# COL778: Principles of Autonomous Systems
# Semester II, 2023-24

## Markov Decision Processes

Rohan Paul

# Outline

- Last Class
  - State Estimation
- This Class
  - Sequential decision-making under uncertainty
    - Markov Decision Processes
- Reference Material
  - Primary reference are the lecture notes. For basic background refer to AIMA Classical Planning Ch. 17 (Sec 17.1 - 17.3). Other reference is Sutton and Barto, Reinforcement Learning Ch 3.

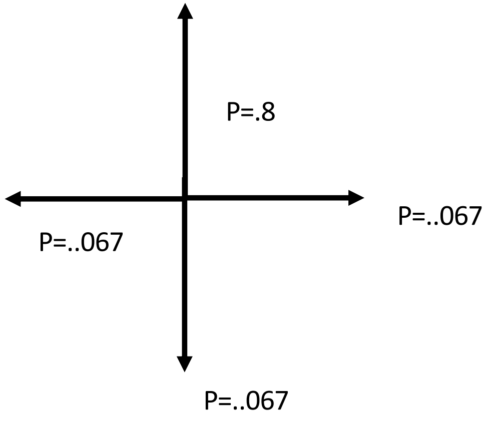
# Acknowledgements

**These slides are intended for teaching purposes only. Some material has been used/adapted from web sources and from slides by Nicholas Roy, Wolfram Burgard, Dieter Fox, Sebastian Thrun, Siddharth Srinivasa, Dan Klein, Pieter Abbeel, Max Likhachev and others.**

# Decision-making over time (deterministic vs. stochastic case)

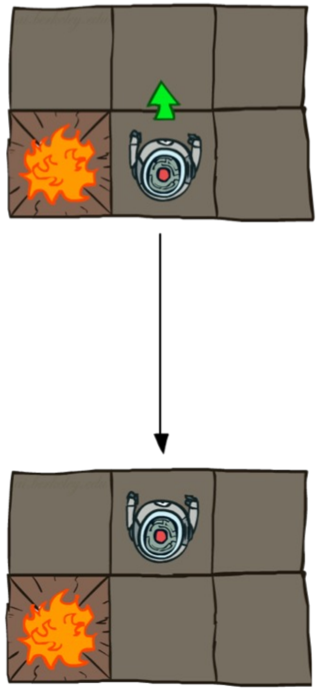
- Previously considered the deterministic case.
  - Actions (up, down, left, right) are deterministic, and each action incurs cost 1.
  - Deterministic means transition function is $T : S \times A \mapsto S$
  - Once the plan was computed, we can simply execute it.
- Now, consider the case where action outcomes are stochastic:
  - Transition function $T : S \times A \times S \mapsto [0, 1]$, $\sum_{s_j} T(\cdot, \cdot, s_j) = 1$.
  - Transition function for $(3, 2)$ and $(3, 1)$ is terminal; once reached, the agent cannot leave those states
  - Reward is $-.1$ everywhere except the terminal states, which have reward $+1$ and $-1$.
  - In this example, with probability .8 action has the "intended" outcome, and with some uniform probability (0.067) the agent ends up in one of the other feasible 4-connected states.
  - If the transition is into an obstacle or outside the grid, the agent's state does not change.

| | | | |
|---|---|---|---|
| -.1 | -.1 | -.1 | +1 |
| -.1 | | -.1 | -1 |
| -.1 | -.1 | -.1 | -.1 |

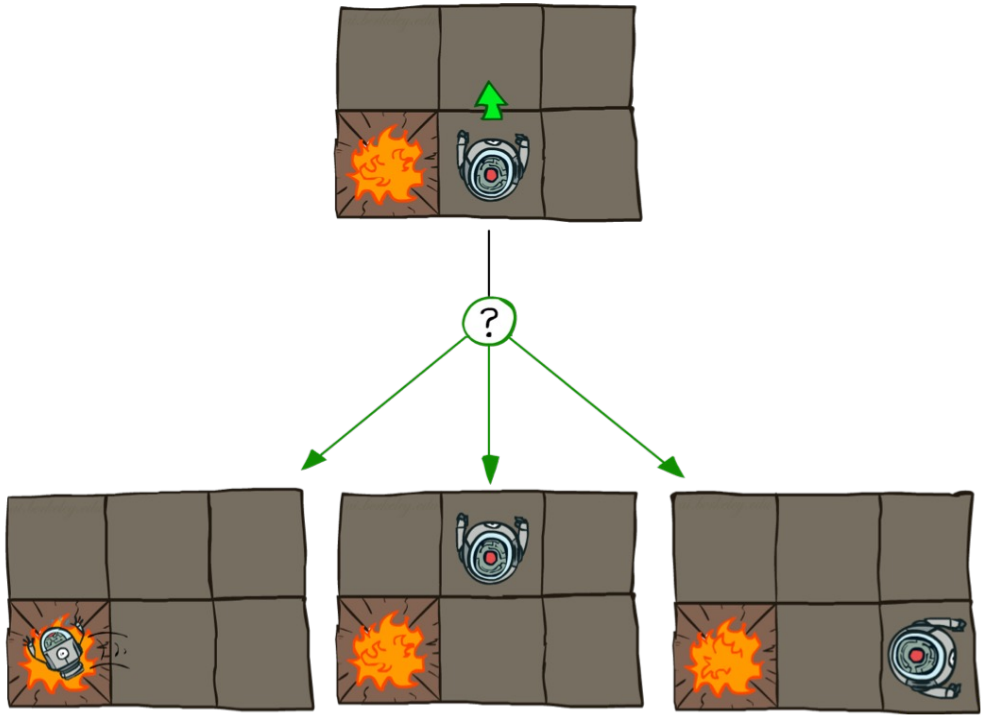


# Plans vs. Policies

Deterministic Grid World



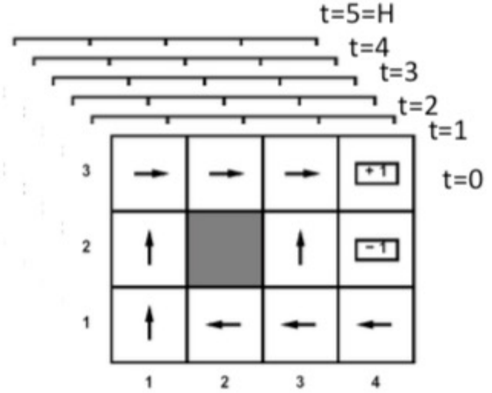
Stochastic Grid World




If the environment was *deterministic*, then we would just need an optimal plan (a sequence of actions) from start to the goal. If there is *non-determinism*, we are not sure where we will land up, hence need a policy that prescribes actions from each state.

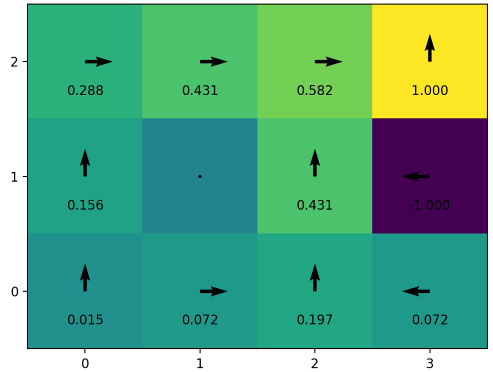
# Plans Vs. Policies

- Deterministic single-agent search problems
  - we wanted an optimal plan, or sequence of actions, from start to a goal
- For MDPs, we want an optimal policy $\pi^*: S \rightarrow A$
  - A policy $\pi$ gives an action for each state
  - The agent arrives at a state and looks up the action according to the policy.




At any time, the agent will land up in a state.
- Which action to take in that state?
- Take the action prescribed by the policy.

- **Will any policy work?**
  - **No. We want an optimal policy is one that maximizes the expected utility if followed**

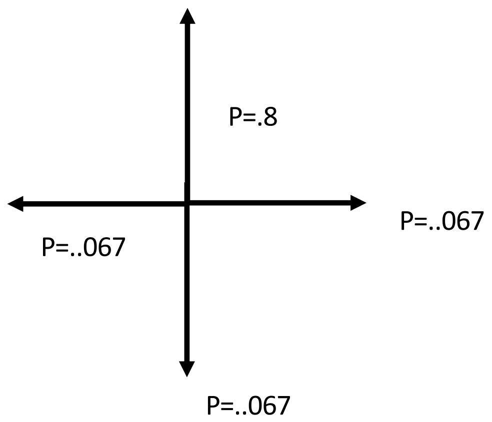



Example of a policy (arrows). The colors indicate the value of a state given the policy (see Value functions in a later slide).

