Lecture 12

CS 6380 Artificial Intelligence

Tutorial: Alpha Beta Pruning (contd.)

Department of Computer Science and Engineering

Indian Institute of Technology Madras (IITM), India

Recap

Minimax in Thirty Seconds

- MAX takes the biggest child, MIN takes the smallest, and values flow up from the leaves

Two players	taking strict turns, with nothing at all happening in between them
Zero sum	my gain is exactly your loss, so one number describes the whole outcome
Perfect information	we both see the entire board at every moment of the game
No dice	nothing random ever intervenes between one decision and the next
A flawless opponent	MIN never blunders, not even once, and we plan against that

Today we break the last three of those on purpose.

You Cannot Beat the Exponent, So Halve It

- Five plies is not good enough, since a club player casually plans six to eight moves ahead
- We will never remove the exponent. What we can do is cut it in half

35

legal moves in a typical chess position

50 million

nodes in just five plies of lookahead

200 million

nodes a strong engine can afford per move

5

plies that buys you with plain minimax

Minimax Energy

Nobody. Absolutely nobody. Not a single person.

Minimax, at move one, carefully evaluating the four billionth position in a line where the opponent hangs the queen four times in a row.

If a line is already bad enough that you would never choose it, you do not need to know exactly how bad it is.

That sentence is the entire algorithm. Everything else is bookkeeping.

You Already Do This Without Writing It Down

- You hold a signed offer worth 100. A second company opens by saying the role tops out at 80
- Nobody sits through the four remaining interview rounds, and that walk out is a pruning decision

You	Your offer of 100	The recruiter saying 80	Walking out early
the MAX node, wanting the largest number you can get	this is alpha, a floor, already in the bank	a ceiling on that branch, and it can only fall	the cutoff, since the exact number changes nothing

Using Alpha and Beta

The Two Numbers

alpha, the floor that belongs to MAX

MAX has already secured at least this much somewhere on the path from the root down to here. It starts at minus infinity and only ever climbs.

beta, the ceiling that belongs to MIN

MIN can already hold you to at most this much on that same path. It starts at plus infinity and only ever falls.

They live on the path not on a node. A node inherits them from its parent and passes them down

They are monotone alpha never falls and beta never rises as you descend, so the window only tightens

They are a window the range of values still worth caring about. The moment it shuts, stop

Two Rules and You Can Trace Any Tree

- Cut when the branch you are exploring can no longer beat something the other player already refused to give up

Cut at a MAX node

stop as soon as v reaches beta or above

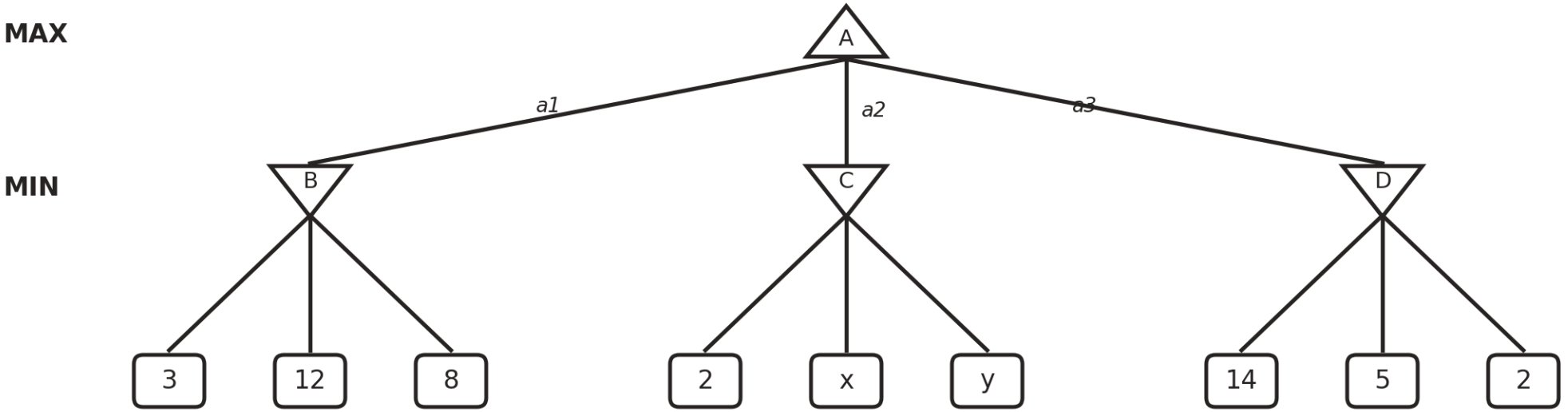
Cut at a MIN node

stop as soon as v falls to alpha or below

MAX updates alpha	and tests its running value against the beta it inherited from above
MIN updates beta	and tests its running value against the alpha it inherited from above
Never the other way	a node writing to the other player's bound is a bug, and a common one

