CS 188 Discussion 4:

Games

Kenny Wang (kwkw@berkeley.edu)
Wed Sep 20, 2023

Slides inspired by Sashrika Pandey and Regina Wang

Administrivia

- Project 2 due on Friday, Sep 22
- Homework is due on Tuesdays
- We have office hours pretty much all day every weekday (12-7), come to Soda 341B!
- Reminder: Need extensions? We will give you extensions!

Today’s Topics

- Games!
  - Minimax
  - Alpha-beta pruning
  - Expectimax

Minimax

- Zero-sum game where opponent plays optimally
- Agents and opponents take turns
  - Maximizing agents/nodes try to **maximize** utility
  - Minimizer agents/nodes try to **minimize** utility

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

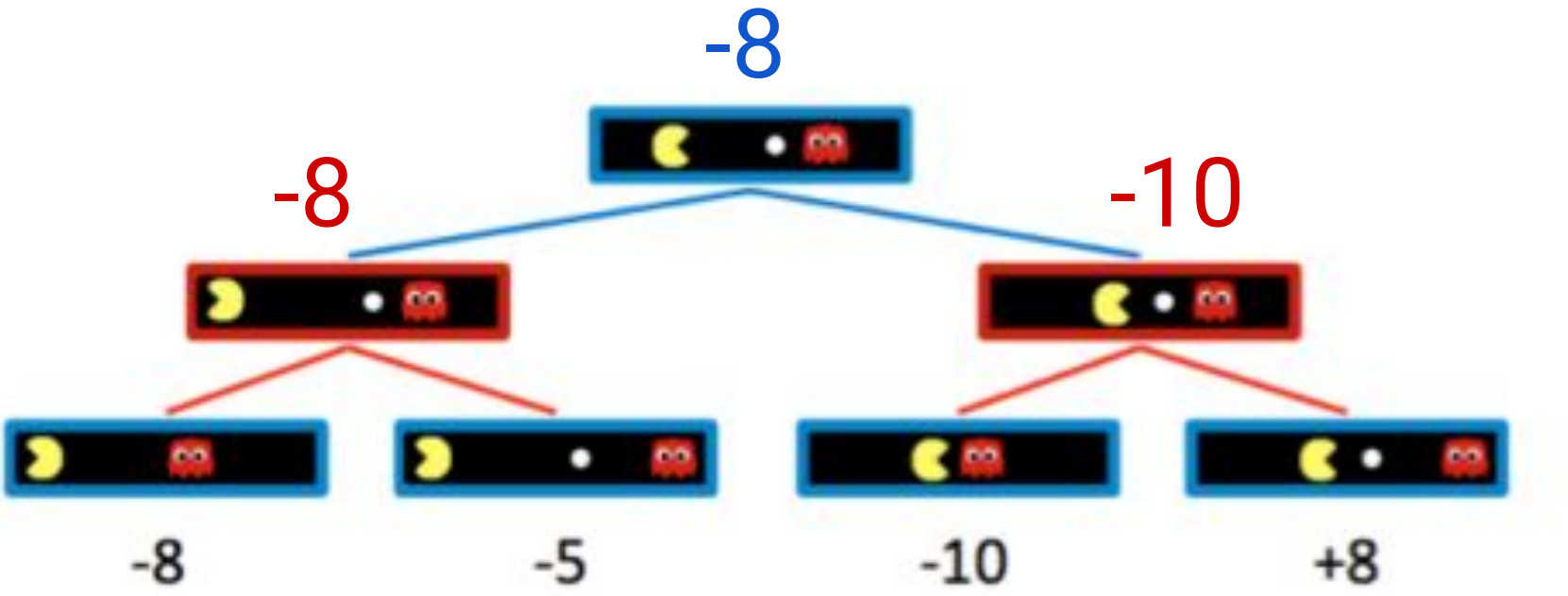
$$\forall \text{ agent-controlled states}, \quad V(s) = \max_{s' \in successors(s)} V(s')$$

$$\forall \text{ opponent-controlled states}, \quad V(s) = \min_{s' \in successors(s)} V(s')$$

$$\forall \text{ terminal states}, \quad V(s) = \text{known}$$

Minimax Example

- Pacman maximizes
- Ghost minimizes



Alpha-Beta Pruning

- Minimax runtime is $O(b^m)$:(
  - b = branching factor, m = approx tree depth where terminal nodes are found
- **Alpha-Beta Pruning:**
  - Say we're looking at some node n. When we go through n's children, if we realize that node n's value is guaranteed to be made redundant by another node, stop checking n's children!
  - Runtime about $O(b^{m/2})$

