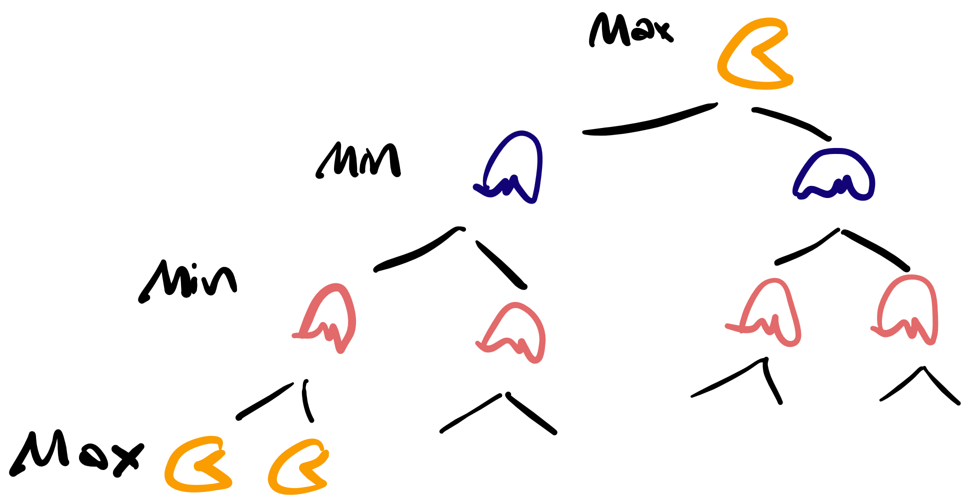
CS 232: Artificial Intelligence

Fall 2024

Prof. Carolyn Anderson
Wellesley College

Reminders

- Shortened class today
- Lepei has help hours on Thursday
- I have help hours Friday from 3:30-4:30 in W422
- Worksheets to practice search algorithms



LOVE DATA WEEK 2024

MULTILINGUAL DATATHON FOR GENERATIVE IMAGE AI

WRITE PROMPTS IN MULTIPLE LANGUAGES TO EXAMINE BIASES IN AI

WITH CS PROFESSOR CAROLYN ANDERSON!

FRIDAY, FEBRUARY 16 , 1-2 PM
SCI H105

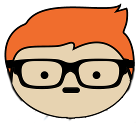
SNACKS PROVIDED!!
SPONSORED BY THE CS DEPARTMENT

Recap

Minimax Summary

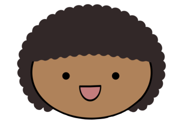
- Rank final game states by their final scores (for tic-tac-toe or chess: win, draw, loss).
- Rank intermediate game states by whose turn it is and the available moves.
  - If it's X's turn, set the rank to that of the *maximum* move available. If a move will result in a win, X should take it.
  - If it's O's turn, set the rank to that of the *minimum* move available. If a move will result in a loss, X should avoid it.

Minimax Values



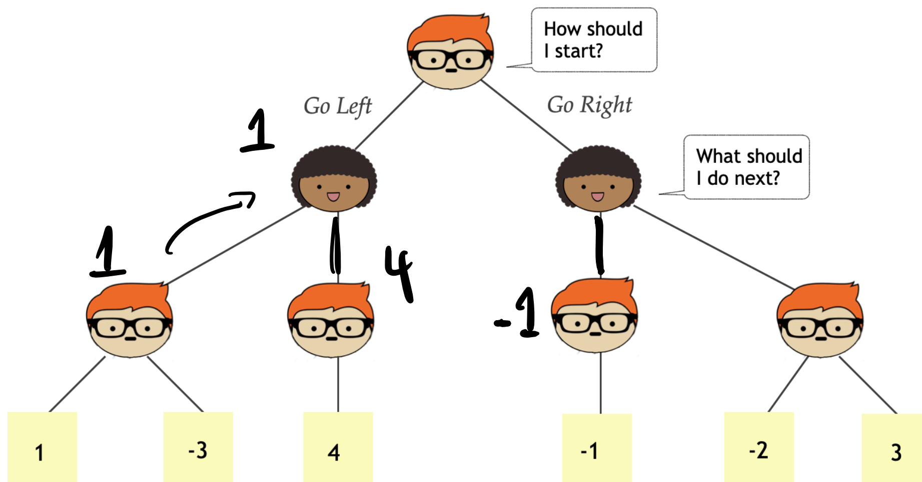
States Under Sam's Control:

