
DRONACHARYA
College of Engineering

INTELLIGENT SYSTEMS (CSE-303-F)

Section A

# Alpha Beta Pruning

# Motivations

- Breadth-first, depth-first, hill-climbing, best-first, and A* assume a non-hostile search space.
- The goals just sit there somewhere in the graph.
- The goals do not try to elude you.
- The 8-puzzle game did not try to stop you from reaching the goal.
- Your tic-tac-toe opponents reacted to your moves randomly.
- But in a real 2-person game, you opponent does try to beat you and make it difficult for you to reach your goal.
- Minimax search can be applied in an adversarial search space.
- Alpha-beta pruning can be used to cut bad branches (moves) in the game tree to improve minimax search time.

# Objectives

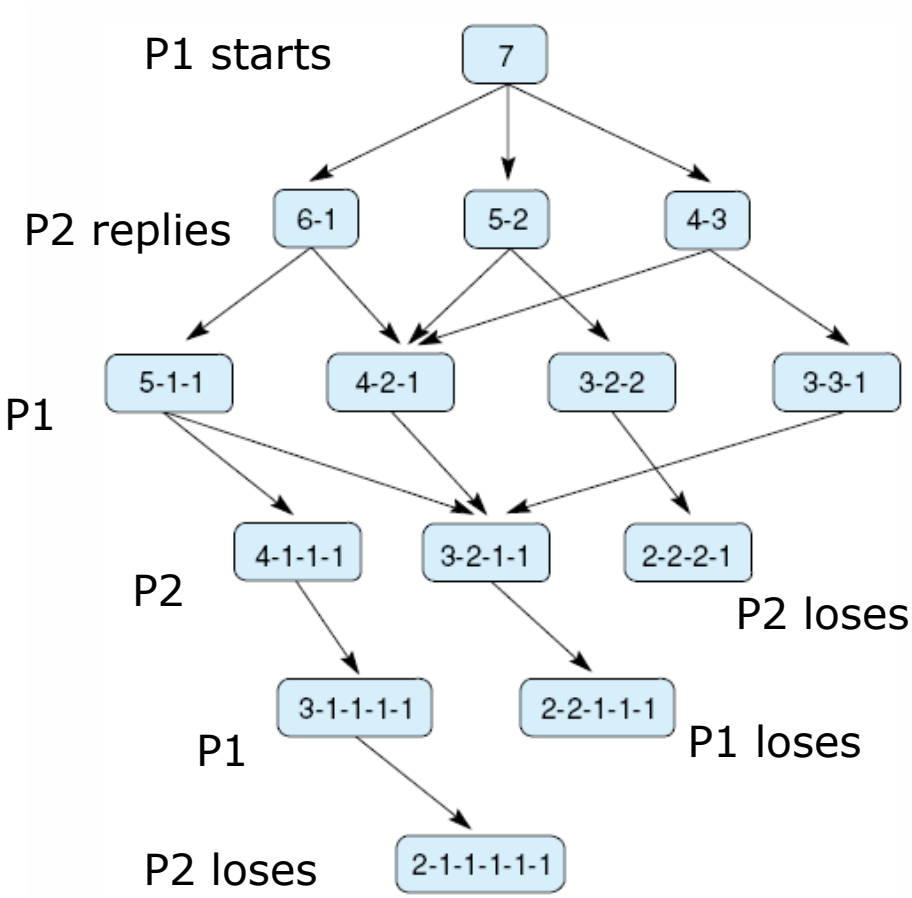
1. Adversarial search space: MAX vs. MIN
2. A simple game tree: Nim-7
3. Minimax on Nim-7
4. Minimax on tic-tac-toe looking 3 plies ahead
5. Alpha-beta pruning

# Two people games

- Solved games
  - Tic-tac-toe
  - Four In A Line
  - Checkers
- Impressive games played by robots
  - Othello bot is much stronger than any human player
  - Computer chess beat the human world champions
  - TD-Gammon ranked among top 3 players in the backgammon world
- Future bot challenges to humans
  - Poker bots play respectfully at world-class level
  - Computer bridge programs play competitively at national level
  - Go bots are getting more serious in the amateur ranking

# Complete game tree for Nim-7

- 7 coins are placed on a table between the two opponents
- A move consists of dividing a pile of coins into two nonempty piles of different sizes
- For example, 6 coins can be divided into piles of 5 and 1 or 4 and 2, but not 3 and 3
- The first player who can no longer make a move loses the game

