Lecture 11

CS 6380 Artificial Intelligence

Tutorial: Alpha Beta Pruning

Department of Computer Science and Engineering

Indian Institute of Technology Madras (IITM), India

Logistics and Recap

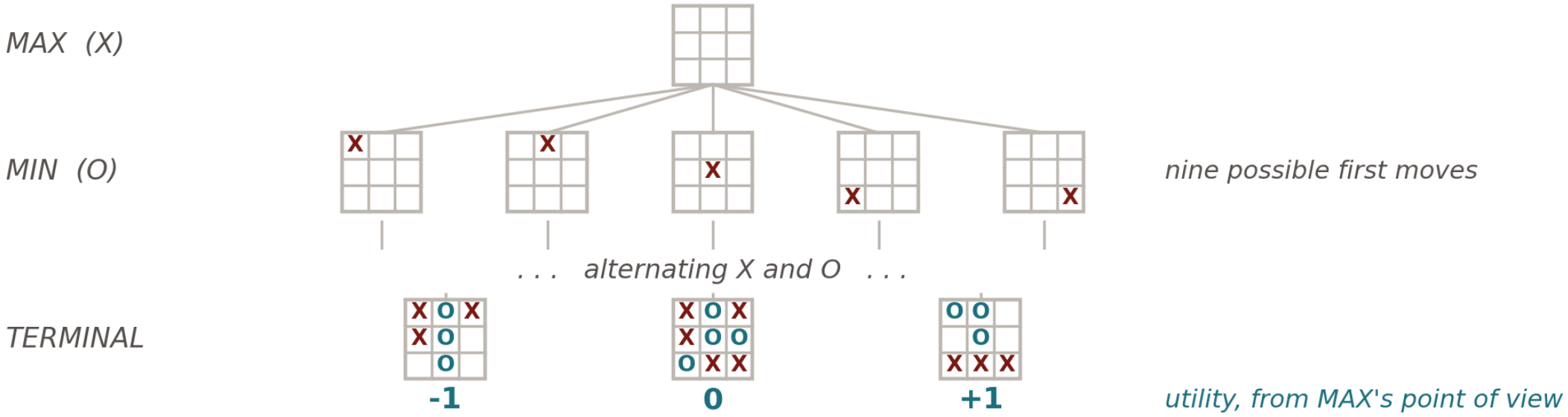
Talk Announcement

- **Title:** Towards Reliable Autonomy: From Individual Agents to Autonomous Teams
- **Date:** August 21, 2026
- **Time: 11**:00 AM – 12:00 PM
- Location: SSB 334

Dr. Sandhya Saisubramanian
Oregon State University, USA
https://www.sandhyasai.com/

The Game Tree

- The initial state with ACTIONS and RESULT defines the game tree, whose nodes are game states and whose edges are moves
- Utility is written from MAX's point of view, so high is good for MAX and bad for MIN, which is where the two names come from
- Tic-tac-toe has under 362,880 terminal nodes, chess over 10^40, so for chess the game tree is a theoretical object we can never build