# Formulating and MDP Model

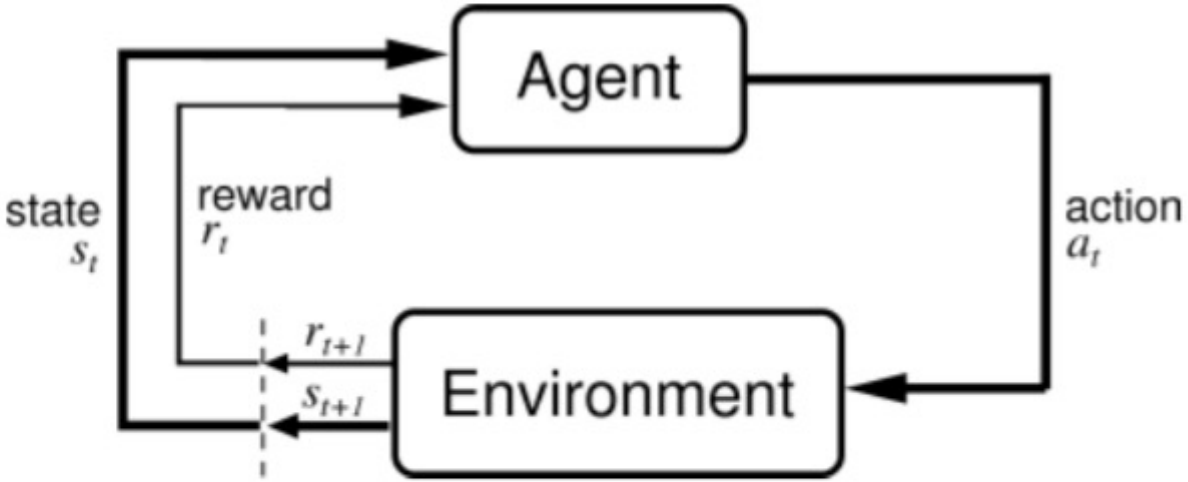
- Recall that the input to any solver is a model $M$.
- The MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma, s^0)$
  - Finite set of states, $\mathcal{S} = \{s^0, \ldots, s^n\}$
  - Finite set of actions, $\mathcal{A} = \{a^0, \ldots, a^m\}$
  - State transition function $T(s^i, a^j, s^k)$ such that $T : S \times A \times S \mapsto [0, 1]$
  - Reward for each state-action-state transition $R(s^i, a^j, s^k)$ such that $R : S \times A \times S \mapsto \mathcal{R}$
  - Discount factor $\gamma \in [0, 1]$
  - Initial state $s_0$.
- Why is it called "Markov"?
  - Because the state dynamics depends *only* on the current state and action. $s_{t+1} \perp\!\!\!\perp s_{t-1} | s_t, a_t$.
- Called a decision process
  - Because we choose the actions

| | | | |
|---|---|---|---|
| -.1 | -.1 | -.1 | +1 |
| -.1 | | -.1 | -1 |
| -.1 | -.1 | -.1 | -.1 |



# MDPs

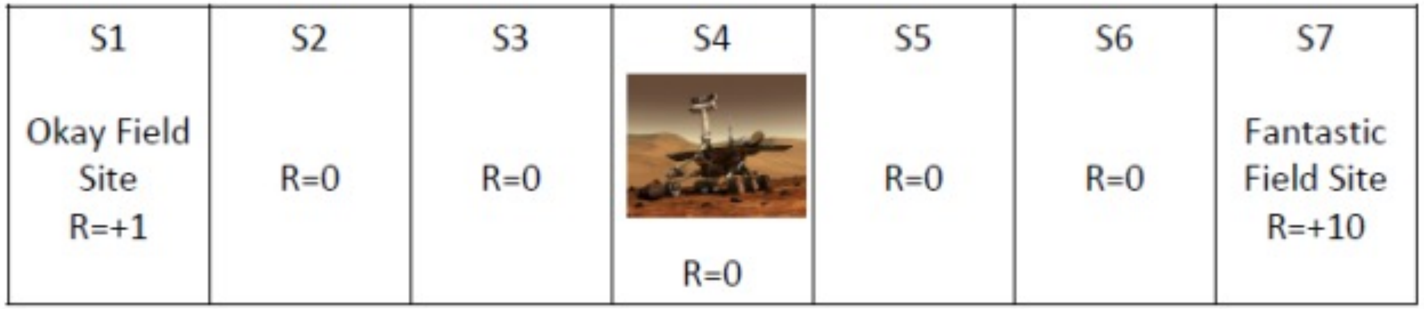
- Another view of an MDP.
- Assume that the state is known.
- Solving an MDP means
  - Find a policy that maximizes the future expected reward.
  - That is, find a prescription of actions from each state such that the future expected reward is maximized.



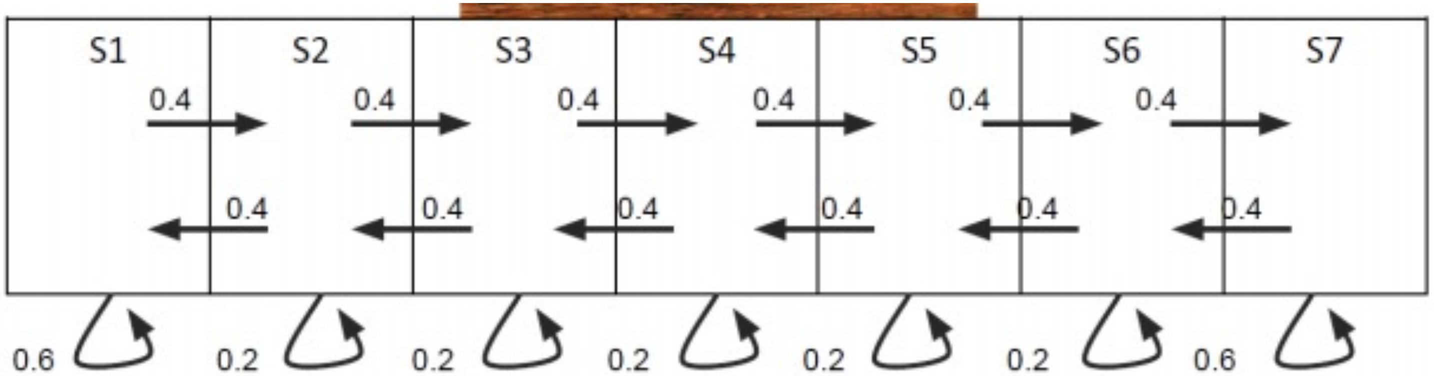

$$\pi^* = \arg\max_{\pi} \mathrm{E}[\sum_{t=0}^{H} \gamma^t R_t(S_t, A_t, S_{t+1})|\pi]$$

# MDP Example: Mars Rover

States and rewards
(encode the agent’s goal)




Transition function.




Actions: Left or Right.
Policy: prescription of actions to states.

# Value Function

- Policy prescribes actions to a particular state (a look up table): $\Pi : S \mapsto A$
- Value function of an MDP for horizon-length $T$ starting at state $s_0$:

$$V_T^\pi(s_0) = E_{s_{0:T}}\left[\sum_{t=0}^{T} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right] \tag{1}$$

$$= \sum_{s_1} p(s_1|s_0, \pi(s_0))\left(R(s_0, \pi(s_0), s_1) + \gamma \cdot E_{s_{1:T}}\left[\sum_{t=1}^{T} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right]\right) \tag{2}$$

$$= \sum_{s_1} p(s_1|s_0, \pi(s_0))\left(R(s_0, \pi(s_0), s_1) + \gamma \cdot V_{T-1}^\pi(s_1)\right). \tag{3}$$

Value function under a policy

- Called a function because it is defined for each state.
- What does it intuitively mean?
- Given the policy what is the goodness of this state. What is the reward the agent can expect from here over a time horizon.

# Value Function

- Vector Equation Formulation
  - We want the full policy over the entire state space, hence $V$ is a vector.
  - For a given policy $\pi : S \mapsto A$,
  - Define $R^\pi$ to be an $S \times 1$ matrix of the expected rewards for the action $\pi(s)$ for each state, and $T^\pi$ to be an $S \times S$ matrix of the transition probability for the action $\pi(s)$ for each state.

$$V_t^\pi = R^\pi + \gamma T^\pi V_{t-1}^\pi \tag{7}$$

- Given the "time horizon" $\tau$, then we can iterate Eqn 7 for $\tau$ steps to obtain the total amount of expected reward over that horizon.
- $V_T$ can be computed recursively by solving for successive value functions $V_0, V_1, \ldots V_t \ldots V_T$.
  - Each recursion is known as a "back-up" and is a form of dynamic programming.

*Vector Equation Formulation (an intuition)*

- *Value function is a discrete function. For each state it is a value. Writing it as a vector we can use the relationship above to compute it.*

# Sequential Decision-making: Assigning rewards to sequences)

- An MDP models a sequential decision making task till a time horizon $\tau$
- What if we don't know $\tau$, and so may need to run for an arbitrarily-long horizon (or literally expect to run forever).
  - Known as the "infinite horizon" setting.
- The discount factor biases the value towards getting more reward sooner.
  - Getting a rupee today is better than getting it tomorrow which is better than the day after.
- Value function is guaranteed to exist in the infinite horizon case if $\gamma < 1$ and the reward function $R$ is bounded.

# Assigning Rewards to Sequences

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

**Discounting appears to be a good model for both animal and human preferences over time.**

- **With discounted rewards, the utility of an infinite sequence is finite.**

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

**Notation: U() and V() are both used to denote value functions in literature.**

# Value Function

- The finite horizon value function given by Eq 3 becomes an increasingly good approximation as $t \to \infty$, and $V^\pi$ represents a fixed point of this recursion.

$$
\begin{aligned}
V^\pi(s) &= E_{s_{0:\infty}}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right] && (8)\\
&= \sum_{s'} p(s'|s,\pi(s))\left(R(s,\pi(s),s') + \gamma \cdot E_{s_{0:\infty}}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right]\right) && (9)\\
&= \sum_{s'} p(s'|s,\pi(s))\left(R(s,\pi(s),s') + \gamma \cdot V^\pi(s')\right). && (10)
\end{aligned}
$$

- We can also solve this fixed point as a vector equation as before:

$$
\begin{aligned}
V^\pi &= R^\pi + \gamma T^\pi V^\pi && (11)\\
\Rightarrow V^\pi - \gamma T^\pi V^\pi &= T^\pi R^\pi && (12)\\
\Rightarrow (I - \gamma T^\pi) V^\pi &= T^\pi R^\pi && (13)\\
\Rightarrow V^\pi &= (I - \gamma T^\pi)^{-1} T^\pi R^\pi && (14)
\end{aligned}
$$

# Policy Evaluation

- An approach is iterative policy evaluation.
- Calculate the utilities (values) if the agent follows a given fixed policy until convergence.
- The computed value function may not be the optimal. It is the "best" we can get with a given policy.

