Artificial Intelligence

CSC 665

tyler dae devlin

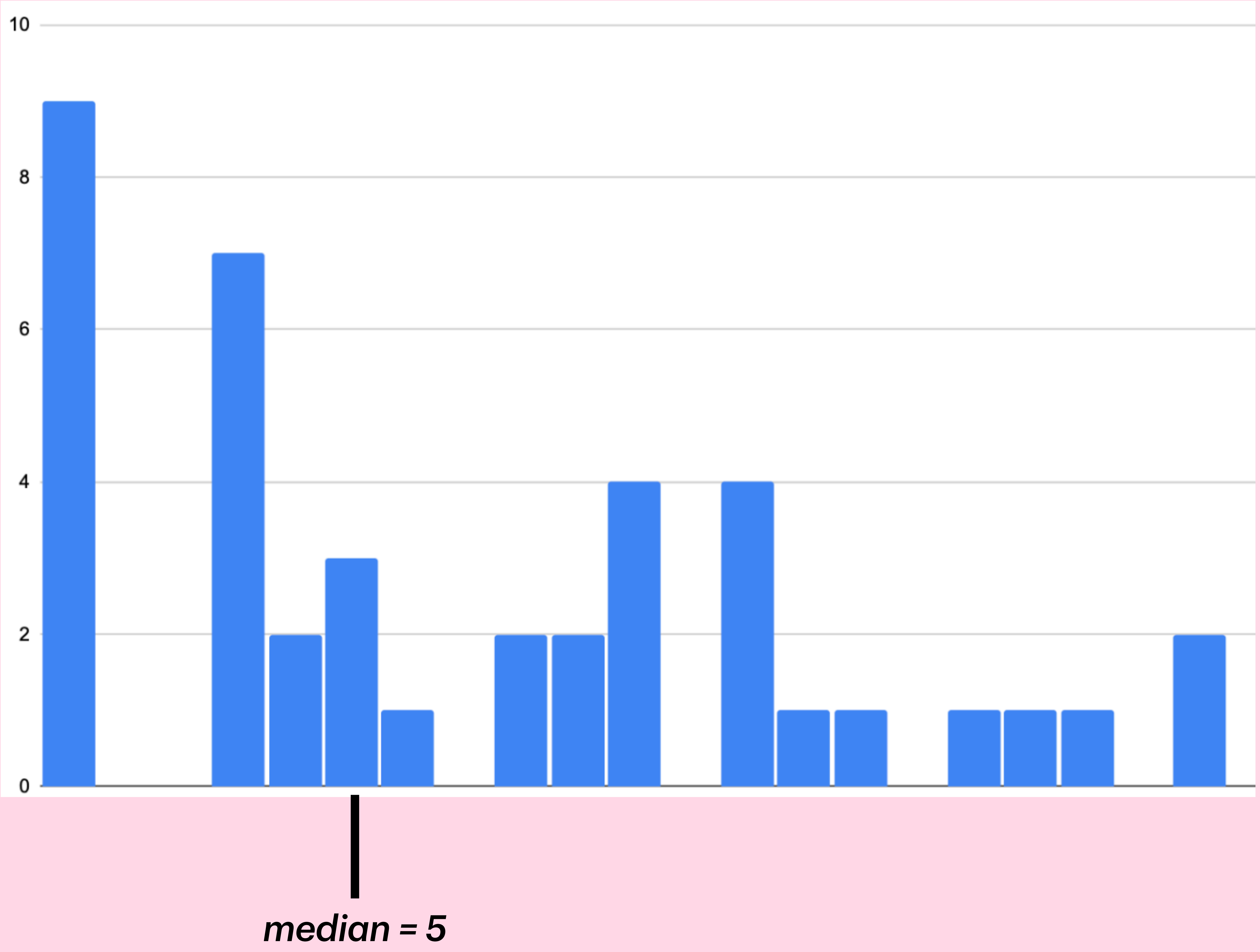
Search IV

9.5.2023

Recap

Administrivia

- Grades posted for homework 0
- **Homework 1** due Monday 9/11
- **Office hours** Tuesdays before class, Thursdays after class
- Campuswire discussion forum for questions (and participation credit)