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



States Under Thelma's Control:

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



Pruning

Pruning

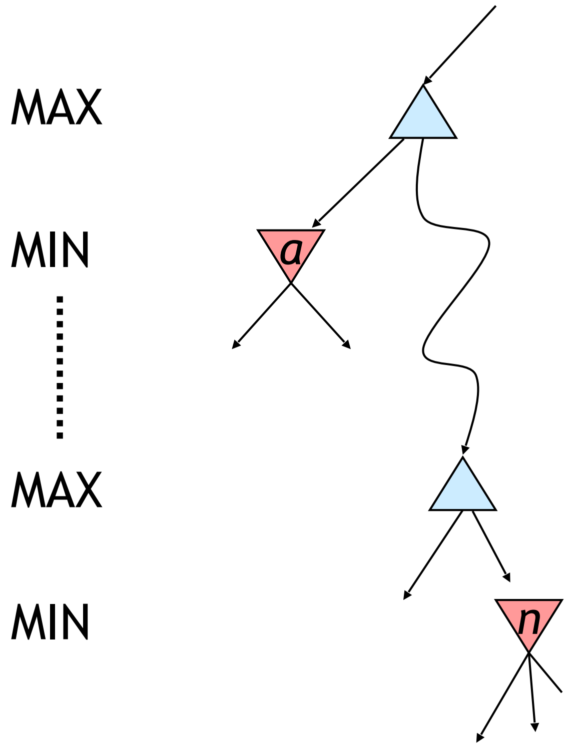
Key idea: give up on paths when you realize that they are worse than options you've already explore.

- Track the **max** score possible for the **minimizing** player (**beta**)
- Track the **min** score possible for the **maximizing** player (**alpha**)

Whenever the **maximum score for beta** becomes less than the **minimum score for alpha**, the maximizing player can stop searching down this path, because it will never be reached.

Alpha-Beta Pruning

- General configuration (MIN version)
  - We're computing the MIN-VALUE at some node *n*
  - We're looping over *n*'s children
  - *n*'s estimate of the childrens' min is dropping
  - Who cares about *n*'s value? MAX
  - Let *a* be the best value that MAX can get at any choice point along the current path from the root
  - If *n* becomes worse than *a*, MAX will avoid it, so we can stop considering *n*'s other children (it's already bad enough that it won't be played)
- MAX version is symmetric



Alpha-Beta Implementation

α: MAX's best option on path to root
β: MIN's best option on path to root

```
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
```

early quitting when α/β cross

```
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
```

