CSE 473

Lecture 15

Markov Decision Processes (MDPs)

Course Overview: Where are we?

- Introduction & Agents
- Search and Heuristics
- Adversarial Search
- Logical Knowledge Representation
- Markov Decision Processes (**MDPs)**
- Reinforcement Learning
- Uncertainty & Bayesian Networks
- Machine Learning

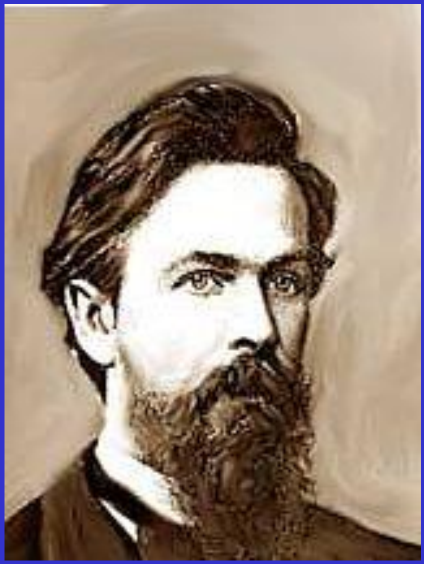
MDPs

Markov Decision Processes

- Planning Under Uncertainty

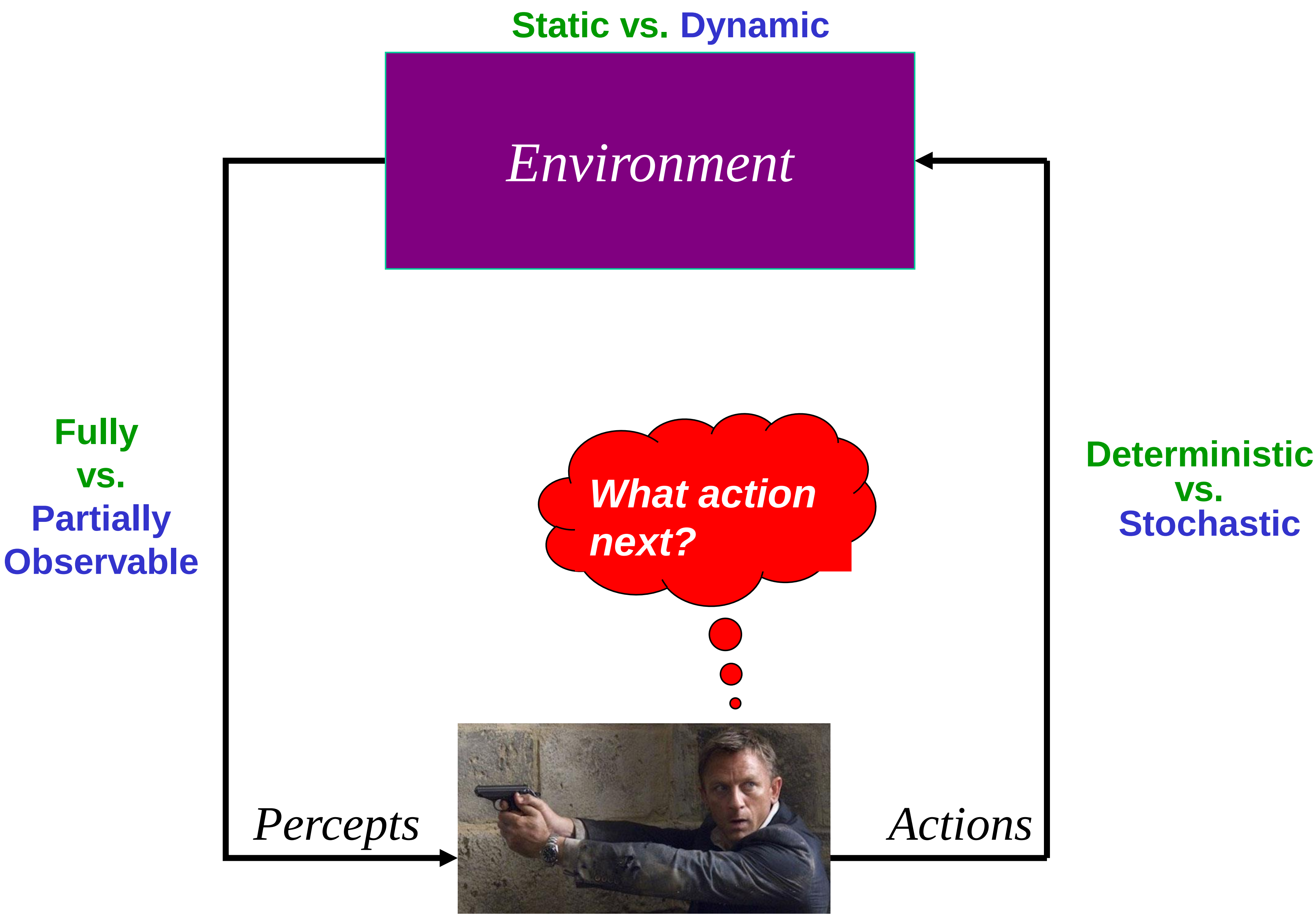
- Mathematical Framework
- Bellman Equation
- Value Iteration
- Policy Iteration

