APPLIED MECHANISM DESIGN FOR SOCIAL GOOD

JOHN P DICKERSON

Lecture #10 – 02/27/2020

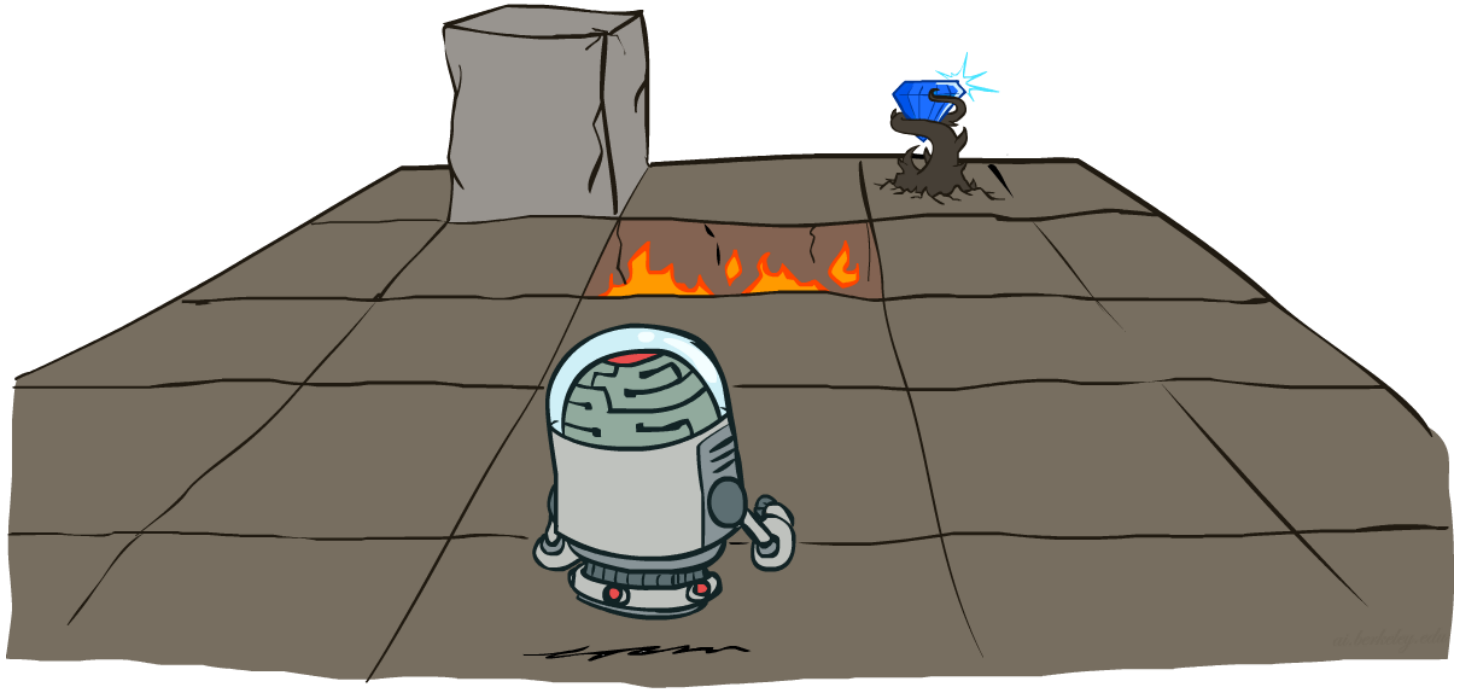
CMSC828M
Tuesdays & Thursdays
2:00pm – 3:15pm

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

ANNOUNCEMENTS

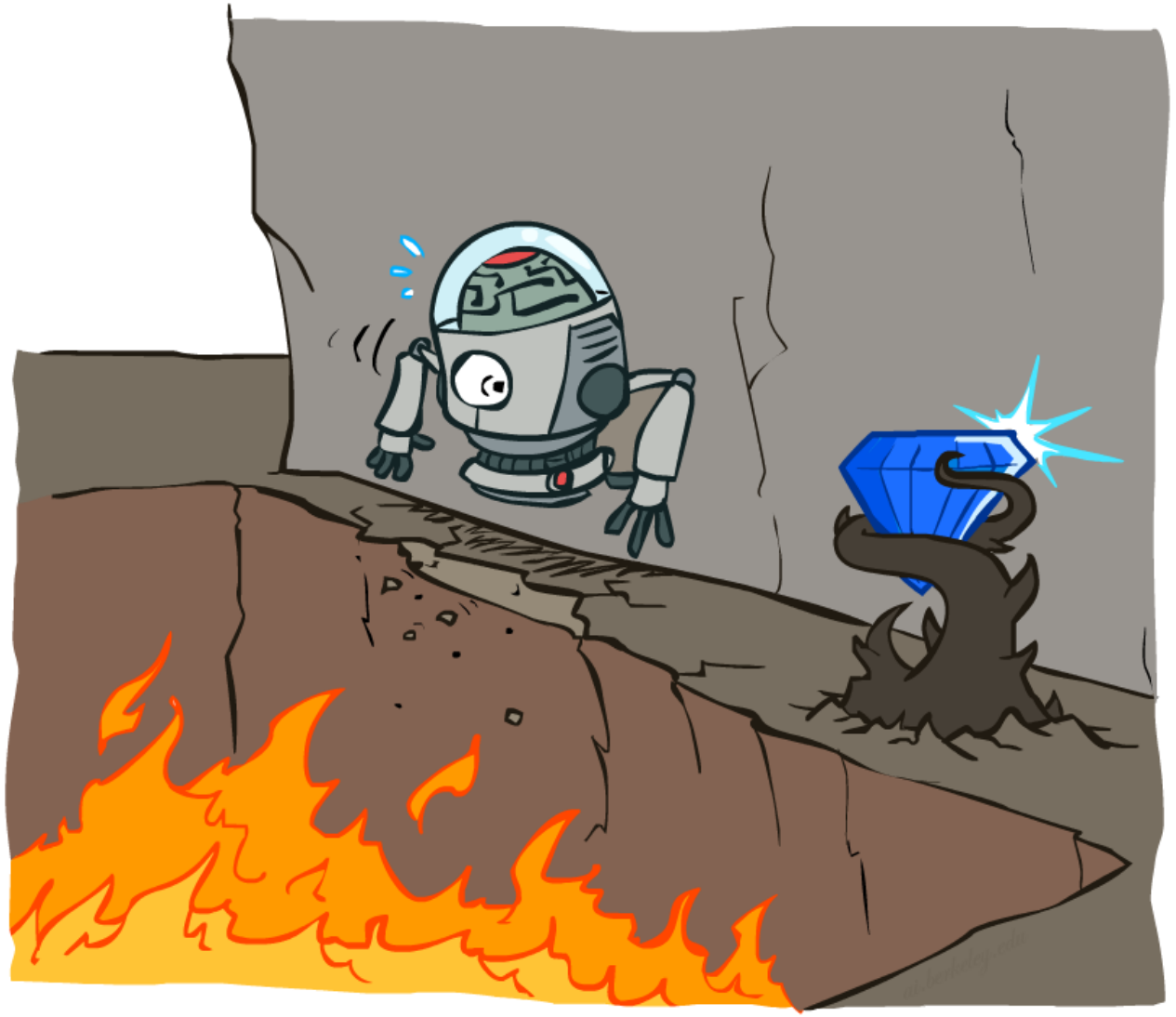
Projects?

MARKOV DECISION PROCESSES

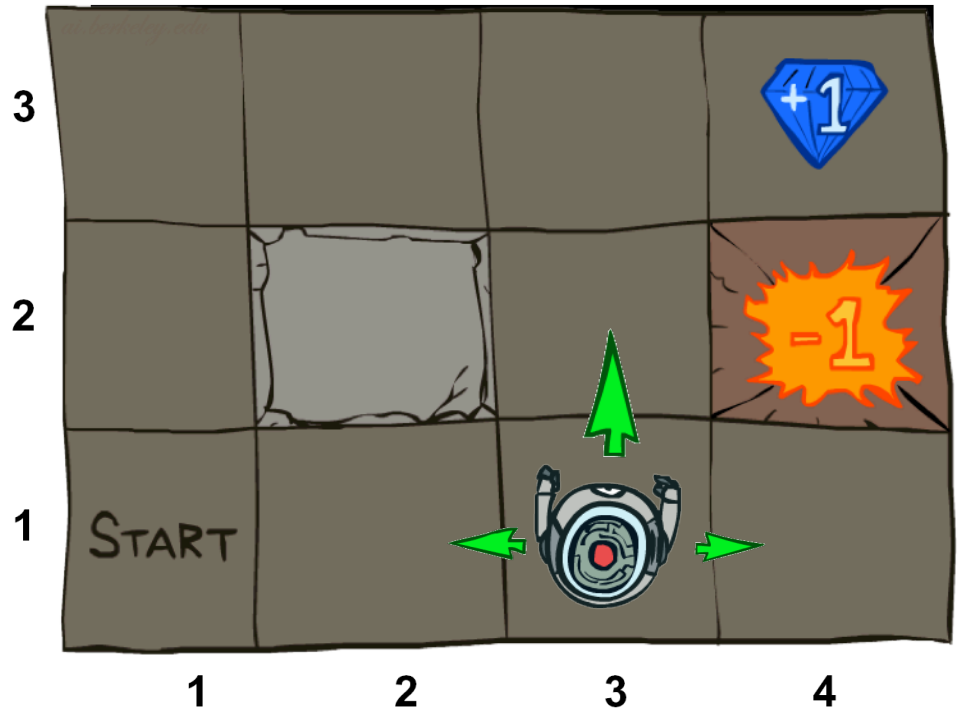


Slide credits: CMU AI and http://ai.berkeley.edu

NON-DETERMINISTIC SEARCH



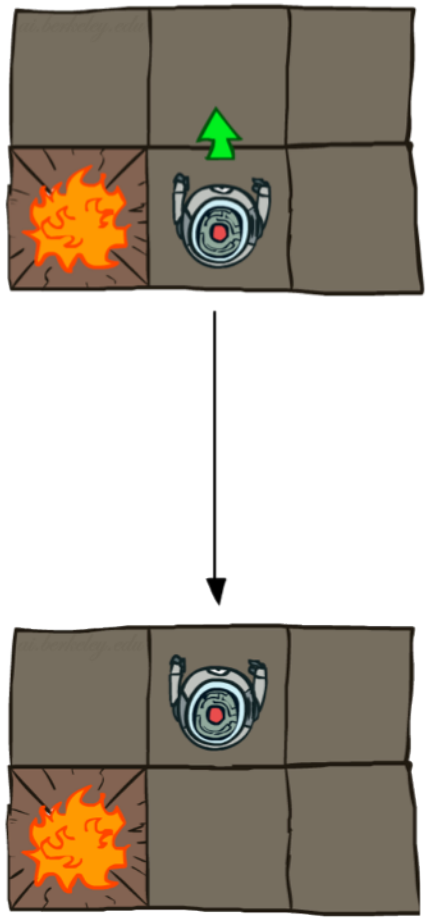
EXAMPLE: GRID WORLD



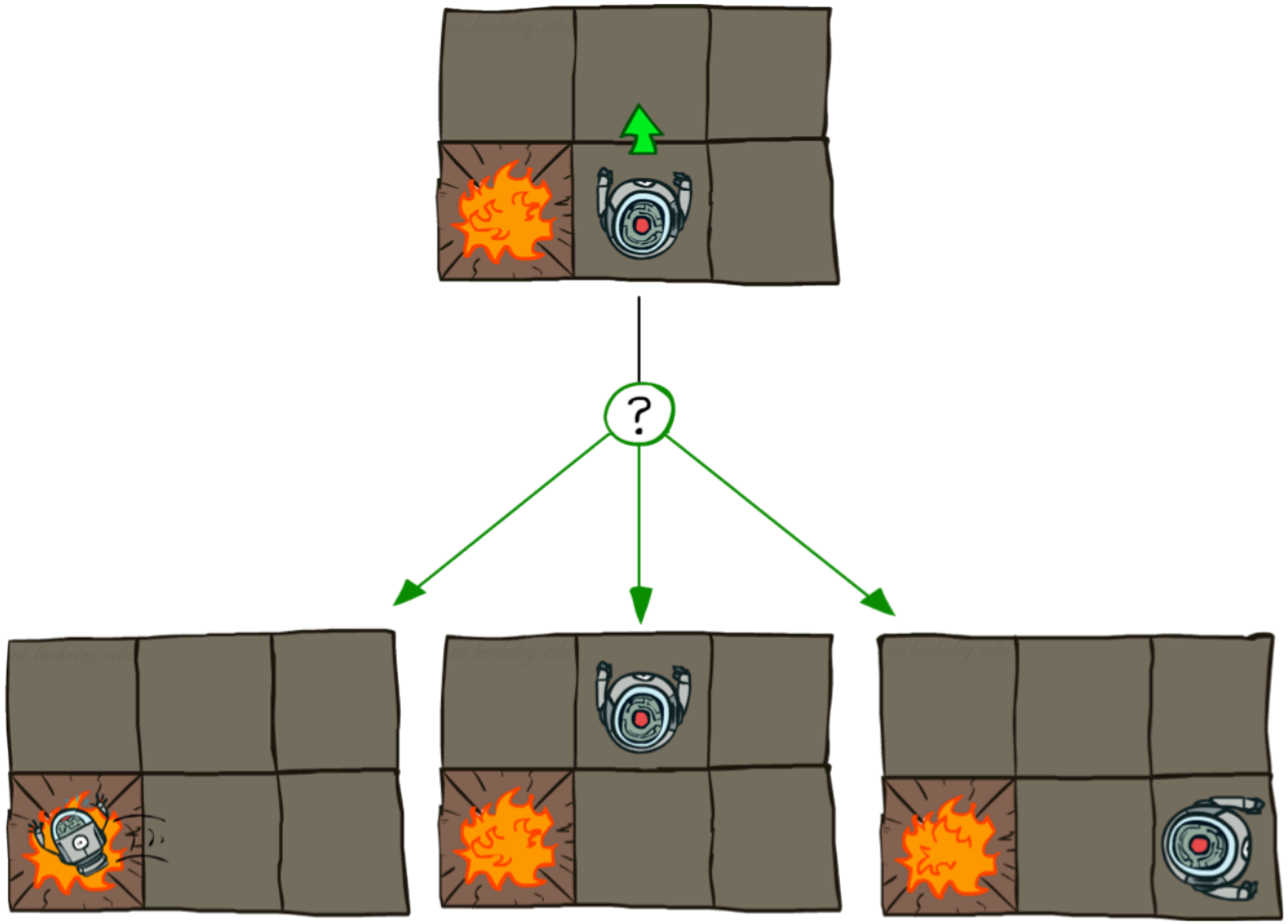
- A maze-like problem
  - The agent lives in a grid
  - Walls block the agent's path
- Noisy movement: actions do not always go as planned
  - If agent takes action North
    - 80% of the time: Get to the cell on the North (if there is no wall there)
    - 10%: West; 10%: East
  - If path after roll dice blocked by wall, stays put
- The agent receives rewards each time step
  - "Living" reward (can be negative)
  - Additional reward at pit or target (good or bad) and will exit the grid world afterward
- Goal: maximize sum of rewards

GRID WORLD ACTIONS

Deterministic Grid World



Stochastic Grid World

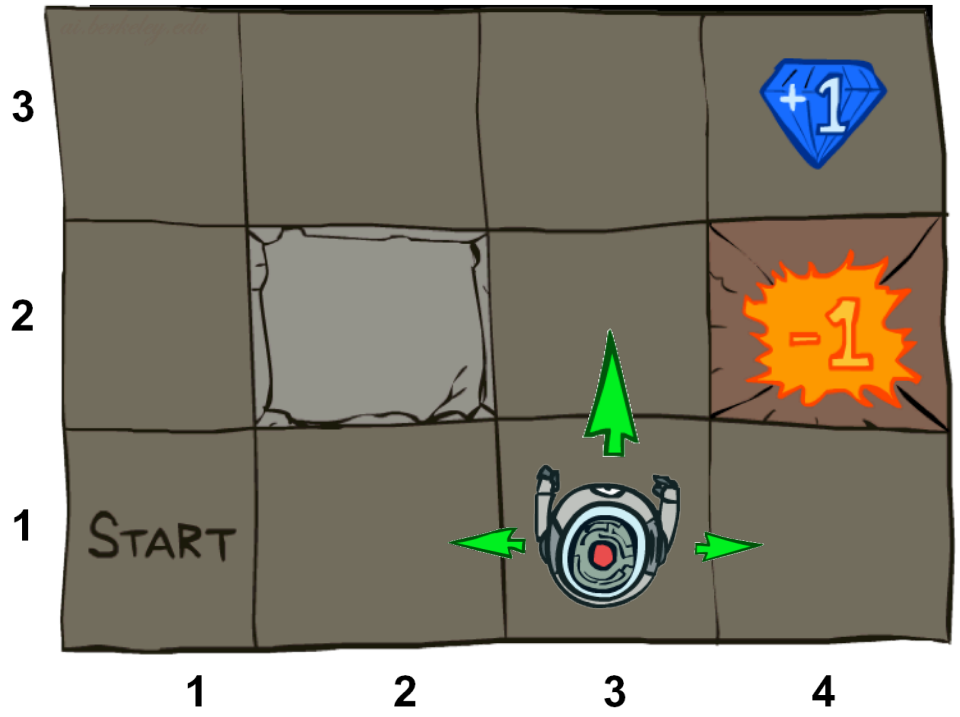


MARKOV DECISION PROCESS (MDP)

An MDP is defined by a tuple (S,A,T,R):

- S: a set of states
- A: a set of actions
- T: a transition function
  - T(s, a, s') where s ∈ S, a ∈ A, s' ∈ S is P(s'| s, a)
- R: a reward function
  - R(s, a, s') is reward at this time step
  - Sometimes just R(s) or R(s')

- Sometimes also have
  - γ: discount factor (introduced later)
  - μ: distribution of initial state (or just start state s_0)
  - Terminal states: processes end after reaching these states



The Grid World problem as an MDP

R($s_{4,2}$, $exit$, $s_{virtual\_terminal}$)=-1

R($s_{4,2}$)=-1, no virtual terminal state

How to define the terminal states & reward function for the Grid World problem?

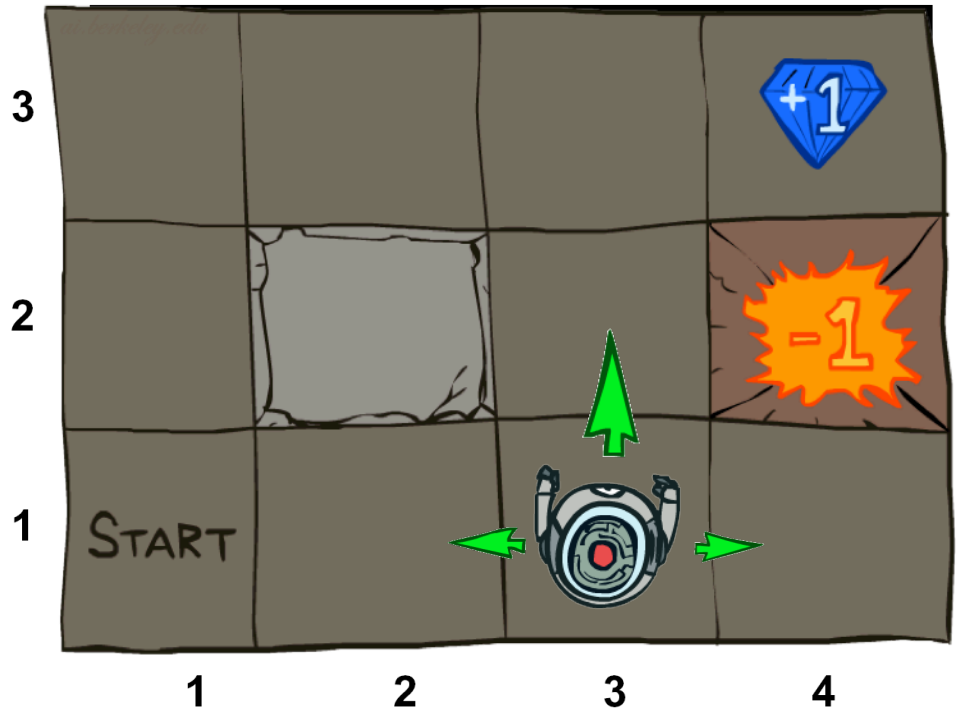
MARKOV DECISION PROCESS (MDP)

An MDP is defined by a tuple (S,A,T,R)

Why is it called Markov Decision Process?

Decision:

Process:



MARKOV DECISION PROCESS (MDP)

An MDP is defined by a tuple (S,A,T,R)

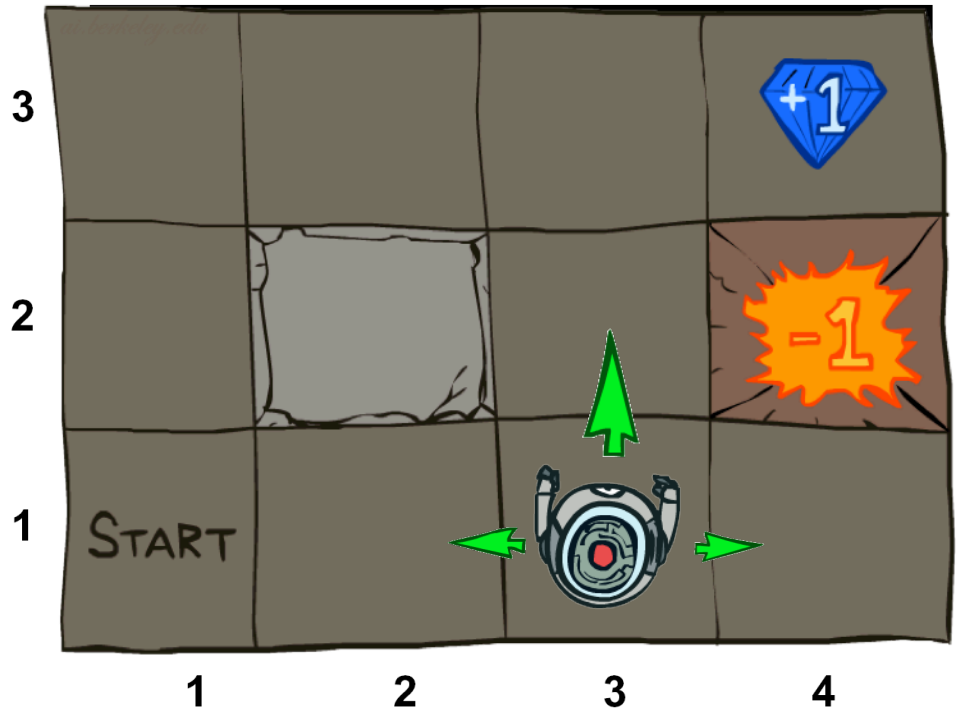
Why is it called Markov Decision Process?

