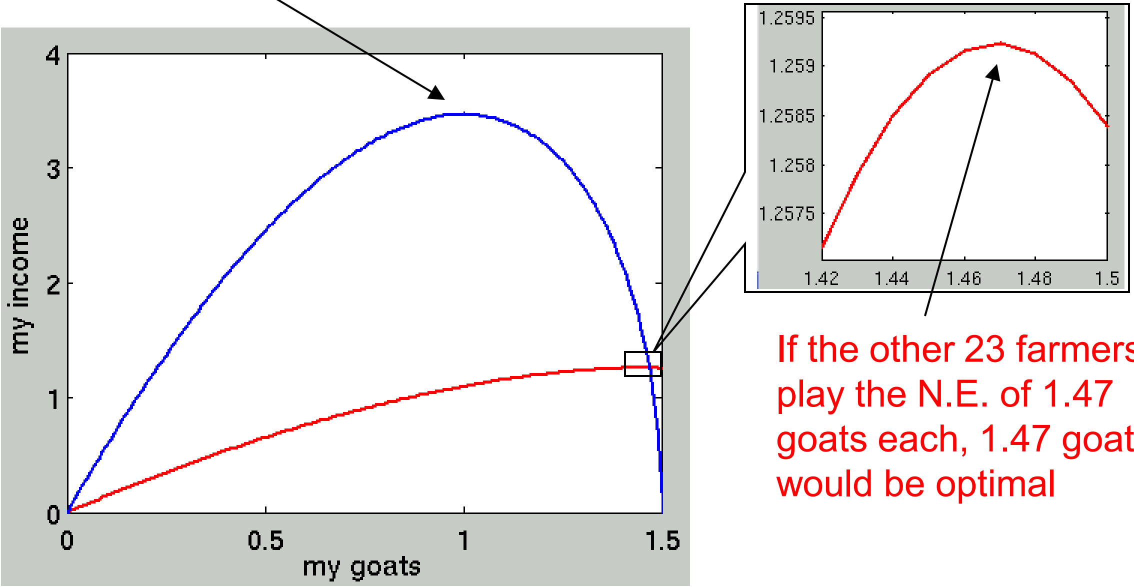
The tragedy of the Commons

- Say there are *n*=24 farmers.
  Each would rationally graze $g_i^* = 72/(2*24+1) = 1.47$ goats
- Each would get 1.25¢

- But if they cooperate and each grazes only 1 goat
- Each would get 3.46¢

The tragedy of the Commons

If all 24 farmers agree on the same number of goats to raise, 1 goat per farmer would be optimal

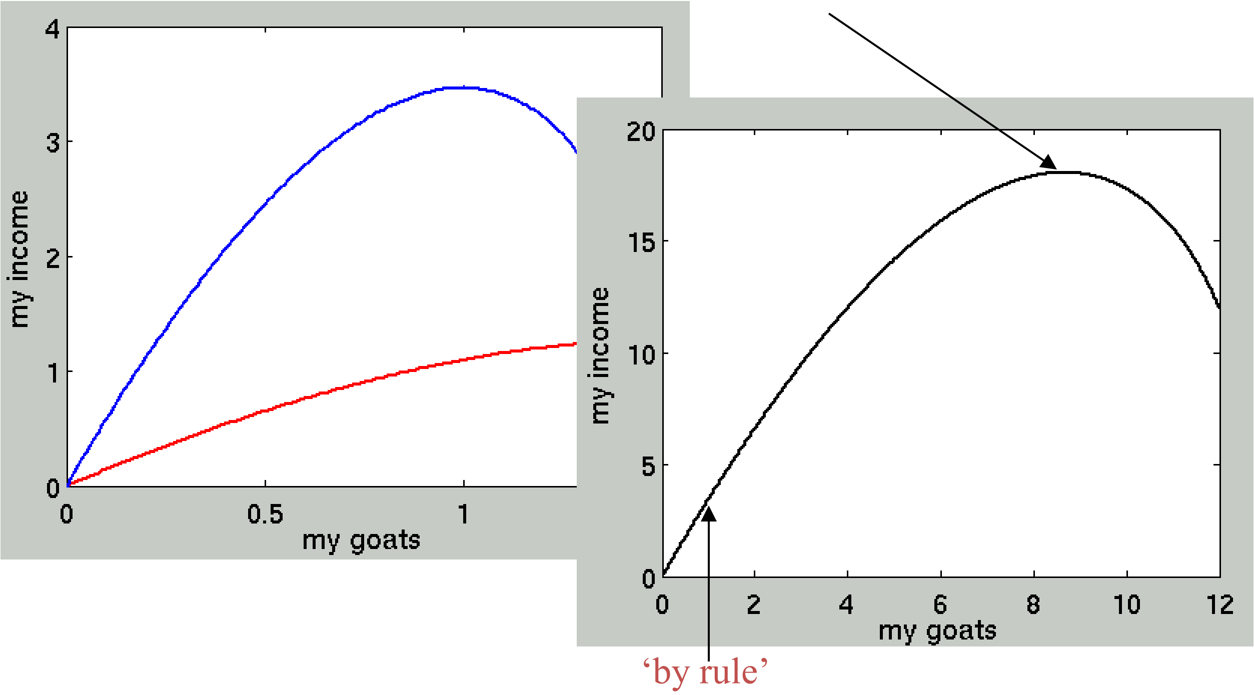


If the other 23 farmers play the N.E. of 1.47 goats each, 1.47 goats would be optimal

The tragedy of the Commons

If all 24 farmers agree on the same number of goats to raise, 1 goat per farmer would be optimal

But this is not a N.E.! A farmer can benefit from cheating (other 23 play at 1):



The tragedy

- Rational behaviors lead to sub-optimal solutions.
- Maximizing individual welfare not necessarily maximizes social welfare
- What went wrong?

Shouldn’t have allowed free grazing?

It’s not just the goats: Common problem for shared resources.

Mechanism design: designing the rules of a game

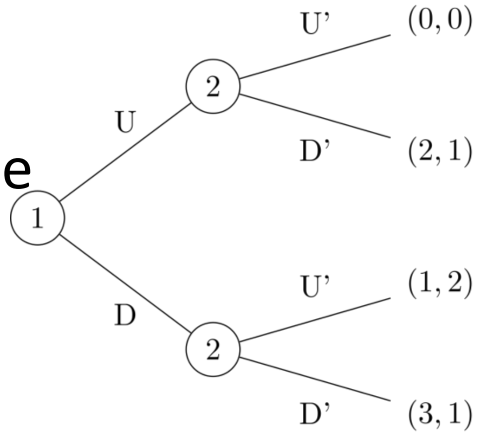
Outline

- Games I Review
- Sequential-move games
  - Game trees, minimax, search approaches
- Speeding up sequential-move game search
  - Pruning, heuristics

Sequential-Move Games

More complex games with multiple moves

- Instead of normal form, **extensive form**
- Represent with a **tree**
- **Rewards at leaves**
- Find strategies: perform search over the tree



Wiki

- Nash equilibrium still well-defined
  - Backward induction

II-Nim: Example Sequential-Move Game

2 piles of sticks, each with 2 sticks.

- Each player takes one or more sticks from pile
- Take last stick: lose

(ii, ii)

- Two players: Max and Min
- If Max wins, its score is **+1**; otherwise **-1**
- Min's score is –Max's (two-player zero-sum)
- Use Max's as the score of the game

Game Trajectory

(ii, ii)

Game Trajectory

(ii, ii)

Max takes one stick from one pile

(i, ii)

Game Trajectory

(ii, ii)

Max takes one stick from one pile

(i, ii)

Min takes two sticks from the other pile

(i,-)

Game Trajectory

(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**

Game tree for II-Nim

Two players:
Max and Min

(ii ii) Max

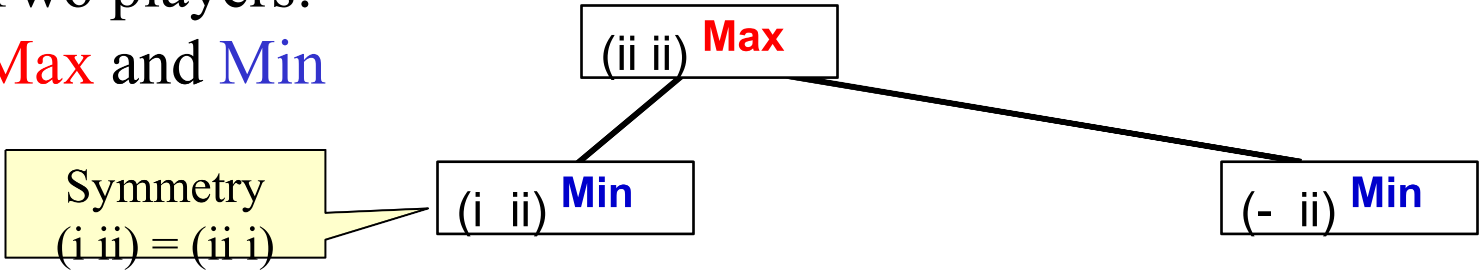
who is to move at this state

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

Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

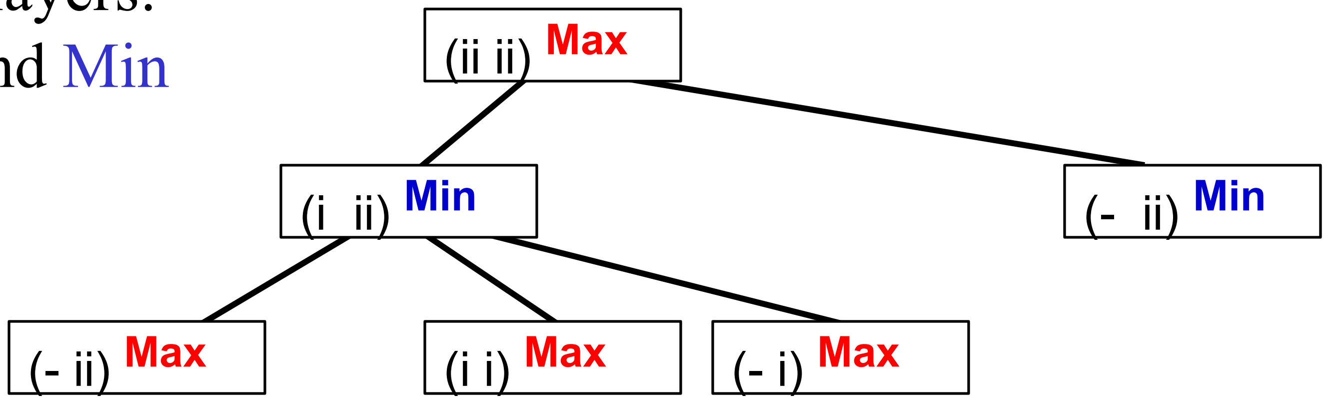
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

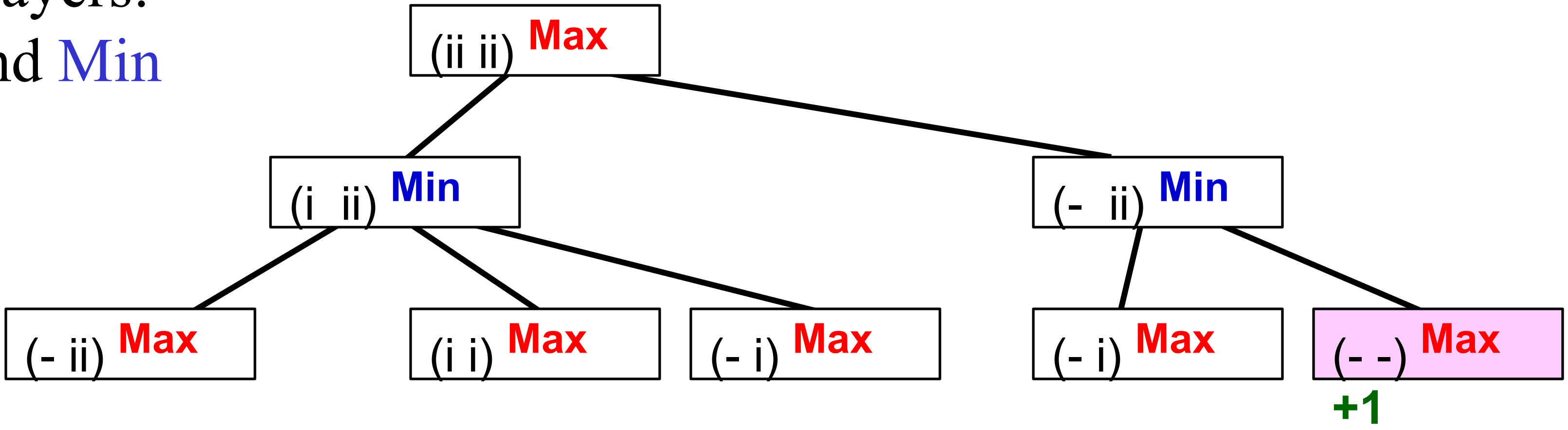
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