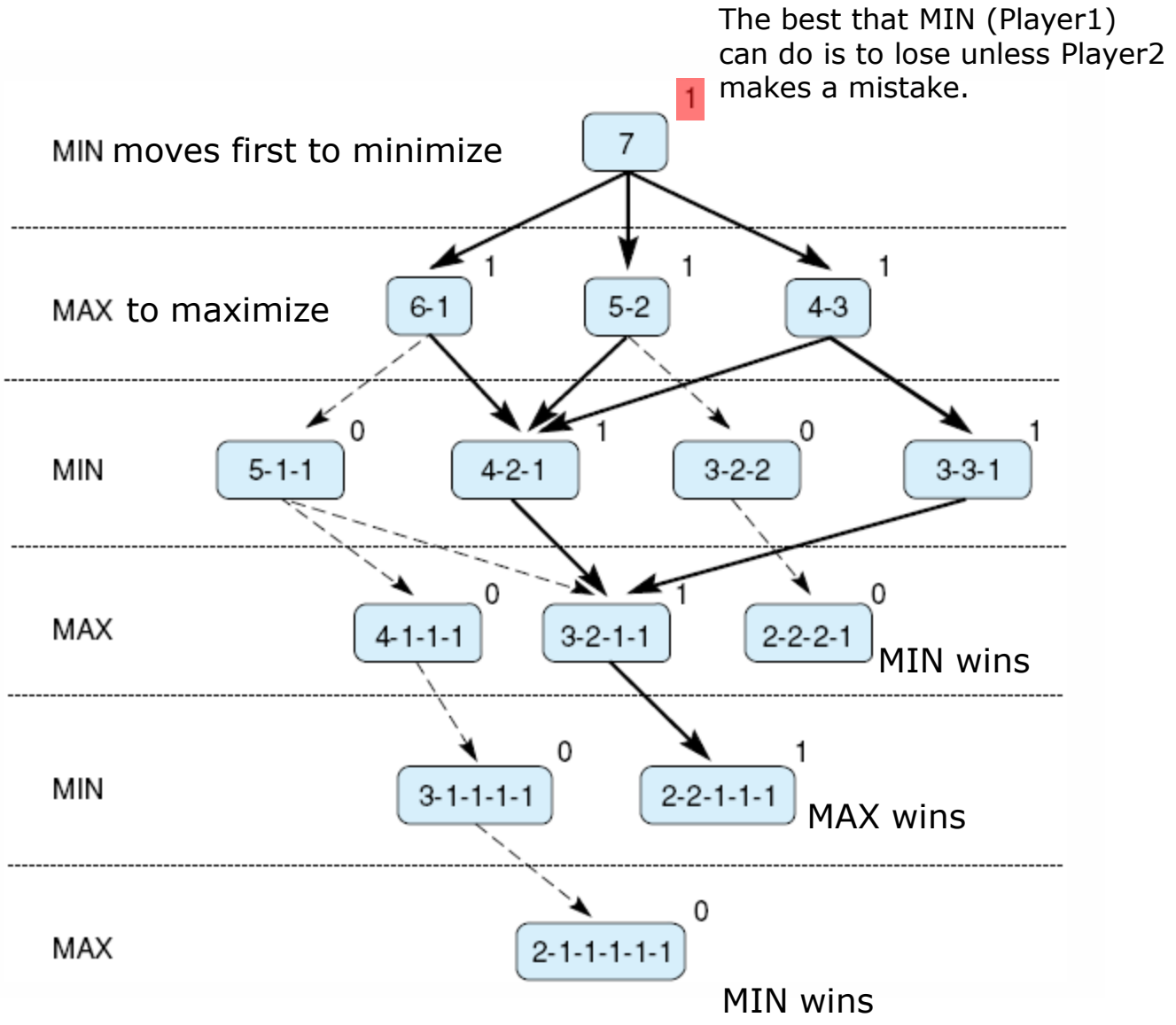


# MIN vs. MAX in a Nim game

Node score = 0 means MIN wins.

1 means MAX wins.

Bold edges indicate forced win for MAX, Player2.