Decision:

Agent decides what action to take at each time step

Process:

The system (environment + agent) is changing over time



WHAT IS "MARKOVIAN" ABOUT MDPS?

Markov property: Conditional on the present state, the future and the past are independent

With respect to MDPs, it means outcome of an action depend only on current state

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

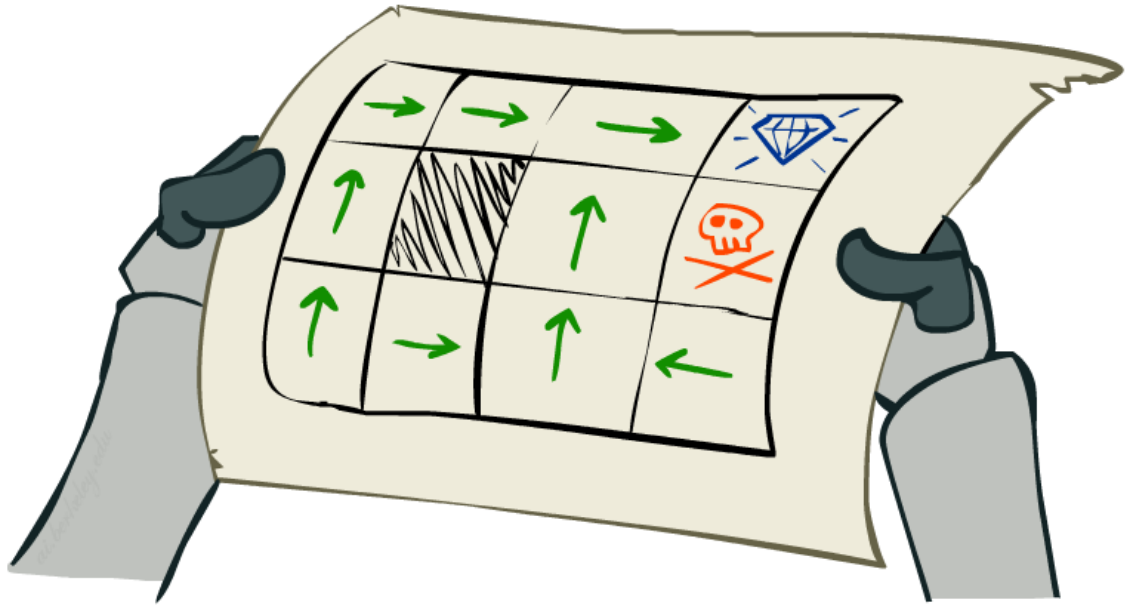
$$=$$

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

Andrey Markov (1856-1922) Russian mathematician

POLICIES

In deterministic single-agent search problems, w
sequence of actions, from start to a goal



For MDPs, we focus on policies

- Policy = map of states to actions
- $\pi(s)$ gives an action for state s

We want an optimal policy π^*: S → A

- An optimal policy is one that maximizes expected utility if followed

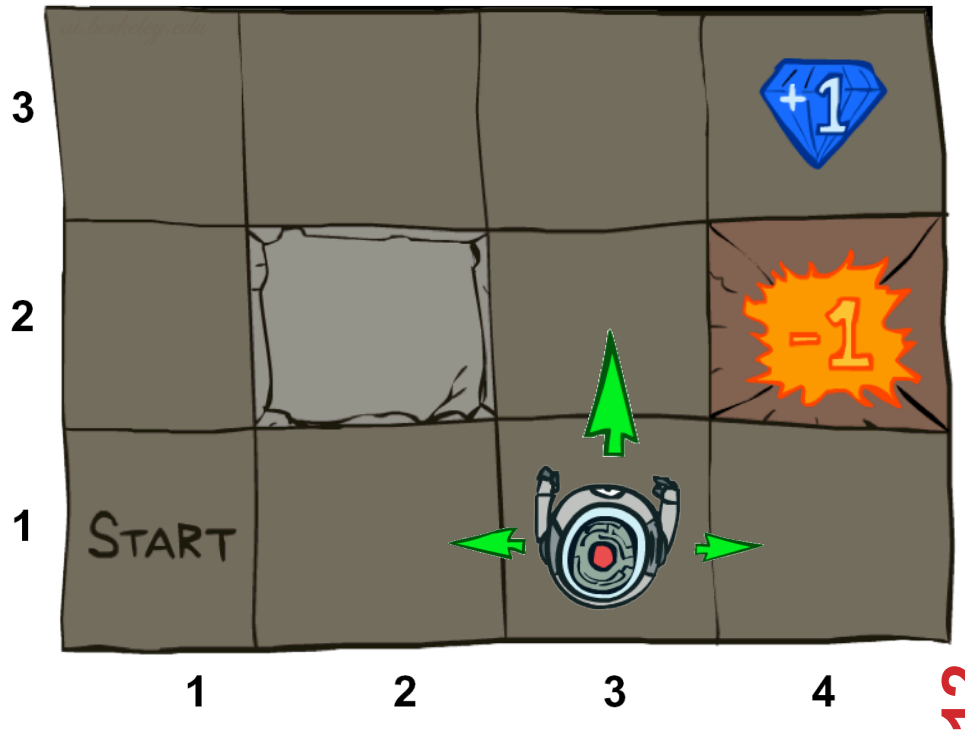
POLICIES

Recall: An MDP is defined S,A,T,R

Keep S,A,T fixed, optimal policy may vary given different R

What is the optimal policy if R(s,a,s')=-1000 for all states other than pit and target?

What is the optimal policy if R(s,a,s')=0 for all states other than pit and target, and reward=1000 and -1000 at pit and target respectively?

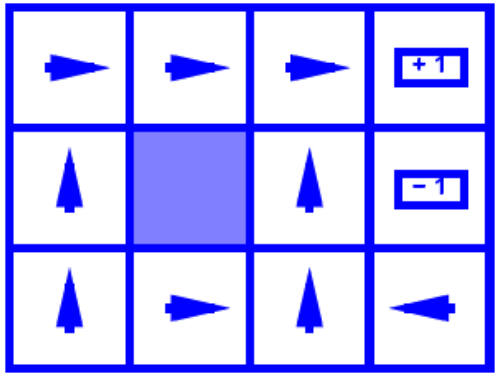


DISCUSSION POINT!

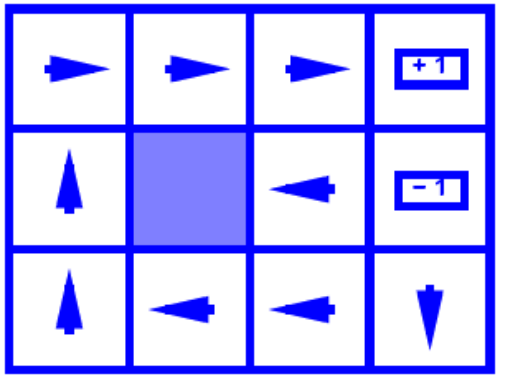
{A, B, C, D} are optimal policies for one of each of the following "reward for living" scenarios: {-0.01, -0.03, -0.04, -2.0}. Which policy maps to which reward setting?

I. {B, A, C, D}
II. {B, C, A, D}
III. {C, B, A, D}
IV. {D, A, C, B}

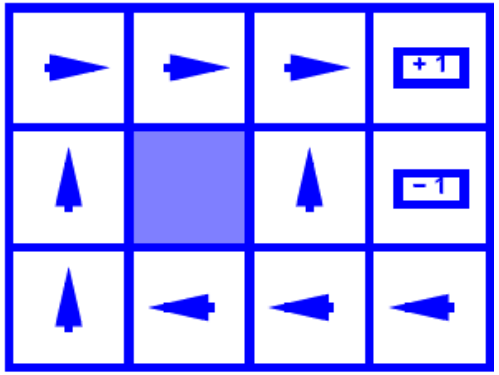
A)



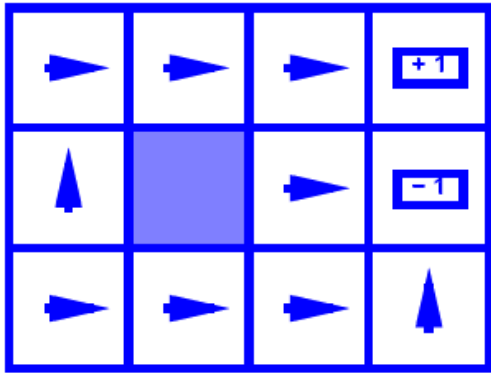
B)



C)



D)



DISCUSSION POINT!

{A, B, C, D} are optimal policies for one of each of the following "reward for living" scenarios: {-0.01, -0.03, -0.04, -2.0}. Which policy maps to which reward setting?

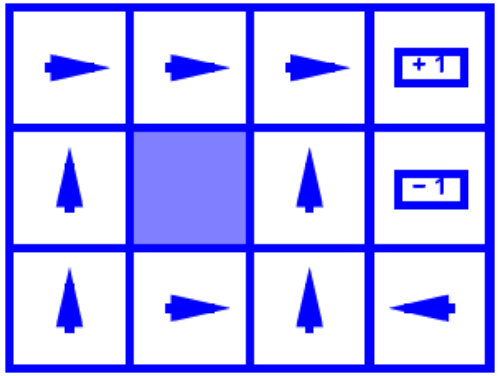
I. {B, A, C, D}

II. {B, C, A, D}

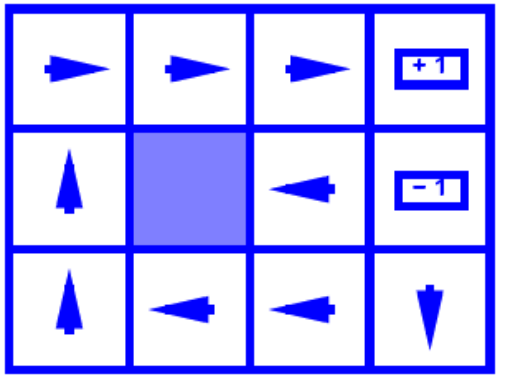
III. {C, B, A, D}

IV. {D, A, C, B}

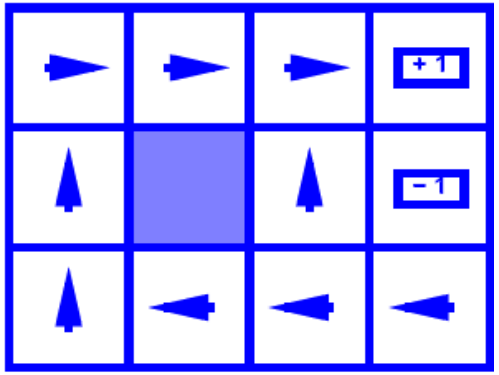
A)



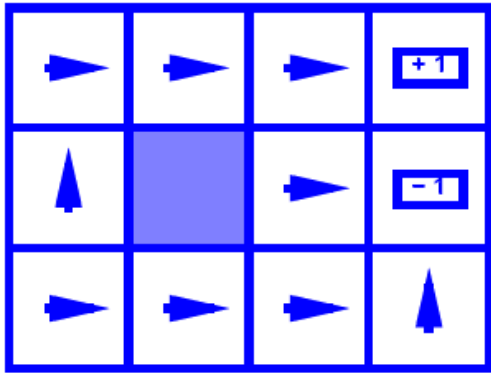
B)



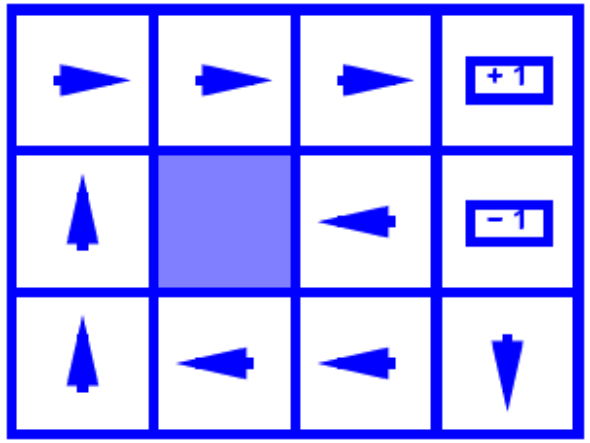
C)



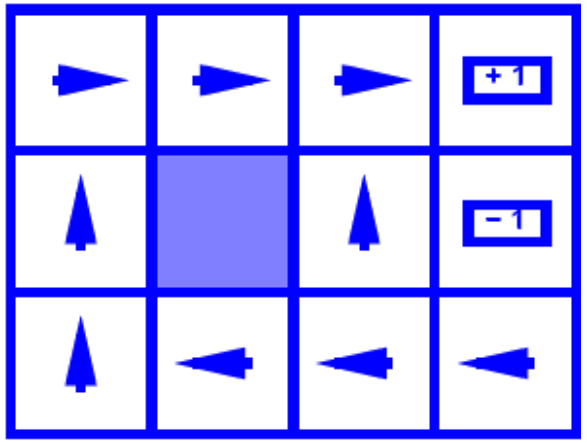
D)



