

---

# CSE 573: Artificial Intelligence

## Winter 2019

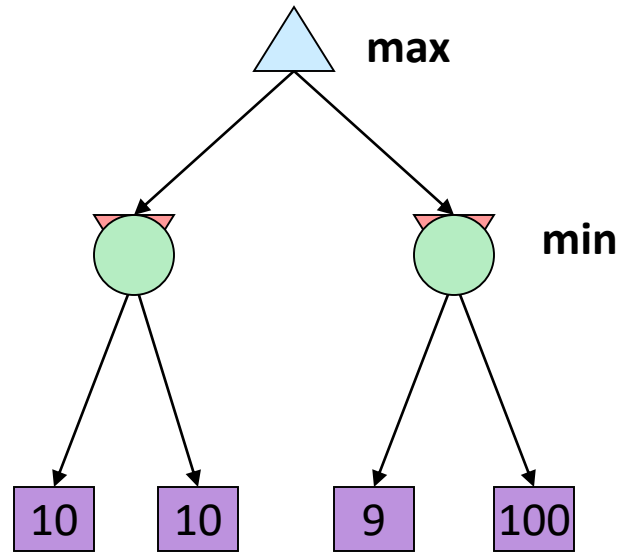
Hanna Hajishirzi  
Expectimax – Complex Games

slides from

Dan Klein, Stuart Russell, Andrew Moore, Dan Weld, Pieter Abbeel, Luke Zettlemoyer