- Reinforcement Learning



Andrey Markov
(1856-1922)

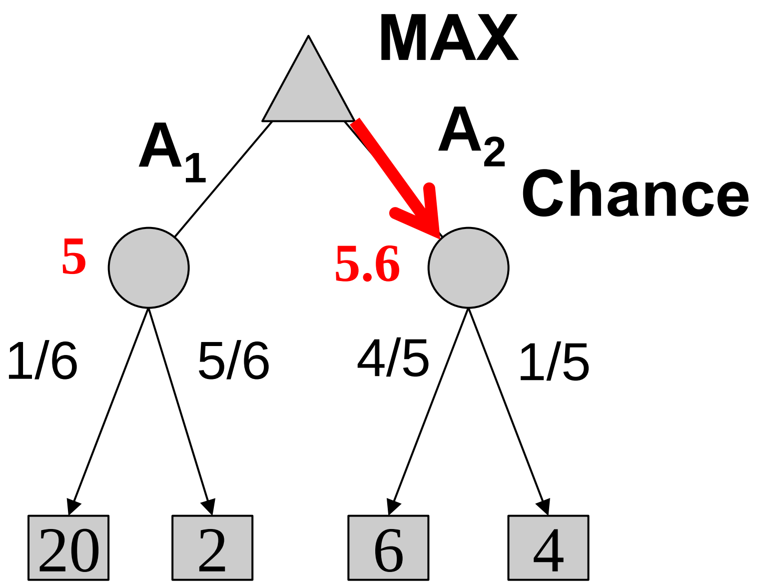
Planning Agent



Review: Expectimax

- What if we don't know what the result of an action will be? E.g.,
  - In Solitaire, next card is unknown
  - In Pacman, the ghosts act randomly

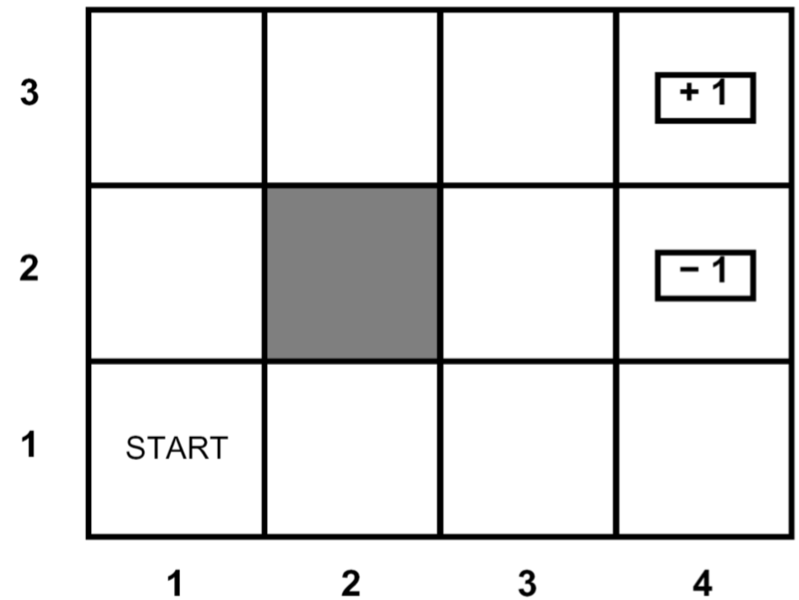
- Can do expectimax search
  - Max nodes as in minimax search
  - Chance nodes, like min nodes, except the outcome is uncertain - take average (expectation) of children
  - Calculate expected utilities