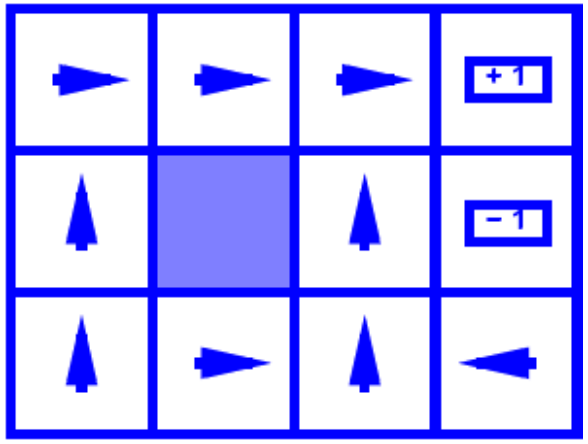
DISCUSSION POINT! POLICIES



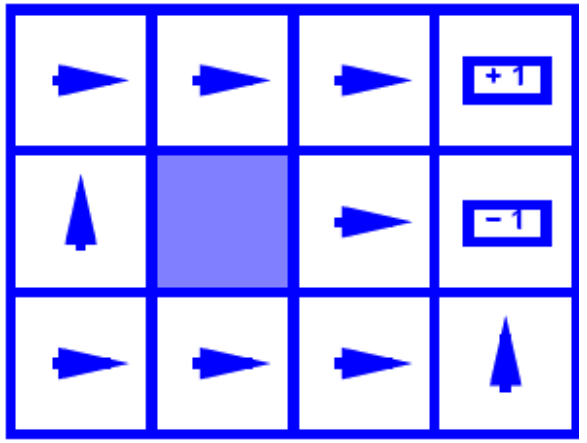
R(s) = -0.01



R(s) = -0.03

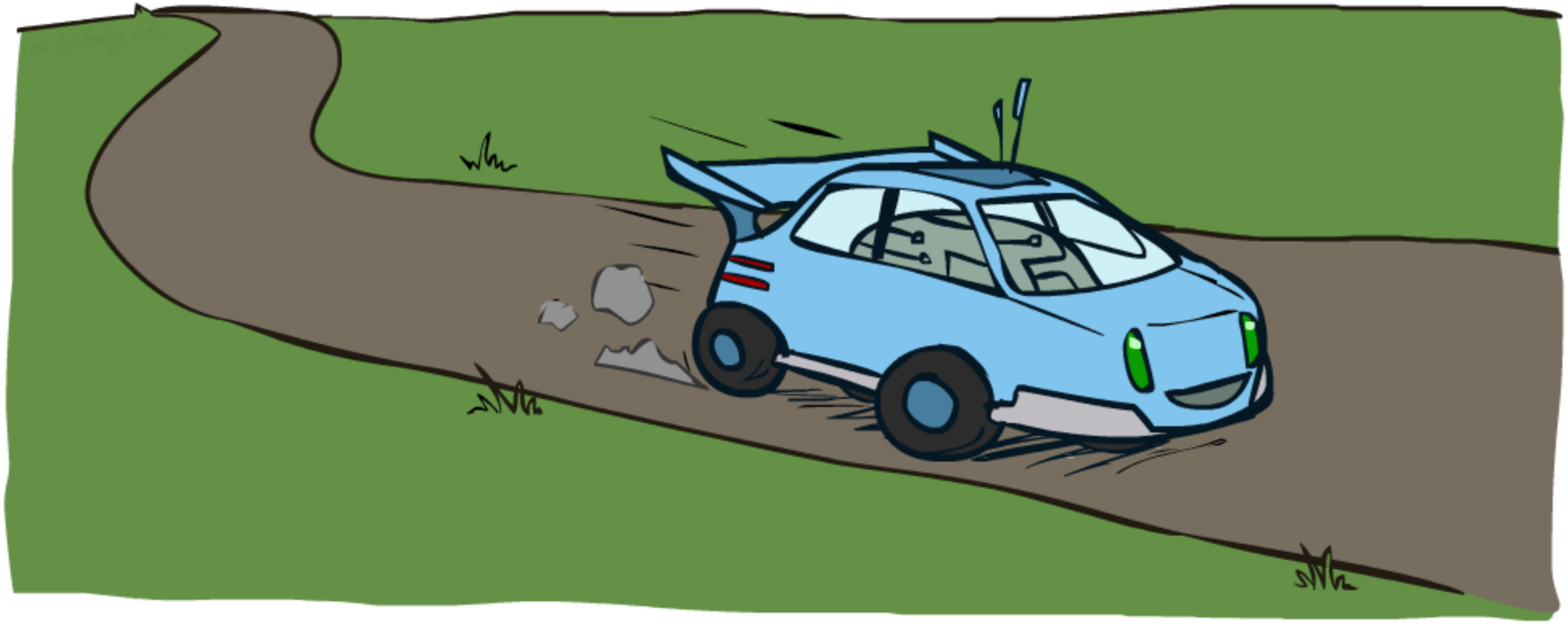


R(s) = -0.4



R(s) = -2.0

EXAMPLE: RACING



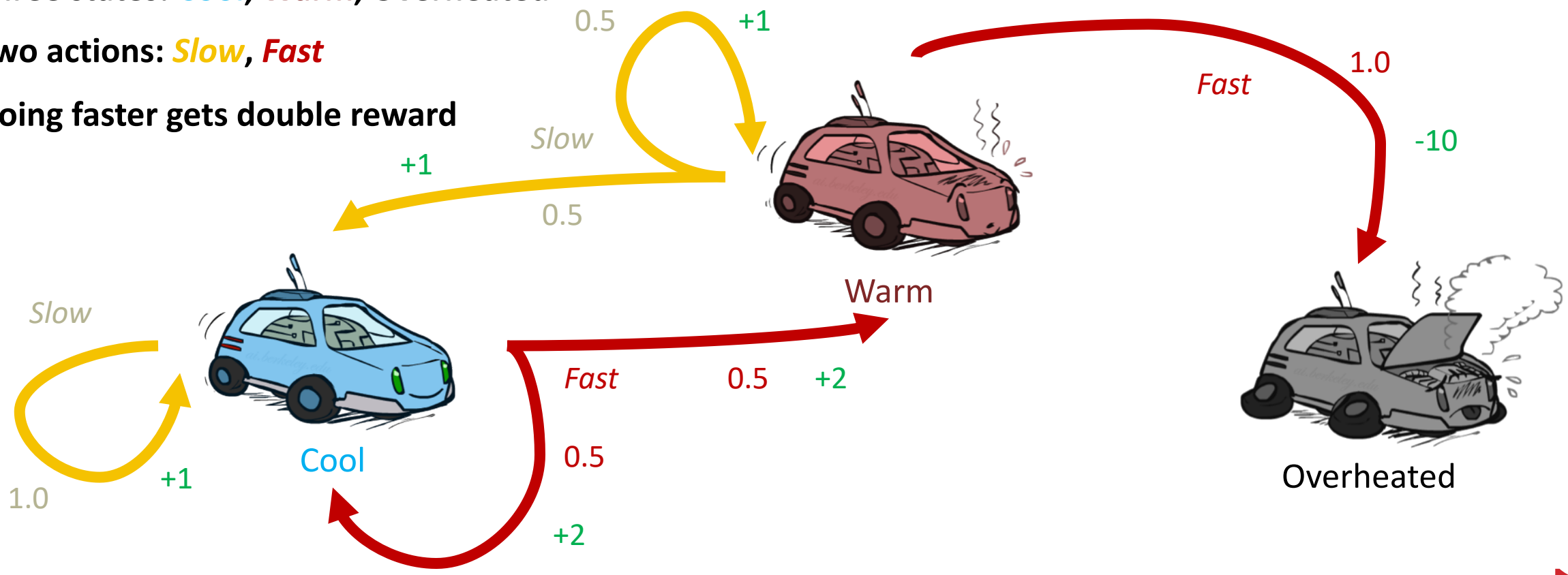
EXAMPLE: RACING

A robot car wants to travel far, quickly

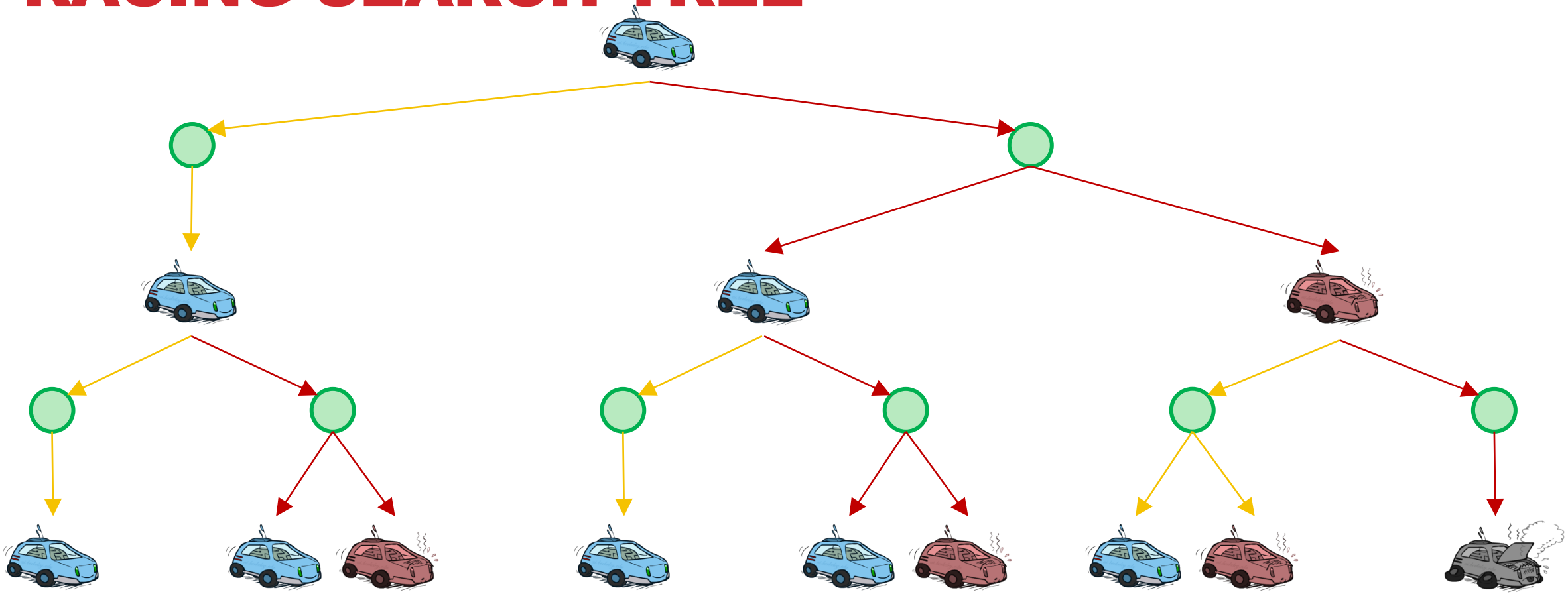
Three states: Cool, Warm, Overheated

Two actions: *Slow*, *Fast*

Going faster gets double reward

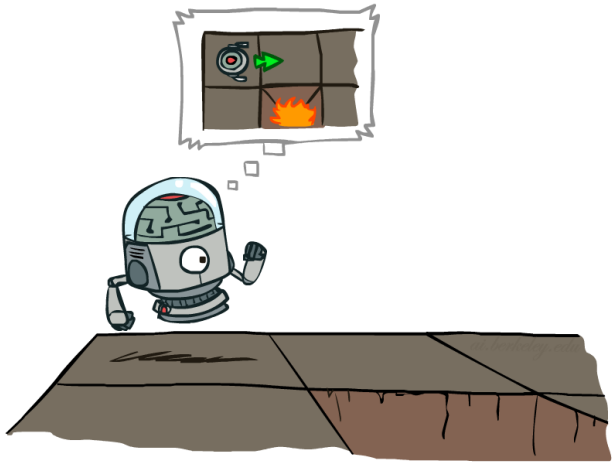


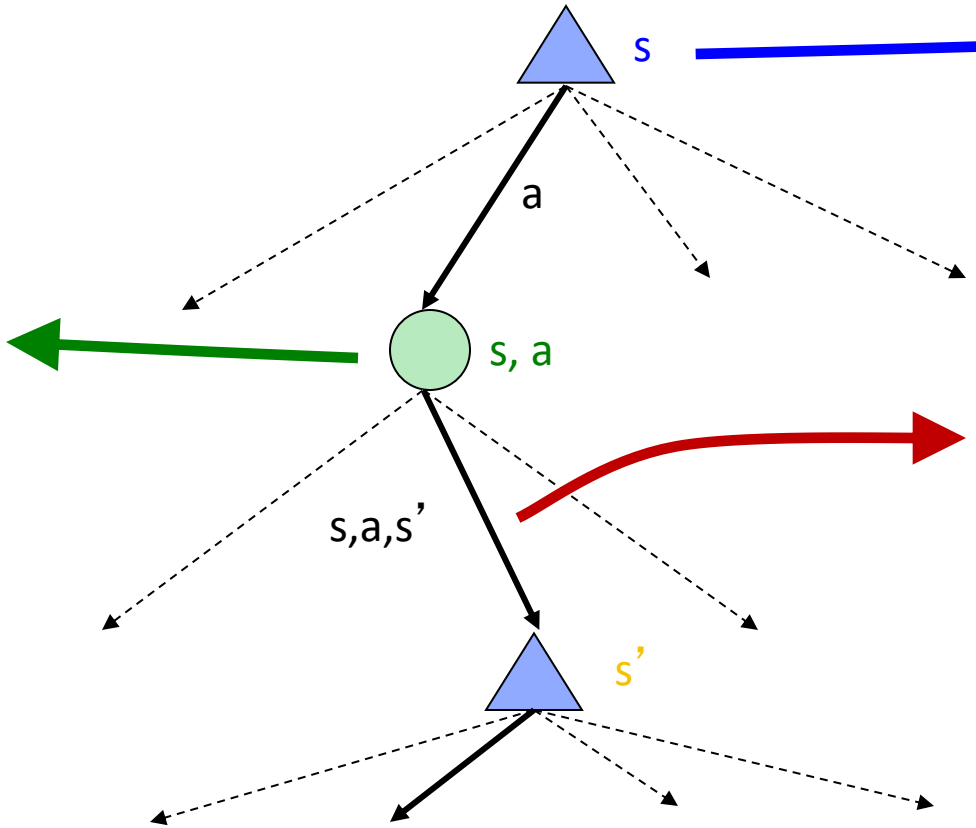
RACING SEARCH TREE



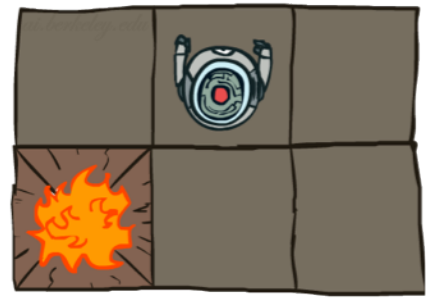
MDP SEARCH TREES

Each MDP state projects a search tree





s is a *state*



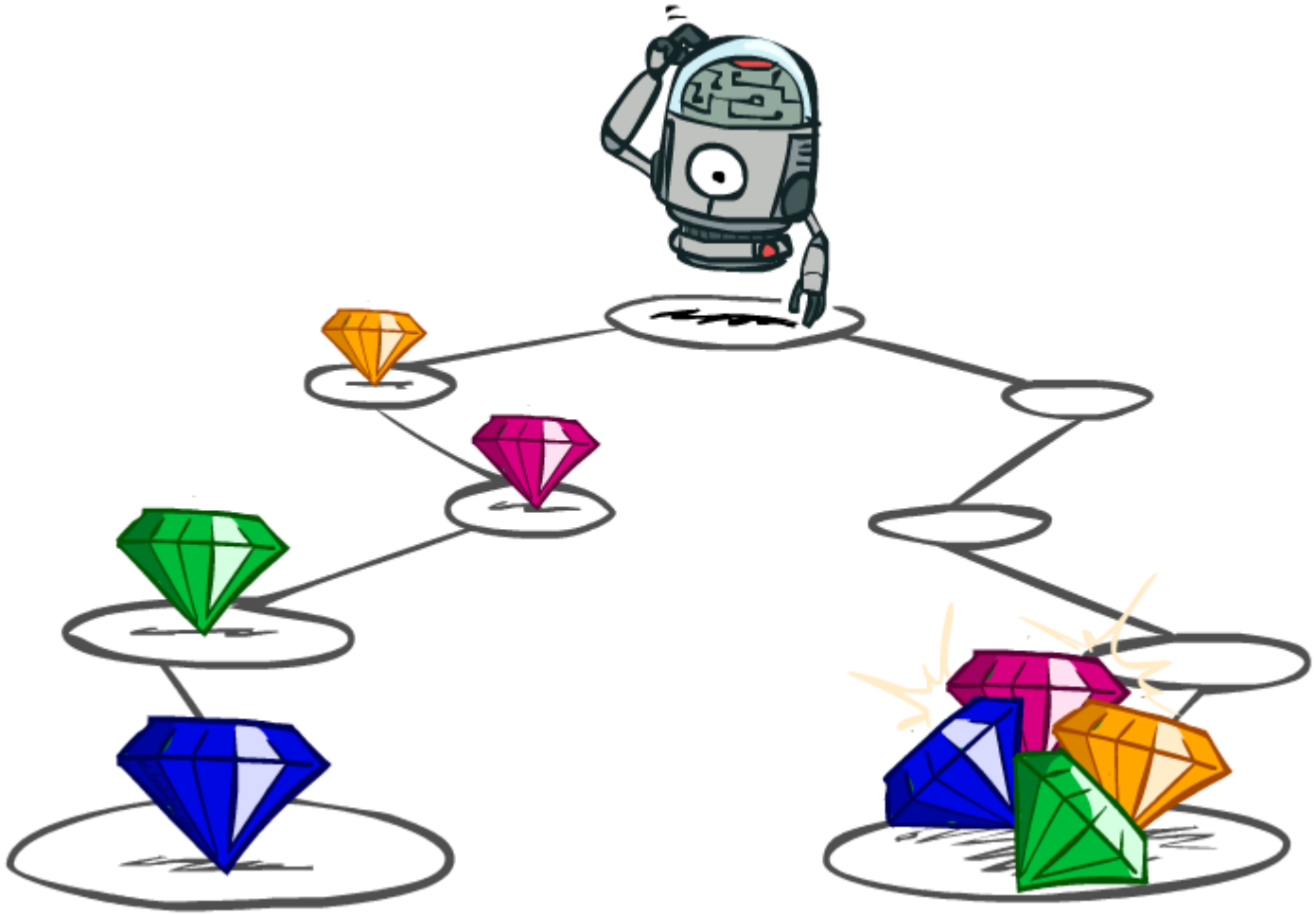
(s, a) is a *q-state*

(s,a,s') called a *transition*

T(s,a,s') = P(s'|s,a)

R(s,a,s')

UTILITIES OF SEQUENCES

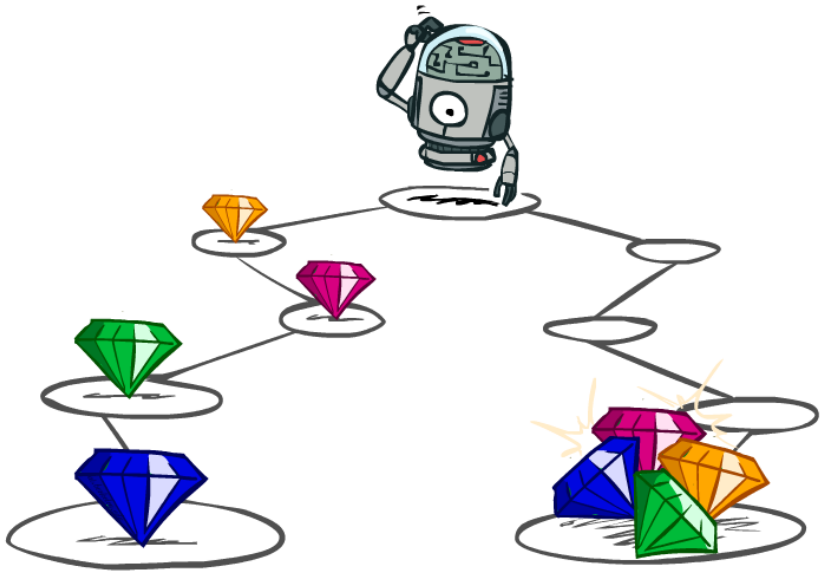


UTILITIES OF SEQUENCES

What preferences should an agent have over reward sequences?

More or less? [1, 2, 2] or [2, 3, 4]

Now or later? [0, 0, 1] or [1, 0, 0]



DISCOUNTING

It's reasonable to maximize the sum of rewards

It's also reasonable to prefer rewards now to rewards later

One solution: utility of rewards decay exponentially

1

Worth Now

γ

Worth Next Step

γ^2

Worth In Two Steps

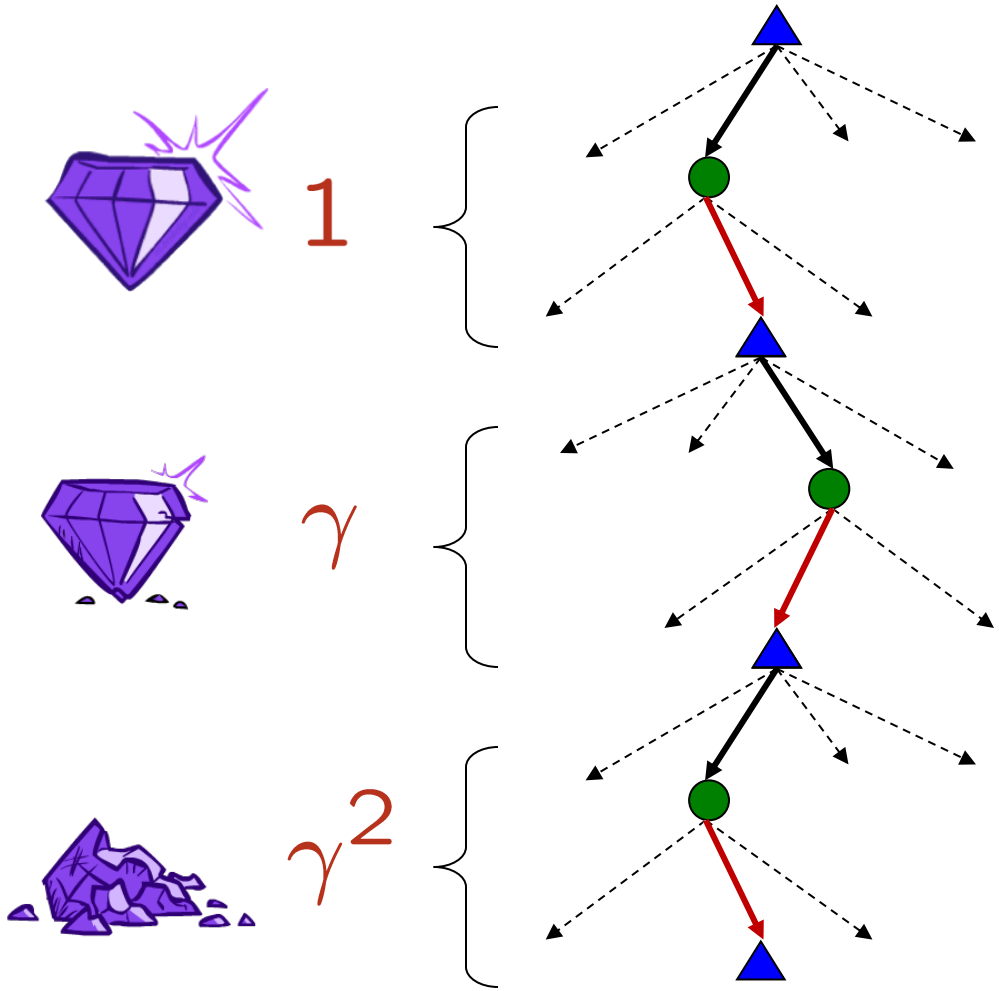
DISCOUNTING

How to discount?

- Each time we descend a level, we multiply in the discount once

Why discount?

- Sooner rewards probably do have higher utility than later rewards
- Also helps our algorithms converge



DISCUSSION POINT!

What is the value of U([2,4,8]) with $\gamma = 0.5$??????????????

(U(·) is the total utility of a reward sequence.)

- 3
- 6
- 7
- 14

Bonus: What is the value of U([8,4,2]) with $\gamma = 0.5$??????????????

DISCUSSION POINT!

What is the value of U([2,4,8]) with $\gamma = 0.5$??????????????

(U(·) is the total utility of a reward sequence.)

- 3
- 6
- 7
- 14

$$\gamma^0 \times 2 + \gamma^1 \times 4 + \gamma^2 \times 8 = 2 + 0.5 \times 4 + 0.5 \times 0.5 \times 8 = 2 + 2 + 2 = 6$$

Bonus: What is the value of U([8,4,2]) with $\gamma = 0.5$??????????????

$$\gamma^0 \times 8 + \gamma^1 \times 4 + \gamma^2 \times 2 = 8 + 0.5 \times 4 + 0.5 \times 0.5 \times 2 = 8 + 2 + 0.5 = 10.5$$

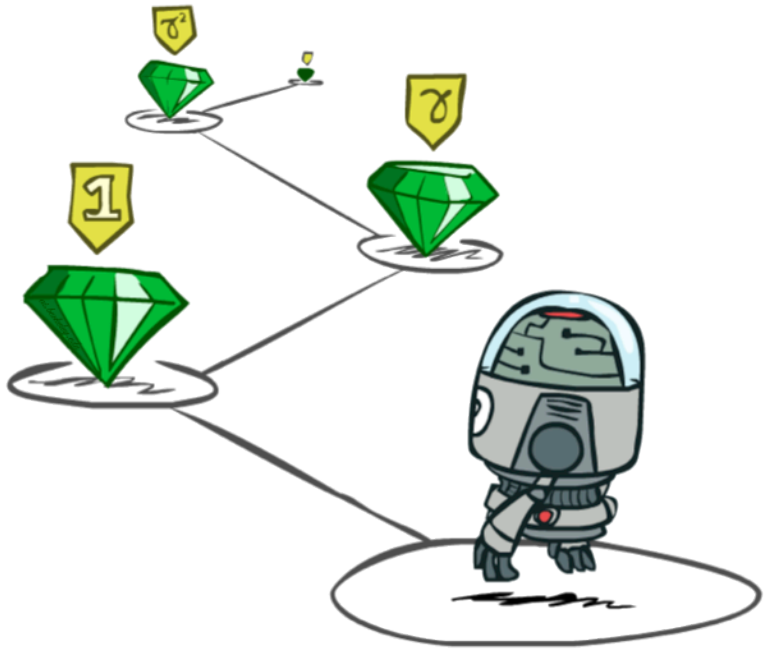
STATIONARY PREFERENCES

Theorem: if we assume stationary preferences:

$$[a_1, a_2, \ldots] \succ [b_1, b_2, \ldots]$$

$$\Updownarrow$$

$$[r, a_1, a_2, \ldots] \succ [r, b_1, b_2, \ldots]$$



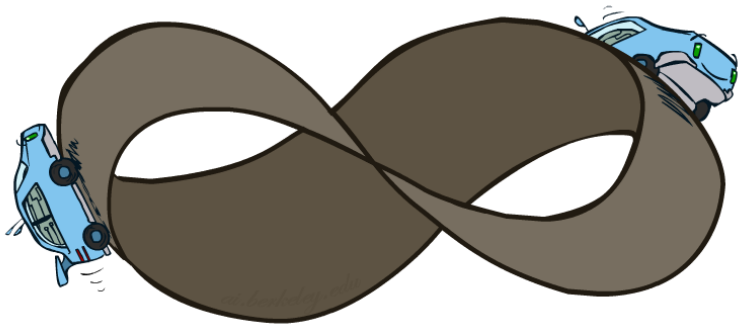
Then: there are only two ways to define utilities

- Additive utility: $U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$
- Discounted utility: $U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$

INFINITE UTILITIES?!

Problem: What if the game lasts forever? Do we get infinite rewards?

Solutions:



- Finite horizon: (similar to depth-limited search)
  - Terminate episodes after a fixed T steps (e.g. life)
  - Gives nonstationary policies (π depends on time left)

- Discounting: use $0 < \gamma < 1$

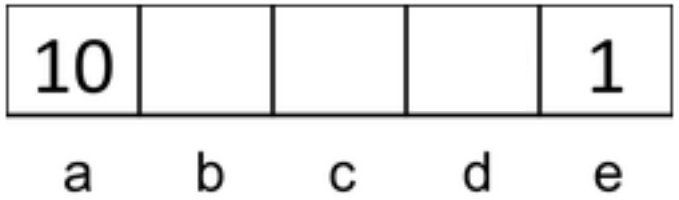
$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1-\gamma)$$

  - Smaller γ means smaller "horizon" – shorter term focus

- Absorbing state: guarantee that for every policy, a terminal state will eventually be reached (like "overheated" for racing)

OPTIMAL POLICY WITH DISCOUNTING

Given

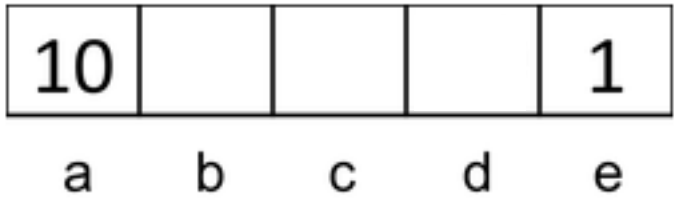


- Actions: East, West, and Exit (only available in exit states a, e)
- Transitions: deterministic

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

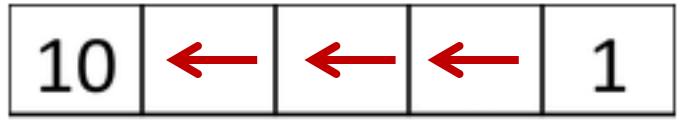
OPTIMAL POLICY WITH DISCOUNTING

Given



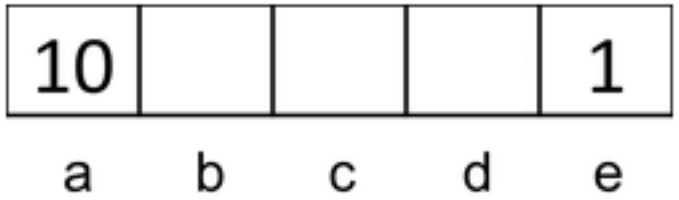
- Actions: East, West, and Exit (only available in exit states a, e)
- Transitions: deterministic

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



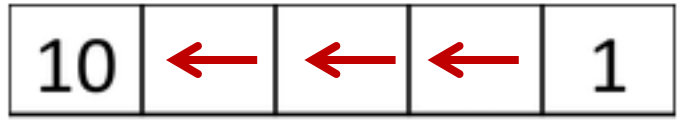
OPTIMAL POLICY WITH DISCOUNTING

Given



- Actions: East, West, and Exit (only available in exit states a, e)
- Transitions: deterministic

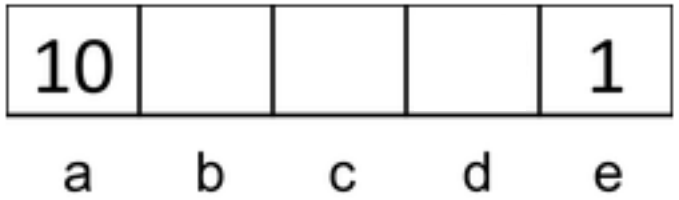
For $\gamma = 1$, what is the optimal policy?