**Algorithm 1** Computing the value of an MDP policy.

POLICY-EVALUATION$(\mathcal{S}, \pi, R, T, \gamma, \epsilon)$

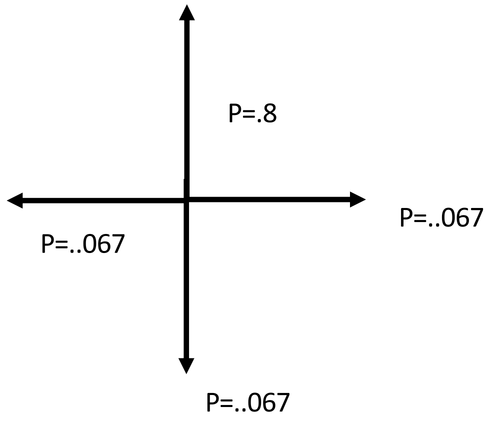
$t = 0$<br>
**for** each state $s \in \mathcal{S}$<br>
    $V_0[s] = 0$<br>
**repeat**<br>
    $change = 0$<br>
    $t = t + 1$<br>
    **for** each state $s \in \mathcal{S}$<br>
        $V_t[s] = \left(\sum_{s'} T(s, \pi[s], s')\,[R(s, \pi[s], s') + \gamma \cdot V_{t-1}(s')]\right)$<br>
        $change = \max(change, V_t[s] - V_{t-1}[s])$<br>
**until** $change < \epsilon$<br>
**return** $V_t$

*Given the policy what is the value of each state when we are using this policy?*

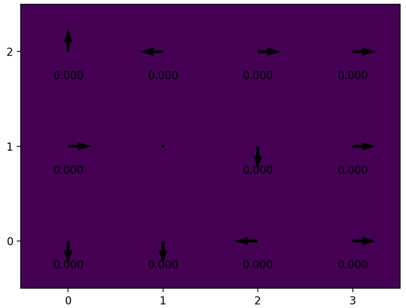
# Policy Evaluation: Example

- Recall the grid world. Let us guess a policy at random, and evaluate it.
- The MDP model:
  - $S = \{(0,0),(0,1),(0,2),(1,1)\ldots(3,2)\}$
  - $T(s,a,s') = .8$ if $s'$ is the nominal destination given $s$ and $a$
  - $T(s,a,s') = .2/n$ if $s'$ is a neighbour of the nominal destination given s and a and is a legal state (for $n$ legal states)
  - $R(s) = +1$ if $s = (3,2)$
  - $R(s) = -1$ if $s = (3,1)$
  - $R(s) = -.1$ otherwise
  - $\gamma = 0.9$

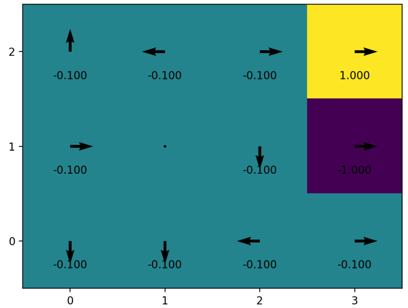
| | | | |
|---|---|---|---|
| -.1 | -.1 | -.1 | +1 |
| -.1 | | -.1 | -1 |
| -.1 | -.1 | -.1 | -.1 |



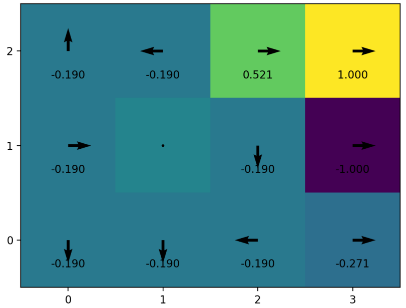
# Policy Evaluation: Example



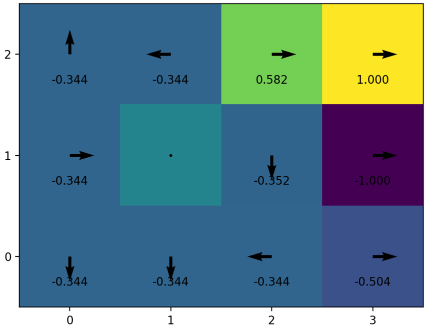

(a) $V^0$




(b) $V^1$




(c) $V^2$




(d) $V^3$

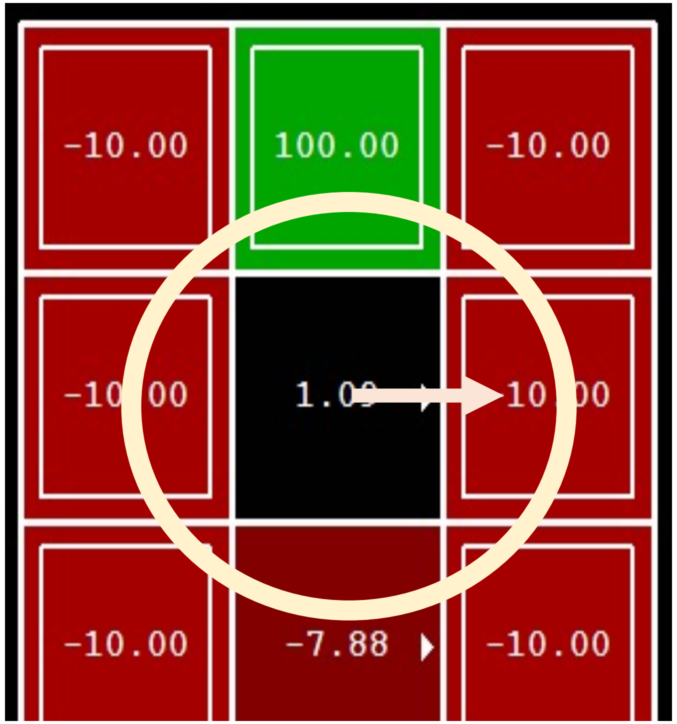
Let us consider the state $(2,2)$

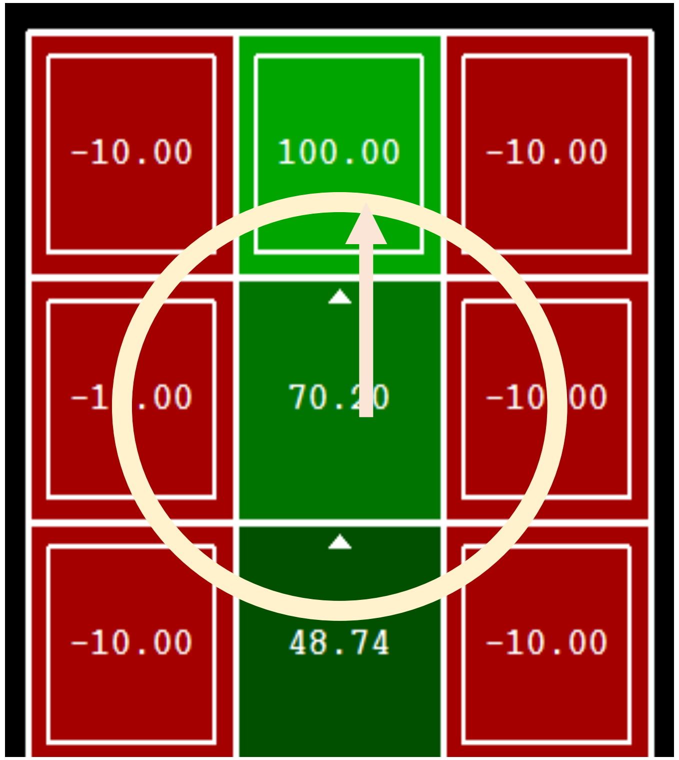
$$
\begin{aligned}
V^1(2,2) &= R((2,2)) + \gamma \sum_{s'} p(s'|s,a) V^0(s') \\
&= -.1 + .9(.8 \times 0 + .1 \times 0 + .1 \times 0) \\
&= -.1 \\
V^2(2,2) &= R((2,2)) + \gamma \sum_{s'} p(s'|s,a) V^1(s') \\
&= -.1 + .9(.8 \times 1 + .1 \times -.1 + .1 \times -1) \\
&= .521
\end{aligned}
$$

This was the evaluation of a random policy. Next, how can we improve the policy?

# Policy Improvement

Illustration in a grid world example.



# Policy Iteration

- Consider improving the policy
  - Given the fixed utility values for states (obtained via policy evaluation)
  - Examine if there is a better policy using one step look ahead.