The Minimax Value

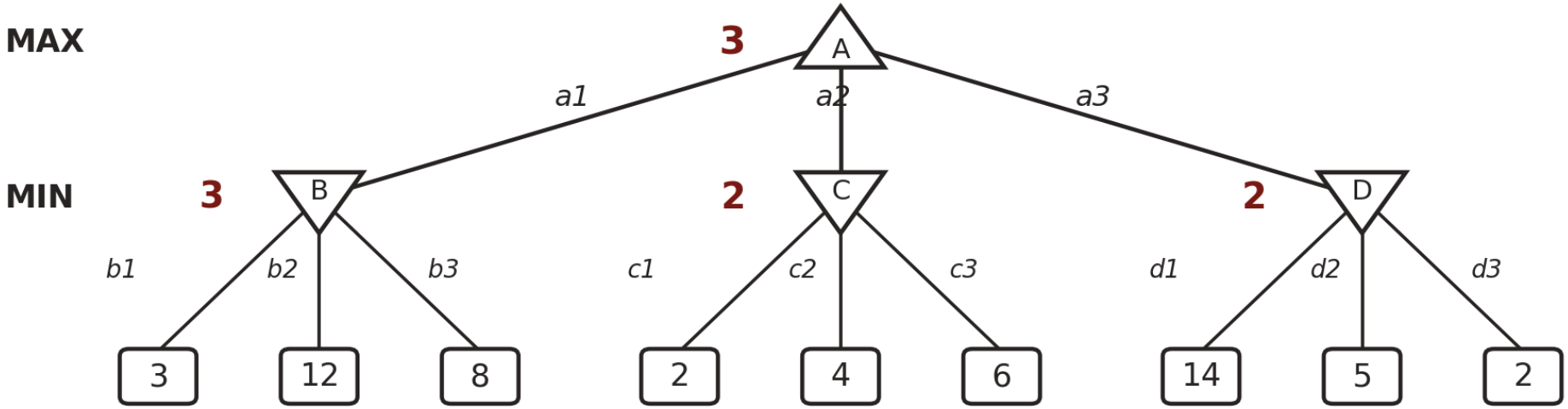
- The minimax value of a node is the utility for MAX of being in that state, assuming both players play optimally to the end
- For a terminal state it is simply the utility, and every other value is built up from those
- MAX moves to the successor of maximum value and MIN moves to the successor of minimum value

```
MINIMAX(s) =
    UTILITY(s)                                  if TERMINAL-TEST(s)
    max over a of MINIMAX(RESULT(s,a))          if PLAYER(s) = MAX
    min over a of MINIMAX(RESULT(s,a))          if PLAYER(s) = MIN
```

the value of a node when both sides play optimally from there on

A Two Ply Game Tree

- This game ends after one move each, so the tree is two plies deep
- B backs up the minimum of 3, 12 and 8, which is 3, and C and D both back up 2
- The root backs up the maximum of 3, 2 and 2, so a1 is the minimax decision



The Minimax Search Algorithm

- The recursion runs all the way down to the leaves, and the values are backed up as it unwinds
- MAX-VALUE and MIN-VALUE are the two cases of the definition, written out and calling each other

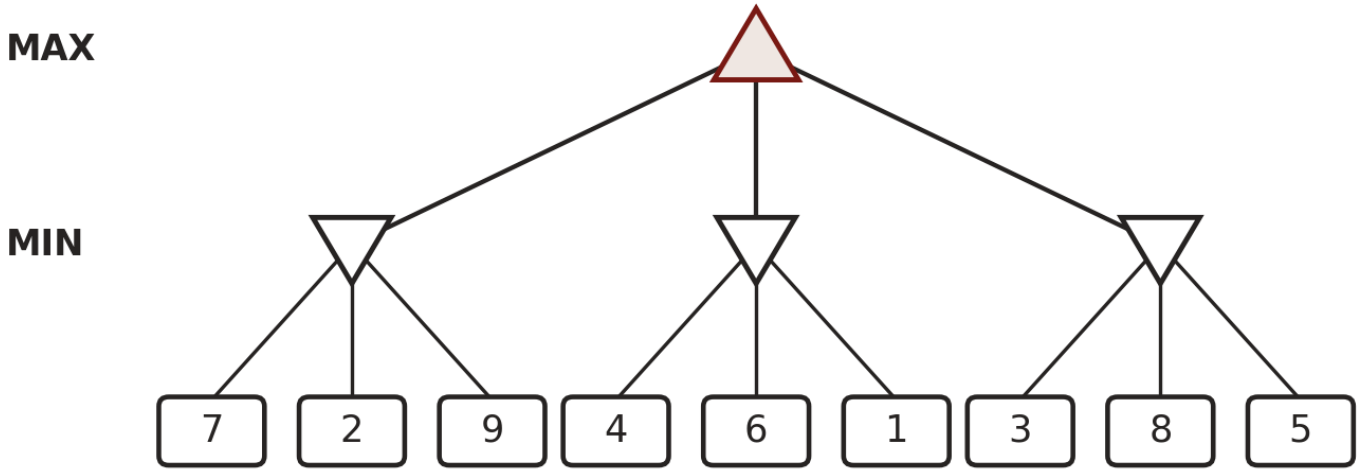
```
MAX-VALUE(state)
if TERMINAL-TEST(state) return UTILITY(state)
v <- minus infinity
for each a in ACTIONS(state)
    v <- MAX(v, MIN-VALUE(RESULT(state,a)))
return v
```

```
MIN-VALUE(state)
if TERMINAL-TEST(state) return UTILITY(state)
v <- plus infinity
for each a in ACTIONS(state)
    v <- MIN(v, MAX-VALUE(RESULT(state,a)))
return v
```