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

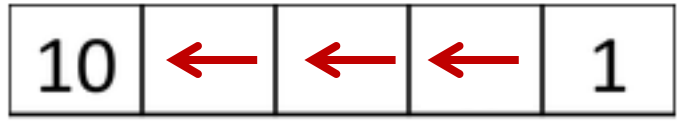
OPTIMAL POLICY WITH DISCOUNTING

Given

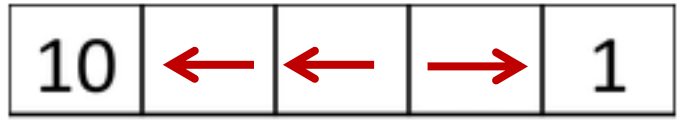


- Actions: East, West, and Exit (only available in exit states a, e)
- Transitions: deterministic

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

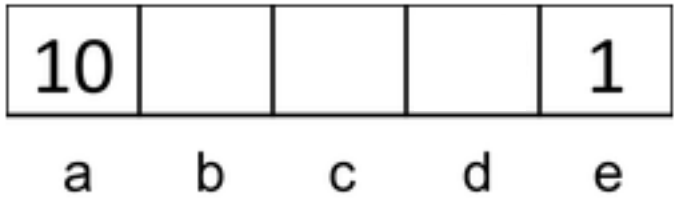


For $\gamma = 0.1$, what is the optimal policy?



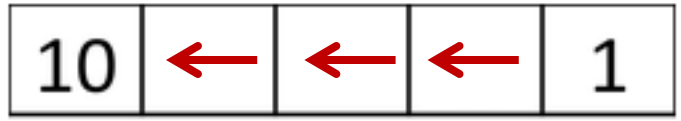
OPTIMAL POLICY WITH DISCOUNTING

Given

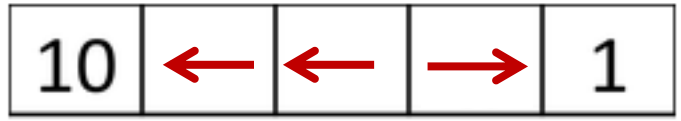


- Actions: East, West, and Exit (only available in exit states a, e)
- Transitions: deterministic

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



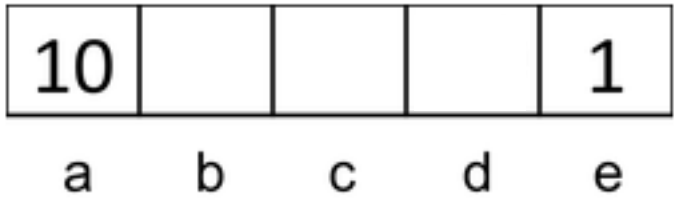
For $\gamma = 0.1$, what is the optimal policy?



For which γ are West and East equally good when in state d?

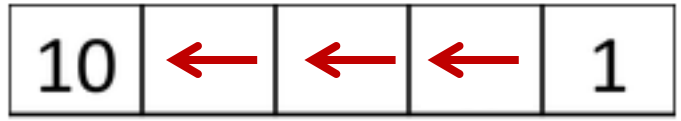
OPTIMAL POLICY WITH DISCOUNTING

Given

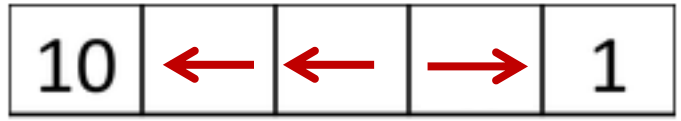


- Actions: East, West, and Exit (only available in exit states a, e)
- Transitions: deterministic

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



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



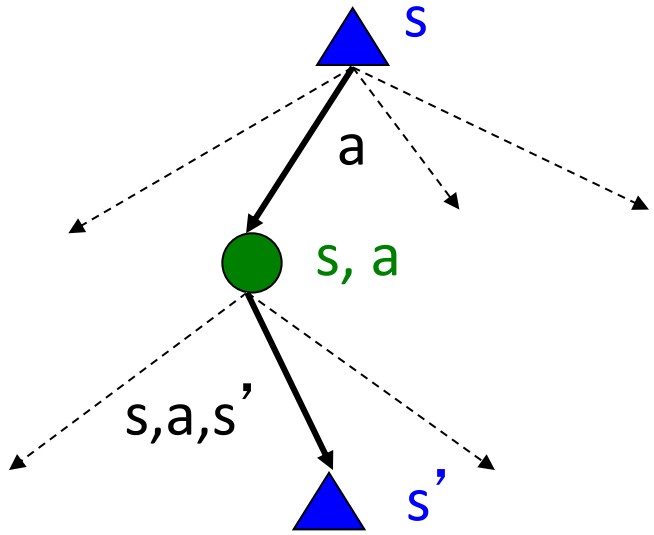
For which γ are West and East equally good when in state d?

$$\gamma^3 \times 10 = \gamma^1 \times 1$$

MDP QUANTITIES (SO FAR!)

Markov decision processes:

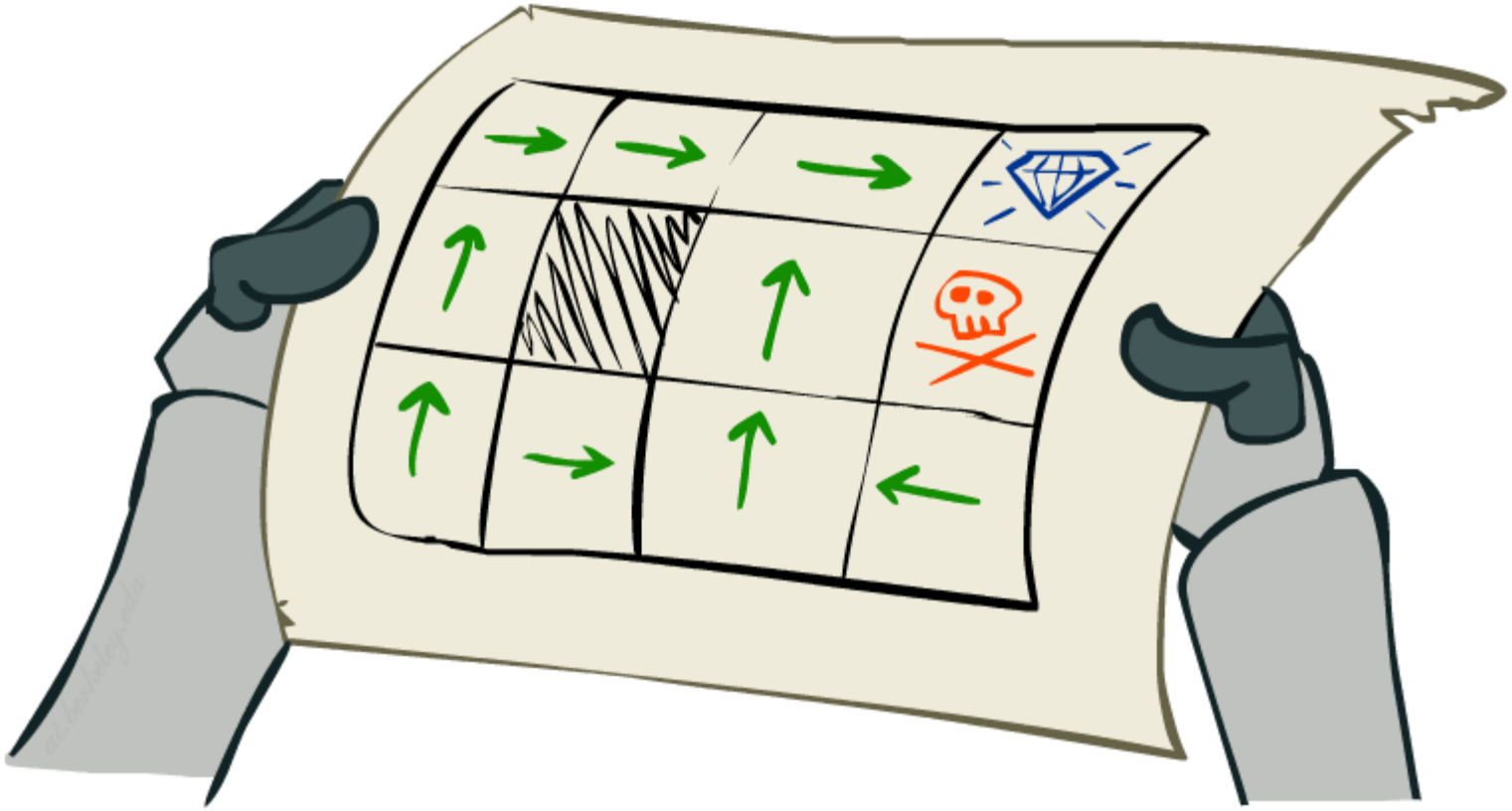
- States S
- Actions A
- Transitions P(s'|s,a) (or T(s,a,s'))
- Rewards R(s,a,s') (and discount γ)
- Start state s0



MDP quantities so far:

- Policy = map of states to actions
- Utility = sum of (discounted) rewards

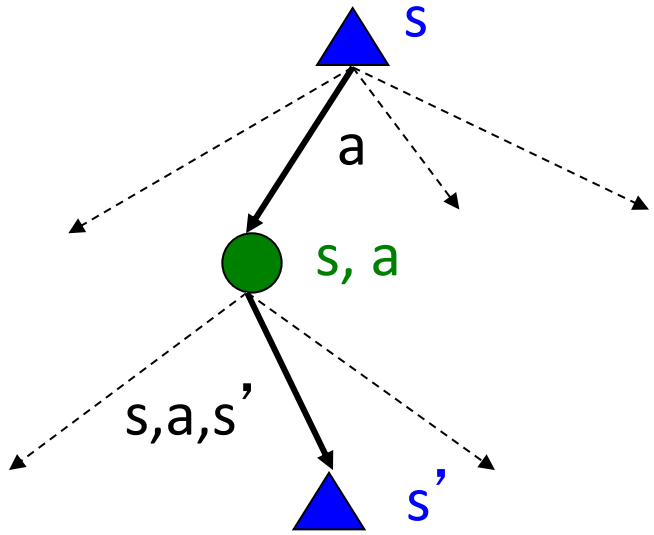
SOLVING MDPS



MDP QUANTITIES

Markov decision processes:

- States S
- Actions A
- Transitions P(s'|s,a) (or T(s,a,s'))
- Rewards R(s,a,s') (and discount γ)
- Start state s_0



MDP quantities:

- Policy = map of states to actions
- Utility = sum of (discounted) rewards
- (State) Value = expected utility starting from a state (max node)
- Q-Value = expected utility starting from a state-action pair, i.e., q-state (chance node)

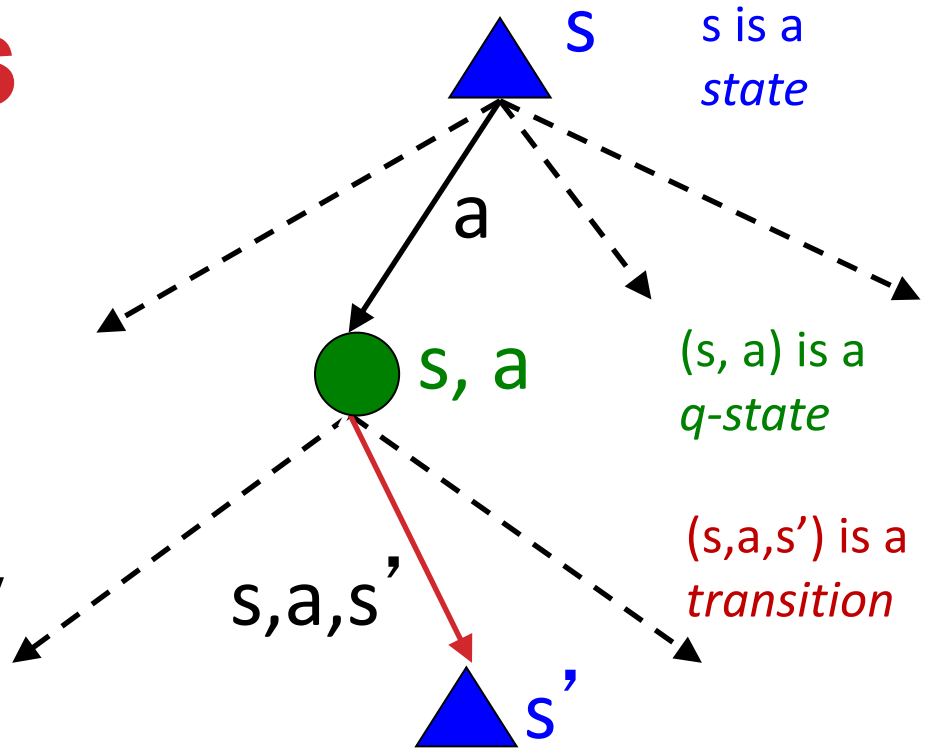
MDP OPTIMAL QUANTITIES

The optimal policy:

- $\pi^*(s)$ = optimal action from state s

The (true) value (or utility) of a state s:

- $V^*(s)$ = expected utility starting in s and acting optimally

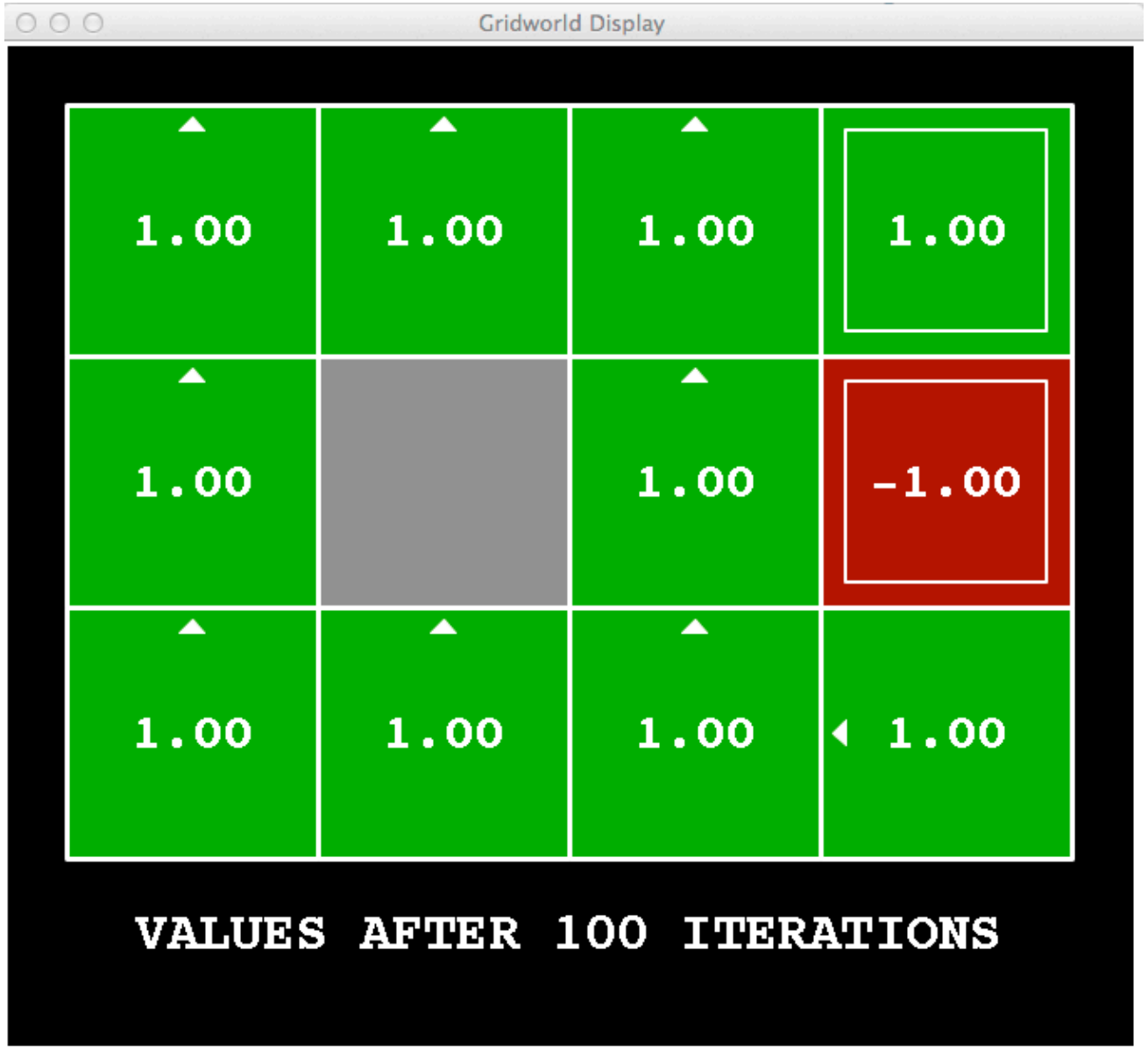


The (true) value (or utility) of a q-state (s,a):

- $Q^*(s,a)$ = expected utility starting out having taken action a from state s and (thereafter) acting optimally

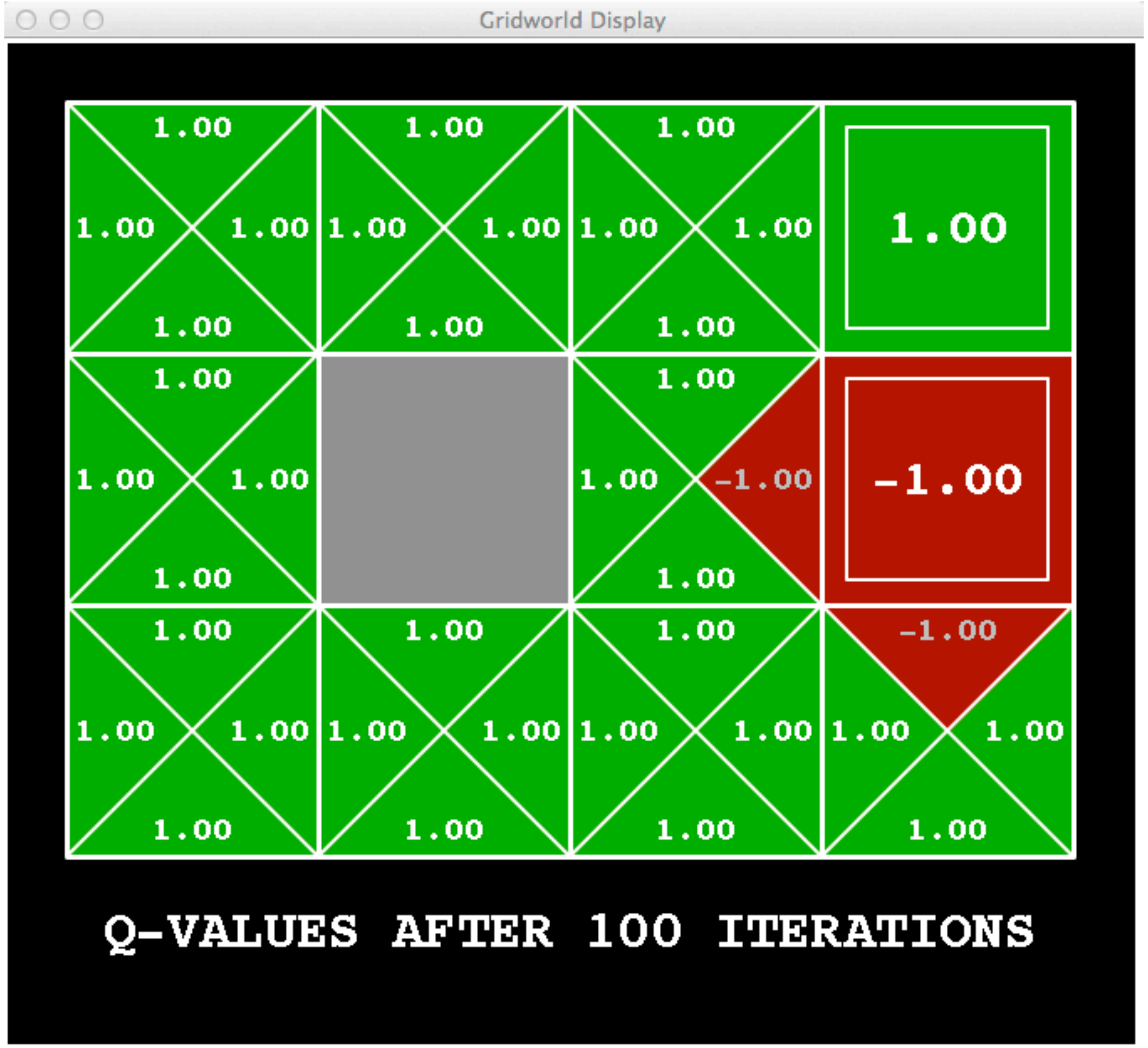
Solve MDP: Find π^*, V^* and/or Q^*

GRIDWORLD V VALUES



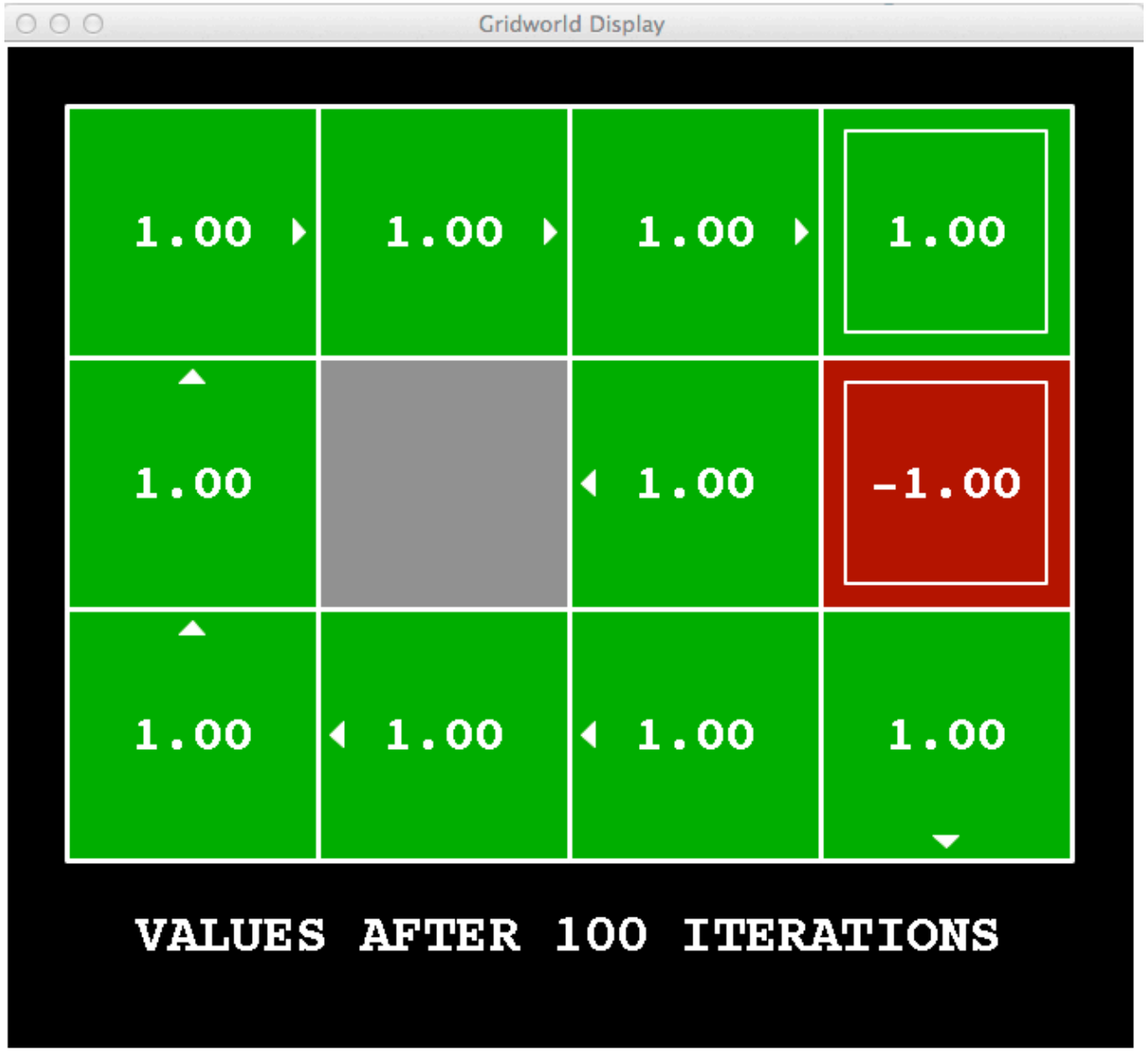
Noise = 0
Discount = 1
Living reward = 0

