

Markov Decision Processes

II



CSE 473: Introduction to Artificial Intelligence



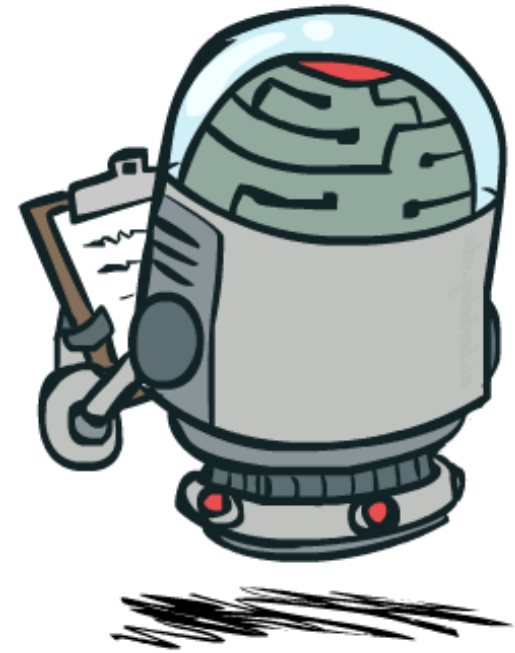
Administrivia

ANNOUNCEMENTS

- **Test 1** on Friday
 - Bring pencil & Husky ID
 - One double-sided 8.5x11" note sheet allowed

ASSIGNMENTS

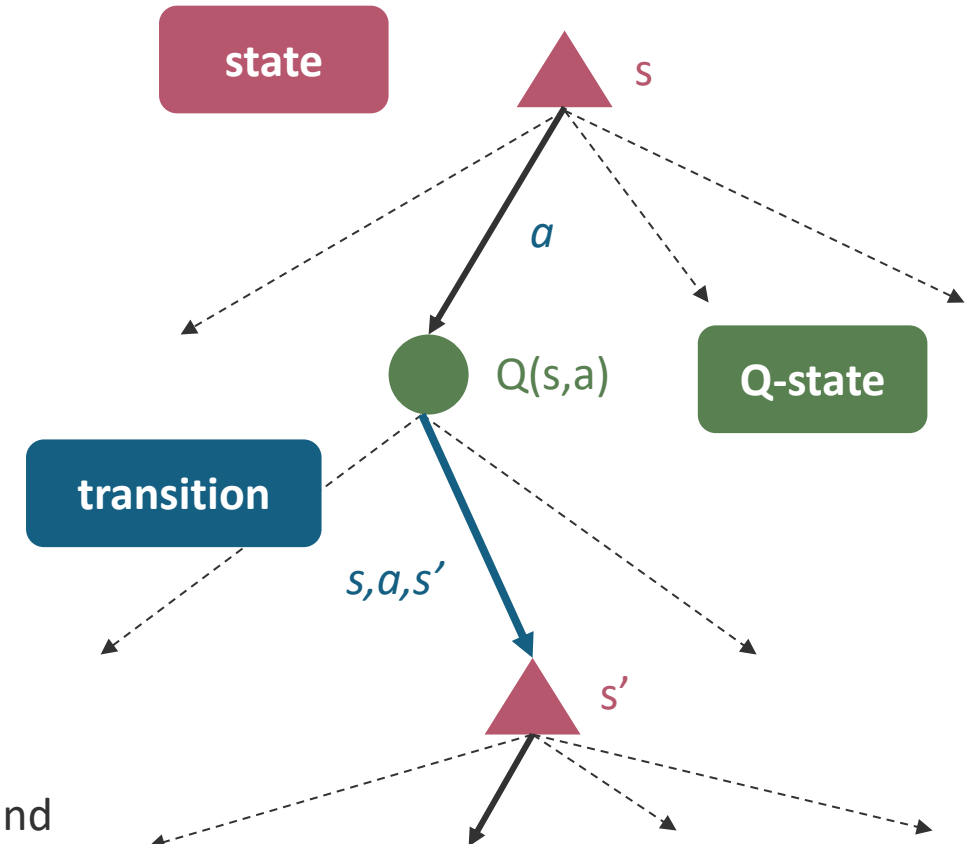
- **Project 1** due tonight
- **Homework 2** due 10.30



Recap: Optimal Quantities

FOUNDATIONS

- Expected utility (value) of policy π at s_0 :
 - $U^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S'_t)]$
- Optimal Policy π^*
 - $\pi^* = \operatorname{argmax}_{\pi} U^\pi(s)$
- Value of a State $U^*(s)$
 - 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
 - $U^*(s) = \max_a Q^*(s, a)$



Recap: Bellman Equations

DEFINITION

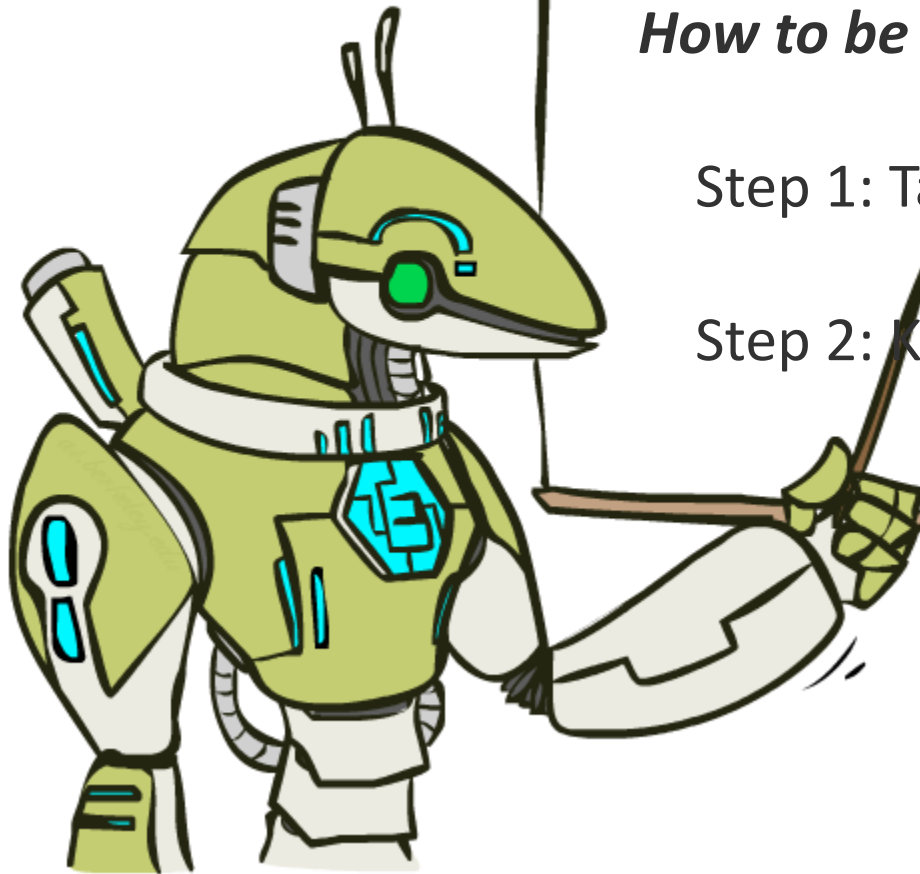
The utility of a state is the optimal reward for the next transition plus the utility of the next state

- Recursive definition of utility (Bellman Equation)

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

- Recursive definition of Q-value

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



How to be optimal:

Step 1: Take correct first action

Step 2: Keep being optimal

It's Pretty Simple...