the two routines call each other all the way down to the leaves

Question 1

What is the minimax value at the root of this tree



left to right generation order

A 1

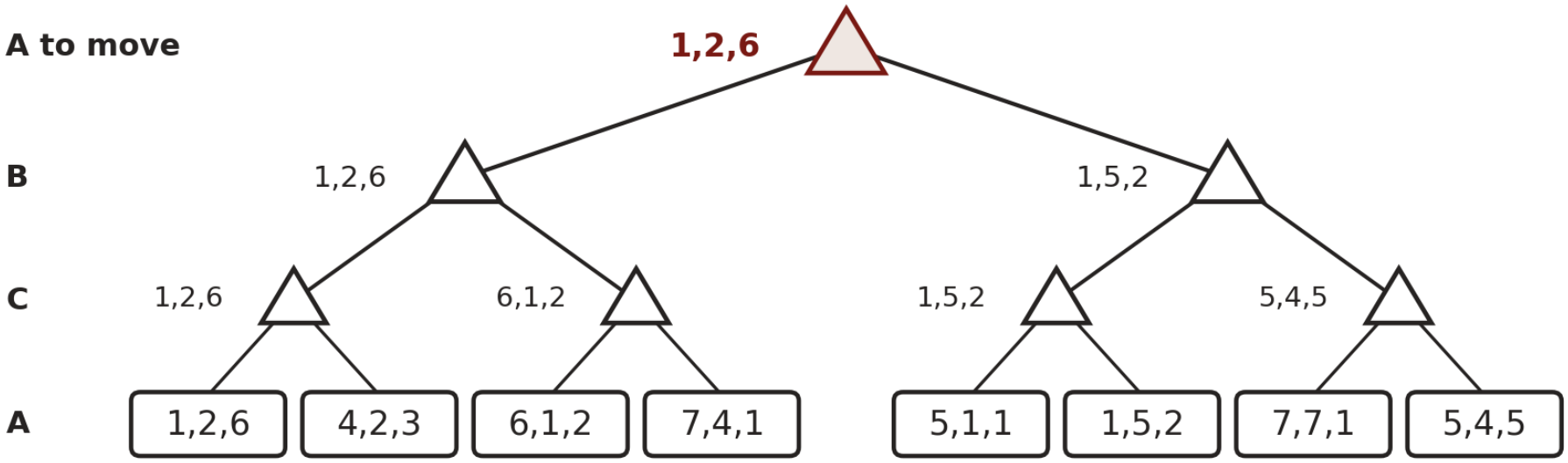
B 2

C 3

D 9

Optimal Decisions in Multiplayer Games

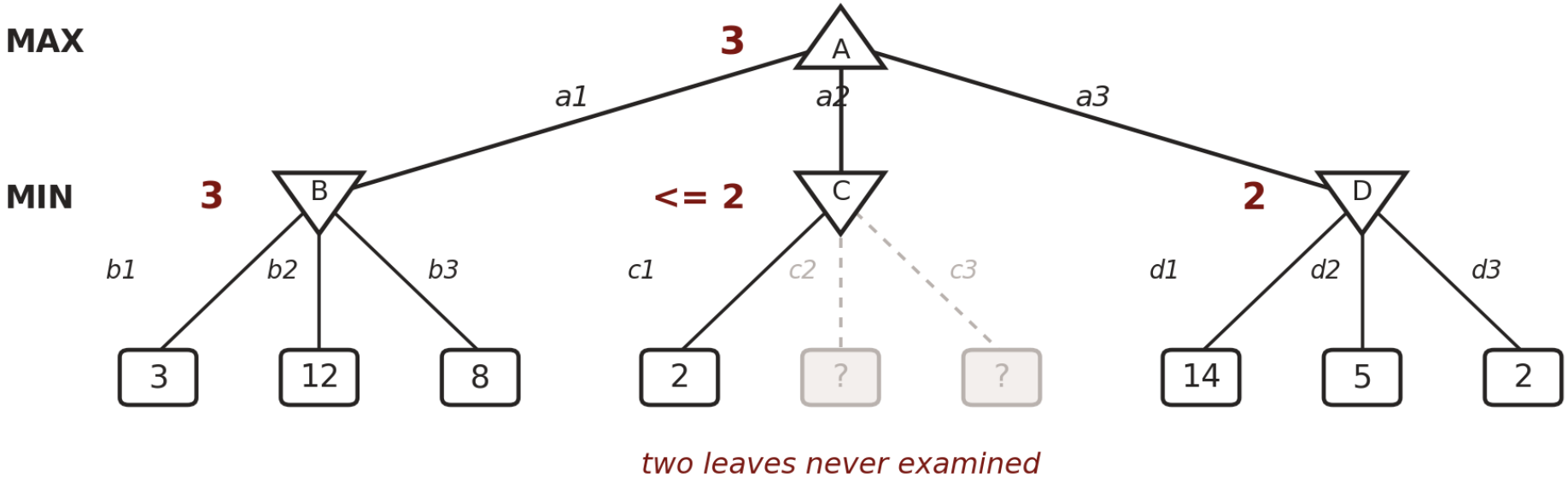
- Replace the single value at each node with a vector holding one utility per player
- Each player backs up the vector whose own component is largest, so there is no minimising any more
- Two player zero sum collapses back to one number because the two components are always opposite



each node carries one utility per player

Alpha Beta on the Two Ply Tree

- After finishing B we know the root is worth at least 3
- The first leaf under C is 2, so C is a MIN node worth at most 2