GRIDWORLD Q VALUES



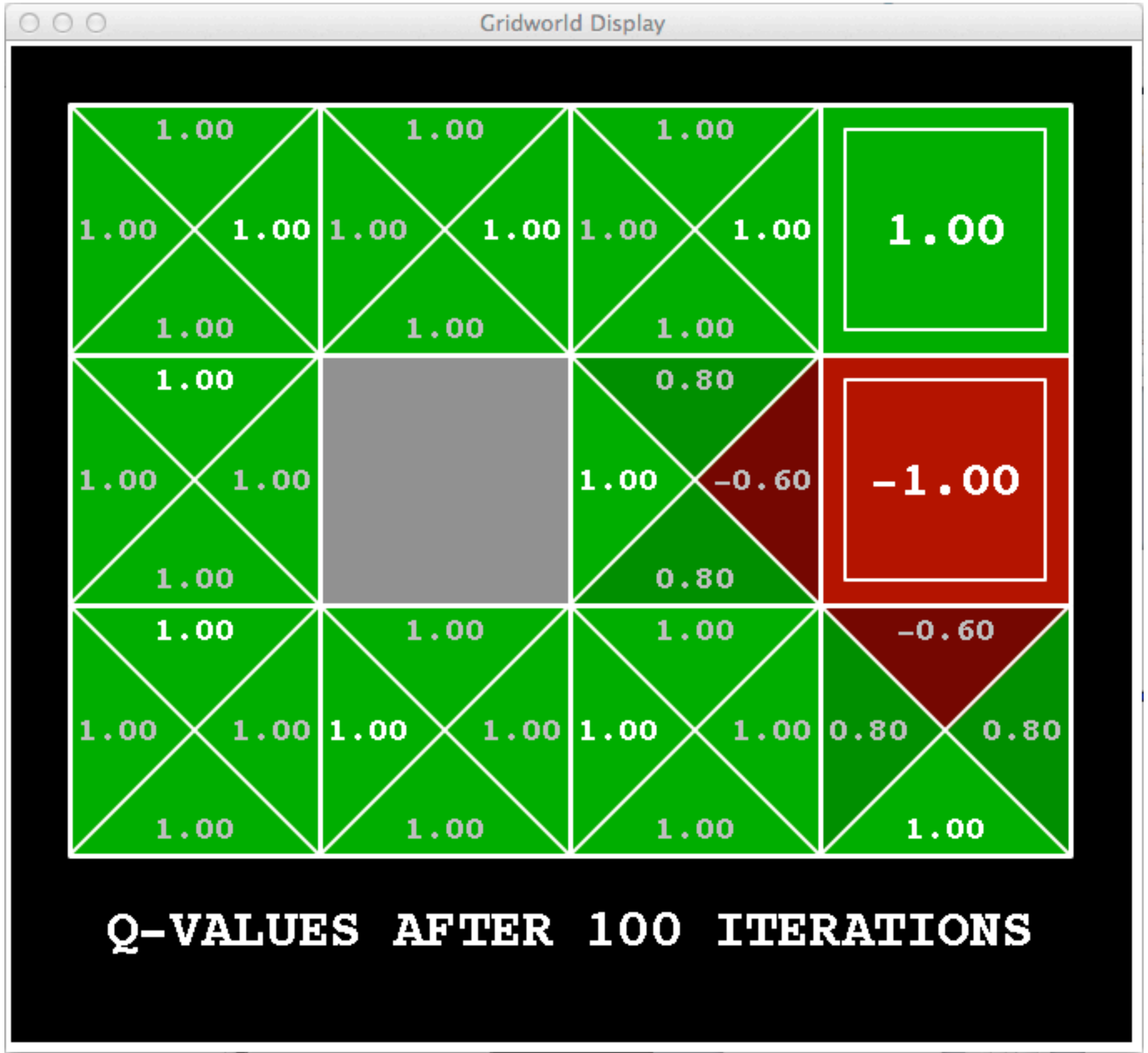
Noise = 0
Discount = 1
Living reward = 0

GRIDWORLD V VALUES



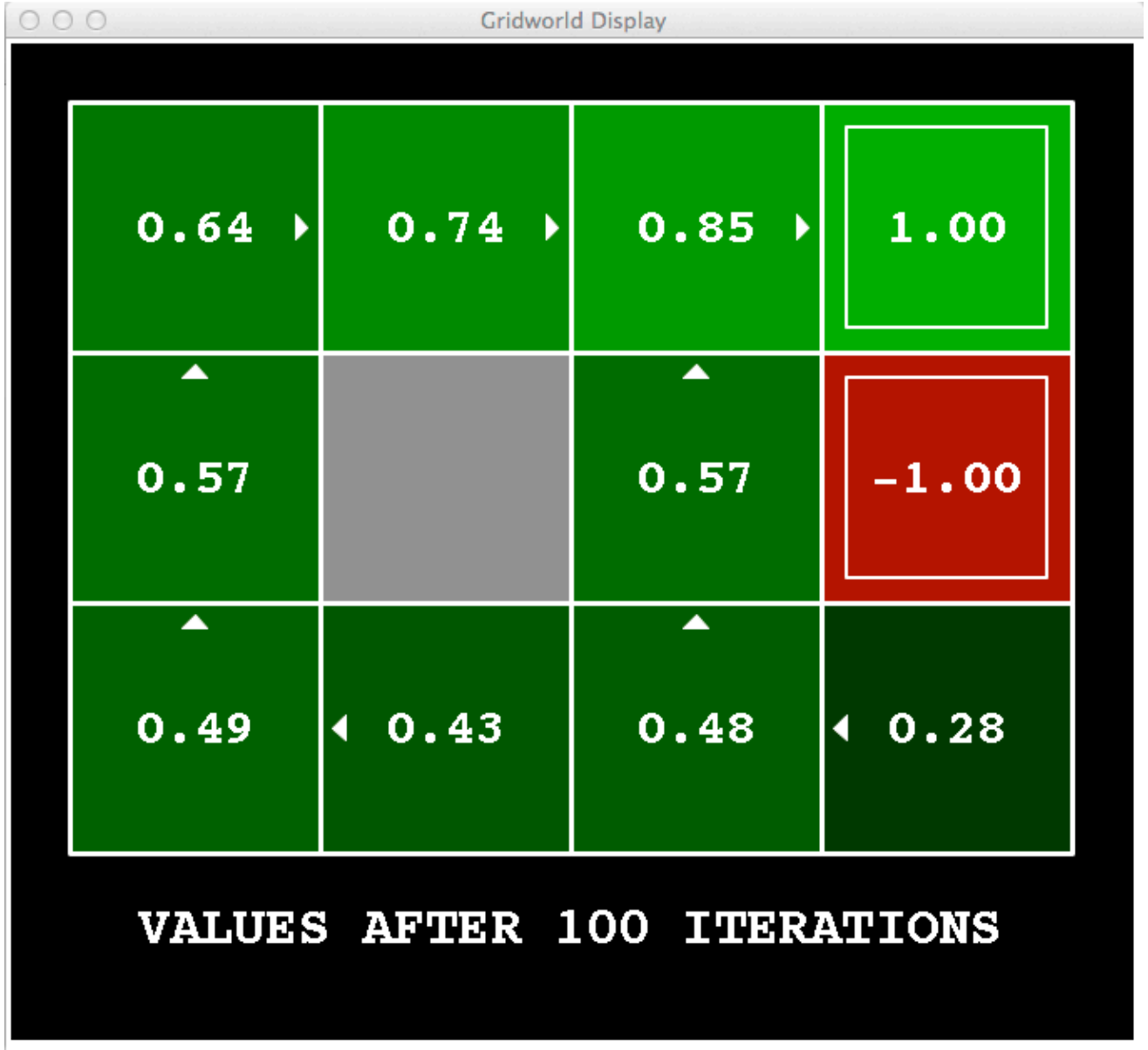
Noise = 0.2
Discount = 1
Living reward = 0

GRIDWORLD Q VALUES



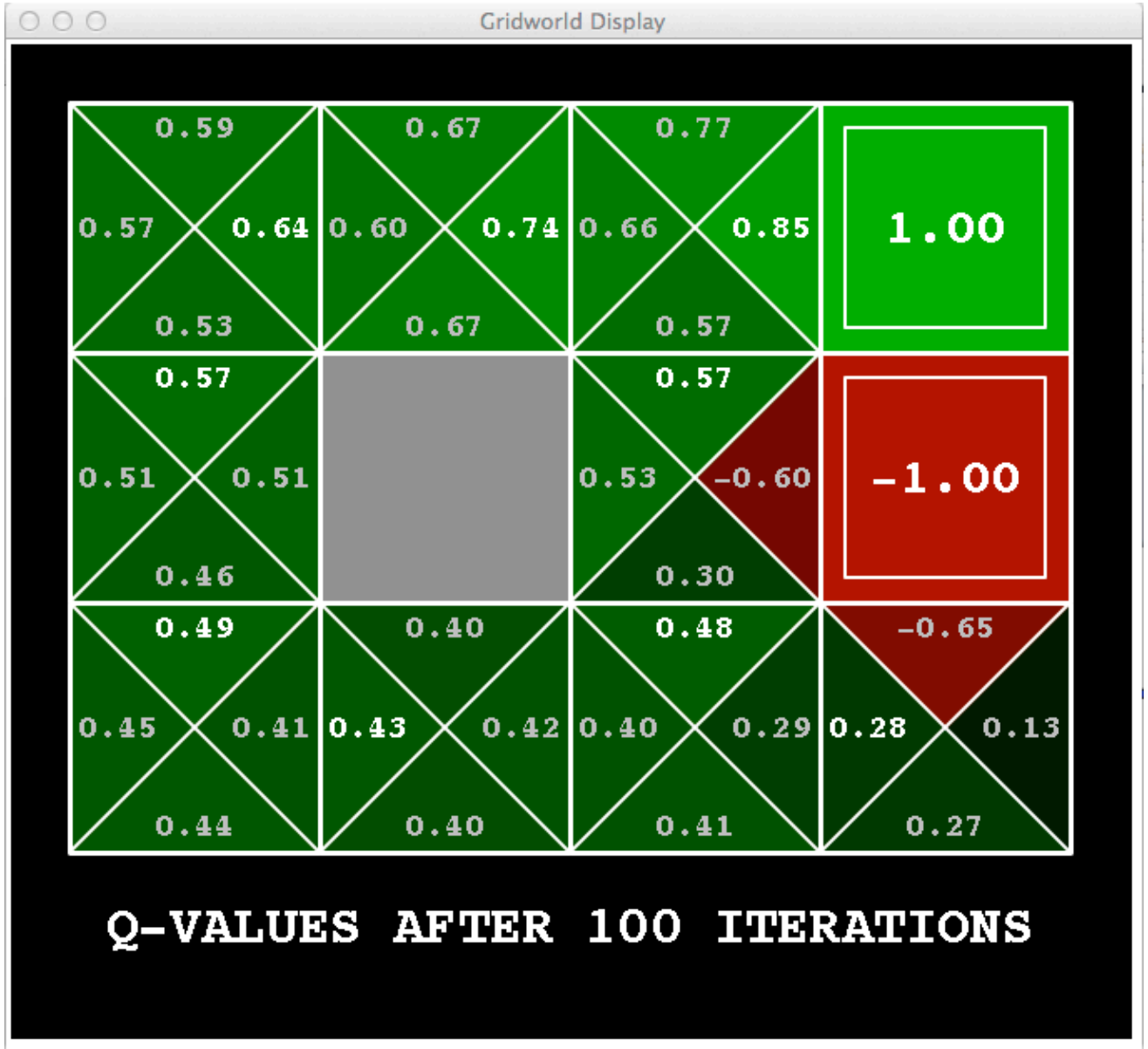
Noise = 0.2
Discount = 1
Living reward = 0

GRIDWORLD V VALUES



Noise = 0.2
Discount = 0.9
Living reward = 0

GRIDWORLD Q VALUES

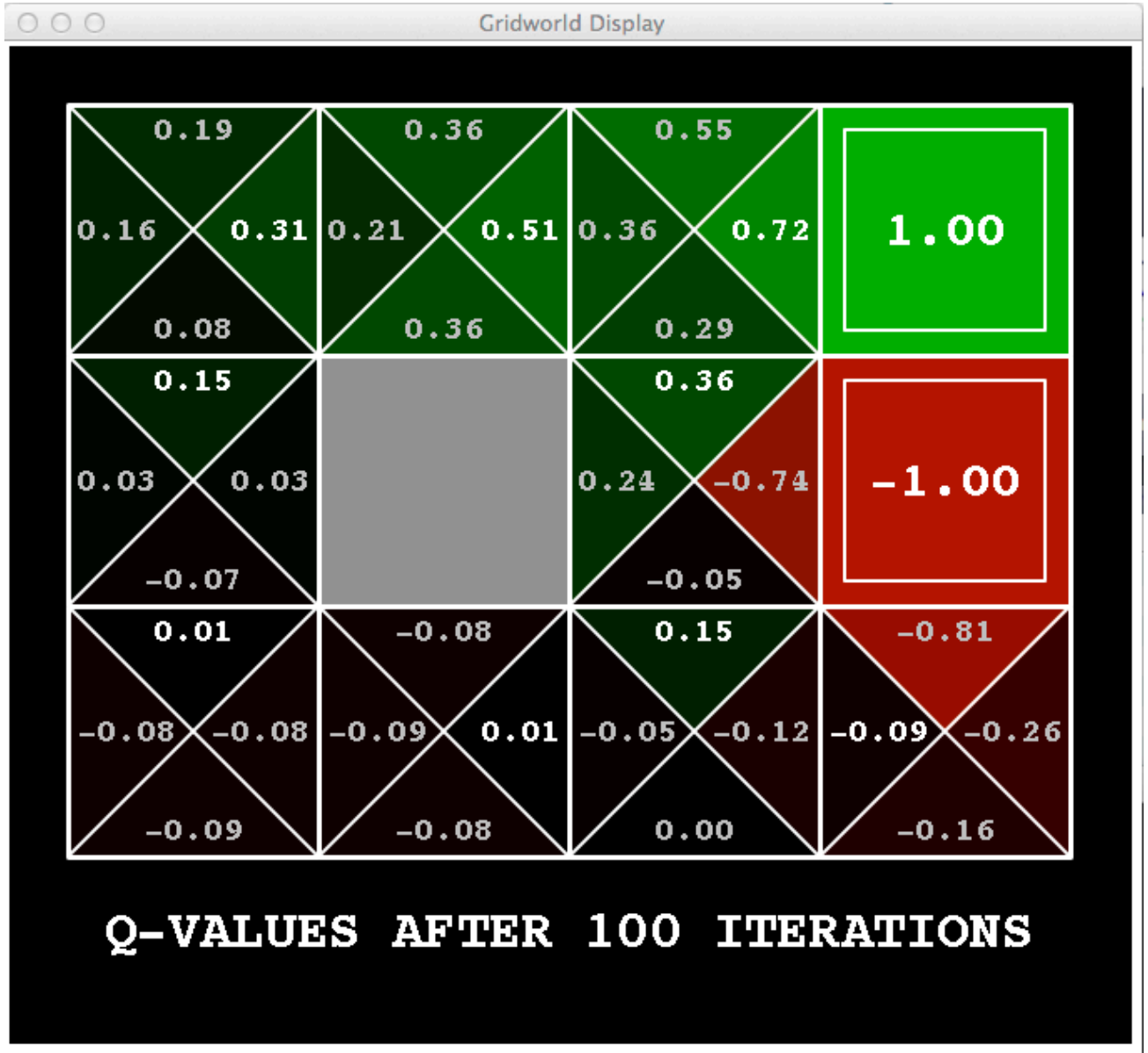


Noise = 0.2
Discount = 0.9
Living reward = 0

GRIDWORLD V VALUES

Noise = 0.2
Discount = 0.9
Living reward = -0.1

GRIDWORLD Q VALUES



Noise = 0.2
Discount = 0.9
Living reward = -0.1

GREAT, SO HOW DO WE COMPUTE THESE VALUES?

VALUES OF STATES

Fundamental operation: compute the (expectimax) value of a state

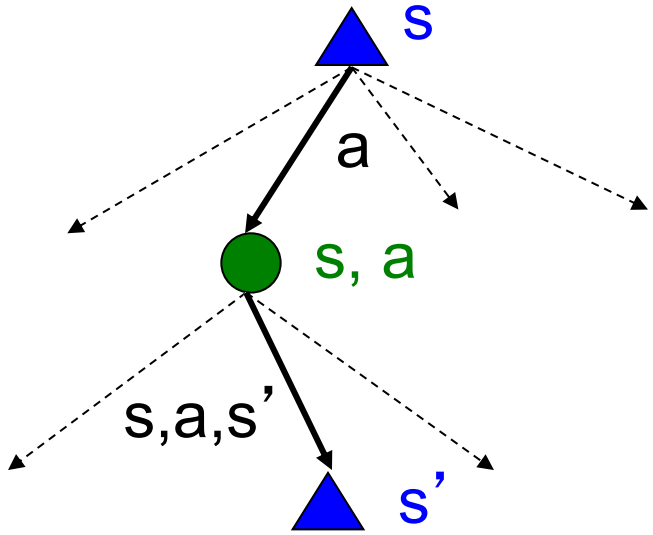
- Expected utility under optimal action
- Average sum of (discounted) rewards

Recursive definition of value:

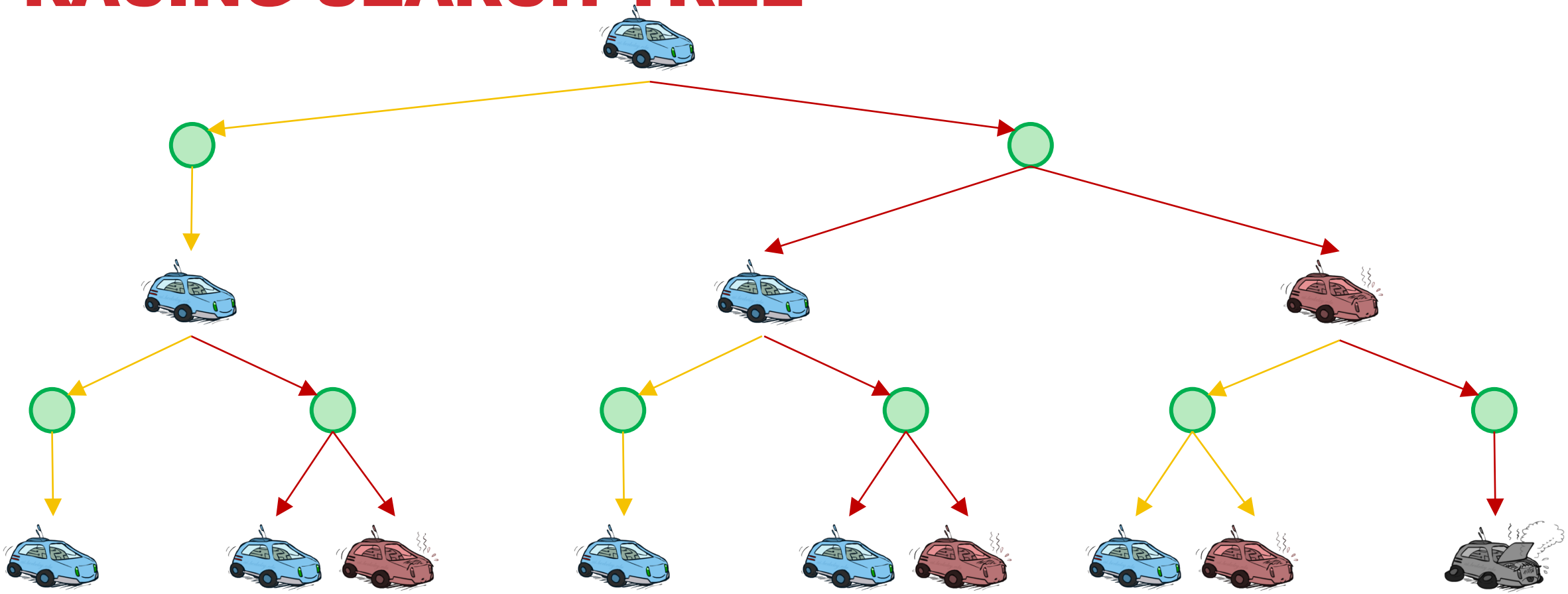
$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

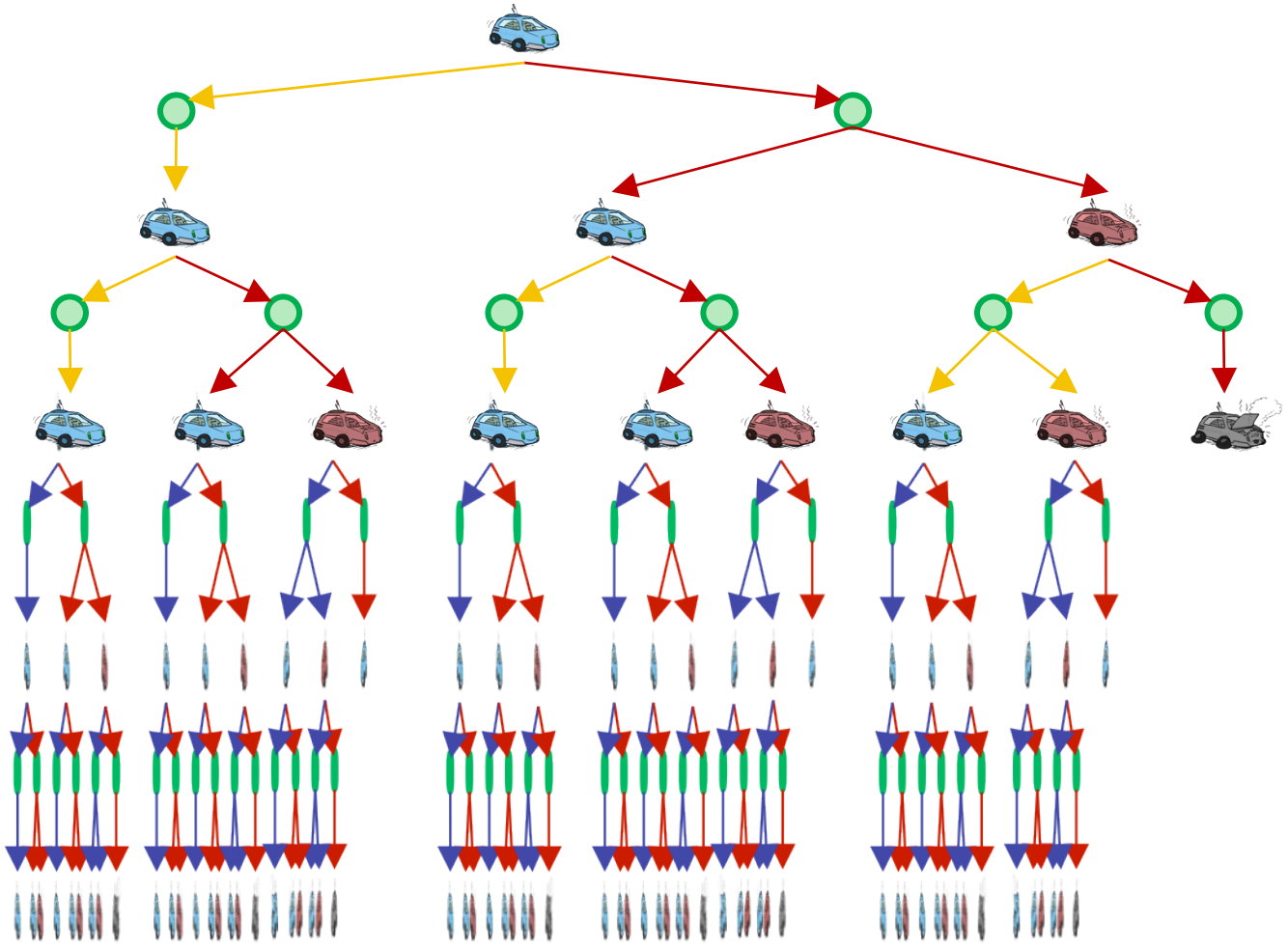
$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$



RACING SEARCH TREE



RACING SEARCH TREE



Enumerate all paths, determine value of each path, choose path with highest value (aka expectimax)…?