How to find $U^*(s)$?

Value Iteration

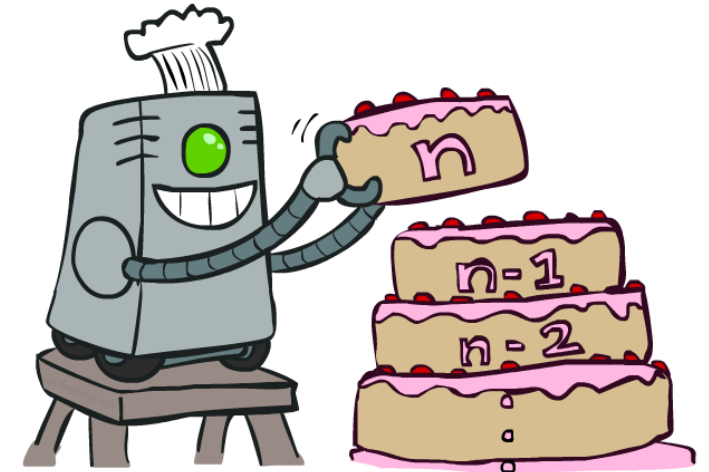
FOUNDATIONS

Iteratively update $U(s)$ values, stop using termination parameter ϵ

- Start with $U_0(s) = 0$
- Repeat until convergence (updates $< \epsilon$):
 - Perform an update using Bellman equation:

$$U_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U_k(s')]$$

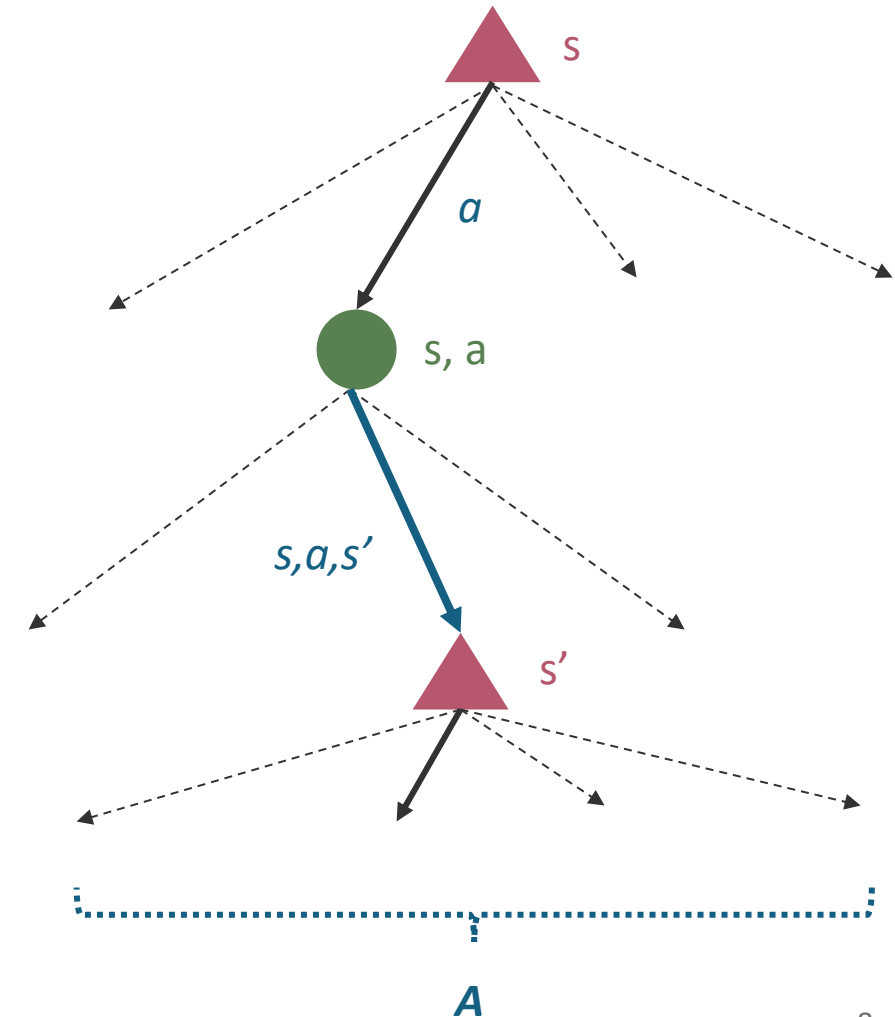
- **Theorem:** Value iteration will converge to optima
 - Approximations get refined towards optimal values
 - Policy may converge before values do