- MAX would never choose C, so the other two children of C are never generated



The Alpha Beta Algorithm

- These are the minimax routines with two lines added to each
- One line returns early once the current value can no longer matter to the caller, and the other keeps the bound up to date

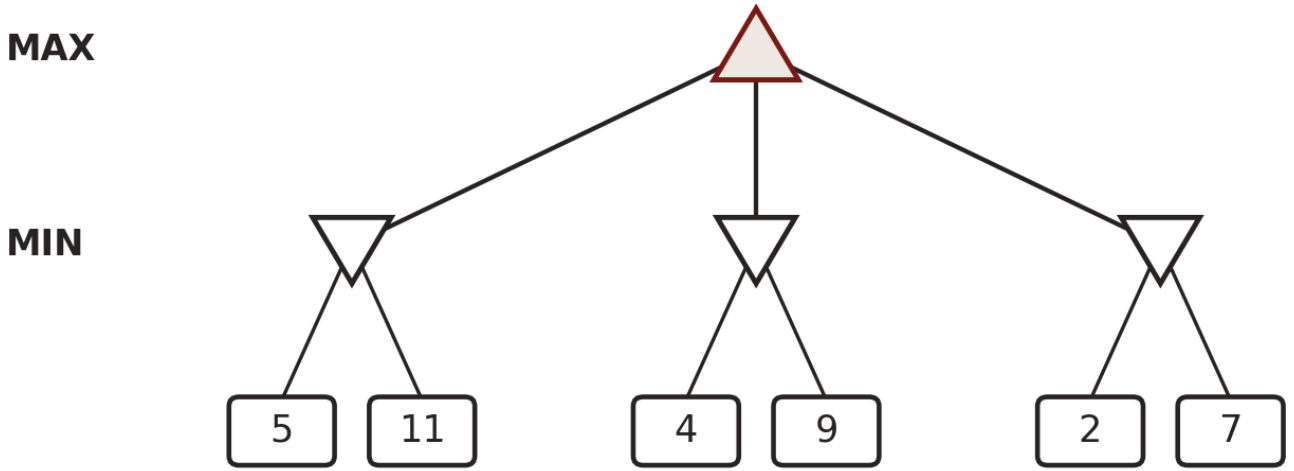
```
MAX-VALUE(state, alpha, beta)
if TERMINAL-TEST(state) return UTILITY(state)
v <- minus infinity
for each a in ACTIONS(state)
    v <- MAX(v, MIN-VALUE(RESULT(state,a), alpha, beta))
    if v >= beta then return v
    alpha <- MAX(alpha, v)
return v
```

```
MIN-VALUE(state, alpha, beta)
if TERMINAL-TEST(state) return UTILITY(state)
v <- plus infinity
for each a in ACTIONS(state)
    v <- MIN(v, MAX-VALUE(RESULT(state,a), alpha, beta))
    if v <= alpha then return v
    beta <- MIN(beta, v)
return v
```