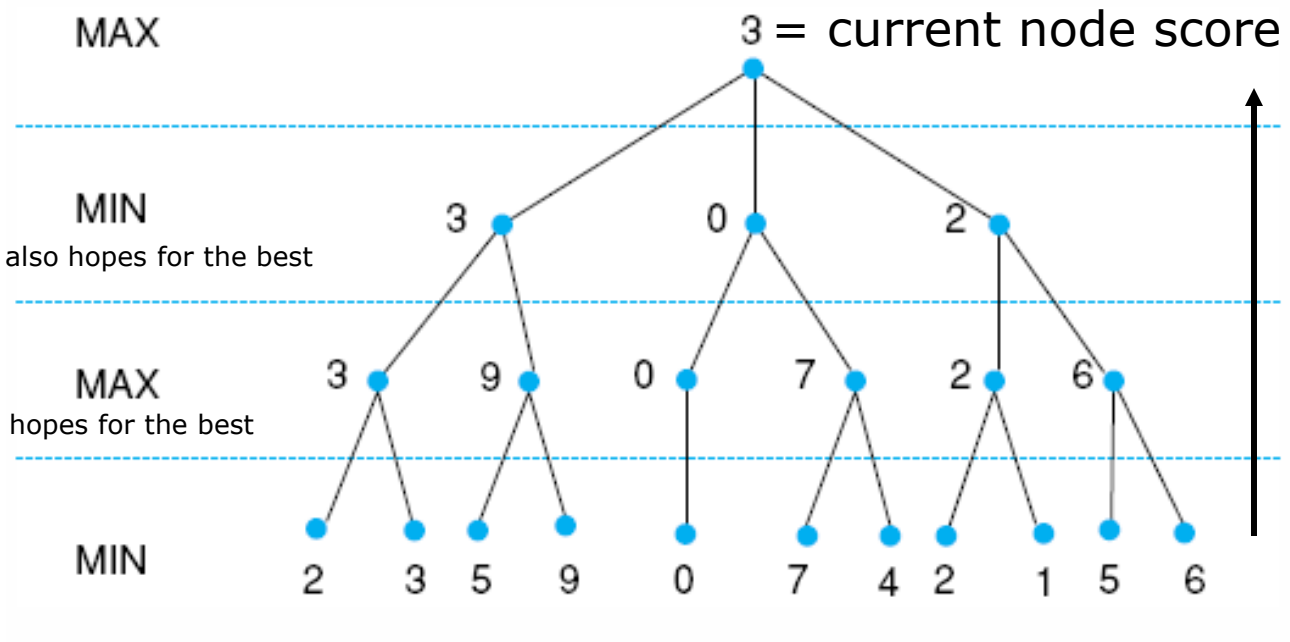


# Minimax to fixed ply depth

- Instead of Nim-7, image the chess game tree.
- Chess game tree is too deep.
  - cannot expand the current node to terminating (leaf) nodes for checkmate.
- Use fixed ply depth look-ahead.
  - Search from current position to all possible positions that are, e.g., 3-plies ahead.
- Use heuristic to evaluate all these future positions.
  - P=1, N=B=3, R=5, Q=9
  - Assign certain weight to certain features of the position (dominance of the center, mobility of the queen, etc.)
  - summarize these factors into a single number.
- Then propagating the scores back to the current node.

# MAX calculates the current node score

Look 3 plies ahead.

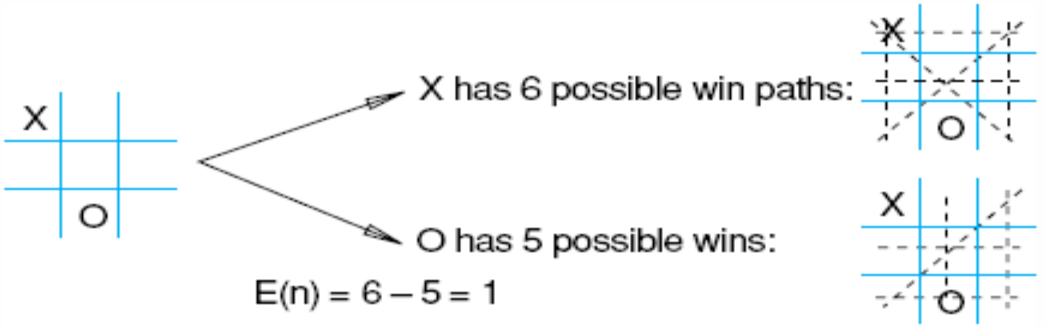



Back propagate the scores up the tree.

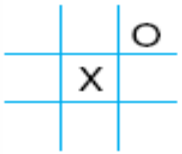
Use heuristic h(n) for each of these future positions.

A stronger heuristic will beat a weaker heuristic.
A farther look-ahead will beat a near-sighted look-ahead.
Computer chess routinely uses complex heuristics analyzing material and positional advantages and looks 40 plies ahead.

# Heuristic measuring for adversarial tic-tac-toe








Heuristic is E(n) = M(n) – O(n)
where M(n) is the total of My possible winning lines
O(n) is total of Opponent's possible winning lines
E(n) is the total Evaluation for state n

Maximize E(n)

E(n) = 0 when my opponent and I have equal number of possibilities.