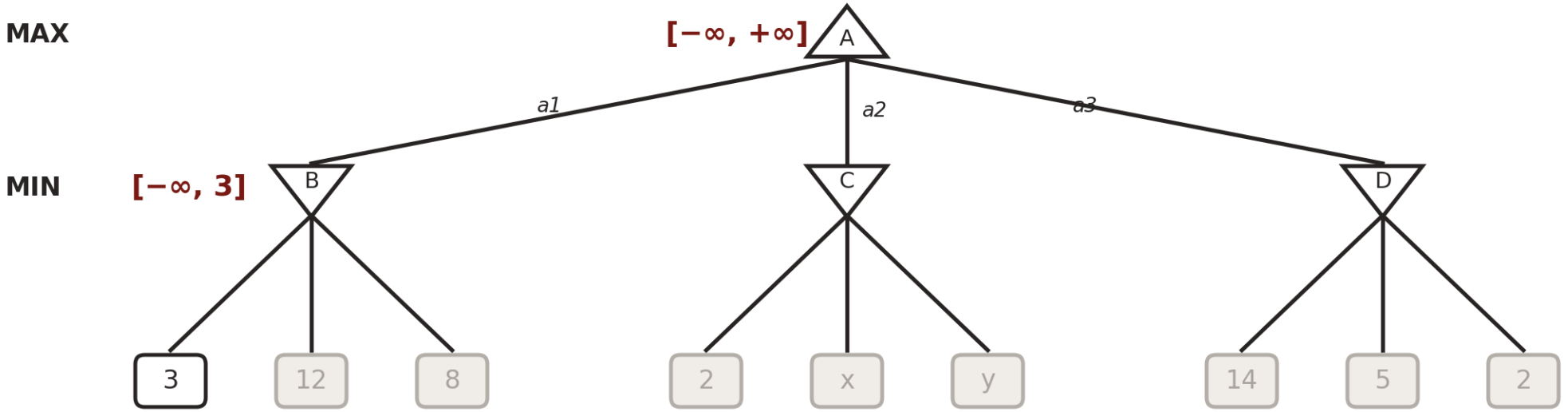
The Tree We Will Trace Together

- Nine leaves, depth two, and we go strictly left to right
- Cover the leaves with your hand and guess how many of the nine we will actually examine



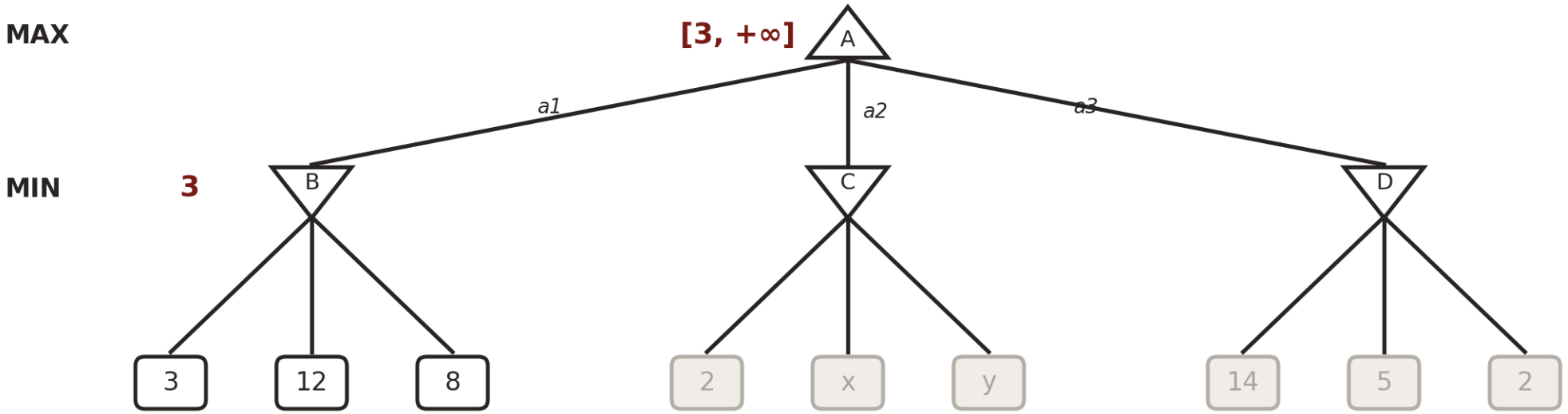
Step One, B Already Has a Ceiling

- B is a MIN node that has seen a 3, so B is at most 3 and can only fall from here
- No cut is possible yet, because alpha is still minus infinity and nothing is in the bank



