CS 471/571(Fall 2023): Introduction to Artificial Intelligence

Lecture 8: Adversarial Search

Thanh H. Nguyen

Source: http://ai.berkeley.edu/home.html

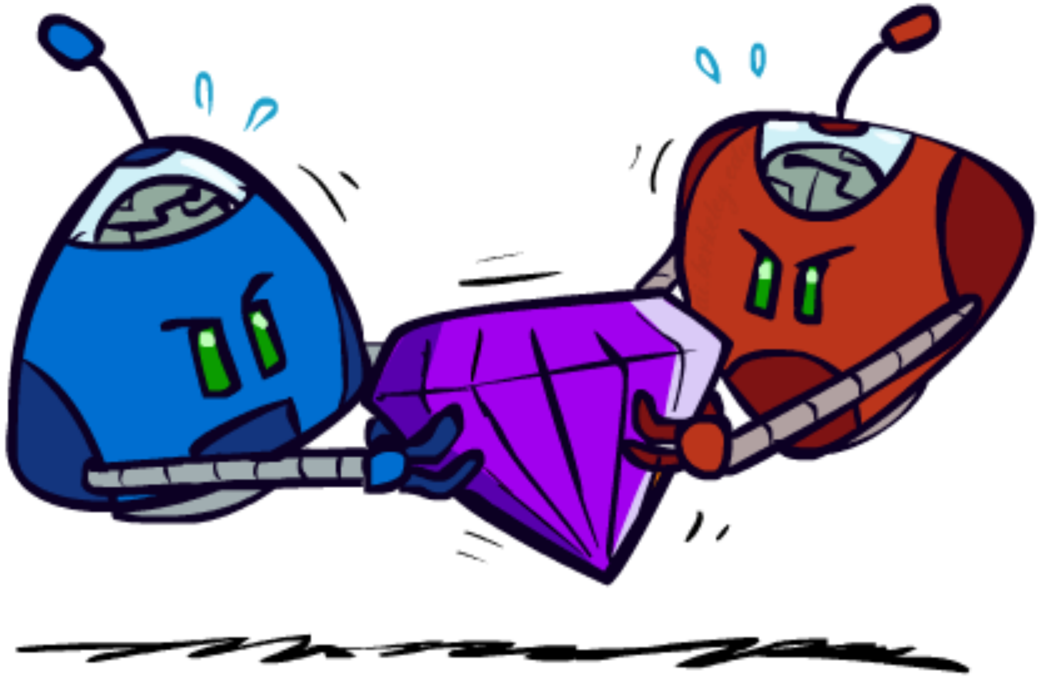
Announcement and Reminder

- Programming project 1:
  - Deadline: Oct 16th, 2023

- Written assignment 2:
  - Will be posted tomorrow
  - Deadline: Oct 25th, 2023

- No office hour today
  - Make-up office hour on Monday (Oct 16th), 1:30 pm – 3:30 pm.

Adversarial Games



Types of Games

- Many different kinds of games!

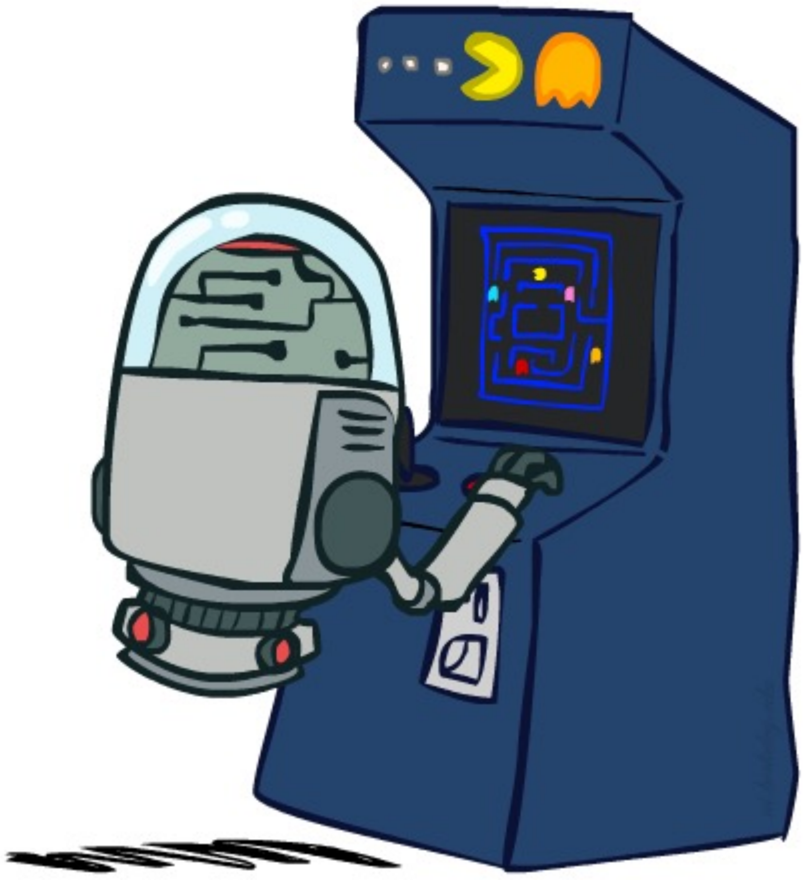
- Axes:
  - Deterministic or stochastic?
  - One, two, or more players?
  - Zero sum?
  - Perfect information (can you see the state)?

- Want algorithms for calculating a strategy (policy) which recommends a move from each state

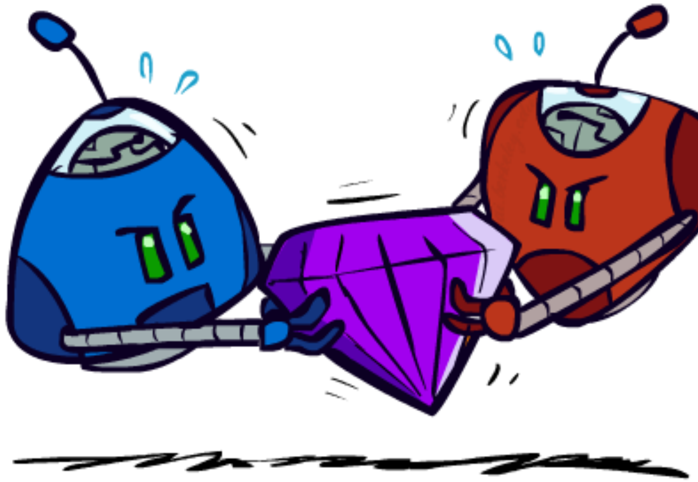
Deterministic Games

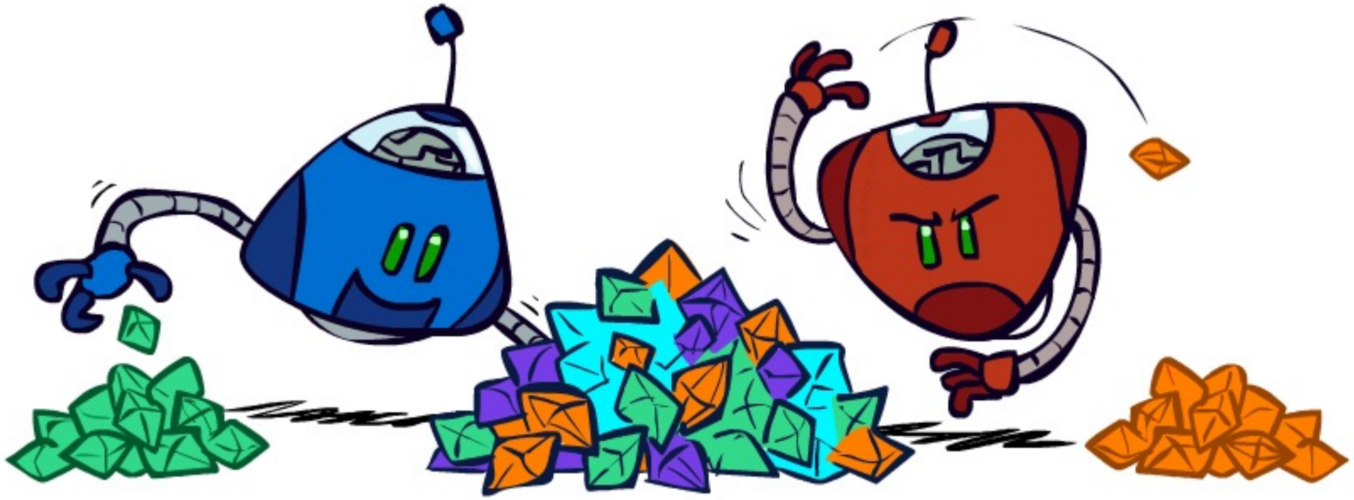
- Many possible formalizations, one is:
  - States: S (start at s_0)
  - Players: P={1...N} (usually take turns)
  - Actions: A (may depend on player / state)
  - Transition Function: SxA $\rightarrow$ S
  - Terminal Test: S $\rightarrow$ {t,f}
  - Terminal Utilities: SxP $\rightarrow$ R

- Solution for a player is a policy: S $\rightarrow$ A



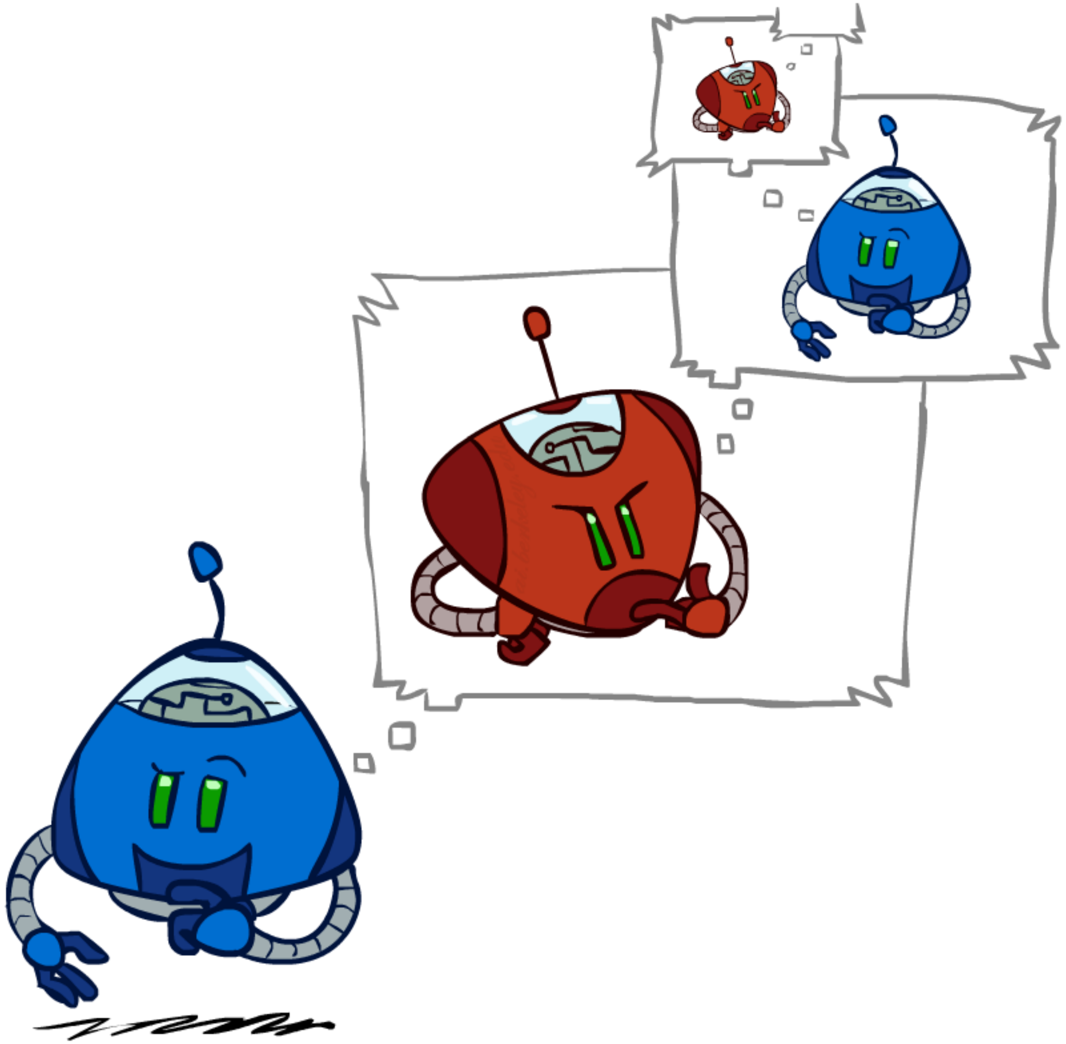
Zero-Sum Games



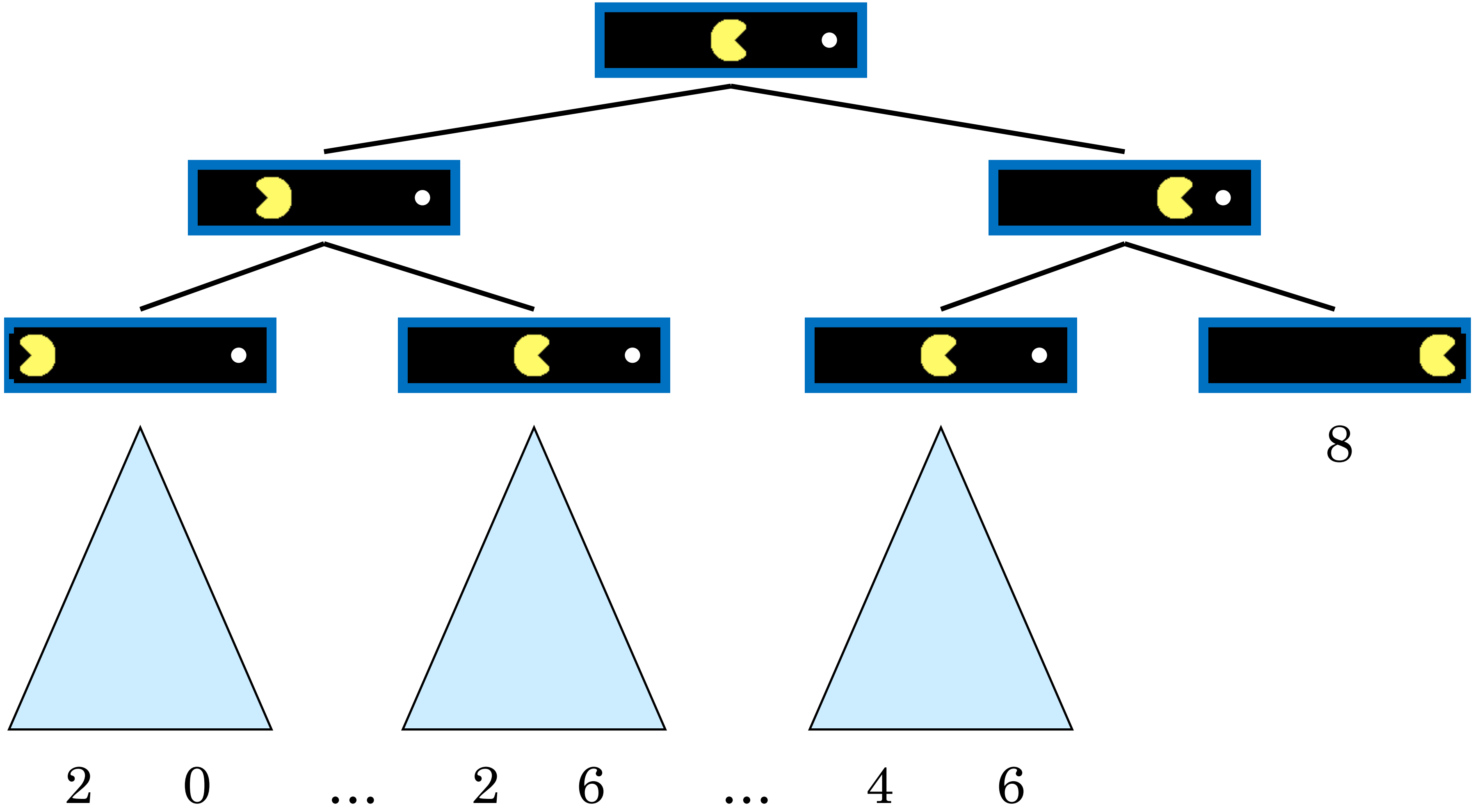


- Zero-Sum Games
  - Agents have opposite utilities (values on outcomes)
  - Lets us think of a single value that one maximizes and the other minimizes
  - Adversarial, pure competition
- General Games
  - Agents have independent utilities (values on outcomes)
  - Cooperation, indifference, competition, and more are all possible
  - More later on non-zero-sum games