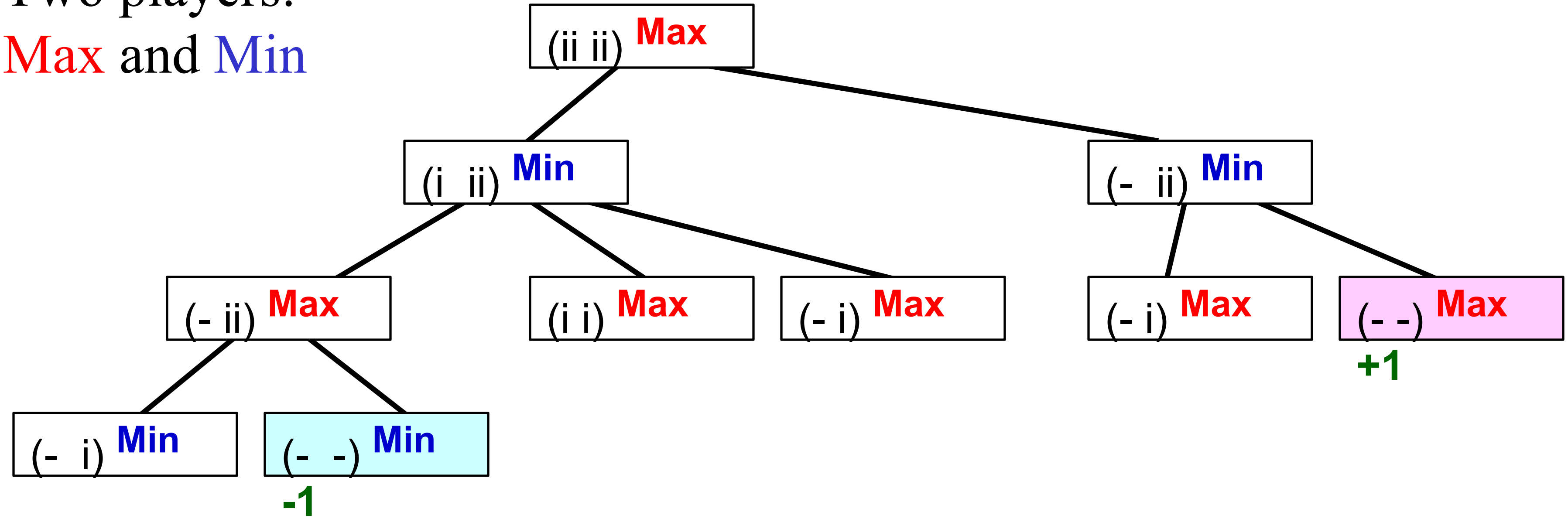
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

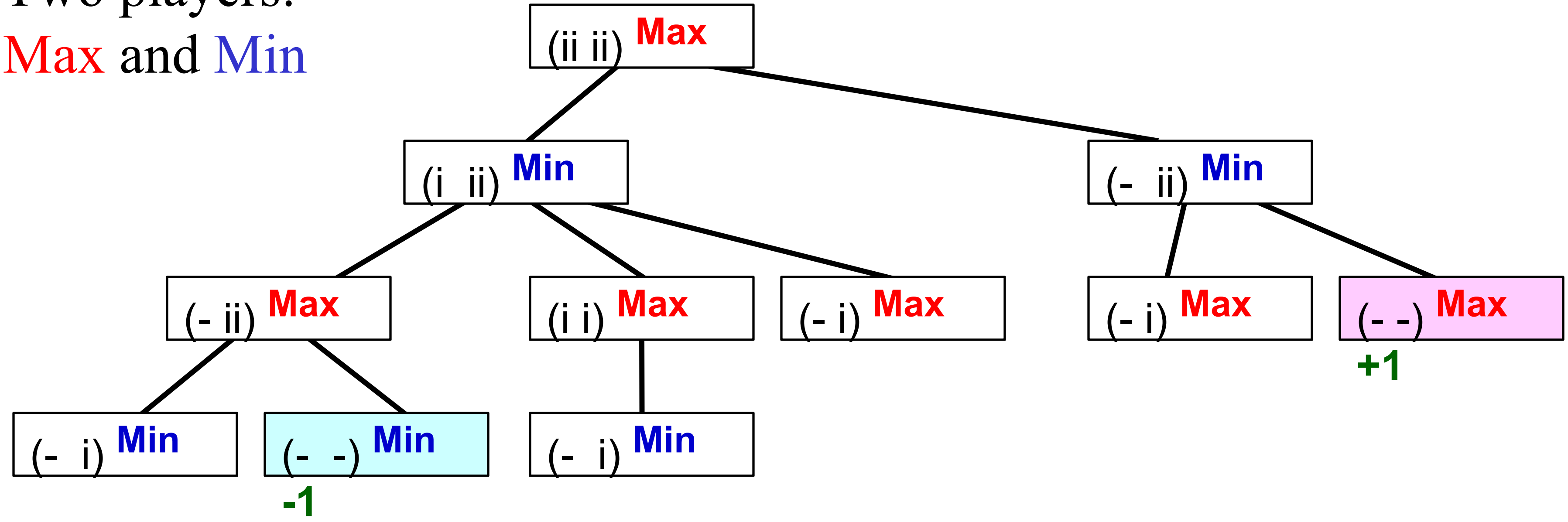
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

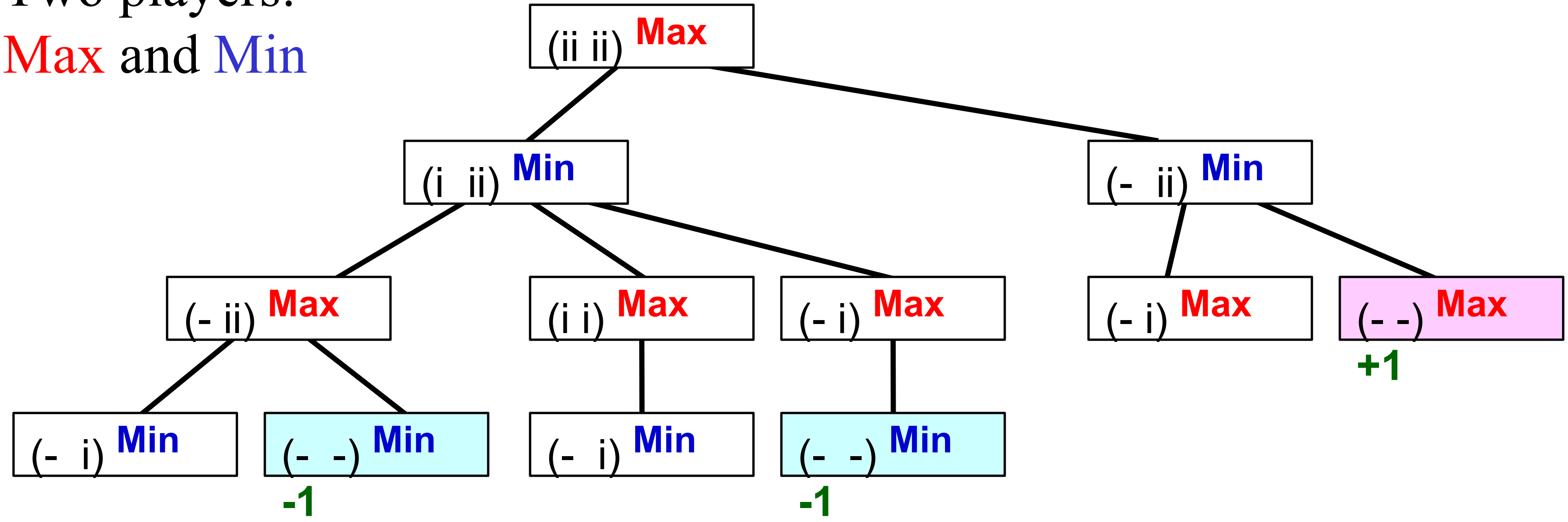
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

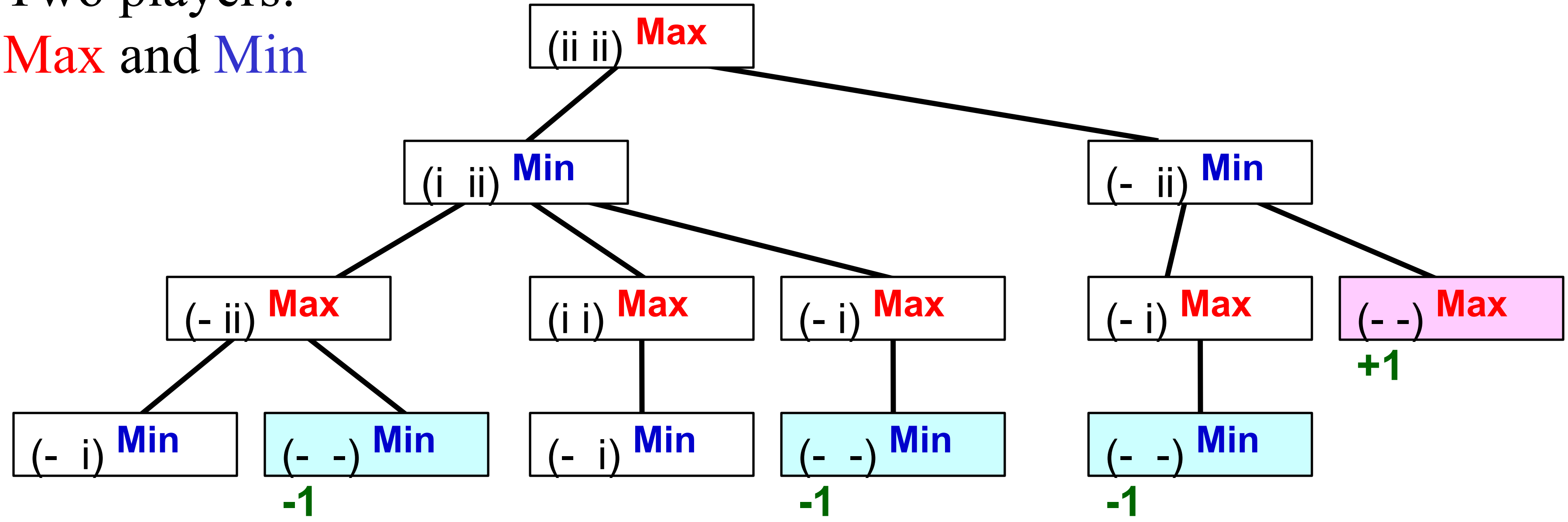
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

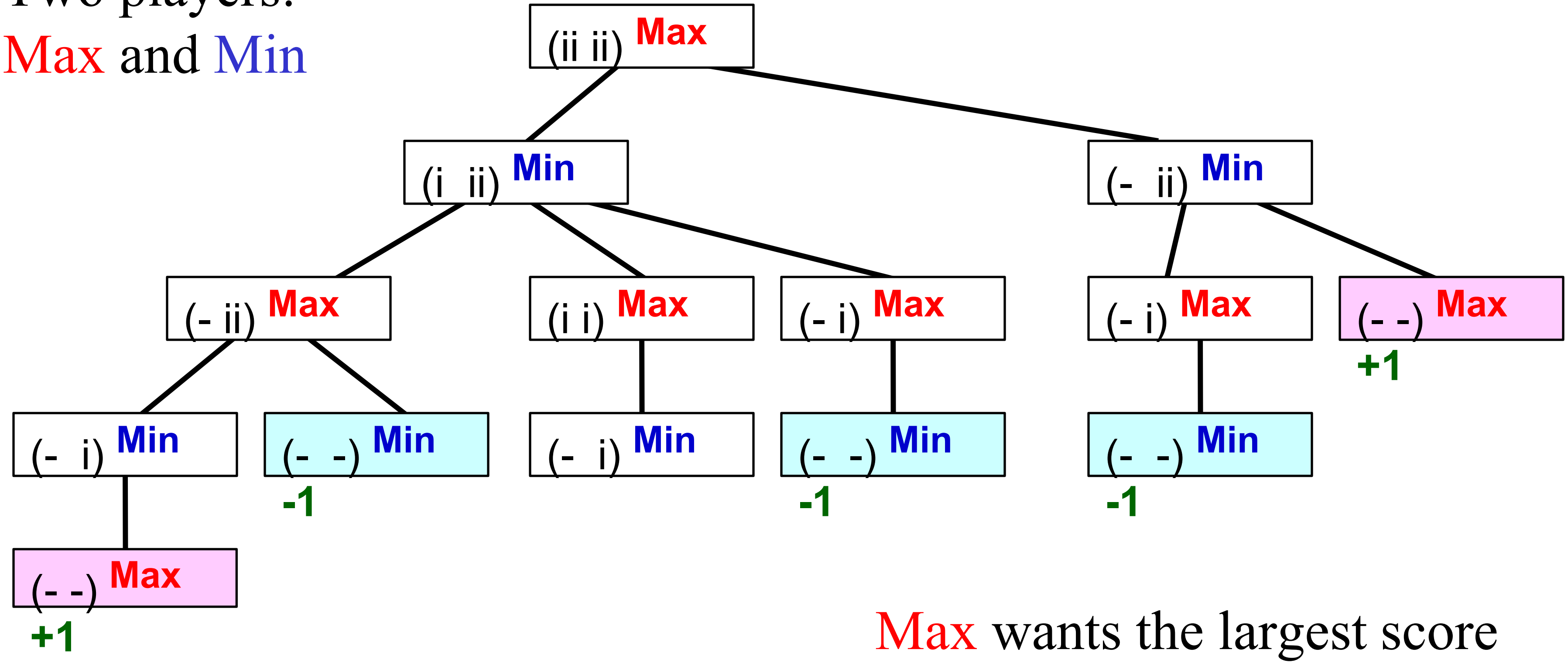
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

