Adversarial Search

Content

- Game Playing
- Min-Max Algorithm
- Alpha Beta Pruning

Adversarial Search

- The Adversarial search is a well-suited approach in a competitive environment, where two or more agents have conflicting goals
- Adversarial Search problem is having competitive activity which involves 'n' players and it is played according to certain set of protocols.
- Game is called adversarial because the goals are conflicting and surrounding environment is competitive .
- Environment Types:
  - Competitive environment
  - Cooperative environment

Game Playing

- Game playing was one of the earliest researched AI problems because
  - Games seem to require **intelligence** (heuristics)
  - **state of game**s are easy to represent
  - only a **restricted number of actions** are allowed and outcomes are defined by precise rules
- Purpose is to get a game solved, meaning that the entire game tree can be built and outcome of the game can be determined even before the game is started.

Two Reasons

- Games are good domain in which to explore machine intelligence:
  - They provide a **structured task** in which it is easy to measure success or failure
  - They did not obviously require large **amounts of knowledge**. They were thought to be solvable by straightforward search from the starting state to a winning position.

Categories of Games

- Two-players vs. Multi-players
  - Chess vs. Monopoly, Poker
- Deterministic vs. Non-deterministic
  - Chess vs. Backgammon
- Perfect Information vs. Imperfect Information
  - Checkers vs. Bridge

Game Playing as Search

- Playing a game involves searching for the **best move**.
- Board games clearly involve notions like **start state**, **goal state**, **operators** etc.
- Therefore, we can usefully import problem solving techniques that we have already met.
- There are important **differences** between **Playing a game and standard search problems.**

Special Characteristics of Game Playing Search

- Main differences are uncertainties introduced by
  - **Presence of an opponent** (competitor). One do not know what the opponent will do until he/she does it.
  - **Complexity** Most interesting games are simply too complex to solve by exhaustive means.

Game Definition

- **Players**: We call them Max and Min.
- **Initial State**: Includes board position and whose turn it is.
- **Operators**: These correspond to legal moves.
- **Terminal Test**: A test applied to a board position, which determines whether the game is over or not. In chess, for example, this would be a checkmate or stalemate situation.
- **Utility Function**: A function which assigns a **numeric value** to a **terminal state**. For example, in chess the outcome is win (+1), lose (-1) or draw (0).

Zero-sum game

- A zero-sum game is one in which no wealth is created or destroyed.
- So, in a two-player zero-sum game, whatever one player wins, the other loses.
- In this game, we can include at least two and a maximum to an infinite number of contestants.
- It assumes a version of perfect competition and perfect information.
- Let us consider a game in which **n** contestant takes part, and contestant **i** has **Ni** courses of action available to him. Then the number of outcomes to a play of the game will be **N1, N2,.....,** **Nn.** So, consider a possible outcome **θ** result in payment **p(i, θ)** to a contestant **i**. Then, the game is called a **Zero-Sum Game,** if every possible result **θ**, we have

$$\sum_{i=1}^{n} p(i,) = 0$$

Zero-sum game :Example

- Chess Game -1,+1, 0 Payoff
- Lets us understand this by taking an example, three-man **A**, **B**, and **C** playing poker and put 100 bucks all, the winner takes all money. If **A** wins the game he gains 200 bucks but **B** and **C** lose 100 bucks each. So, the total payment to all contestants after the play is zero. Similarly, if **B** or **C** wins too.
- In financial markets, futures and options are considered non-zero games because contracts represent agreements between two parties and, if one investor loses, the asset is transferred to another investor. Poker, Gambling, Matching Pennies, Chess, Tennis are also some examples of **Zero-Sum Game**.

Non Zero sum Game

Dont have algebraic sum of payoff as zero.

One player winning the game does not necessarily means that the other player lost the game.

Two types:

1. Positive sum Game/ Cooperative game
   Educational game
1. Negative sum Game/Negative sum game

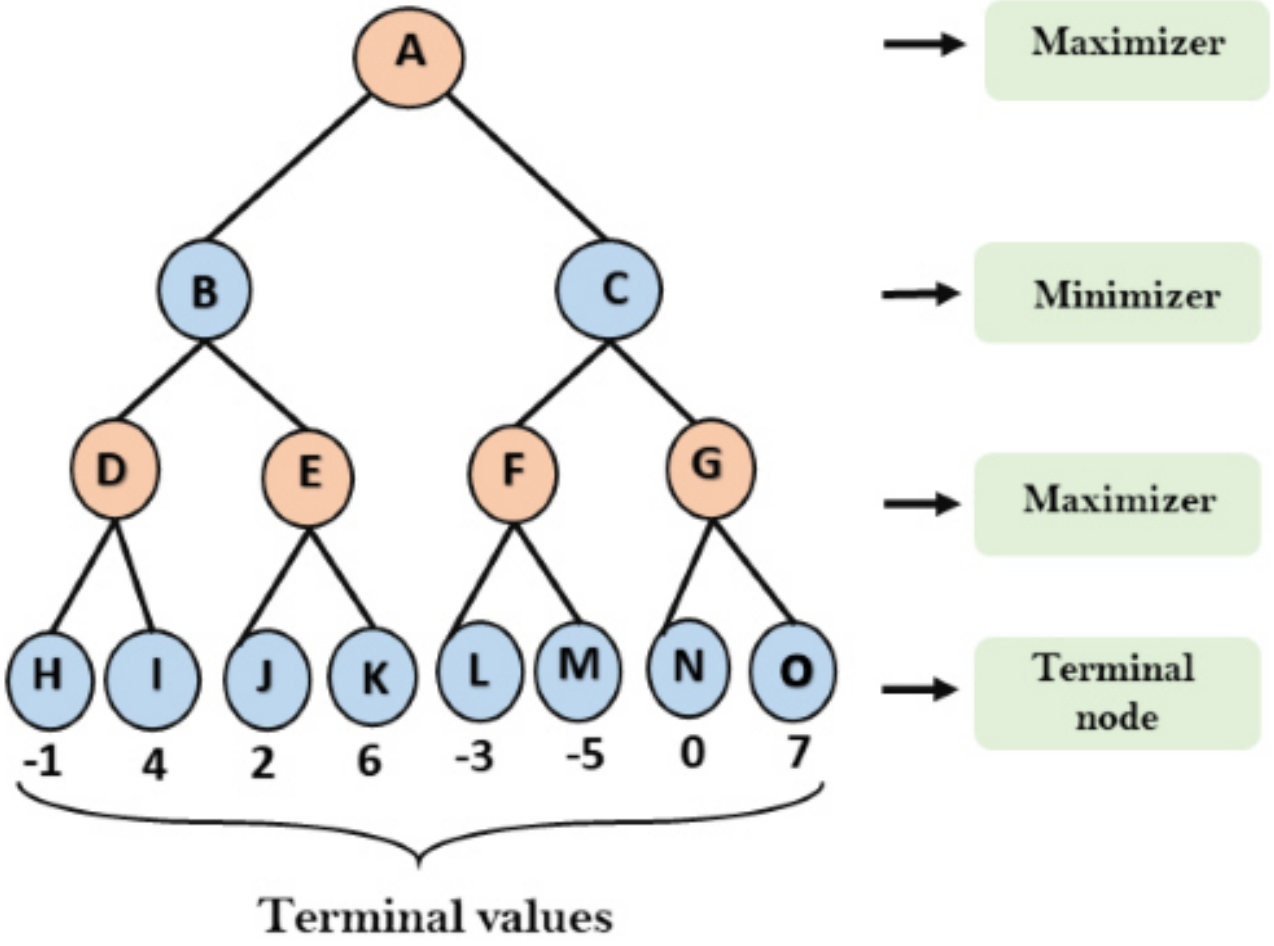
Min Max Algorithm

- It is designed to determine the optimal strategy for MAX and thus to decide what the best first move is.

