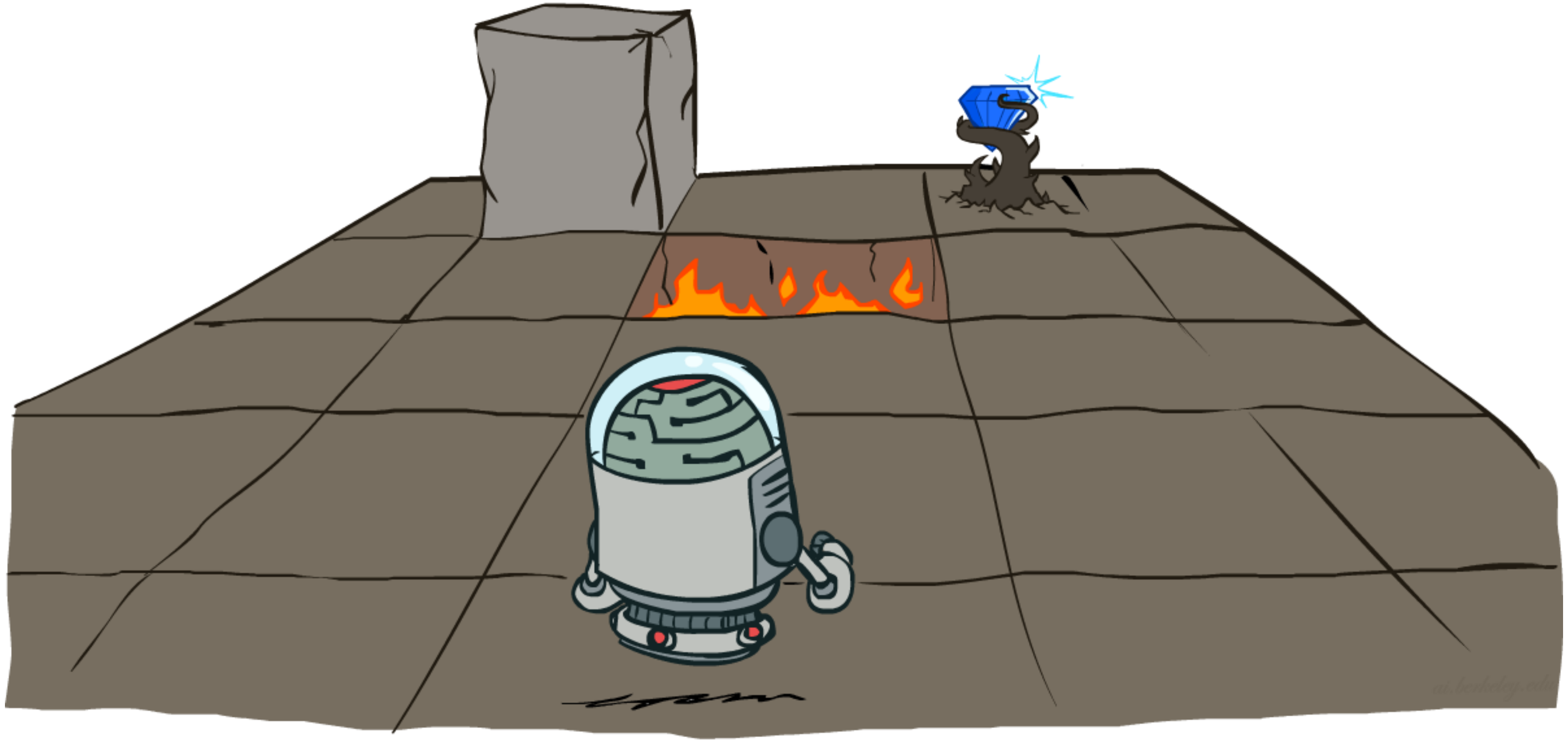
CS 188: Artificial Intelligence

Markov Decision Processes



Instructor: Stuart Russell and Dawn Song

University of California, Berkeley

Recap: Markov Decision Process (MDP)

- What is a Markov Decision Process?

Andrey Markov
(1856-1922)

Recap: Markov Decision Process (MDP)

- What is a Markov Decision Process?
  - State transition model is markov
  - Utility function is additive discounted rewards
- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition model $T(s, a, s')$
    - Probability that a from s leads to s', i.e., $P(s' \mid s, a)$
  - A reward function $R(s, a, s')$ for each transition
  - A start state
  - Possibly a terminal state (or absorbing state)
  - Utility function which is additive discounted rewards

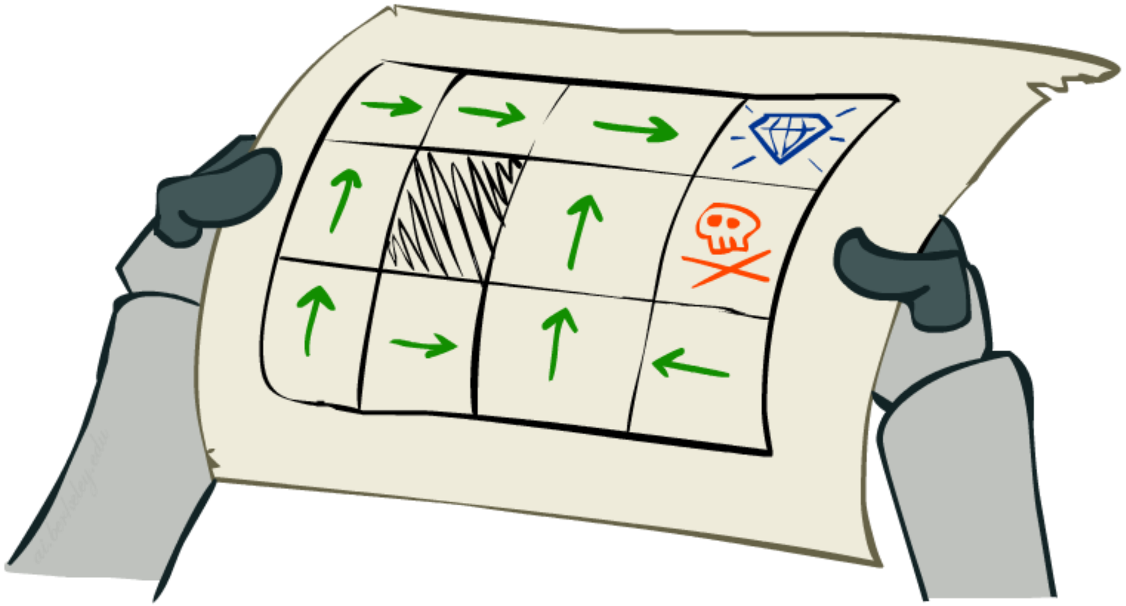
$$U_h([s_0, a_0, s_1, a_1, s_2, \ldots]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \cdots$$

where $\gamma \in [0,1]$ is the ***discount factor***

Andrey Markov
(1856-1922)

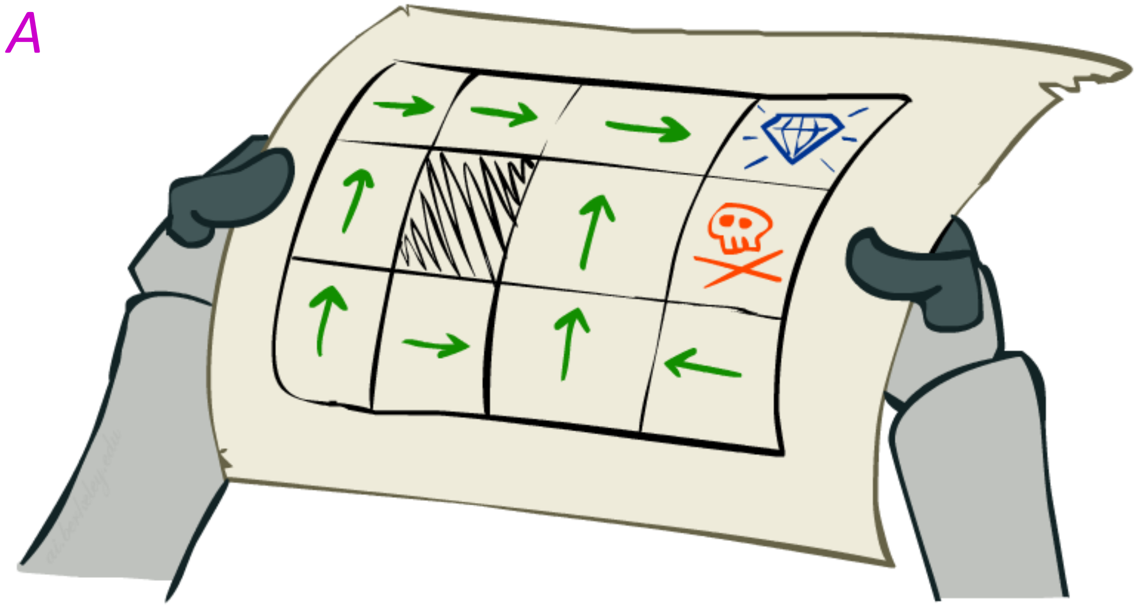
Recap: Policies

- What is a policy?