- **Search:** make decisions by looking ahead
- **Logic:** deduce new facts from existing facts
- **Constraints:** find a way to satisfy a given specification
- **Probability:** reason quantitatively about uncertainty
- **Learning:** make future predictions from past observations

Search

Modeling: start state, actions, costs, transition model, goal test

Inference:

- Uninformed: backtracking, DFS, BFS, UCS
- Informed: greedy search and A* with heuristics via problem relaxation

Adversarial Game-Playing


Connect Four
The Classic Vertical Four-In-A-Row Game
AGES 7 and Up
MB
MILTON BRADLEY

Can we model Connect Four as a search problem?

[modeling attempt on board]

Need to make some changes...

Modeling a game

Start state: $s_0 \in S$

Possible actions: $\text{Actions}(s) \subseteq A$

Transition model: $\text{Succ}(s, a) \in S$

Goal test: $\text{IsEnd}(s) \in \{\text{True}, \text{False}\}$

Agent utility: $\text{Utility}(s) \in \mathbb{R}$

Whose turn: $\text{Player}(s) \in P$

state space S, action set A, player set P, real numbers $\mathbb{R}$

Example: chess

s_0 = starting chess board

Actions(s) = legal chess moves available to Player(s)

Succ(s, a) = board state resulting from taking action a

IsEnd(s) = whether s is a checkmate or stalemate

$$\text{Utility}(s) = \begin{cases} +\infty & \text{if white wins} \\ -\infty & \text{if black wins} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Player}(s) = \begin{cases} \text{white} & \text{if an even number of turns have passed} \\ \text{black} & \text{if an odd number of turns have passed} \end{cases}$$

Two key characteristics of games

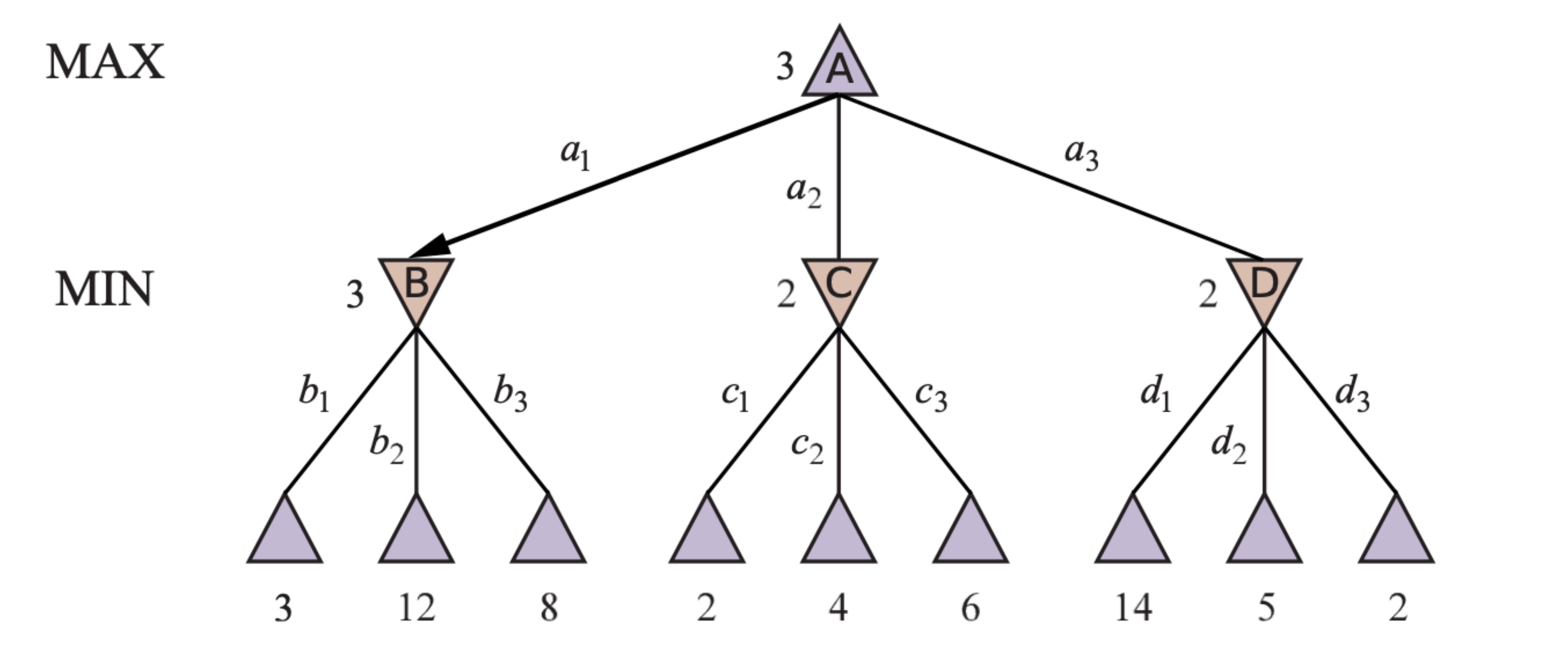
Different players in control at different nodes — one maximizing player and one minimizing player.