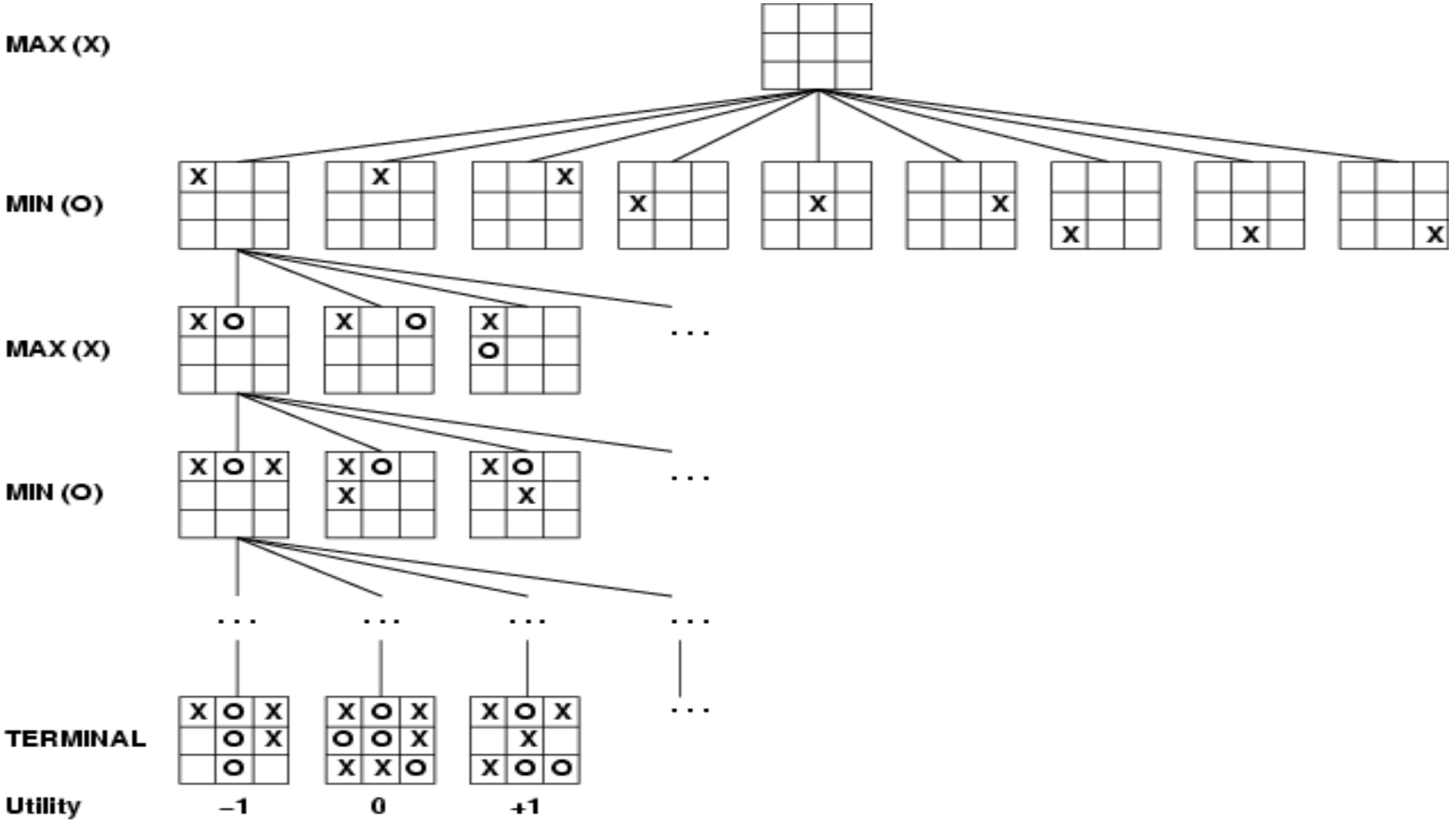
- The algorithm consists of 5 steps:
  - Generate the whole game tree, all the way down to the terminal states
  - Apply the utility function to each terminal state to get its value.
  - Use the utility of the terminal states to determine the utility of the nodes one level higher up in the search tree
  - Continue backing up the values from leaf nodes toward the root, one layer at a time
  - Eventually, the backed up values reach the top of the tree; at that point, MAX chooses the move that leads to the highest value.

Min-Max Rules

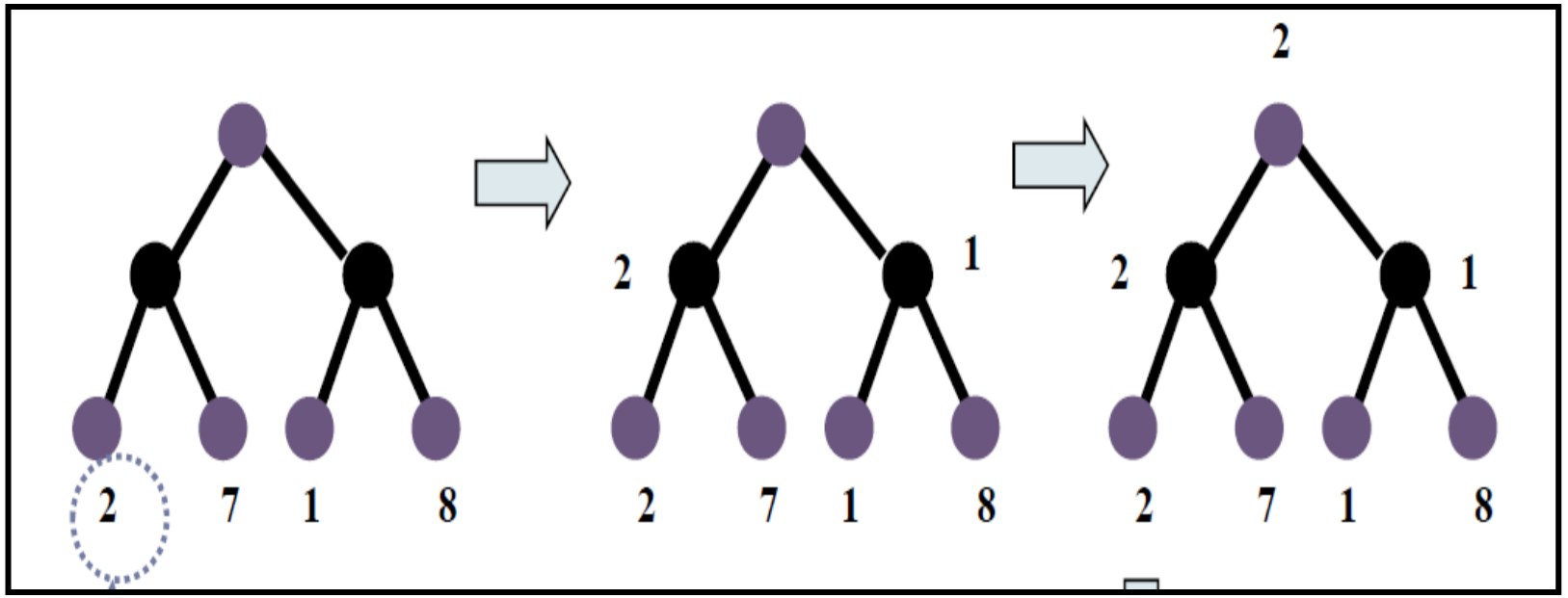
- If it is my turn to move, then the root is labelled a "MAX" node; otherwise it is labelled a "MIN" node indicating my opponent's turn.
- Each level of the tree has nodes that are all MAX or all MIN; nodes at level i are of the opposite kind from those at level i+1
- Complete game tree: includes all configurations that can be generated from the root by legal moves (all leaves are terminal nodes)
- Incomplete game tree: includes all configurations that can be generated from the root by legal moves to a given depth (looking ahead to a given steps)


