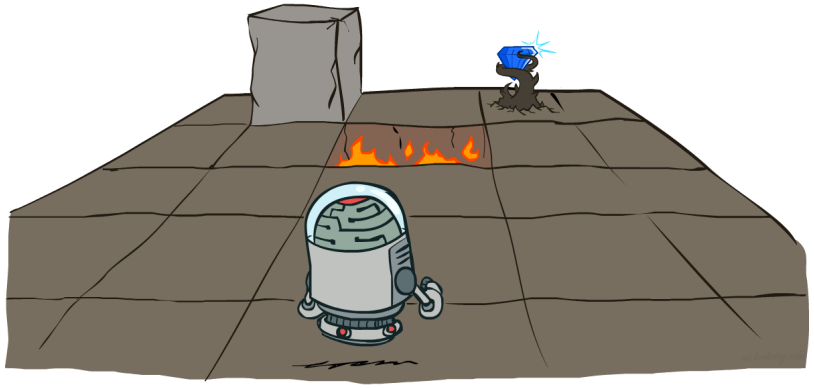
Markov Decision Processes

CSE 473: Introduction to Artificial Intelligence



James Weichert (jpw@cs)
cs.uw.edu/473
acknowledgements

Welcome!
What's something that didn't go to plan?
sli.do **#cse473**

Administrivia

ANNOUNCEMENTS

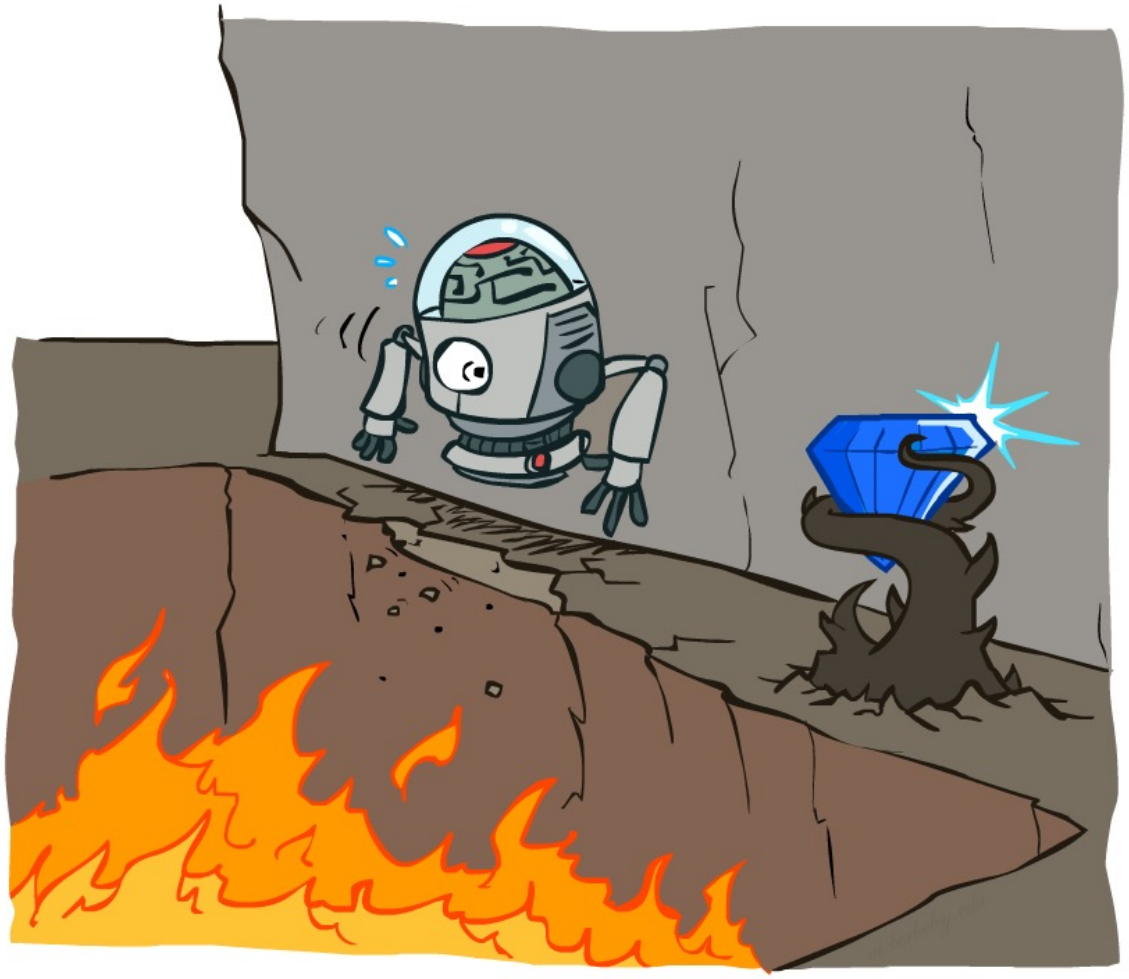
- For each project, make sure to fill out the **AI usage reflection** (cs.uw.edu/473/projects)
- **Test 1** on Friday
  - One double-sided 8.5x11" note sheet allowed
- James out of town 7.8 - 7.16

ASSIGNMENTS

- **Project 1 due** this Thursday (7.9)
- **Project 2** releasing early
  - **due** next Thursday (7.16)