Adversarial Search



Single-Agent Trees

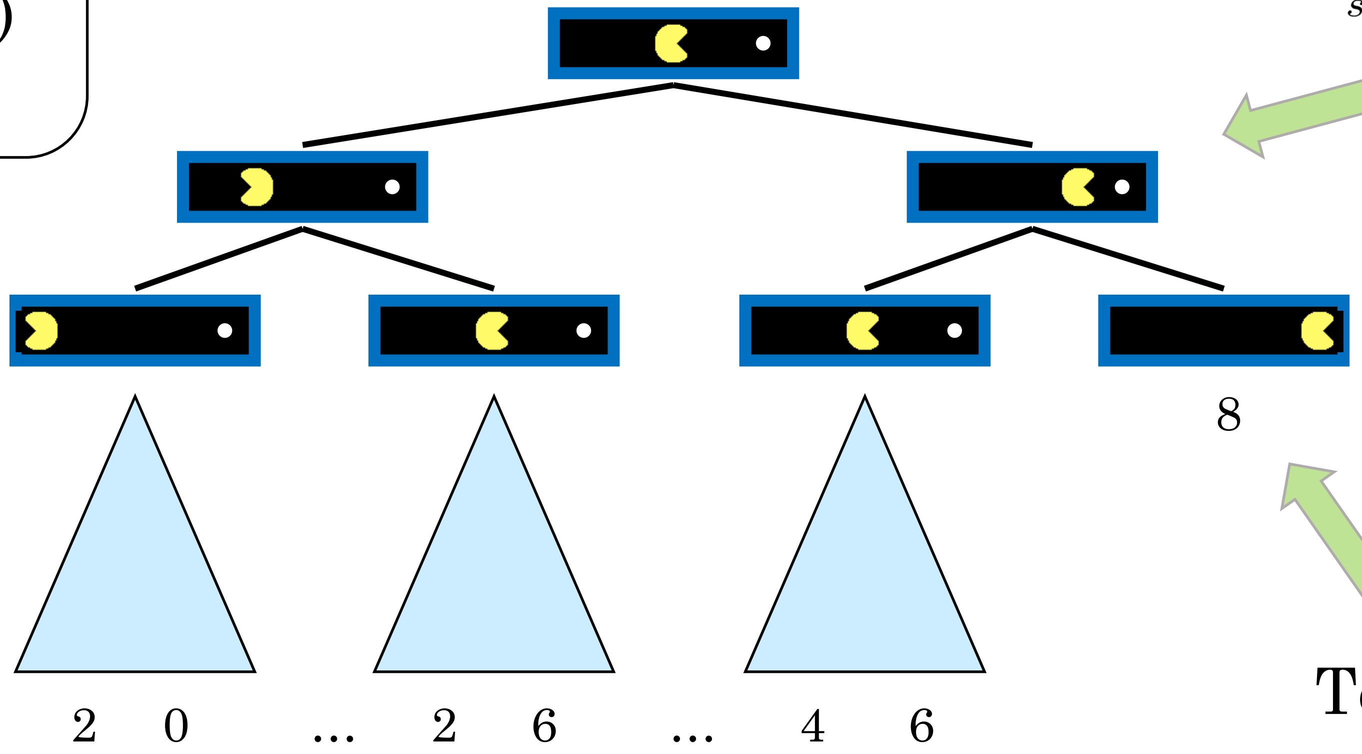


Value of a State

Value of a state:
The best achievable outcome (utility) from that state

Non-Terminal States:

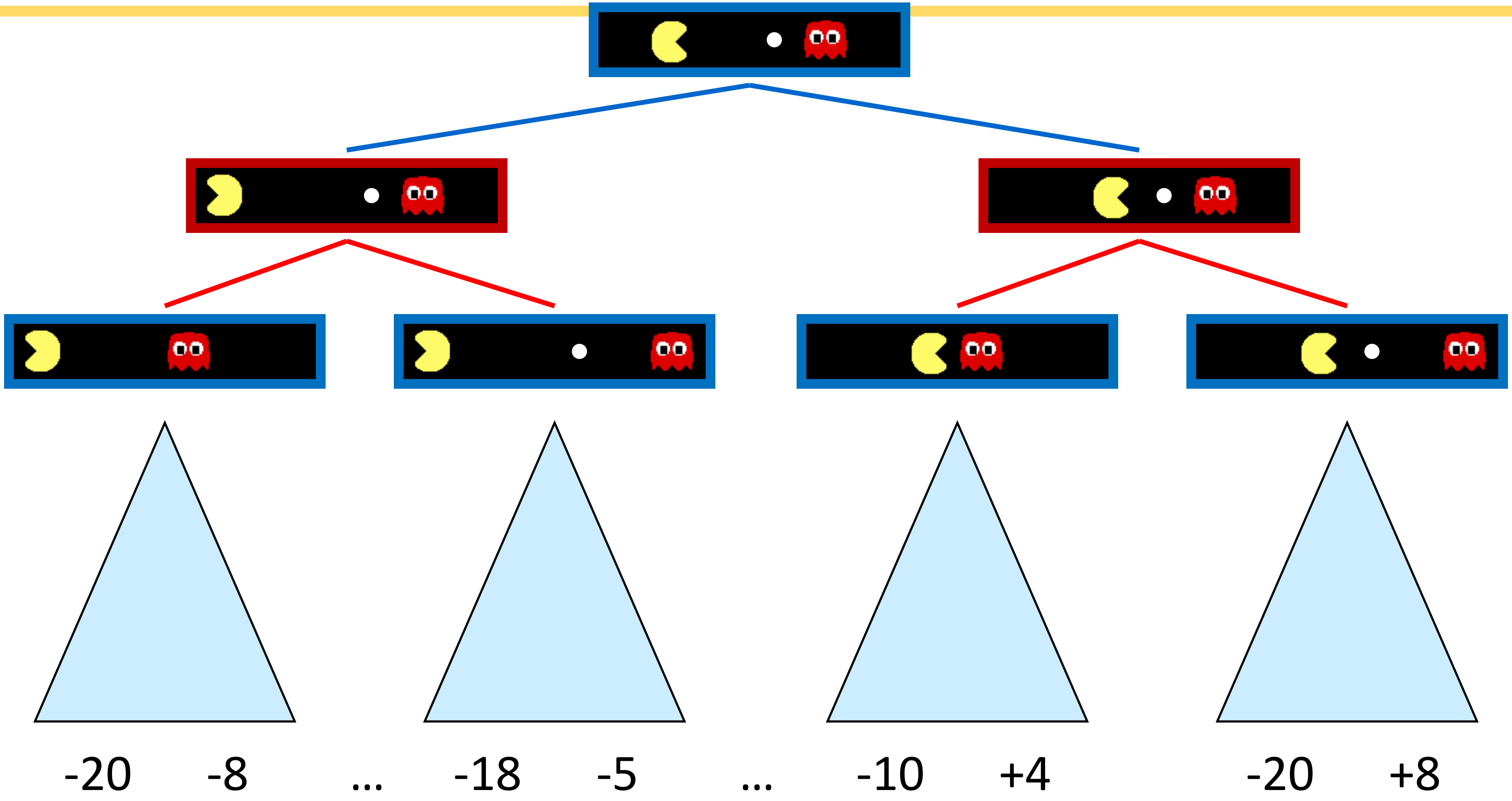
$$V(s) = \max_{s' \in \text{children}(s)} V(s')$$



Terminal States:

$$V(s) = \text{known}$$

Adversarial Game Trees



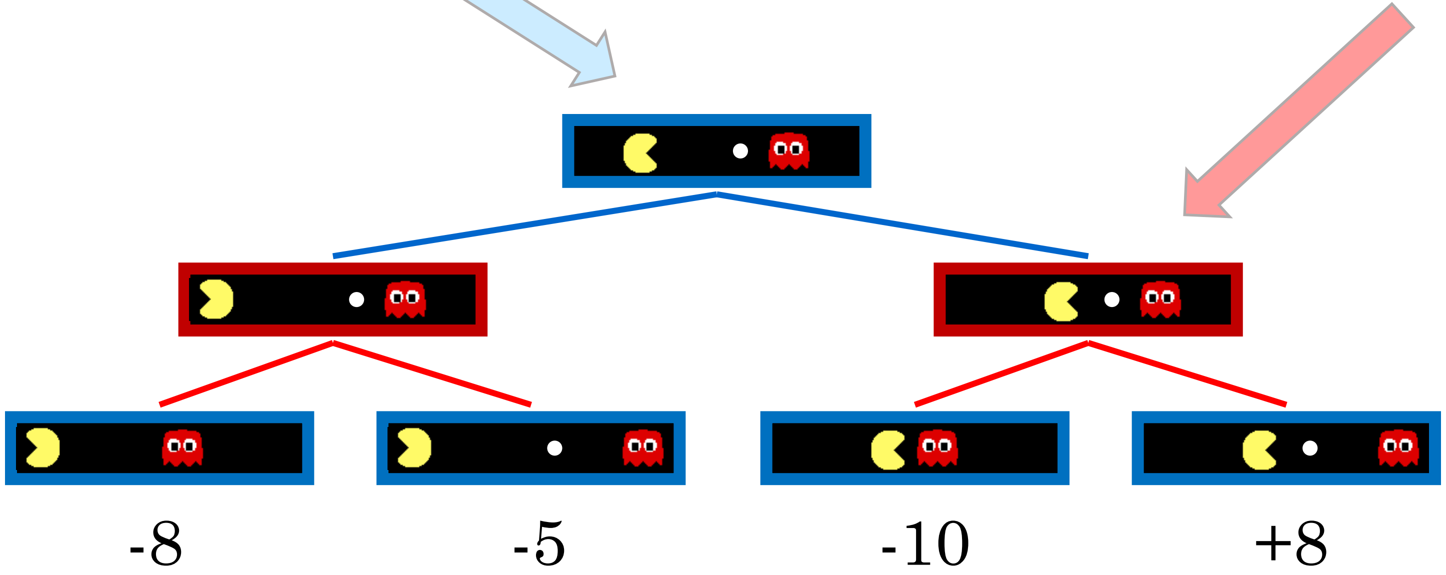
Minimax Values

States Under Agent's Control:

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

States Under Opponent's Control:

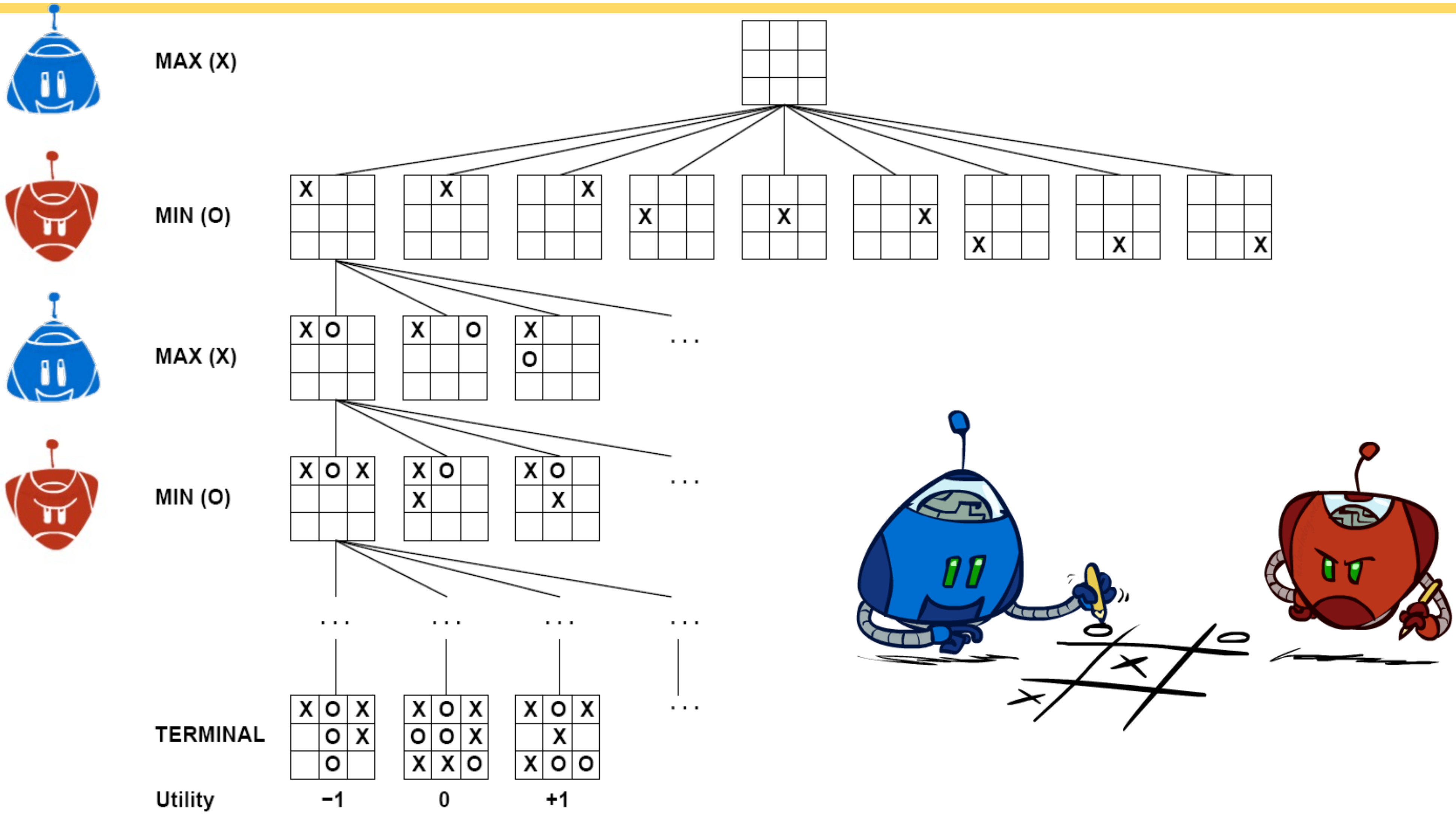
$$V(s') = \min_{s \in \text{successors}(s')} V(s)$$



Terminal States:

$$V(s) = \text{known}$$

Tic-Tac-Toe Game Tree

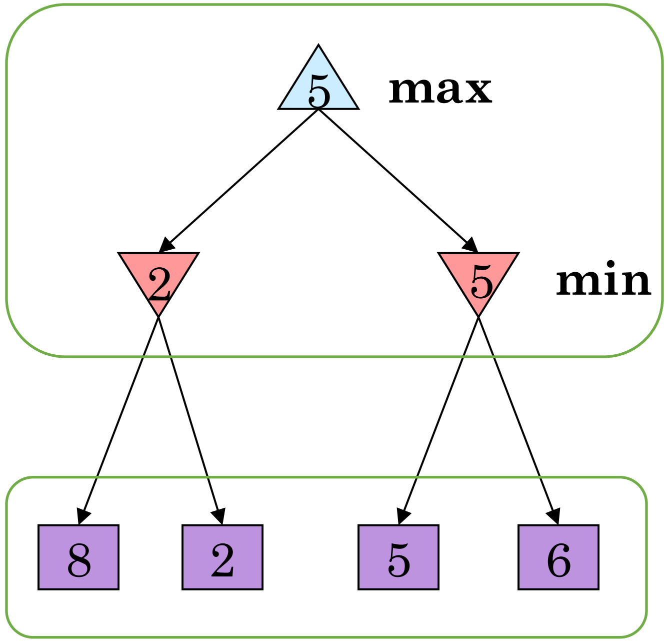


Adversarial Search (Minimax)

- Deterministic, zero-sum games:
  - Tic-tac-toe, chess, checkers
  - One player maximizes result
  - The other minimizes result

- Minimax search:
  - A state-space search tree
  - Players alternate turns
  - Compute each node's minimax value: the best achievable utility against a rational (optimal) adversary

Minimax values: computed recursively



Terminal values: part of the game

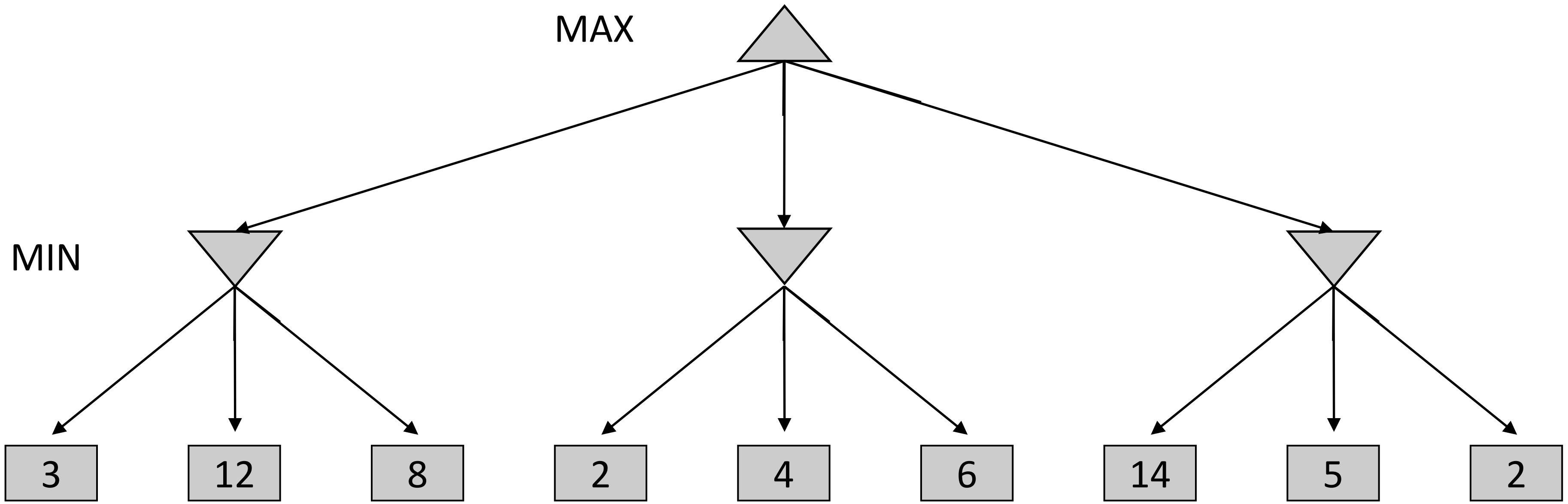
Minimax Implementation

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)
```

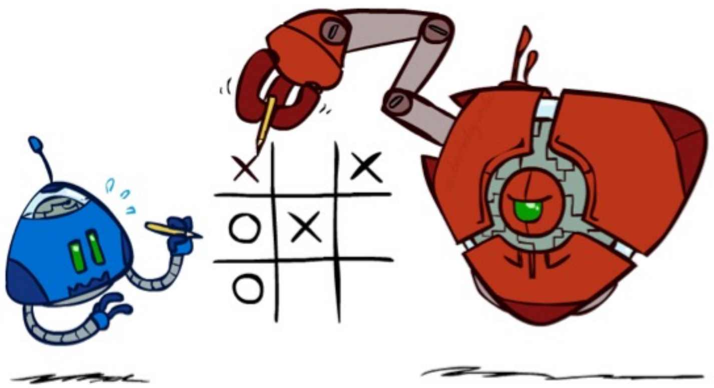
```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