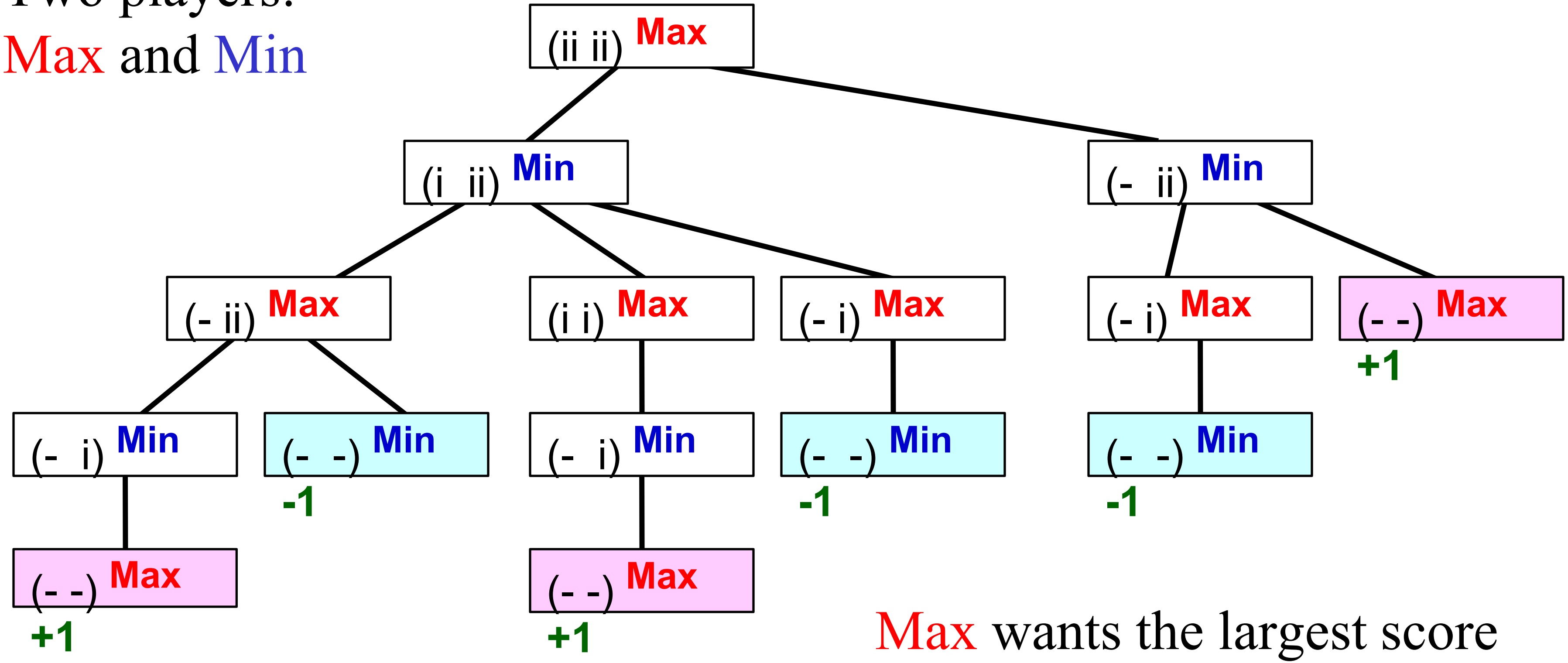
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

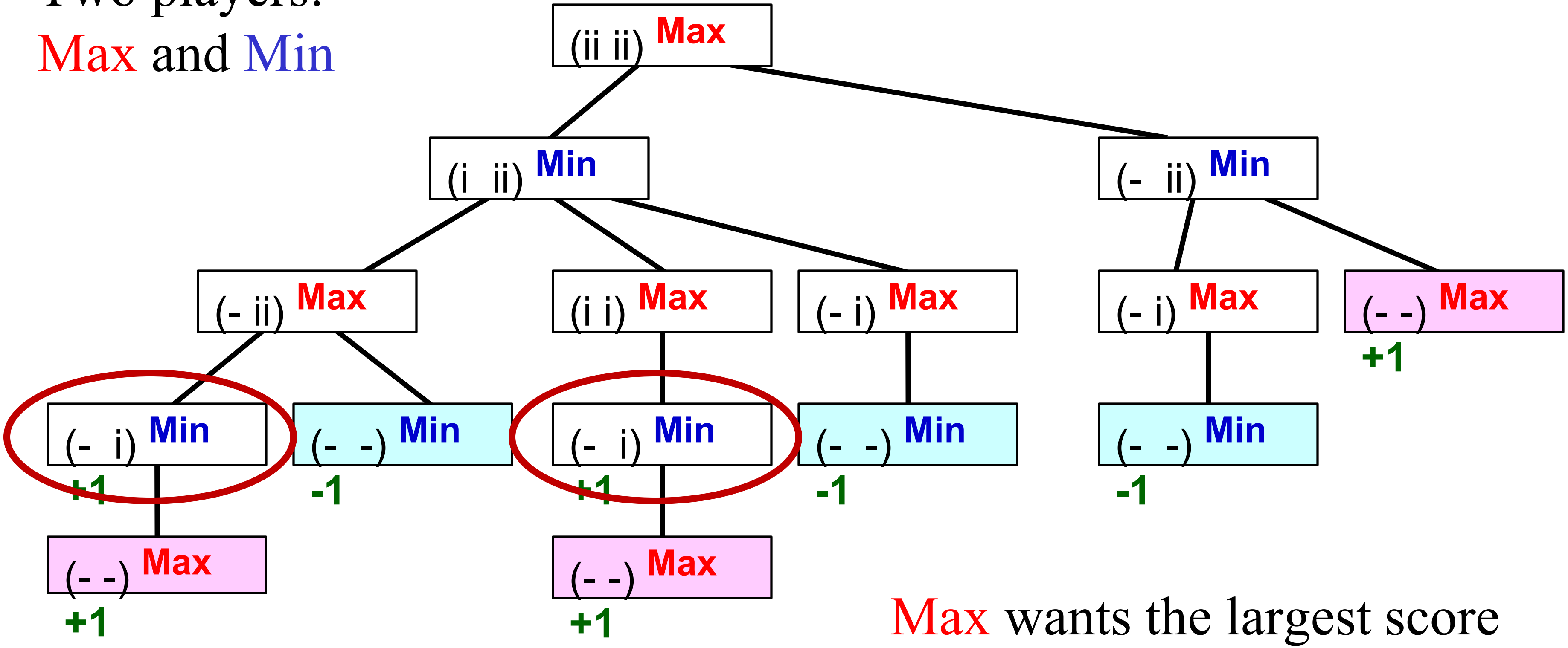
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

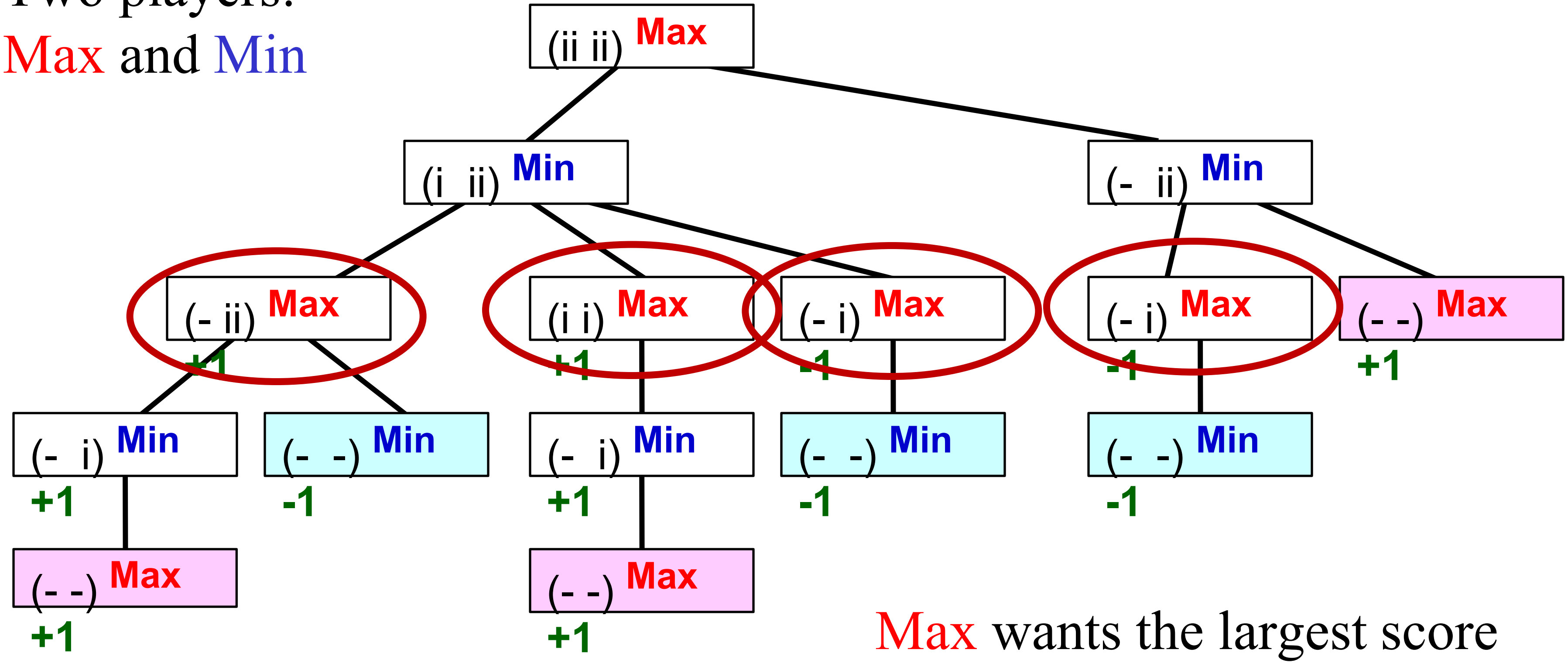
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

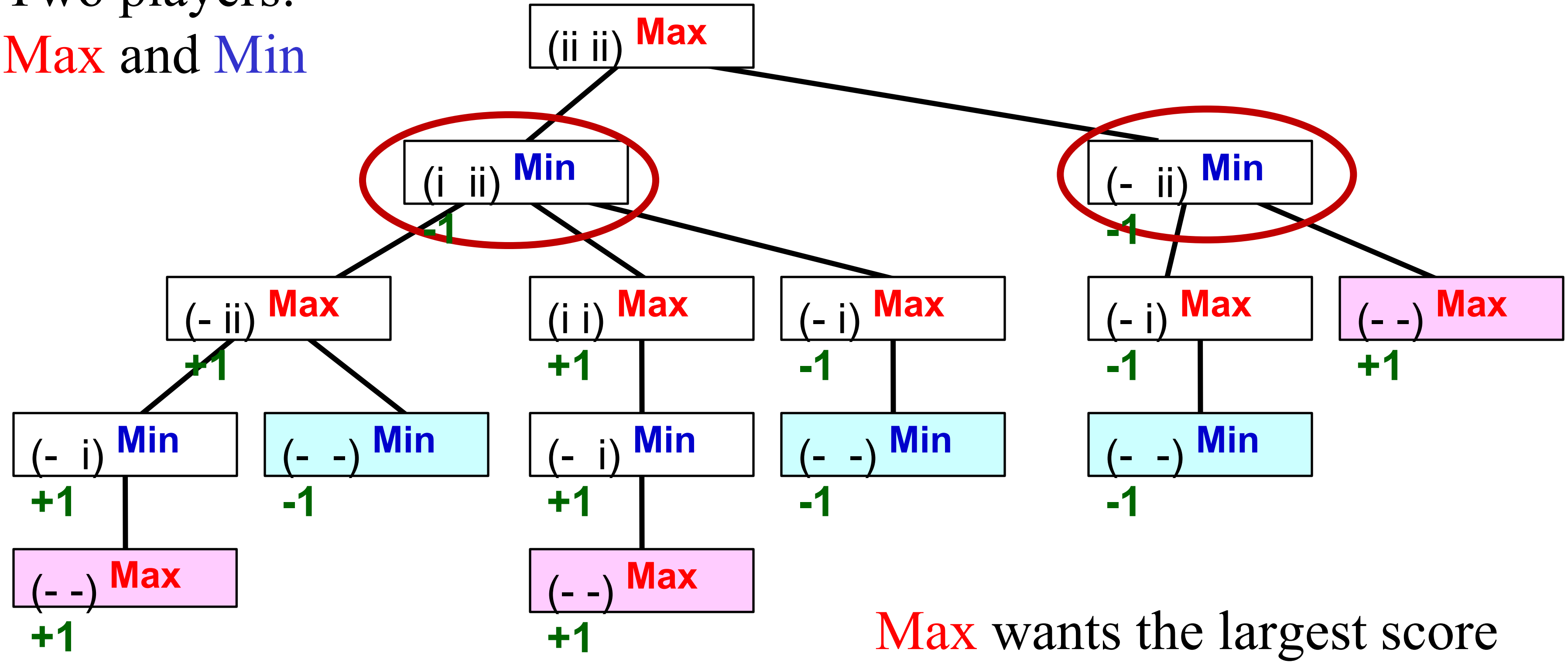
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