Non-Deterministic Search

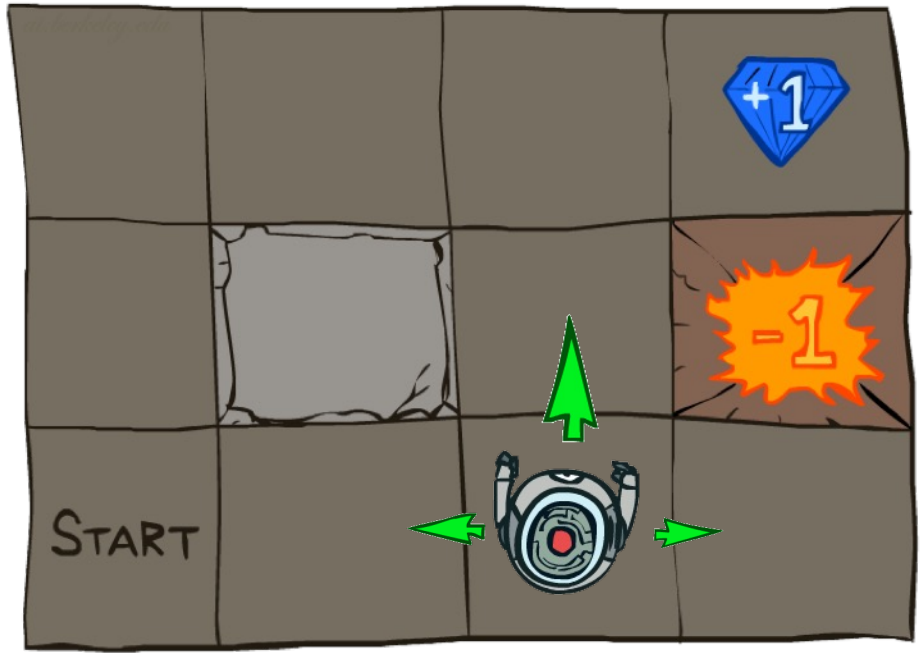
CONTEXT



Grid World

EXAMPLE

- Maze-Like Environment
- Noisy Movement
  - Actions don't always go as planned
  - e.g. 80% of time, **NORTH** action takes the agent north, 10% of time agent ends up going west, 10% of time agent ends up going east
  - If there's a wall in the direction the agent would have gone, the agent doesn't move
- Reward
  - Agent receives small living reward *r* (maybe negative)
  - Big rewards at terminal states