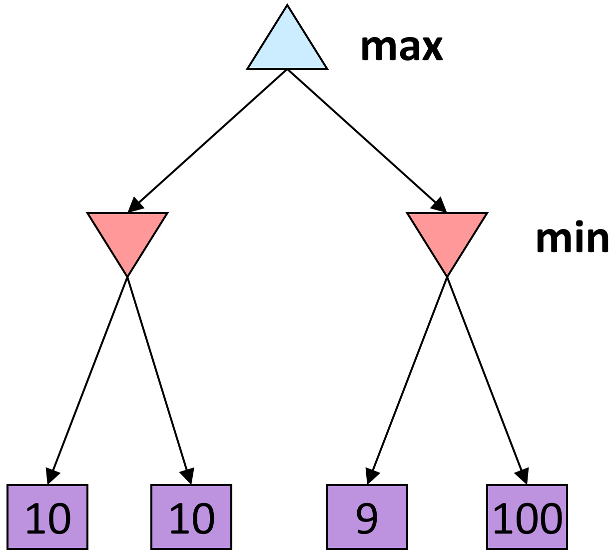
```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

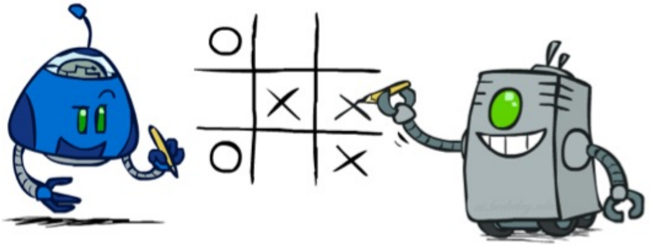
Minimax Example



Minimax Properties



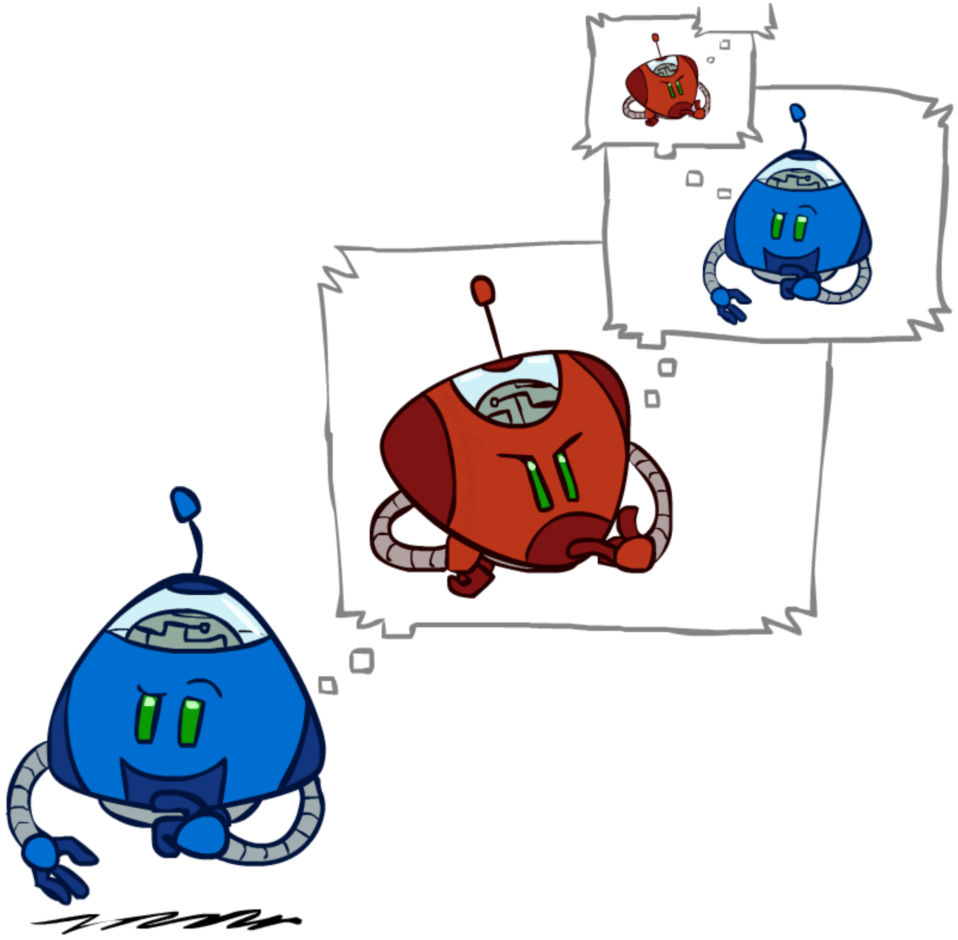




Optimal against a perfect player. Otherwise?

Minimax Efficiency

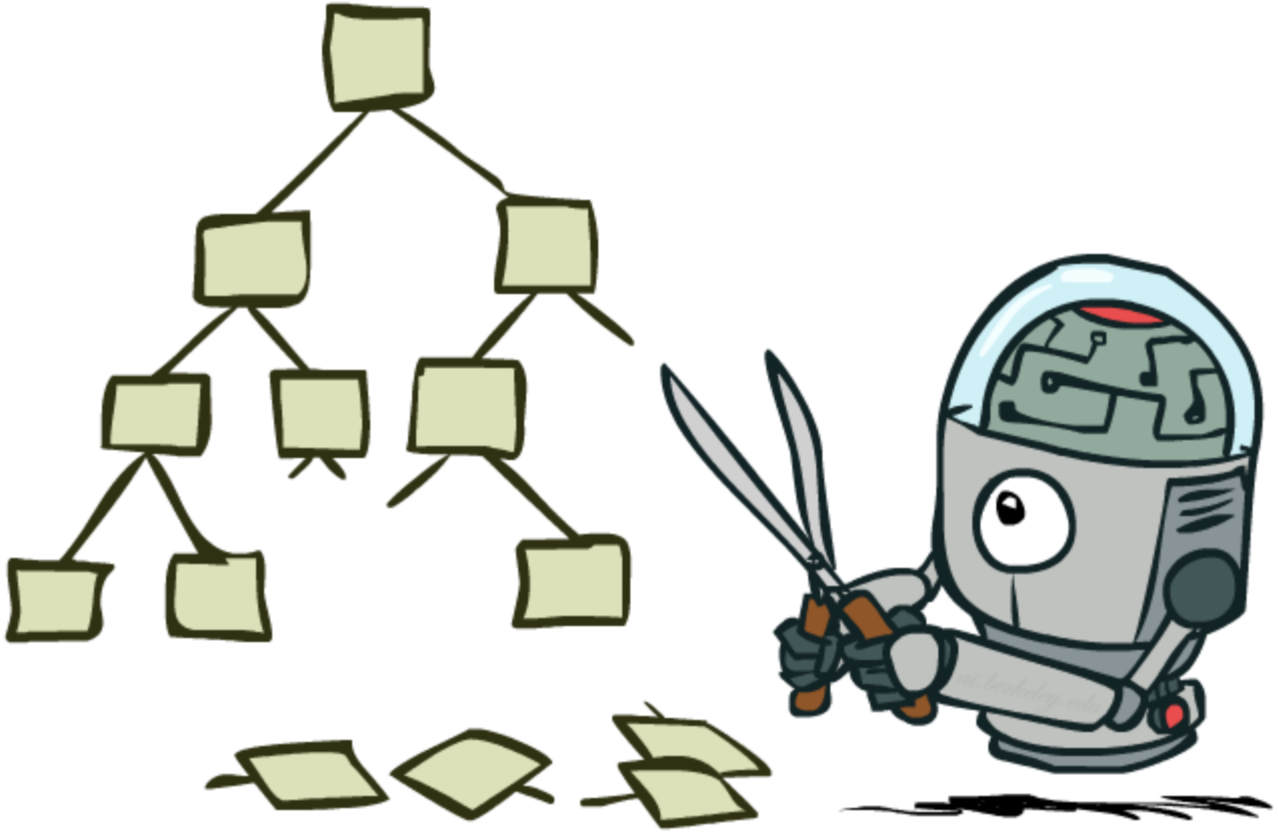
- How efficient is minimax?
  - Just like (exhaustive) DFS
  - Time: $O(b^m)$
  - Space: $O(bm)$



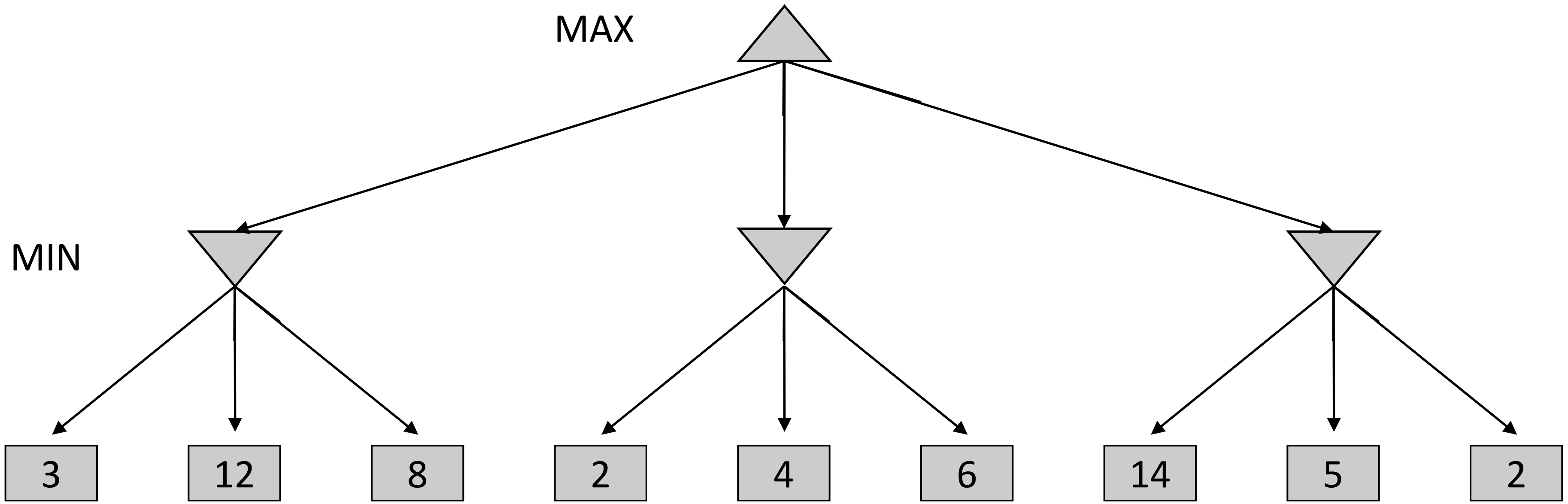
- Example: For chess, $b \approx 35$, $m \approx 100$
  - Exact solution is completely infeasible
  - But, do we need to explore the whole tree?

Resource Limits

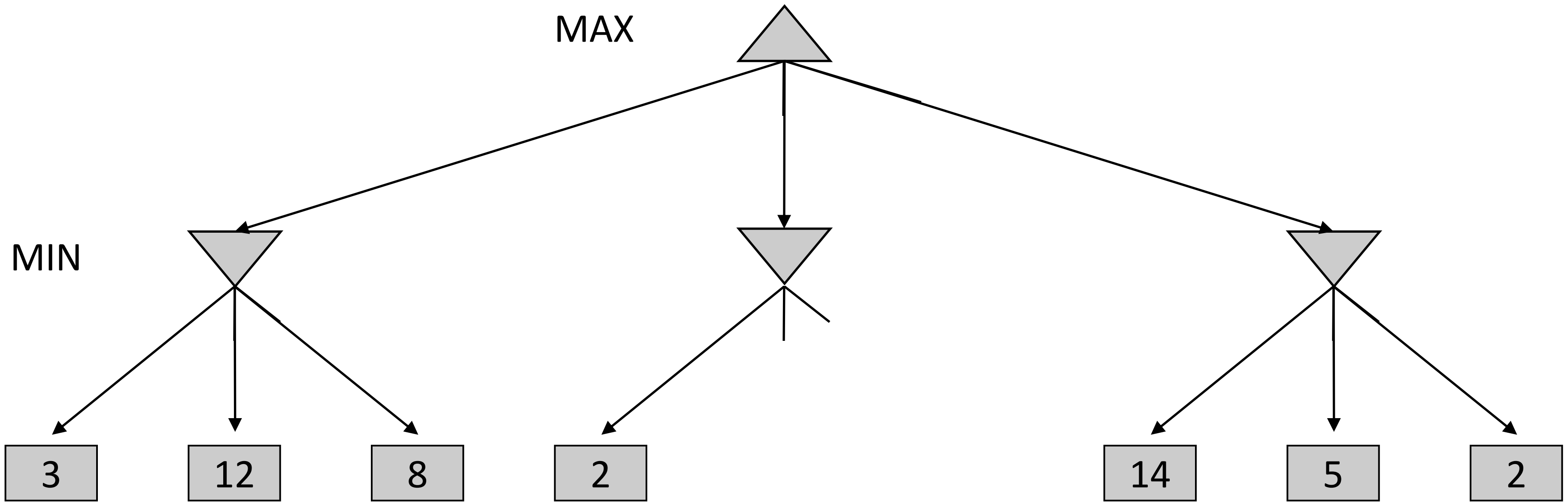
Game Tree Pruning



Minimax Example



Minimax Pruning



Alpha-Beta Pruning

- **Alpha** α: value of the best choice so far for MAX (lower bound of Max utility)
- **Beta** β: value of the best choice so far for MIN (upper bound of Min utility)
- Expanding at MAX node **n**: update α
  - If a child of **n** has value greater than β, stop expanding the MAX node **n**
  - Reason: MIN parent of **n** would not choose the action which leads to **n**

- At MIN node **n**: update β
  - If a child of **n** has value less than α, stop expanding the MIN node **n**
  - Reason: MAX parent of **n** would not choose the action which leads to **n**

Alpha-Beta Implementation

```
def value(state, α, β):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state, α, β)
    if the next agent is MIN: return min-value(state, α, β)
```

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

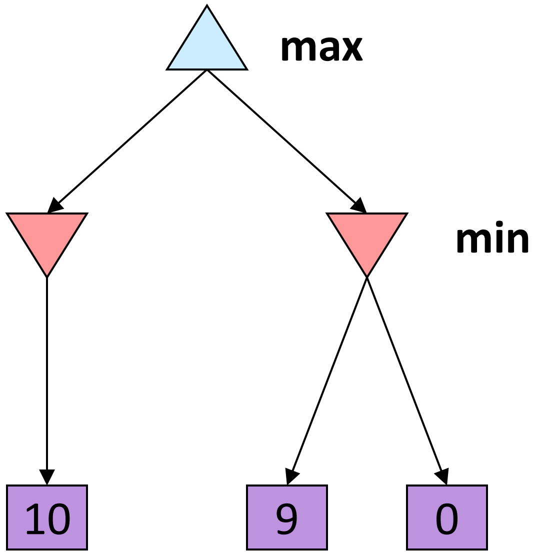
```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

Alpha-Beta Pruning Properties

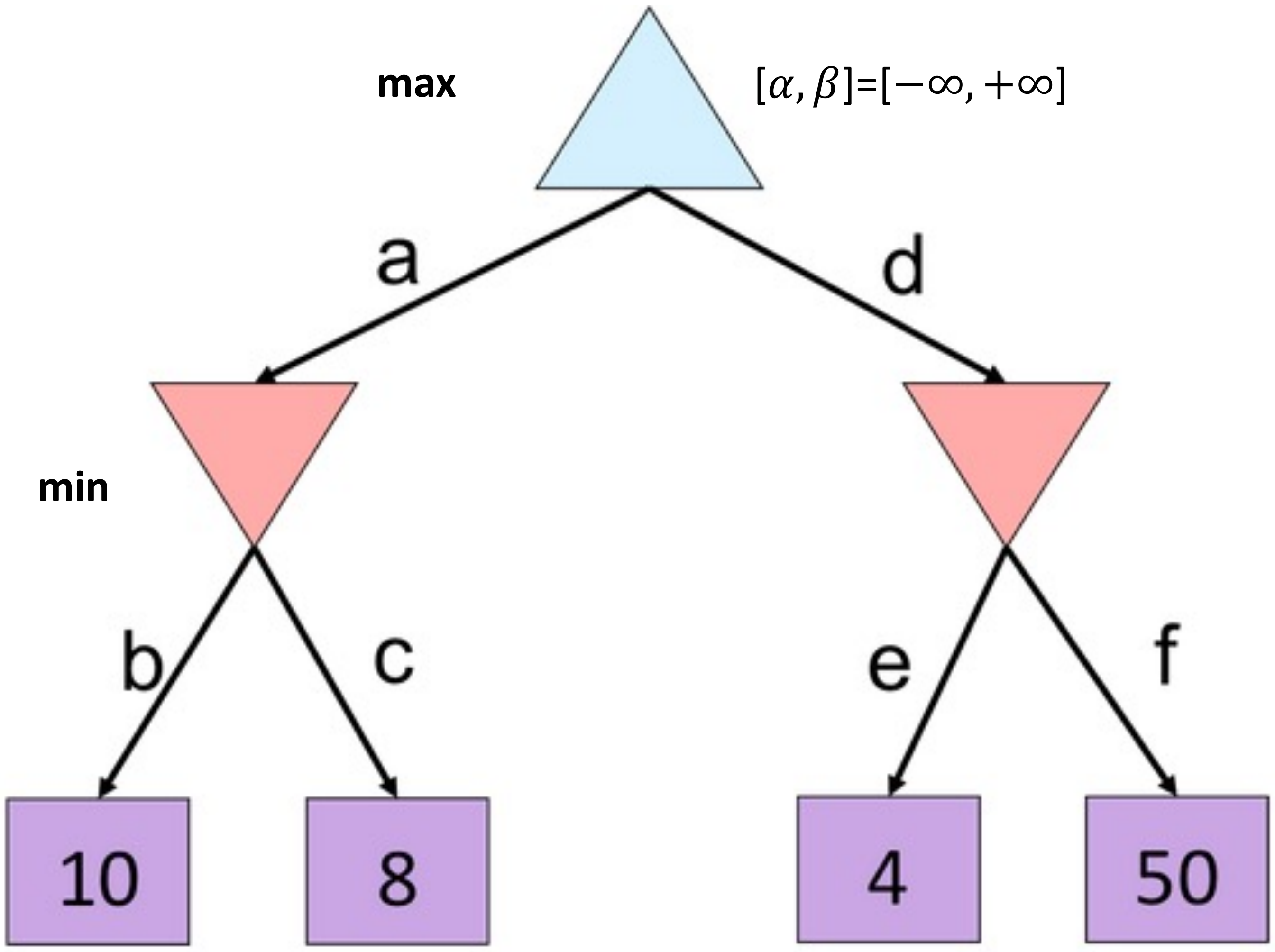
- This pruning has no effect on minimax value computed for the root!