RACING SEARCH TREE

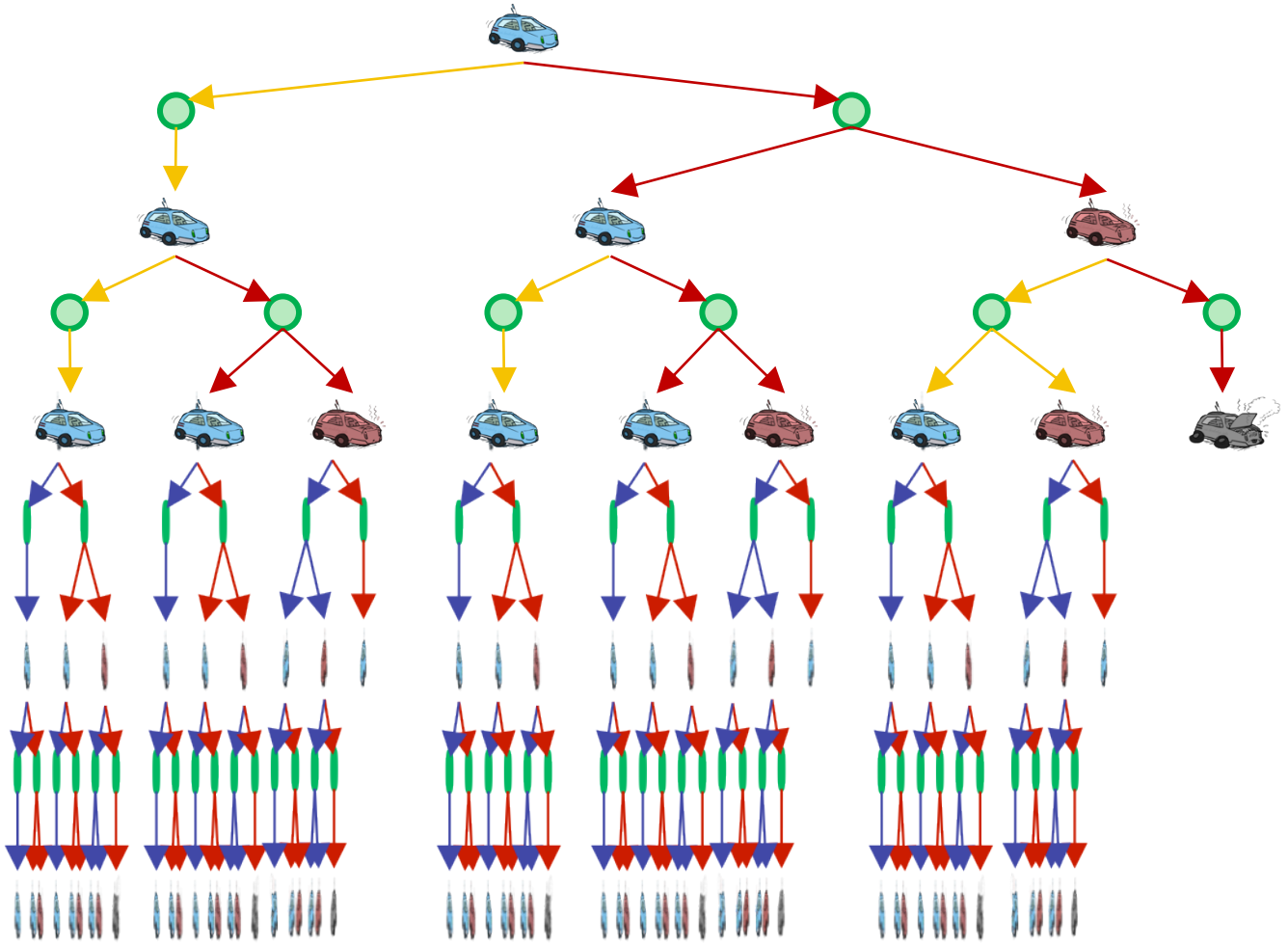
We're doing way too much work with expectimax!

Problem: States are repeated

- Idea: Only compute needed quantities once

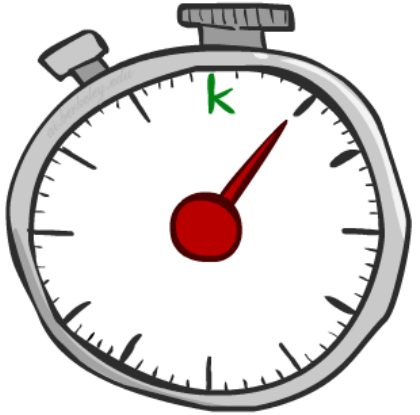
Problem: Tree goes on forever

- Idea: Do a depth-limited computation, but with increasing depths until change is small
- Note: deep parts of the tree eventually don't matter if $\gamma < 1$



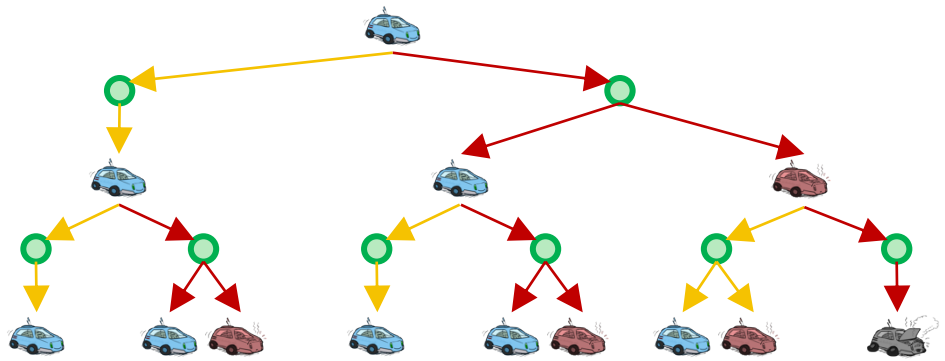
TIME-LIMITED VALUES

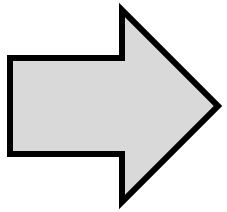
Key idea: time-limited values

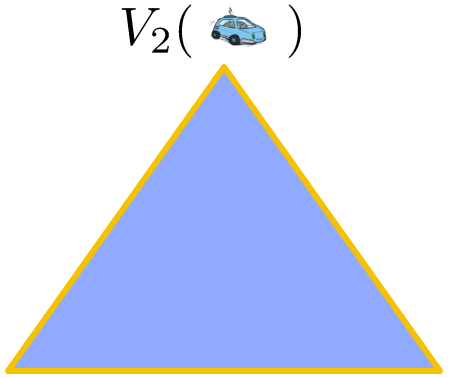


Define $V_k(s)$ to be the optimal value of s if the game ends in k more time steps

- Equivalently, it's what a depth-k expectimax would give from s

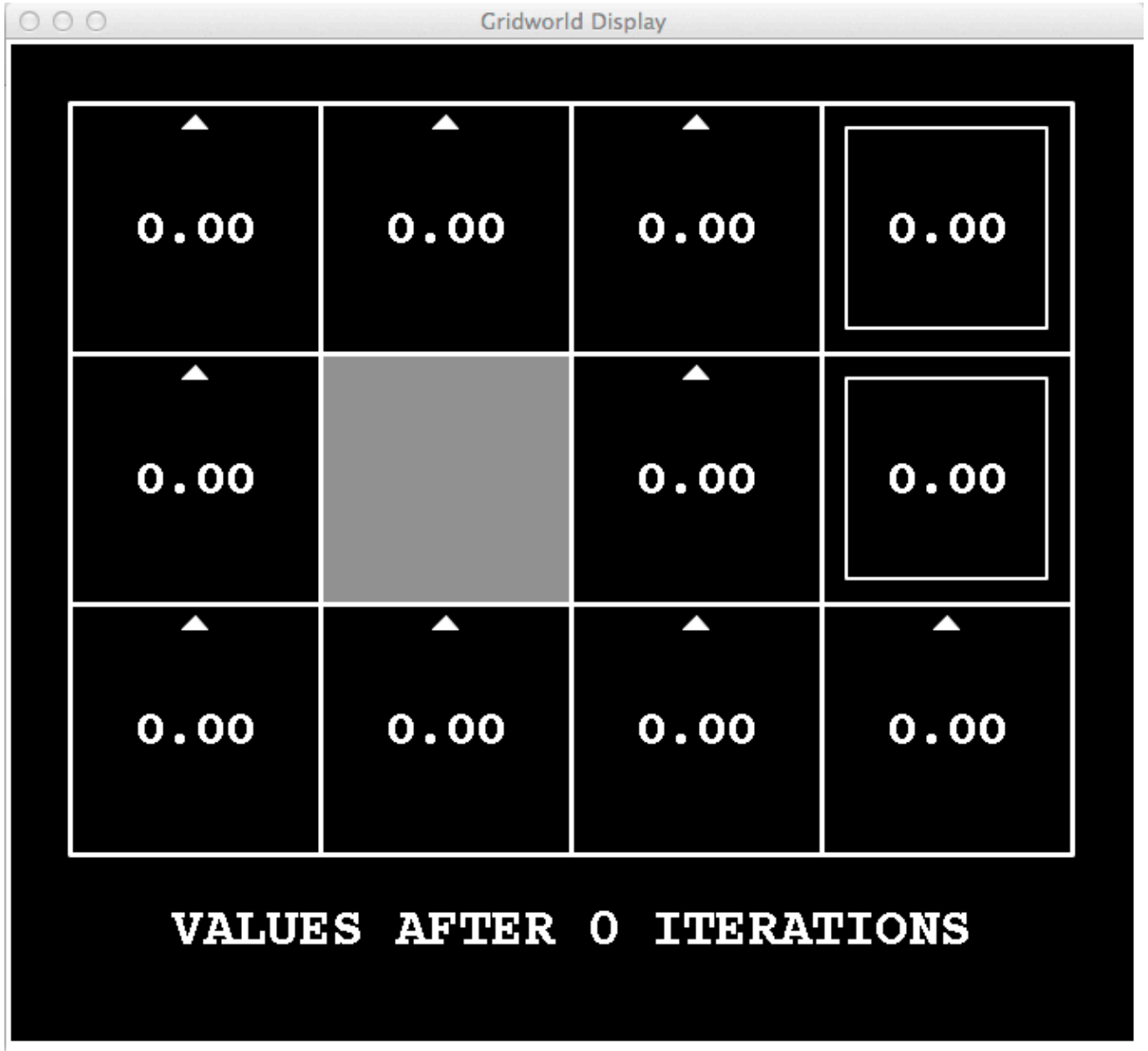






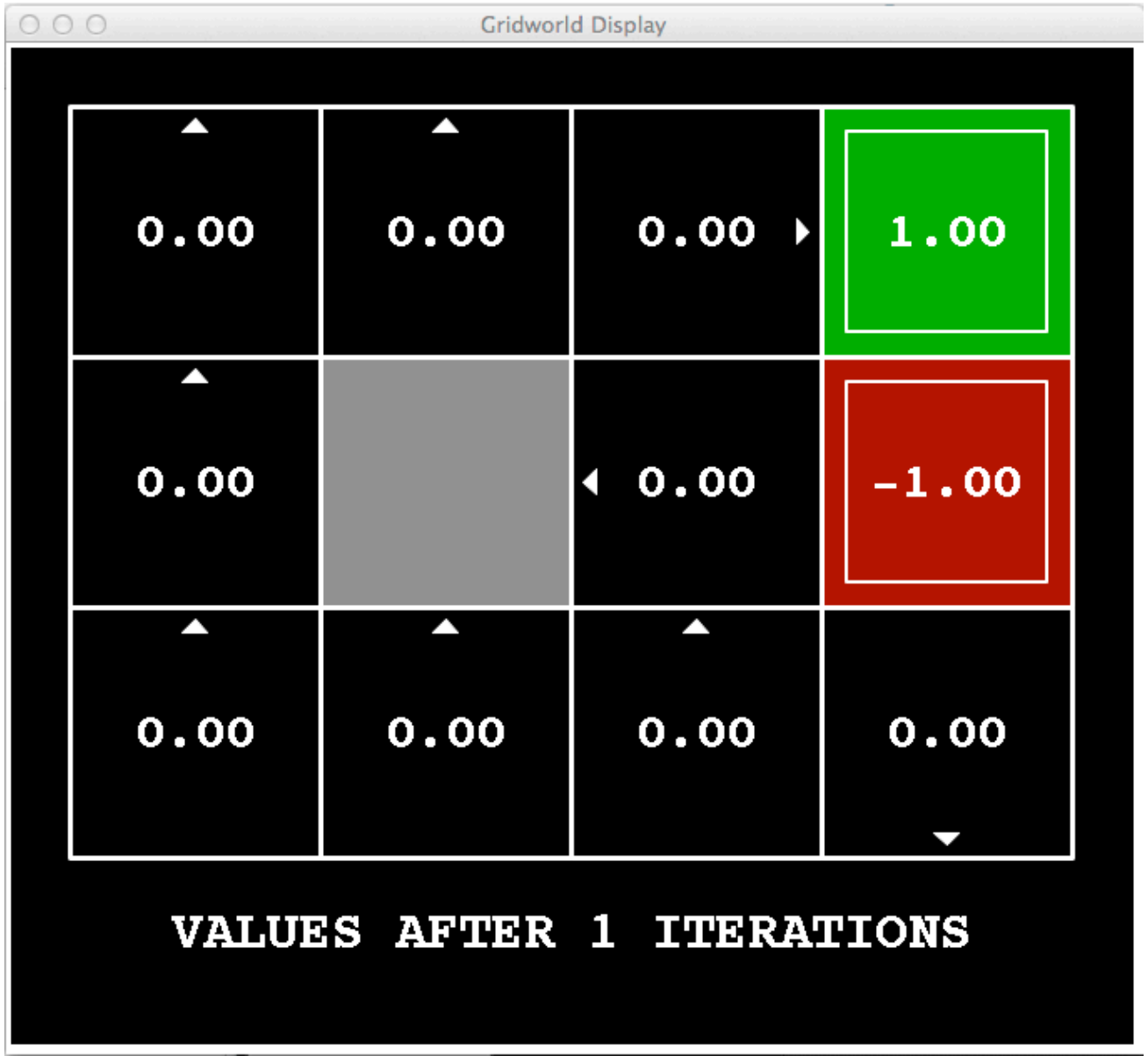
(Also watch the actions change as we go along.)

K=0



Noise = 0.2
Discount = 0.9
Living reward = 0

K=1



Noise = 0.2
Discount = 0.9
Living reward = 0

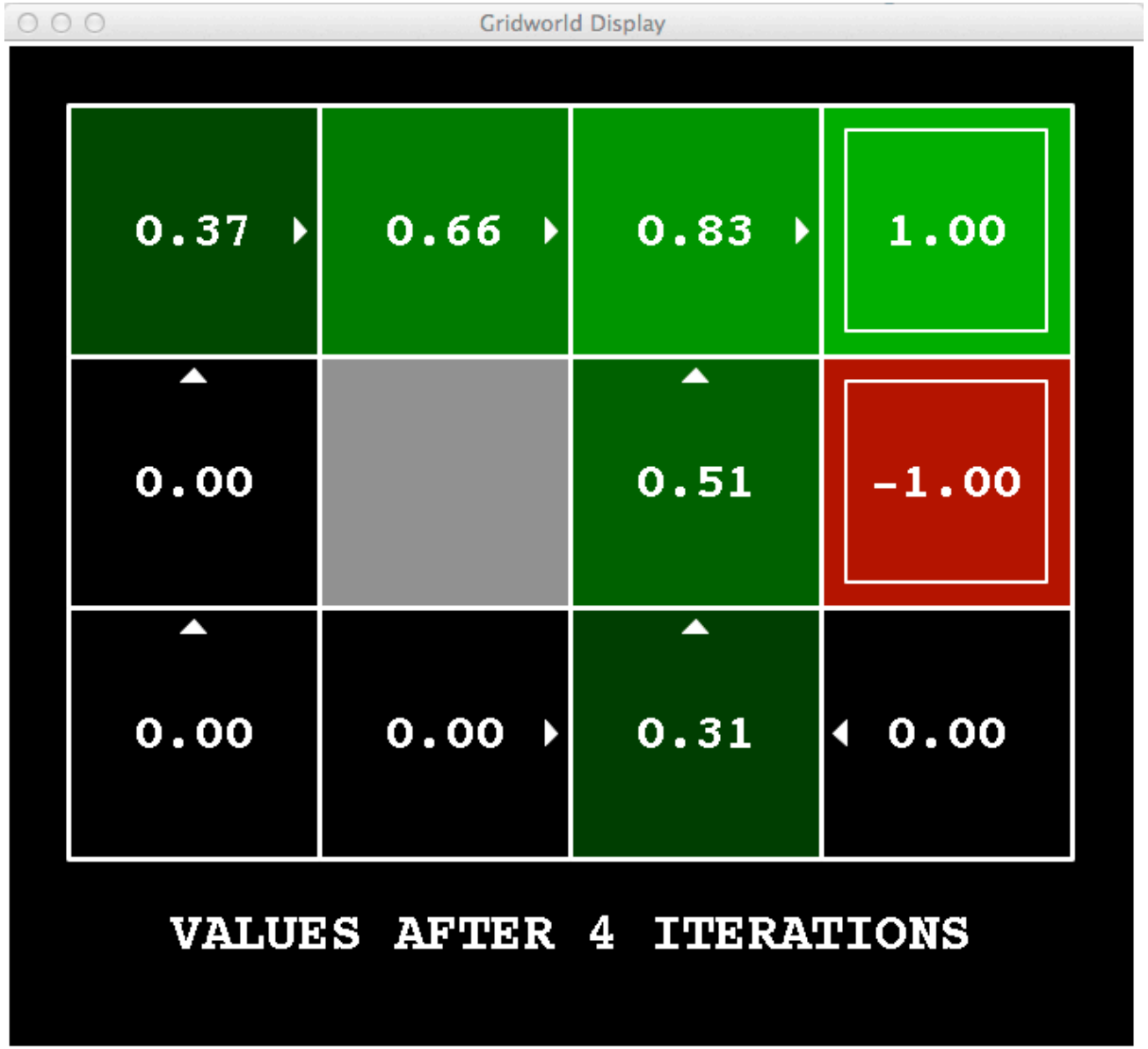
K=2

Noise = 0.2
Discount = 0.9
Living reward = 0

K=3

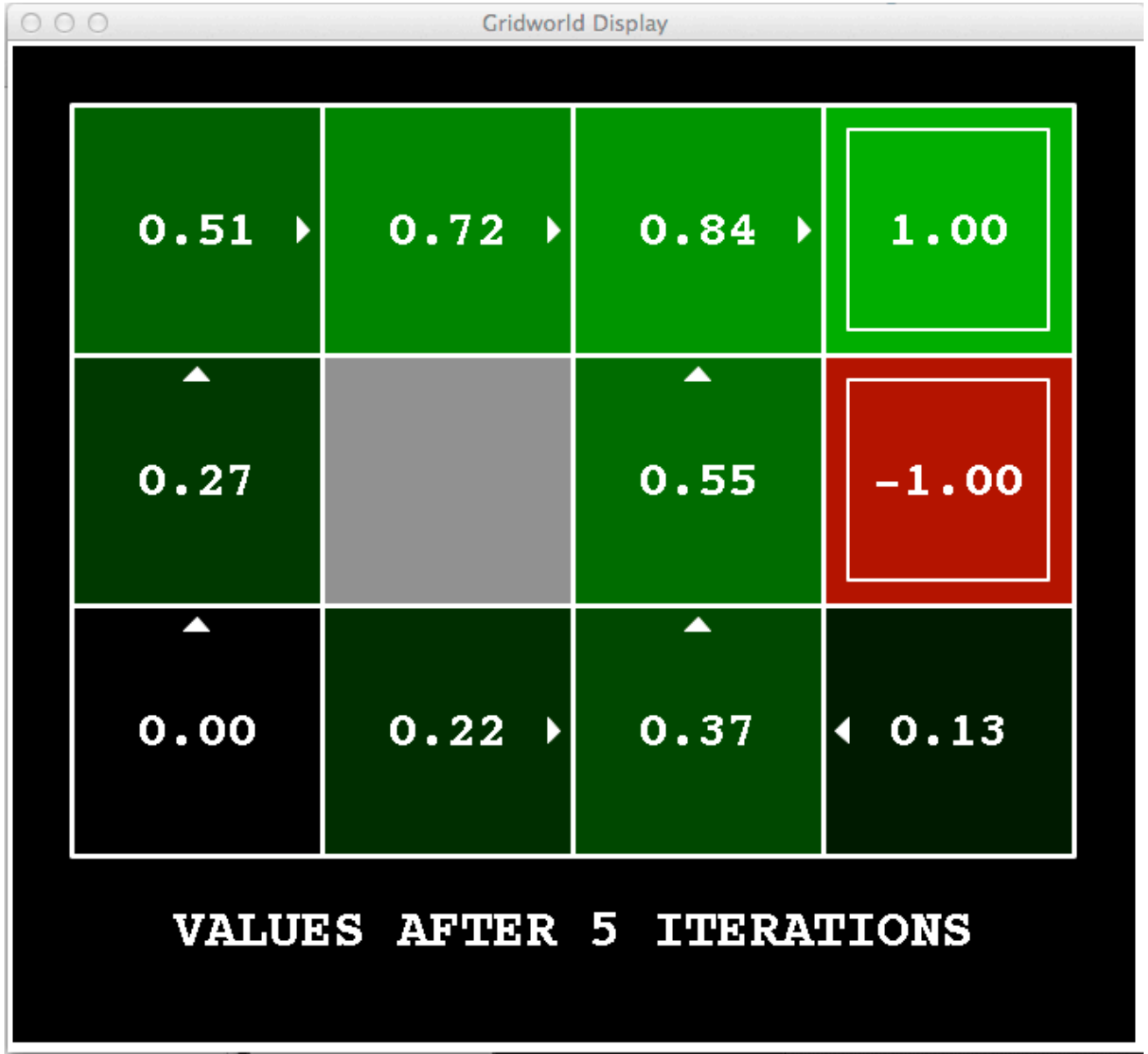
Noise = 0.2
Discount = 0.9
Living reward = 0