# Tic-tac-toe, MAX vs MIN, 2-ply look-ahead

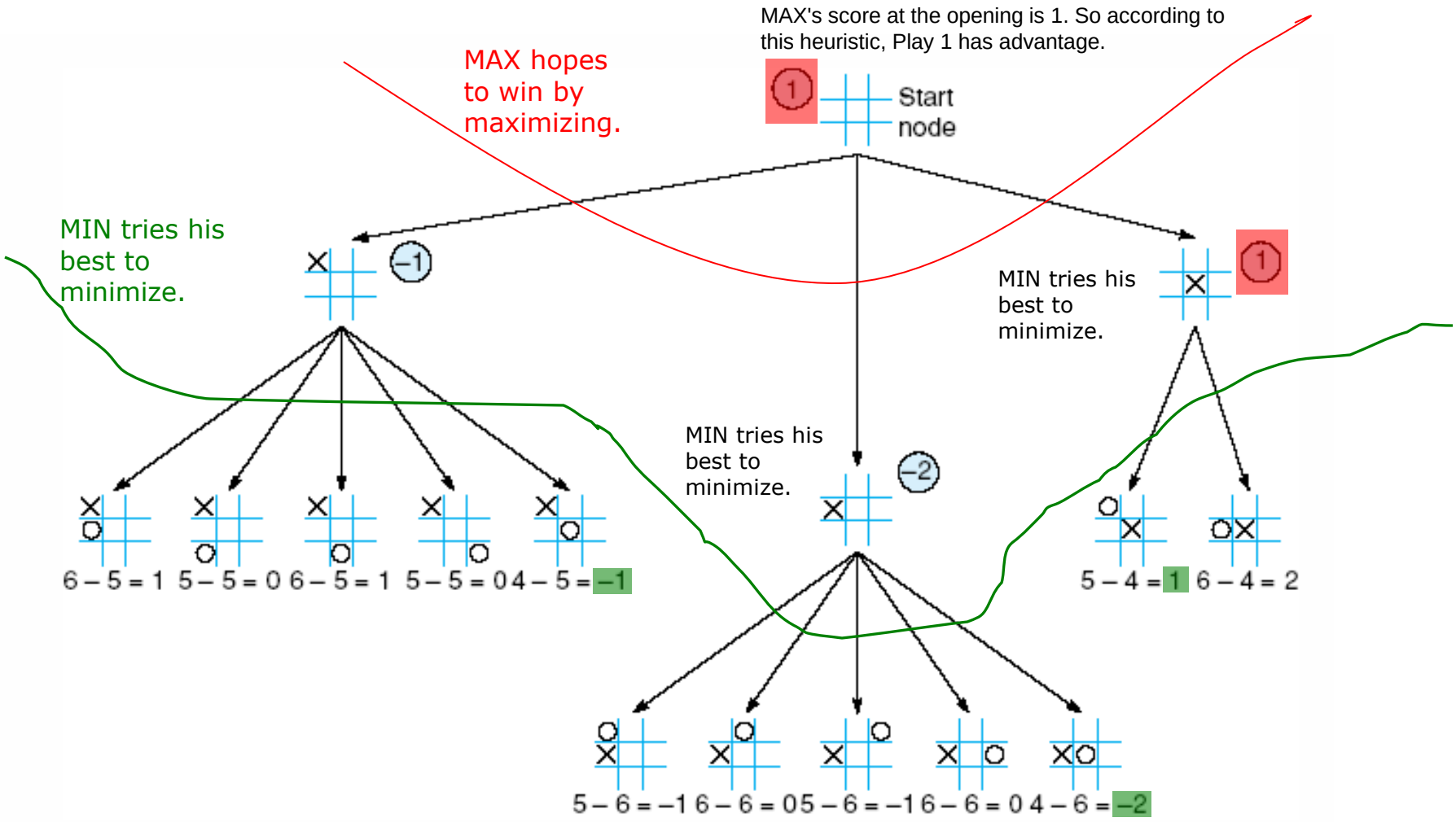
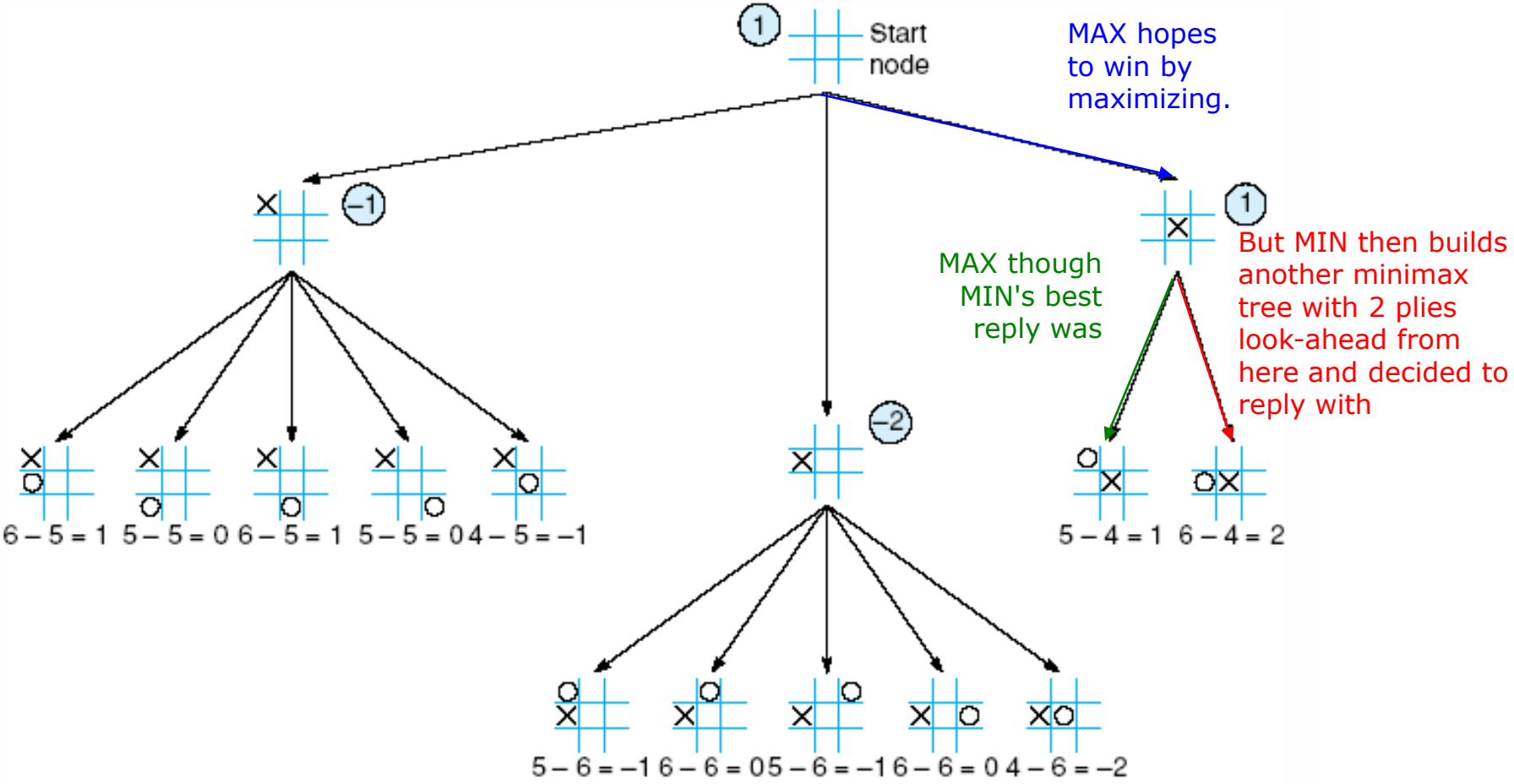