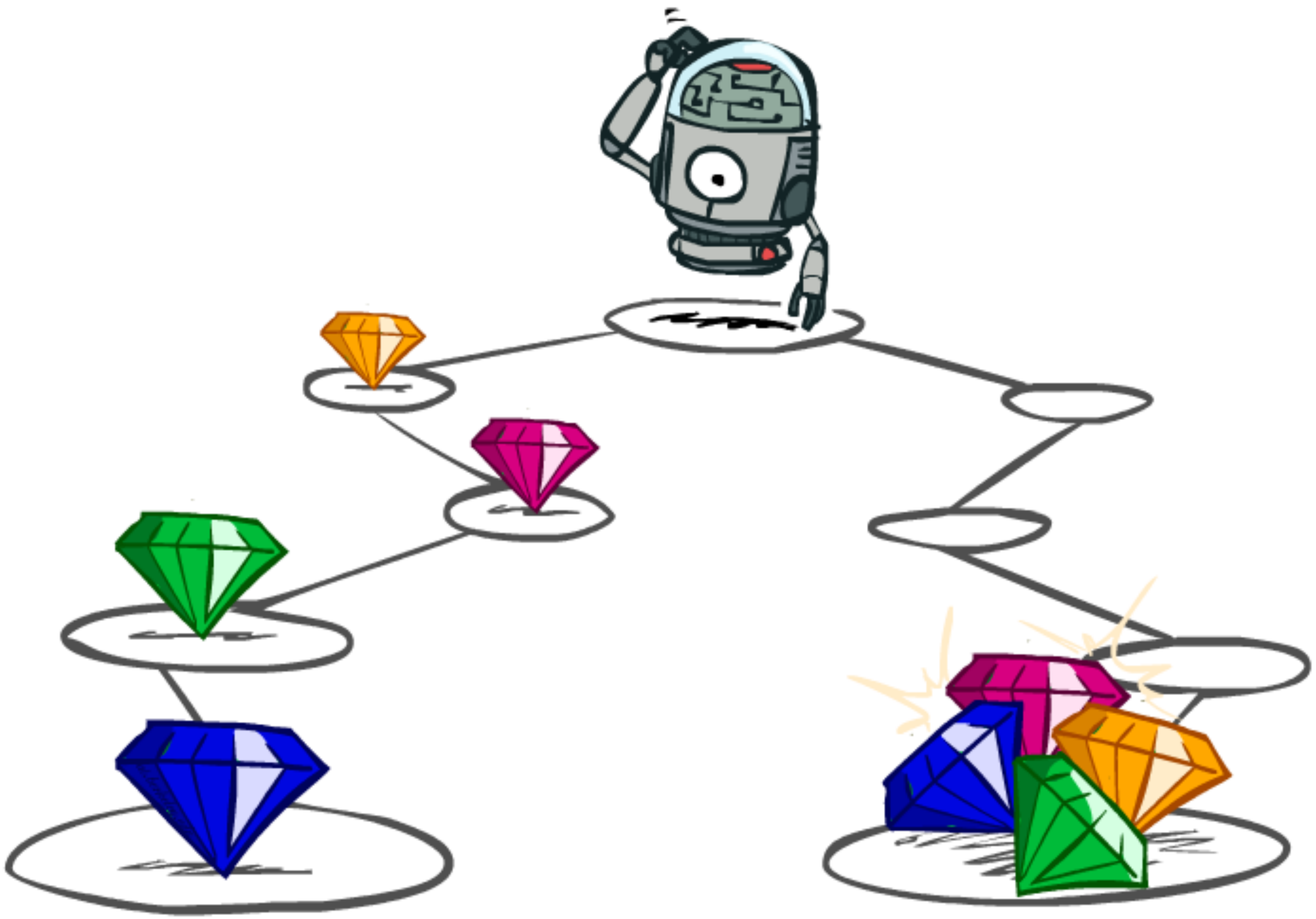
Recap: Policies

- A policy π gives an action for each state, $\pi: S \rightarrow A$



- For MDPs, we want an optimal ***policy*** $\pi^*: S \rightarrow A$
  - An optimal policy maximizes expected utility

Recap: Utilities of Sequences



Recap: Stationary Preferences

- Theorem: if we assume ***stationary preferences***:

  $[s_0, a_0, s_1, a_1, s_2, \ldots] \succ [s'_0, a'_0, s'_1, a'_1, s'_2, \ldots]$, $s_0 = s'_0$, $a_0 = a'_0$, and $s_1 = s'_1$

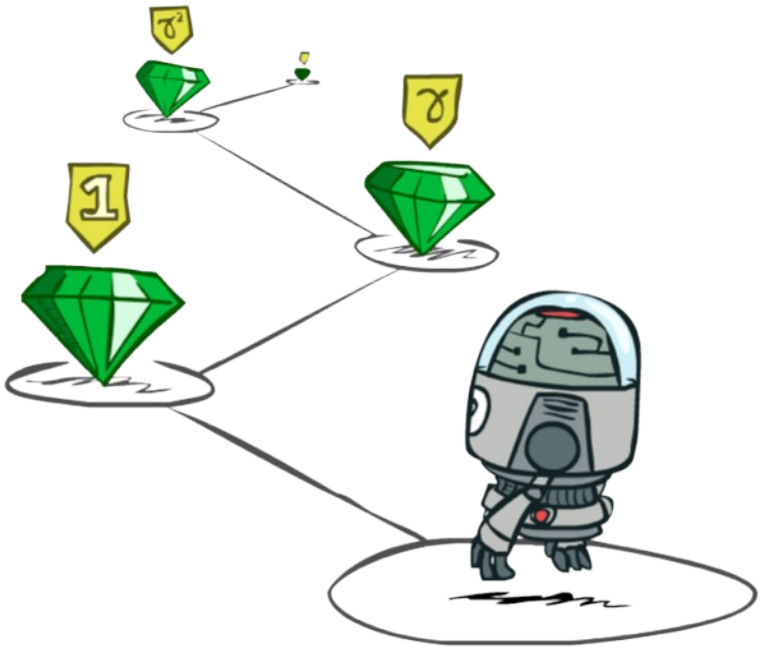
  $\Leftrightarrow \quad [s_1, a_1, s_2, \ldots] \succ [s'_1, a'_1, s'_2, \ldots]$

  then there is only one way to define utilities:

  - ***Additive discounted utility***:

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

where $\gamma \in [0,1]$ is the ***discount factor***



Discounting

Worth r now

Worth γr next step

Worth $\gamma^2 r$ in two steps

- Discounting conveniently solves the problem of infinite reward streams!
  - Geometric series: $1 + \gamma + \gamma^2 + \ldots = 1/(1 - \gamma)$
  - Assume rewards bounded by $\pm R_{max}$
  - Then $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$ is bounded by $\pm R_{max}/(1 - \gamma)$
- (Another solution: environment contains a ***terminal state***; ***and*** agent reaches it with probability 1)

Quiz: Discounting

- Given:

10				1
a	b	c	d	e

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

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

10				1

- Quiz 2: For $\gamma = 0.1$, what is the optimal policy?

10				1

- Quiz 3: For which γ are West and East equally good when in state d?

The utility of a policy

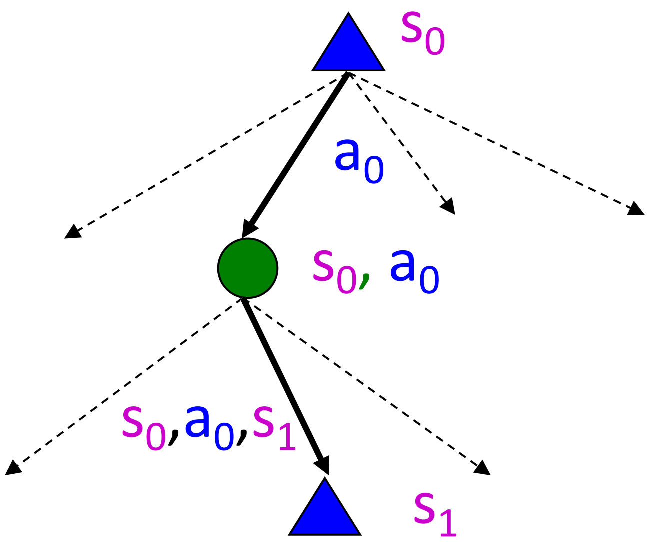
- Executing a policy π from any state s_0 generates a sequence

  $s_0, \pi(s_0), s_1, \pi(s_1), s_2, \ldots$
- This corresponds to a sequence of rewards

  $R(s_0, \pi(s_0), s_1), R(s_1,\pi(s_1), s_2), \ldots$
- This reward sequence happens with probability

  $P(s_1 \mid s_0, \pi(s_0)) \times P(s_2 \mid s_1, \pi(s_1)) \times \ldots$
- The value (expected utility) of π in s_0 is written $U^\pi(s_0)$
  - It's the sum over all possible state sequences of (discounted sum of rewards) x (probability of state sequence)

$$U^\pi(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})\right]$$



Optimal Quantities

- The optimal policy:

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

  Gives highest $U^{\pi}(s)$ for any π

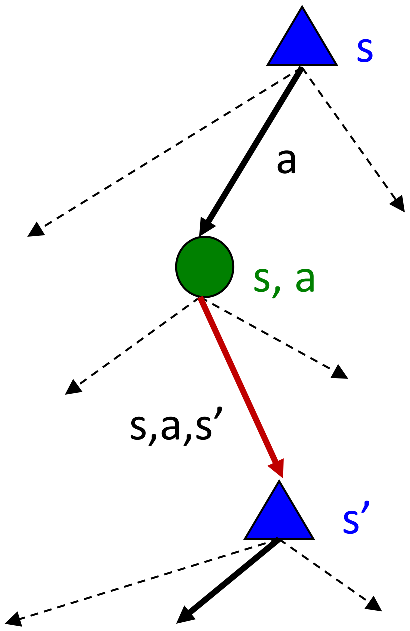
- The value (utility) of a state s:

  $U^*(s) = U^{\pi^*}(s)$ = expected utility starting in s and acting optimally

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

  $Q^*(s,a)$ = expected utility of taking action a in state s and (thereafter) acting optimally

  $U^*(s)$? $Q^*(s,a)$



s is a *state*

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

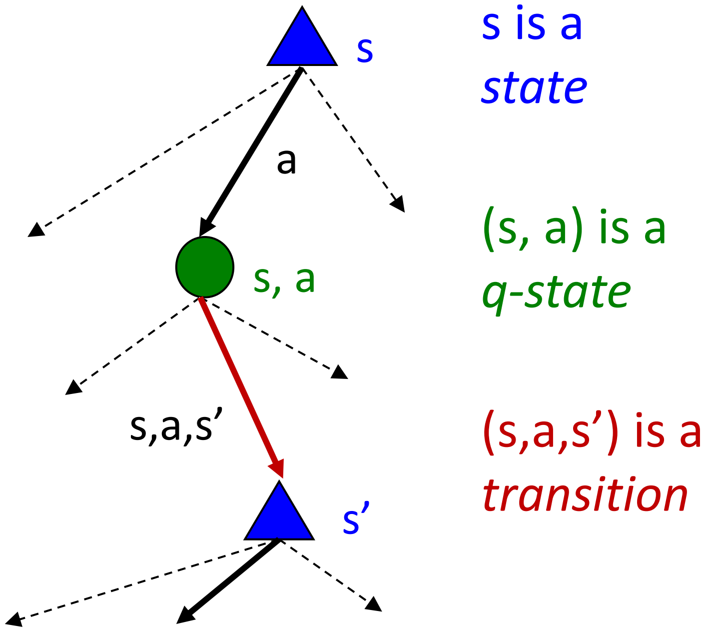
(s,a,s') is a *transition*

Optimal Quantities

- The optimal policy:
  $\pi^*(s)$ = optimal action from state s
  Gives highest $U^\pi(s)$ for any π

- The value (utility) of a state s:
  $U^*(s) = U^{\pi^*}(s)$ = expected utility starting in s and acting optimally

- The value (utility) of a q-state (s,a):
  $Q^*(s,a)$ = expected utility of taking action a in state s and (thereafter) acting optimally
  $U^*(s) = \max_a Q^*(s,a)$



Bellman equations (Shapley, 1953)

- The value/utility of a state is
  - The expected reward for the next transition plus the discounted value/utility of the next state, assuming the agent chooses the optimal action

- Hence we have a recursive definition of value (Bellman equation):

$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U(s')]$$

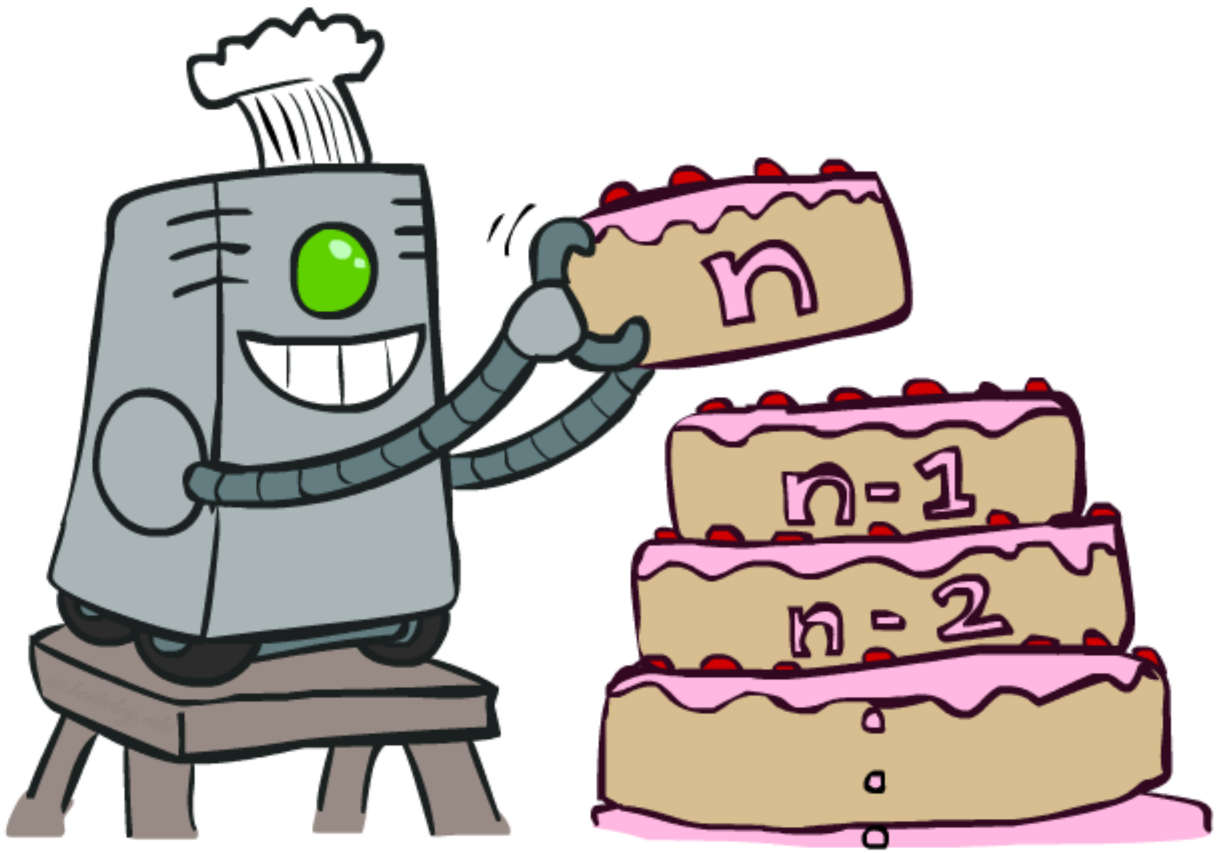
- Similarly, Bellman equation for Q-functions

$$\begin{aligned} Q(s,a) &= \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma\, U(s')] \\ &= \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')] \end{aligned}$$

Solving MDPs

- Value iteration
- Policy iteration

Value Iteration



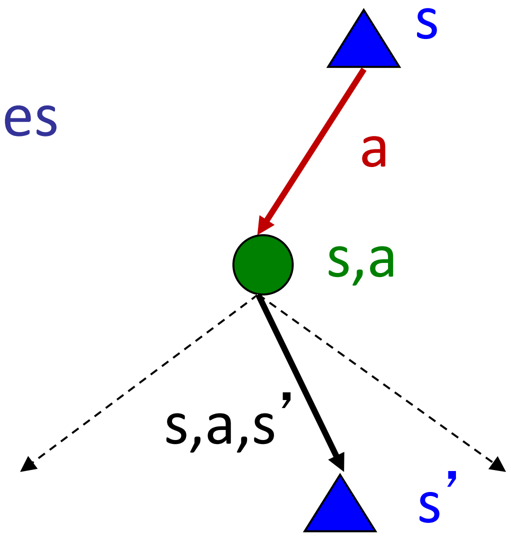
Value Iteration

- Start with (say) $U_0(s) = 0$ and some termination parameter ε
- Repeat until convergence (i.e., until all updates smaller than ε)
  - Do a ***Bellman update*** (essentially one ply of expectimax) from each state:

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

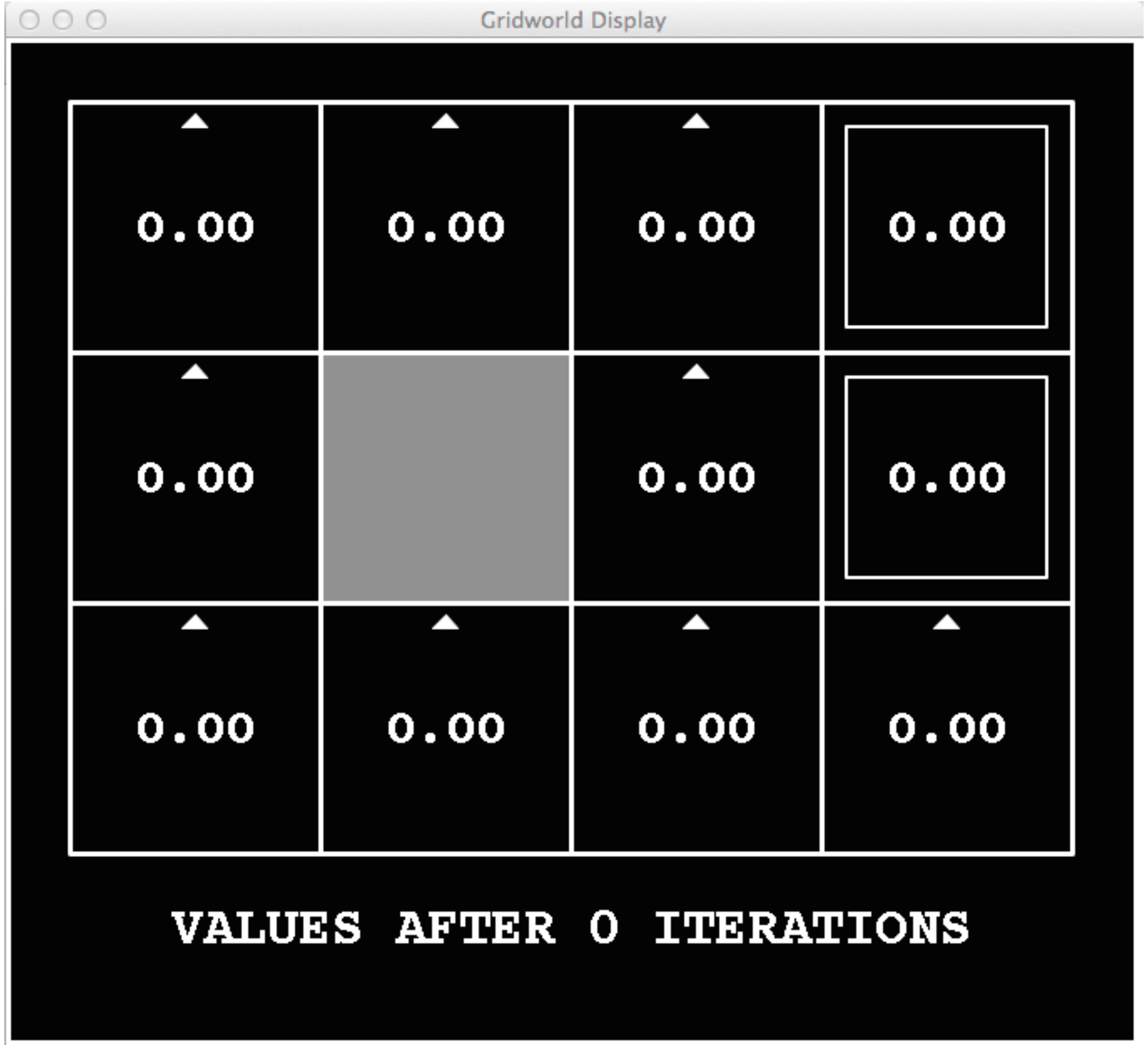
$\boldsymbol{U} \leftarrow \mathrm{B}\boldsymbol{U}$

- Theorem: will converge to unique optimal values
- Runtime per iteration?
  - Complexity of each iteration: $O(S^2A)$



k=0

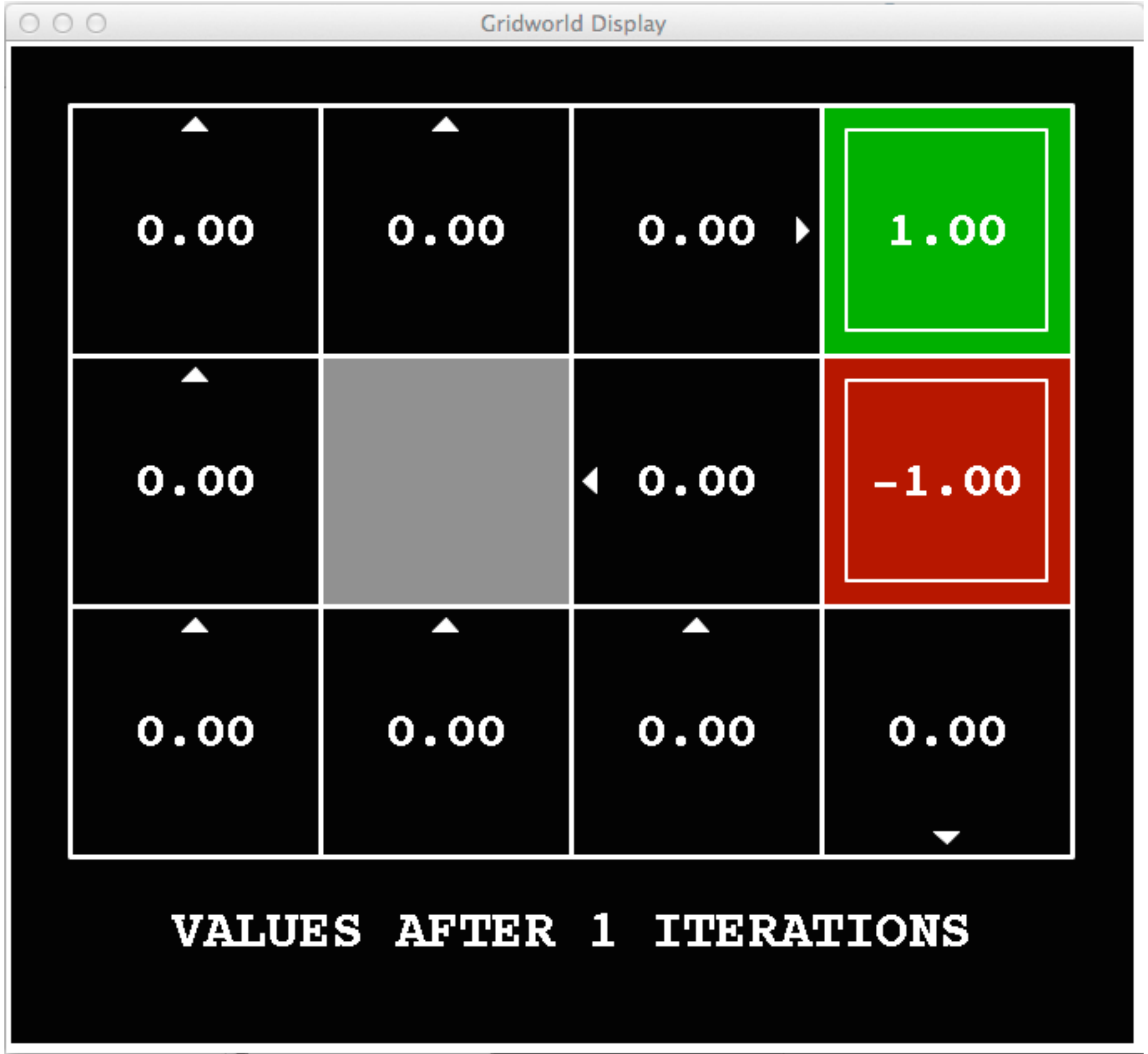
$$U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U_i(s')]$$