All **utility is concentrated at terminal nodes** (i.e. leaves in a tree) — don't know whether a move is good or bad until the game is over.

What should you do?

- Given a game state *s*, what action in Actions(*s*) should you take?
- Depends on who you are — assume you are the maximizing player, *max*
- *max*'s best action depends on what *min* does on the next turn
- But *min*'s best action depends on *max*'s move on the next next turn
- … which depends on *min*'s move on the next next next turn
- And so on …

[minimax game tree on board]


MAX
MIN
A
B
C
D
3
3
2
2
a_1
a_2
a_3
b_1
b_2
b_3
c_1
c_2
c_3
d_1
d_2
d_3
3
12
8
2
4
6
14
5
2

Minimax recurrence

Let $V(s)$ denote the minimax value of the game starting at state s.

(These are the node values from the previous example.)

$$V(s) = \begin{cases} \text{Utility}(s) & \text{if IsEnd}(s) \\ \max_{a \in \text{Actions}(s)} V(\text{Succ}(s, a)) & \text{if Player}(s) = \text{max} \\ \min_{a \in \text{Actions}(s)} V(\text{Succ}(s, a)) & \text{if Player}(s) = \text{min} \end{cases}$$

Expanding the recurrence, gives an expression of the form

$$V(s_0) = \max_{a_0}(\min_{a_1}(\max_{a_2}(\ldots \min_{a_n} \text{Utility}(\text{Succ}(s_n, a_n))\ldots)))$$

Game trees are exponentially large

- **250K** possible tic-tac-toe games
- **288B** possible chess games after just 8 moves
- **10^{29000}** total possible chess games (vs. 10^{80} atoms in universe)