- Values of intermediate nodes might be wrong
  - Important: children of the root may have the wrong value
  - So the most naïve version won't let you do action selection

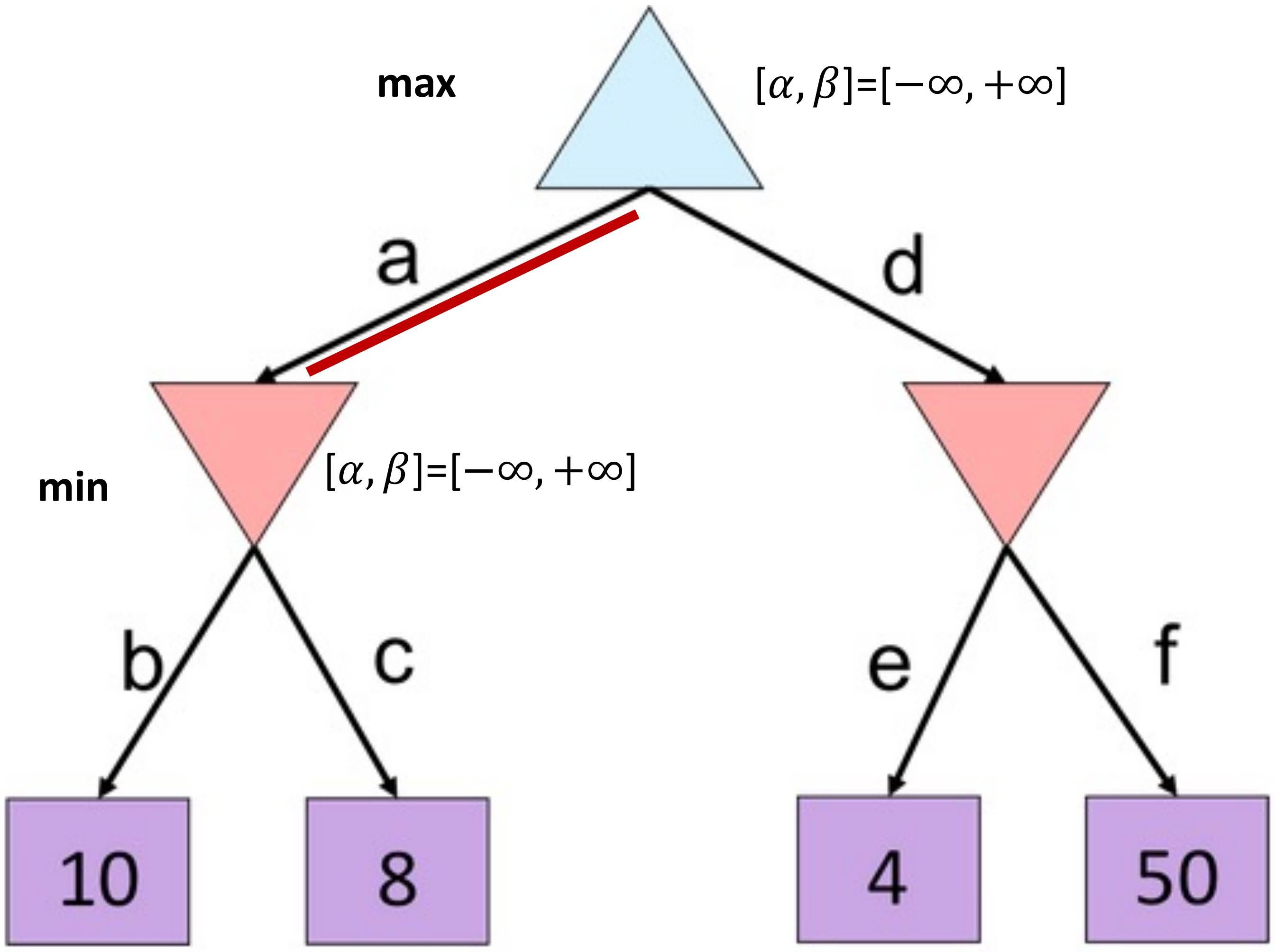
- Good child ordering improves effectiveness of pruning