Fig. 4.23

# MAX makes his first move



# MAX's 2nd move: look ahead analysis

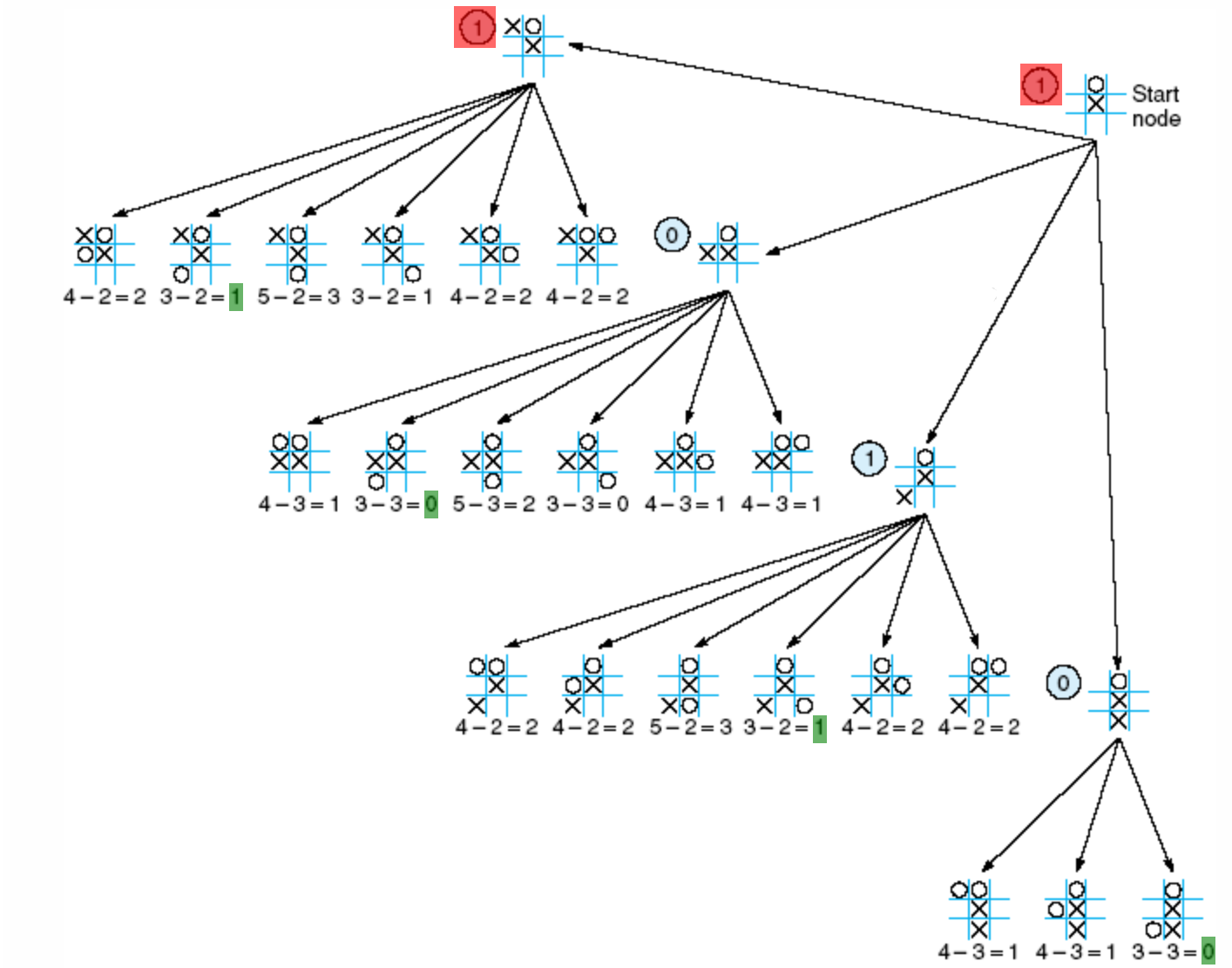
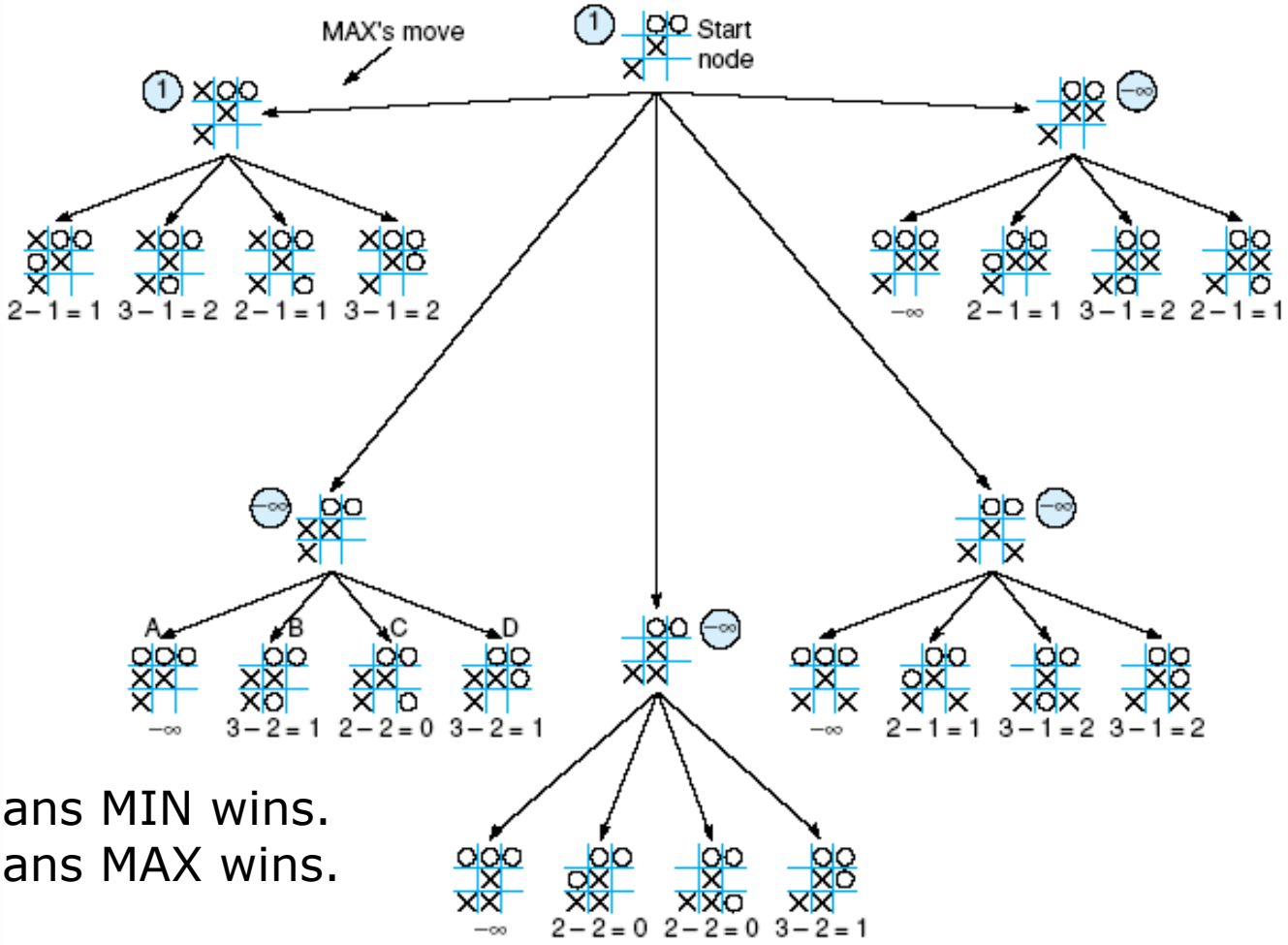