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
-1
4
2
6
-3
-5
0
7
Terminal values
Maximizer
Minimizer
Maximizer
Terminal node

Partial search Tree for the game of Tic-Tac-Toe



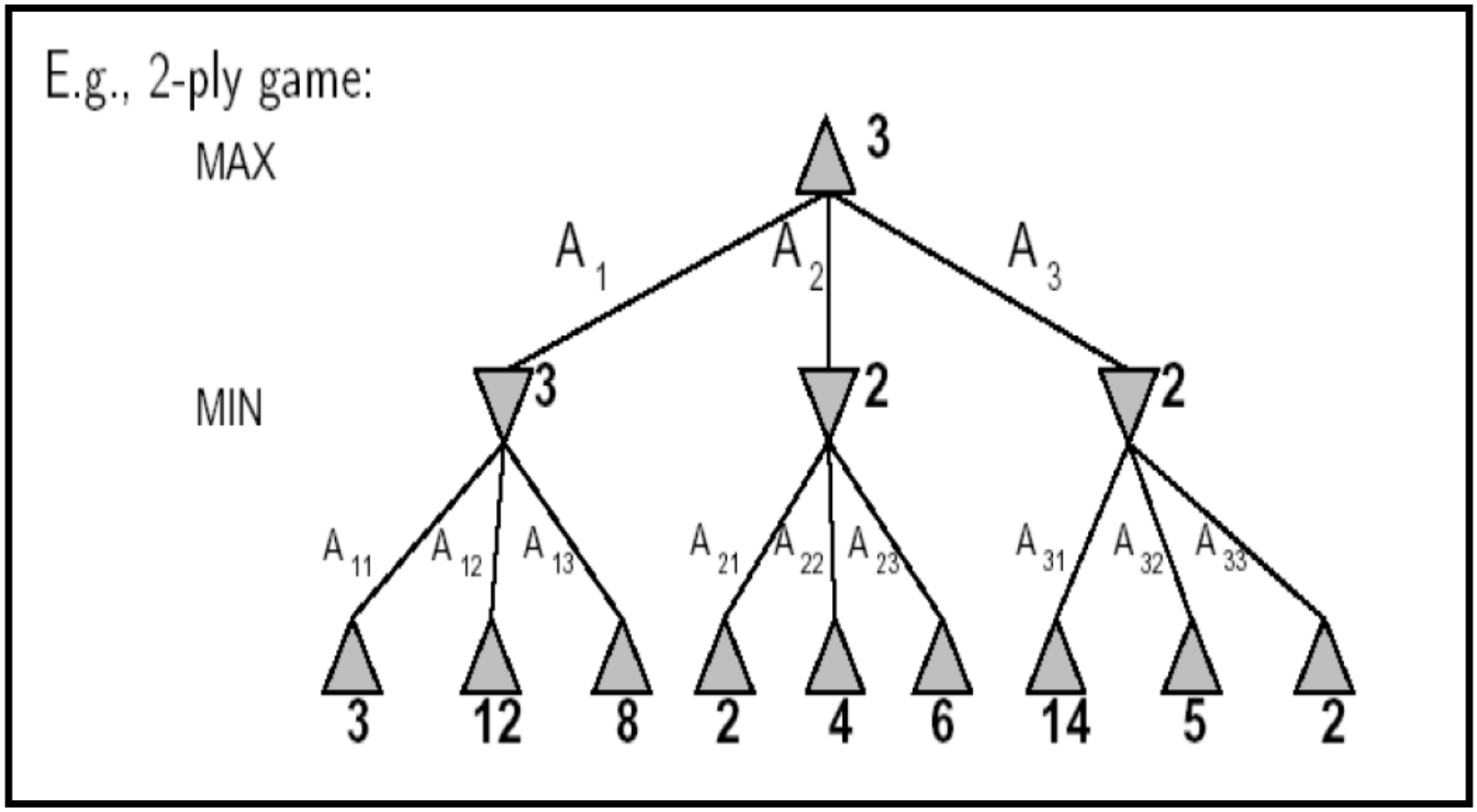
Min-Max Search



Max

Min

Min-Max Algorithm Example



Properties of Min-Max

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

Alpha –Beta Pruning

- The process of eliminating a branch of the search tree from consideration without examining is called pruning search tree.

- Alpha beta pruning when applied to a standard tree , it returns the same as min-max would, but prunes away the branches that cannot possibly influence the final decision.

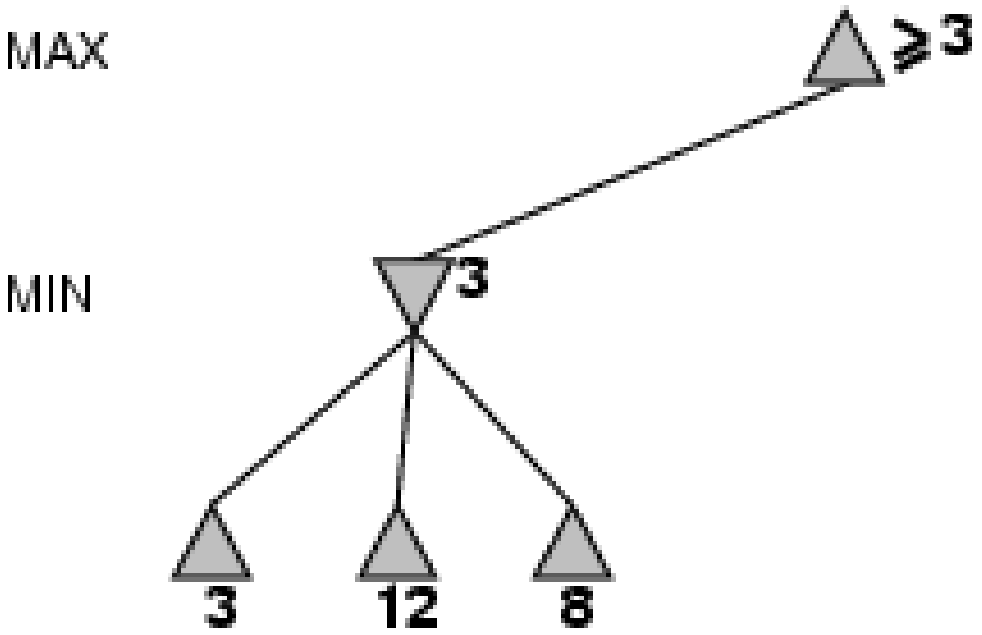
α-β pruning

- **Alpha-beta pruning**
  - Carry alpha and beta values down during search
  - alpha can be changed only at MAX nodes
  - beta can be changed only at MIN nodes
  - Pruning occurs whenever alpha >= beta