Noise = 0.2
Discount = 0.9
Living reward = 0

k=1

$$U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U_i(s')]$$



Noise = 0.2
Discount = 0.9
Living reward = 0

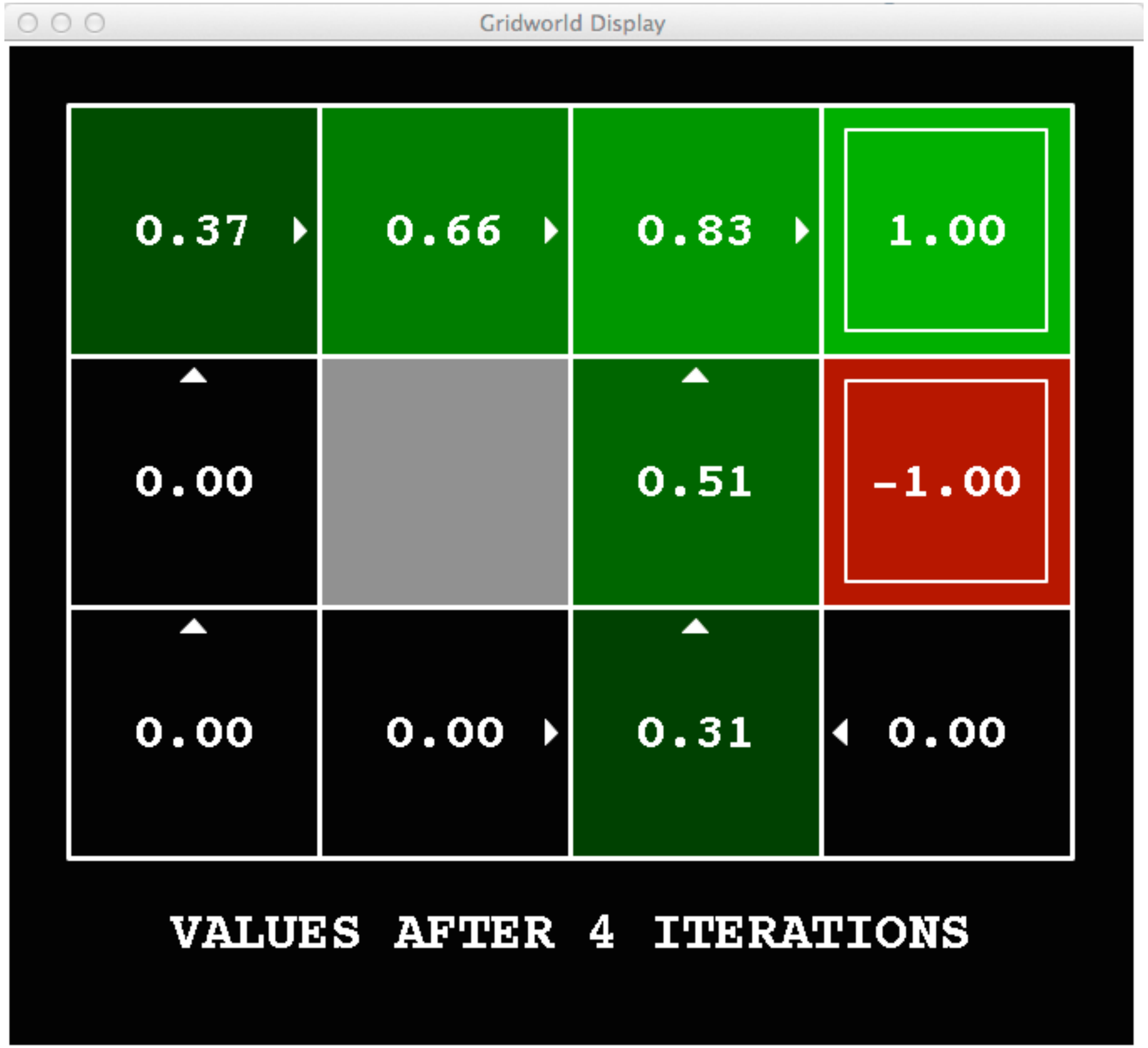
k=2

Noise = 0.2
Discount = 0.9
Living reward = 0

k=3

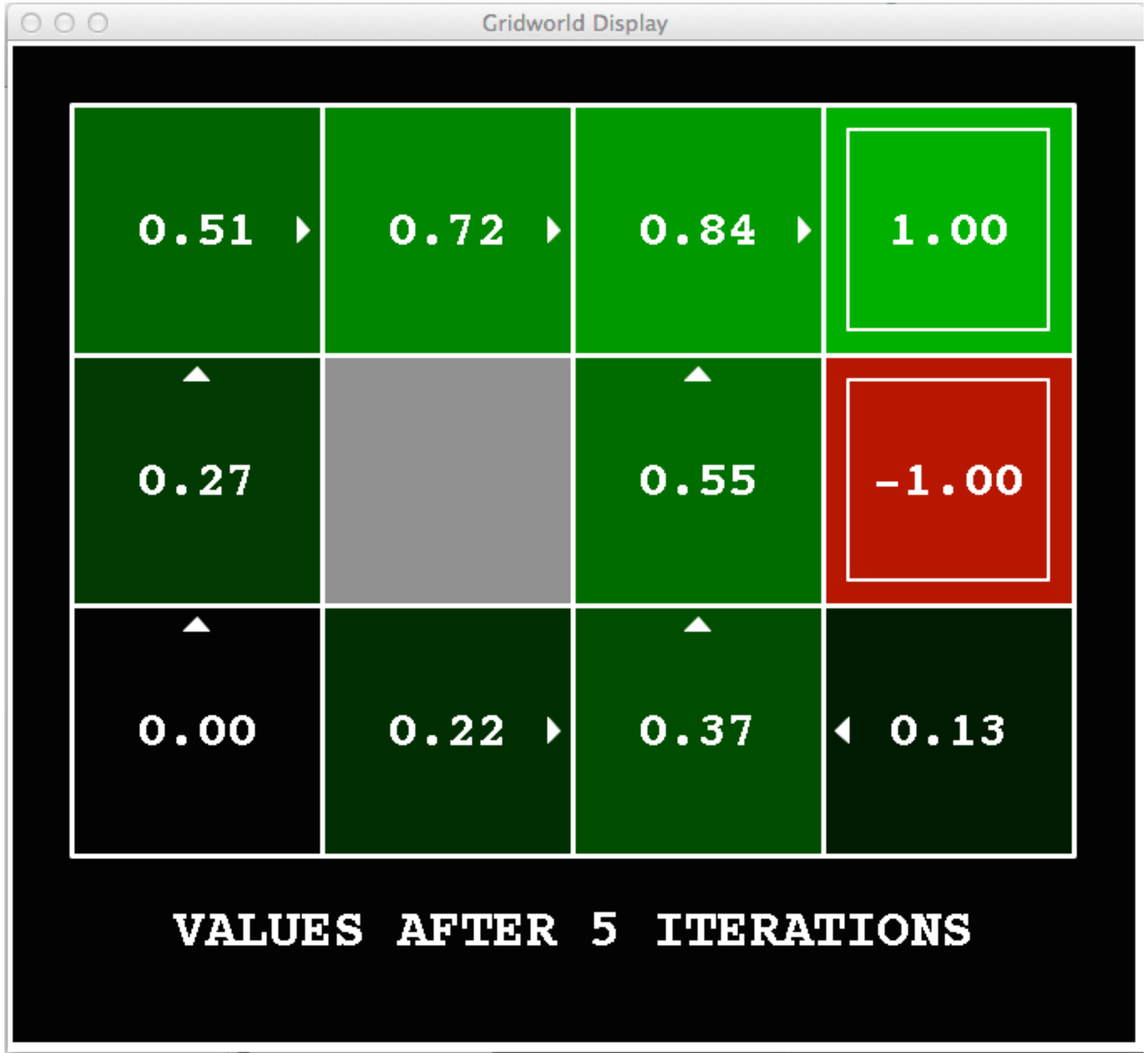
Noise = 0.2
Discount = 0.9
Living reward = 0

k=4



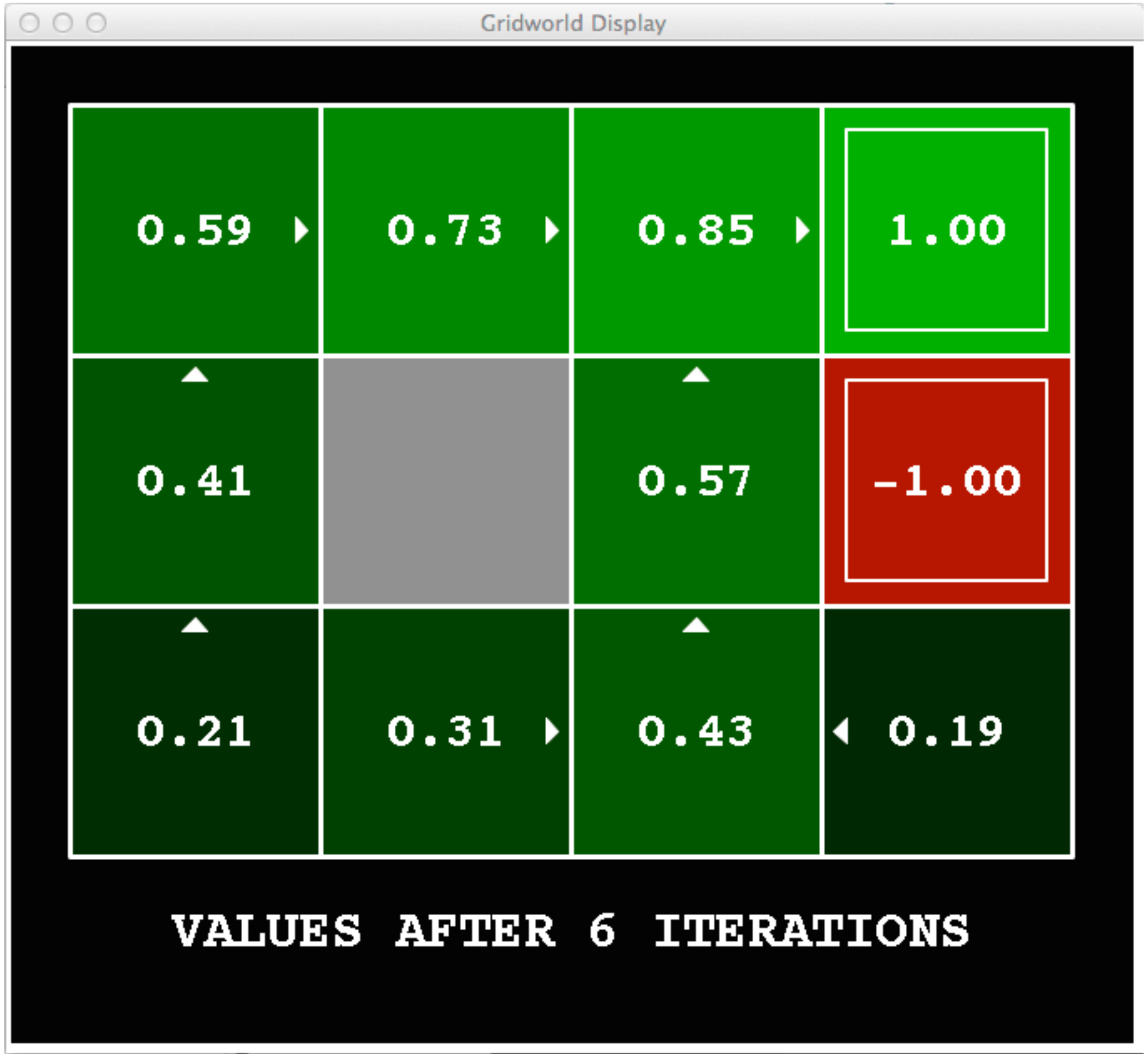
Noise = 0.2
Discount = 0.9
Living reward = 0