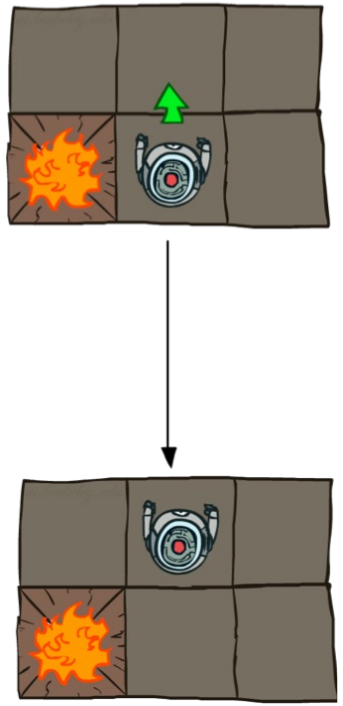


Goal: maximize sum of rewards

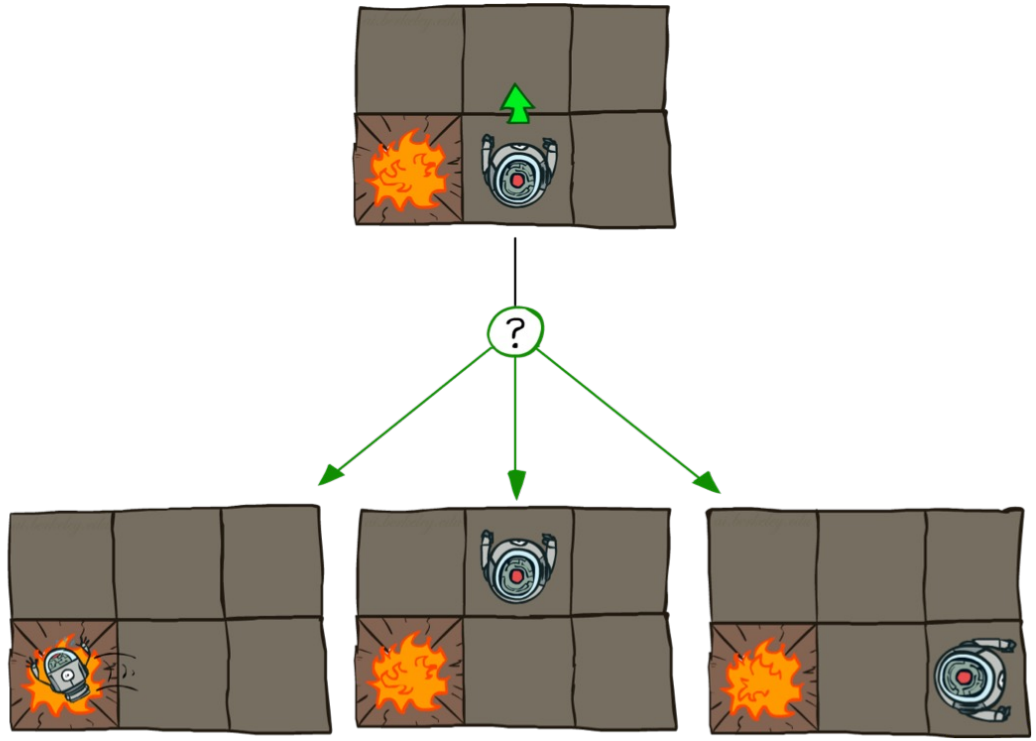
Grid World Actions

EXAMPLE

Deterministic



Stochastic



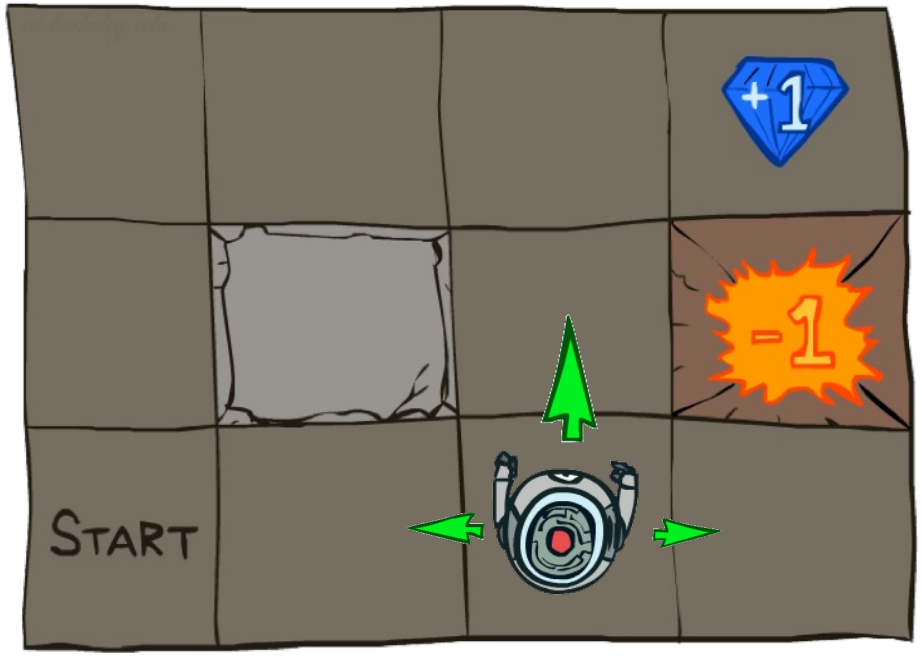
Markov Decision Process

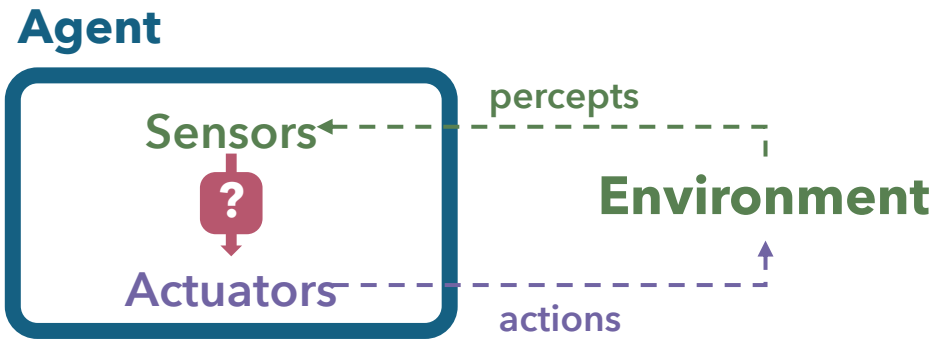
FOUNDATIONS

- Markov Decision Process (MDP) defined as:
  - States $s \in S$
  - Actions $a \in A$
  - Transition model $T(s, a, s')$

    $$T(s, a, s') = P(s'|s, a)$$
  - Reward function $R(s, a, s')$
  - Start state s_0
  - Utility function $U(s)$

MDPs are fully observable ***stochastic*** search problems





Markov Assumption

DEFINITION

- Environment History
  - $[s_0, a_0, s_1, a_1, \dots, s_t]$
- Conditional Independence
  - Under the Markov assumption, successor state depends only on the previous state-action pair
  - Equivalent statement: *"given the present state, the future is independent of the past"*

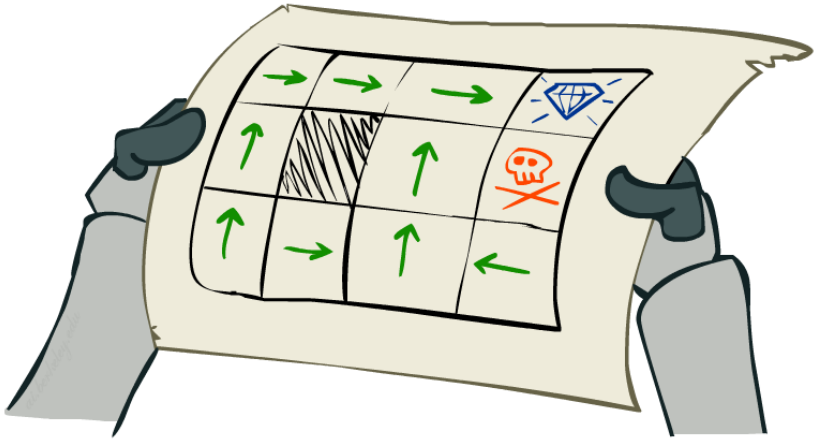
$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}, \dots, A_1 = a_1, S_0 = s_0)$$

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

Policies

DEFINITION

- A **policy** maps states to actions $\pi: S \rightarrow A$
- Deterministic Search
  - Search for an optimal ***plan*** from the start to the goal
- Stochastic Search
  - Calculate an optimal ***policy*** to choose actions based on states



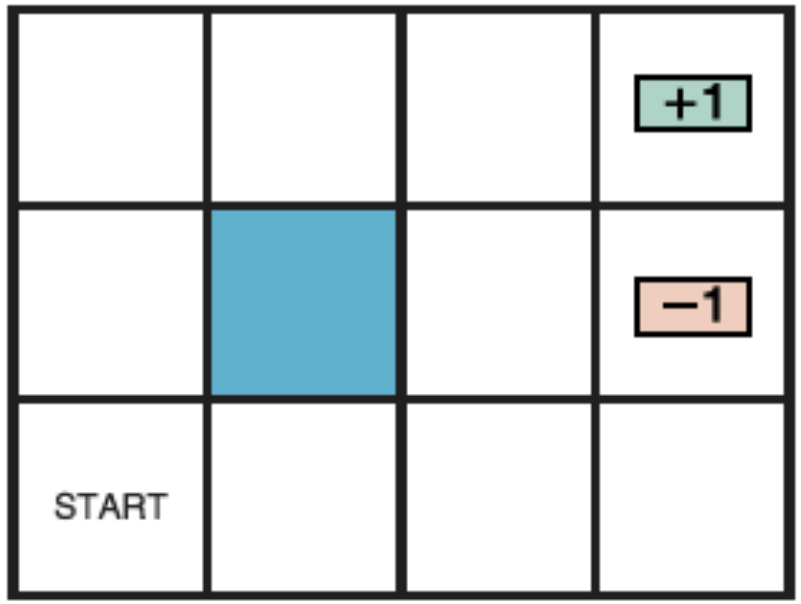
QUESTIONS

Questions?