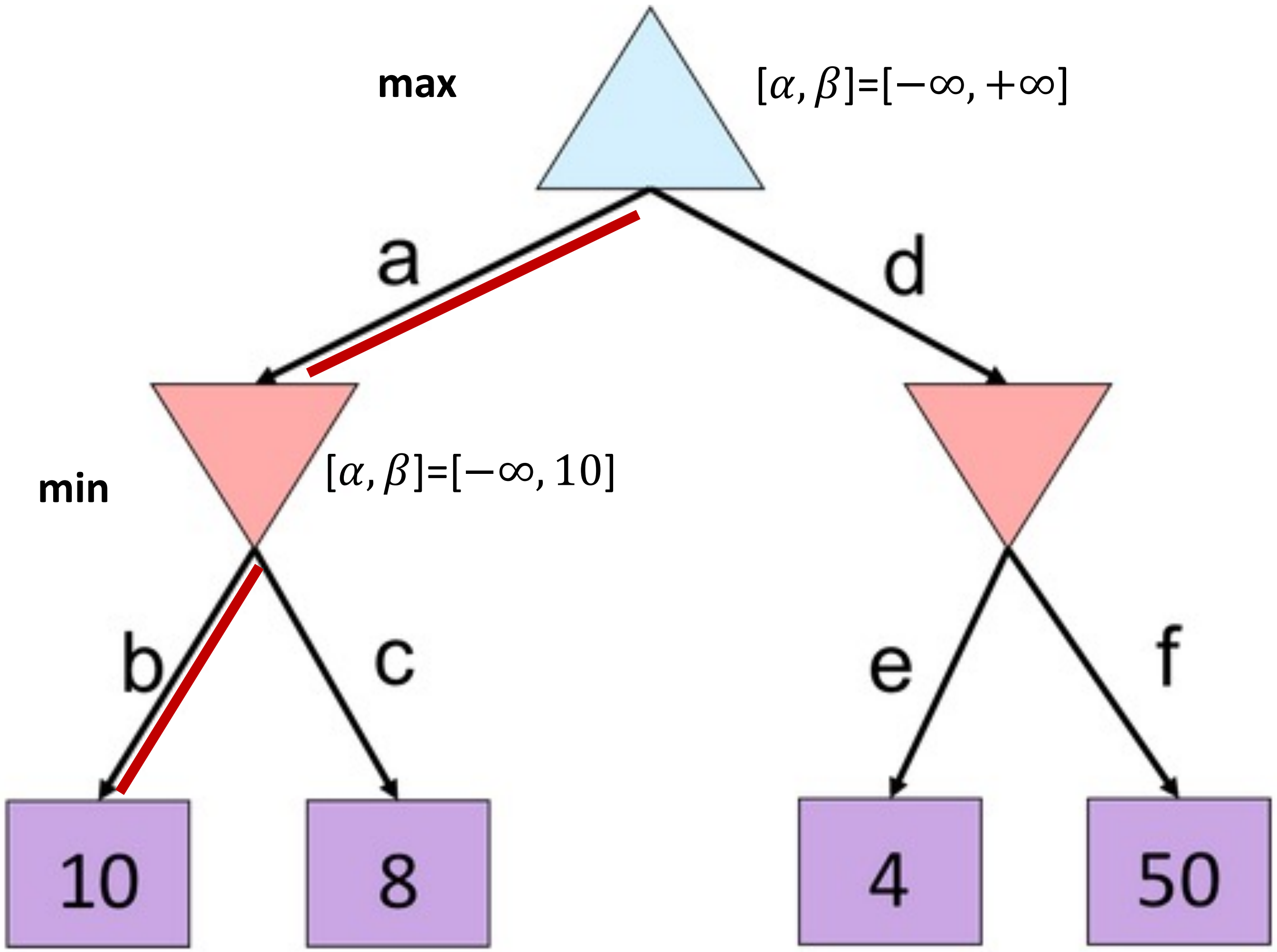
Alpha-Beta Quiz



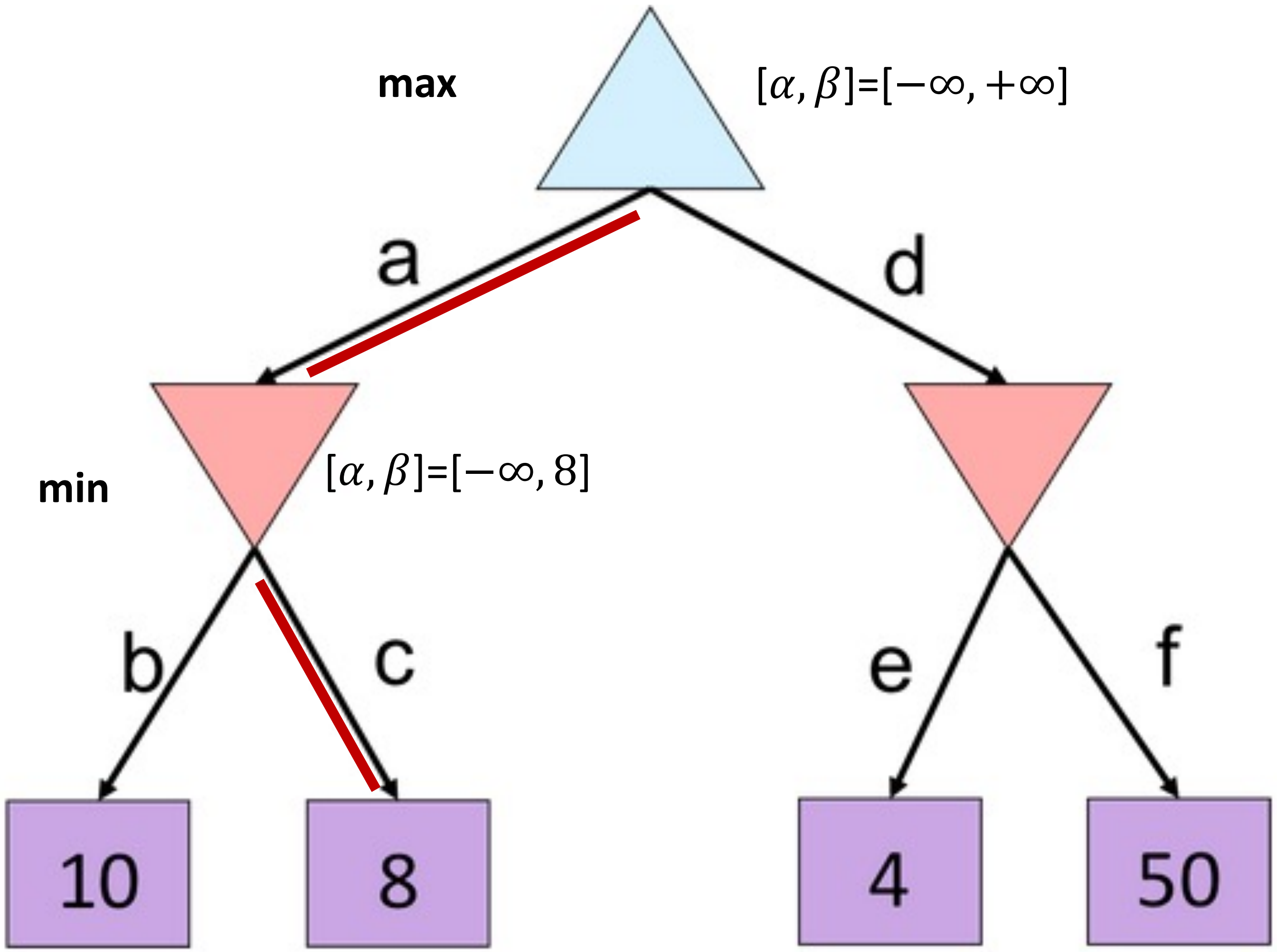
Alpha-Beta Quiz



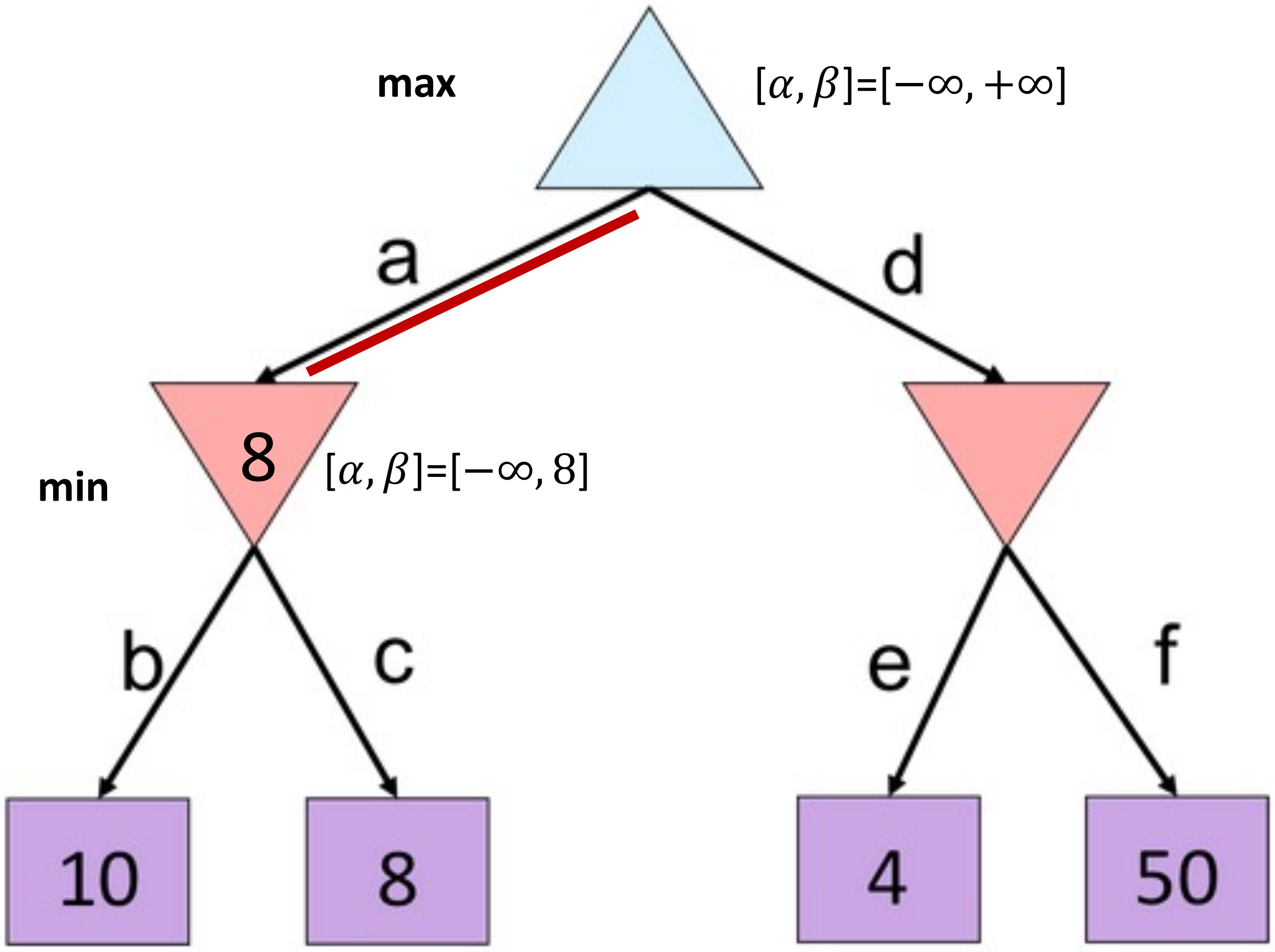
Alpha-Beta Quiz



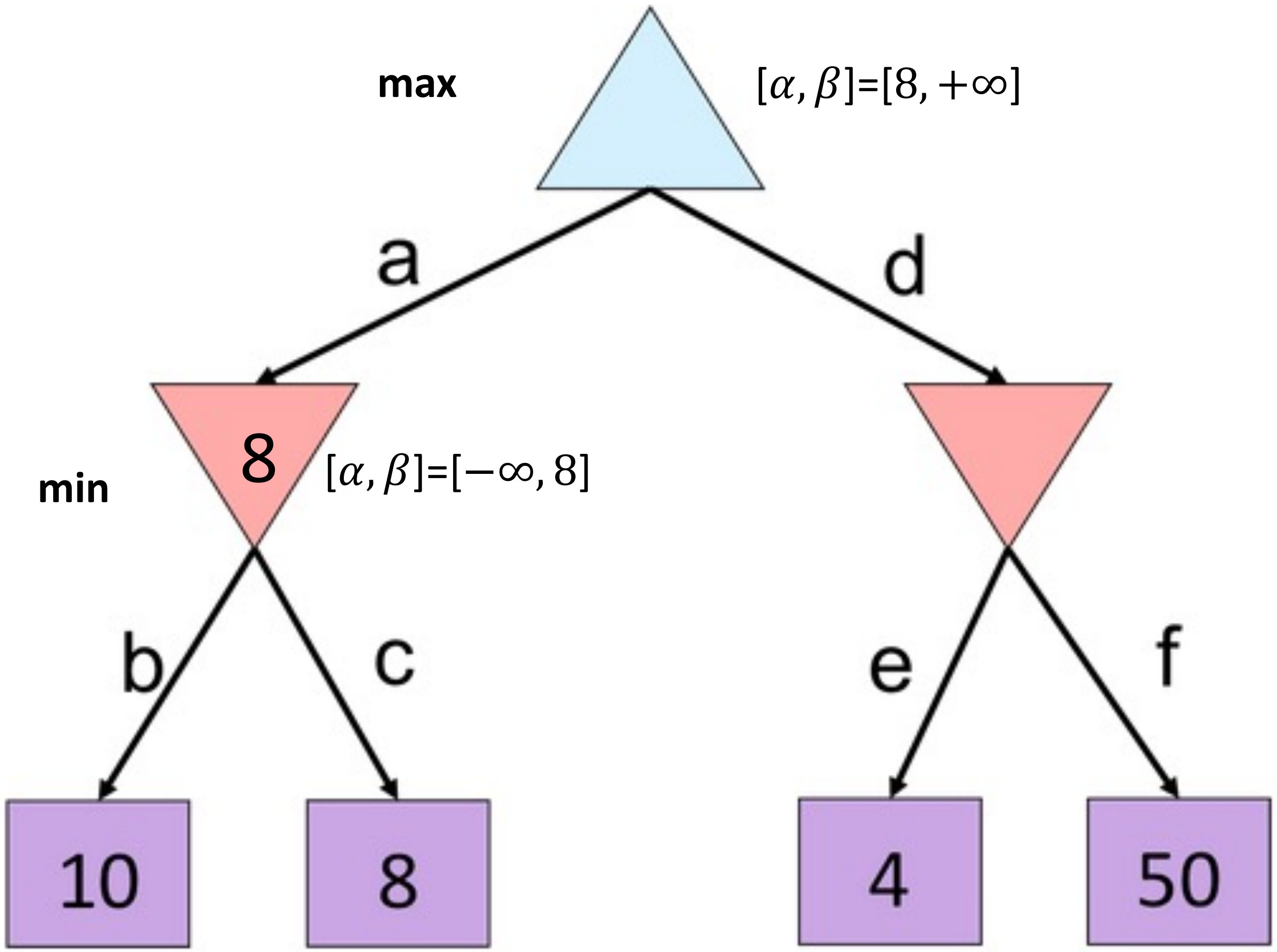
Alpha-Beta Quiz



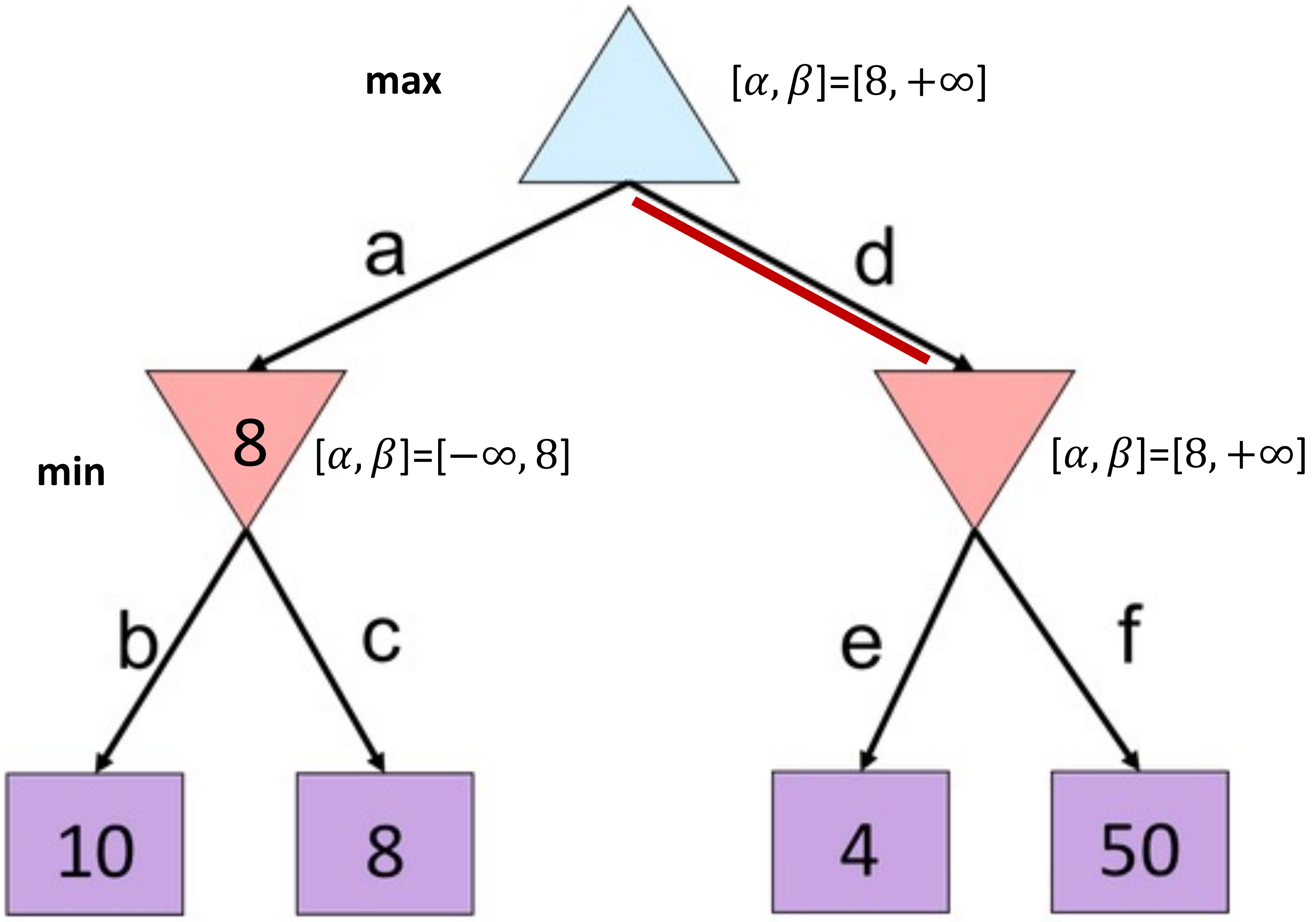
Alpha-Beta Quiz



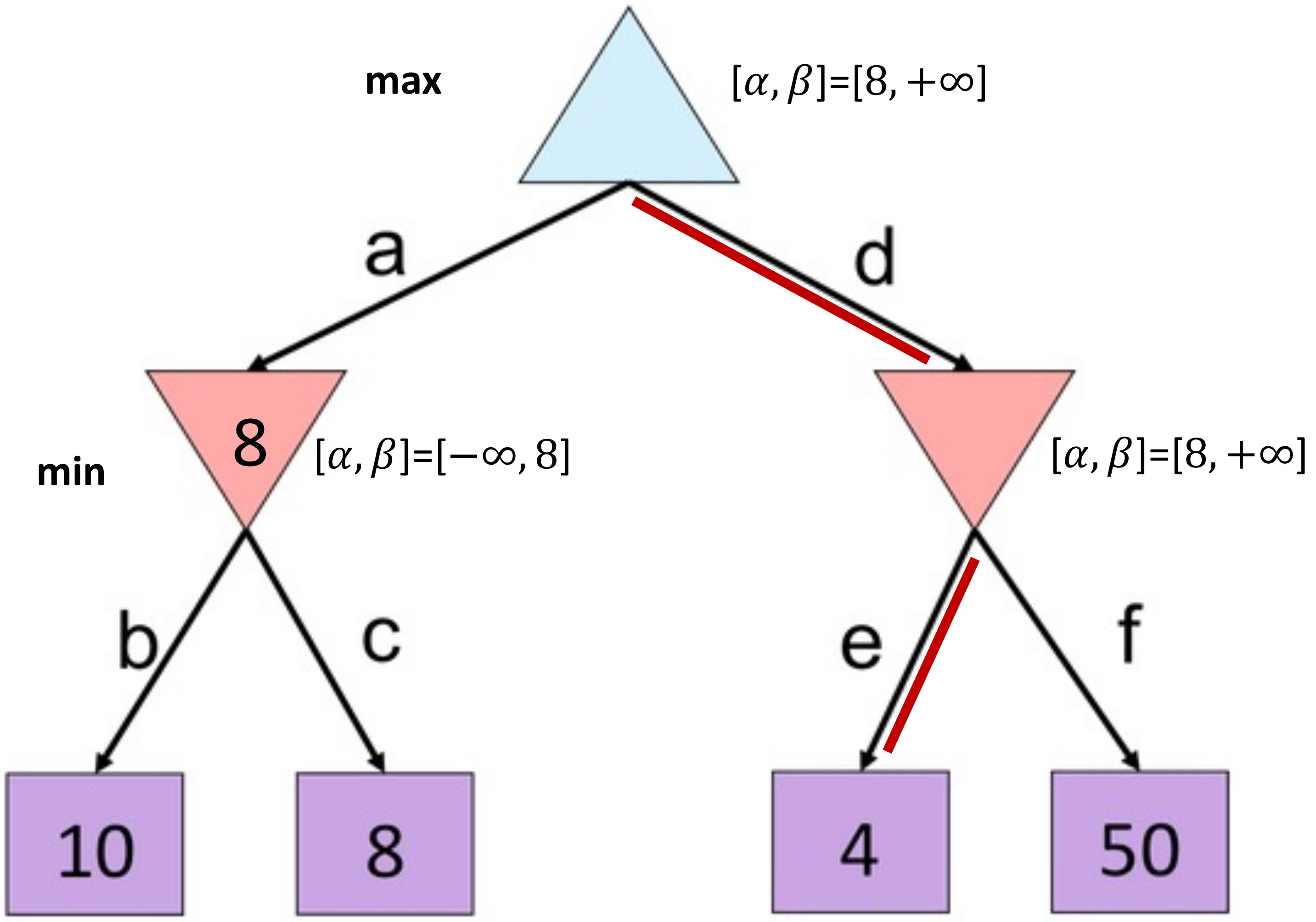
Alpha-Beta Quiz