Properties of Value Iteration

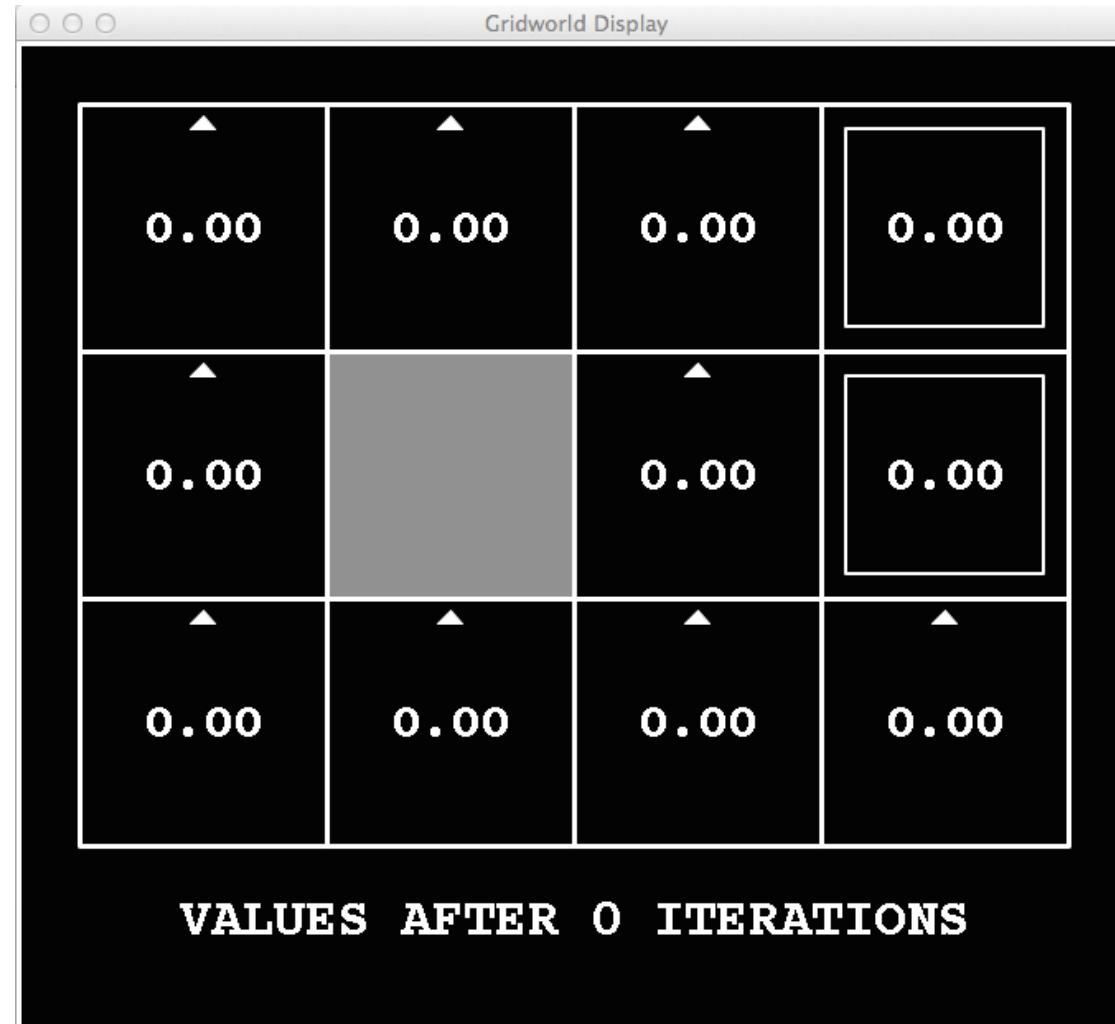
DEFINITION

- Bellman equations *characterize* optimal values
 - $U^*(s) = \max_a [\sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U^*(s')]]$
- Value iteration *computes* optimal values
 - $U_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U_k(s')]$
 - Value iteration is a fixed point solution method
- Complexity
 - Runtime per iteration: $O(S^2A)$



Grid World Value Iteration ($k = 0$)

EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 1$)

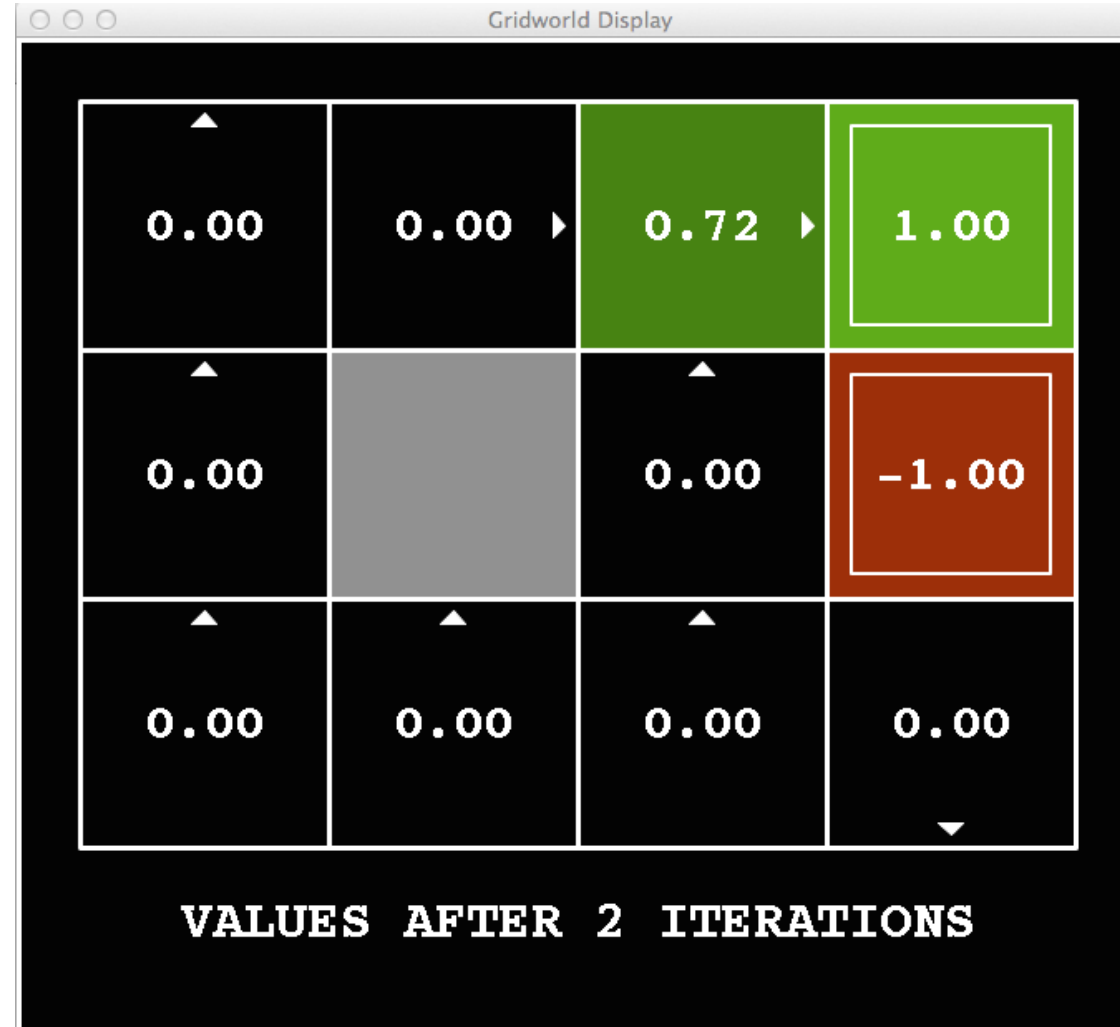
EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 2$)