# Worst-Case vs. Average Case

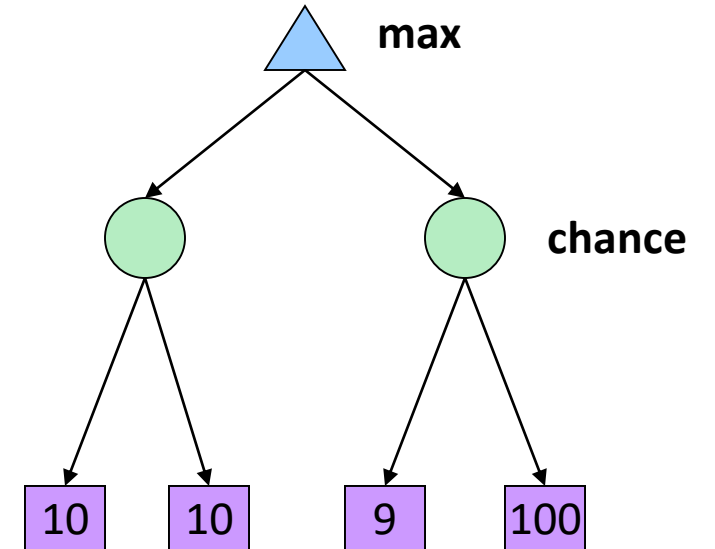
---



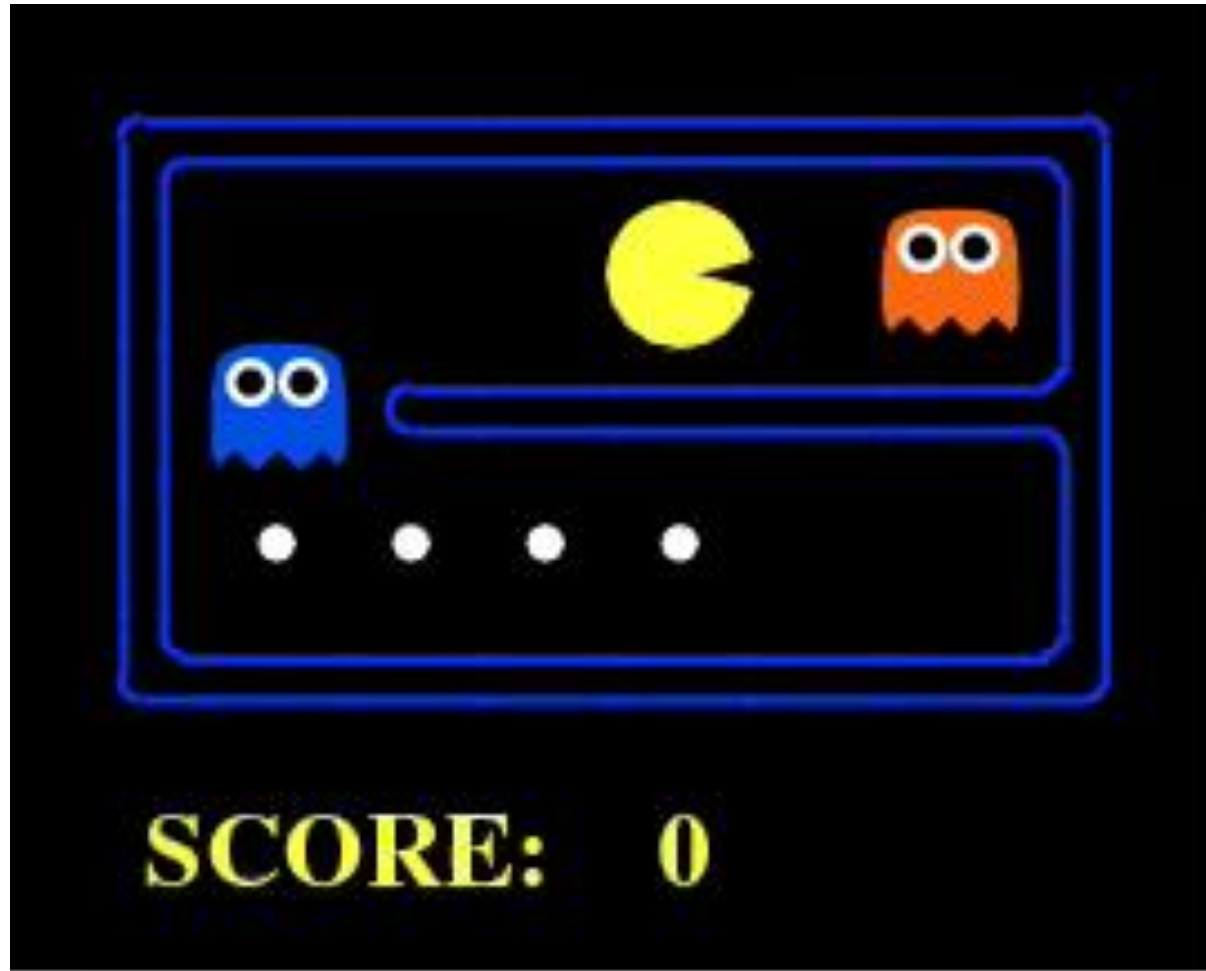
Idea: Uncertain outcomes controlled by chance, not an adversary!

# Expectimax Search

- Why wouldn't we know what the result of an action will be?
  - Explicit randomness: rolling dice
  - Unpredictable opponents: the ghosts respond randomly
  - Actions can fail: when moving a robot, wheels might slip
- Values should now reflect average-case (expectimax) outcomes, not worst-case (minimax) outcomes
- **Expectimax search**: compute the average score under optimal play
  - Max nodes as in minimax search
  - Chance nodes are like min nodes but the outcome is uncertain
  - Calculate their **expected utilities**
  - I.e. take weighted average (expectation) of children
- Later, we'll learn how to formalize the underlying uncertain-result problems as **Markov Decision Processes**