minimax plus the four marked lines, and nothing else

Question 2

Generating children left to right, how many leaves does alpha beta never examine



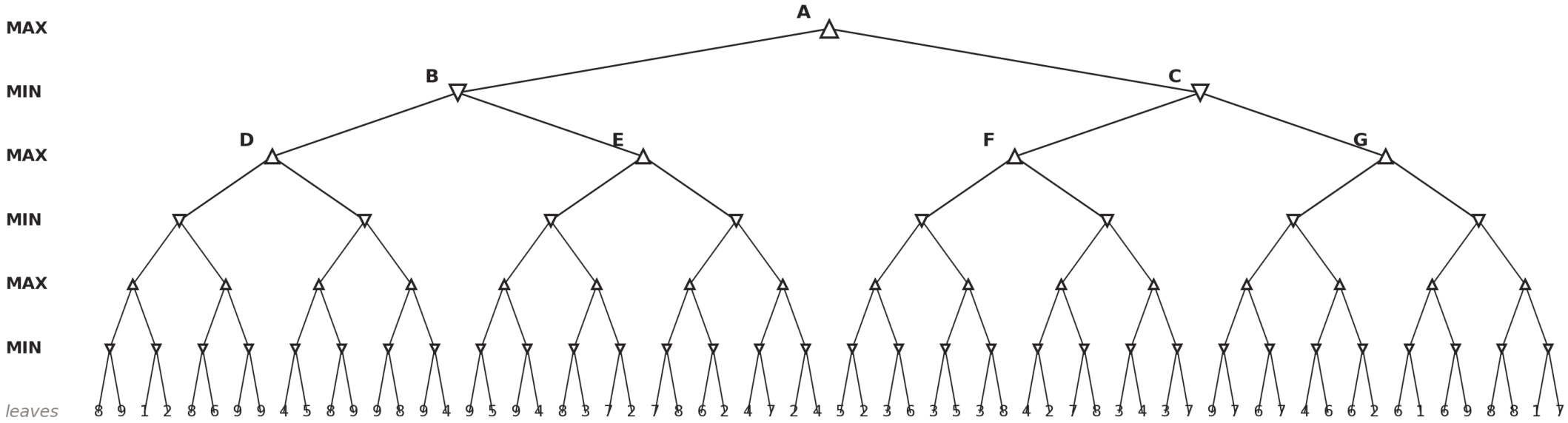
alpha beta examines leaves left to right

A 0	**B** 1	**C** 2	**D** 3

Tutorial: Alpha Beta Pruning

The Tree

- Six plies, so MAX and MIN alternate three times each before the leaves
- Two moves at every node, giving sixty-four terminal positions
- Children are always generated left to right, which is what makes the answers definite



children are generated left to right

How to Trace It by Hand

- Start at the root with alpha at minus infinity and beta at plus infinity
- Write the current alpha and beta beside each node as you descend, and cross out branches as you cut them