K=4



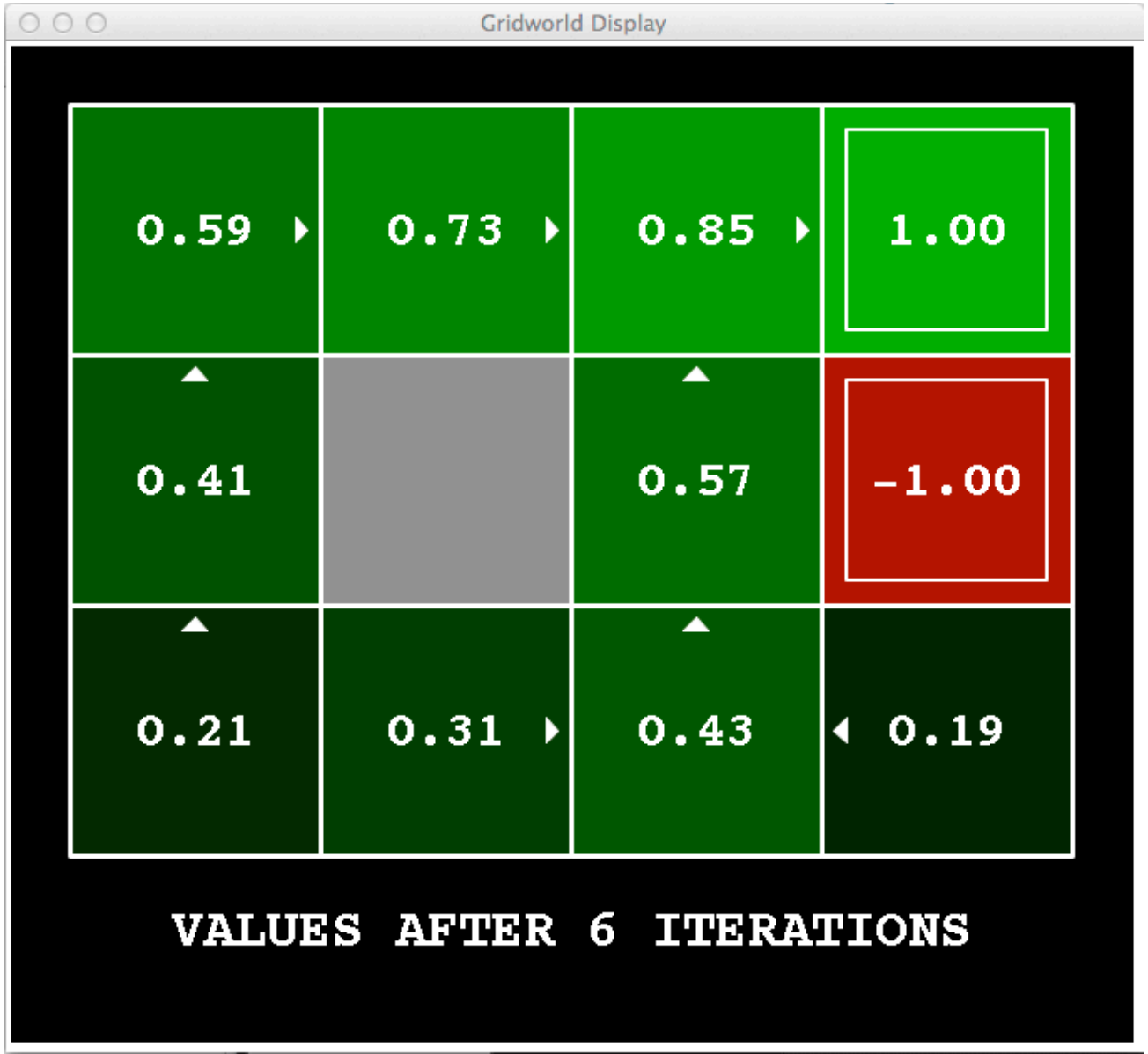
Noise = 0.2
Discount = 0.9
Living reward = 0

K=5



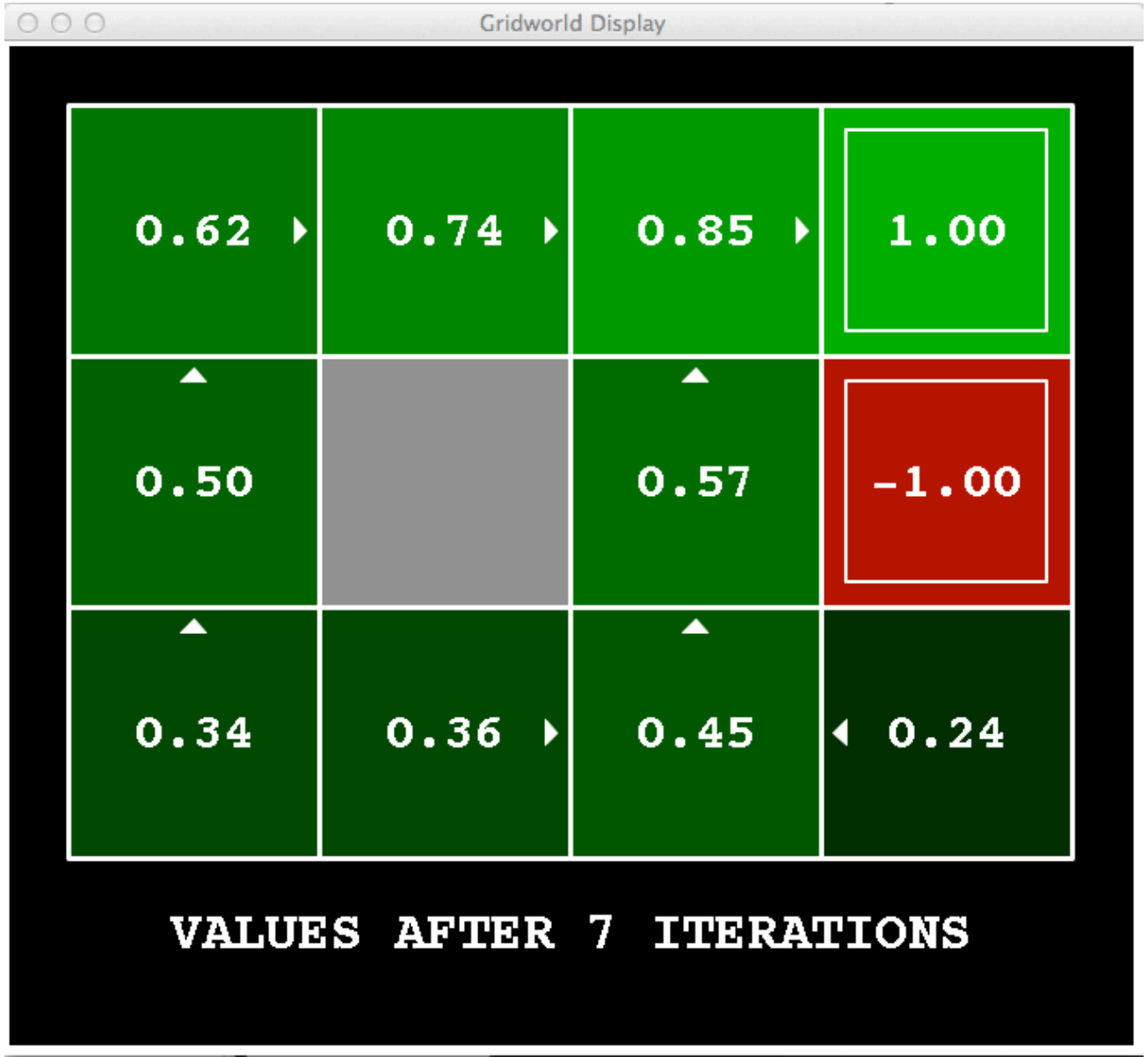
Noise = 0.2
Discount = 0.9
Living reward = 0

K=6



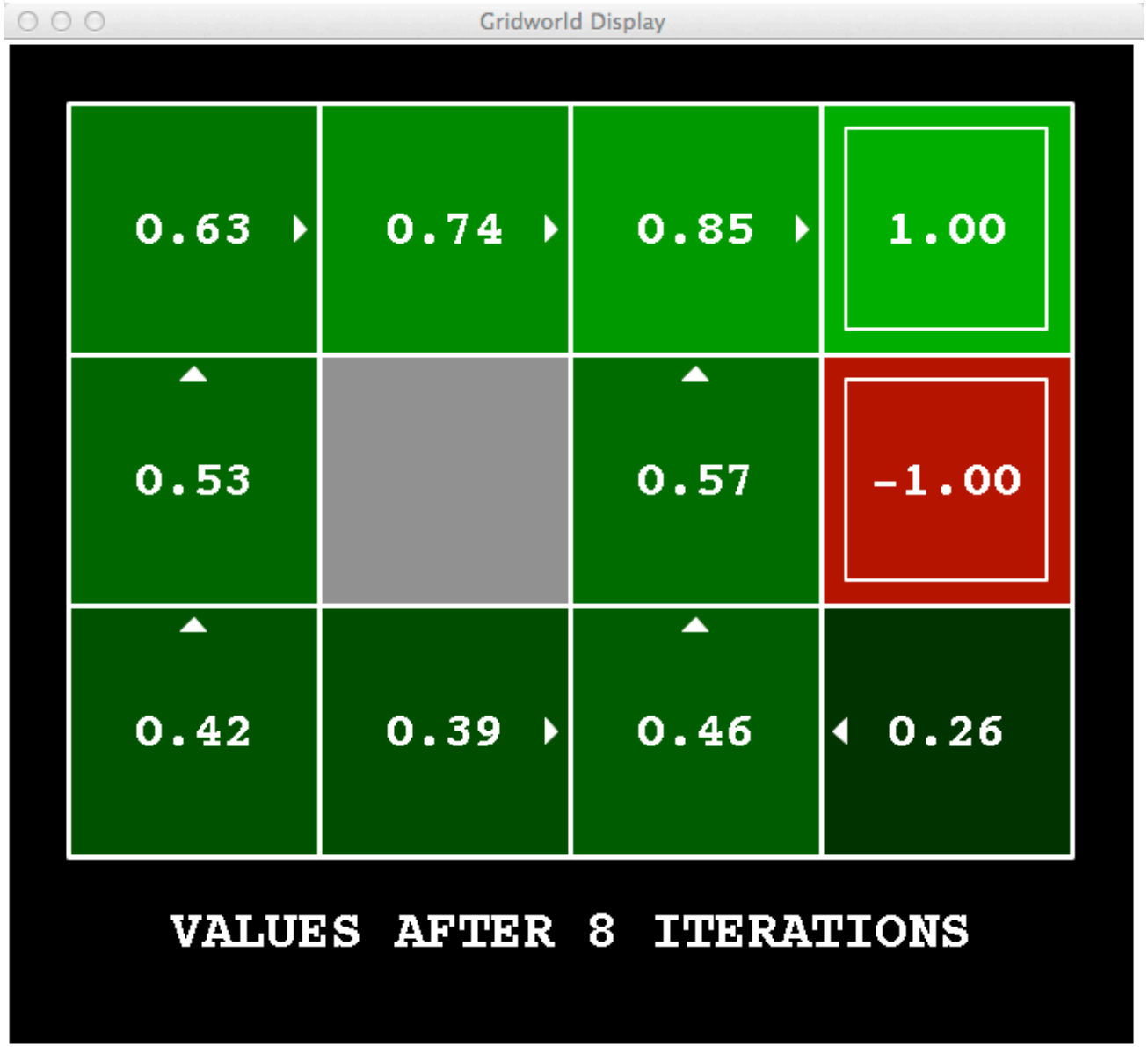
Noise = 0.2
Discount = 0.9
Living reward = 0

K=7



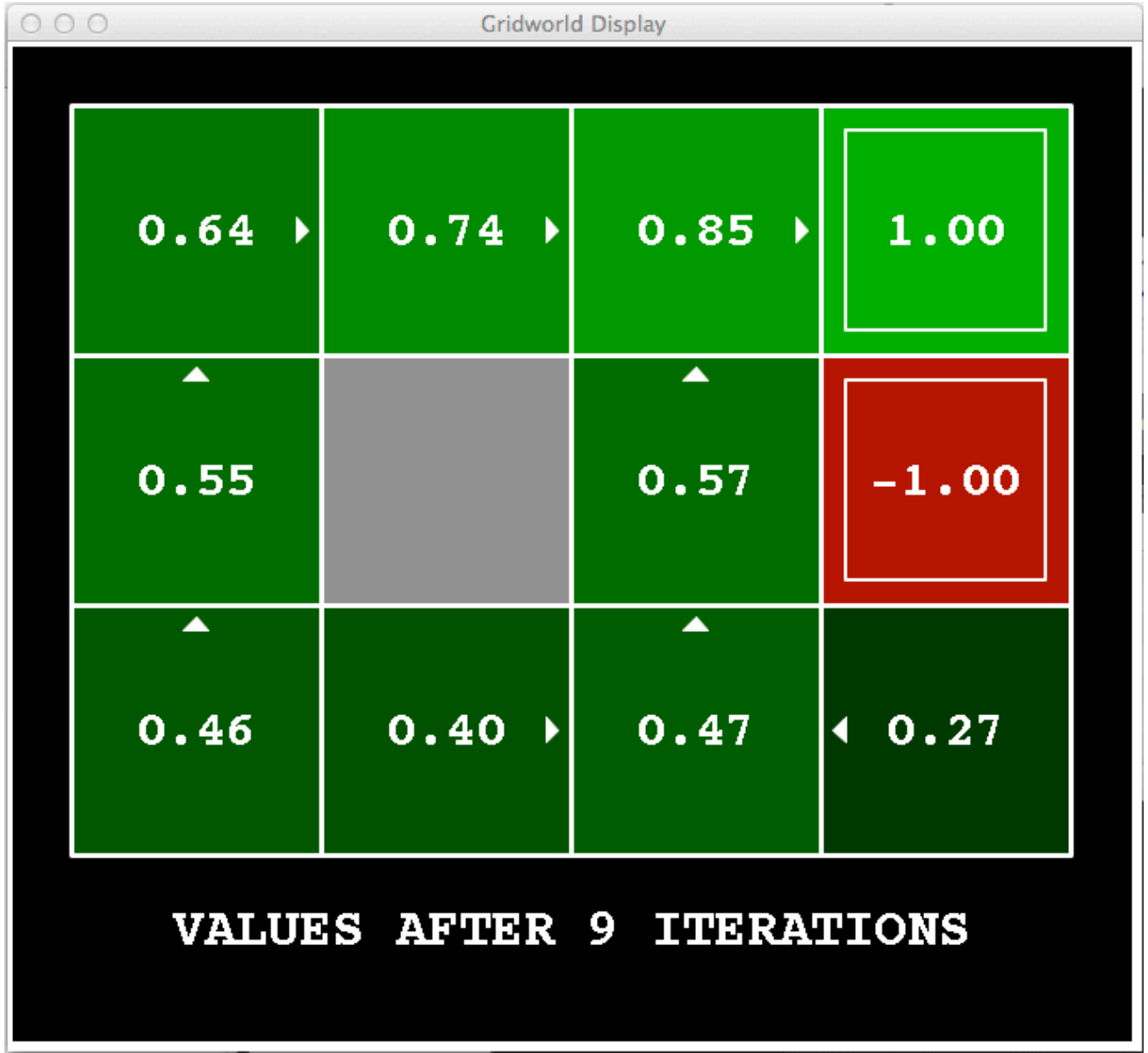
Noise = 0.2
Discount = 0.9
Living reward = 0

K=8



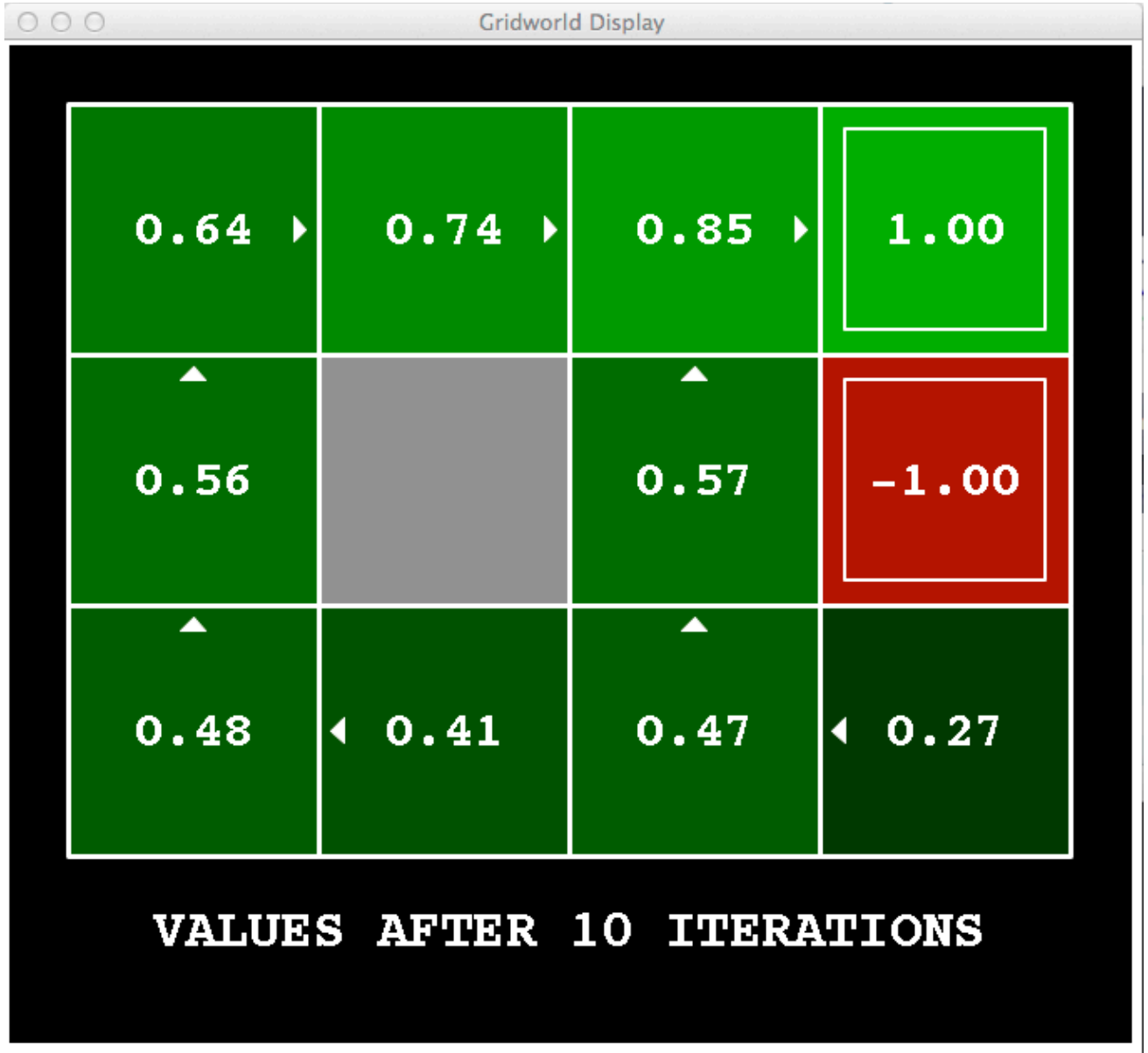
Noise = 0.2
Discount = 0.9
Living reward = 0