Fig. 4.24

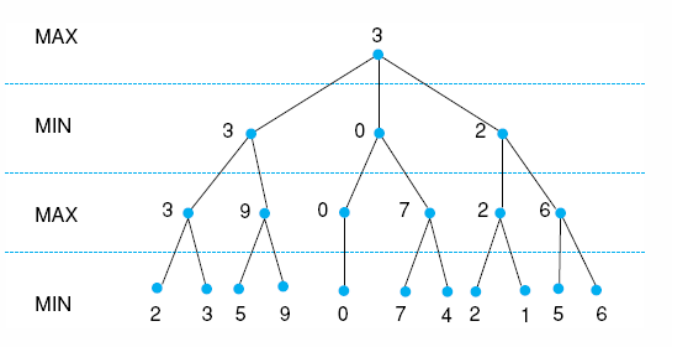
# MAX's 3rd move: look ahead analysis



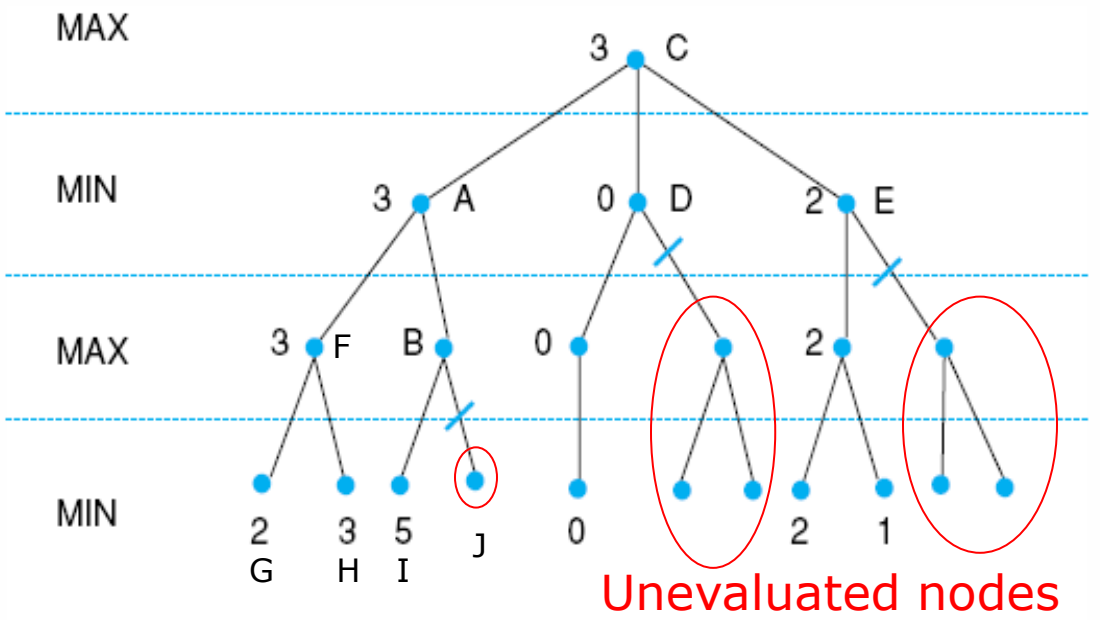

−∞ means MIN wins.
+∞ means MAX wins.