Carry down

alpha and beta are passed into every recursive call

Update coming back

a MAX node raises alpha, a MIN node lowers beta

Cut at a MAX node

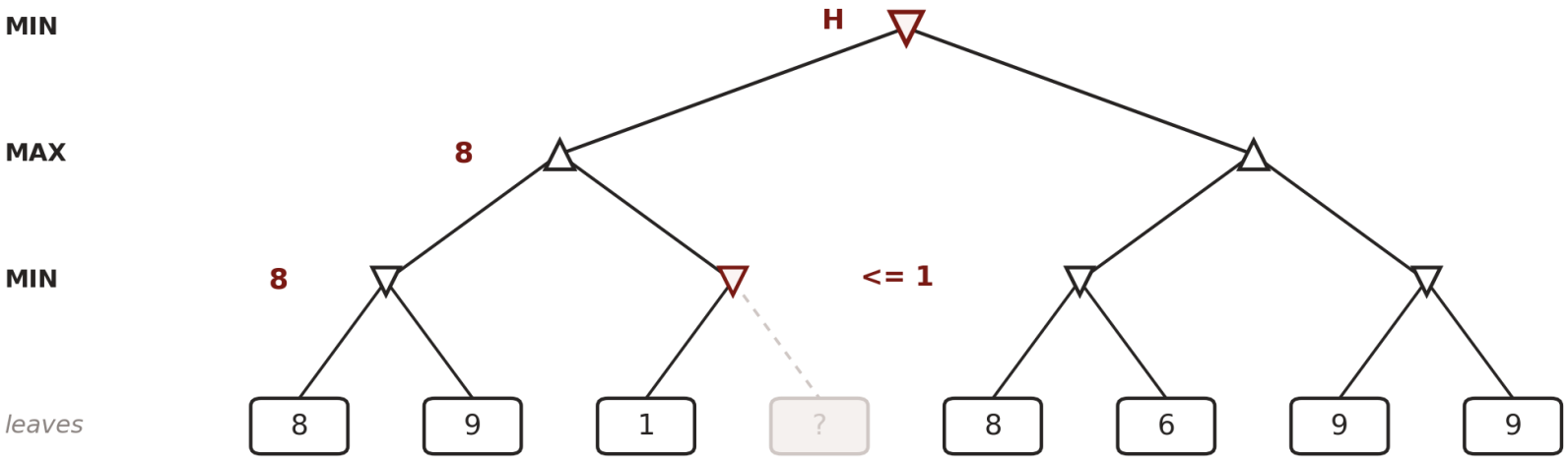
stop as soon as v reaches beta or above

Cut at a MIN node

stop as soon as v falls to alpha or below

Where the First Cut Happens

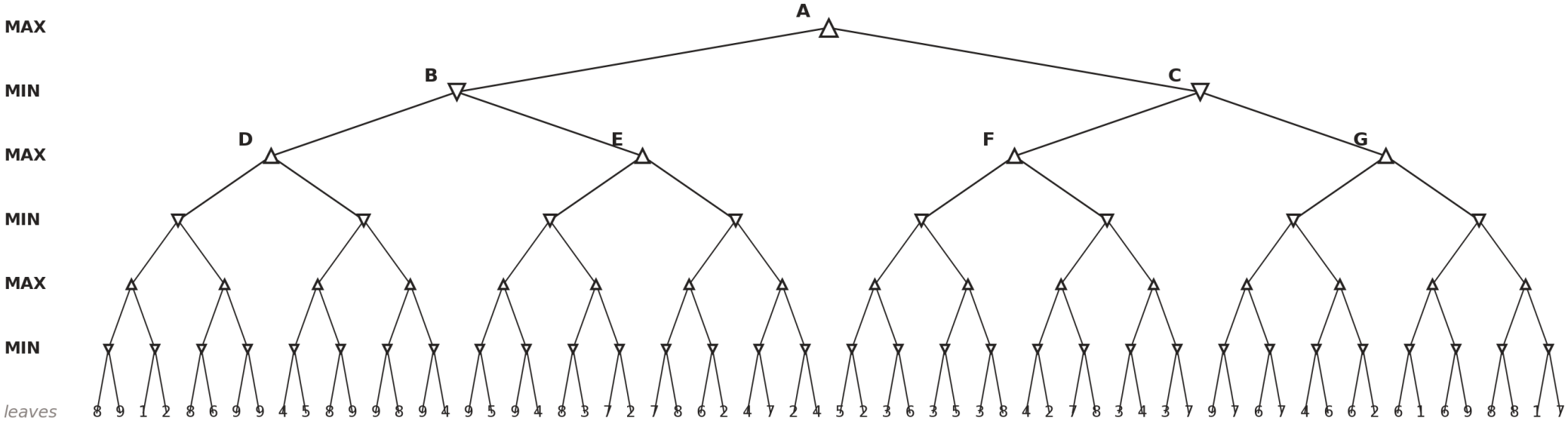
- Take the leftmost MIN node H and work up from its four pairs of leaves
- The first pair backs up 8, so the MAX node above it now has alpha at 8