**Algorithm 2** An algorithm for computing an MDP policy by repeated evaluation and improvement.

Policy-Iteration($\mathcal{S}, \mathcal{A}, R, T, \gamma, \epsilon$)

**for** each state $s \in \mathcal{S}$
    $i = rand(0, |\mathcal{A}|)$
    $\pi[s] = a_i$
$t = 0$
$V_0 =$ Policy-Evaluation($\mathcal{S}, \pi, R, T, \gamma, \epsilon$)
**repeat**
    $change = 0$
    $t = t + 1$
    **for** each state $s \in \mathcal{S}$
        $\pi_t[s] = \text{argmax}_a \left(\sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma \cdot V_{t-1}(s')\right]\right)$
    $V_t =$ Policy-Evaluation($\mathcal{S}, \pi, R, T, \gamma, \epsilon$)
    **for** each state $s \in \mathcal{S}$
        $change = change + V_t[s] - V_{t-1}[s]$
**until** $change < \epsilon$

Key Idea:
- Evaluate and improve the policy.
- Interleave evaluation and improvement

# Policy Iteration

Remember to evaluate the policy, we fixed $\pi$ and constructed $R^{\pi}$ an $T^{\pi}$.

1. $\pi(s) = \text{argmax}\, a \sum_{s'} T(s,a,s')\,[R(s,a,s') + \gamma V(s')]$
2. Construct new $T^{\pi}, R^{\pi}$
3. $V = (I - \gamma T^{\pi})^{-1} R^{\pi}$

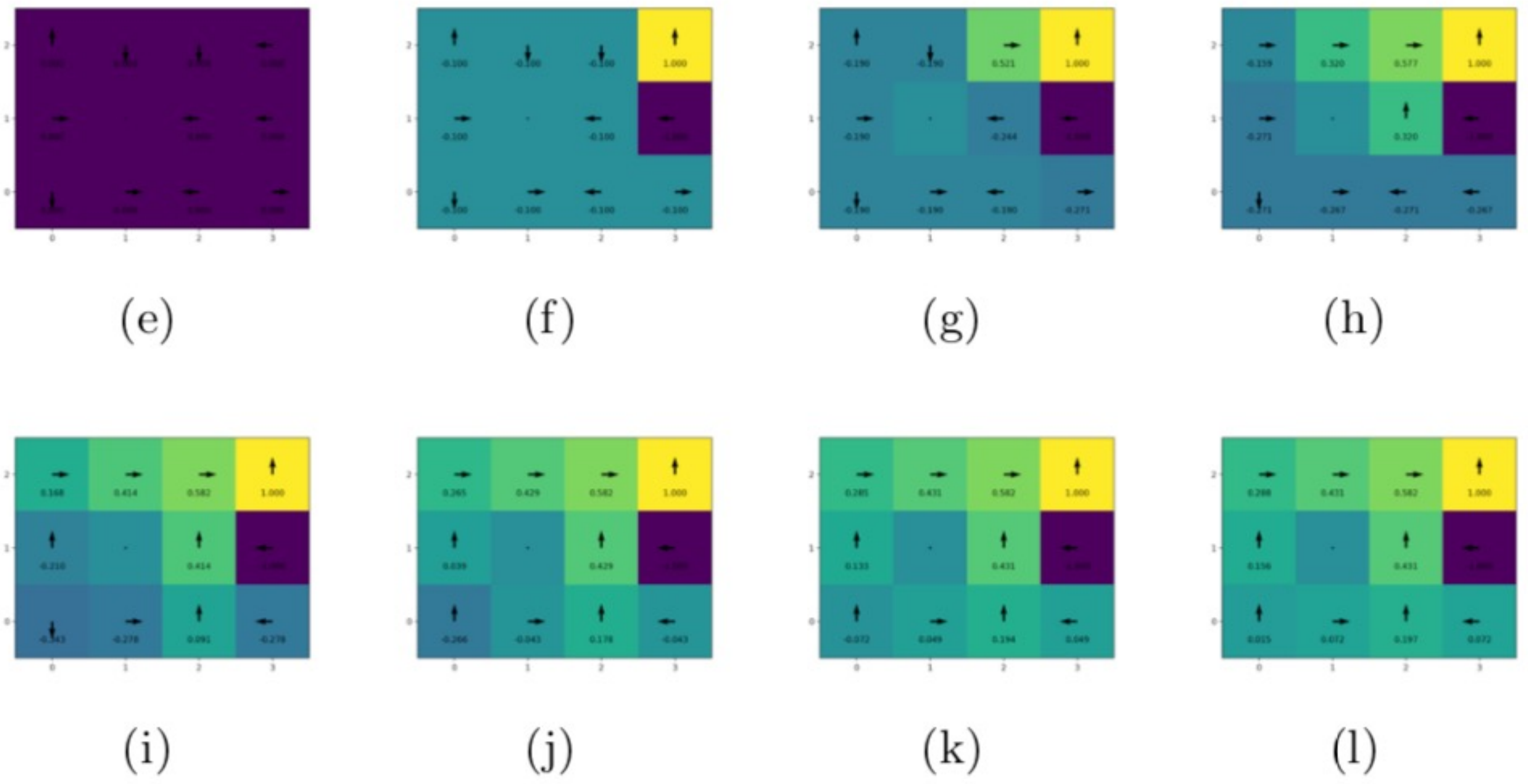
Policy-Iteration$(\mathcal{S}, \mathcal{A}, R, T, \gamma, \epsilon)$

**for** each state $s \in \mathcal{S}$<br>
    $i = rand(0, |\mathcal{A}|)$<br>
    $\pi[s] = a_i$<br>
$t = 0$<br>
$V_0 =$ Policy-Evaluation$(\mathcal{S}, \pi, R, T, \gamma, \epsilon)$<br>
**repeat**<br>
    $change = 0$<br>
    $t = t + 1$<br>
    **for** each state $s \in \mathcal{S}$<br>
        $\pi_t[s] = \text{argmax}_a \left(\sum_{s'} T(s,a,s')\,[R(s,a,s') + \gamma \cdot V_{t-1}(s')]\right)$<br>
    $V_t =$ Policy-Evaluation$(\mathcal{S}, \pi, R, T, \gamma, \epsilon)$<br>
    **for** each state $s \in \mathcal{S}$<br>
        $change = change + V_t[s] - V_{t-1}[s]$<br>
**until** $change < \epsilon$

# Policy Iteration Example

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*,$$

- Interleaving policy evaluation and policy improvement.
- Notice the policy changing over time.
- We are guaranteed to reach the optimal policy.



# Important Quantities

**The value (utility) of a state s:**

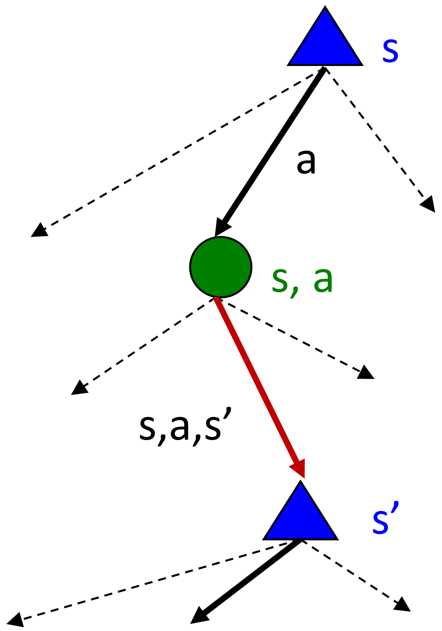
$V^*(s)$ = expected utility starting in s and acting optimally

The value (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

**The optimal policy:**

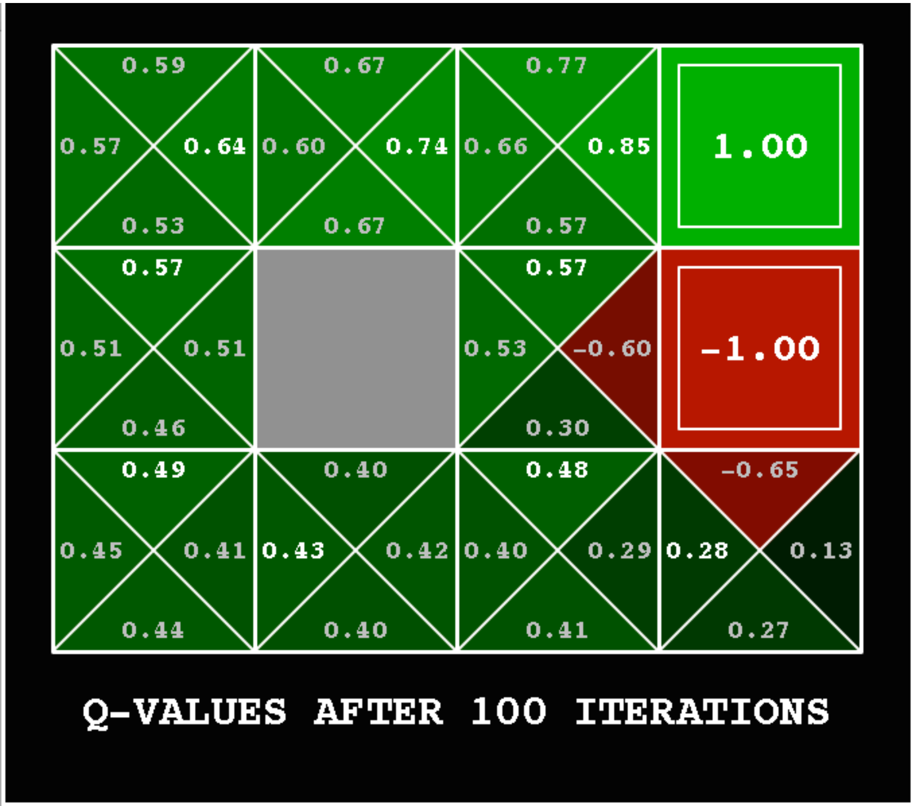
$\pi^*(s)$ = optimal action from state s



# Grid World Values

Value (utility) of states V(s) for all states


Value (utility) of a q-states Q(s,a) for all states and all actions at each state.