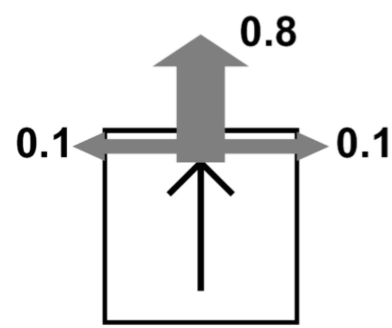


- Today, we formalize this as a Markov Decision Process
  - Handles ***intermediate rewards*** & ***infinite search trees***
  - More efficient processing

Example: Grid World

- Walls block the agent's path
- Agent's actions are noisy:
  - 80% of the time, North action takes the agent North (assuming no wall)
  - 10% - actually go West
  - 10% - actually go East
  - If there is a wall in the chosen direction, the agent stays put
- Small "living" penalty (e.g., -0.04) each step
- Big reward/penalty (e.g., +1 or - 1) comes at the end
- Goal: maximize sum of rewards

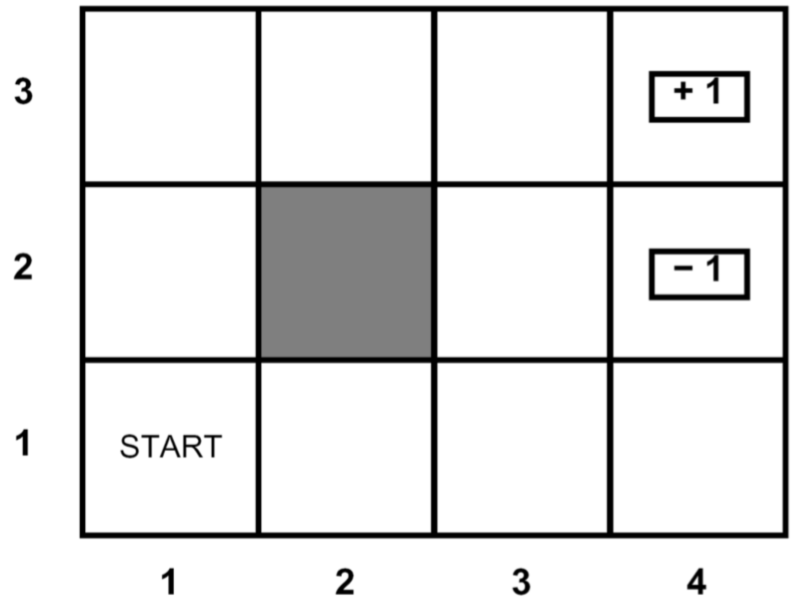


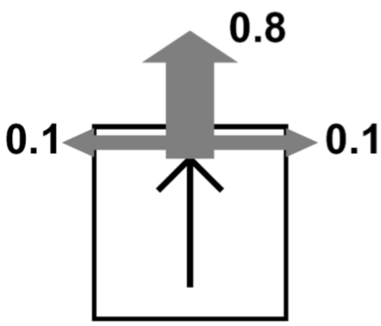


Markov Decision Processes

- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function T(s,a,s’)
    - Probability that action a in s leads to s’

      i.e., P(s’ | s,a)
    - Also called “the model”
  - A reward function R(s, a, s’)
    - Sometimes just R(s) or R(s’)
  - A start state
  - Maybe a terminal state





What is Markov about MDPs?

- "Markov" generally means that
  - Given the present state, the future is ***independent*** of the past

Andrey Markov
(1856-1922)

- For Markov decision processes, "Markov" means:

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots S_0 = s_0)$$

$$=$$

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

Next state only depends on current state and action

Solving MDPs

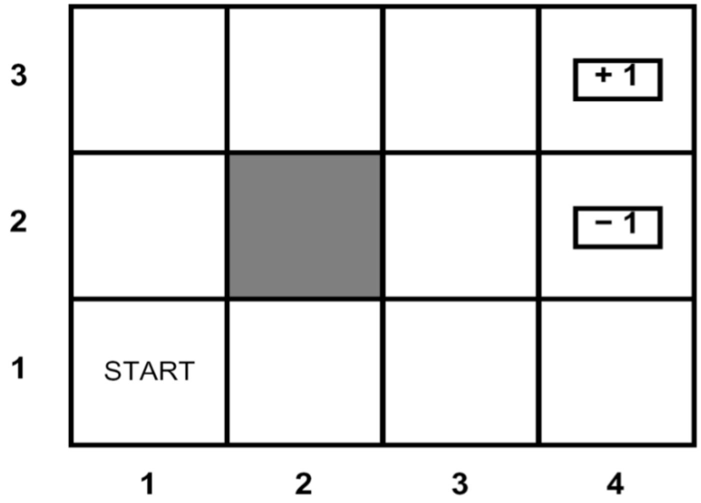
- In deterministic search problems, want an optimal path or plan (sequence of actions) from start to a goal
- MDP: Stochastic actions, don’t know what next state will be
- Instead of path/plan, use an optimal policy $\pi^*: S \rightarrow A$
  - Policy π prescribes an action for every state
  - Defines a reflex agent
  - An optimal policy maximizes expected reward if followed

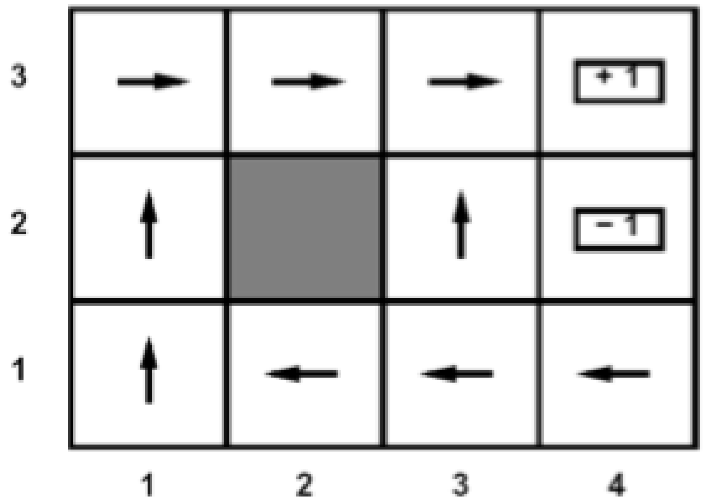
Solving MDPs

Optimal policy?