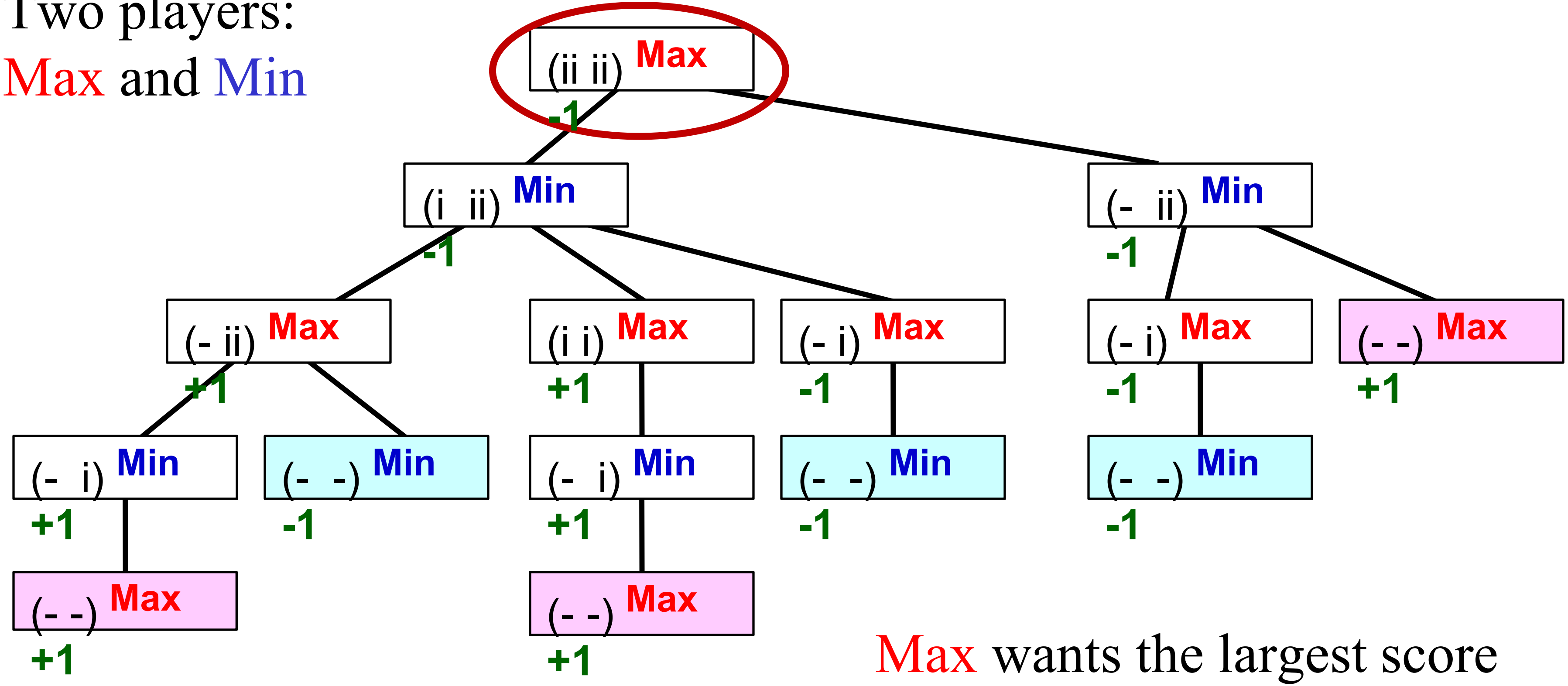
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

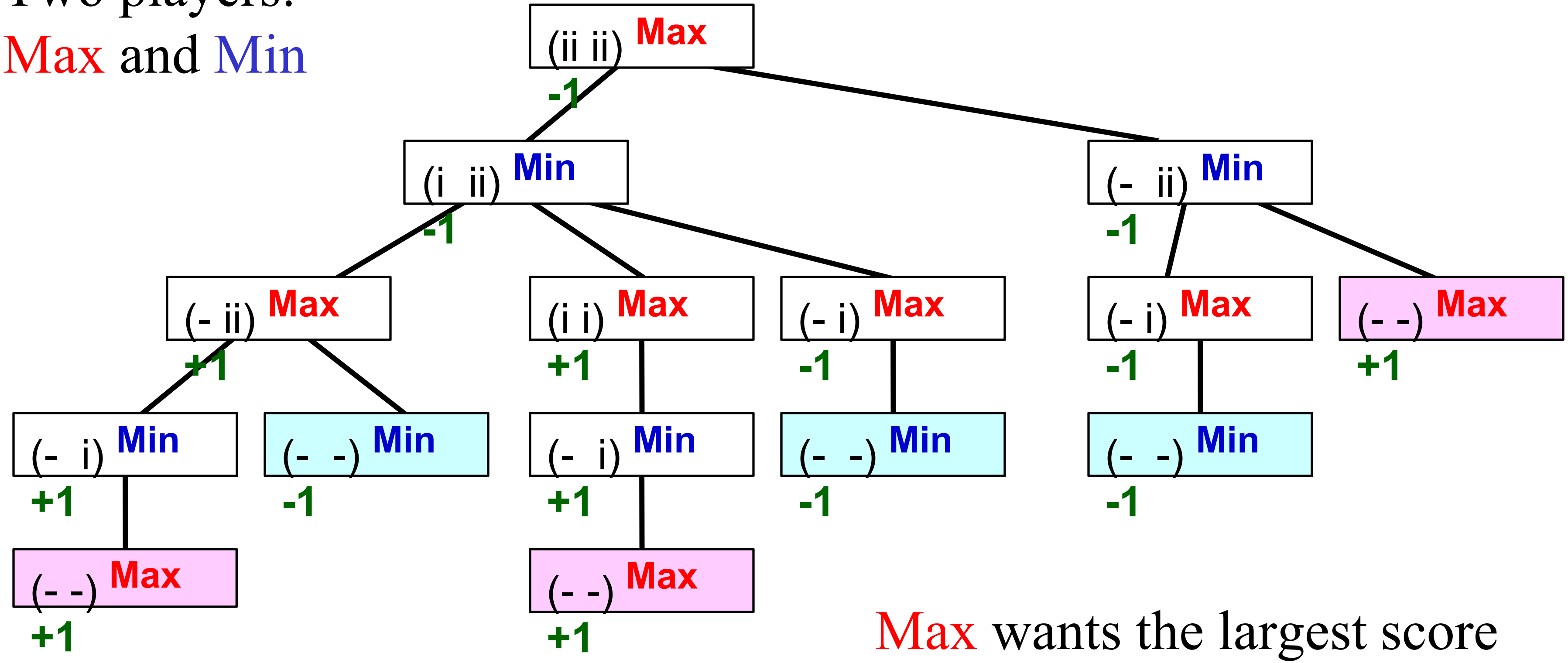
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

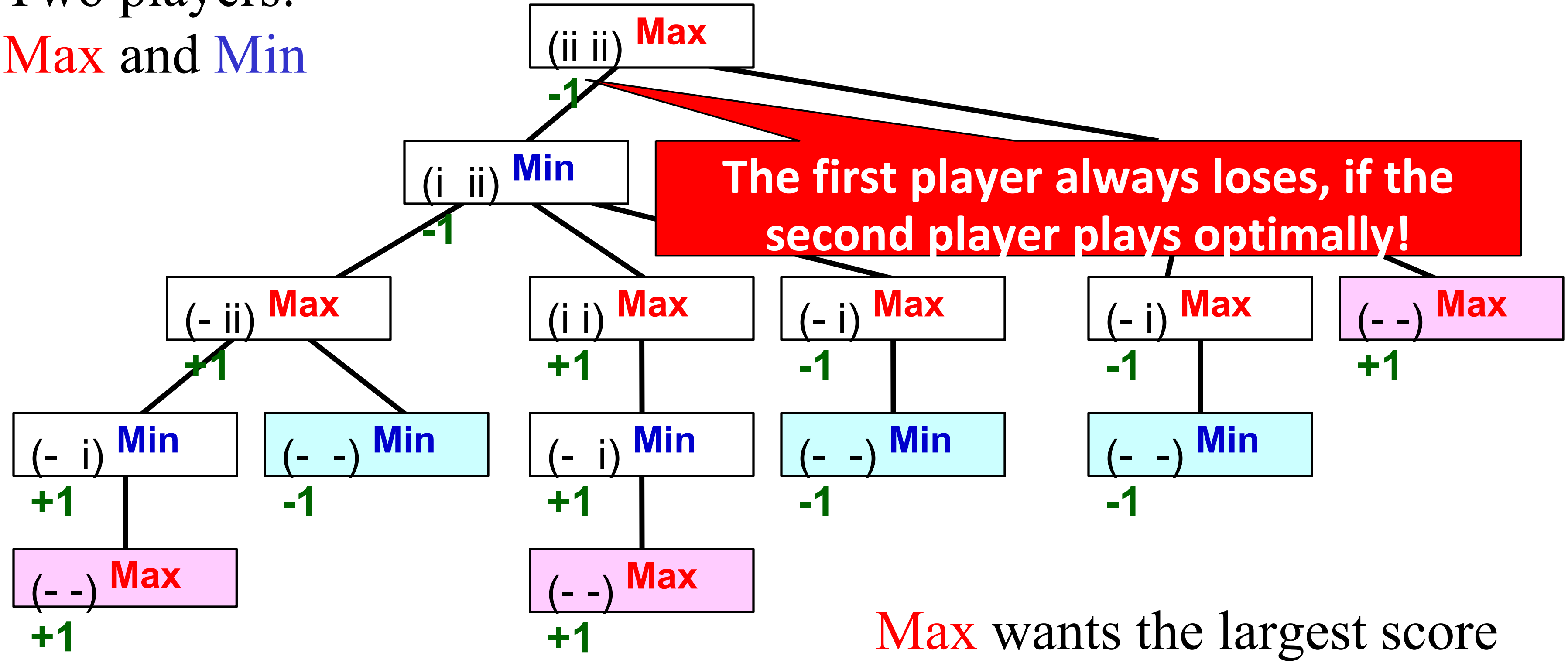
Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Game tree for II-Nim

Two players:
Max and Min



Max wants the largest score
Min wants the smallest score

Break & Quiz

Q 2.1: We are playing a game where Player A goes first and has 4 moves. Player B goes next and has 3 moves. Player A goes next and has 2 moves. Player B then has one move.

How many nodes are there in the minimax tree, including termination nodes (leaves)?

- A. 23
- B. 65
- C. 41
- D. 2

Break & Quiz

Q 2.1: We are playing a game where Player A goes first and has 4 moves. Player B goes next and has 3 moves. Player A goes next and has 2 moves. Player B then has one move.

How many nodes are there in the minimax tree, including termination nodes (leaves)?

- A. 23
- **B. 65**
- C. 41
- D. 2

Break & Quiz

Q 2.1: We are playing a game where Player A goes first and has 4 moves. Player B goes next and has 3 moves. Player A goes next and has 2 moves. Player B then has one move.

How many nodes are there in the minimax tree, including termination nodes (leaves)?

- A. 23
- **B. 65** (1 + 4 + 4*3 + 4*3*2 + 4*3*2 = 65. Note the root and leaf nodes.)
- C. 41
- D. 2

Our Approach So Far

We find the minimax value/strategy bottom up

- Minimax value: score of terminal node when both players play optimally
  - **Max's** turn, take max of children
  - **Min's** turn, take min of children