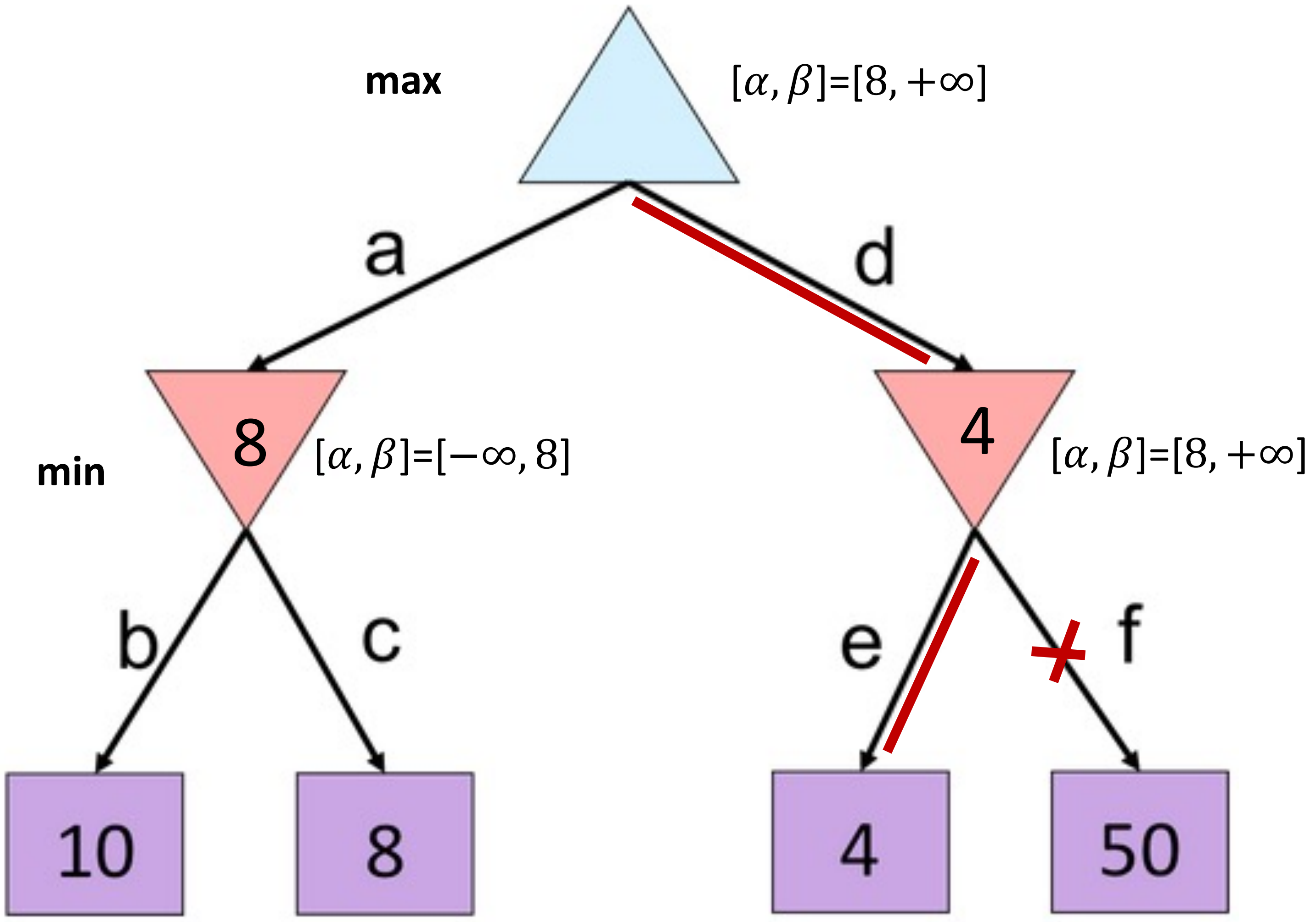
Alpha-Beta Quiz



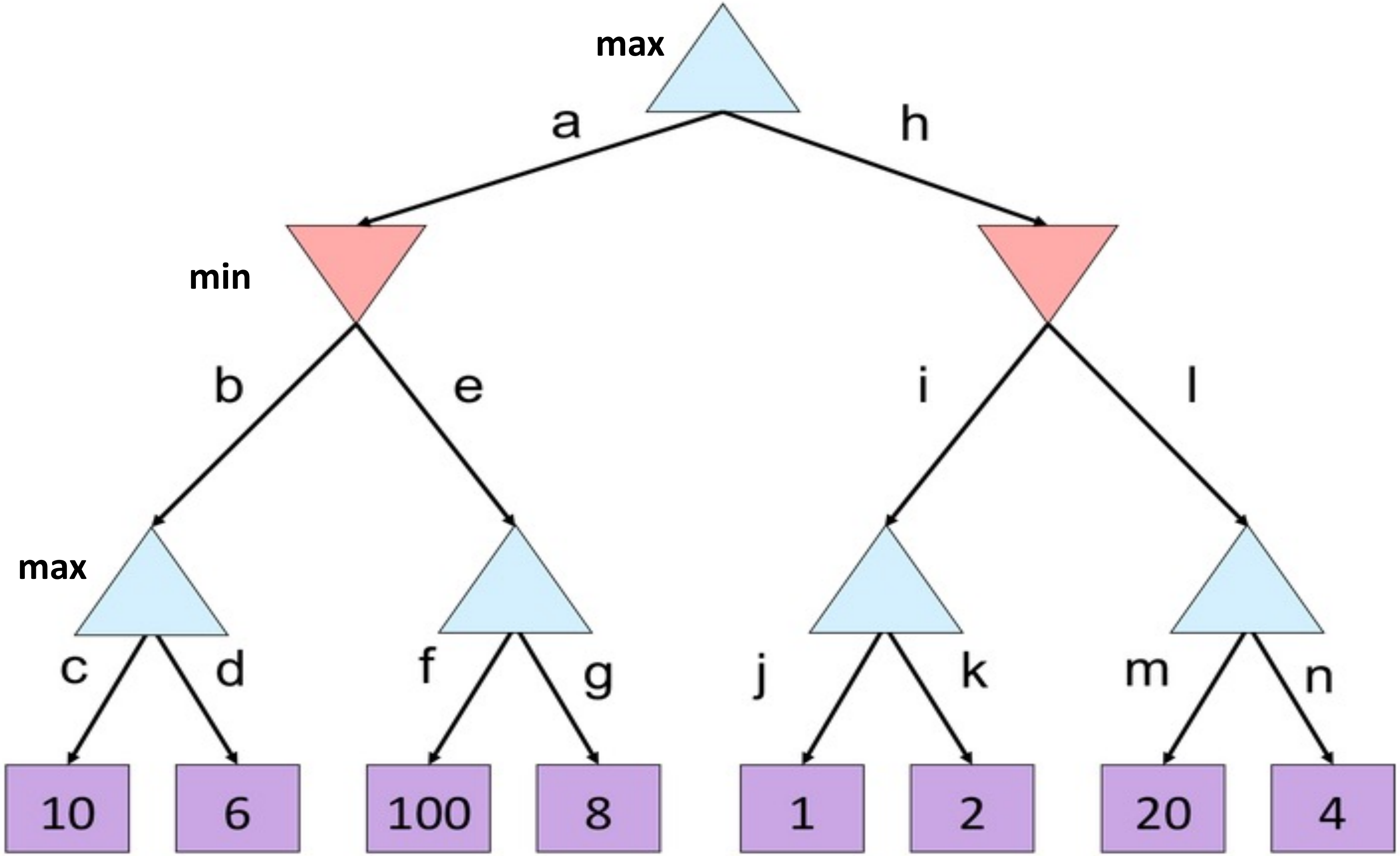
Alpha-Beta Quiz



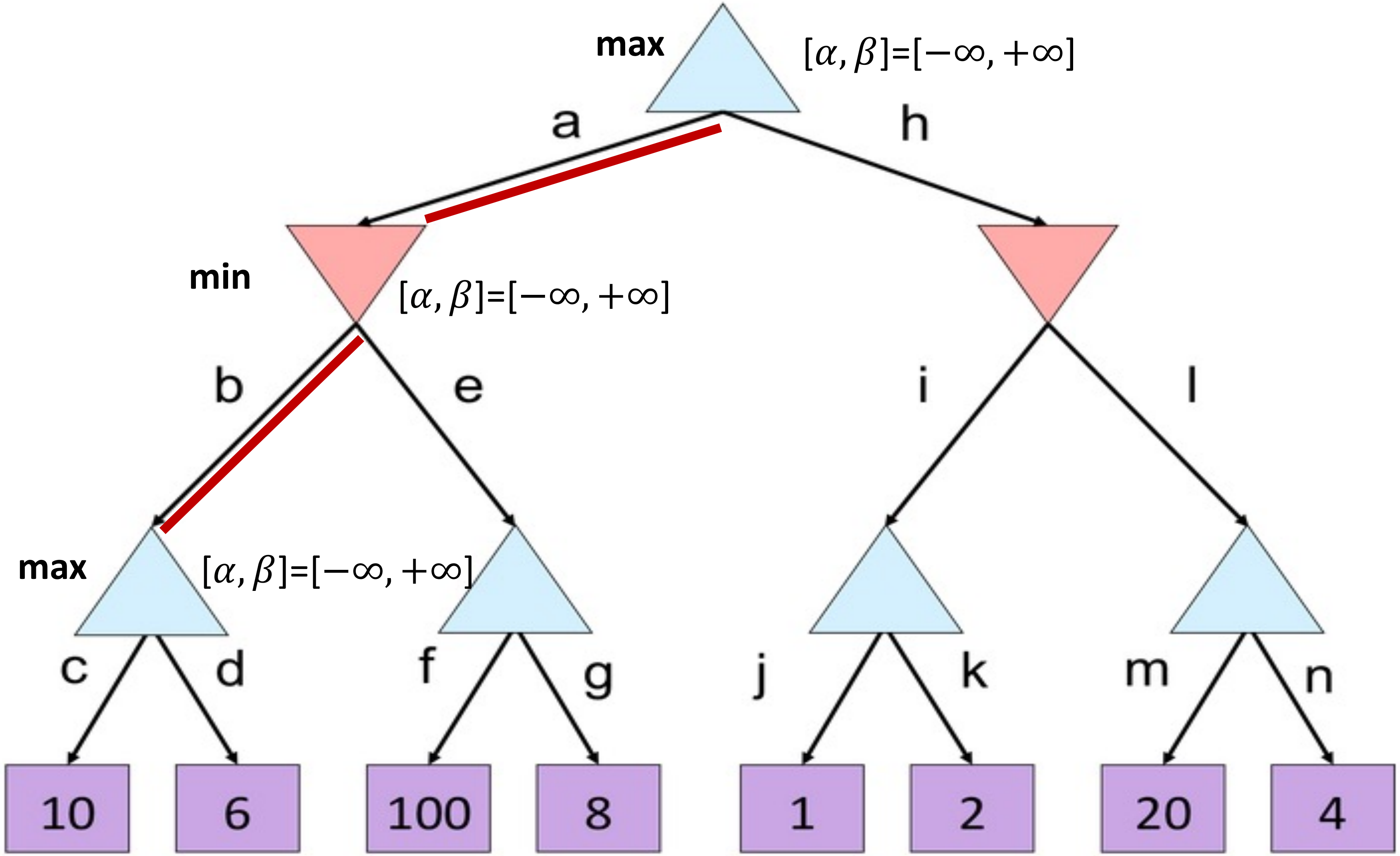
Alpha-Beta Quiz



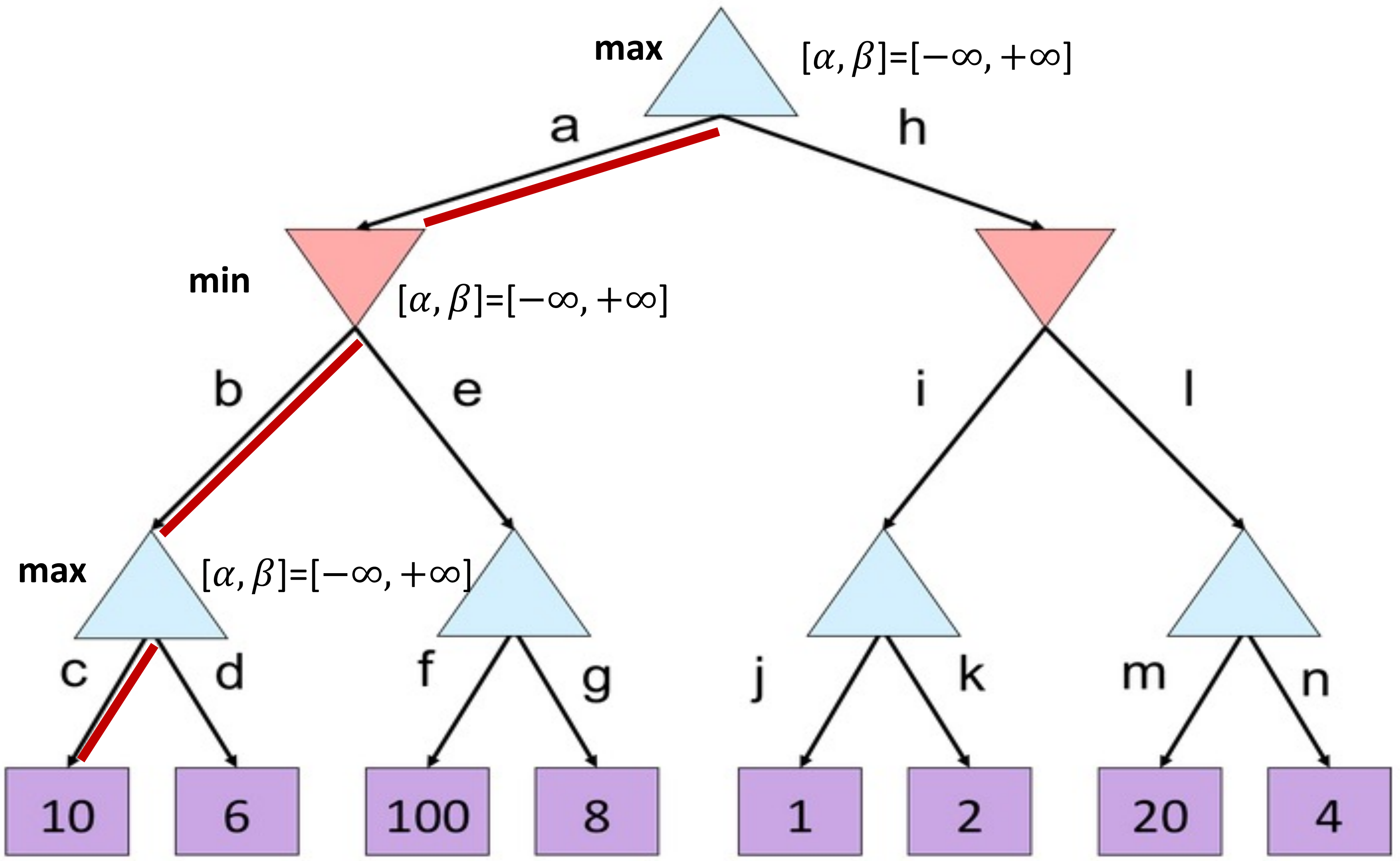
Alpha-Beta Quiz 2



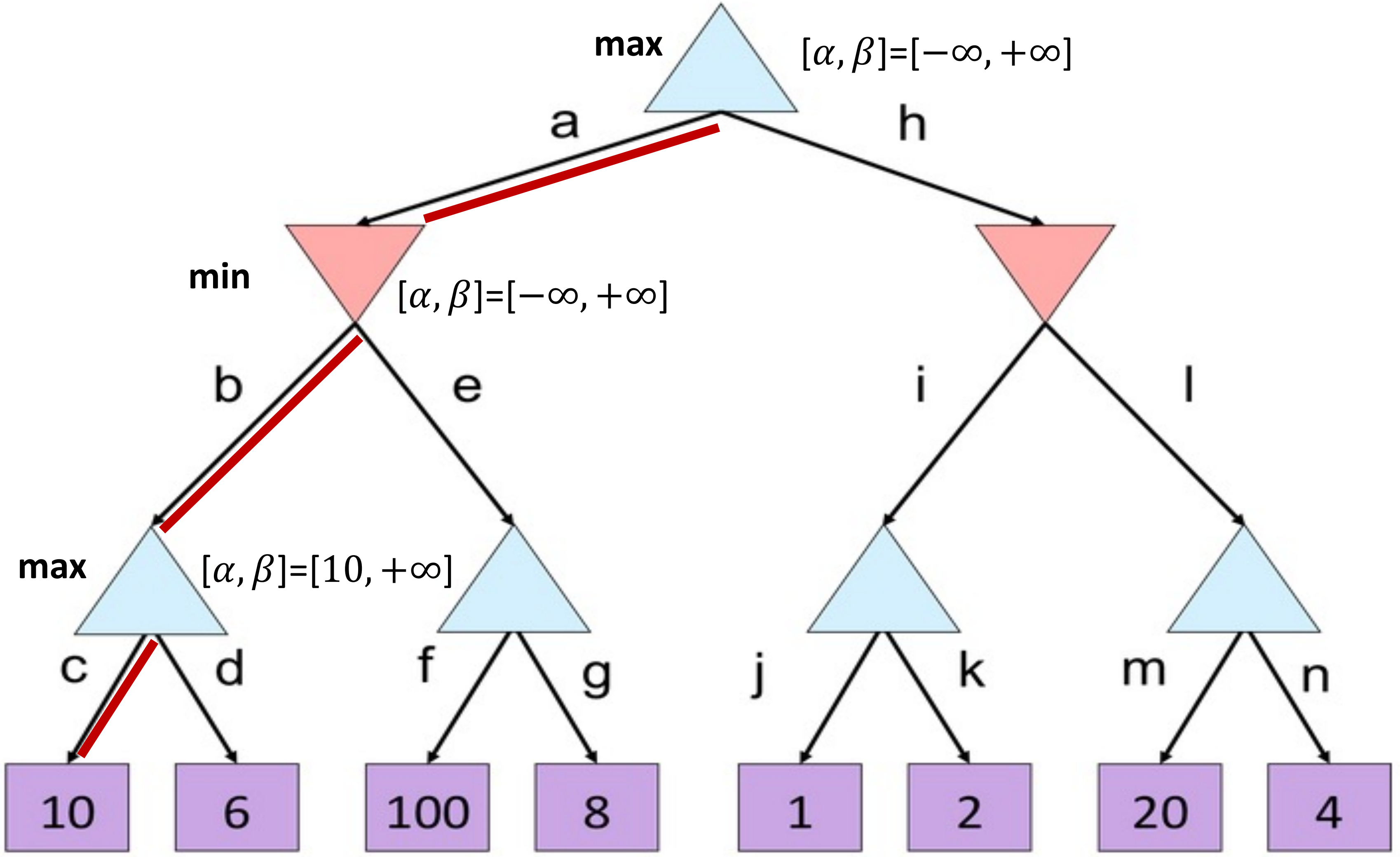
Alpha-Beta Quiz 2



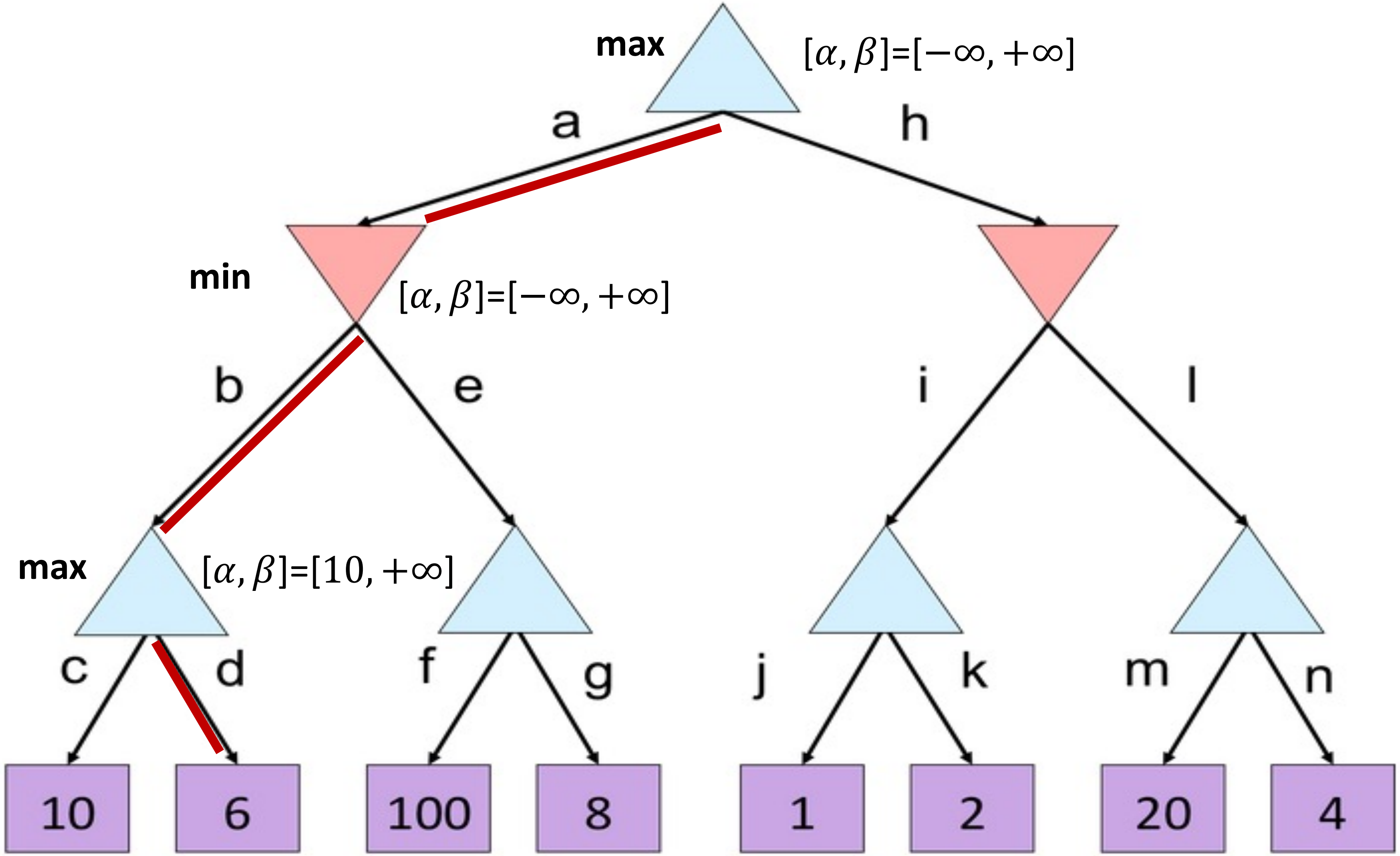
Alpha-Beta Quiz 2