early quitting when β < α

Alpha-Beta Pruning Example

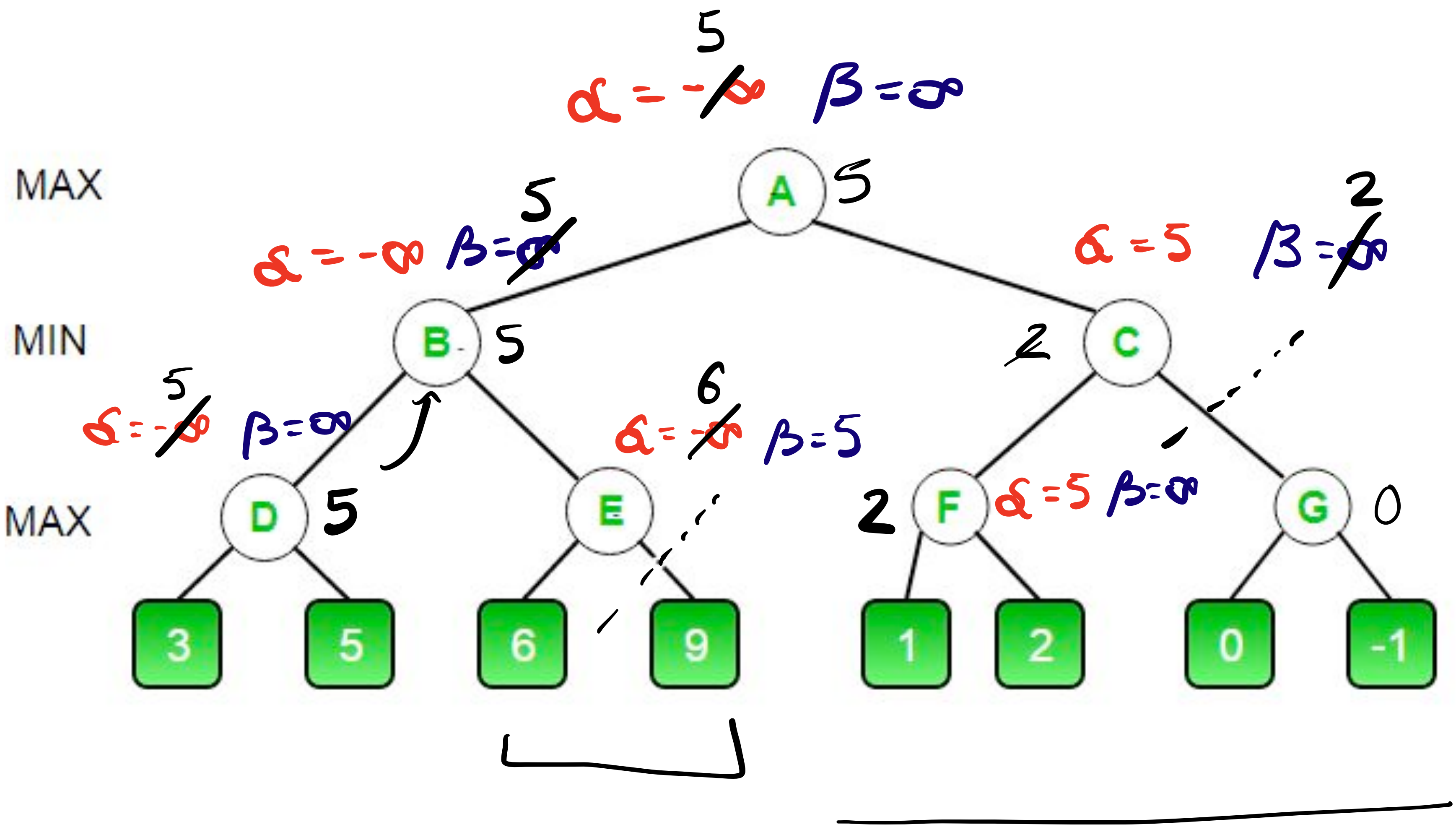


Image from www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning

Alpha-Beta Pruning Properties

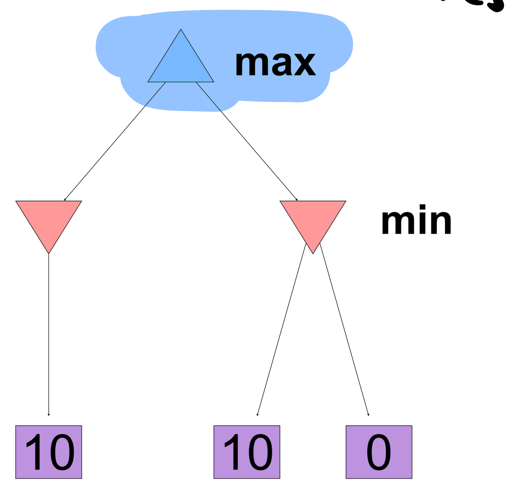
- This pruning has no effect on minimax value computed for the root!

- Values of intermediate nodes might be wrong i.e. they are bounds not exact values
  - Important: children of the root may have the wrong value
  - So the most naïve version won't let you do action selection