- Can implement this as depth-first search: **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 := – infinity
                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 := infinity
                for each s' in Succs(s)
                    β := min( β , Max-value(s'))
      return β
```

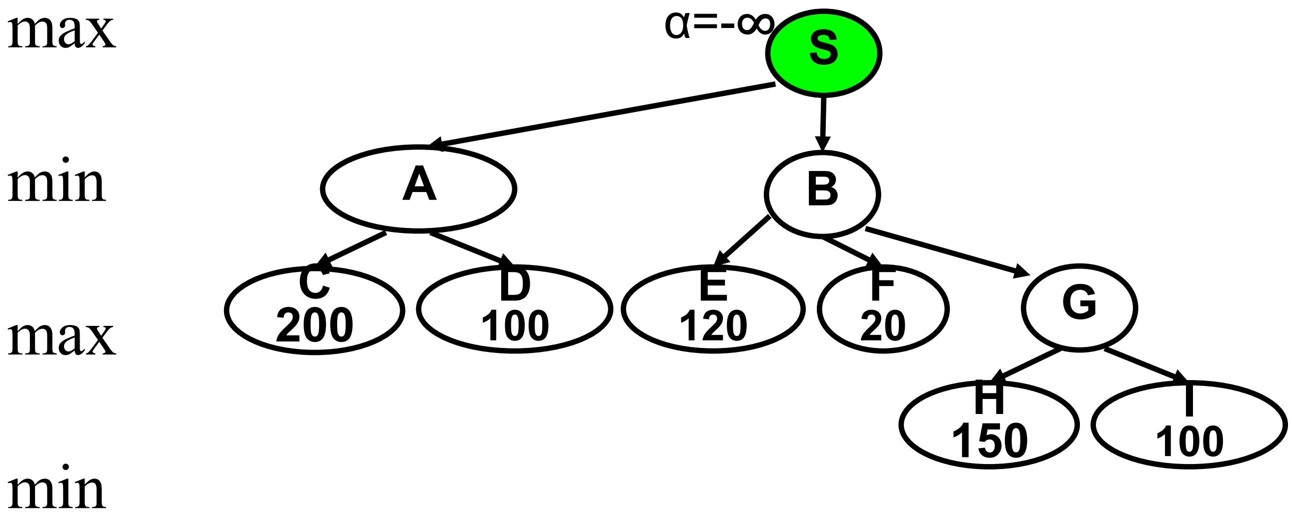
Time complexity?

- $O(b^m)$

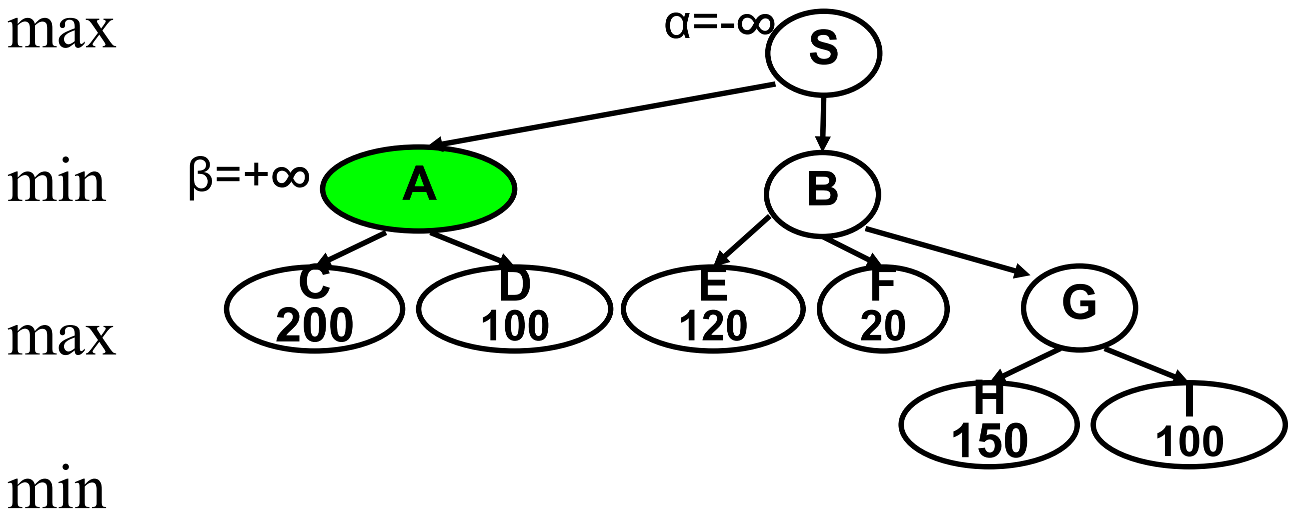
Space complexity?

- $O(bm)$

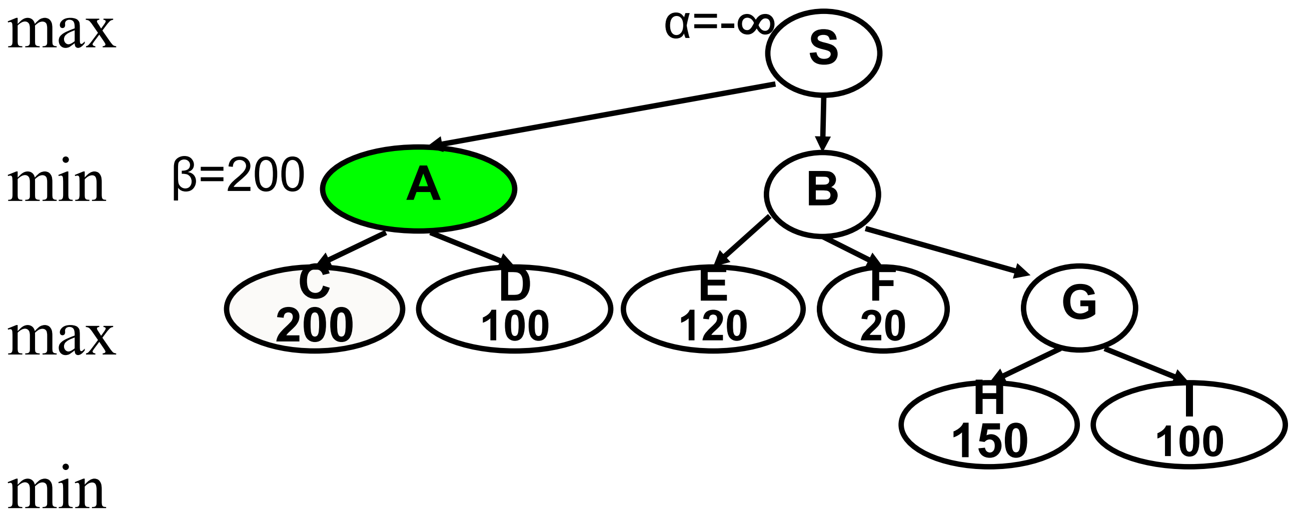
Minimax algorithm in execution



Minimax algorithm in execution

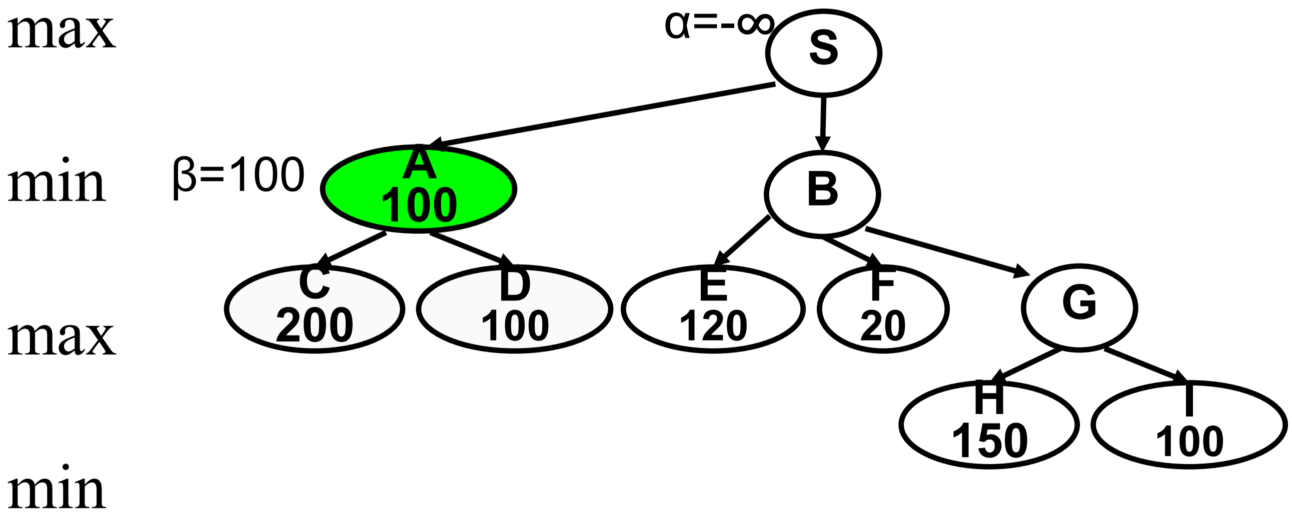


Minimax algorithm in execution

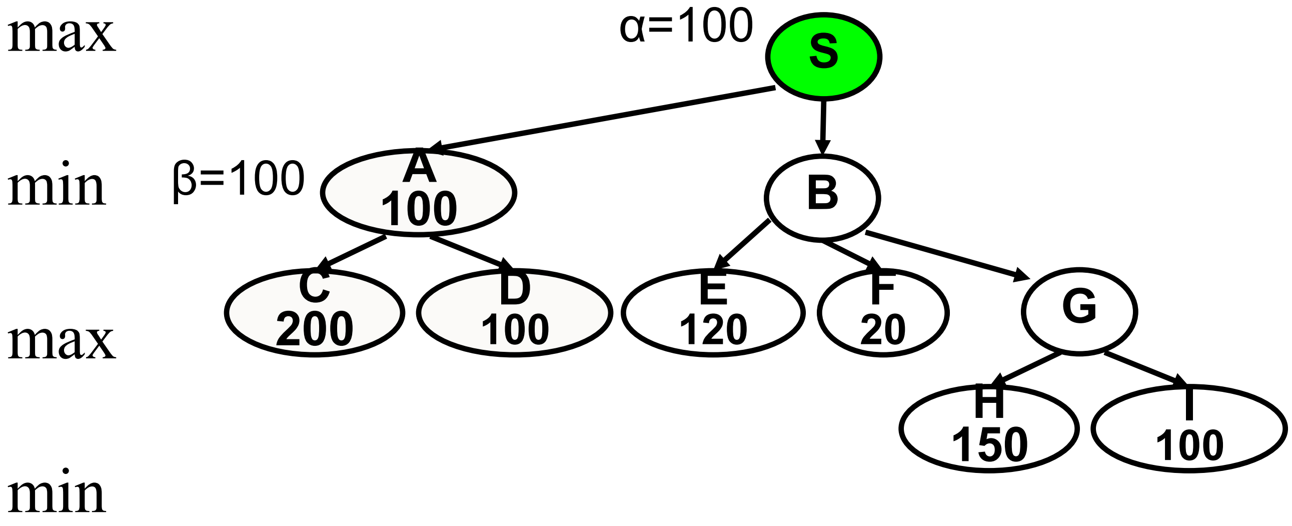


The execution on the terminal nodes is omitted.

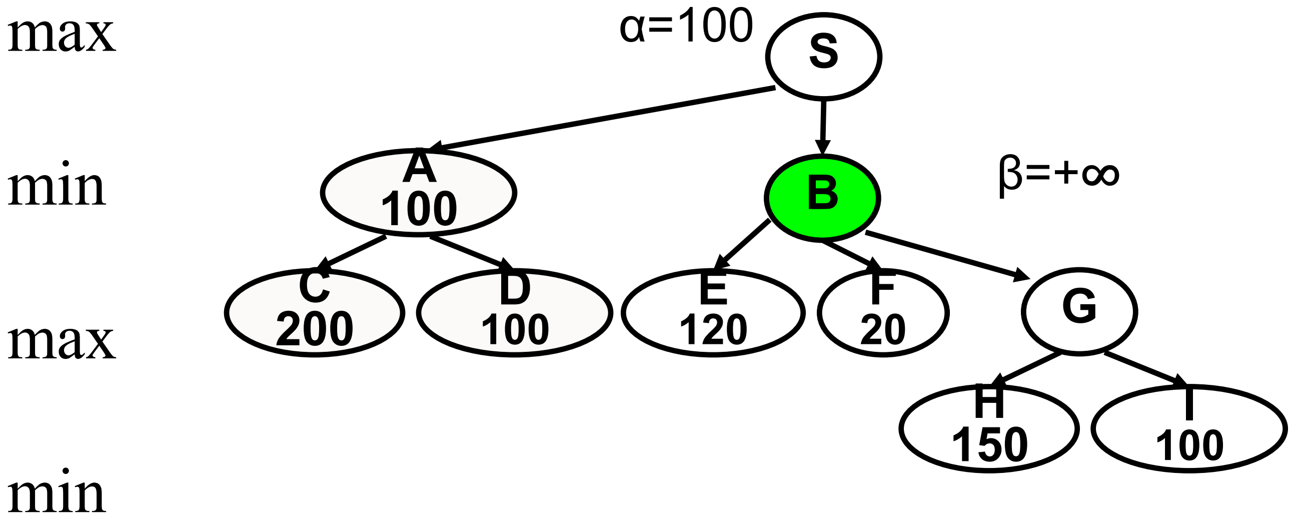
Minimax algorithm in execution



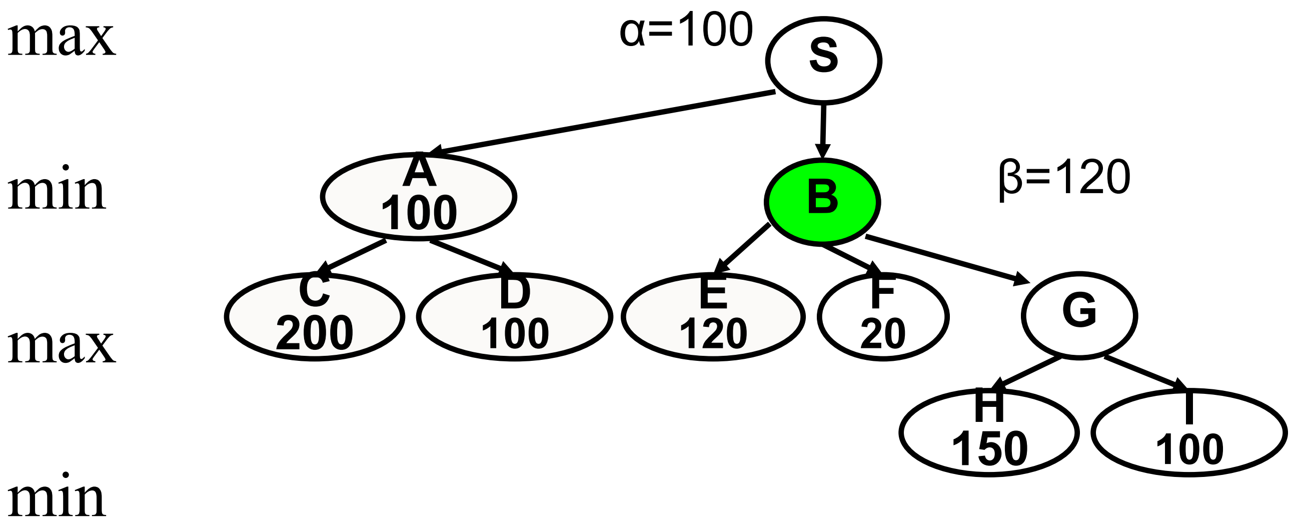
Minimax algorithm in execution



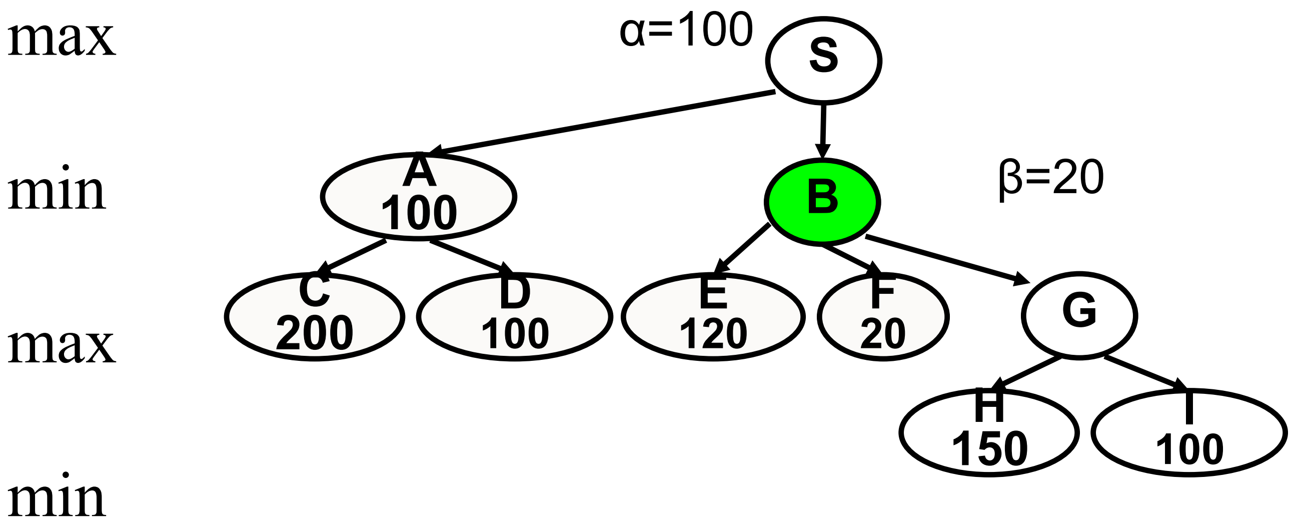
Minimax algorithm in execution



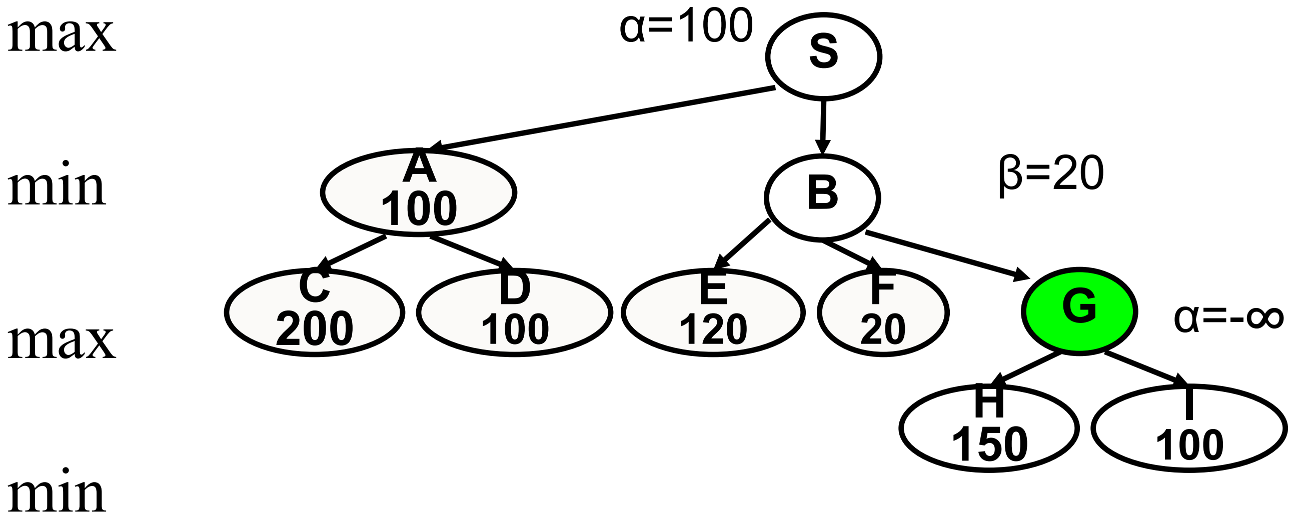
Minimax algorithm in execution



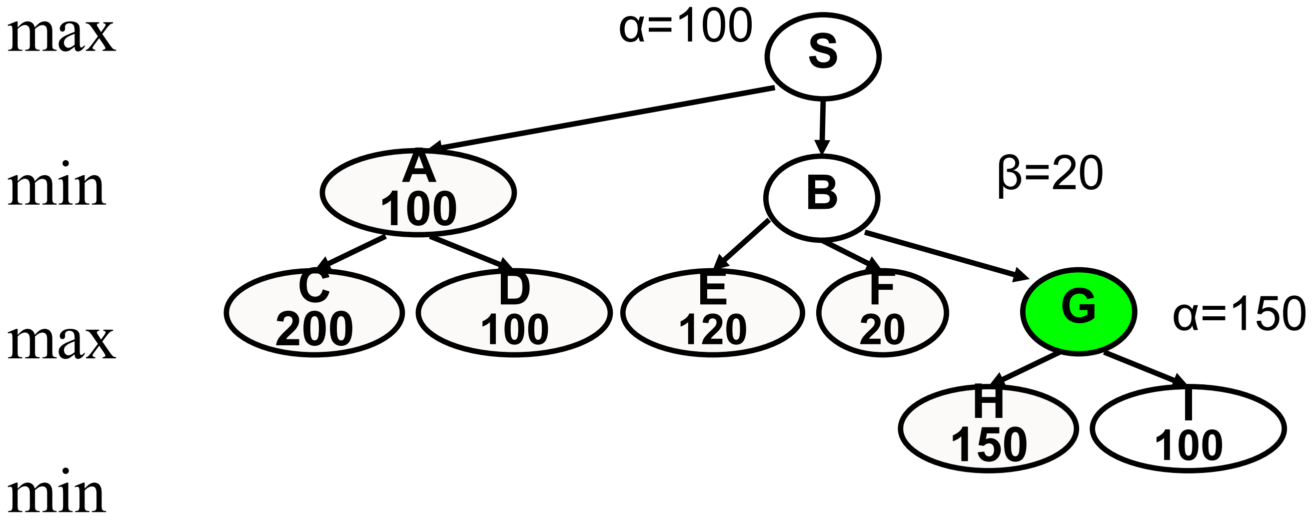
Minimax algorithm in execution



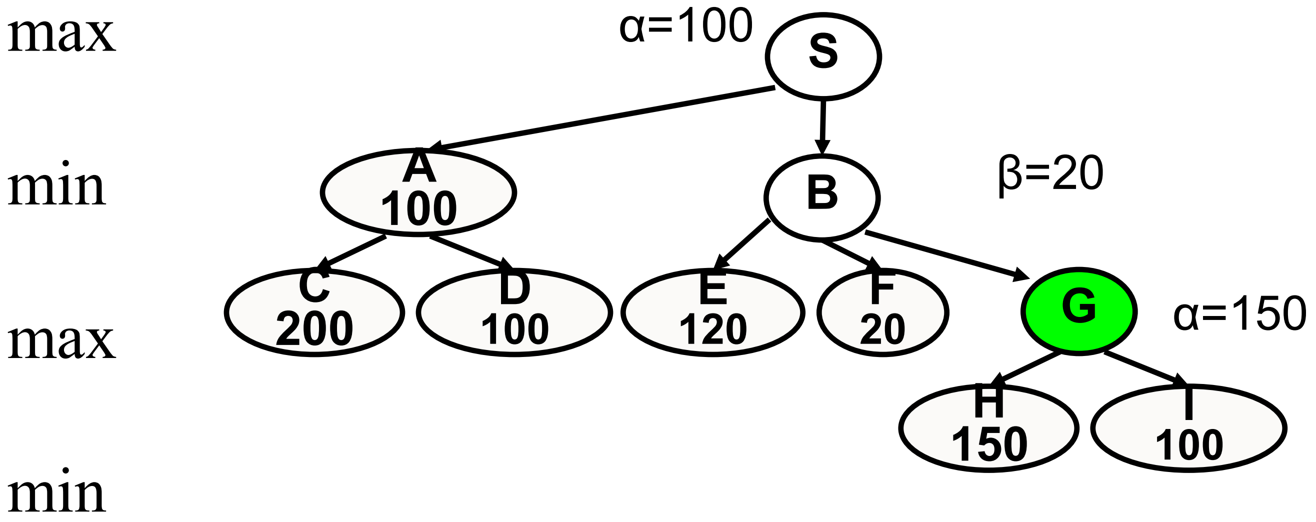
Minimax algorithm in execution