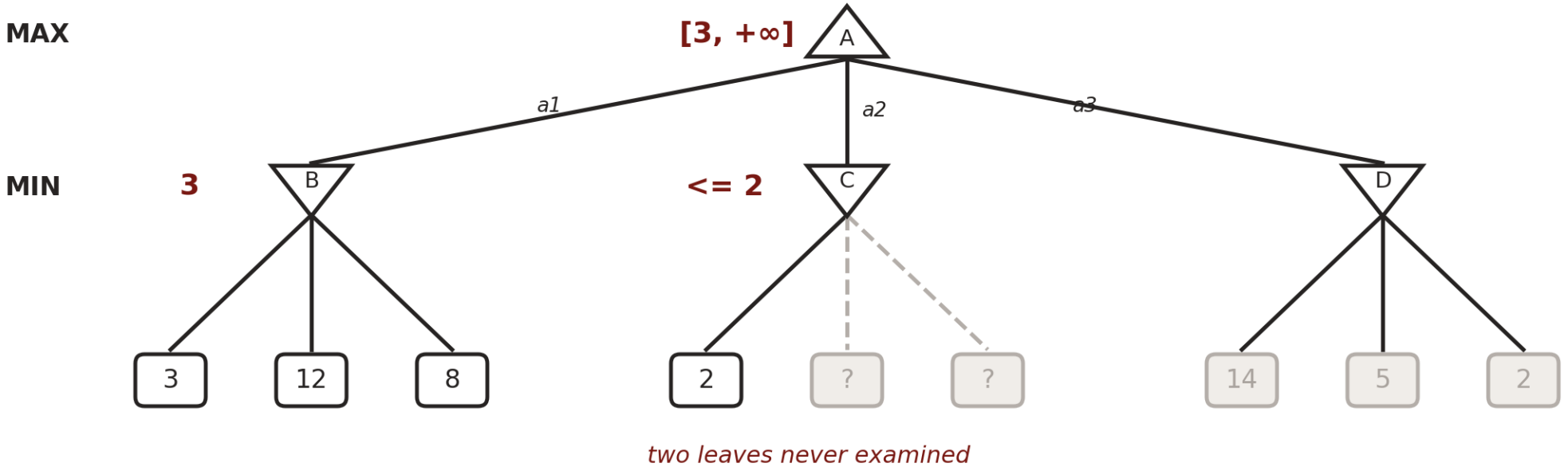
Step Two, The Root Has Money in the Bank

- 12 then 8 arrive and MIN ignores both, so with every child seen B is exactly 3
- The root is now at least 3, so alpha becomes 3 and the search finally grows teeth