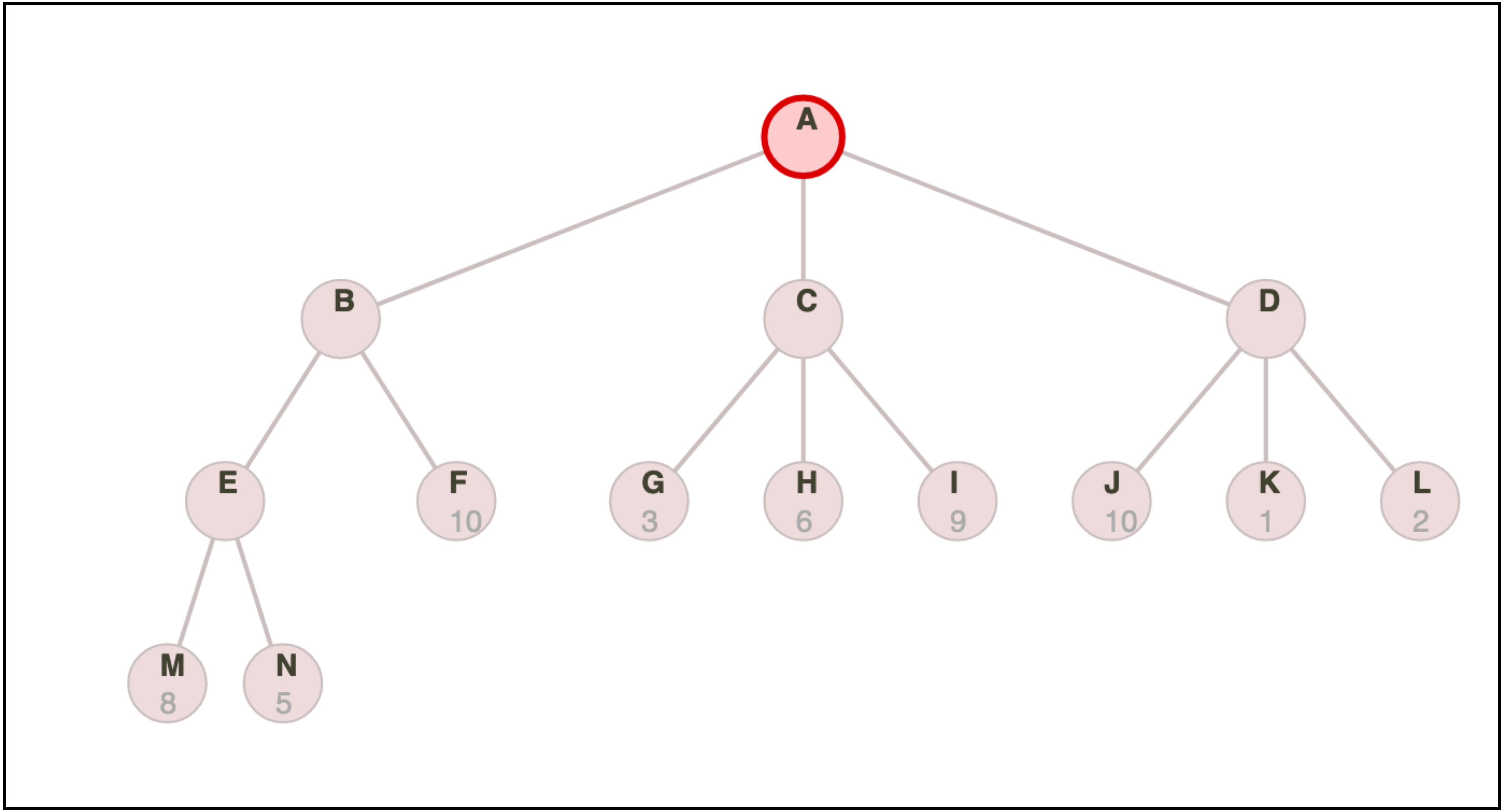
- Good child ordering improves effectiveness of pruning

- With "perfect ordering":
  - Time complexity drops to $O(b^{m/2})$
  - Doubles solvable depth!
  - Full search of, e.g. chess, is still hopeless…