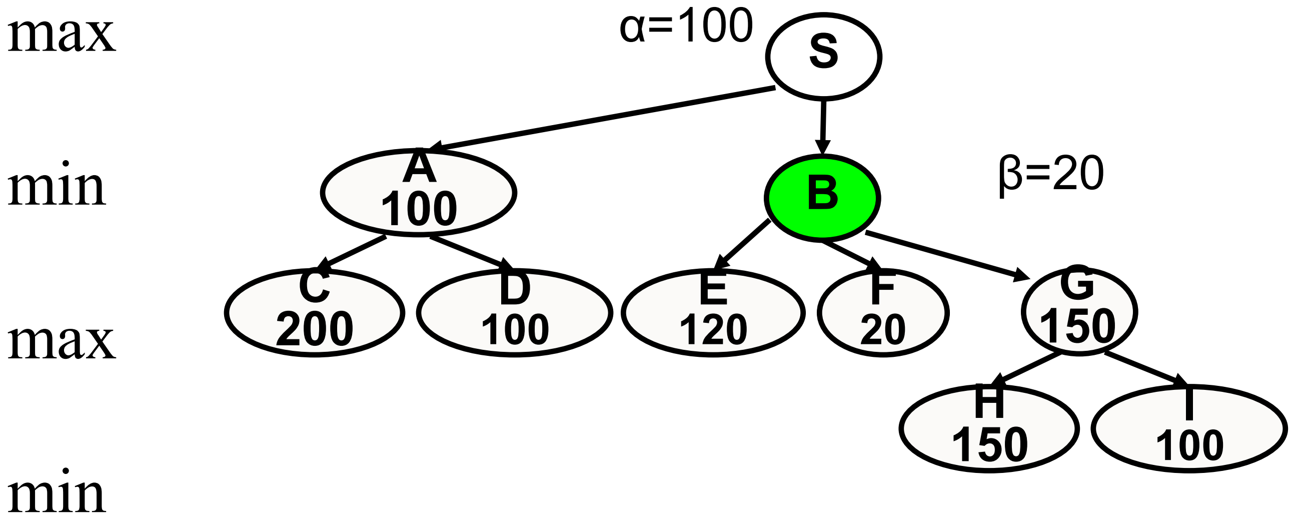
Minimax algorithm in execution



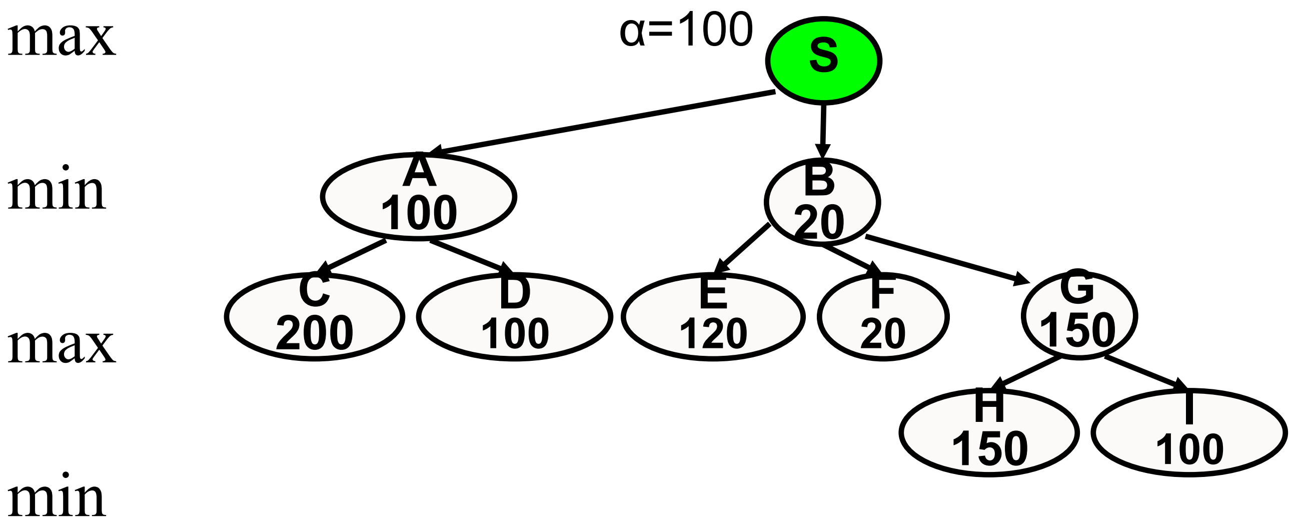
Minimax algorithm in execution



Minimax algorithm in execution



Minimax algorithm in execution



Break & Quiz

Q 2.2: During minimax tree search, must we examine every node?

- A. Always
- B. Sometimes
- C. Never

Break & Quiz

Q 2.2: During minimax tree search, must we examine every node?

- A. Always
- **B. Sometimes**
- C. Never

Break & Quiz

Q 2.2: During minimax tree search, must we examine every node?

- A. Always (No: consider layer *k*, where we take the max of all the mins of its children at layer *k+1*. If the current value of a min node at *k+1* already smaller than the current max, we don't need to continue the minimization.)
- **B. Sometimes**
- C. Never (No: the event above may simply not happen).

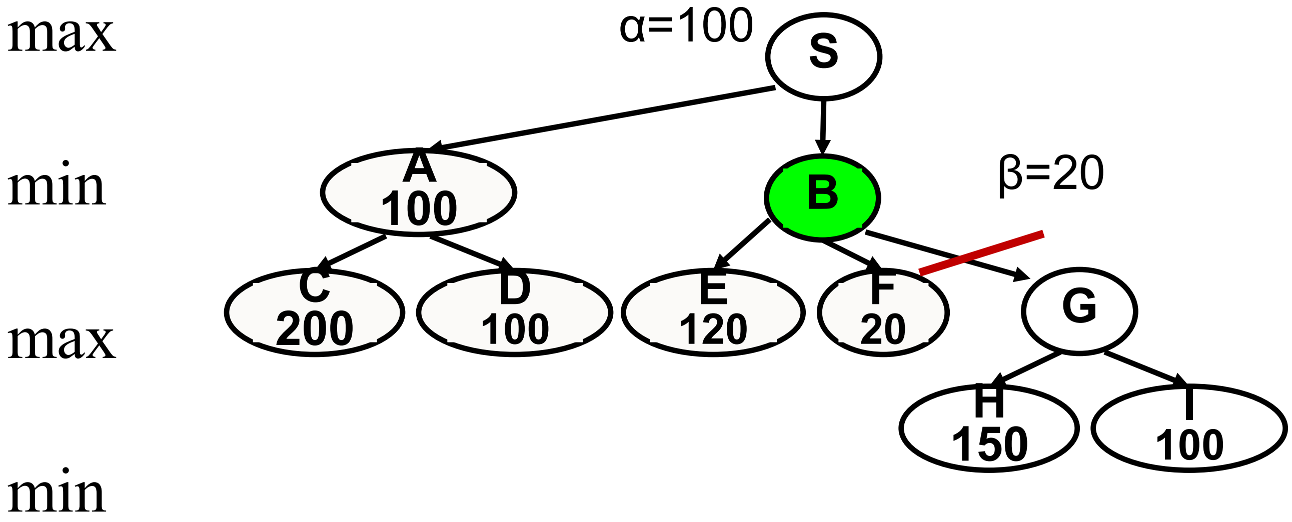
Can We Do Better?

One **downside**: we had to examine the entire tree

An idea to speed things up: **pruning**

- Goal: want the same minimax value, but faster
- We can get rid of bad branches
- Same principle as quiz question

Minimax algorithm in execution



Alpha-beta pruning