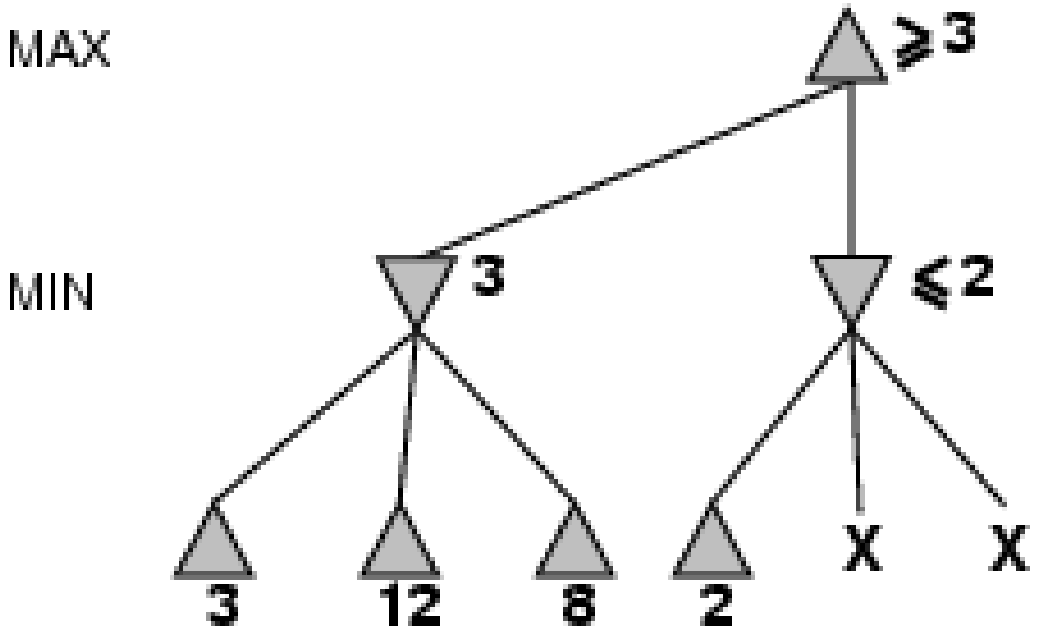
- **alpha cut-off**:
  - Given a Max node n, cut-off the search below n (i.e., don't generate any more of n's children) **if alpha(n) >= beta(n)** (alpha increases and passes beta from below)

- **beta cut-off:**
  - Given a Min node n, cut-off the search below n (i.e., don't generate any more of n's children) **if beta(n) <= alpha(n)** (beta decreases and passes alpha from above)