k=5



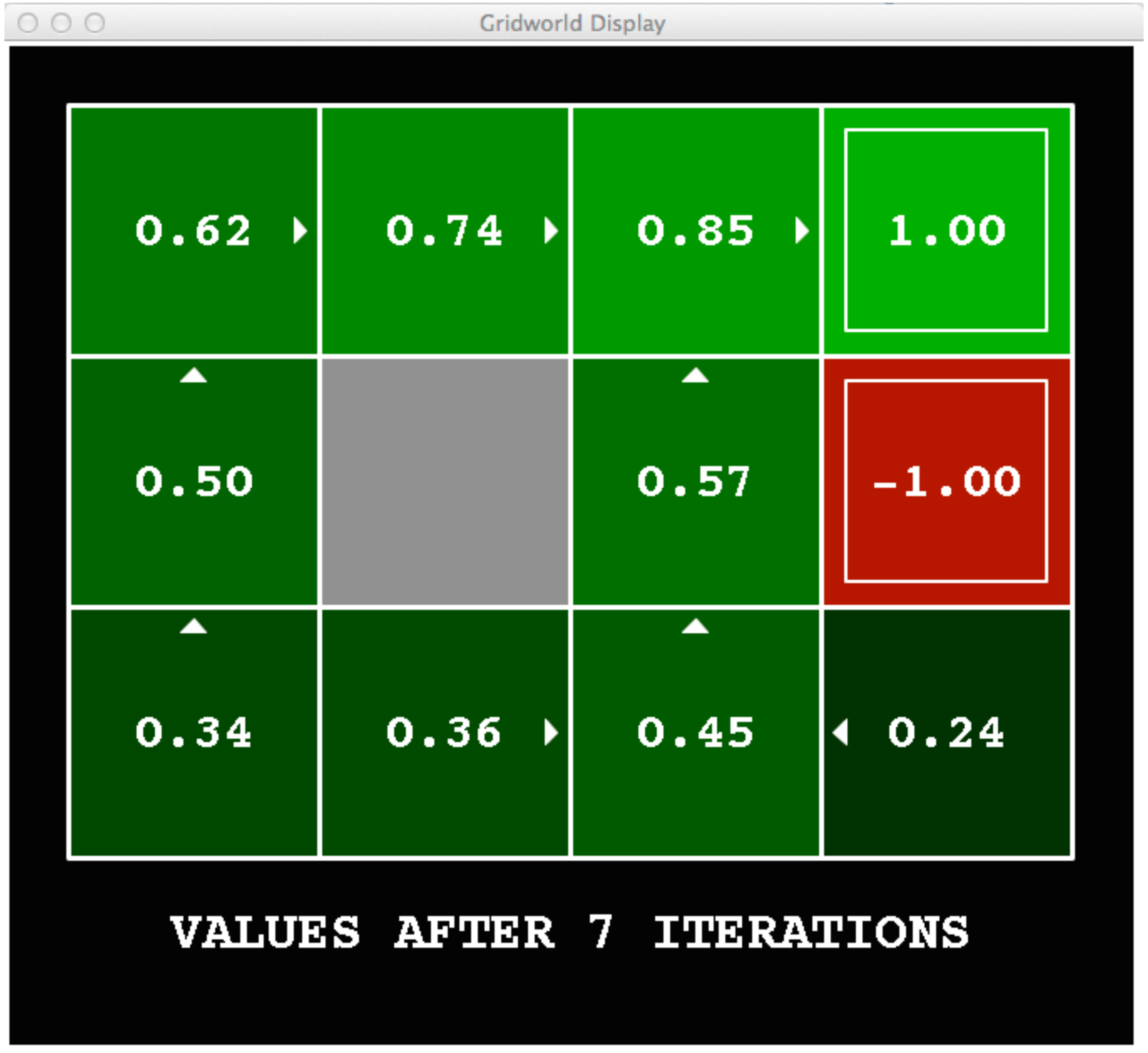
Noise = 0.2
Discount = 0.9
Living reward = 0

k=6



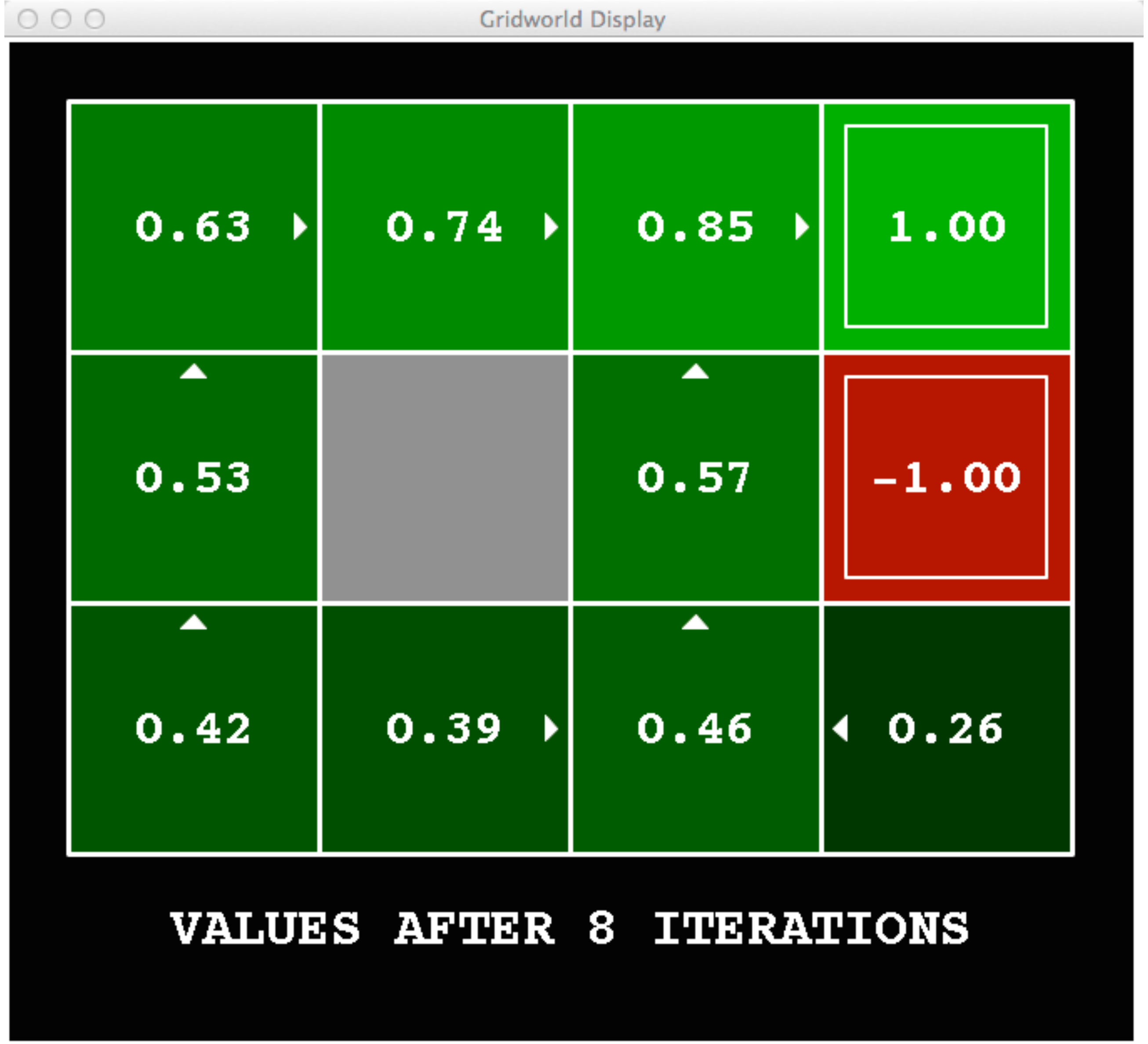
Noise = 0.2
Discount = 0.9
Living reward = 0