K=9



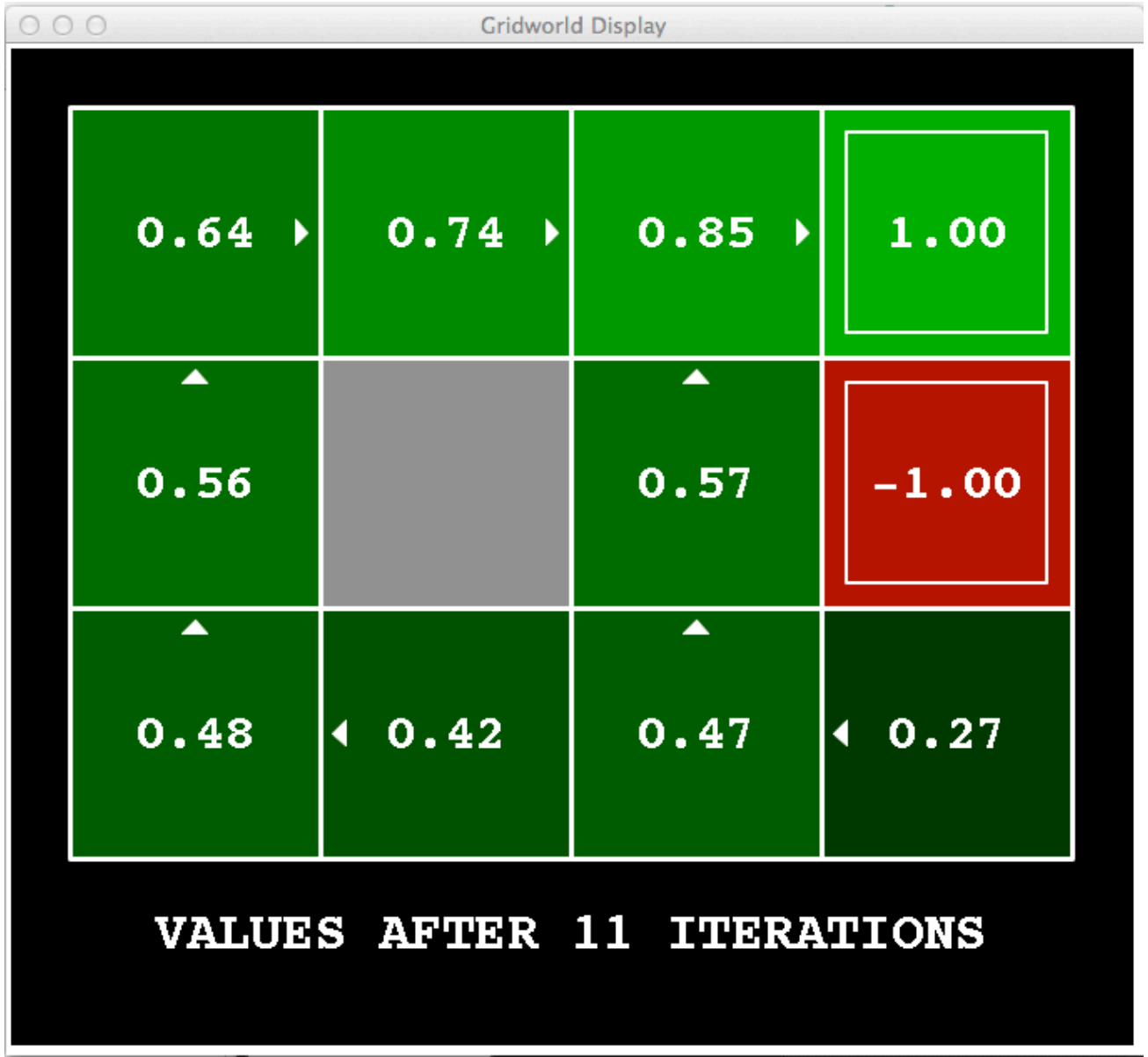
Noise = 0.2
Discount = 0.9
Living reward = 0

K=10



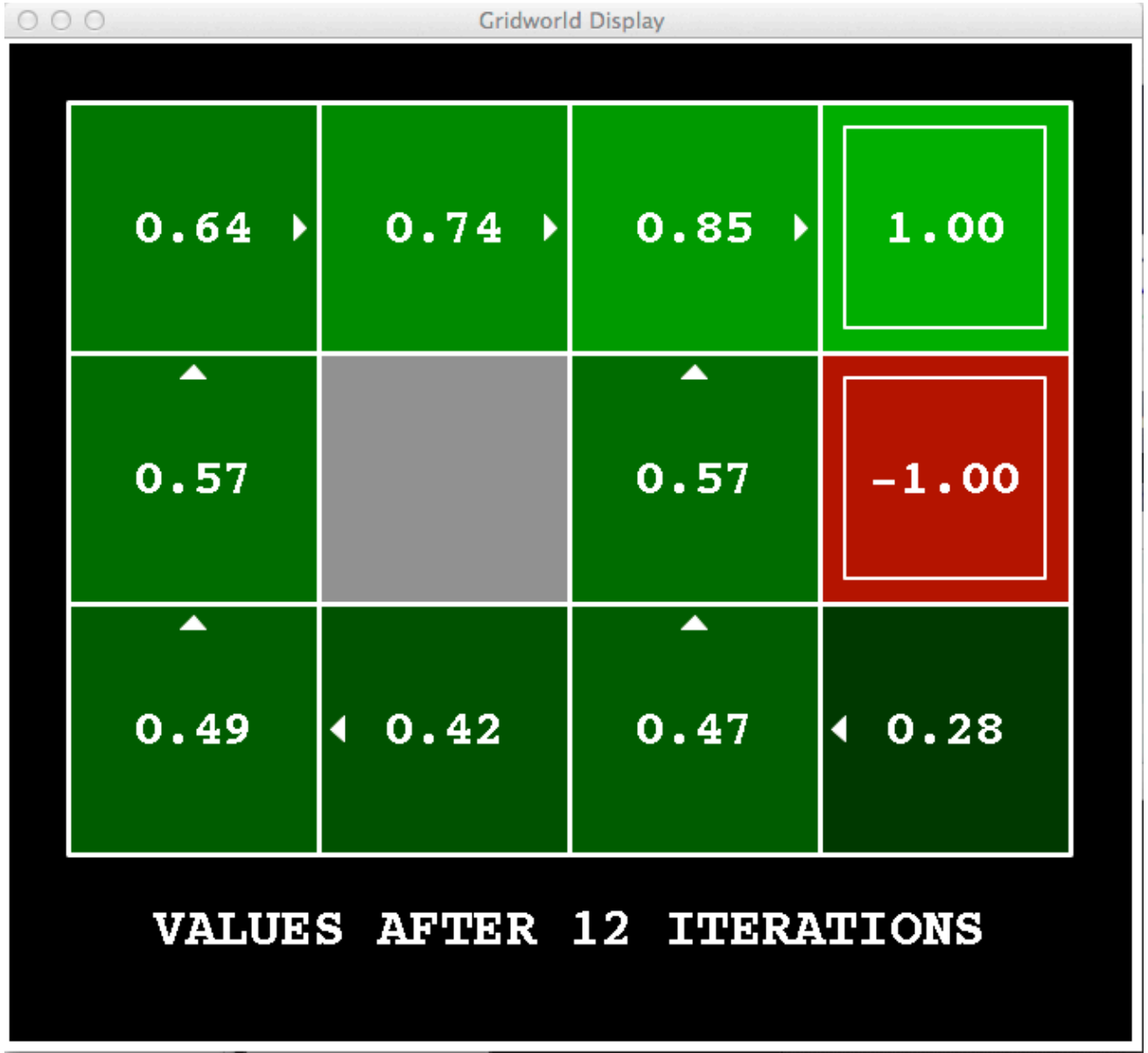
Noise = 0.2
Discount = 0.9
Living reward = 0

K=11



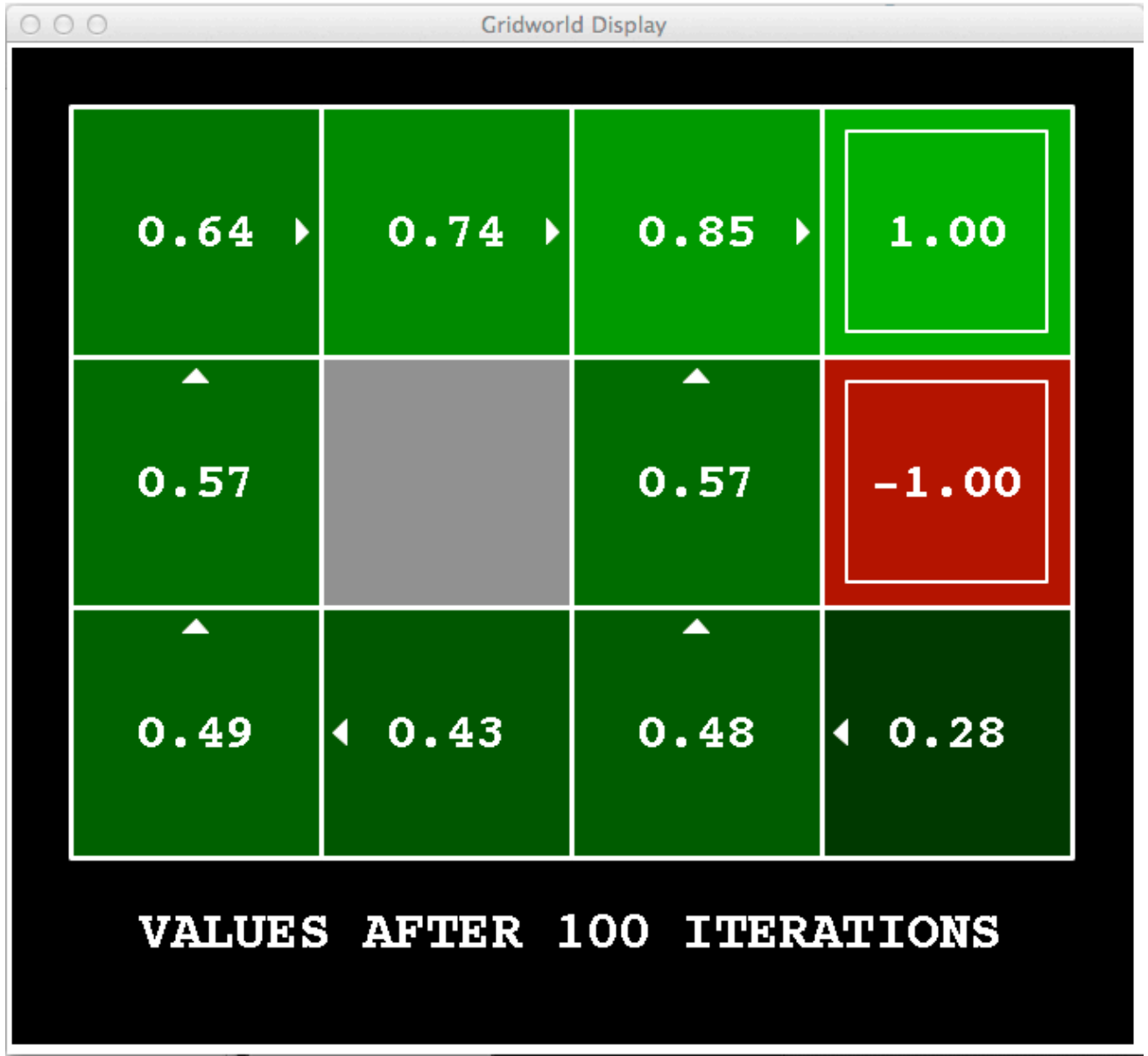
Noise = 0.2
Discount = 0.9
Living reward = 0

K=12



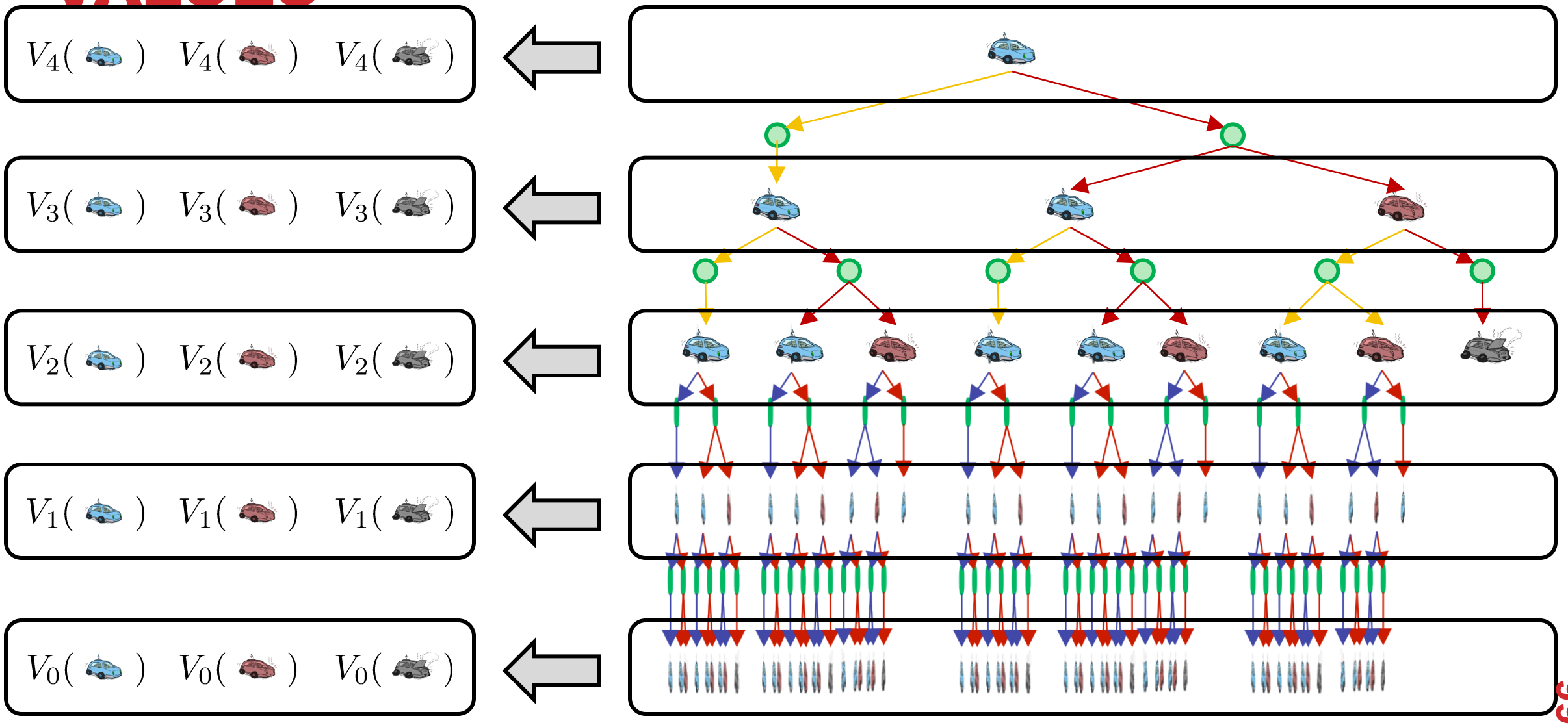
Noise = 0.2
Discount = 0.9
Living reward = 0

K=100

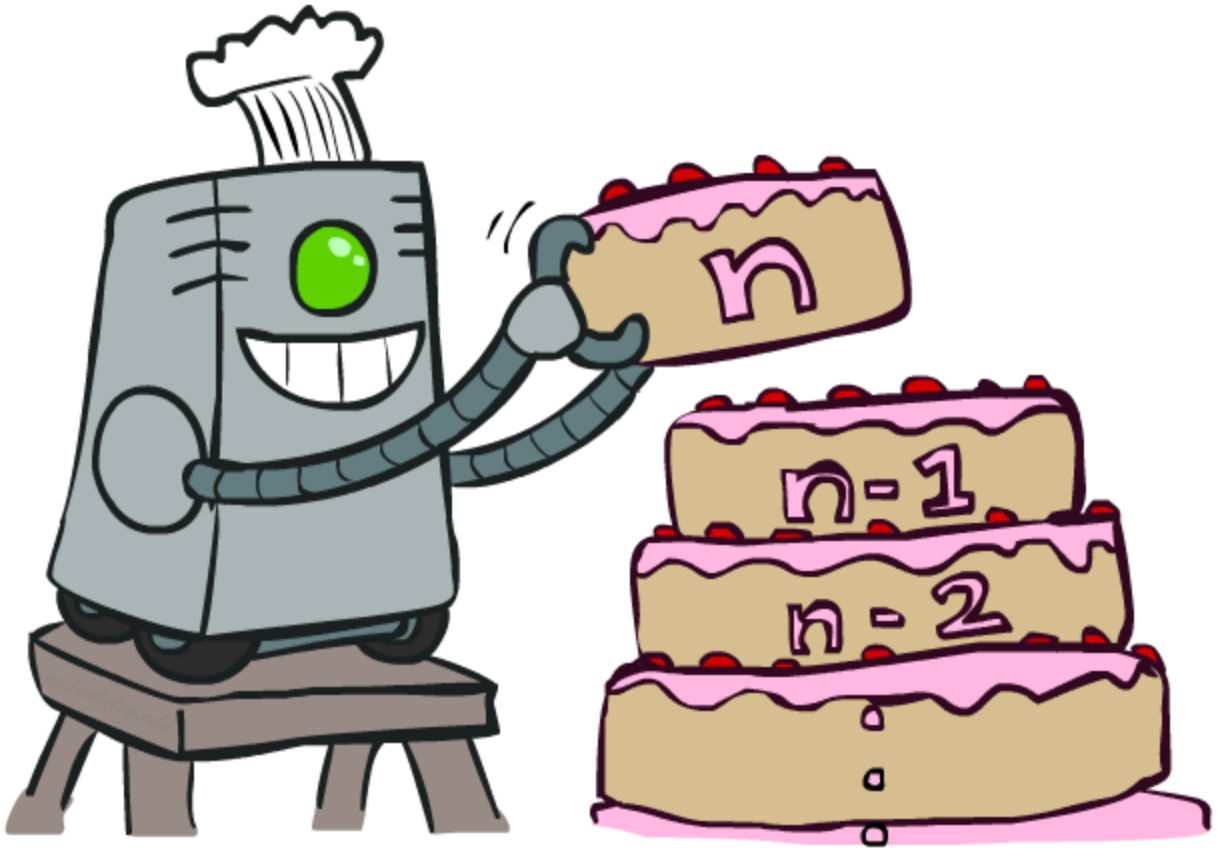


Noise = 0.2
Discount = 0.9
Living reward = 0

COMPUTING TIME-LIMITED VALUES



VALUE ITERATION

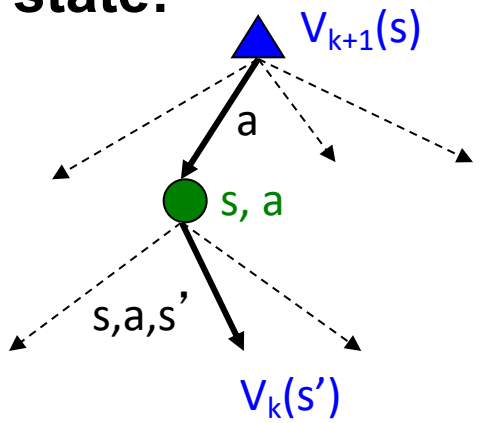


VALUE ITERATION

Start with $V_0(s) = 0$: no time steps left means an expected reward sum of zero

Given vector of $V_k(s)$ values, do one ply of expectimax from each state:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s') \right]$$

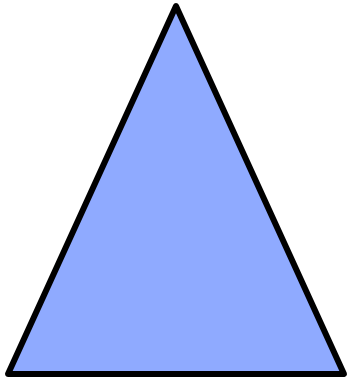


Repeat until convergence

Complexity of each iteration: $O(S^2A)$

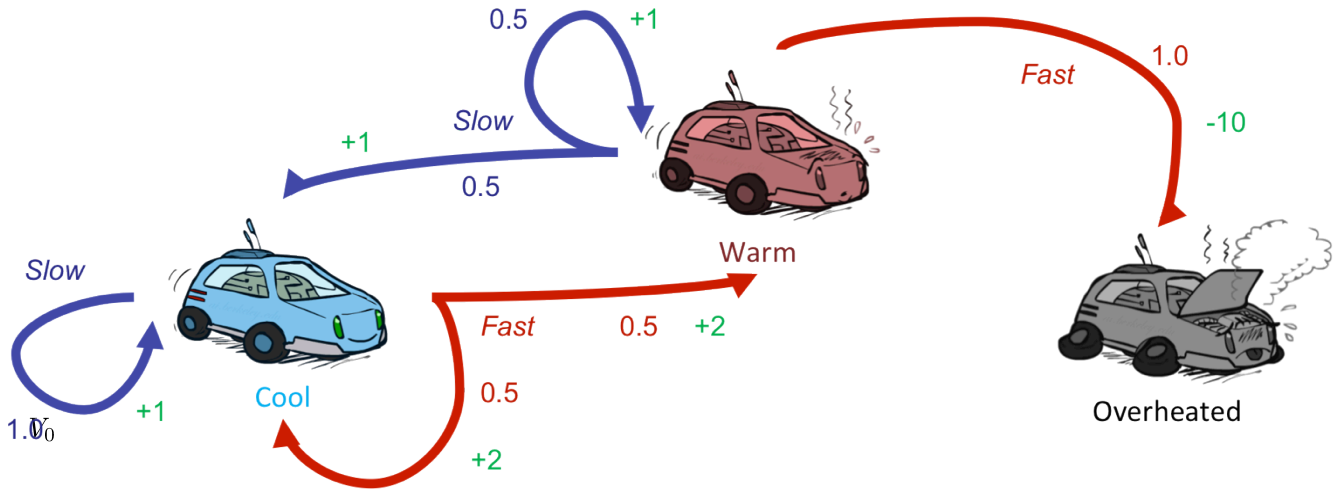
Theorem: will converge to unique optimal values

- Basic idea: approximations get refined towards optimal values
- Policy may converge long before values do



EXAMPLE: VALUE ITERATION

V_2	3.5	2.5	0
V_1	2	1	0
	0	0	0



Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s') \right]$$

CONVERGENCE

How do we know the V_k vectors are going to converge?

Case 1: If the tree has maximum depth M, then V_M holds the actual untruncated values

Case 2: If the discount is less than 1

- Sketch: For any state V_k and V_{k+1} can be viewed as depth k+1 expectimax results in nearly identical search trees
- The difference is that on the bottom layer, V_{k+1} has actual rewards while V_k has zeros
- That last layer is at best all R_{MAX}
- It is at worst R_{MIN}
- But everything is discounted by γ^k that far out
- So V_k and V_{k+1} are at most γ^k max|R| different
- So as k increases, the values converge