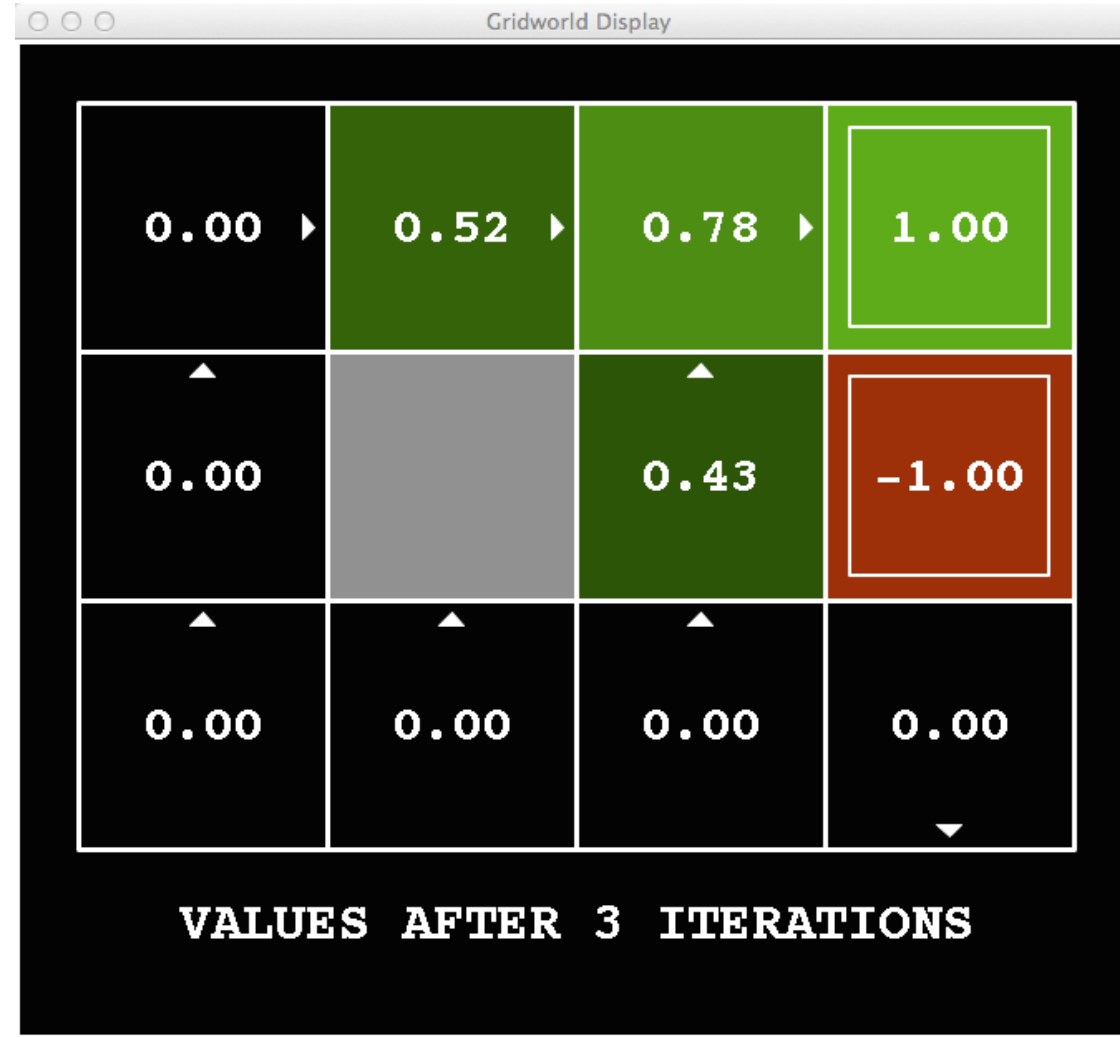
EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 3$)

EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 4$)

EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 5$)

EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 6$)

EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 7$)

EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 8$)

EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 9$)

EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 10$)

EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Grid World Value Iteration ($k = 100$)

EXAMPLE



Noise = 0.2
Discount = 0.9
Living Reward = 0

Questions?

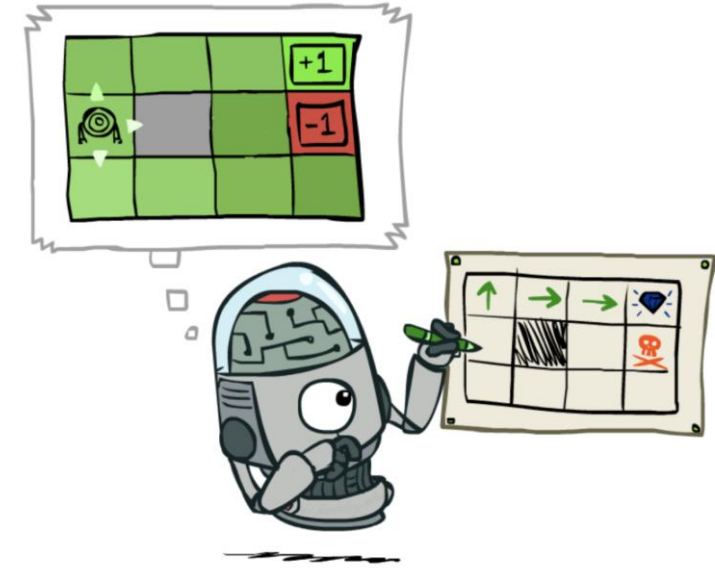
How to find policy from values?

Policy Extraction

FOUNDATIONS

W

- How should the agent act given $U(s)$?
 - Maximize expected utility!