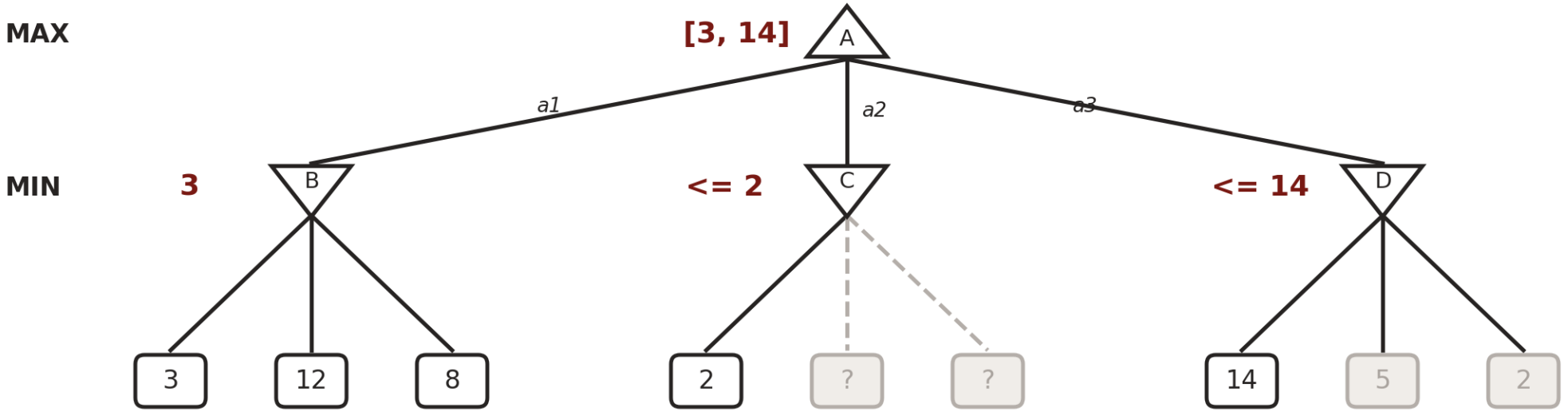
Step Three, The First Cut

- C sees a 2, so C is at most 2, while MAX already holds 3 through B
- At C we have v equal to 2 and alpha equal to 3, so v is at or below alpha and the last two leaves die



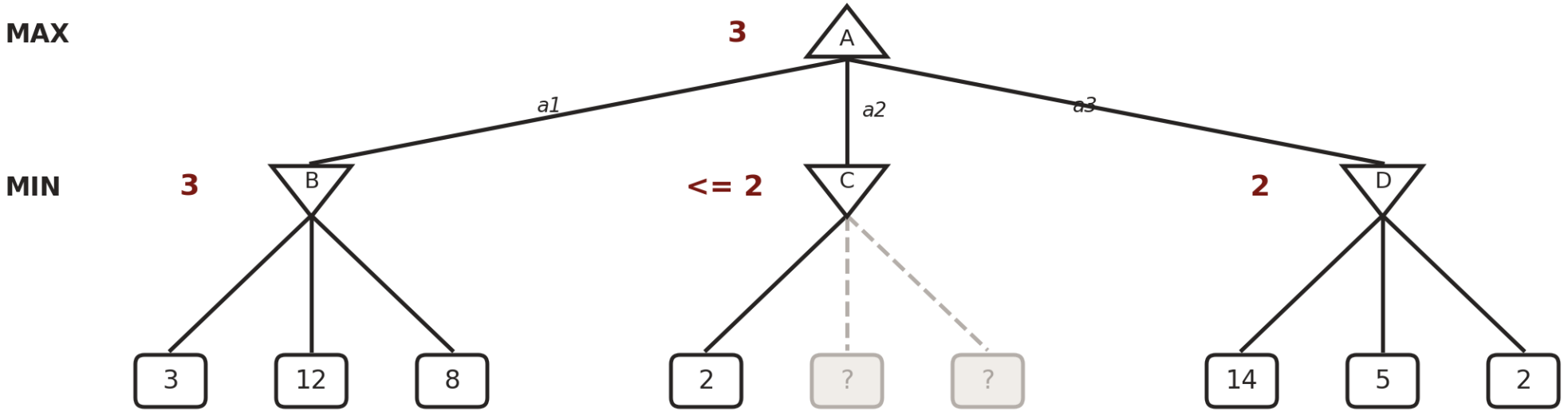
Step Four, D Refuses to Be Pruned

- D sees 14 first, and 14 does not fall at or below alpha, so D might still be the best move
- Alpha beta never gambles. It skips only the work it can prove is irrelevant



The Answer, and Why It Is Provably Right

- max(min(3,12,8), min(2,x,y), min(14,5,2)) is max(3, z, 2) where z is at most 2, so the root is 3
- The values of x and y are mathematically incapable of changing the answer



seven leaves evaluated, and the answer is the one minimax gives

Thirty Second Check

- Few of these claims are wrong. Shout out which, and say why

A If x turns out to be 1000, the root value becomes 1000

B Alpha beta found a different move than minimax would have

C Visiting D before C might have pruned more

Thirty Second Check

- Few of these claims are wrong. Shout out which, and say why

A If x turns out to be 1000, the root value becomes 1000

False. MIN is choosing at C, already holds a 2, and would never hand over a 1000.

B Alpha beta found a different move than minimax would have

False. Identical value and identical move, on every tree, every single time.

C Visiting D before C might have pruned more

FALSE, given the values, its same amount of pruning.