Fig. 4.25

# Alpha-beta pruning example




Minimax without pruning




Depth-first search
Visit C, A, F,
Visit G, heuristics evaluates to 2
Visit H, heuristics evaluates to 3
Back up {2,3} to F. max(F)=3

Back up to A. β(A)=3. Temporary min(A) is 3.
3 is the ceiling for node A's score.

Visit B according to depth-first order.
Visit I. Evaluates to 5.
Max(B)>=5. α(B)=5.

A has $\beta = 3$ (A will be no larger than 3)
B is $\beta$ pruned, since $5 > 3$
C has $\alpha = 3$ (C will be no smaller than 3)
D is $\alpha$ pruned, since $0 < 3$
E is $\alpha$ pruned, since $2 < 3$
C is 3

It does not matter what the value of J is, min(A)=3. β-prune J.

Alpha-beta pruning improves search efficiency of minimax without sacrificing accuracy.

# Alpha-beta pruning

min(A) = -1

max(S) must >= -1

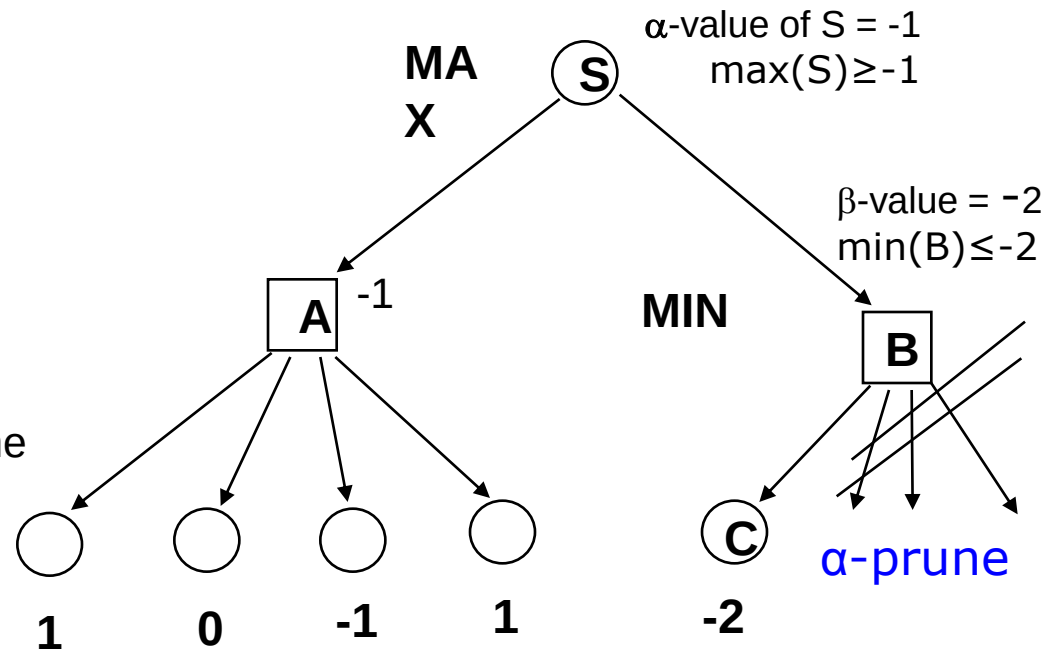
Lower bound: α-value of S = -1

h(C) = -2

min (B) <= -2

Upper bound: β-value = -2

Final value of B can never exceed current value of S. We can prune the other children of B.




- Proceed in a depth-first fashion in the n-ply look-ahead search tree.
- Find the score for the top of this tree.
- During the search, creates two values alpha and beta
- α-value associated with MAX can never decrease
  - MAX's eventually score is at least as good as the current α-value
- β-value associated with of MIN can never increase
  - MIN's eventually score is at least as good as the current β-value
- α-prune: β-value <= α-value of a MAX ancestor
- β -prune: Any MAX node having α-value <= β-value of any MIN ancestor

# Conclusion

- Minimax search is designed for the adversarial search space, MAX vs MIN.
- Before MAX makes a move, he looks n plies ahead in a DFS manner.
- Apply heuristic function for the states at the end of the look ahead level.
- Propagate these values back up to the current state.
  - Use alpha-beta pruning to cut bad branches (moves) in the game tree to improve search time.
- Choose the move that maximizes the current node score.
- Then it is MIN's turn of doing similar look-ahead and pruning to decide his move.
- The players alternate until game over.