```
function Max-Value (s,α,β)
inputs:
    s: current state in game, Max about to play
    α: best score (highest) for Max along path to s
    β: best score (lowest) for Min along path to s
output: min(β , best-score (for Max) available from s)

    if ( s is a terminal state )
    then return ( terminal value of s )
    else for each s' in Succ(s)
        α := max( α , Min-value(s',α,β))
        if ( α ≥ β ) then return β   /* alpha pruning */
    return α
```

```
function Min-Value(s,α,β)
output: max(α , best-score (for Min) available from s )

    if ( s is a terminal state )
    then return ( terminal value of s)
    else for each s' in Succs(s)
        β := min( β , Max-value(s',α,β))
        if (α ≥ β ) then return α   /* beta pruning */
    return β
```

Starting from the root:

Max-Value(root, -∞, +∞)

Alpha-Beta Pruning

How effective is **alpha-beta pruning**?

- Depends on the order of successors!
  - Best case, the #of nodes to search is $O(b^{m/2})$
  - Happens when each player's best move is the leftmost child.
  - The worst case is no pruning at all.

- In DeepBlue, the average branching factor was about 6 with alpha-beta instead of 35-40 without.

Minimax With Heuristics

Note that long games may require huge computation

- To deal with this: limit ***d*** for the search depth
- **Q**: What to do at depth *d*, but no termination yet?
  - **A**: Use a heuristic evaluation function *e(x)*