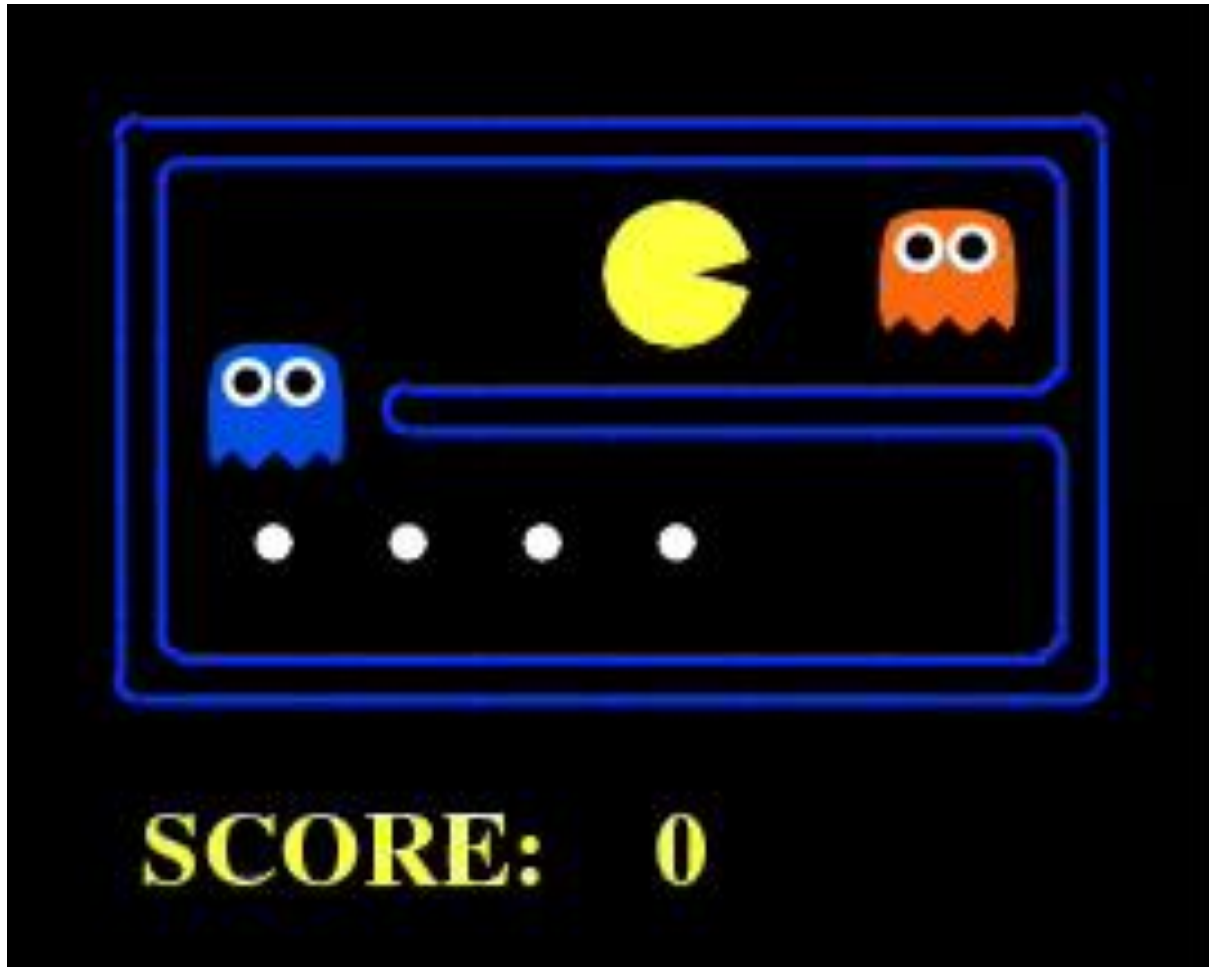


# Minimax vs Expectimax (Min)



No point in trying

# Minimax vs Expectimax (Min)



3 ply look ahead, ghosts  
move randomly

Wins some of the  
games

# Expectimax Pseudocode

```
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 EXP: return exp-value(state)
```

```
def max-value(state):
```

```
    initialize v =  $-\infty$ 
```

```
    for each successor of state:
```

```
        v = max(v, value(successor))
```

```
    return v
```

```
def exp-value(state):
```

```
    initialize v = 0
```

```
    for each successor of state:
```

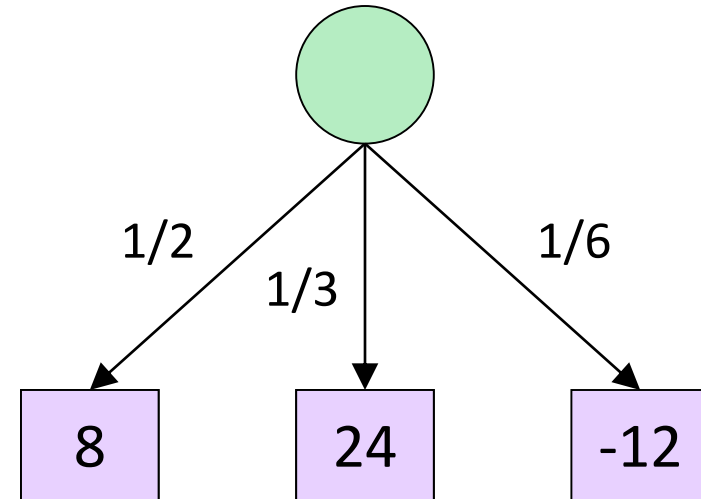
```
        p = probability(successor)
```

```
        v += p * value(successor)
```

```
    return v
```

# Expectimax Pseudocode

```
def exp-value(state):  
    initialize v = 0  
    for each successor of state:  
        p = probability(successor)  
        v += p * value(successor)  
    return v
```



$$v = (1/2) (8) + (1/3) (24) + (1/6) (-12) = 10$$

# Next Time: MDPs!

---