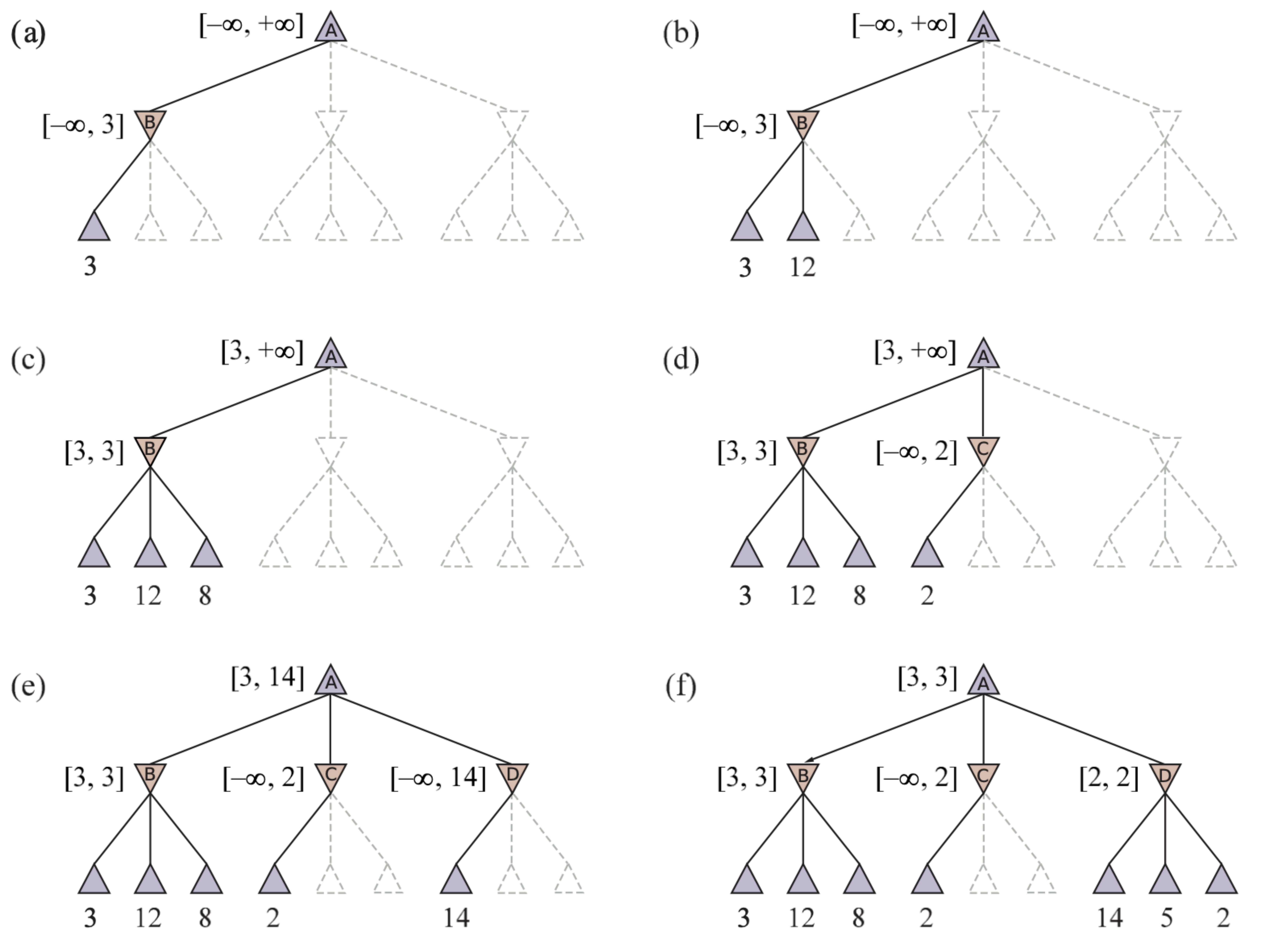
Computing the minimax recurrence down to the leaf nodes is usually **not feasible**.
Need a way to **speed up** decision making.

Two ways to speed up

- **Estimate** $V(s)$ using domain knowledge, which allows you to run a depth-limited search. (Same basic idea as informed search with a heuristic.)
- **Prune** subtrees whose root node value can't possibly be better than a lower bound we've already found.

Won't discuss first approach, but you should know alpha-beta pruning (on the following slides).

[alpha-beta pruning on board]


(a)
[−∞, +∞]
A
[−∞, 3]
B
3
(b)
[−∞, +∞]
A
[−∞, 3]
B
3
12
(c)
[3, +∞]
A
[3, 3]
B
3
12
8
(d)
[3, +∞]
A
[3, 3]
B
[−∞, 2]
C
3
12
8
2
(e)
[3, 14]
A
[3, 3]
B
[−∞, 2]
C
[−∞, 14]
D
3
12
8
2
14
(f)
[3, 3]
A
[3, 3]
B
[−∞, 2]
C
[2, 2]
D
3
12
8
2
14
5
2