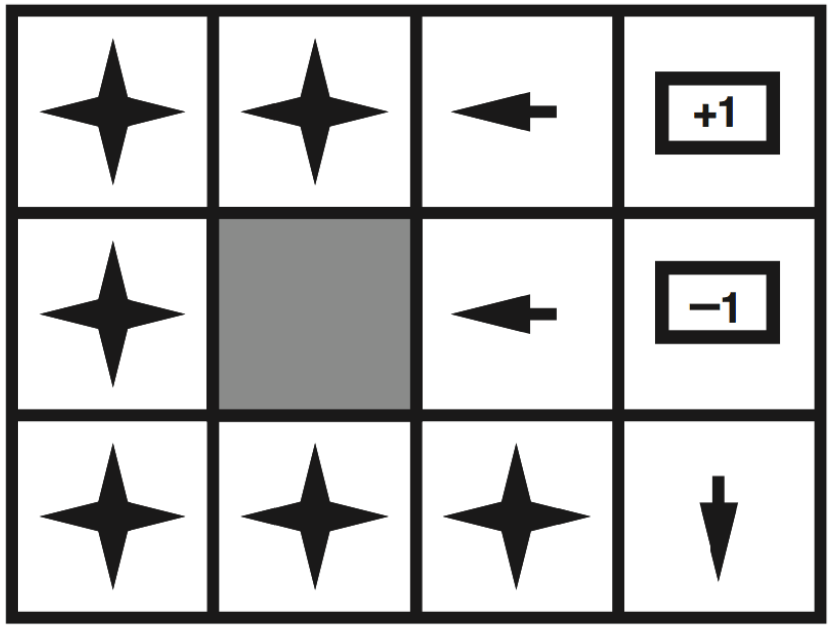
live and on sli.do #cse473

Optimal Policies

EXAMPLE



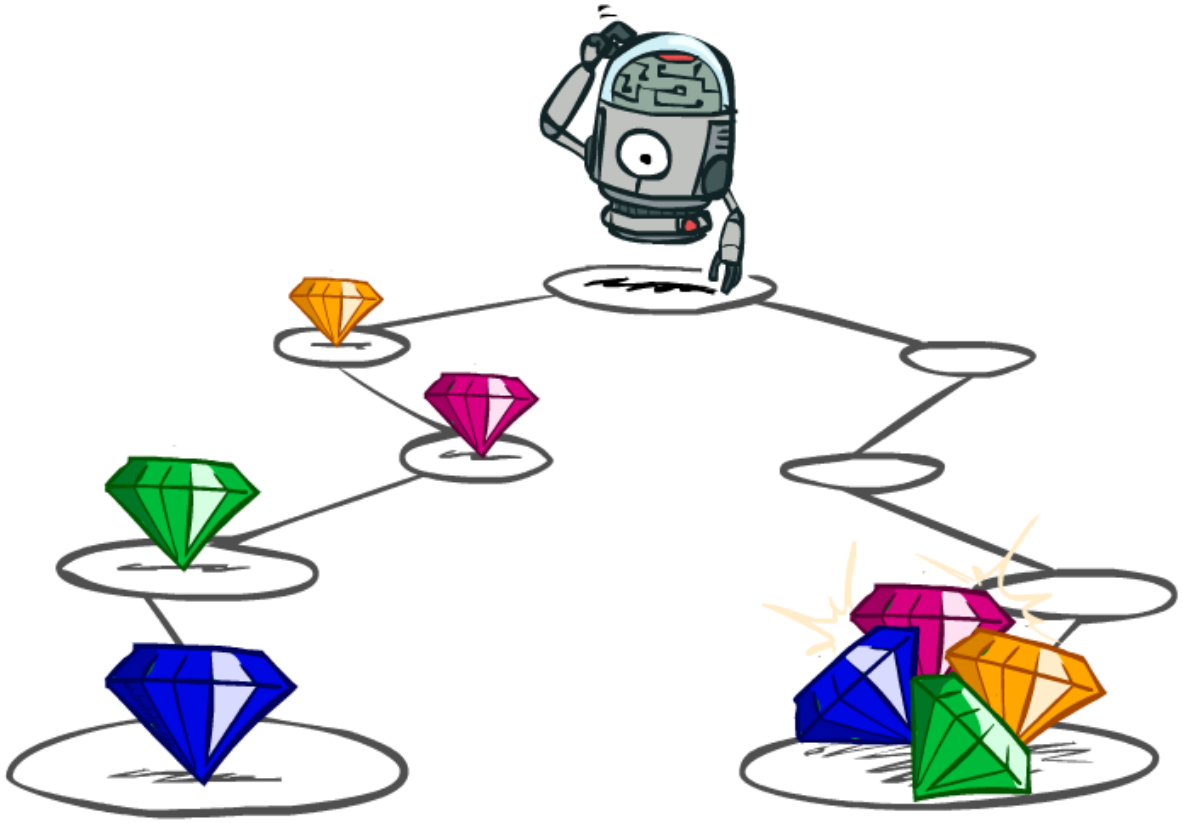
r > 0



r > 0

Sequential Utilities

CONTEXT



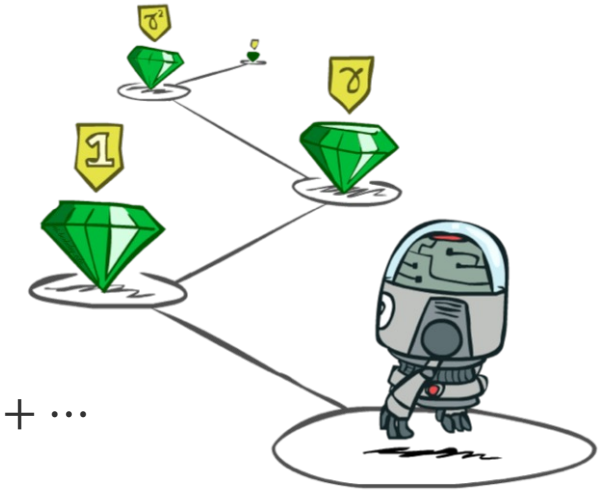
Stationary Preferences

DEFINITION

- If we prefer something now, we should also prefer it later
  - i.e. preference regardless of history