Noise = 0.2
Discount = 0.9
Living reward = 0

# Bellman Equations

- Definition of **"optimal"** utility via a simple **one-step** lookahead relationship amongst optimal values.

Optimal value function for a state s maximizes the Q-values of the state s and applicable actions a.

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

Q value function: how can we express it? Take the action a on state s and then act optimally.

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

Recursive way to write the value function (we have seen this definition before).

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

- The utility of a state is the **immediate reward** for that state plus the **expected discounted utility of the next state** assuming that the agent is acting **optimally**.

# Value Iteration

- Key Idea
  - Calculate the optimal value function for each state. Then use the optimal value function to extract the optimal policy.
- Bellman Equations
  - The utility of a state is the immediate reward for that state plus the expected discounted utility of the next state assuming that the agent is acting optimally.

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

**What is the intuition?**

- Value of a state is not arbitrary, it is influenced by the neighbors.
- The Bellman equation encode this relationship.
- Methods to compute the policy exploit this relationship.

# Value Iteration

- Bellman equations ***characterize*** the optimal values:

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

- Value iteration ***computes*** them with a fixed-point iteration also called the Bellman update/backup:

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

- In essence, value iteration provides the optimal policy for a finite horizon problem of length one, then two-step, then three step, ……… and so on.
- Equivalently, one step of policy evaluation and one step of policy improvement.

# Value Iteration

**Value Iteration, for estimating $\pi \approx \pi_*$**

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
|   $v \leftarrow V(s)$
|   $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
|   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

Sutton and Barto.

# Stopping Criteria

Assuming and infinite horizon MDP with a discount factor less then a relationship between successive difference between value functions and w.r.t to the optimal can be shown.

The relationship leads to a stopping criteria for value iteration.

$$\text{if} \quad ||U_{i+1} - U_i|| < \epsilon(1-\gamma)/\gamma \quad \text{then} \quad ||U_{i+1} - U|| < \epsilon$$

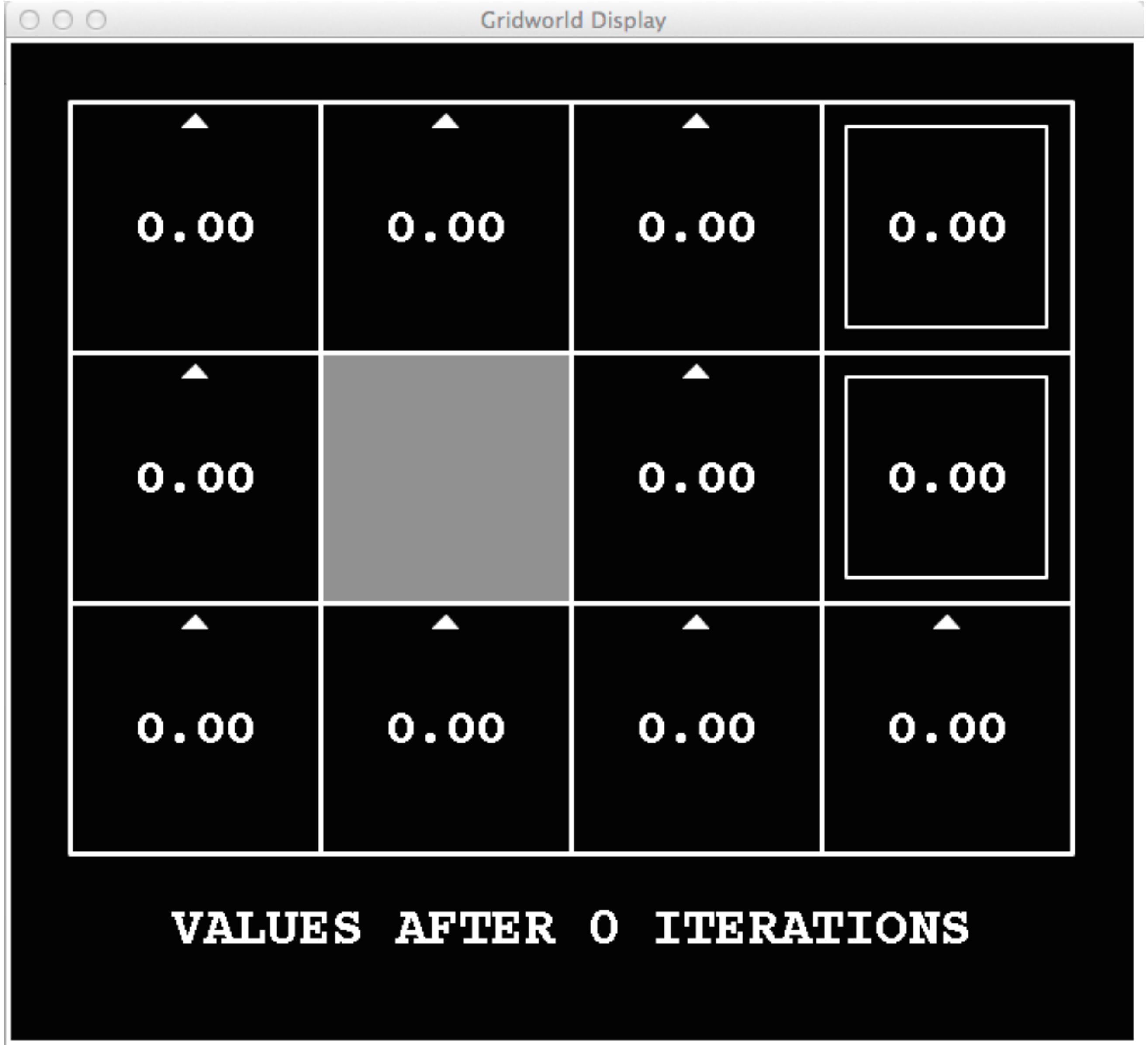
**function** VALUE-ITERATION($mdp, \epsilon$) **returns** a utility function
  **inputs:** $mdp$, an MDP with states $S$, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s)$, discount $\gamma$
    $\epsilon$, the maximum error allowed in the utility of any state
  **local variables:** $U$, $U'$, vectors of utilities for states in $S$, initially zero
    $\delta$, the maximum change in the utility of any state in an iteration

  **repeat**
    $U \leftarrow U'$; $\delta \leftarrow 0$
    **for each** state $s$ in $S$ **do**
      $U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$
      **if** $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  **until** $\delta < \epsilon(1-\gamma)/\gamma$
  **return** $U$