```
function MINIMAX(x, d) returns an estimate of x's utility value
   inputs: x, current state in game
           d, an upper bound on the search depth
   if x is a terminal state then return Max's payoff at x
   else if d = 0 then return e(x)
   else if it is Max's move at x then
        return max{MINIMAX(y, d−1) : y is a child of x}
   else return min{MINIMAX(y, d−1) : y is a child of x}
```

Credit: Dana Nau

Heuristic Evaluation Functions

- e(x) can be any computable function of x; e.g. a weighted sum of features (like our linear models)

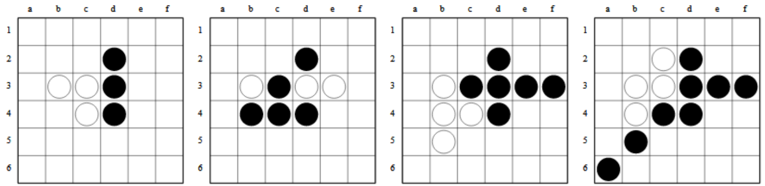
$$e(x) = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_n f_n(x)$$

- Chess example: $f_i(x)$ = **difference** between number of white and black, with *i* ranging over piece types.
  - Set weights according to piece importance
  - E.g., 1(# white pawns - # black pawns) + 3(#white knights - # black knights)

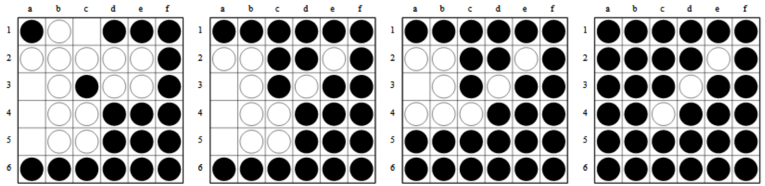
Going Further

- Monte Carlo tree search (MCTS)
  - Uses random sampling of the search space
  - Choose some children (heuristics to figure out #)
  - Record results, use for future play
  - Self-play

- AlphaGo and other big results!



The agent (Black) learns to capture walls and corners in the early game



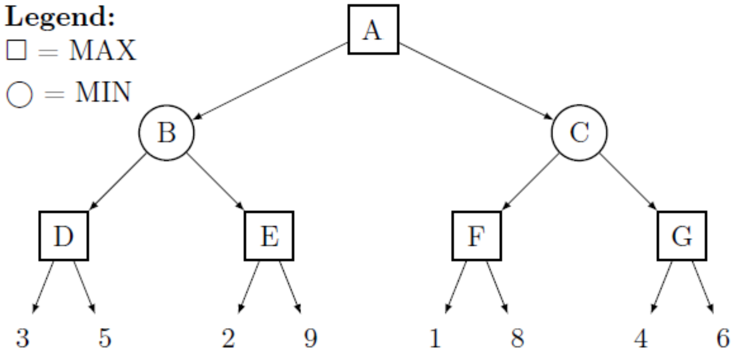
The agent (Black) learns to force passes in the late game

Credit: Surag Nair

Break & Quiz

Q 2.3 Consider the game tree shown in the figure below. The root node (A) represents the current state of the game where it is the MAX player's turn.
Assuming both players play optimally according to the Minimax algorithm, what is the value of the root node A?

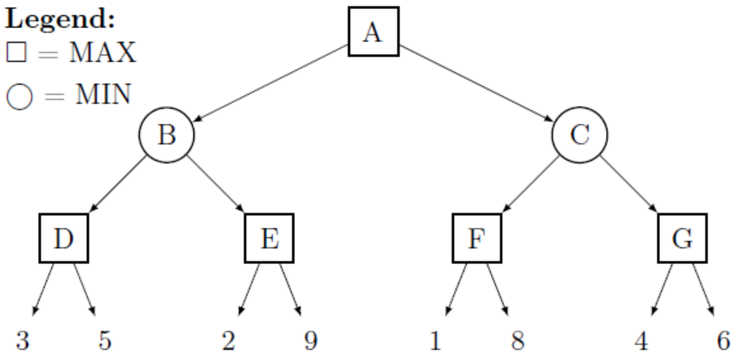
A. 4
B. 5
C. 6
D. 8



Break & Quiz

Q 2.3 Consider the game tree shown in the figure below. The root node (A) represents the current state of the game where it is the MAX player's turn.
Assuming both players play optimally according to the Minimax algorithm, what is the value of the root node A?

A. 4
B. 5
C. 6
D. 8



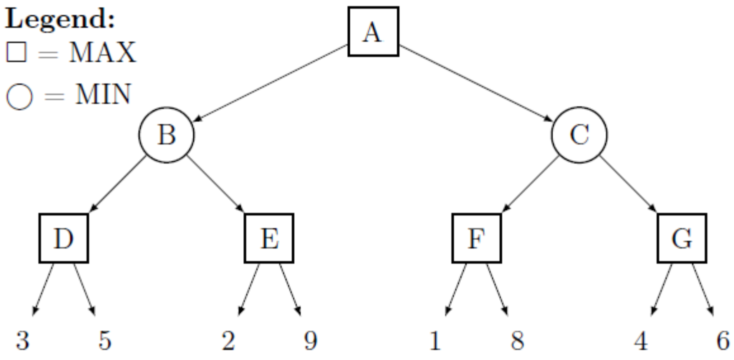
Break & Quiz

Q 2.3 Consider the game tree shown in the figure below. The root node (A) represents the current state of the game where it is the MAX player's turn.
Assuming both players play optimally according to the Minimax algorithm, what is the value of the root node A?

A. 4
B. 5
C. **6**
D. 8

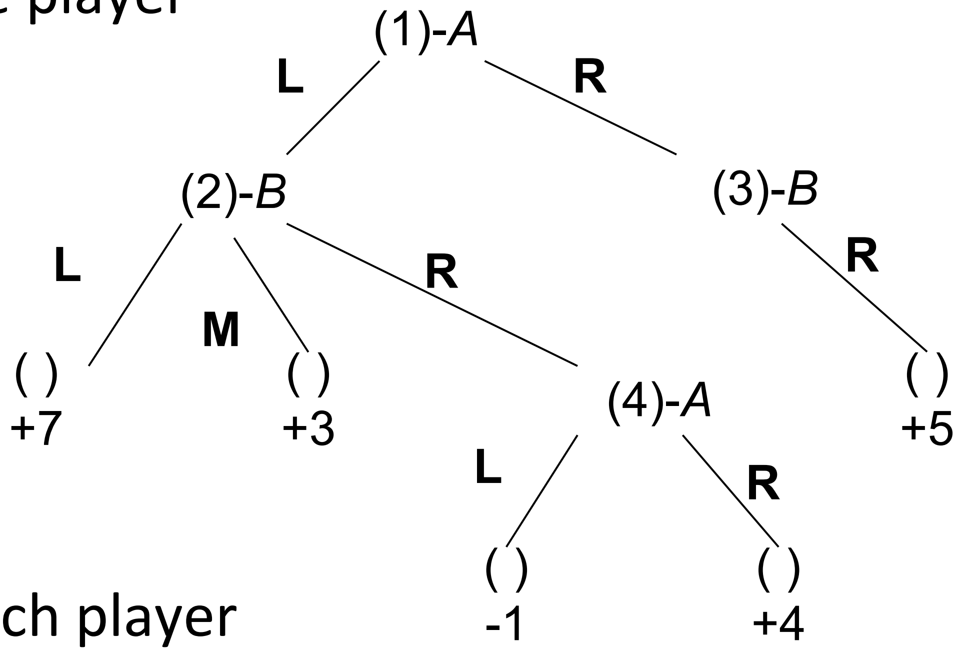
The Minimax algorithm propagates values from the leaf nodes up to the root.

1. Level 2 (MAX Nodes): The MAX player chooses the highest value among the children.
   - Node D: max(3, 5) = 5
   - Node E: max(2, 9) = 9
   - Node F: max(1, 8) = 8
   - Node G: max(4, 6) = 6
2. Level 1 (MIN Nodes): The MIN player chooses the lowest value among the children passed up from Level 2.
   - Node B: min(D,E) = min(5, 9) = 5
   - Node C: min(F,G) = min(8, 6) = 6
3. Level 0 (Root Node): The MAX player chooses the highest value among the children passed up from Level 1.
   - Node A: max(B,C) = max(5, 6) = 6



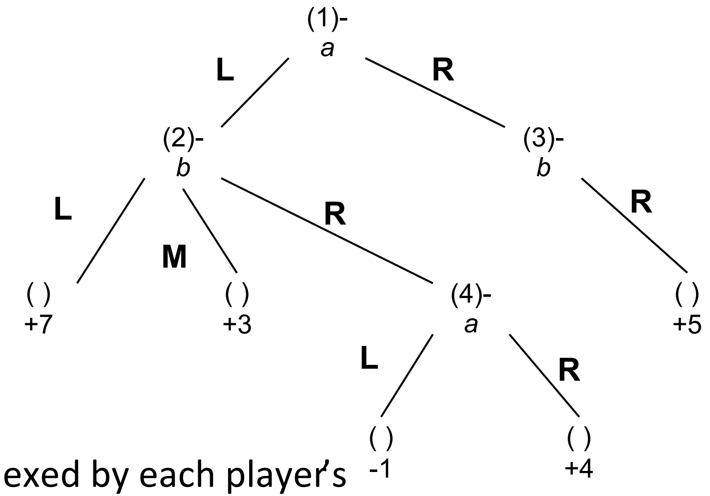
From Extensive Form back to Normal Form Game

- A pure strategy for a player is the mapping between all possible states the player can see, to the move the player would make.
- Player A has 4 pure strategies:
  - A's strategy I: (1→L, 4→L)
  - A's strategy II: (1→L, 4→R)
  - A's strategy III: (1→R, 4→L)
  - A's strategy IV: (1→R, 4→R)
- Player B has 3 pure strategies:
  - B's strategy I: (2→L, 3→R)
  - B's strategy II: (2→M, 3→R)
  - B's strategy III: (2→R, 3→R)
- How many pure strategies if each player can see *N* states, and has *b* moves at each state?



Matrix Normal Form of games

A's strategy I: (1→L, 4→L)
A's strategy II: (1→L, 4→R)
A's strategy III: (1→R, 4→L)
A's strategy IV: (1→R, 4→R)
B's strategy I: (2→L, 3→R)
B's strategy II: (2→M, 3→R)
B's strategy III: (2→R, 3→R)

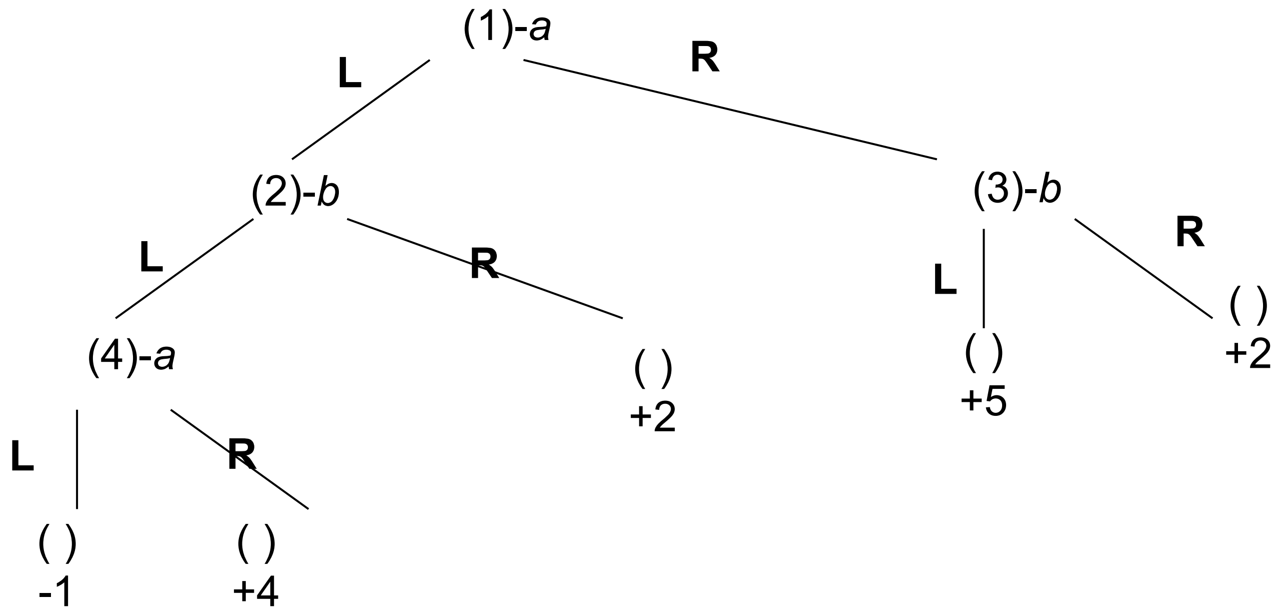


The matrix normal form is the game value matrix indexed by each player's strategies.

	B-I	B-II	B-III
A-I	7	3	-1
A-II	7	3	4
A-III	5	5	5
A-IV	5	5	5

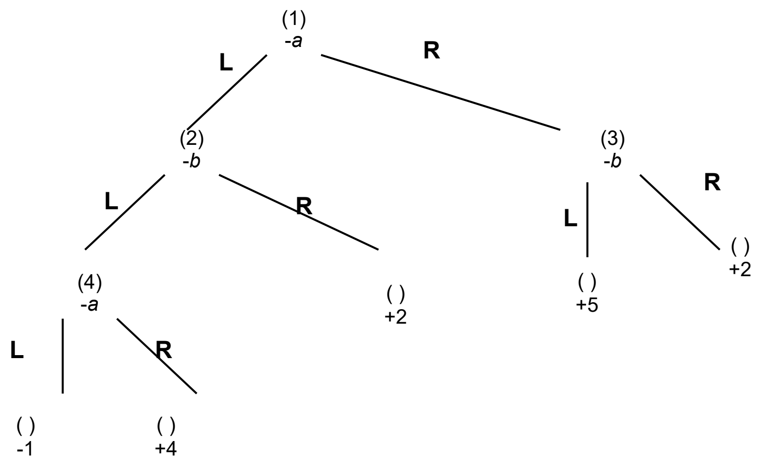
The matrix encodes every outcome of the game! The rules etc. are no longer needed.

Another example of normal form



- How many pure strategies does A have?
- How many does B have?
- What is the matrix form of this game?

Matrix normal form example



	B-I	B-II	B-III	B-IV
A-I	-1	-1	2	2
A-II	4	4	2	2
A-III	5	2	5	2
A-IV	5	2	5	2

- How many pure strategies does A have? 4

A-I (1→L, 4→L) A-II (1→L,4→R) A-III (1→R,4→L) A-IV (1→R, 4→R)

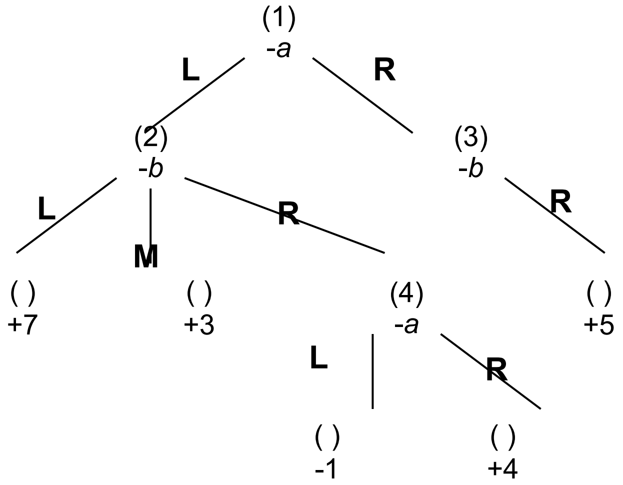
- How many does B have? 4

B-I (2→L, 3→L) B-II (2→L,3→R) B-III (2→R,3→L) B-IV (2→R, 3→R)

- What is the matrix form of this game?

Minimax in Matrix Normal Form

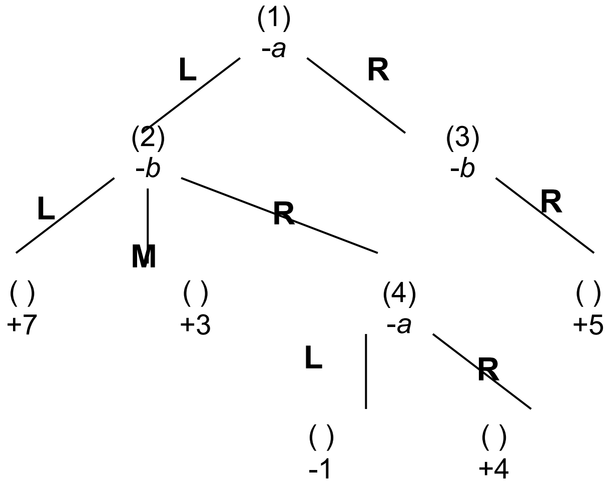
- Player A: for each strategy, consider all B's counter strategies (a row in the matrix), find the minimum value in that row. Pick the row with the maximum minimum value.
- Here maximin=5



	B-I	B-II	B-III
A-I	7	3	-1
A-II	7	3	4
A-III	5	5	5
A-IV	5	5	5

Minimax in Matrix Normal Form

- Player B: find the maximum value in each column. Pick the column with the minimum maximum value.
- Here minimax = 5



Fundamental game theory result (proved by von Neumann):

In a 2-player, zero-sum game of perfect information (sequential moves), Minimax==Maximin. And there always exists an optimal pure strategy for each player.

	B-I	B-II	B-III
A-I	7	3	-1
A-II	7	3	4
A-III	5	5	5
A-IV	5	5	5

Minimax in Matrix Normal Form

Interestingly, A can tell B in advance what strategy A will use (the maximin), and this information will not help B!

Similarly B can tell A what strategy B will use.

In fact A knows what B's strategy will be.

And B knows A's too.

And A knows that B knows

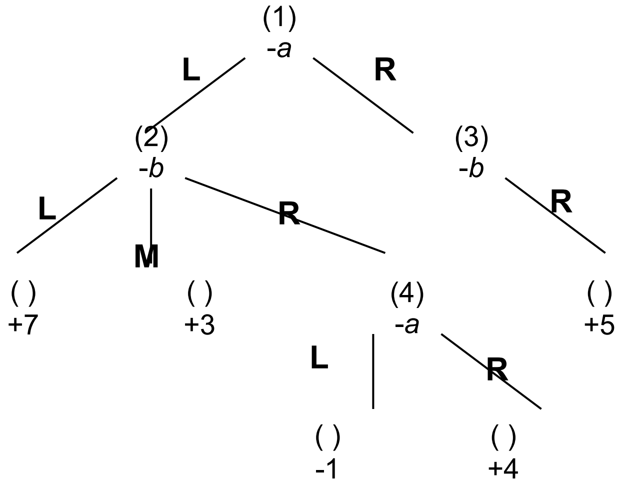
…

The game is at an equilibrium

ult

ame

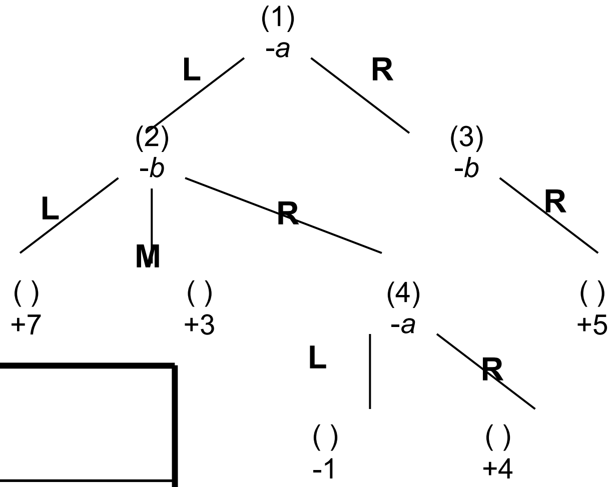
…al pure strategy for each player.



	B-I	B-II	B-III
A-I	7	3	-1
A-II	7	3	4
A-III	5	5	5
A-IV	5	5	5

Minimax in Matrix Normal Form

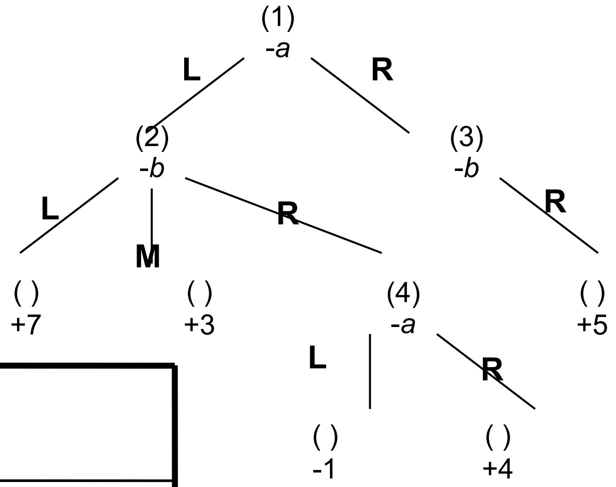
- We can also check for mutual best responses



	B-I	B-II	B-III
A-I	7	3	-1
A-II	7	3	4
A-III	5	5	5
A-IV	5	5	5

Minimax in Matrix Normal Form

- We can also check for mutual best responses



	B-I	B-II	B-III
A-I	7	3	-1
A-II	7	3	4
A-III	5	5	5
A-IV	5	5	5

Suggested Readings

Textbook: Artificial Intelligence: A Modern Approach (4th edition). Stuart Russell and Peter Norvig. Pearson, 2020.

- Chapters 5, 18