$$[s_0, a_0, s_1, a_1, s_2, \dots] > [s'_0, a'_0, s'_1, a'_1, s'_2, \dots], s_0 = s'_0, a_0 = a'_0, s_1 = s'_1$$

$$\Leftrightarrow [s_1, a_1, s_2, \dots] > [s'_1, a'_1, s'_2, \dots]$$



- Only one way to define finite utilities:

$$U([s_0, a_0, s_1, a_1, s_2, \dots]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \cdots$$

with **discount factor** γ

Discounting

FOUNDATIONS

worth γ now

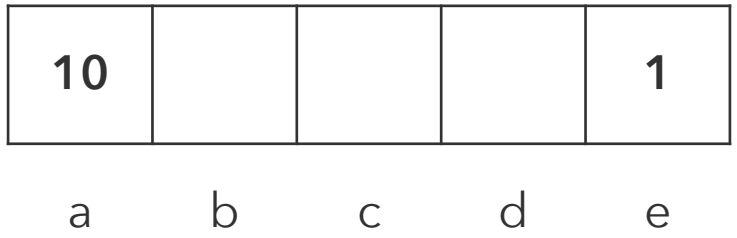
worth γr next step

worth $\gamma^2 r$ in two steps

- Discounting solves the problem of infinite reward streams
  - Geometric series: $1 + \gamma + \gamma^2 + \cdots = \frac{1}{(1-\gamma)}$
  - Assume rewards bounded by $\pm R_{\max}$
  - Then $r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots$ is bounded by $\pm \frac{R_{\max}}{(1-\gamma)}$

Discounting Practice

EXAMPLE



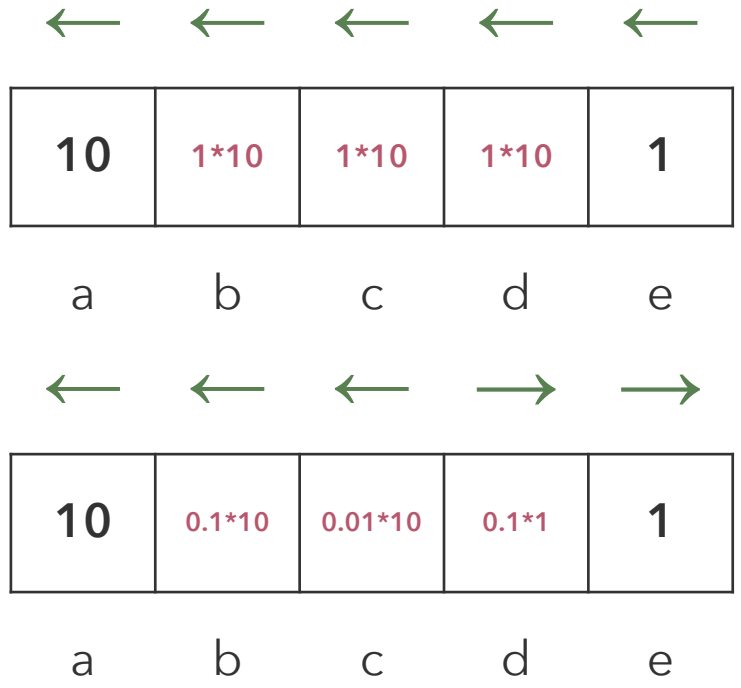
Consider this Grid World:

- Deterministic transitions
- Actions **EAST**, **WEST**, **EXIT** (only available at **a** and **e**)

- For $\gamma = 1$, what is the optimal policy?

- For $\gamma = 0.1$, what is the optimal policy?

For what γ are **EAST** *and* **WEST** *equally good at d?*



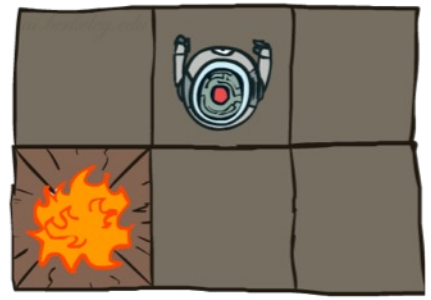
QUESTIONS

Questions?