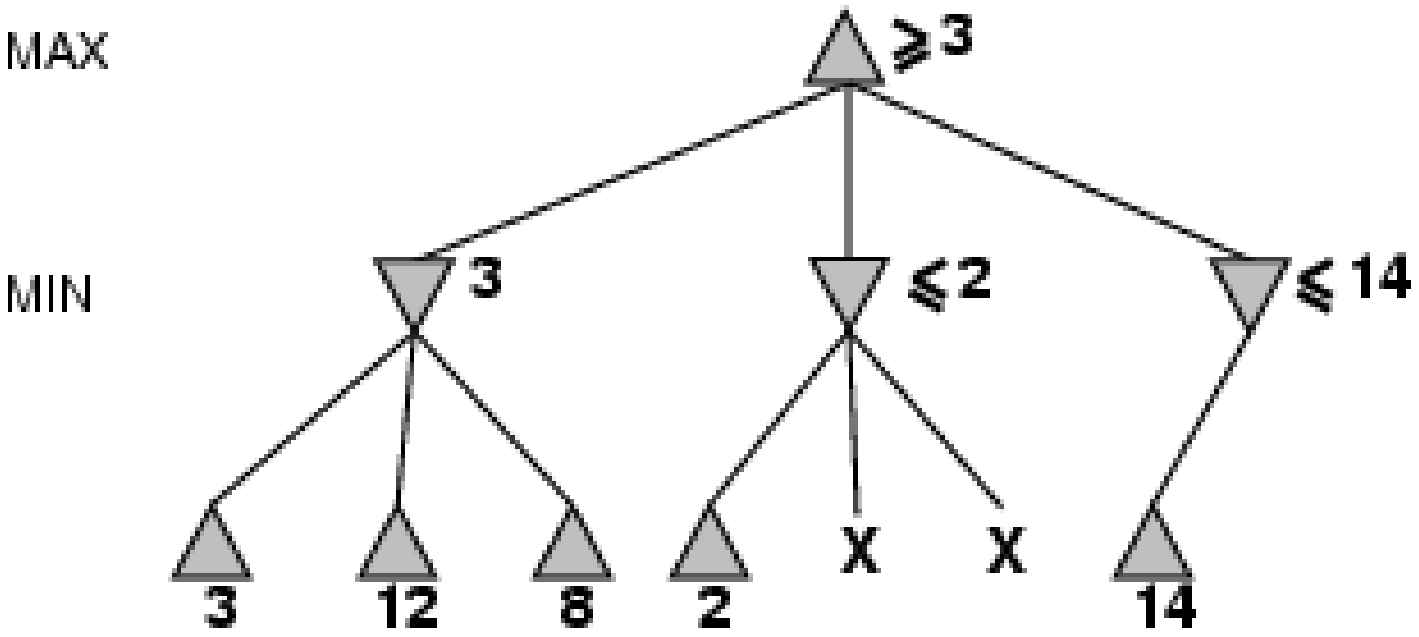
α-β pruning example



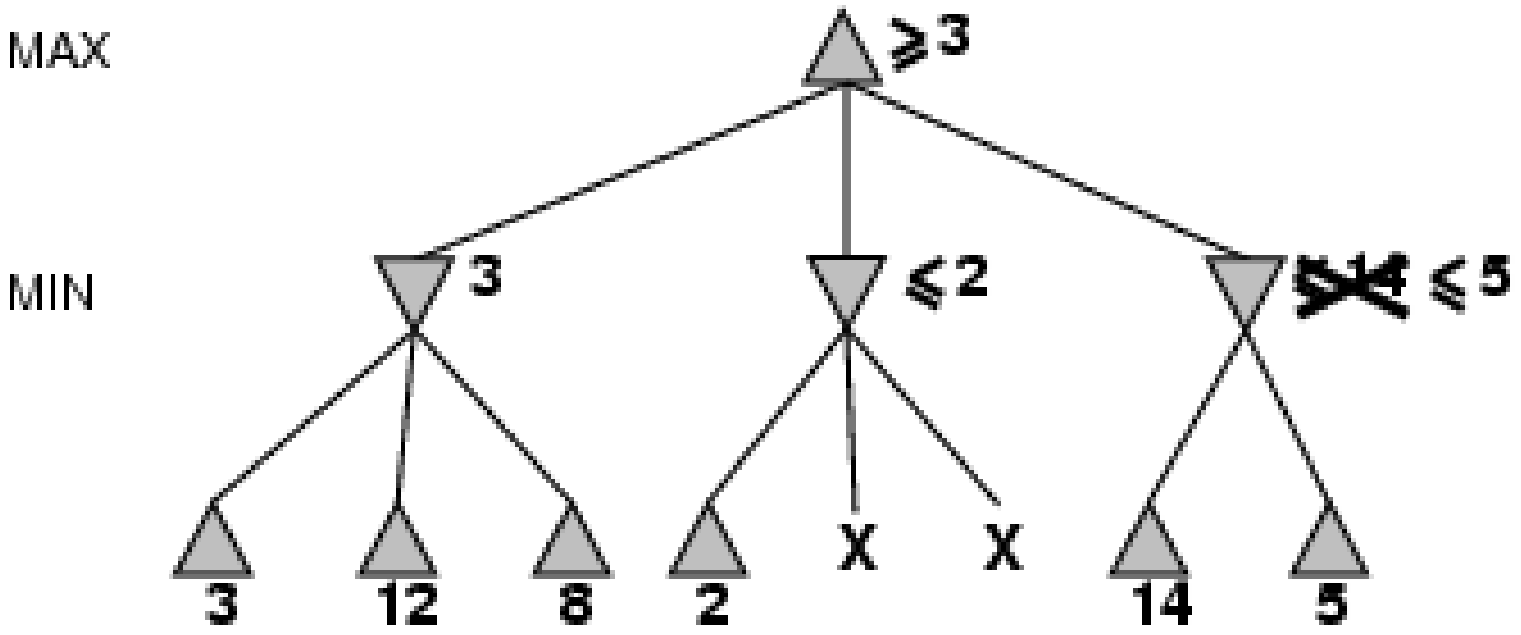
α-β pruning example



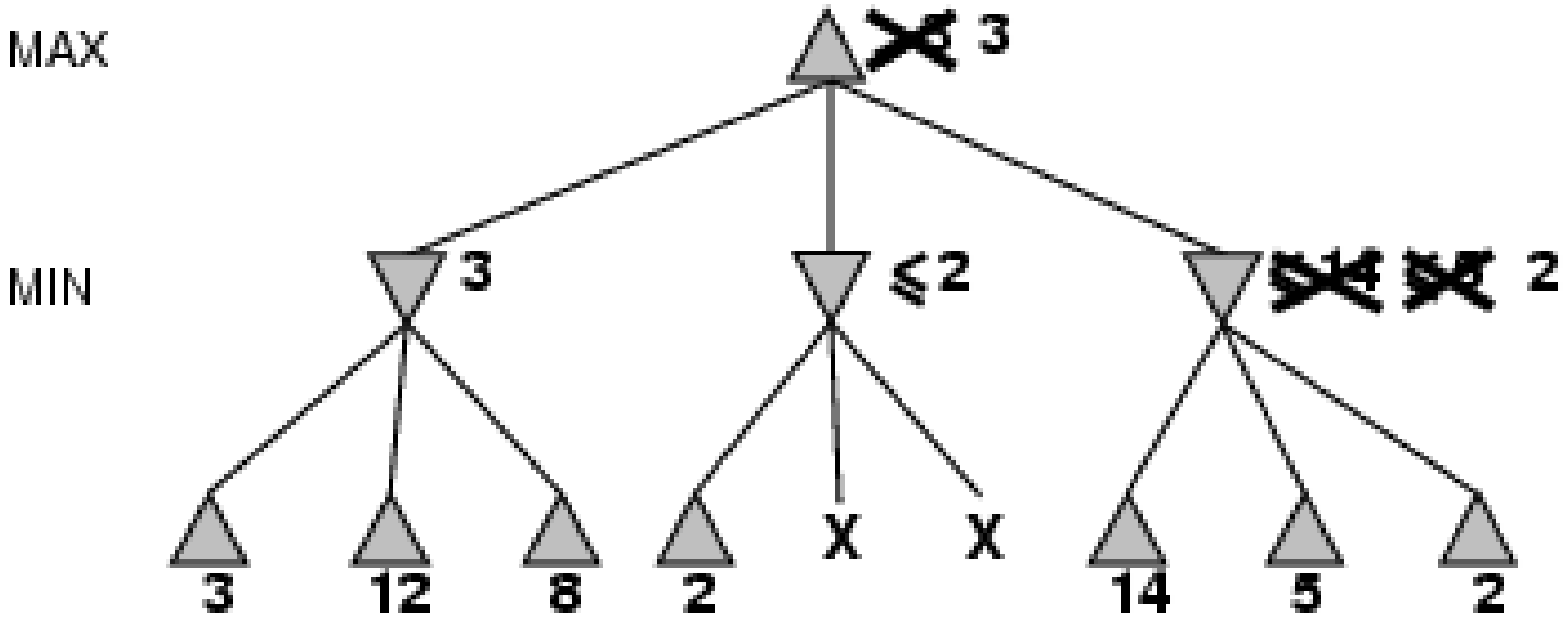
α-β pruning example



α-β pruning example



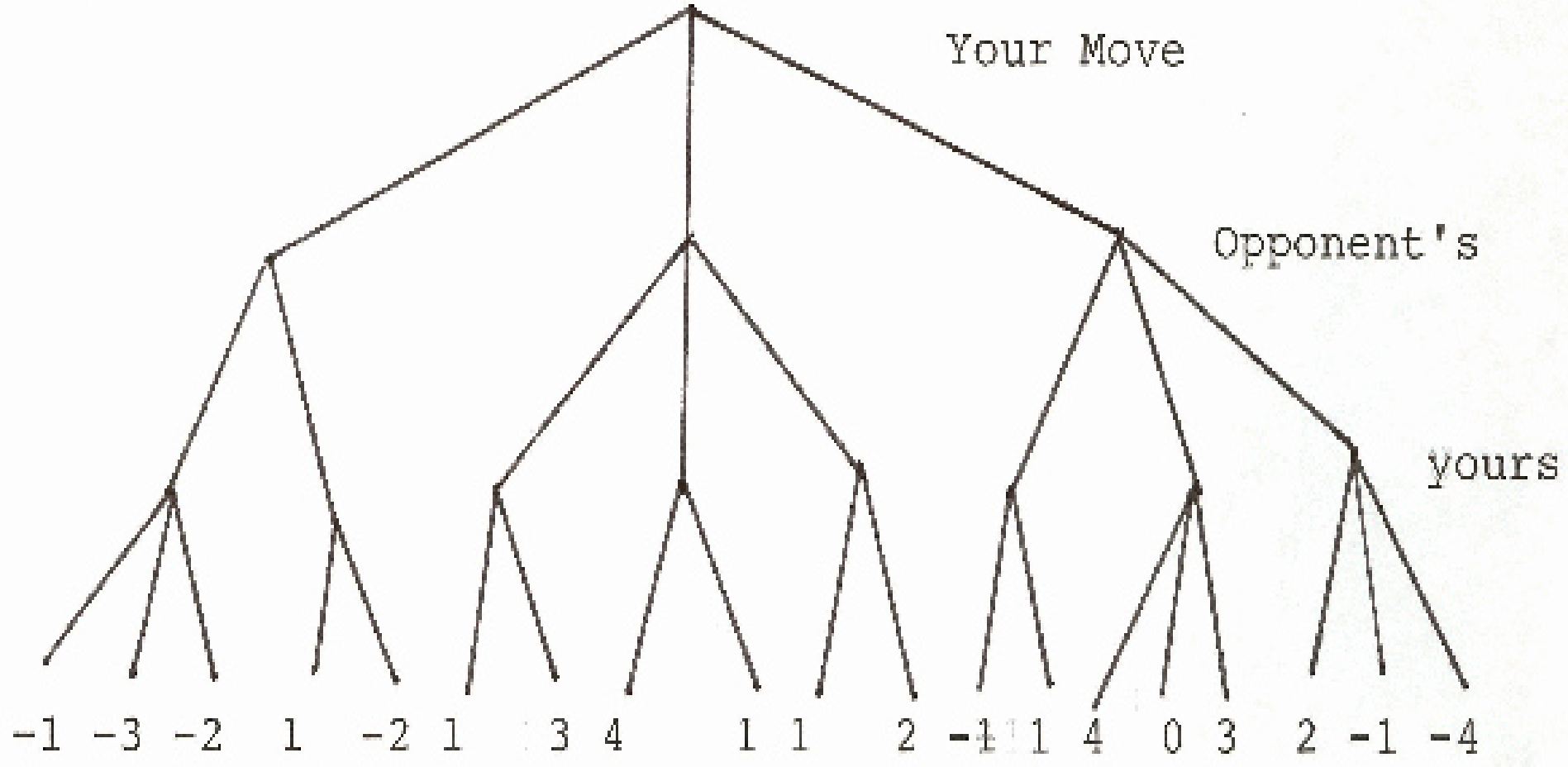