Assume R(s, a, s') = -0.04

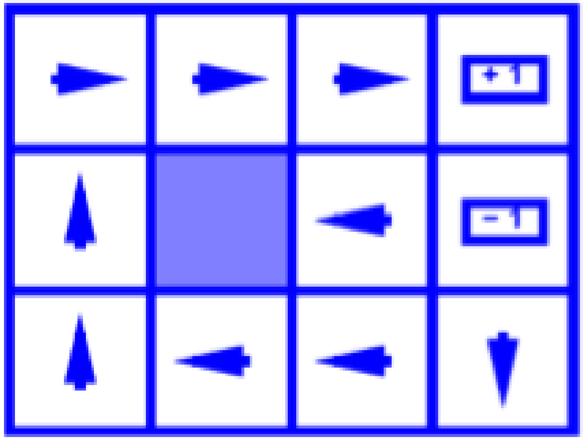
for all non-terminal s



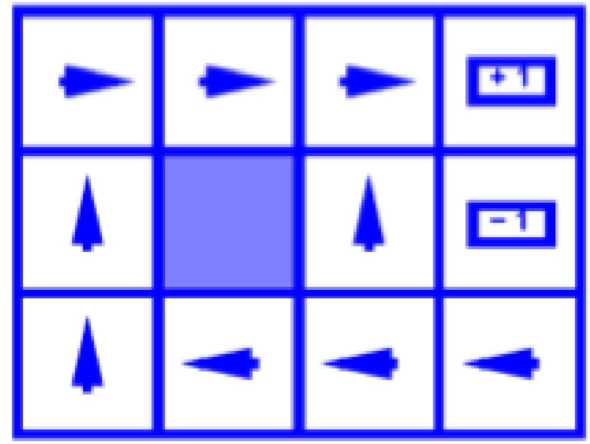


More Example Optimal Policies

Conservative

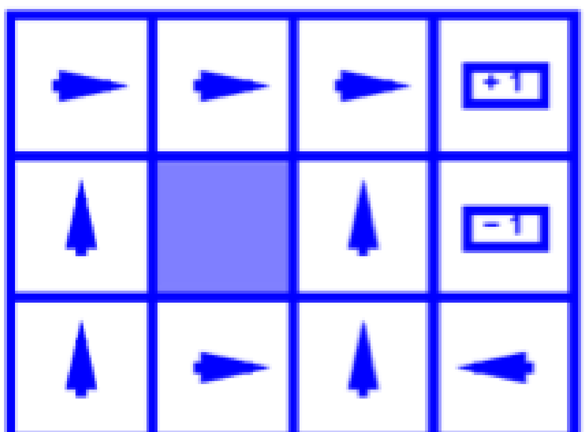


R(s) = -0.01



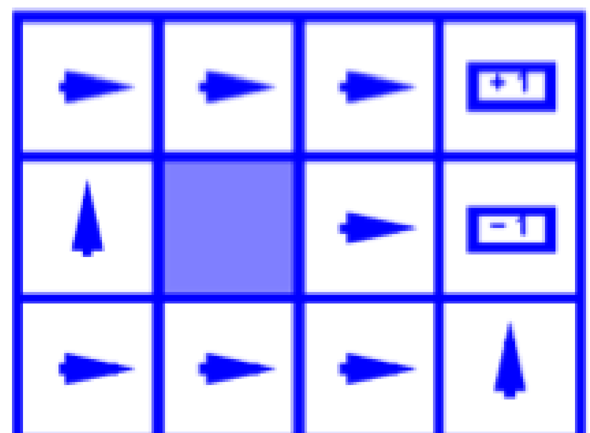
R(s) = -0.04

Aggressive



R(s) = -0.4

Suicidal

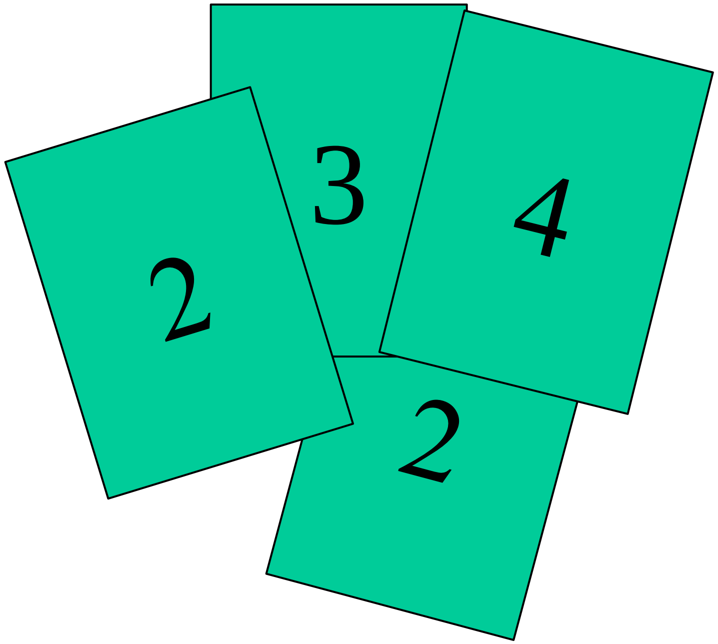


R(s) = -2.0

Another Example:
High-Low Card Game

Example: High-Low

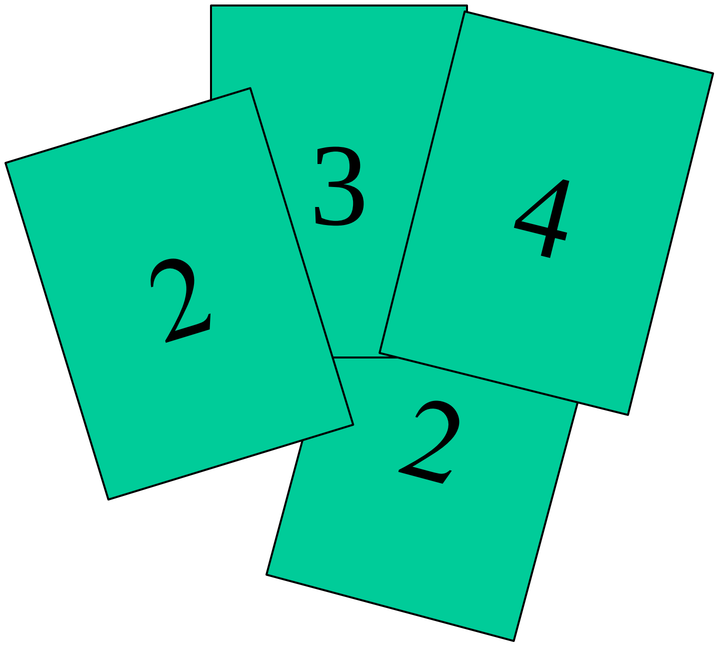
- Suppose three card types: 2, 3, 4
  - Infinite deck, twice as many 2's
- Start with 3 showing
- After each card, say "high" or "low"
- New card is revealed
  - If you're right, you win the points shown on the new card
  - Tie: no reward, choose again
  - If you're wrong, game ends