AIMA Ch. 17. Sec. 17.2

# Value Iteration

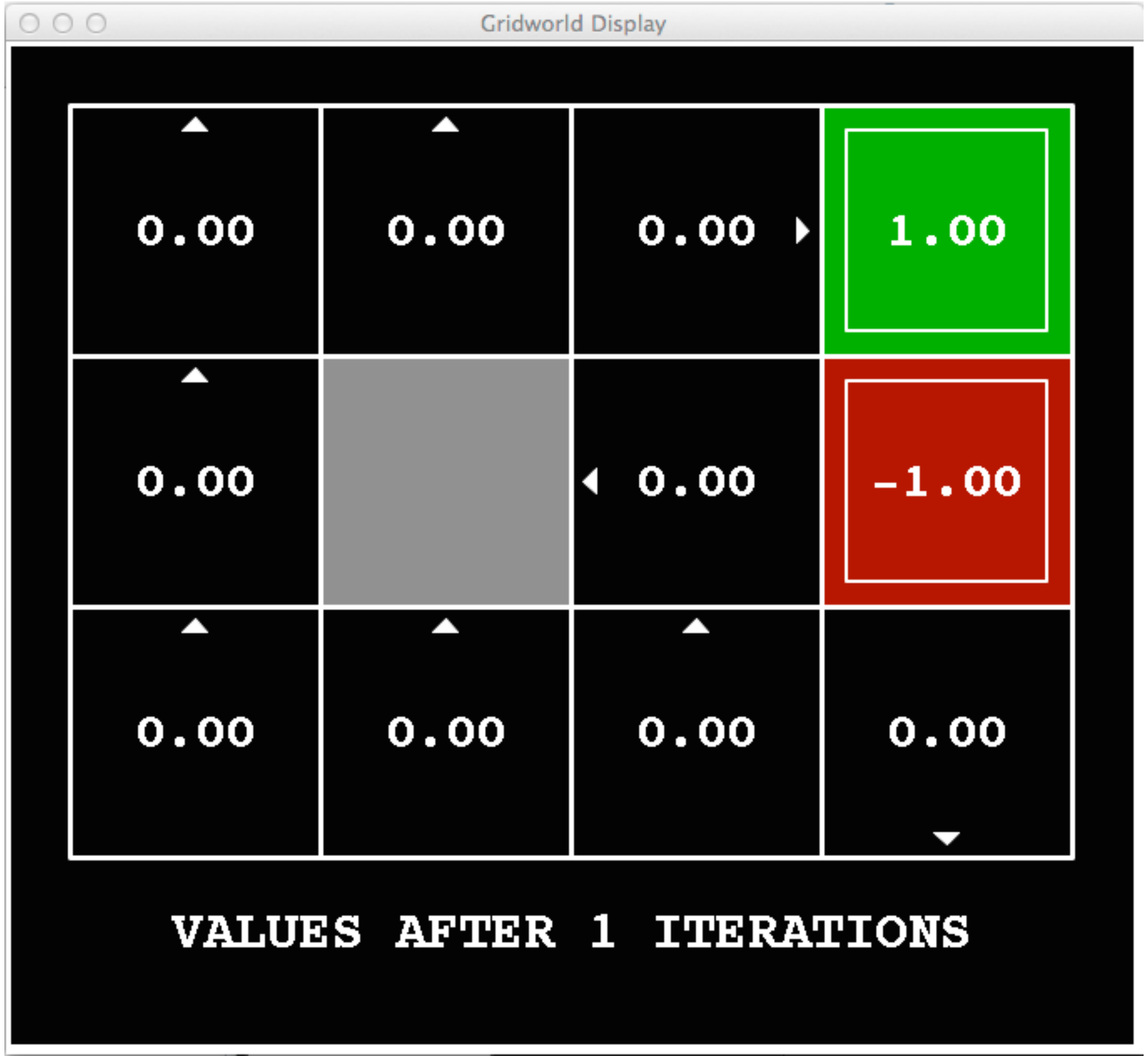
k=0




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

k=1




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

k=2


Noise = 0.2
Discount = 0.9
Living reward = 0

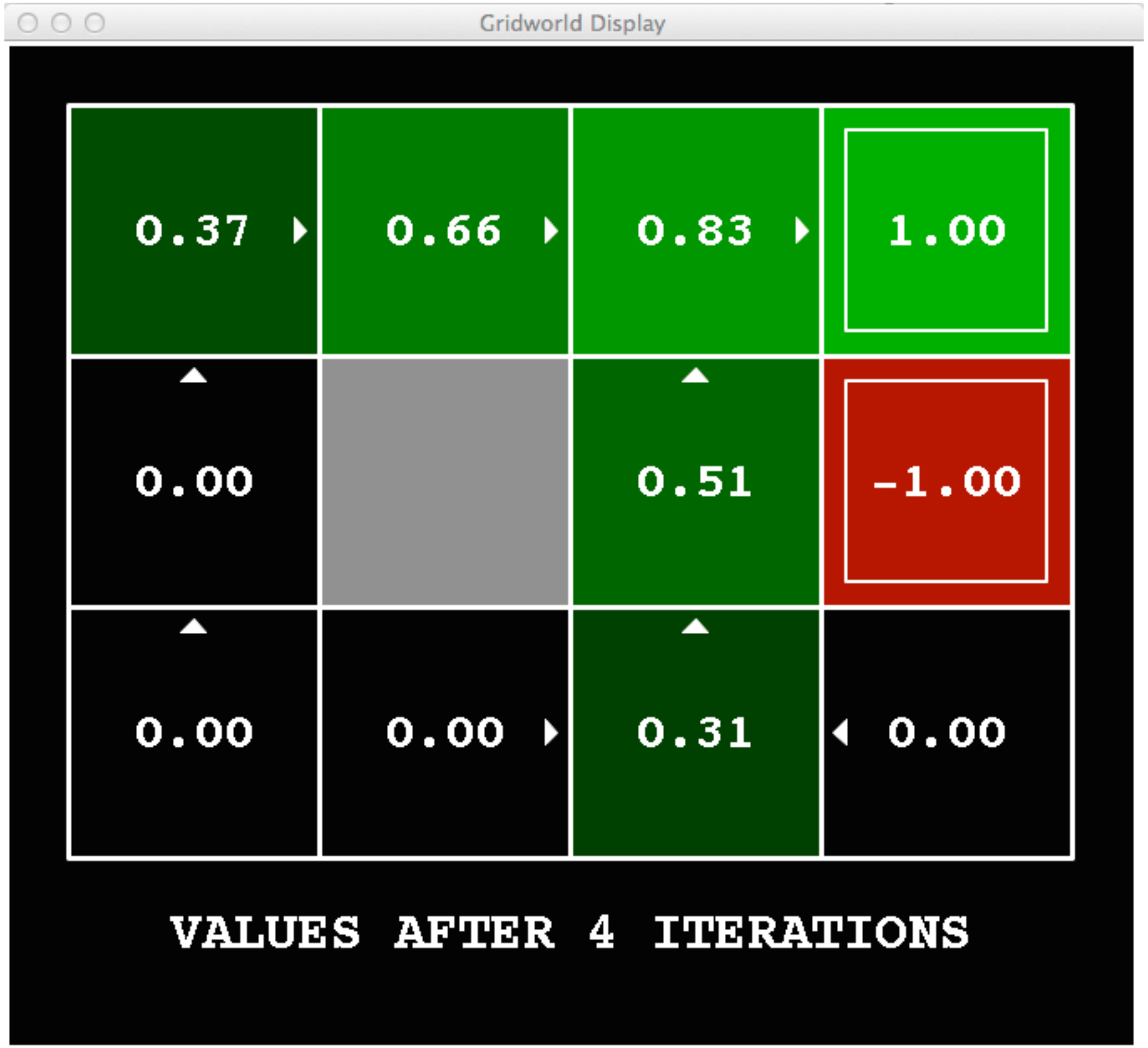
# Value Iteration

k=3


Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

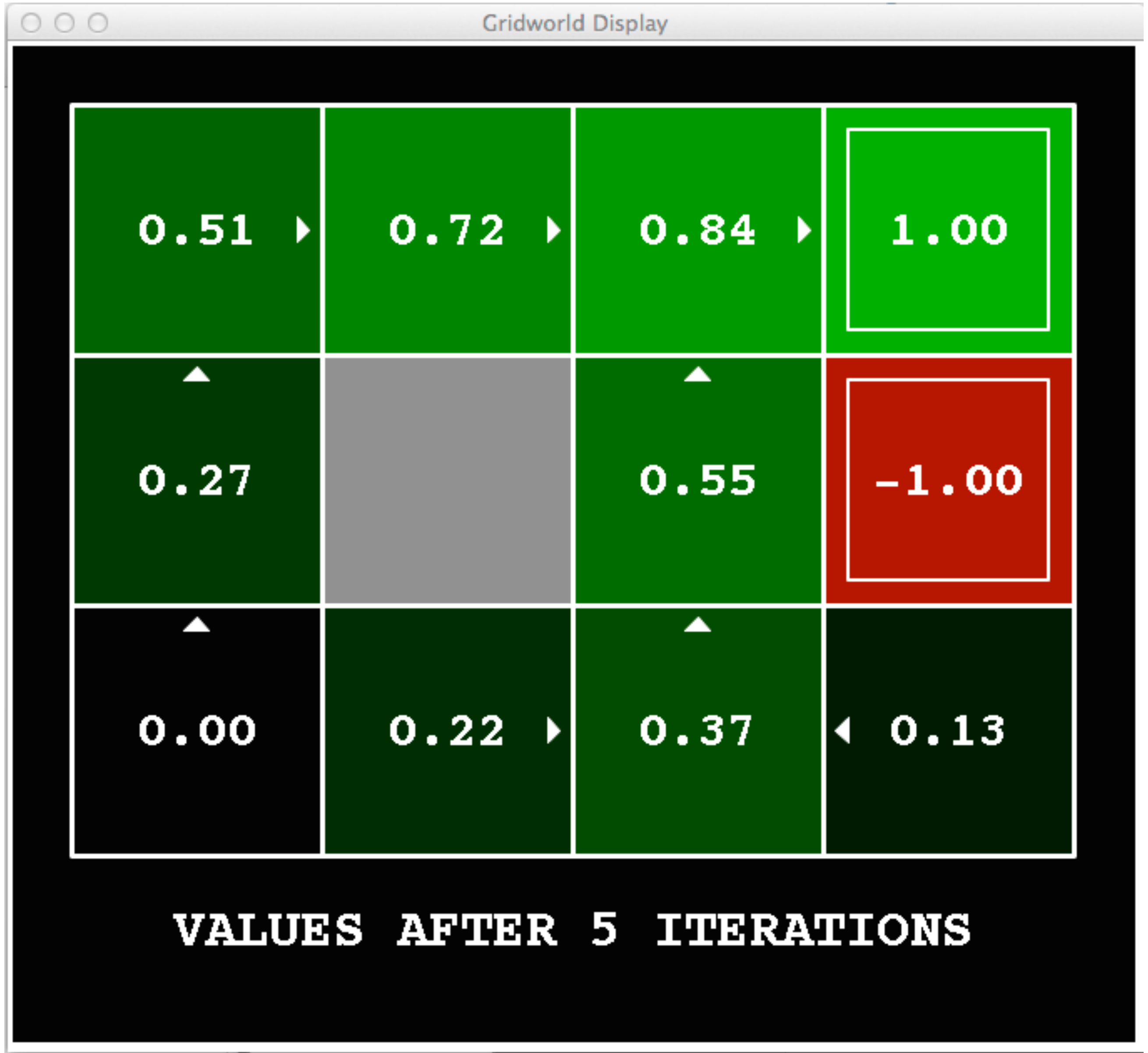
k=4




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

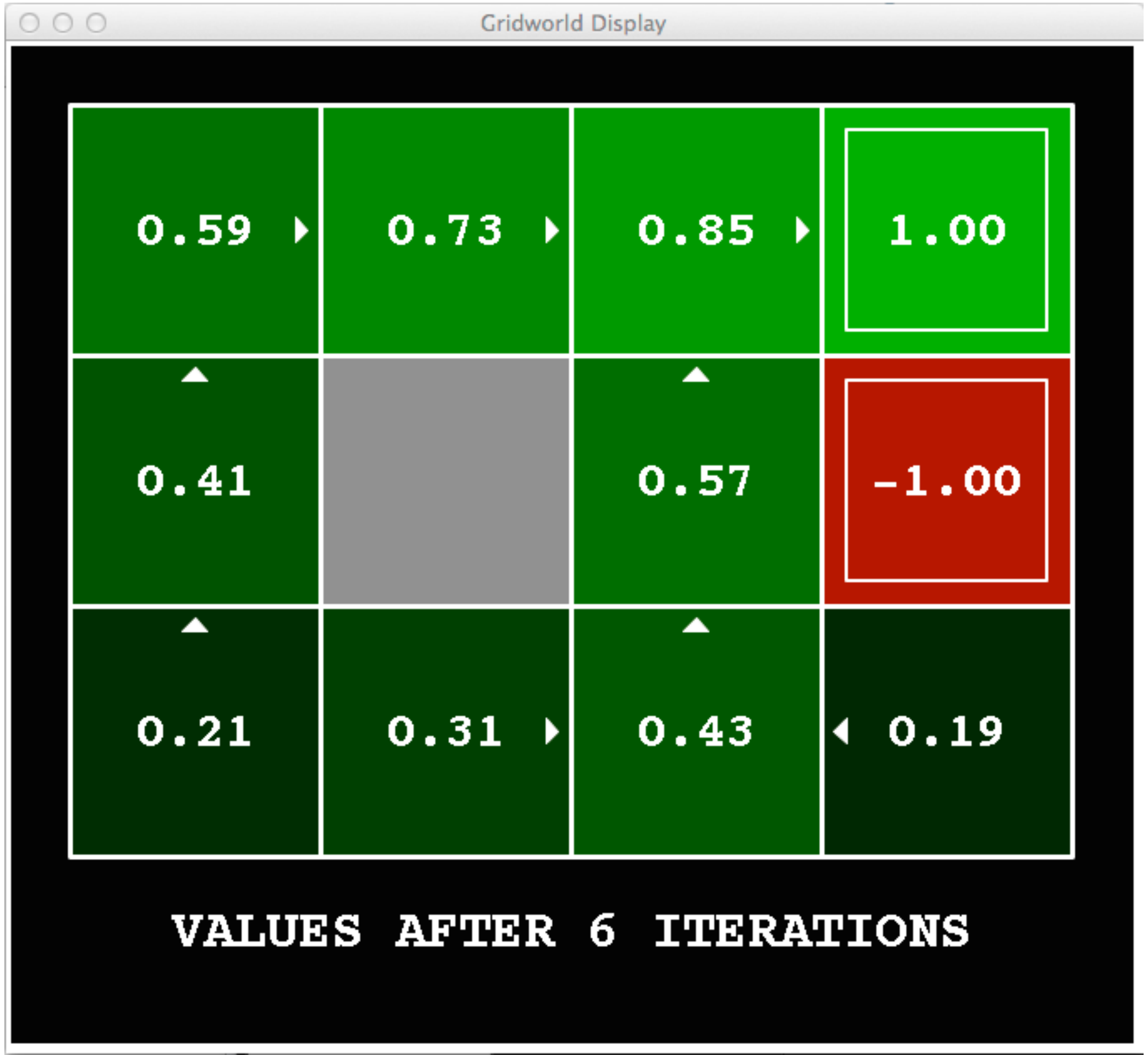
k=5




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

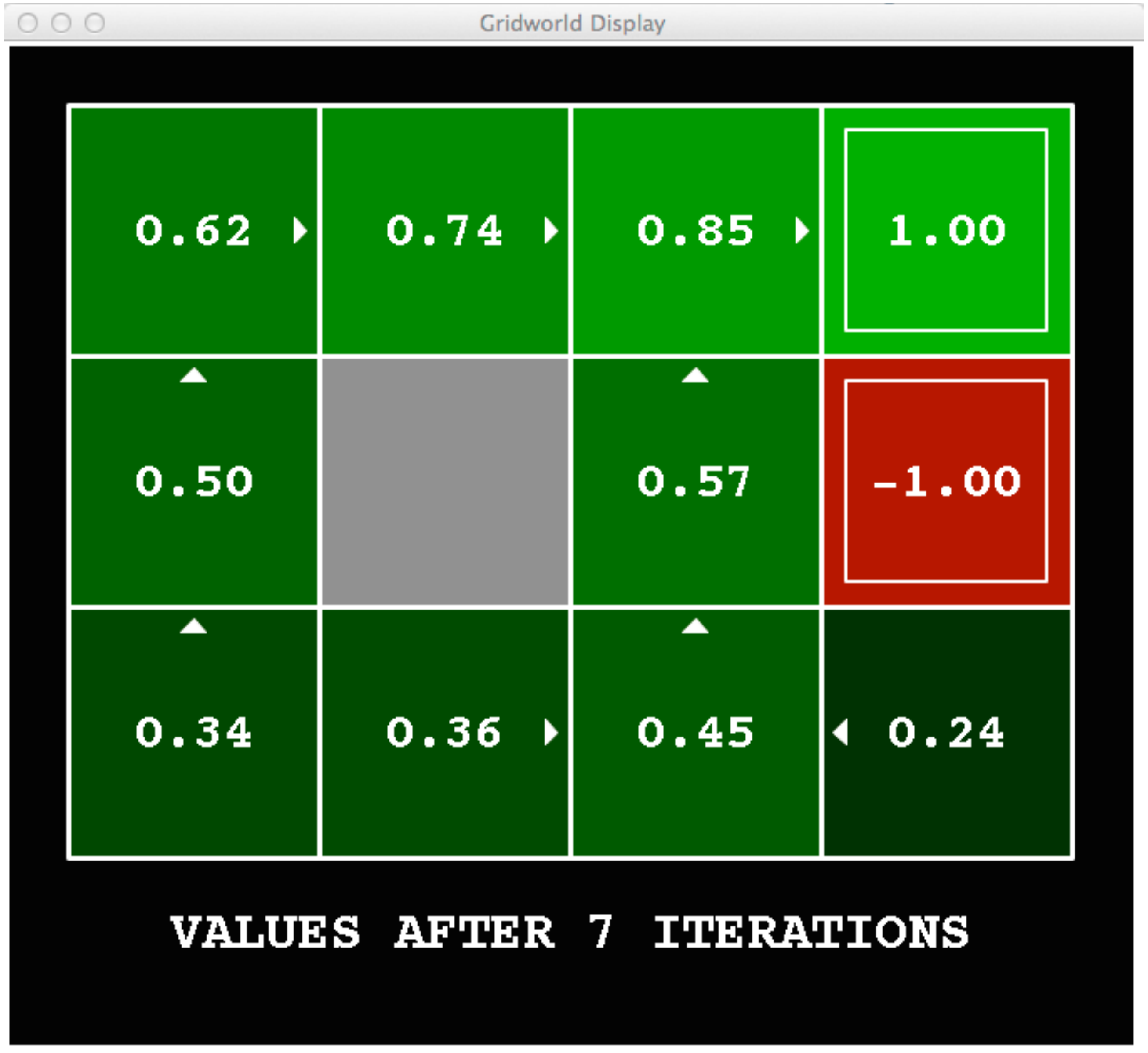
k=6




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