Minimax Plus Four Lines

- Delete the four marked lines and plain minimax is back

```
MAX-VALUE(state, alpha, beta)

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

```
MIN-VALUE(state, alpha, beta)

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

Check before you update test the cut condition first, then raise your own bound. Swapping the two is the classic bug

A cut returns a bound not always the true value of that node. The root value is still exact, which is all we needed

Where Marks Quietly Disappear

Pruning at the root

The root has no ancestor, so its beta stays at plus infinity forever and no cut can ever fire on a direct child of the root.

An equals sign on a pruned node

A node that cut early has a bound, not a value. Write at most 2 rather than equals 2, because markers look for exactly this.

Comparing the wrong way

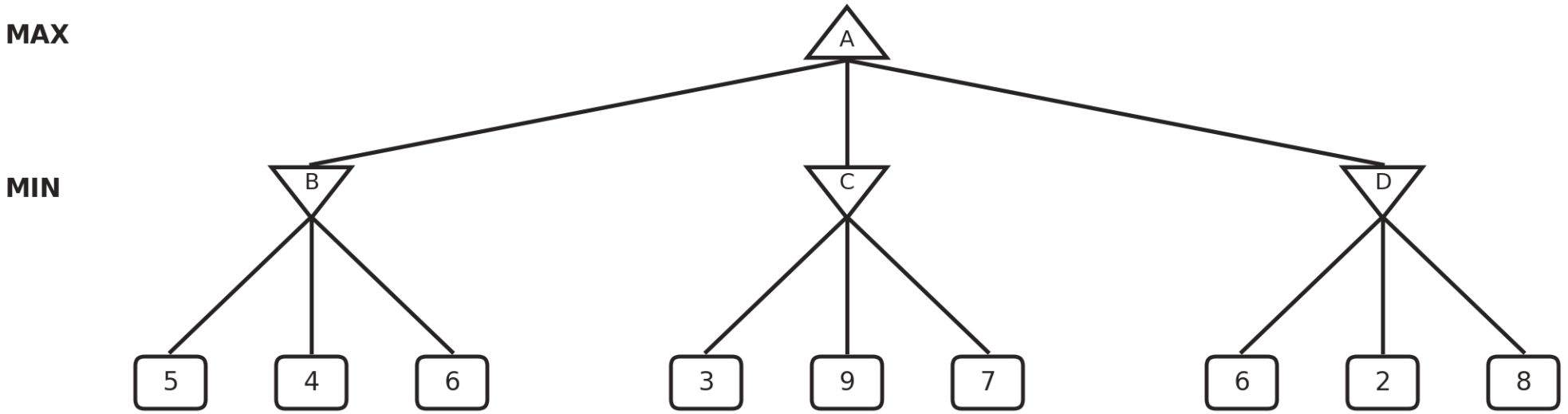
MIN compares against alpha and MAX compares against beta, always against the bound inherited from above. Your own bound can never fire.

Assuming a fixed leaf count

Reorder the children and a different number of leaves survives, so a question has to state the order before it can have an answer.

Question 1 From Last Class

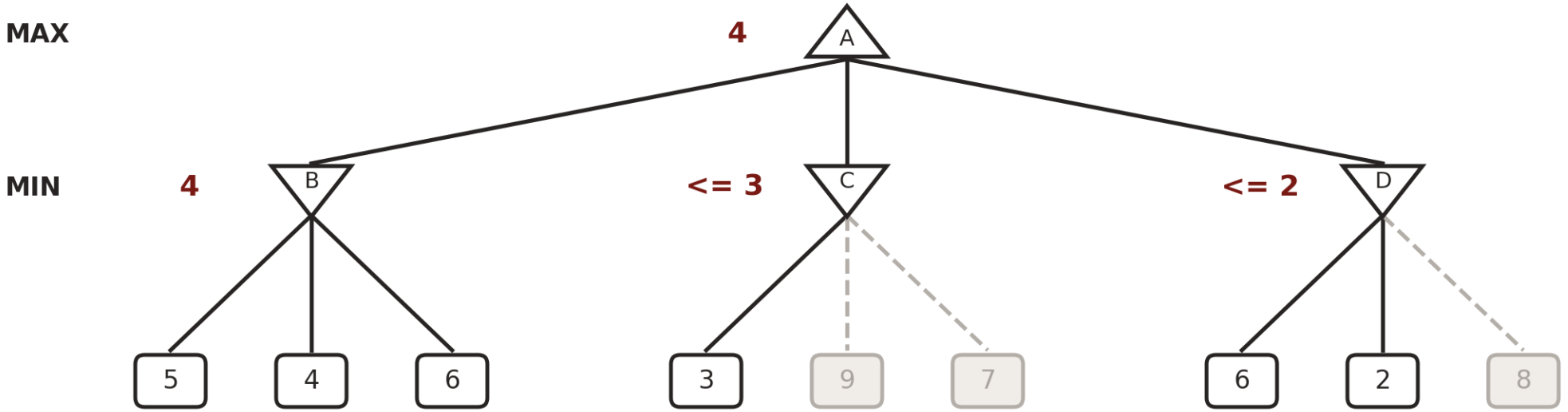
- Left to right. Value every node, mark every cut, and count the leaves you evaluated