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
```

```
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 (continued)

- α represents MAX's best (highest) available option–a minimum value for any MIN nodes
  - if a MIN node has a child ≤ α, it will never be high enough to change the MAX nodes above!
  - α is updated in MAX nodes to be its current best (highest) value
- β represents MIN's best (lowest) available option–a maximum value for any MAX nodes
  - if a MAX node has a child ≥ β, it will never be small enough to change the MIN nodes above!
  - β is updated in MIN nodes to be its current best (lowest) value
- α and β are passed down to children

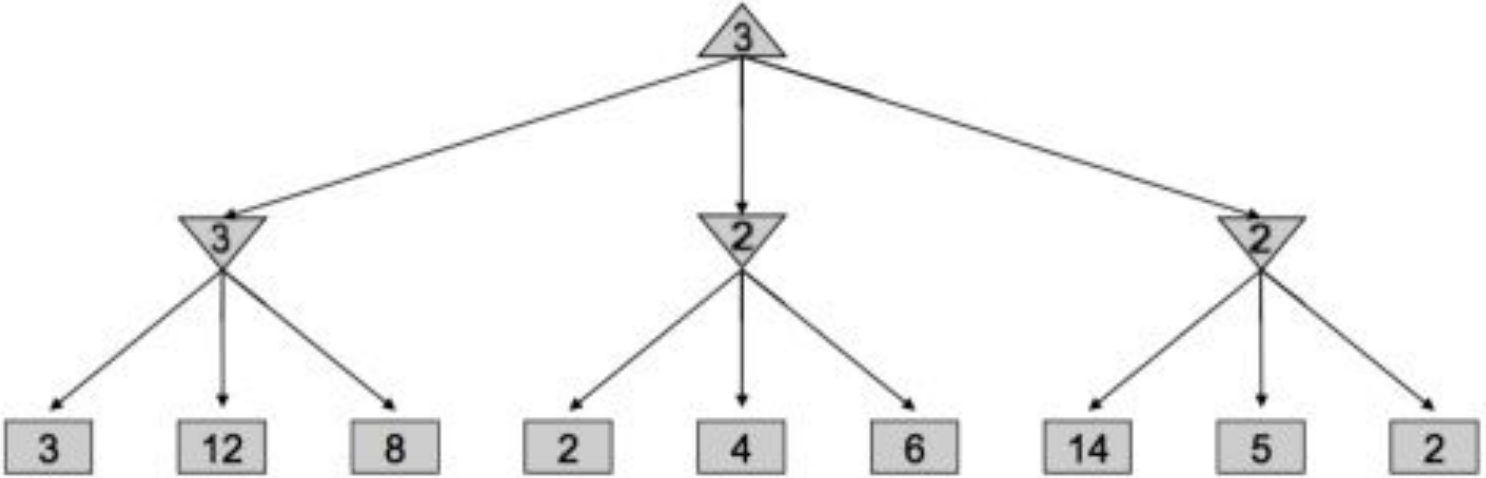
α: MAX's best option on path to root
β: MIN's best option on path to root

```
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
```

```
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
```