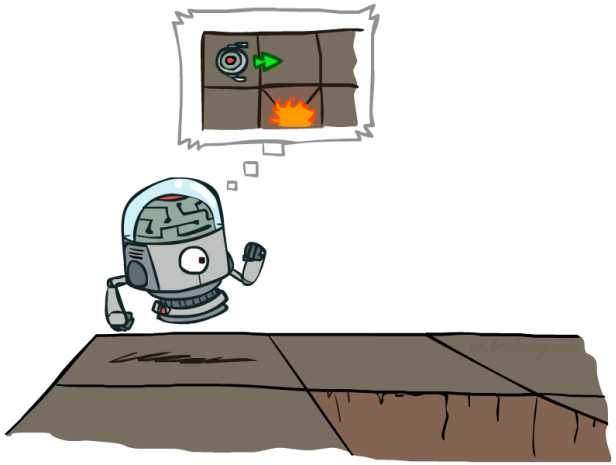
live and on sli.do #cse473

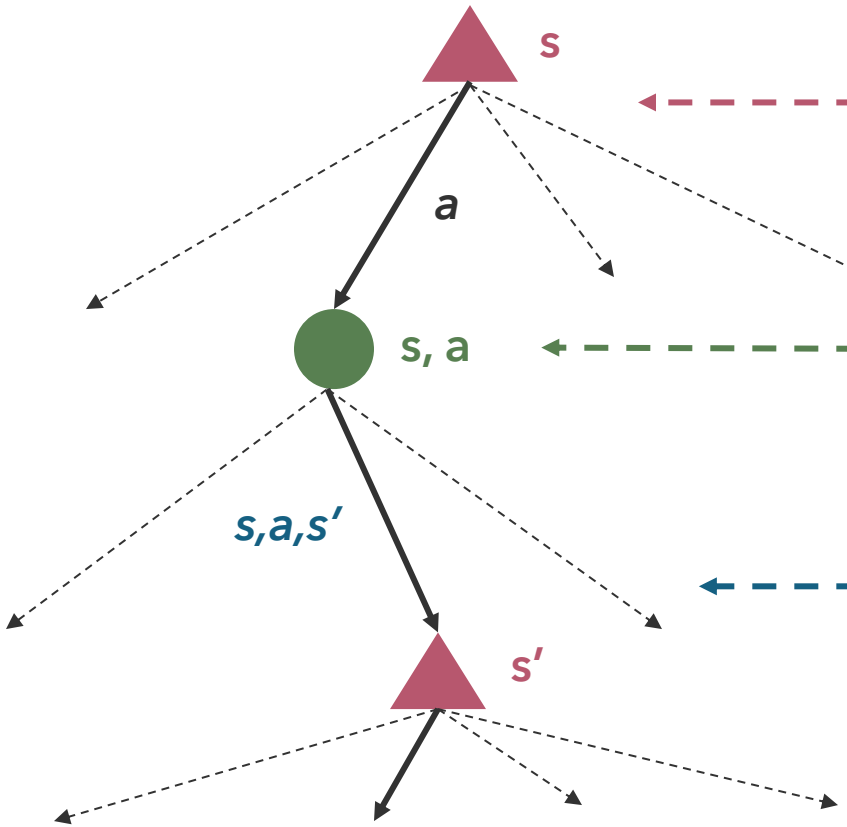
MDP Search Trees

FOUNDATIONS



MDPs can be represented as **expectimax search** trees





The children of **state nodes** are **state-action pairs**

Each **state-action pair** is a chance node

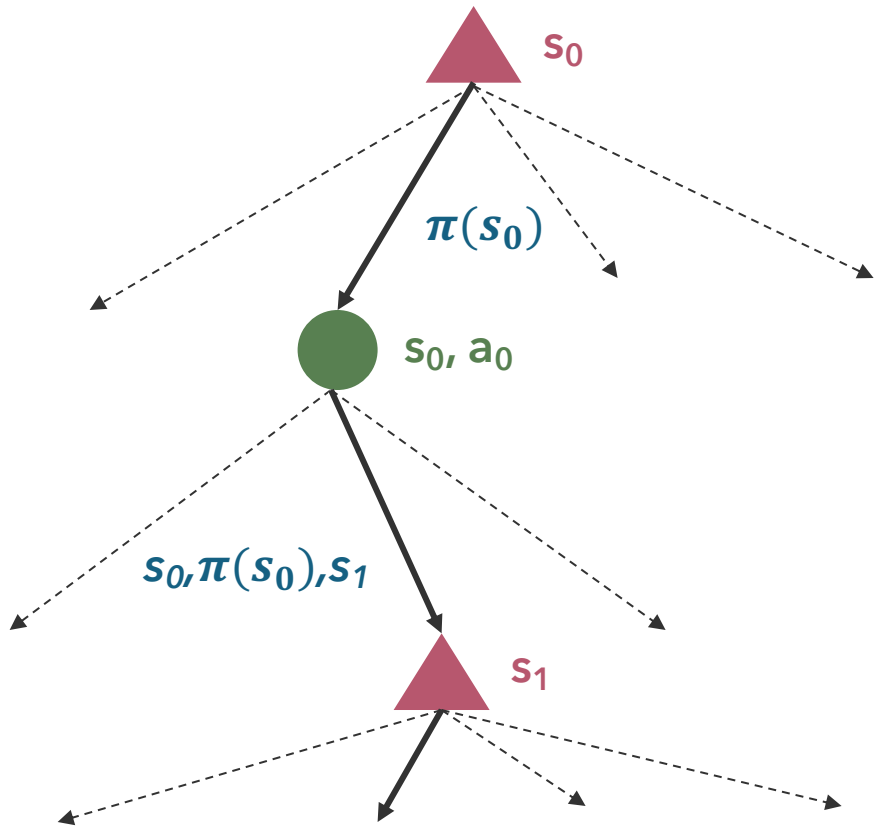
The children of chance nodes are **successor nodes**

Utility of a Policy

DEFINITION

- Executing policy π from state s_0
  - generates state-action sequence $s_0, \pi(s_0), s_1, \pi(s_1), s_2, \ldots$
- Corresponding to rewards
  - $\gamma^0 R(s_0, \pi(s_0), s_1) + \gamma^1 R(s_1, \pi(s_1), s_2) + \cdots$
- Occurring with joint probability
  - $P(s_1|s_0, \pi(s_0)) \cdot P(s_2|s_1, \pi(s_1)) \cdot \cdots$
- Expected utility (value) of policy π at s_0 :