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

- This is **policy extraction**, since it finds the policy π_U implied by U

Problems with Value Iteration

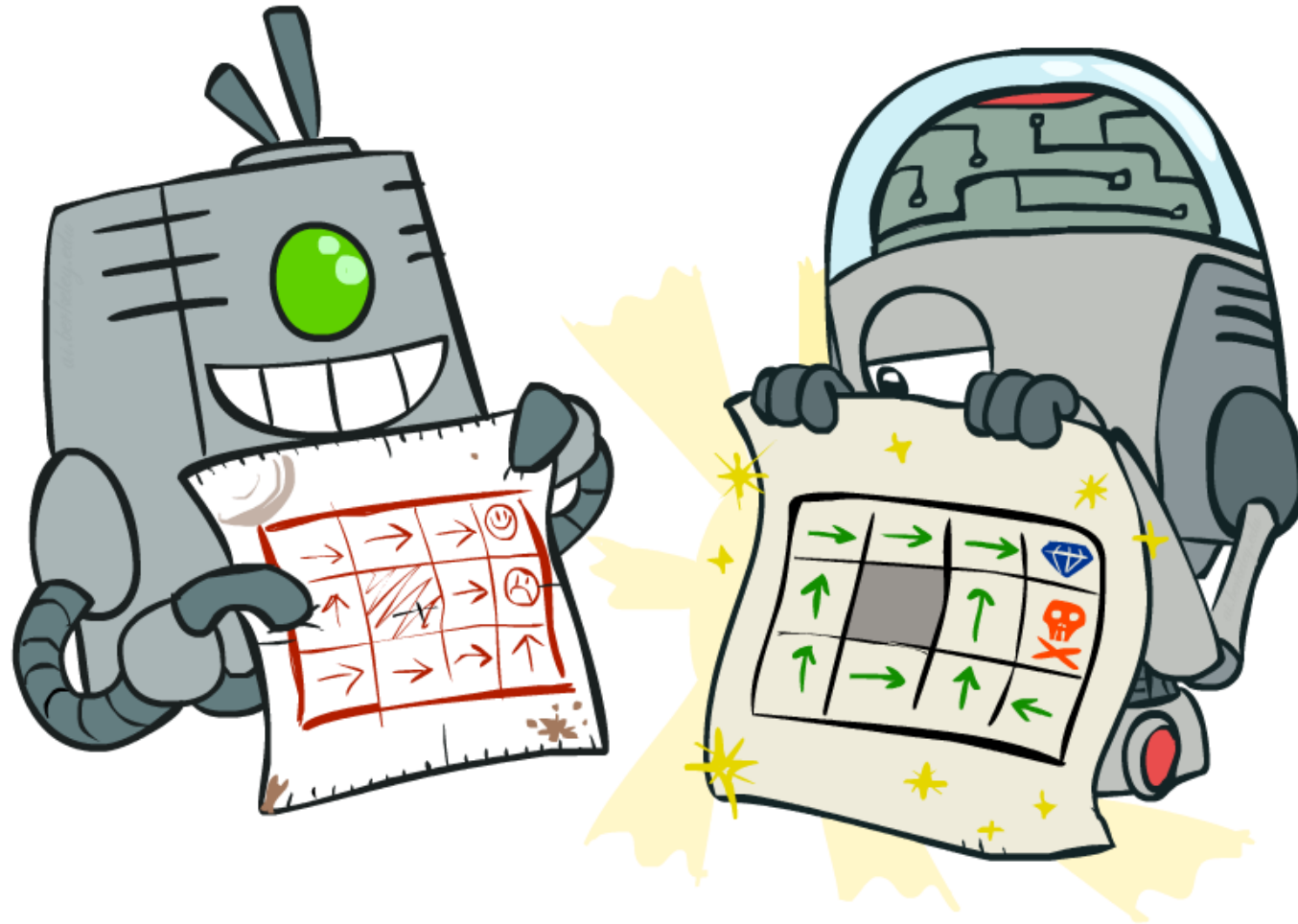
CONTEXT

Value iteration repeats Bellman updates until convergence

$$U_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U_k(s')]$$

- **Problem 1:** It's slow — $O(S^2A)$ per iteration
- **Problem 2:** argmax at each state barely changes
- **Problem 3:** Policy often converges long before values

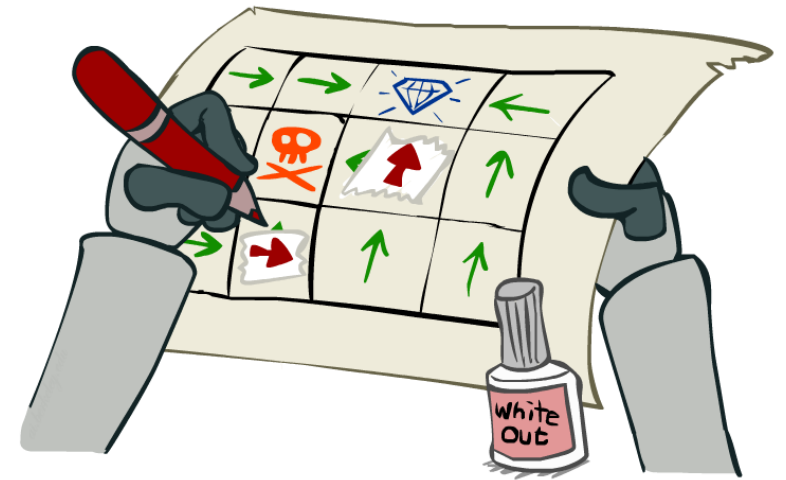
Questions?



A Better Approach...

Policy Iteration

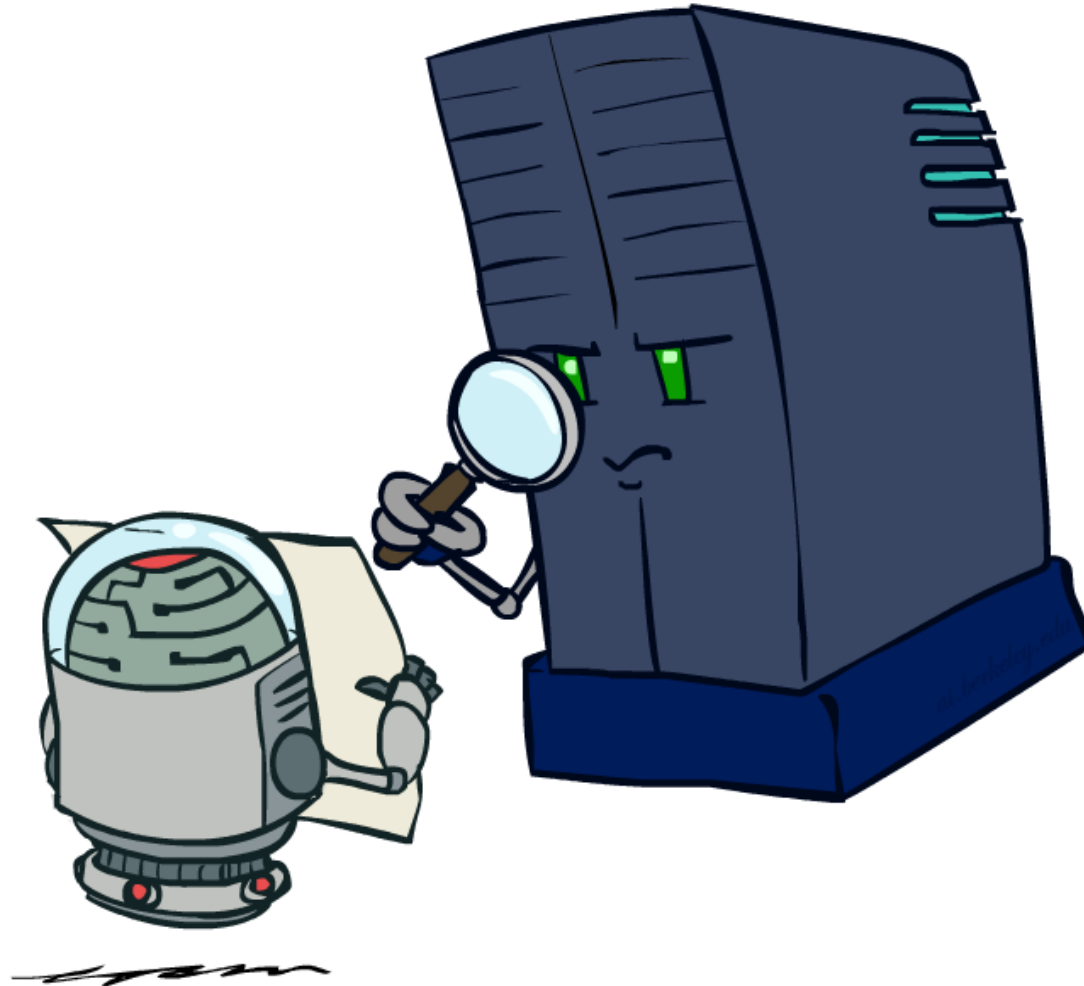
- Idea:
 - Make the policy implied by U explicit and compute long-term implications for values
- Repeat until no change in policy:
 - Step 1: **Policy Evaluation**
calculate value U^{π_k} from current policy π_k
 - Step 2: **Policy Improvement**
extract new implied policy π_{k+1} from U^{π_k}
- Properties:
 - Still optimal!
 - Can converge (much) faster under some conditions