α-β pruning example



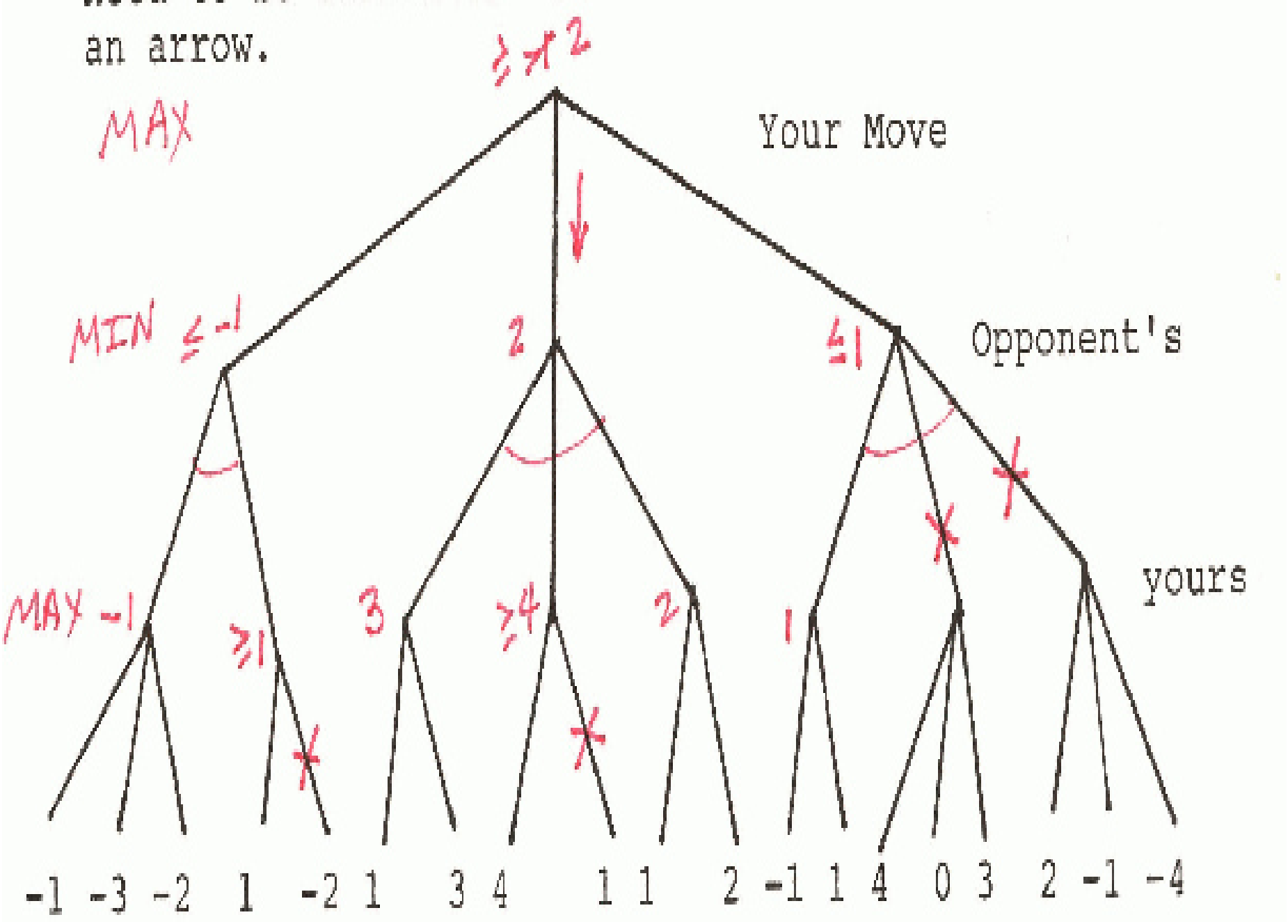
Properties of α-β

- Pruning does not affect final result
- Good move ordering improves effectiveness of pruning
- With "perfect ordering," time complexity = $O(b^{m/2})$