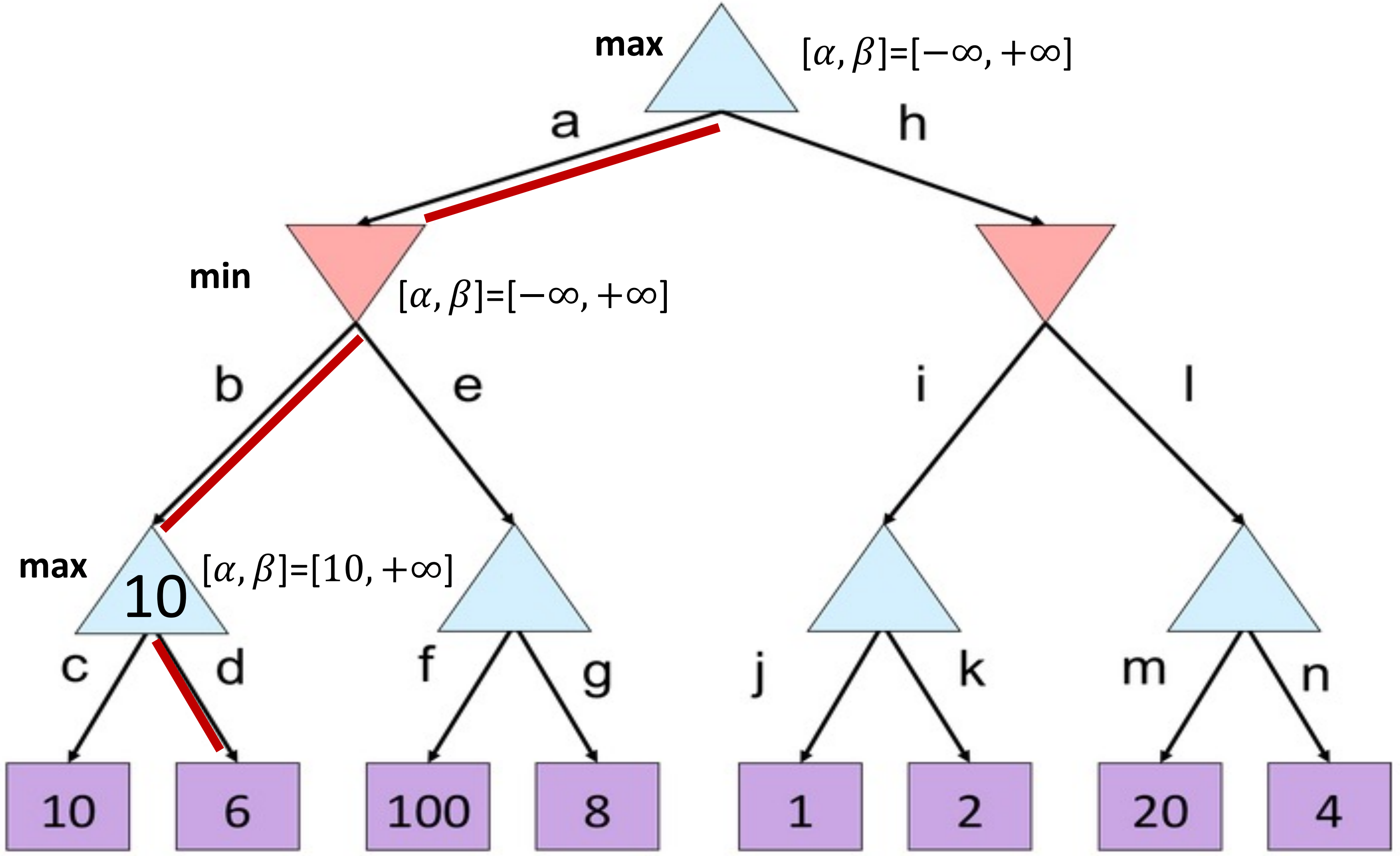
Alpha-Beta Quiz 2



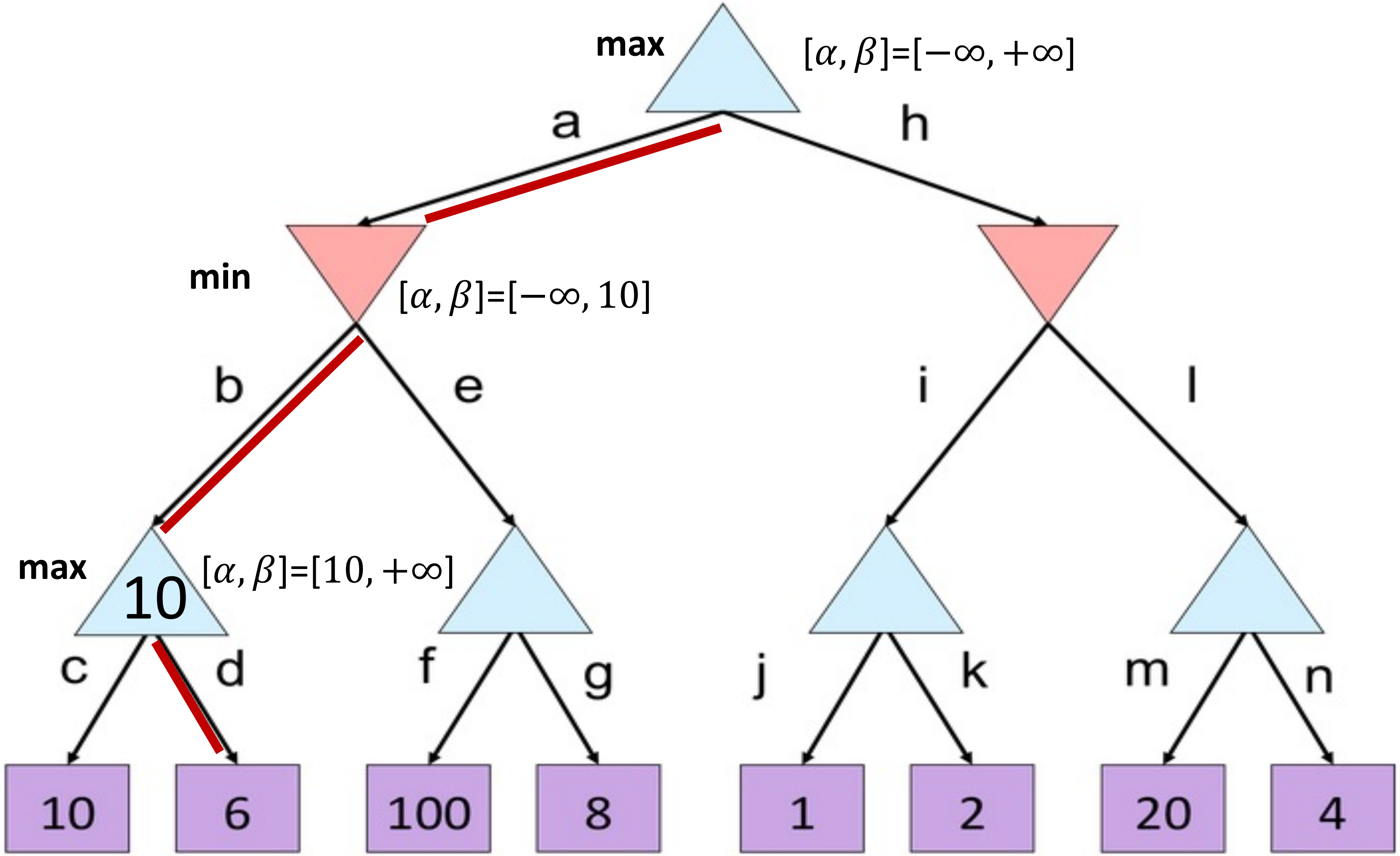
Alpha-Beta Quiz 2



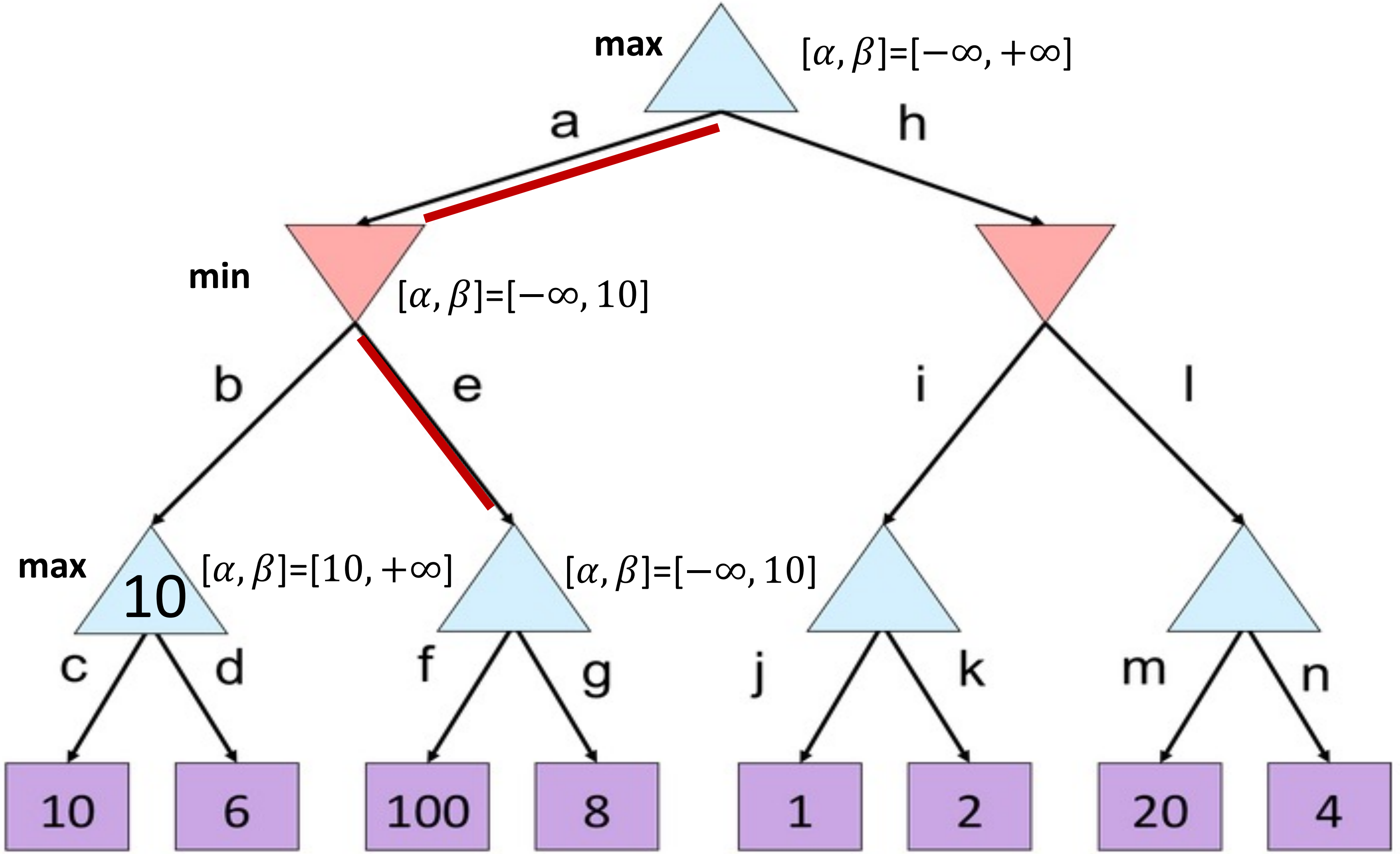
Alpha-Beta Quiz 2



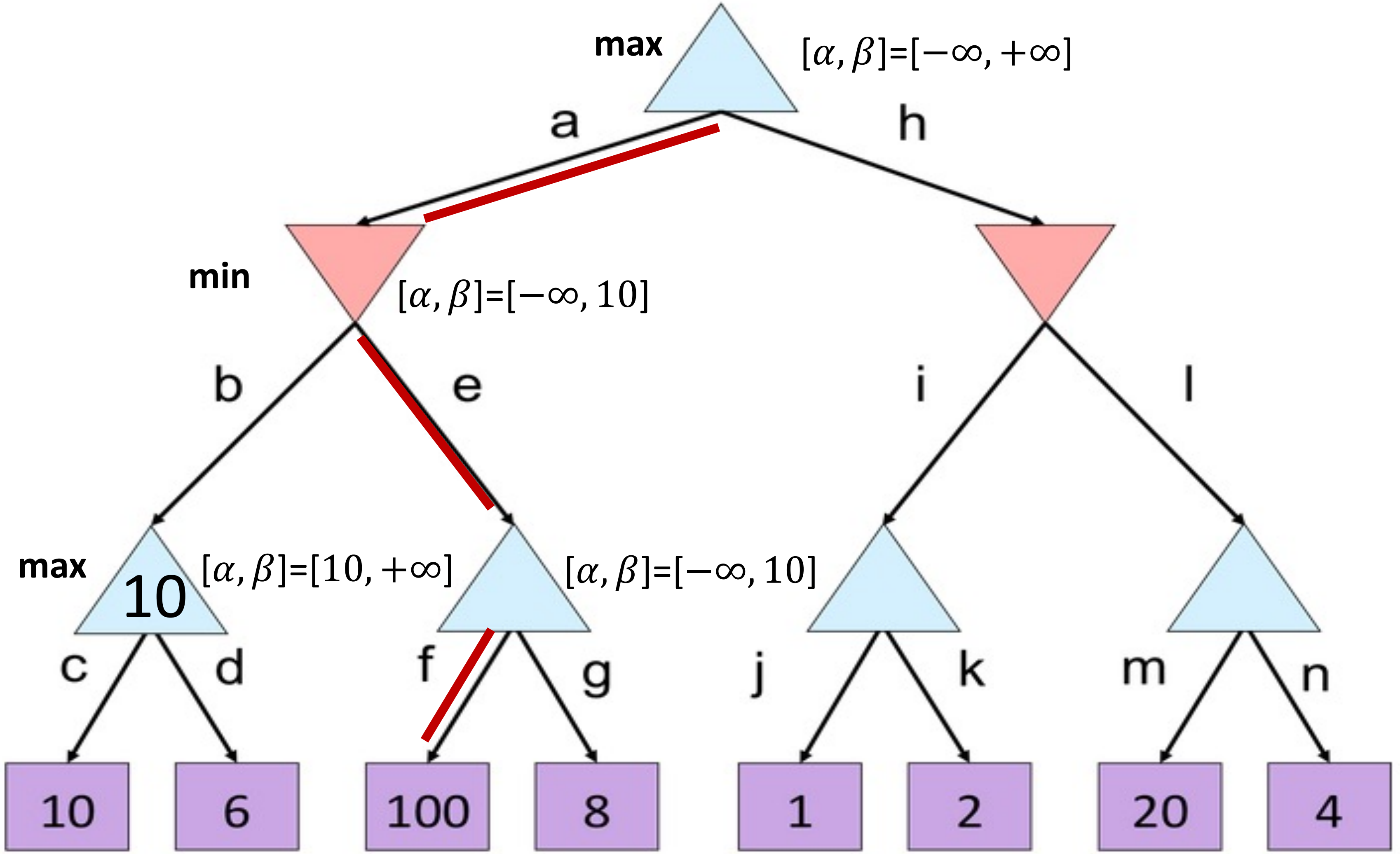
Alpha-Beta Quiz 2



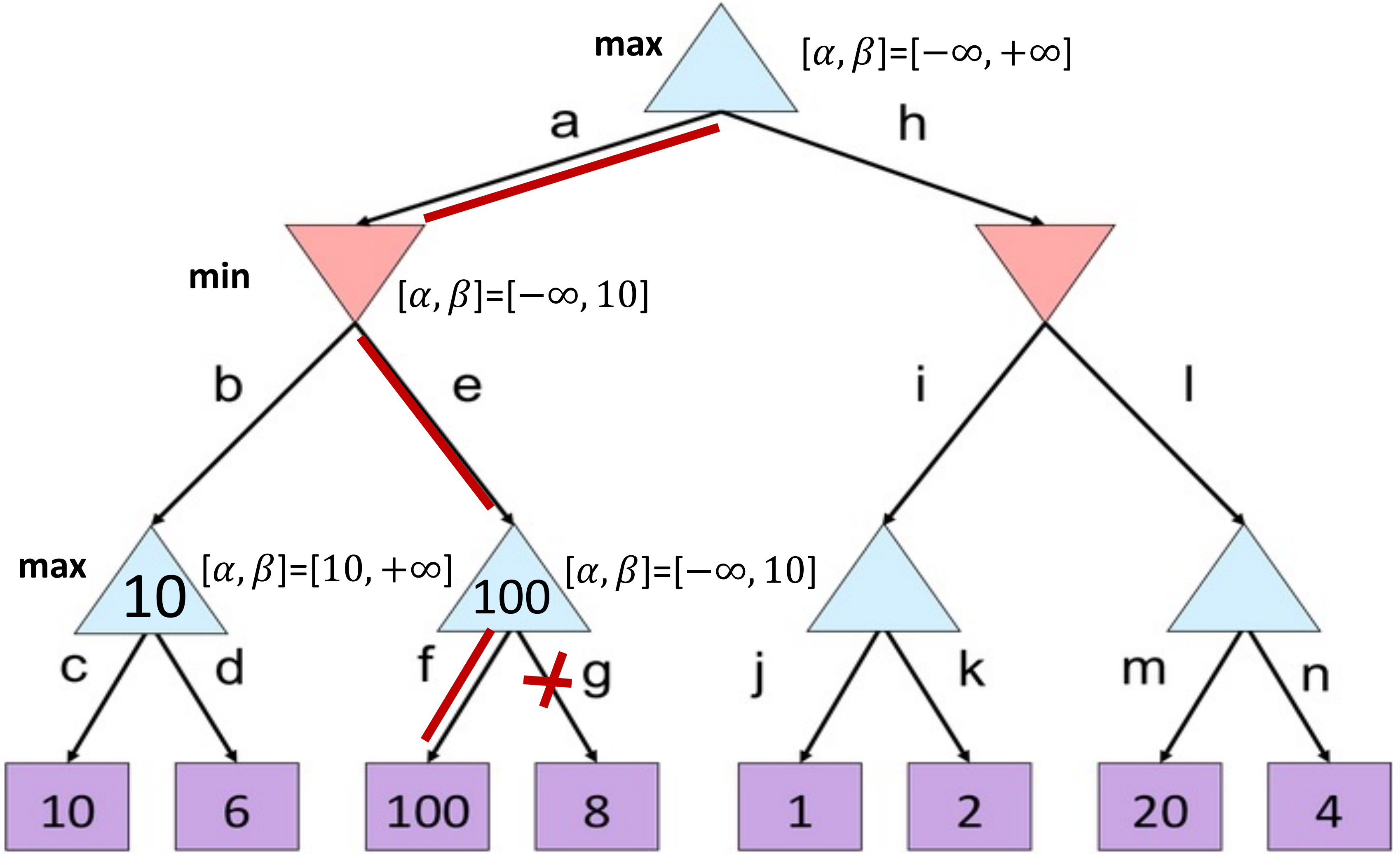
Alpha-Beta Quiz 2



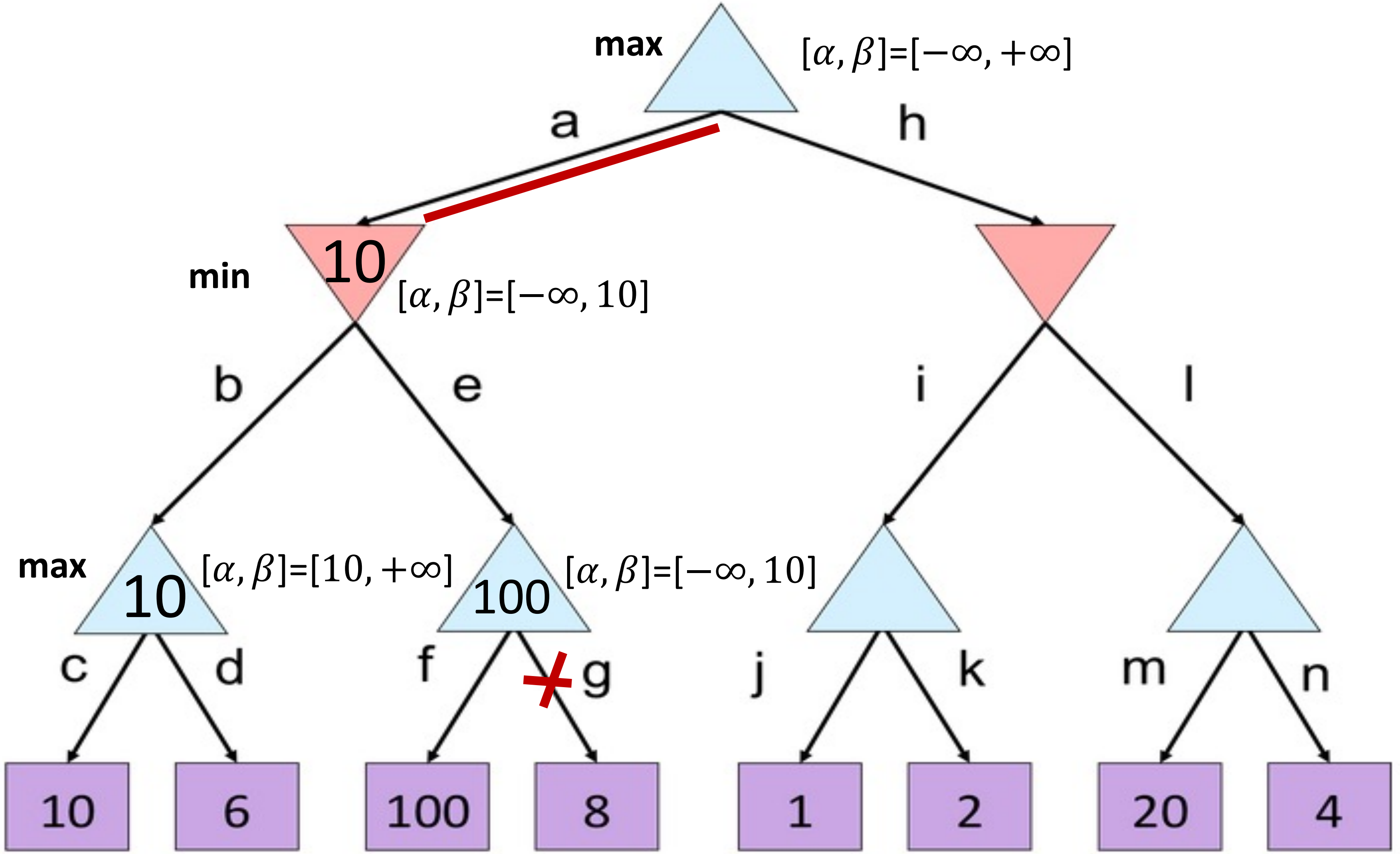
Alpha-Beta Quiz 2