- The next pair opens with 1, which is already at or below 8, so its second leaf is never generated



alpha is 8 after the first pair, so the 2 is never generated

Question 1

What is the minimax value at the root A



children are generated left to right

A 3

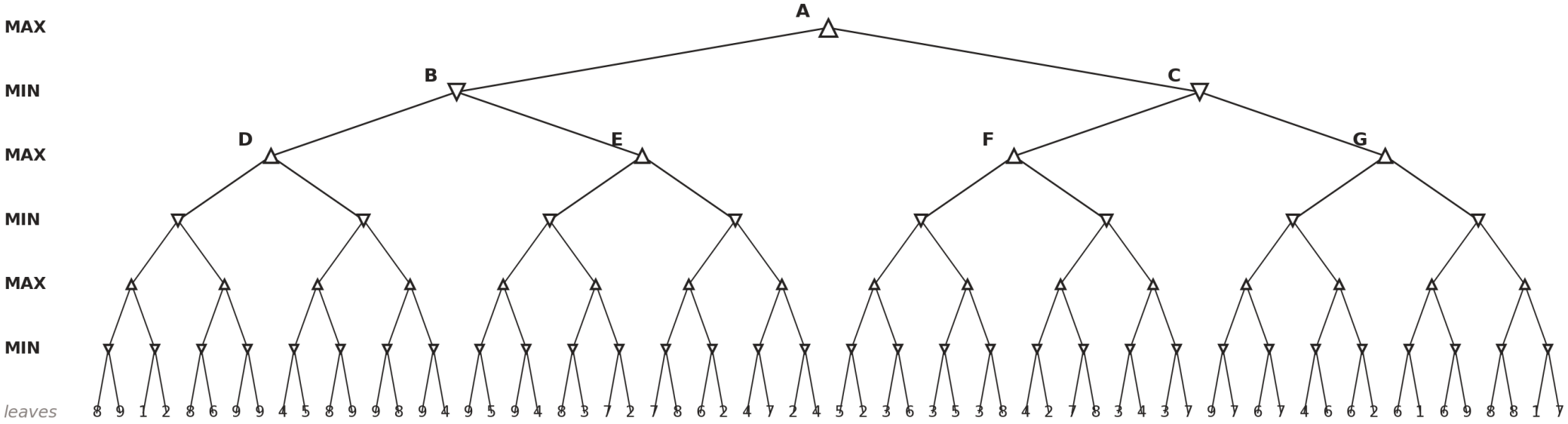
B 4

C 6

D 8

Question 2

Generating left to right, how many of the sixty-four leaves does alpha beta examine



children are generated left to right

A 24

B 31

C 39

D 48