k=7



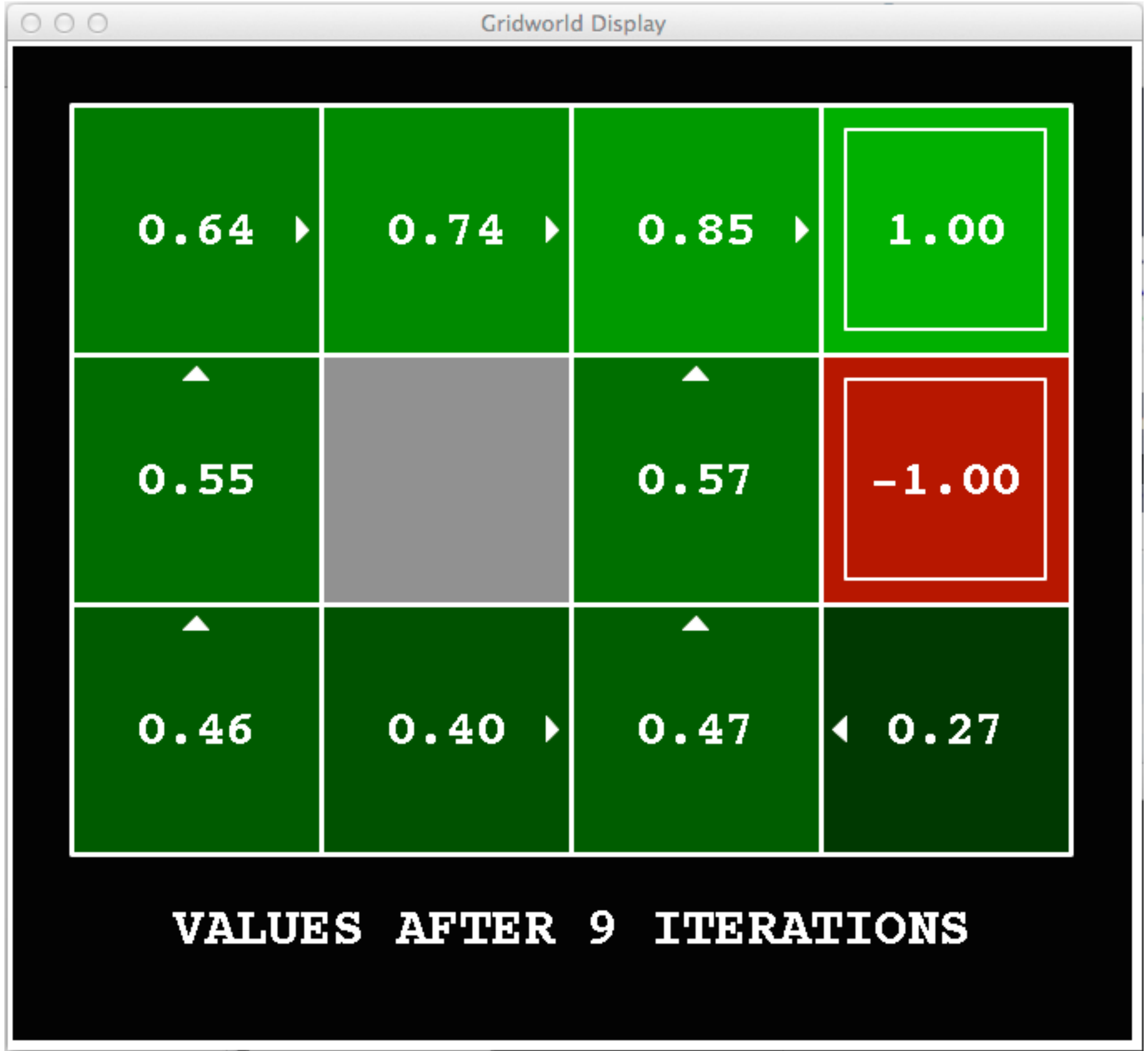
Noise = 0.2
Discount = 0.9
Living reward = 0

k=8



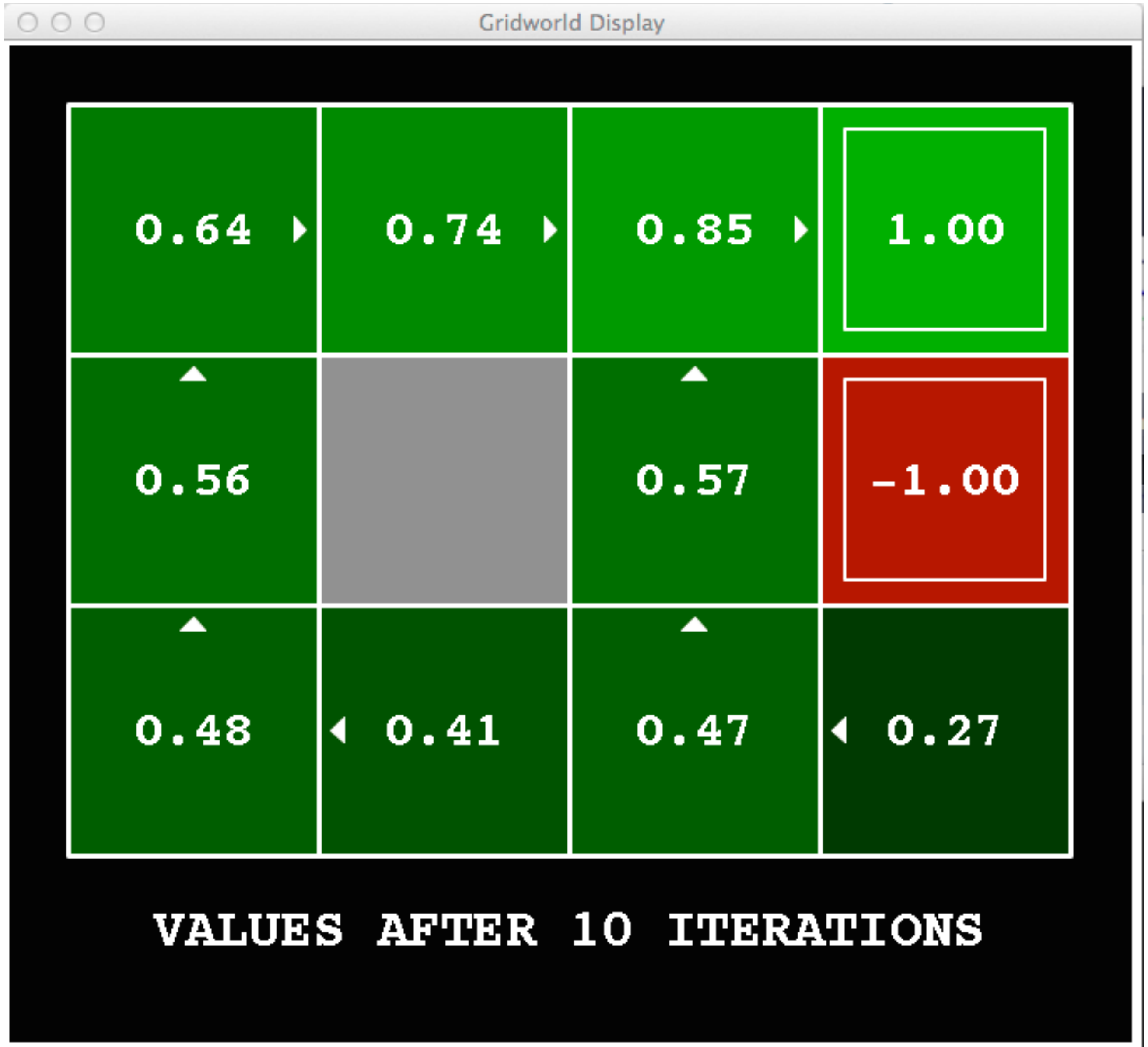
Noise = 0.2
Discount = 0.9
Living reward = 0

k=9



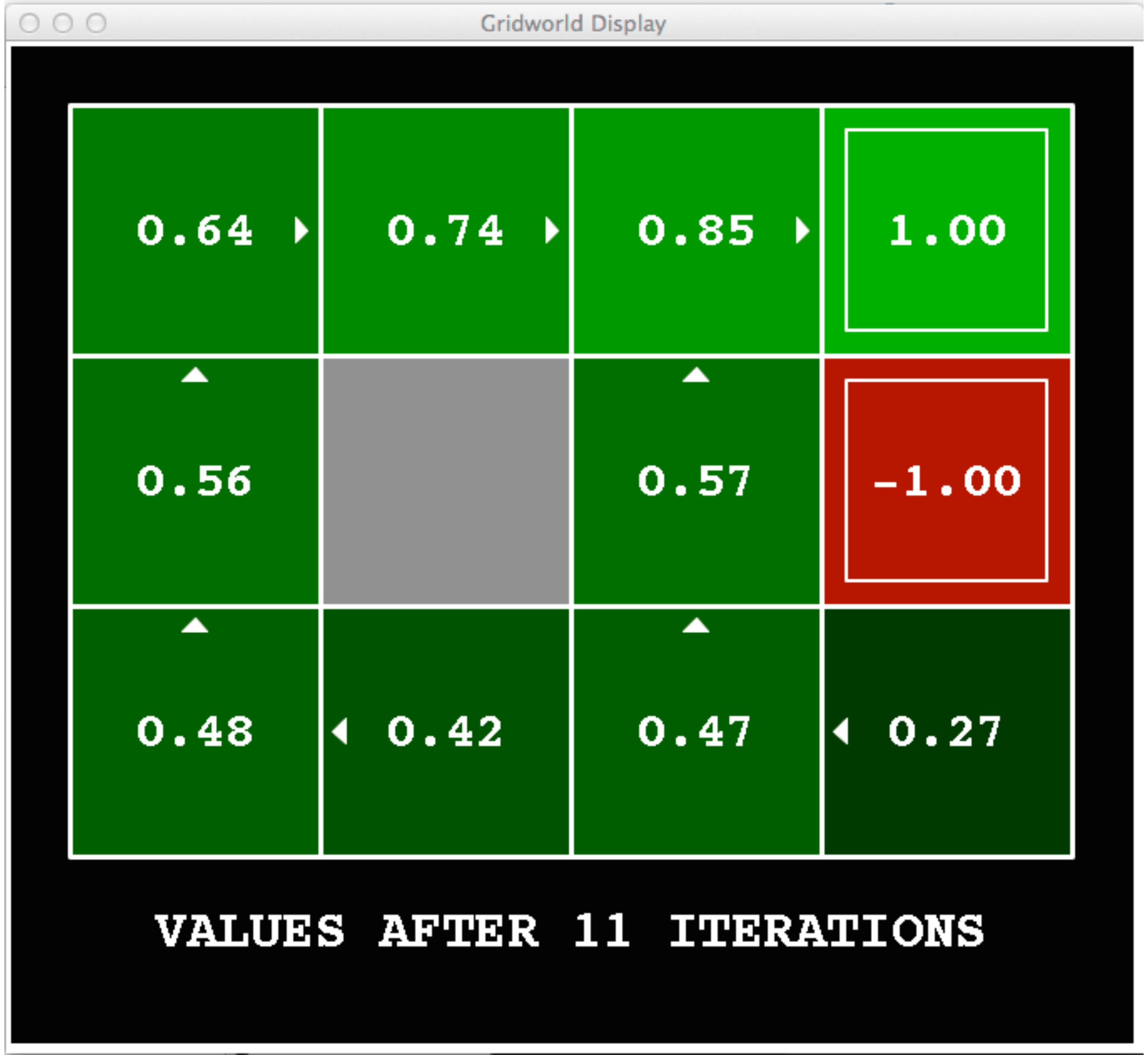
Noise = 0.2
Discount = 0.9
Living reward = 0