- Differences from expectimax problems:
  - #1: get rewards as you go
  - #2: you might play forever!

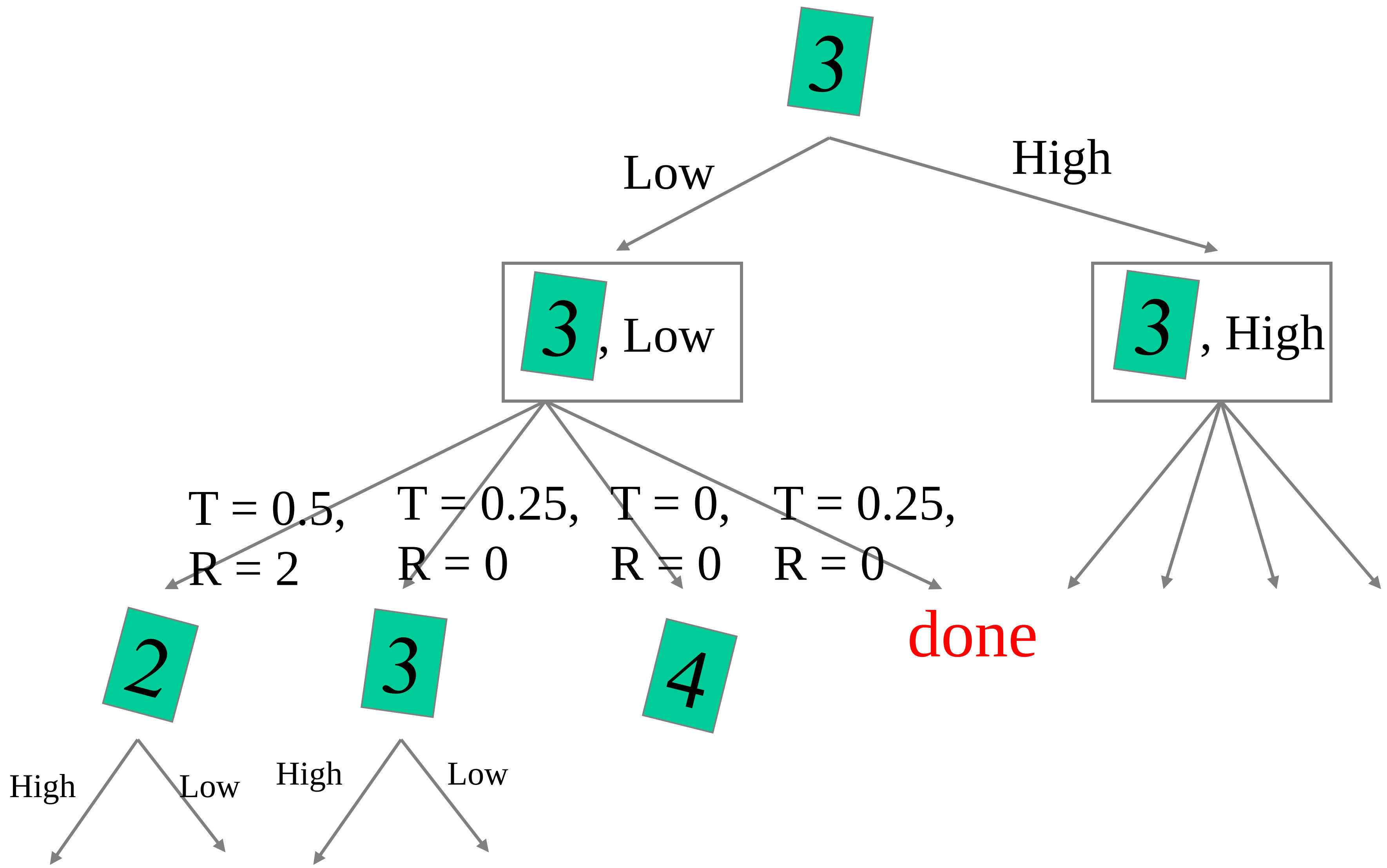
High-Low as an MDP

- States:
  - 2, 3, 4, done
- Actions:
  - High, Low
- Model: T(s, a, s') = P(s' | s, a):
  - P(s'=4 | 4, Low) = 1/4
  - P(s'=3 | 4, Low) = 1/4