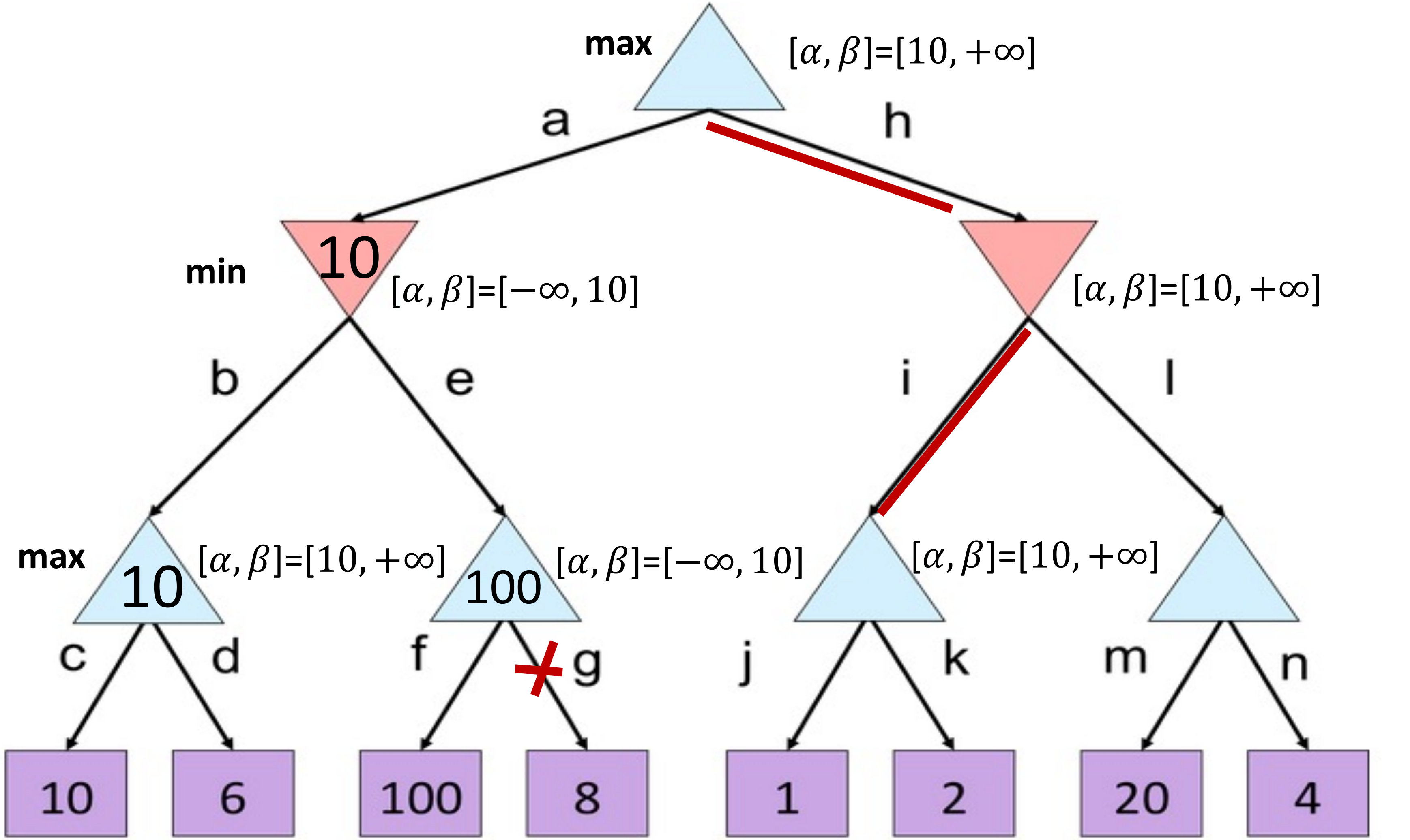
Alpha-Beta Quiz 2



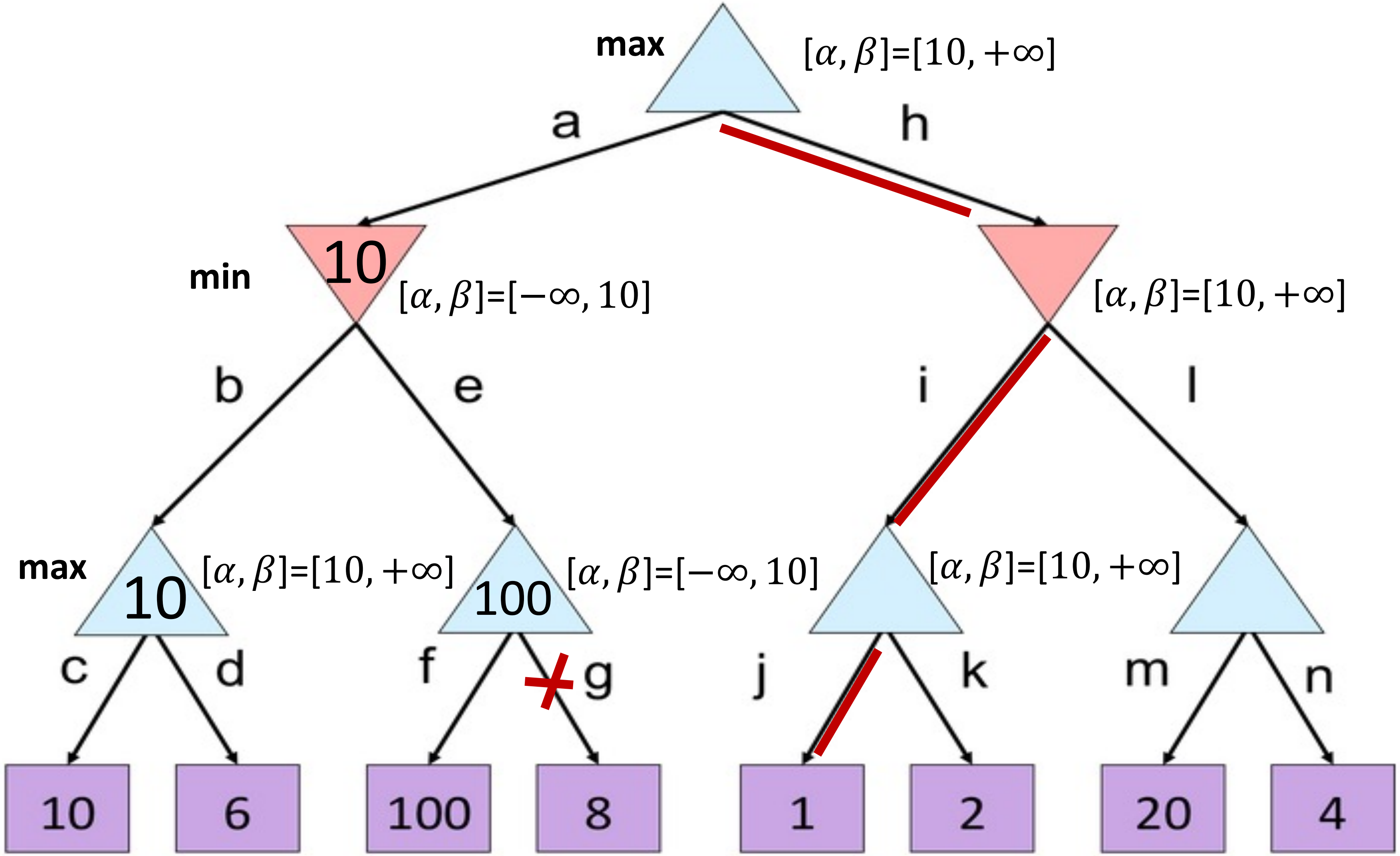
Alpha-Beta Quiz 2



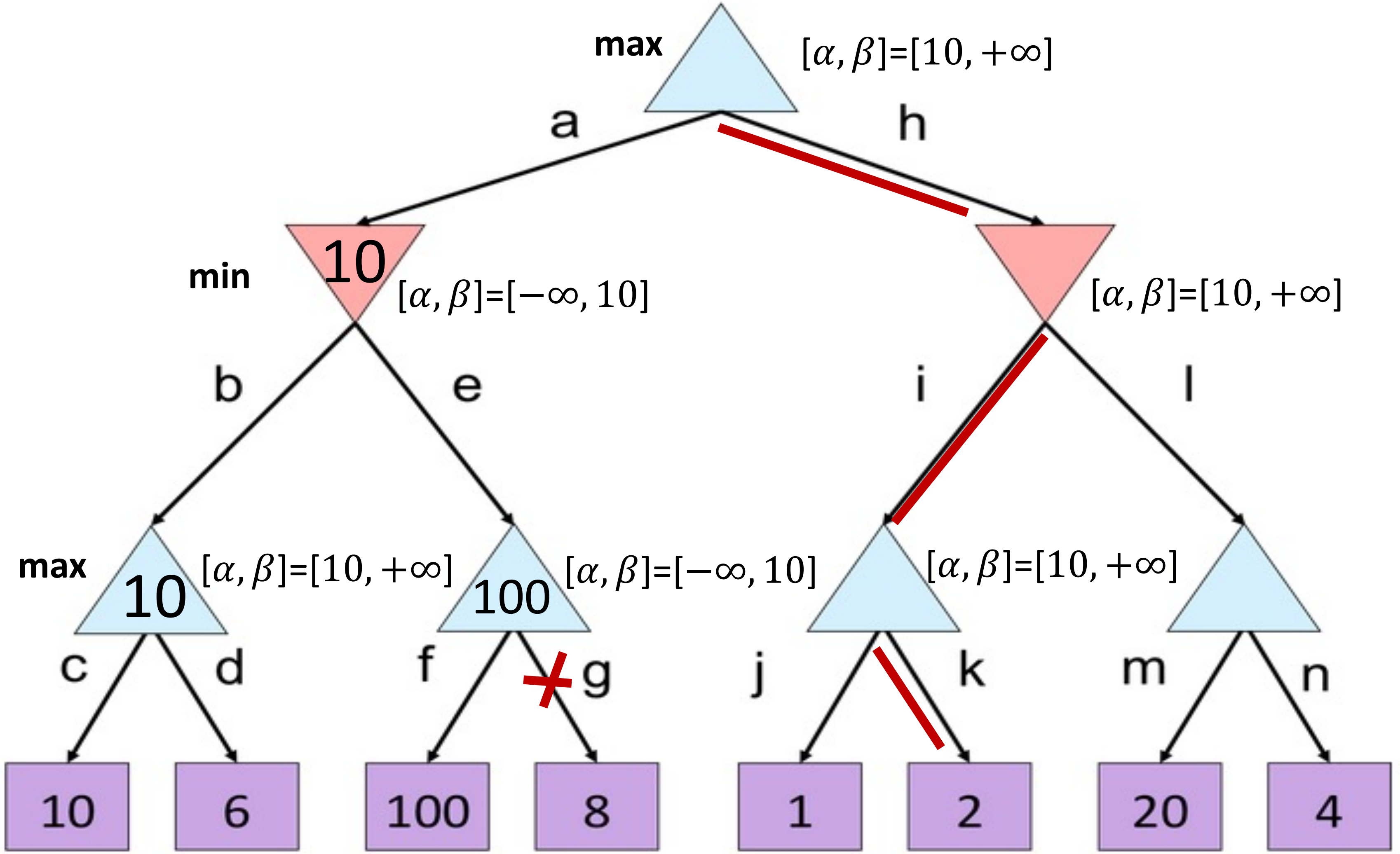
Alpha-Beta Quiz 2



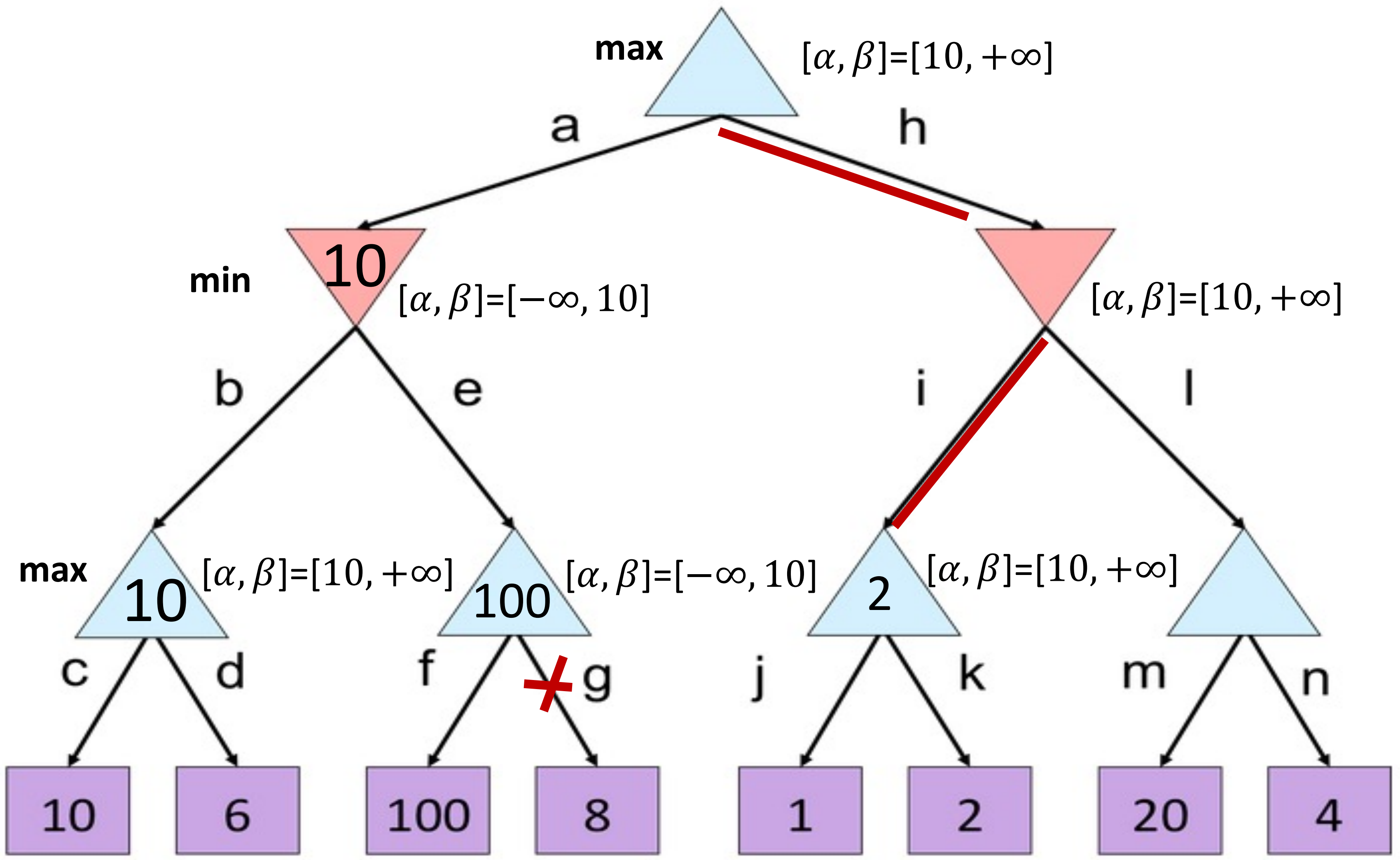
Alpha-Beta Quiz 2



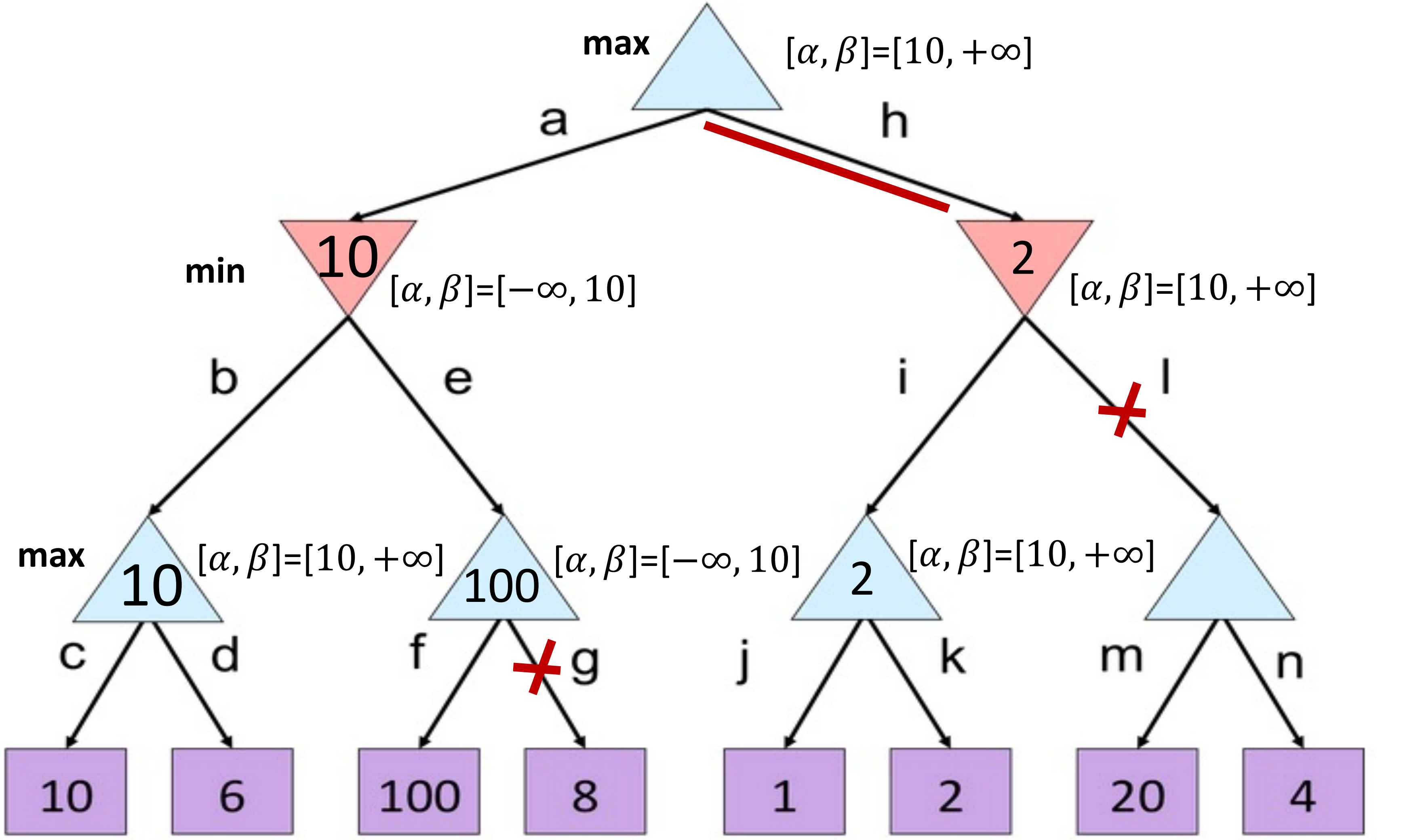
Alpha-Beta Quiz 2



Alpha-Beta Quiz 2



Alpha-Beta Quiz 2



Alpha-Beta Quiz 2