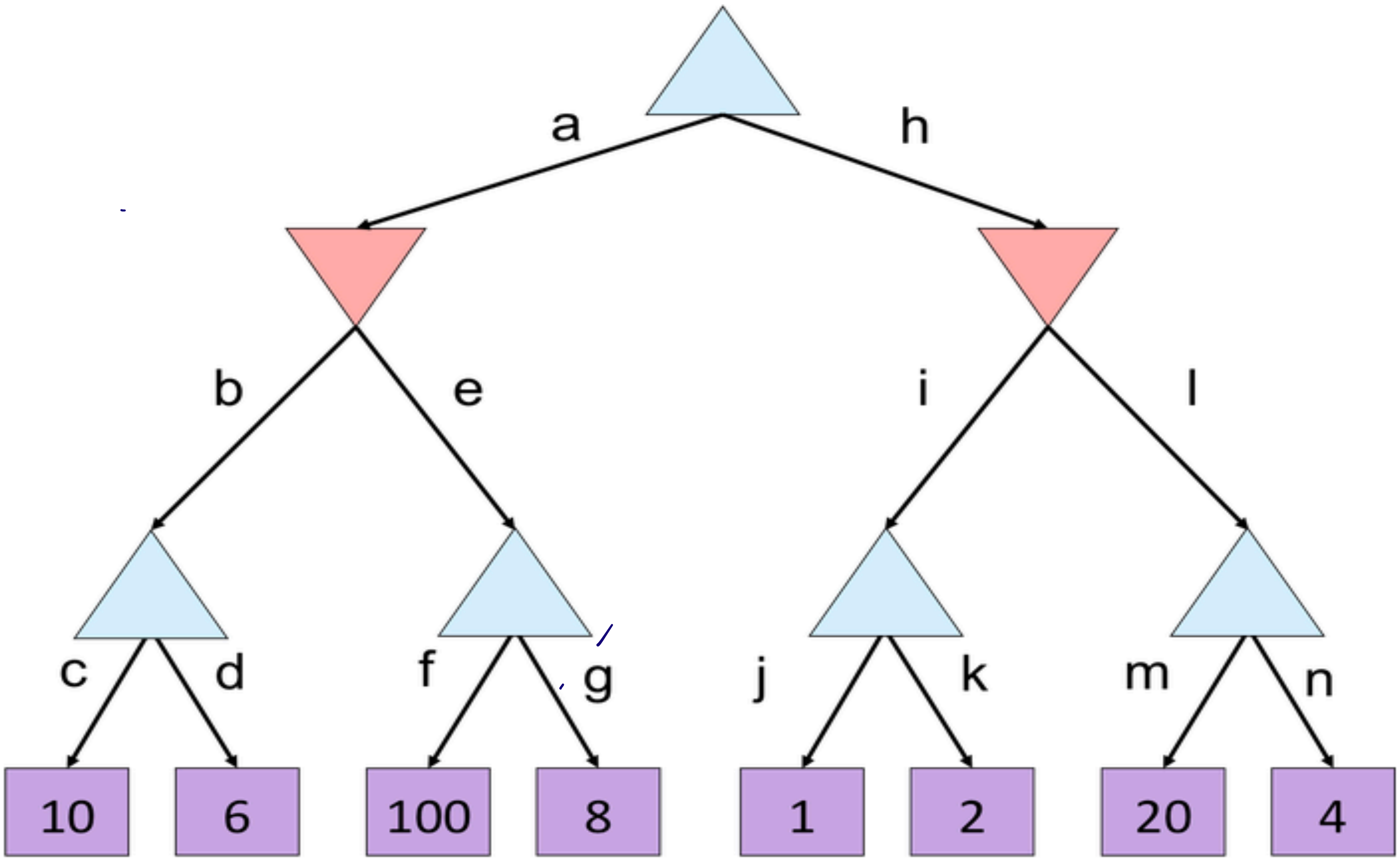
- This is a simple example of metareasoning (computing about what to compute)



Another Demo



Alpha-Beta Quiz



Resource Limits

- Problem: In realistic games, cannot search to leaves!
- Solution: Depth-limited search
  - Instead, search only to a limited depth in the tree
  - Replace terminal utilities with an evaluation function for non-terminal positions
- Example:
  - Suppose we have 100 seconds, can explore 10K nodes / sec
  - So can check 1M nodes per move
  - α-β reaches about depth 8 – decent chess program
- Guarantee of optimal play is gone
- More plies makes a BIG difference
- Use iterative deepening for an anytime algorithm