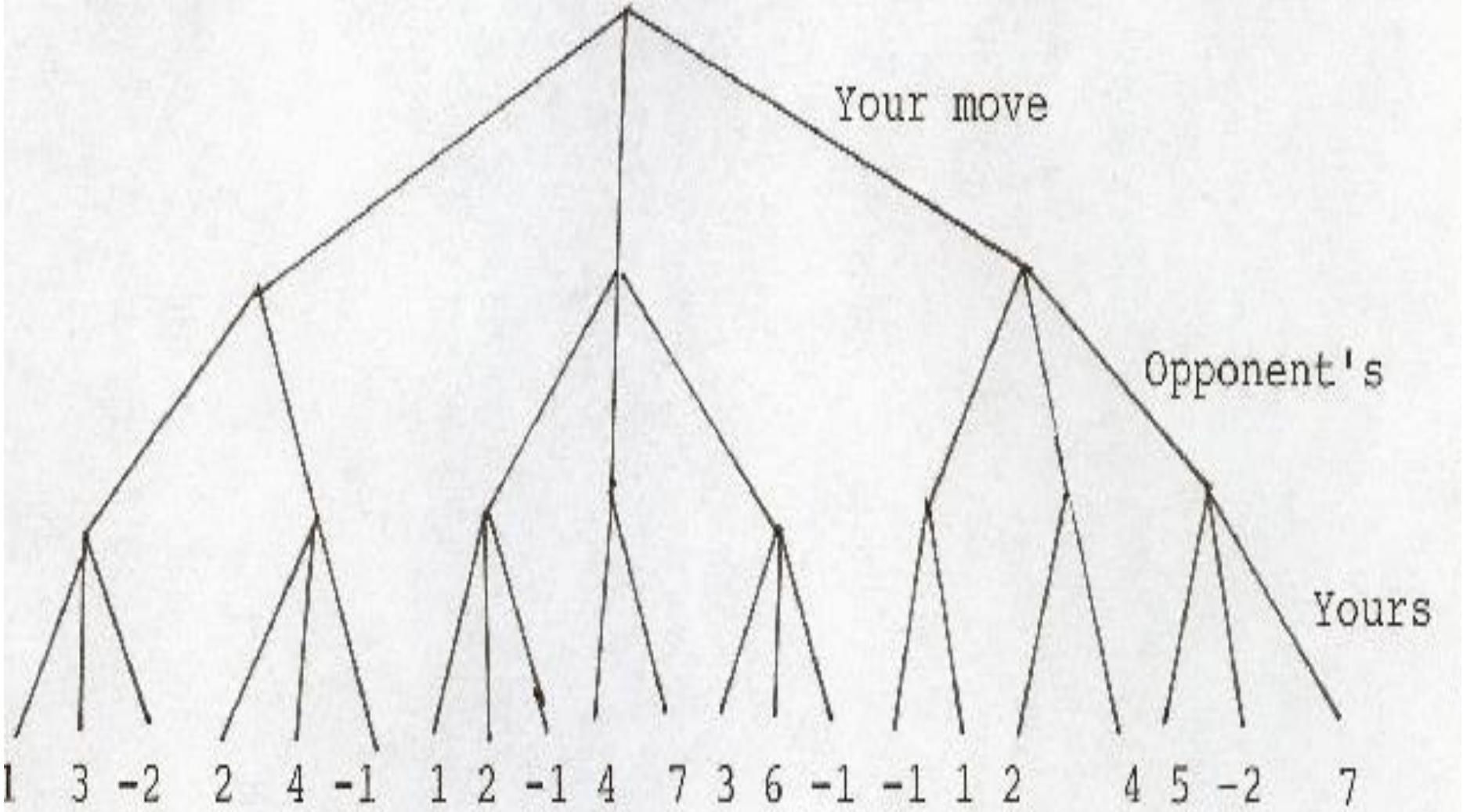
4. Apply the alpha-beta algorithm to the following game tree to find the values of the nodes. Mark with an X those branches of the tree that do not need to be searched. Indicate the best move with an arrow.

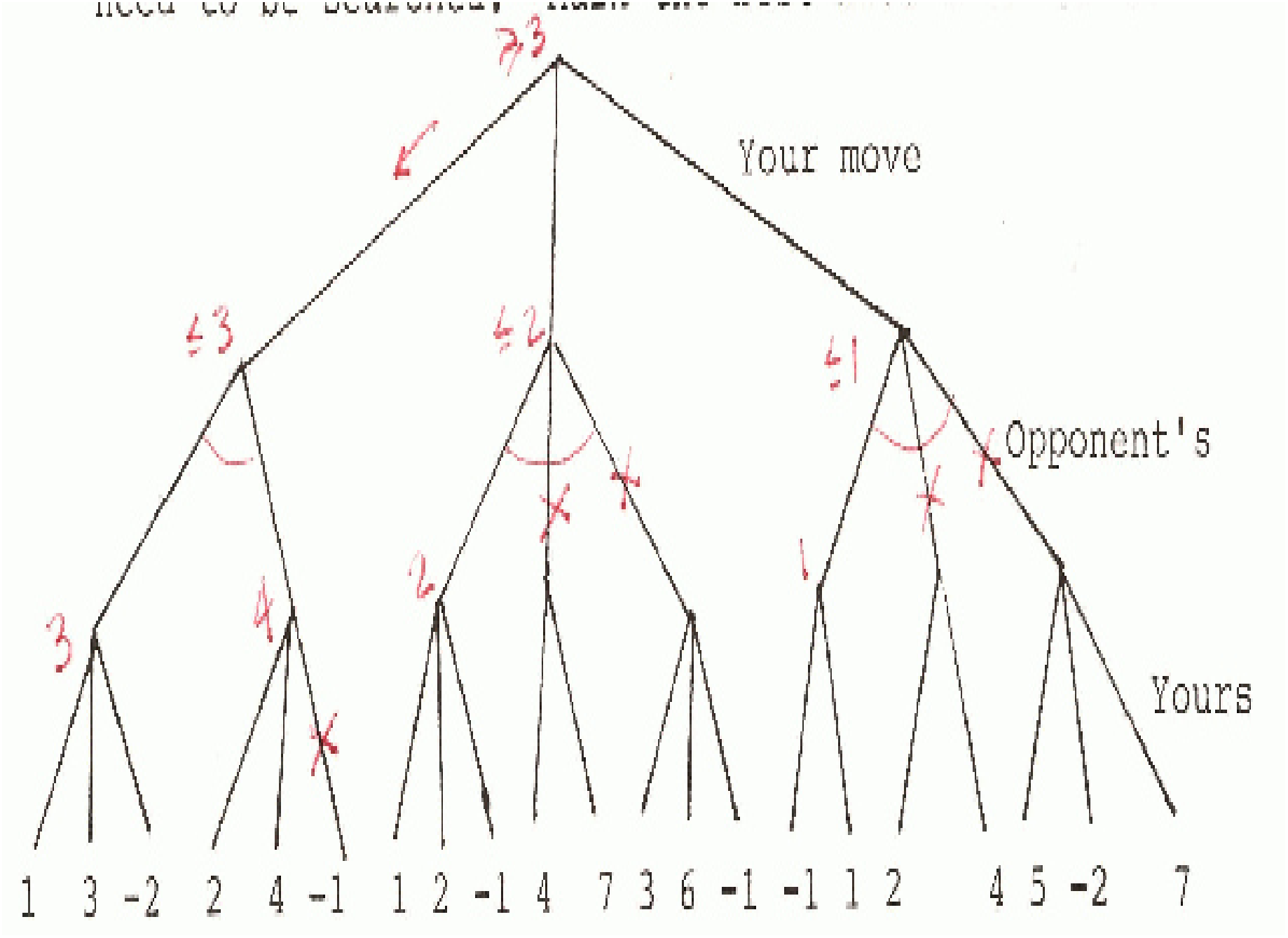


an arrow.