A 2	**B** 3	**C** 4	**D** 6

Question 1, Solved

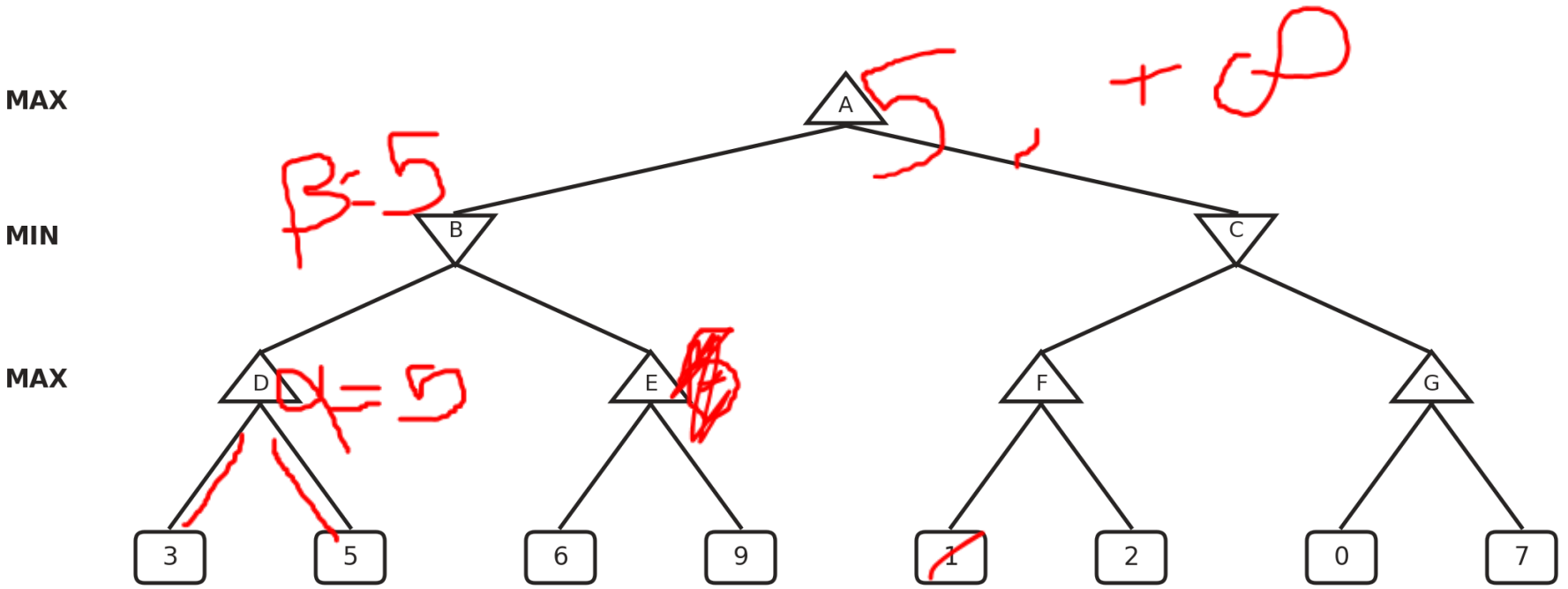
- B settles at 4 so alpha becomes 4, then C sees 3 and dies, then D survives a 6 before a 2 kills it
- The 9 under C is irrelevant, because MIN is choosing there and already holds a 3



six leaves evaluated, and the move is B

Question 2 From Last Class

- Three plies, so a MAX node can now cut on a beta inherited from above
- Predict first. Can a whole subtree disappear here, or only single leaves



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