How to Evaluate Policies?



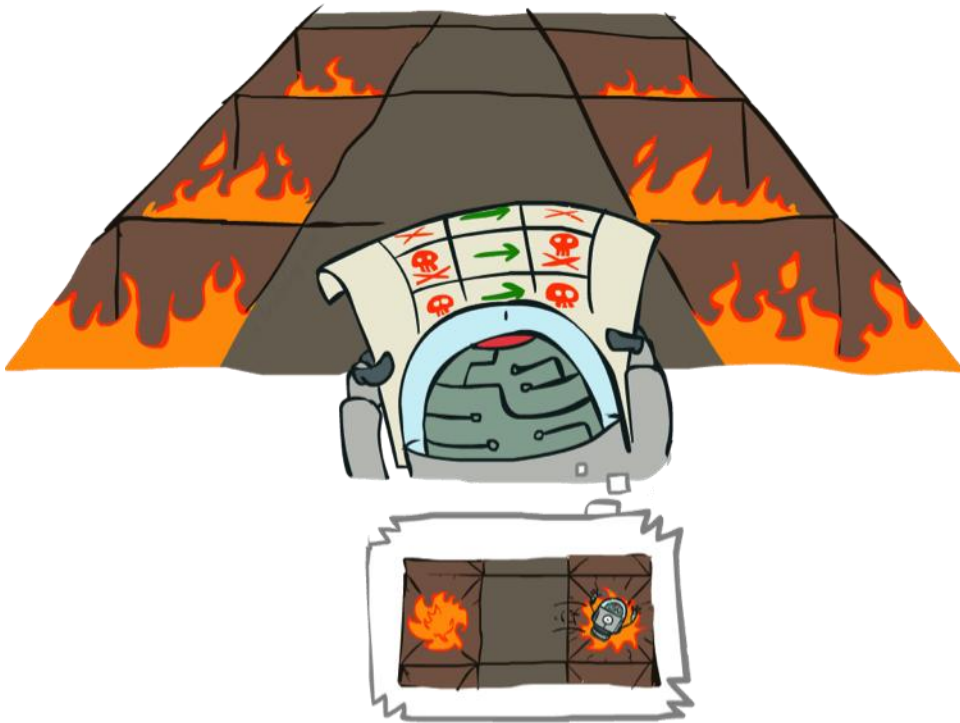
CONTEXT



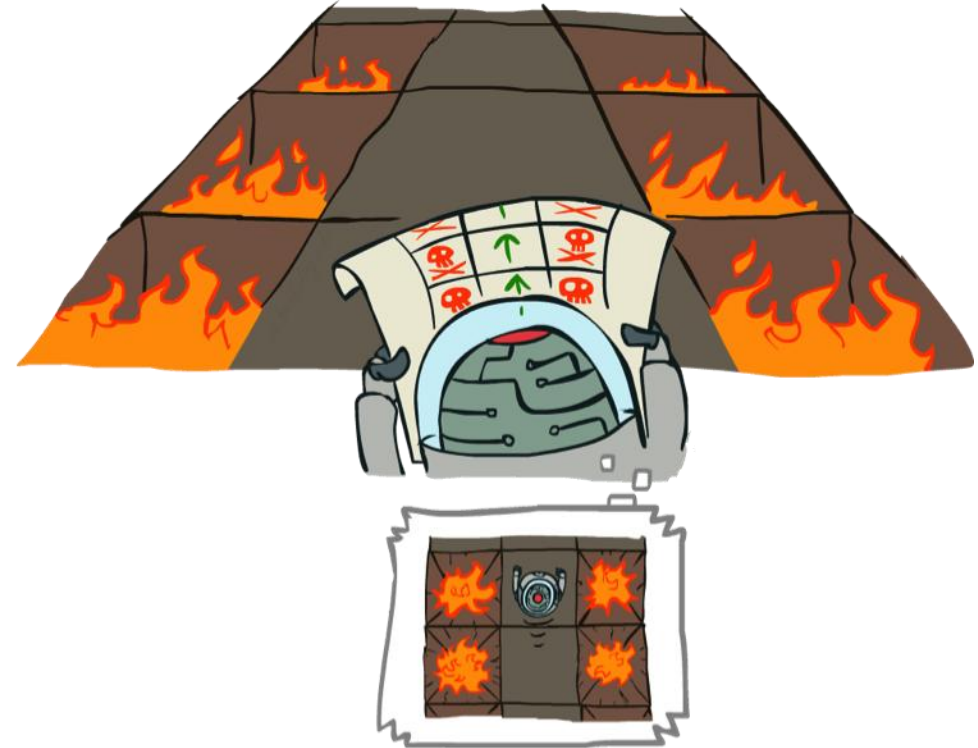
Evaluating Policies

EXAMPLE

π : always go right



π : always go straight



Evaluating Policies from Utilities

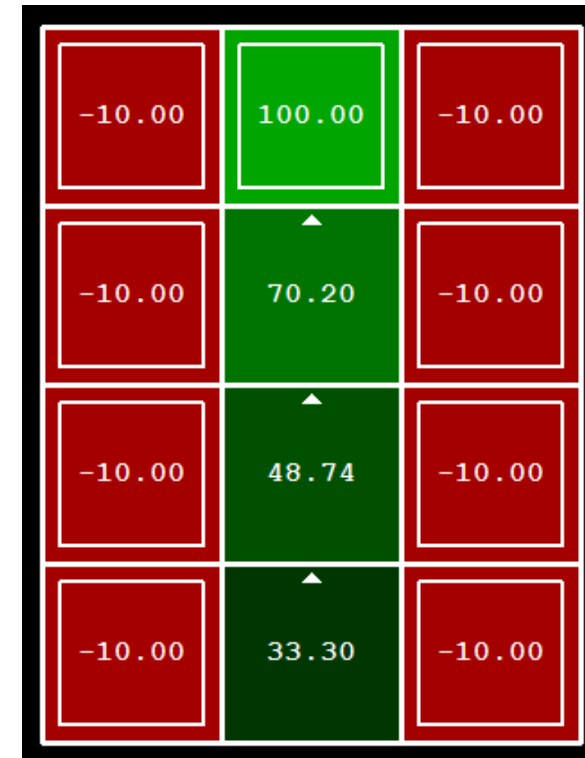


EXAMPLE

π : always go right



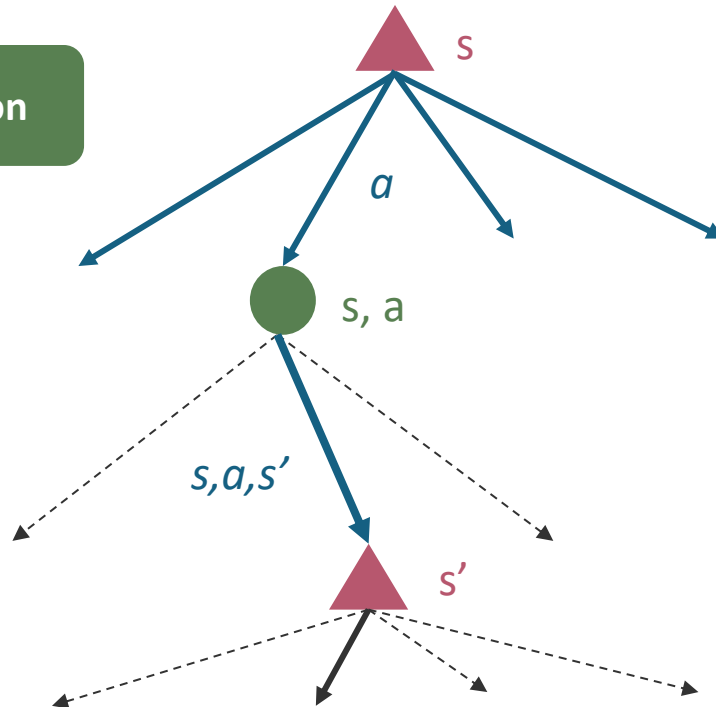
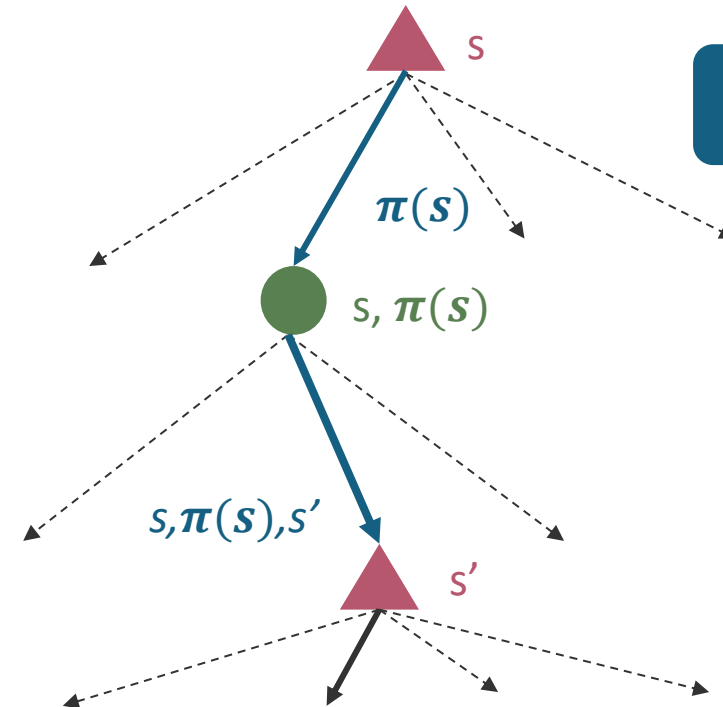
π : always go straight



Fixed Policies

CONTEXT

Optimal Action