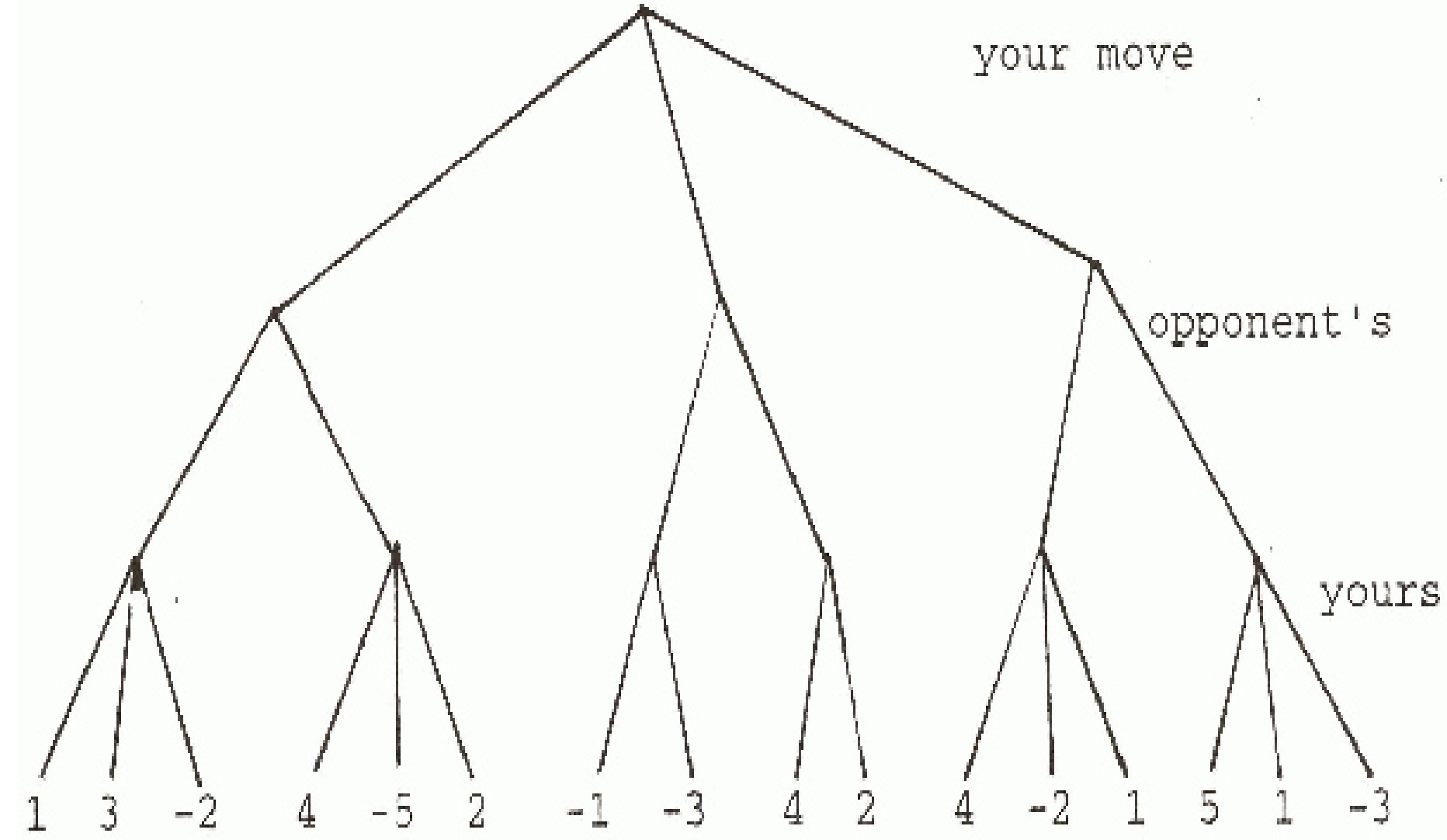

MAX
≥ 2
Your Move
MIN ≤ -1
2
≤ 1
Opponent's
MAX -1
≥ 1
3
≥ 4
2
1
yours
-1 -3 -2 1 -2 1 3 4 1 1 2 -1 1 4 0 3 2 -1 -4

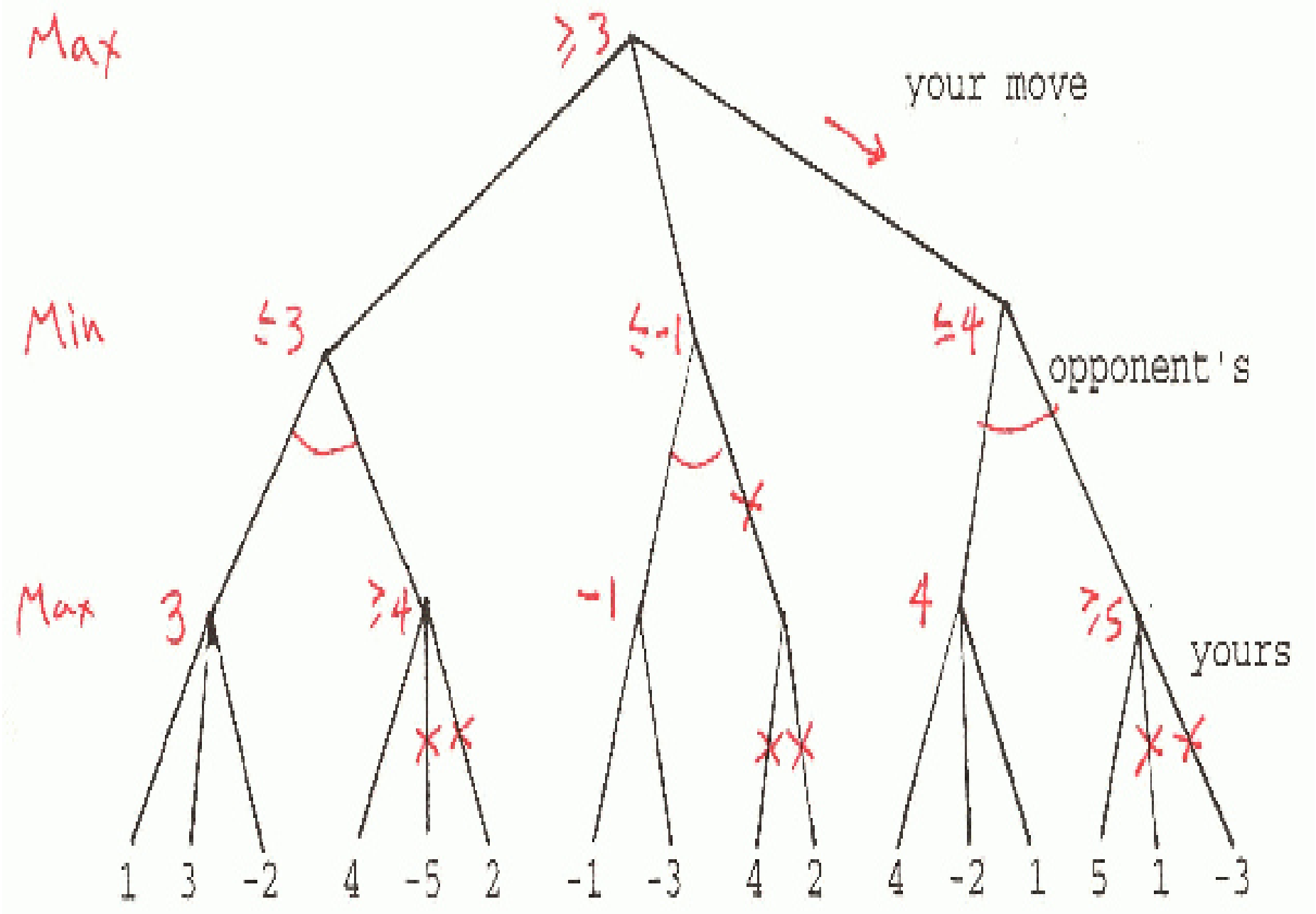
4. Apply the alpha-beta algorithm to the following game tree to find the values of the nodes. Mark with an X those branches of the tree that do not need to be searched. Mark the best move with an arrow.




Your move
Opponent's
Yours
1 3 -2 2 4 -1 1 2 -1 4 7 3 6 -1 -1 1 2 4 5 -2 7

Apply the alpha-beta algorithm to the following game tree to find the values of the nodes. Mark with an X those branches of the tree that do not need to be searched. Indicate the best move with an arrow.




Max
≥3
your move
Min
≤3
≤-1
≤4
opponent's
Max
3
≥4
-1
4
≥5
yours
1 3 -2 4 -5 2 -1 -3 4 2 4 -2 1 5 1 -3