k=10



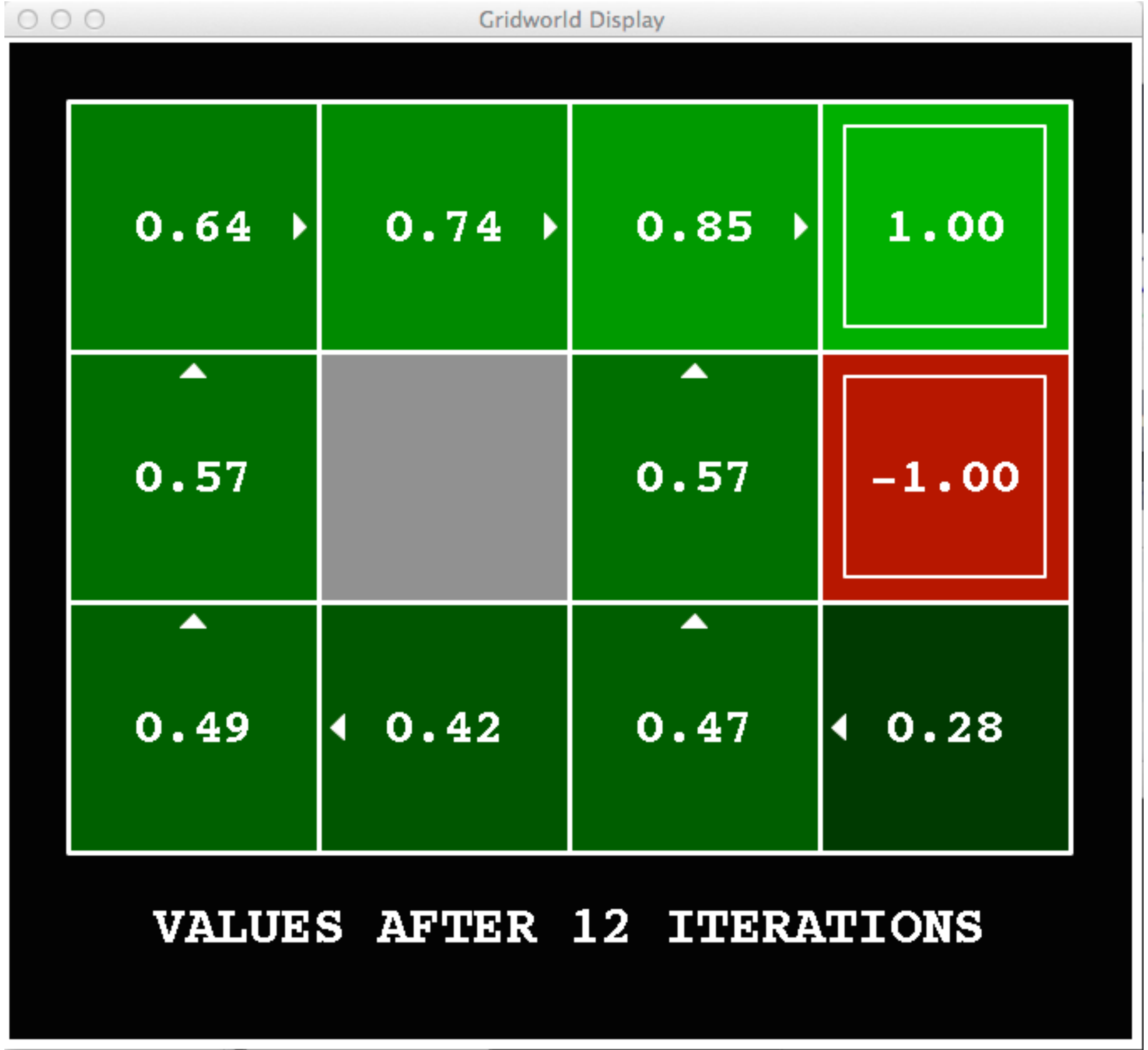
Noise = 0.2
Discount = 0.9
Living reward = 0

k=11



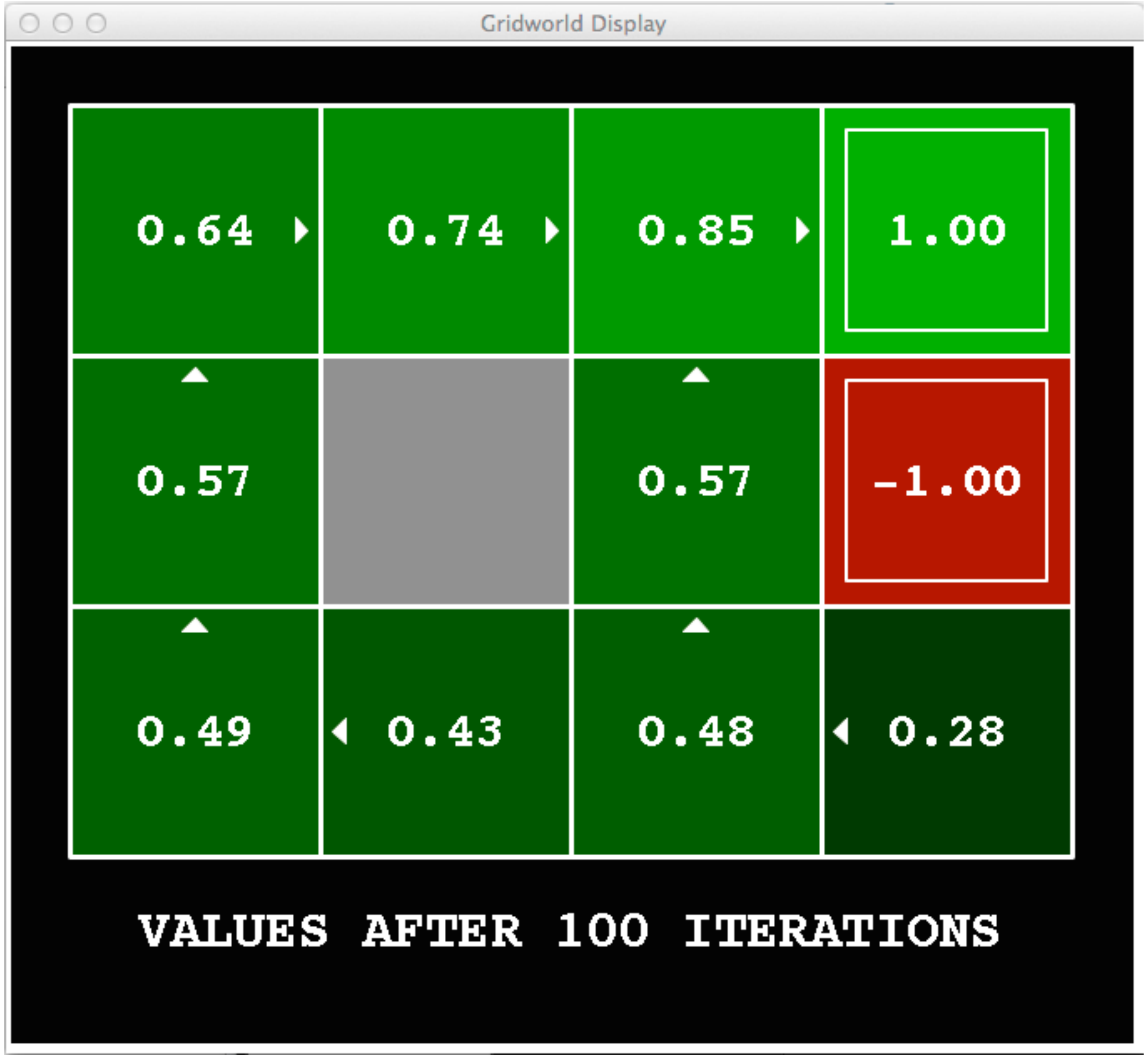
Noise = 0.2
Discount = 0.9
Living reward = 0

k=12



Noise = 0.2
Discount = 0.9
Living reward = 0

k=100



Noise = 0.2
Discount = 0.9
Living reward = 0