  - P(s'=2 | 4, Low) = 1/2
  - P(s'=done | 4, Low) = 0
  - P(s'=4 | 4, High) = 1/4
  - P(s'=3 | 4, High) = 0
  - P(s'=2 | 4, High) = 0
  - P(s'=done | 4, High) = 3/4
  - …



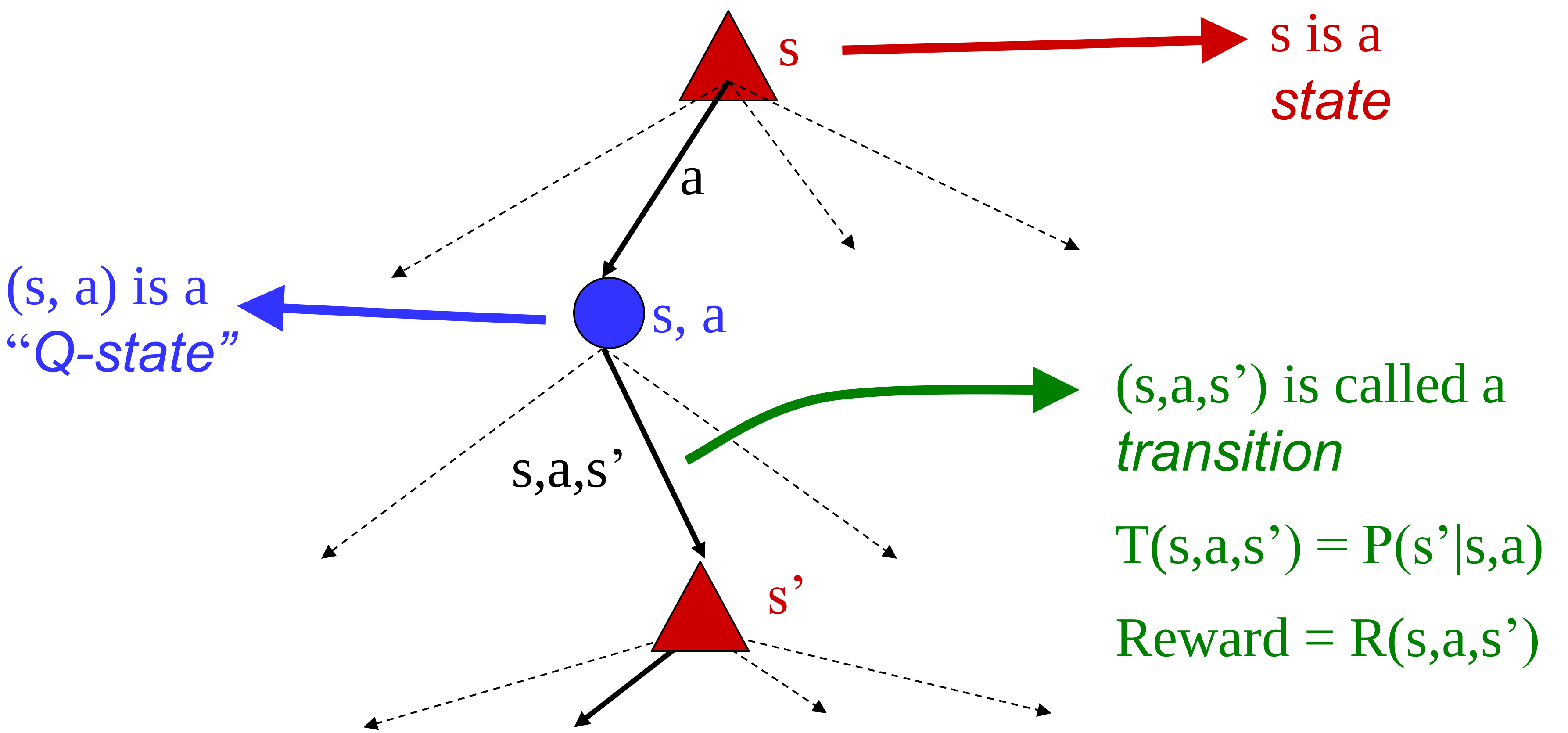
- Rewards: R(s, a, s'):
  - Number shown on s' if s' > s $\wedge$ a="High" etc.
  - 0 otherwise
- Start: 3