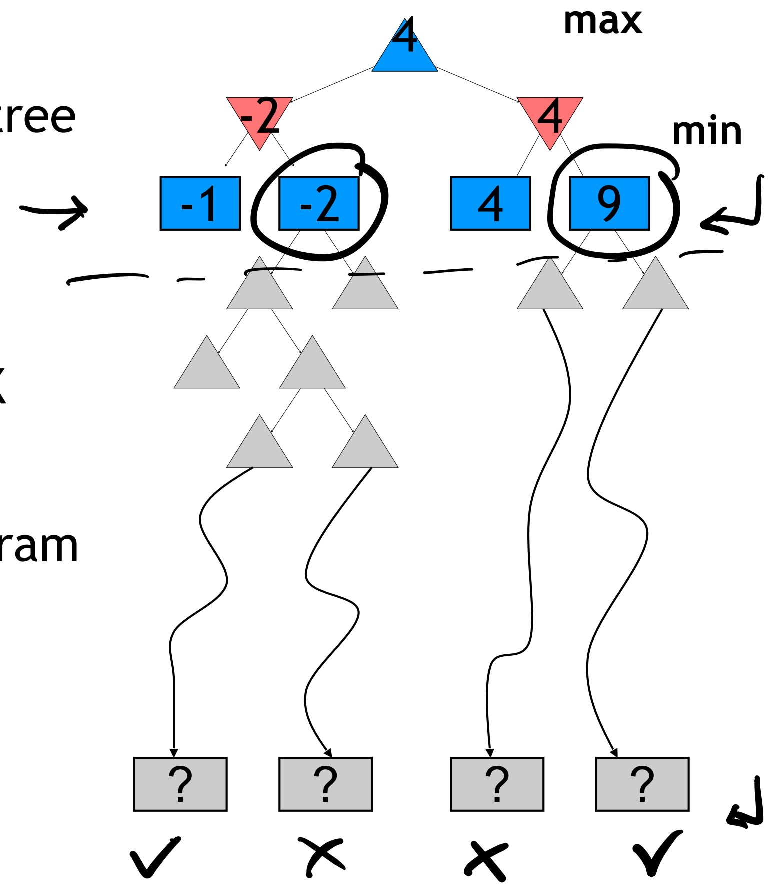
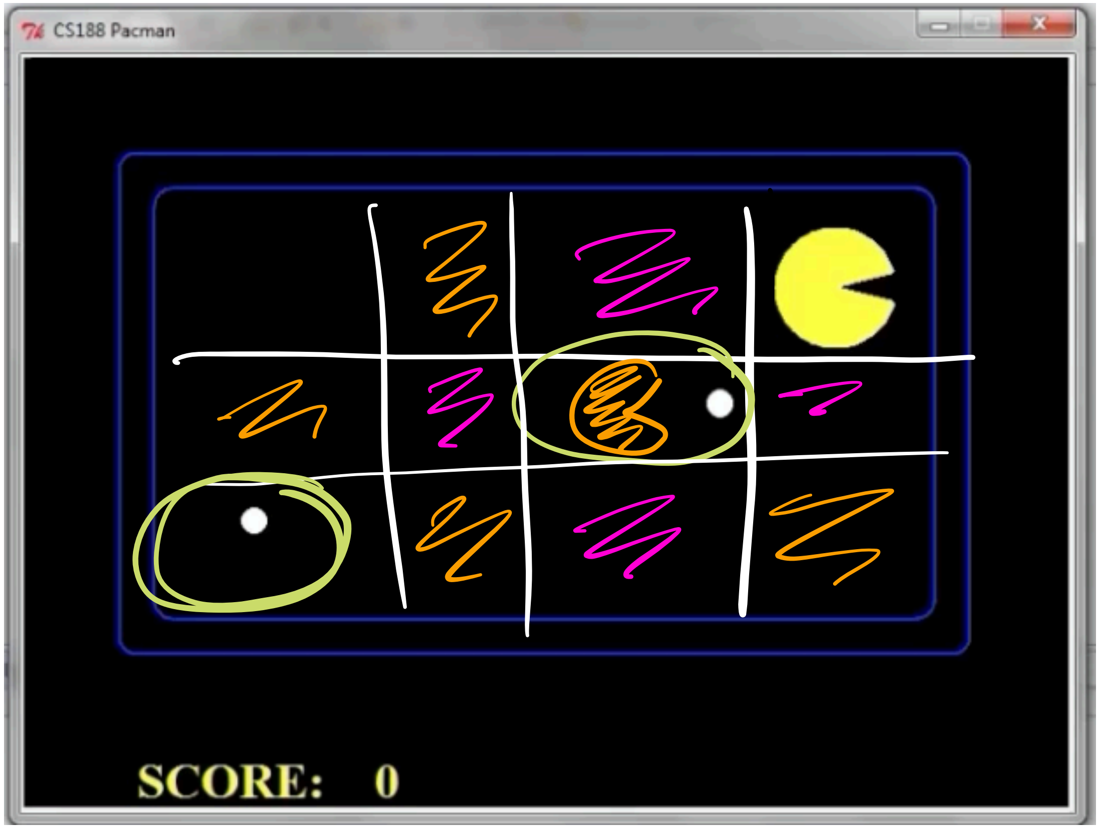


Image from CS188 Intro to AI at UC Berkeley

Resource Limits



Ask a friend:
If we set the depth limit to 2, what will Pacman do?