$$U^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S'_t)\right]$$

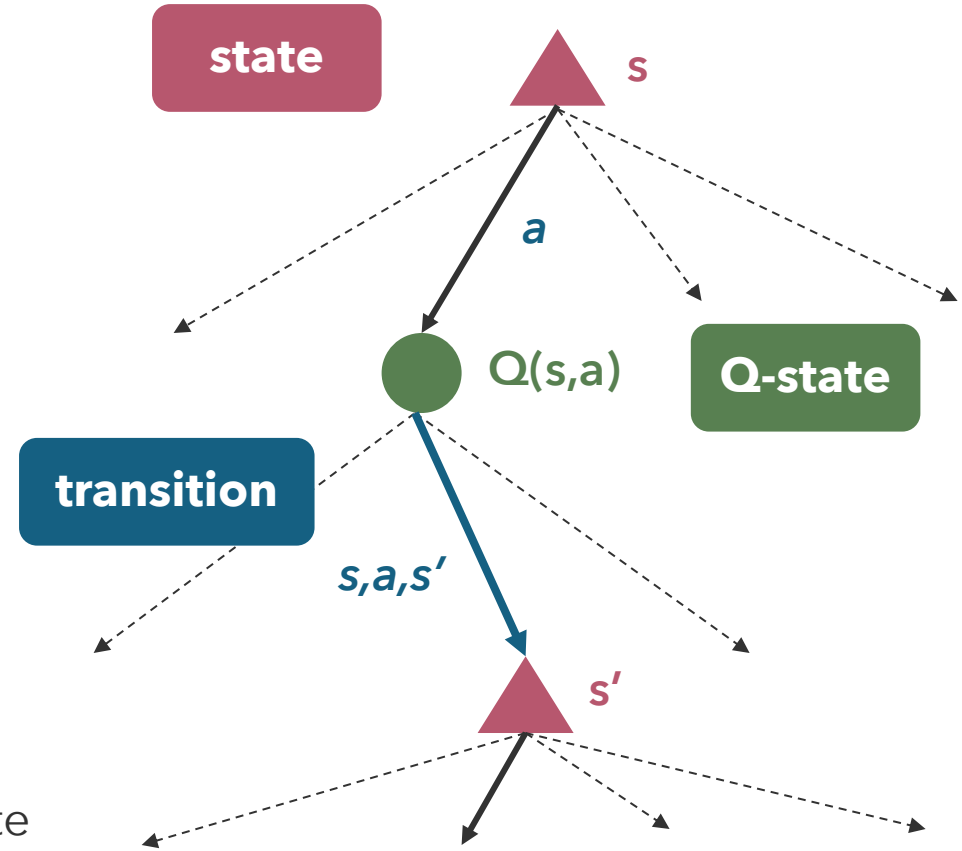


Optimal Quantities

FOUNDATIONS

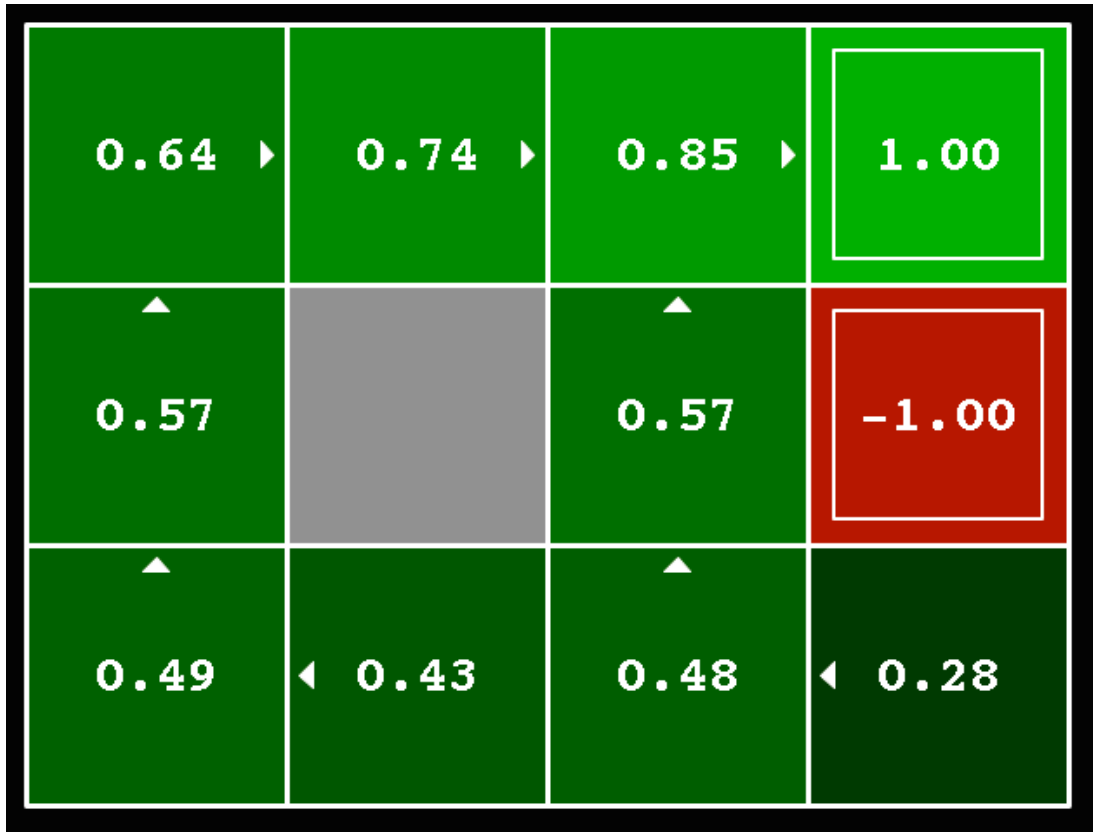
- Optimal Policy π^*
  - $\pi^* = \text{argmax}_\pi U^\pi(s)$
- Value of a State $U^*(s)$
  - $U^*(s) = U^{\pi^*}(s)$
  - Expected utility starting at state s and acting optimally
- Q-Value $Q(s, a)$
  - $Q^*(s, a)$ is the expected utility of taking action a from state s and acting optimally thereafter