Expectimax-like Search Tree for High-Low



MDP Search Trees

- Each MDP state gives an expectimax-like search tree



Utilities of Reward Sequences

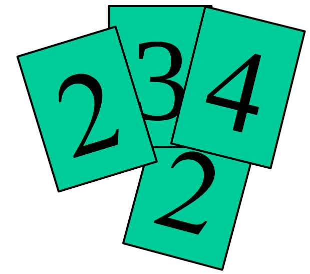
- What is an "optimal" policy?
  - Each transition s,a,s' produces a reward (+ve, -ve, or 0)
  - Need to define utility of a *sequence* of rewards

- Idea 1:
  - Additive utility:

$$U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$$

Defining Utilities

- Problem: Infinite state sequences have infinite total reward



- Solutions:
  - Impose a *Finite Horizon* (deadline):
    - Terminate episodes after a fixed T steps (e.g. life)
    - Gives nonstationary policies (π depends on time left)
  - Absorbing state: guarantee that a terminal state will eventually be reached (like “done” for High-Low)
  - Discounting: Make infinite sum finite using γ ($0 < \gamma < 1$)

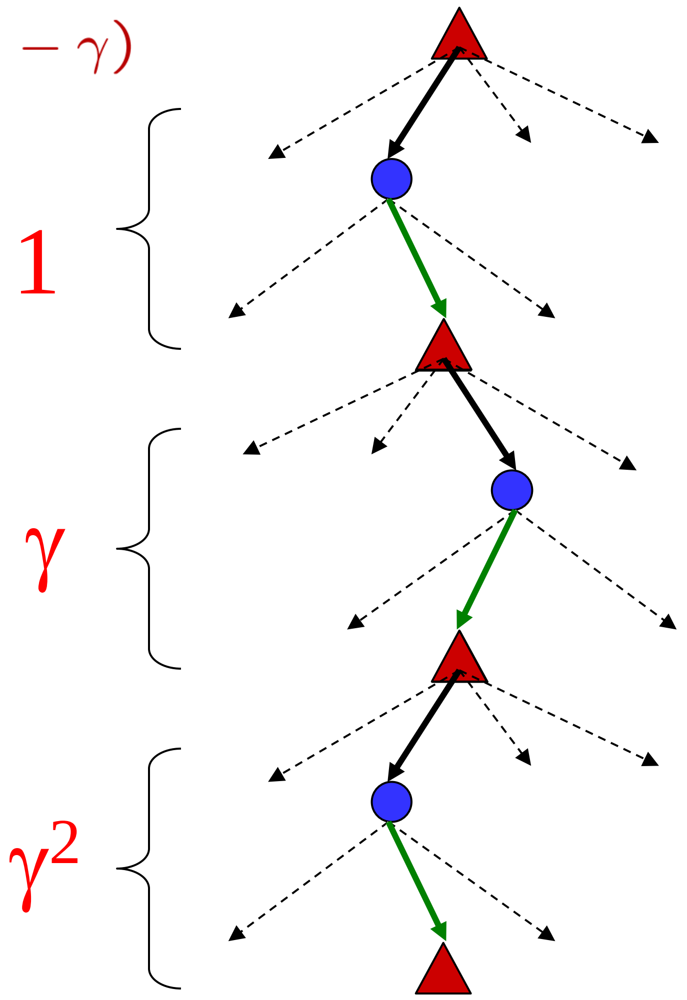
$$U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$$

$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\text{max}}/(1-\gamma)$$

Discounting Rewards

$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1-\gamma)$$

- Discount rewards by $\gamma < 1$ each time step
  - Sooner rewards have higher utility than later rewards
  - Also helps the algorithms converge



Next Time

- Using utility to find the optimal policy
- To do
  - Read chapters 13 and 17