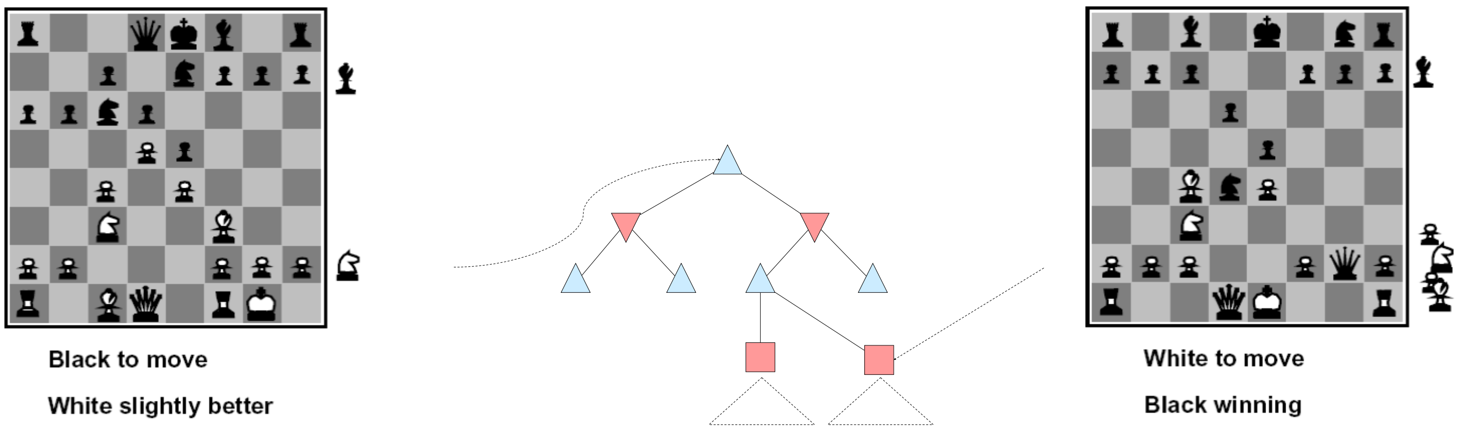
Why Pacman Starves

- A danger of replanning agents!
  - He knows his score will go up by eating the dot now (west, east)
  - He knows his score will go up just as much by eating the dot later (east, west)
  - There are no point-scoring opportunities after eating the dot (within the horizon, two here)
  - Therefore, waiting seems just as good as eating: he may go east, then back west in the next round of replanning!

Evaluation Functions

Evaluation Functions

- Evaluation functions score non-terminals in depth-limited search



- Ideal function: returns the actual minimax value of the position
- In practice: typically weighted linear sum of features:
- e.g. $f_1(s)$ = (num white queens – num black queens), etc.

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

Images from CS188 Intro to AI at UC Berkeley

Evaluation for Pacman

Talk to a friend: what would be a good evaluation function for Pacman?

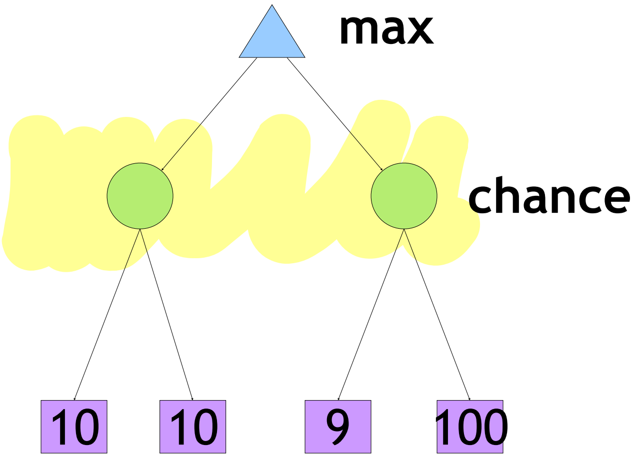
Expectimax

Expectimax Search

In life, outcomes are often **uncertain**. It can be hard to predict exactly what will happen when we take actions.

- Explicit randomness: rolling dice
- Unpredictable opponents
- Actions can fail: robot might slip while navigating

We can model this using Chance nodes!



Expectimax Search

Like Minimax, but now there are chance nodes. For chance nodes, we take an **average** of their outcomes (children) **weighted** by the probabilities of each path.