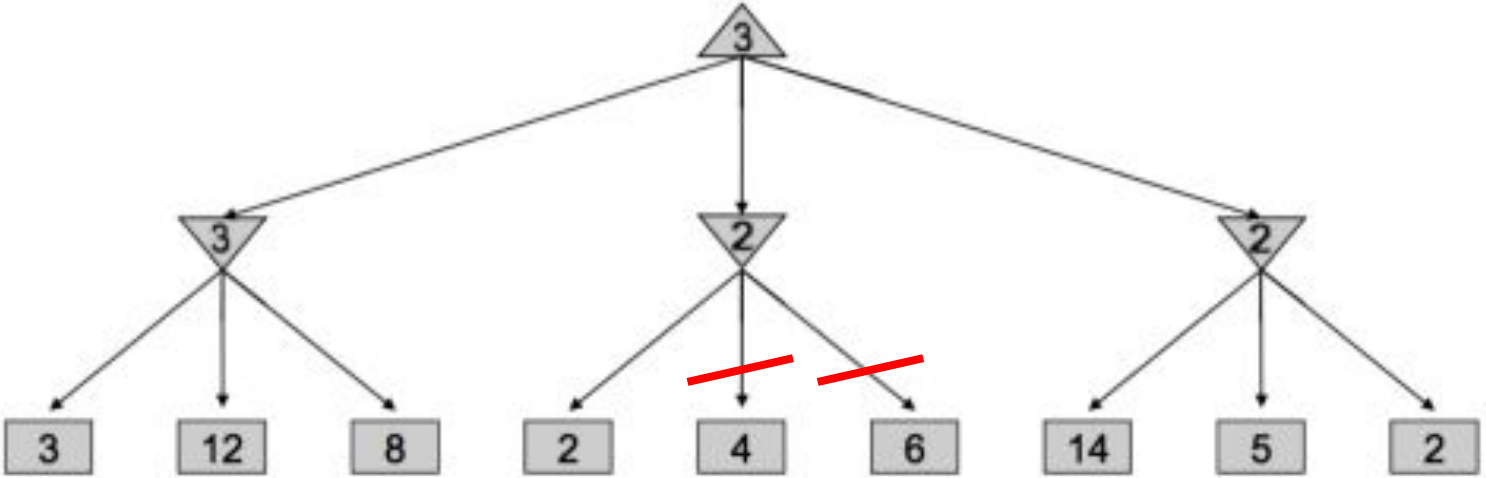
Alpha-Beta Pruning (continued)

- Example for me to copy onto the board



Alpha-Beta Pruning (continued)

- Solution



Worksheet 1(a) + 1(b)

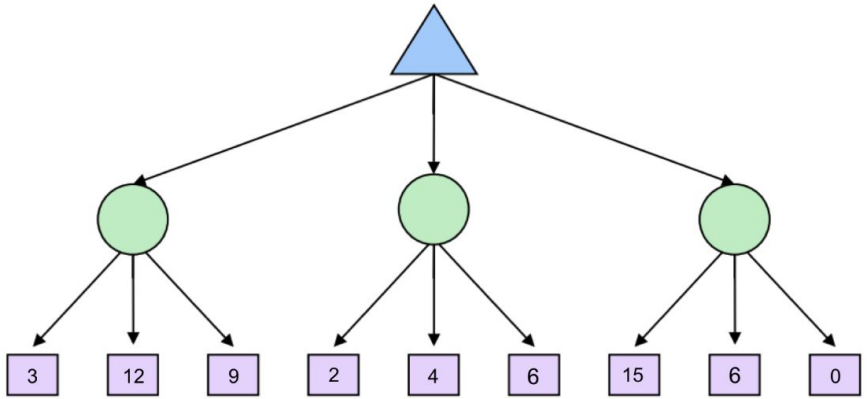
Expectimax

- Minimizer nodes replaced by **chance nodes**, which find the expected value of their children
- Captures non-optimal behavior

$$\forall \text{ agent-controlled states}, \quad V(s) = \max_{s' \in successors(s)} V(s')$$

$$\forall \text{ chance states}, \quad V(s) = \sum_{s' \in successors(s)} p(s'|s) V(s')$$

$$\forall \text{ terminal states}, \quad V(s) = \text{known}$$



Worksheet 1(c) + 1(d)

Worksheet 2

Thank you for attending!

Attendance link:

- https://tinyurl.com/cs188fa23

Discussion No: 4

Remember my name is Kenny

My email: kwkw@berkeley.edu