Fixed Policy π 

- Expectimax over all actions is complicated
- If we fixed some policy $\pi(s)$ per state, the tree would be much simpler...

Fixed Policy Utilities

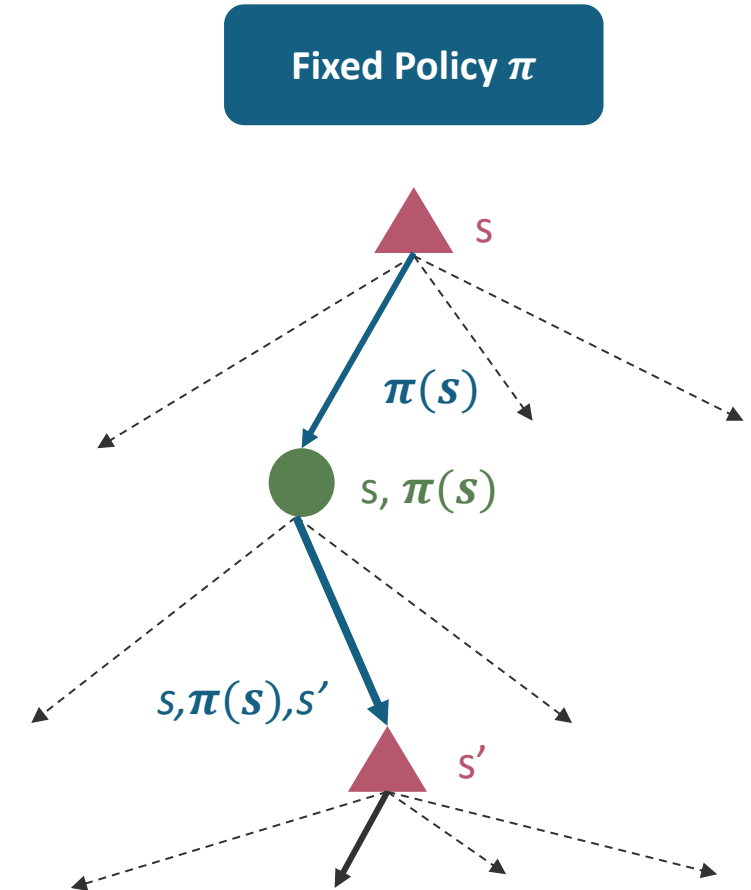
CONTEXT

- Define the utility of state s under fixed policy π
 - $U^\pi(s)$ is the expected total utility starting at s and following π
- Recursive relationship:

$$U_{i+1}(s) \leftarrow \sum_{s'} P(s'|s, \pi(s)) [R(s, \pi(s), s') + \gamma U_i(s')]$$