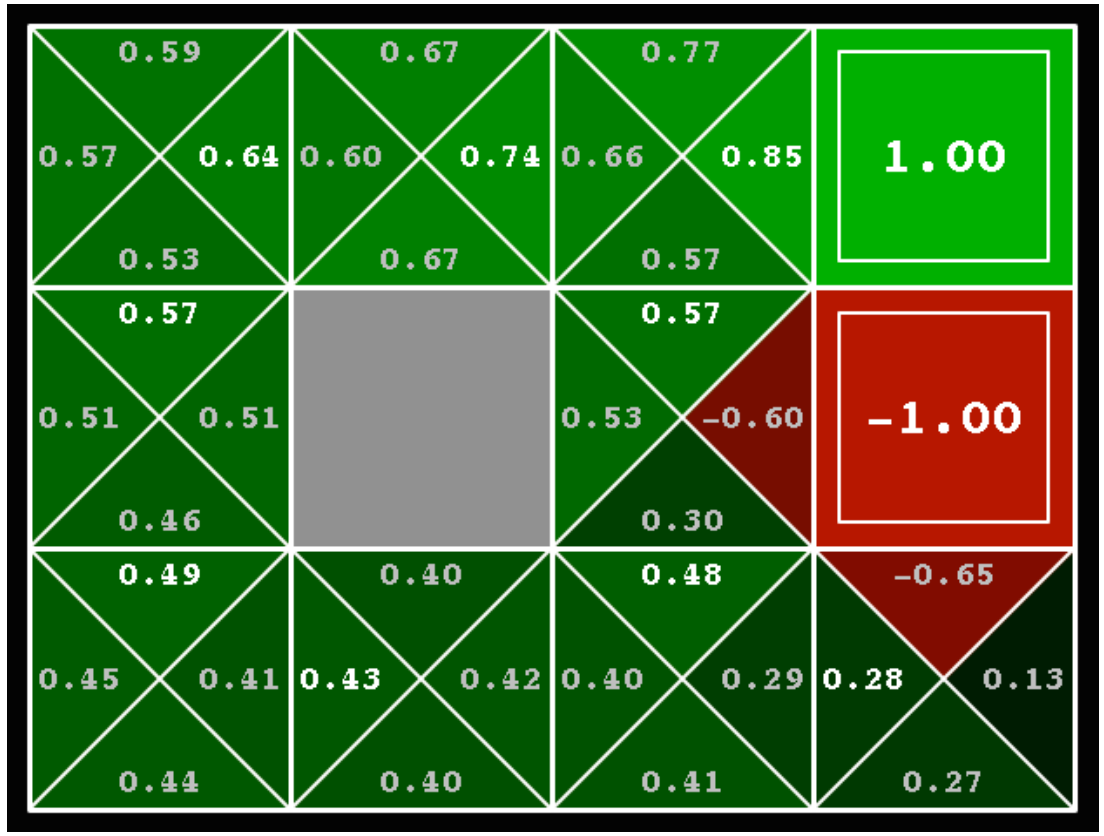
$$U^*(s) = \max_a Q^*(s, a)$$



Grid World Values

EXAMPLE





U*(s)

Q*(s,a)

Bellman Equations

DEFINITION

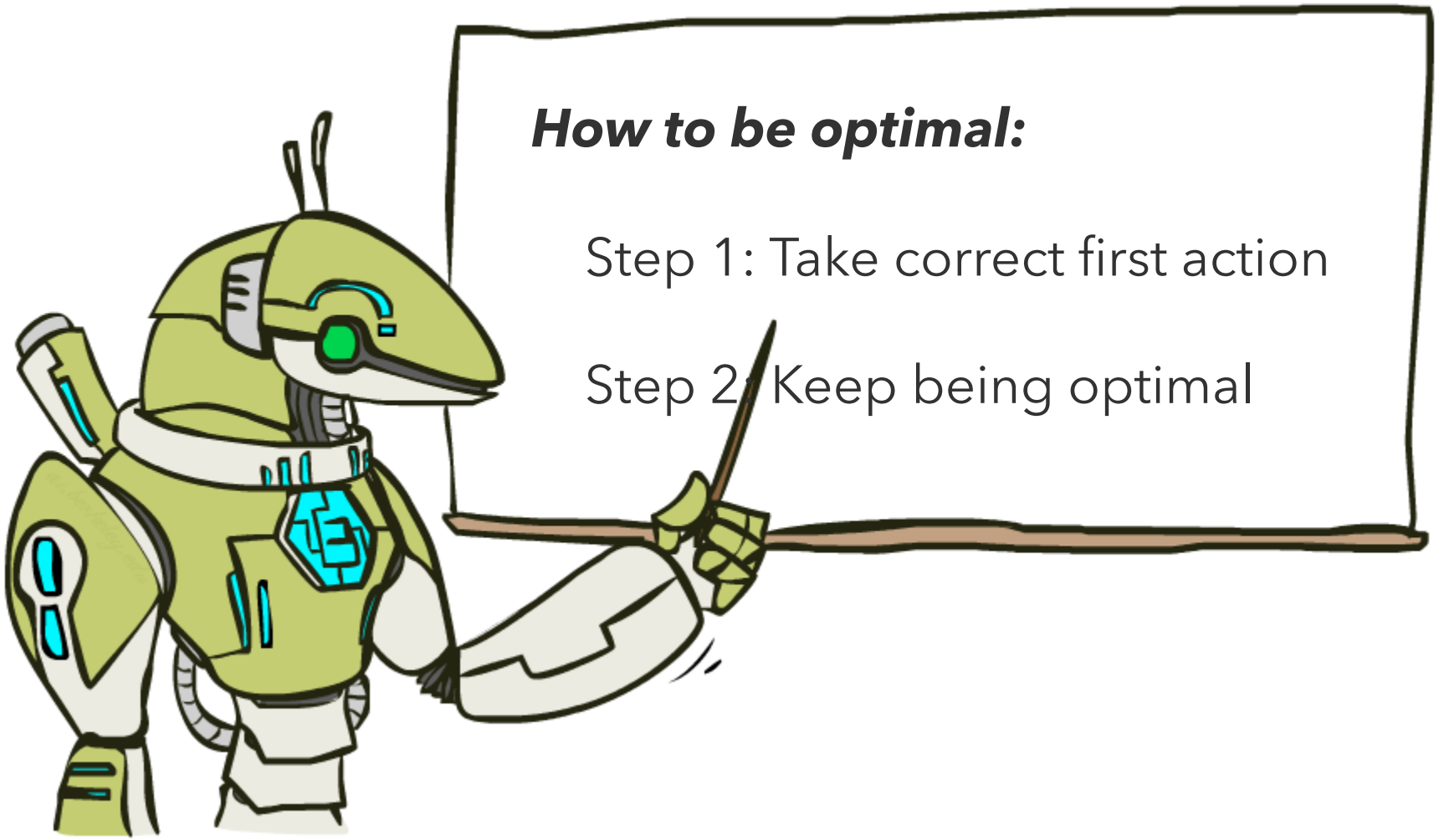
The utility of a state is the optimal reward for the next transition plus the utility of the next state

- Recursive definition of utility (Bellman Equation)

$$U(s) = \max_a \left[\sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma U(s') \right] \right] = \max_a Q(s,a)$$

- Recursive definition of Q-value

$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$



It's Pretty Simple…

QUESTIONS

Questions?

live and on sli.do #cse473

that's it for today!

SUMMARY

- MDPs are fully observable stochastic search problems
- Discounting prevents infinite rewards
- MDPs are expectimax search trees with states and Q-states

UPCOMING

- Value Iteration
- Policy Iteration

REMINDERS

- Complete **Practice Problem 6**
- Do **Project 1** (due 7.9)