- Approximation of system of linear equations:

$$U_i(s) = \sum_{s'} P(s'|s, \pi(s)) [R(s, \pi(s), s') + \gamma U_i(s')]$$



Policy Evaluation

FOUNDATIONS

Calculating U^π for fixed policy π :

- **Option 1:** Recursive Bellman Updates

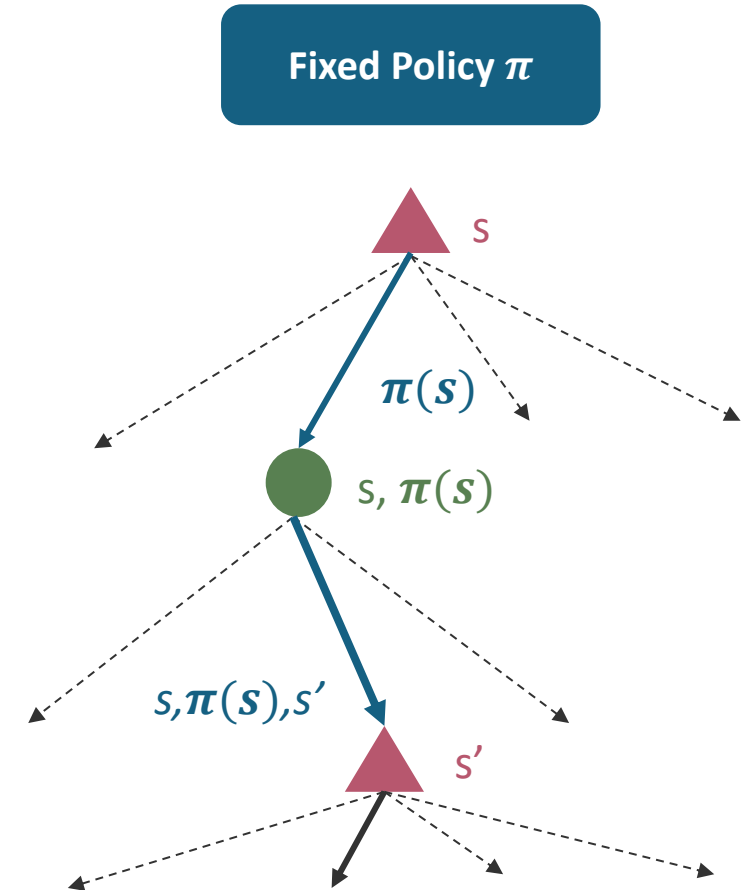
$$U_0^\pi(s) = 0$$

$$U_{i+1}^\pi(s) \leftarrow \sum_{s'} P(s'|s, \pi(s)) [R(s, \pi(s), s') + \gamma U_i(s')]$$

Efficiency: $O(s^2)$ per iteration

- **Option 2:** Solve the Linear System of Equations

$$U_i(s) = \sum_{s'} P(s'|s, \pi(s)) [R(s, \pi(s), s') + \gamma U_i(s')]$$



Policy Iteration Updates

FOUNDATIONS

- **Policy Evaluation**

- Iterate until values converge:

$$U_{i+1}(s) \leftarrow \sum_{s'} P(s'|s, \pi_i(s)) [R(s, \pi_i(s), s') + \gamma U_i(s)]$$

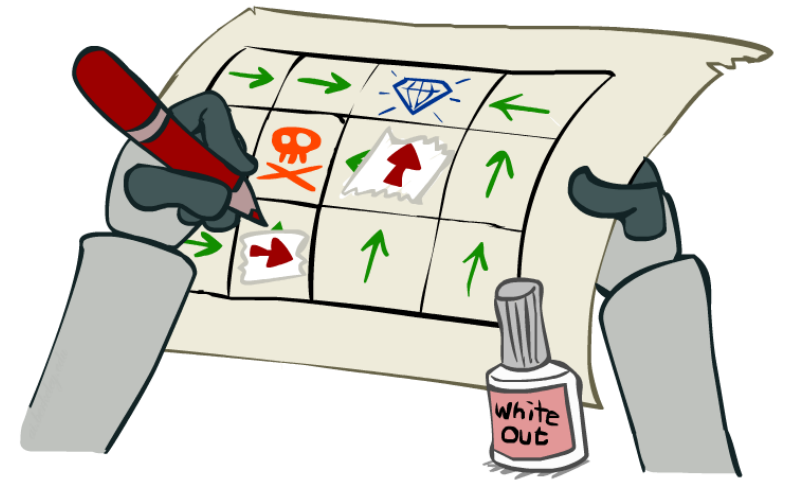
- Quicker: $O(S^2)$

- **Policy Improvement**

- One-step lookahead:

$$\pi_{i+1}(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U^{\pi_i}(s')]$$

- Slower: $O(S^2A)$



Questions?

Comparison: Value vs. Policy Iteration

CONTEXT

Value Iteration

$$U_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U_k(s')]$$

- Every iteration updates the utilities and (implicitly) the policy
- Doesn't track the policy, but taking max over actions implicitly re-computes it
- Therefore, value iteration is expensive

Policy Iteration

$$U_{i+1}(s) \leftarrow \sum_{s'} P(s'|s, \pi_i(s)) [R(s, \pi_i(s), s') + \gamma U_i(s)]$$

$$\pi_{i+1}(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U^{\pi_i}(s')]$$

- Update the utilities using a fixed policy, saving computational time
- After policy is evaluated, update policy using new utilities

that's it for today!

SUMMARY

- Value iteration computes values by taking the max over possible actions
- Policy iteration considers a fixed policy, then improves the policy using calculated values

UPCOMING

- Reinforcement Learning: learning by doing!

REMINDERS

- Do **Project 1**