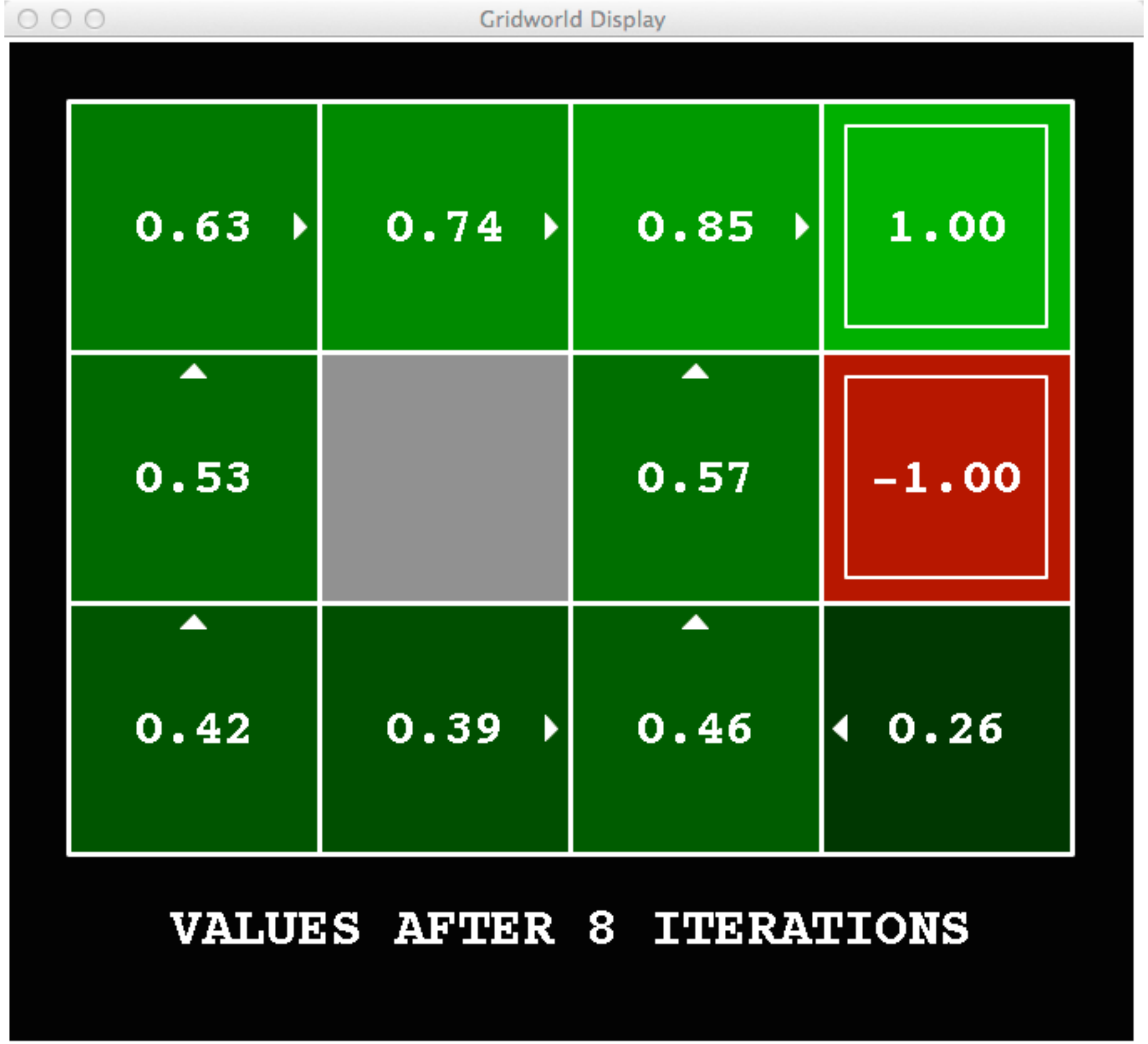
k=7




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

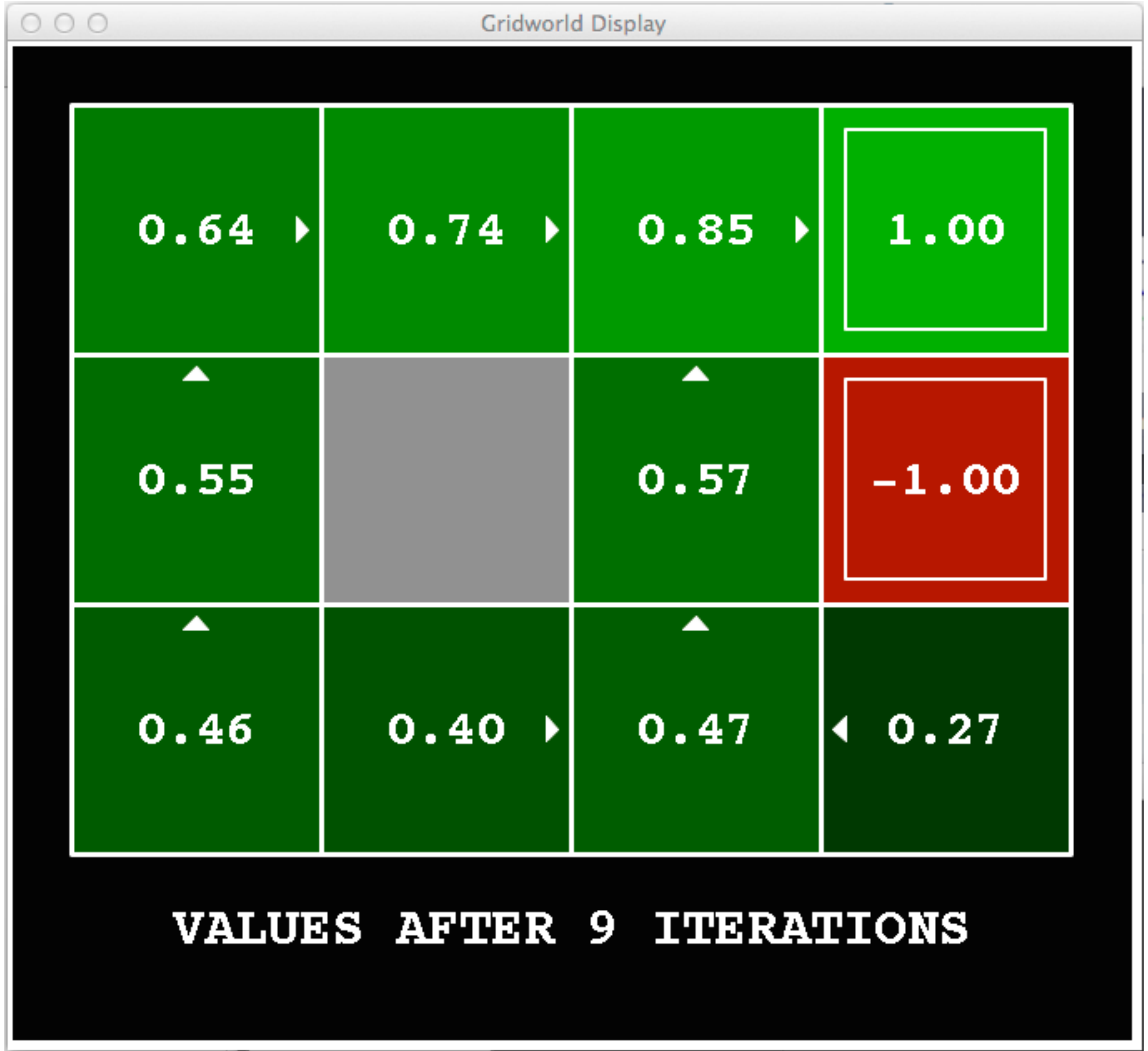
k=8




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

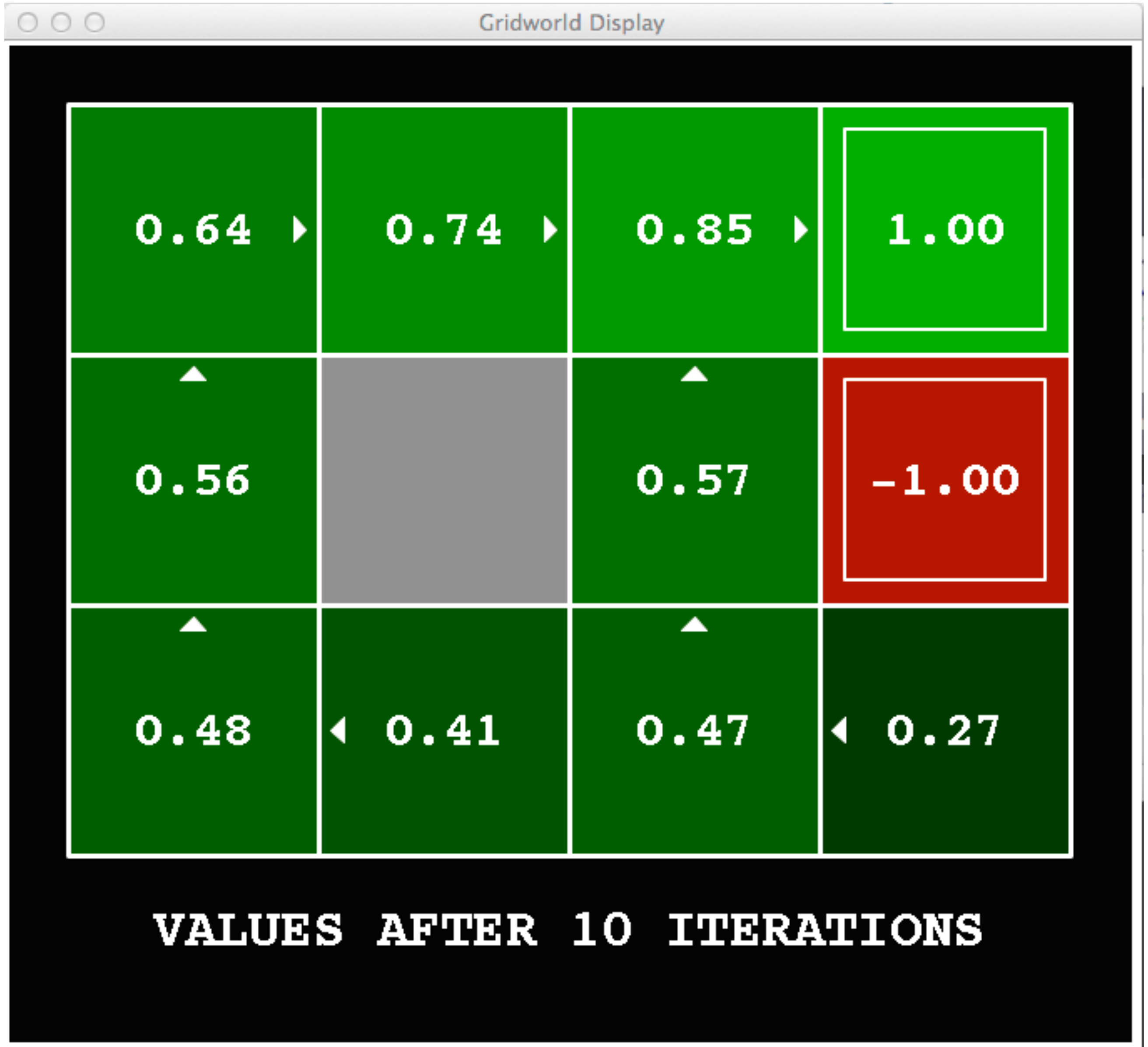
k=9




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

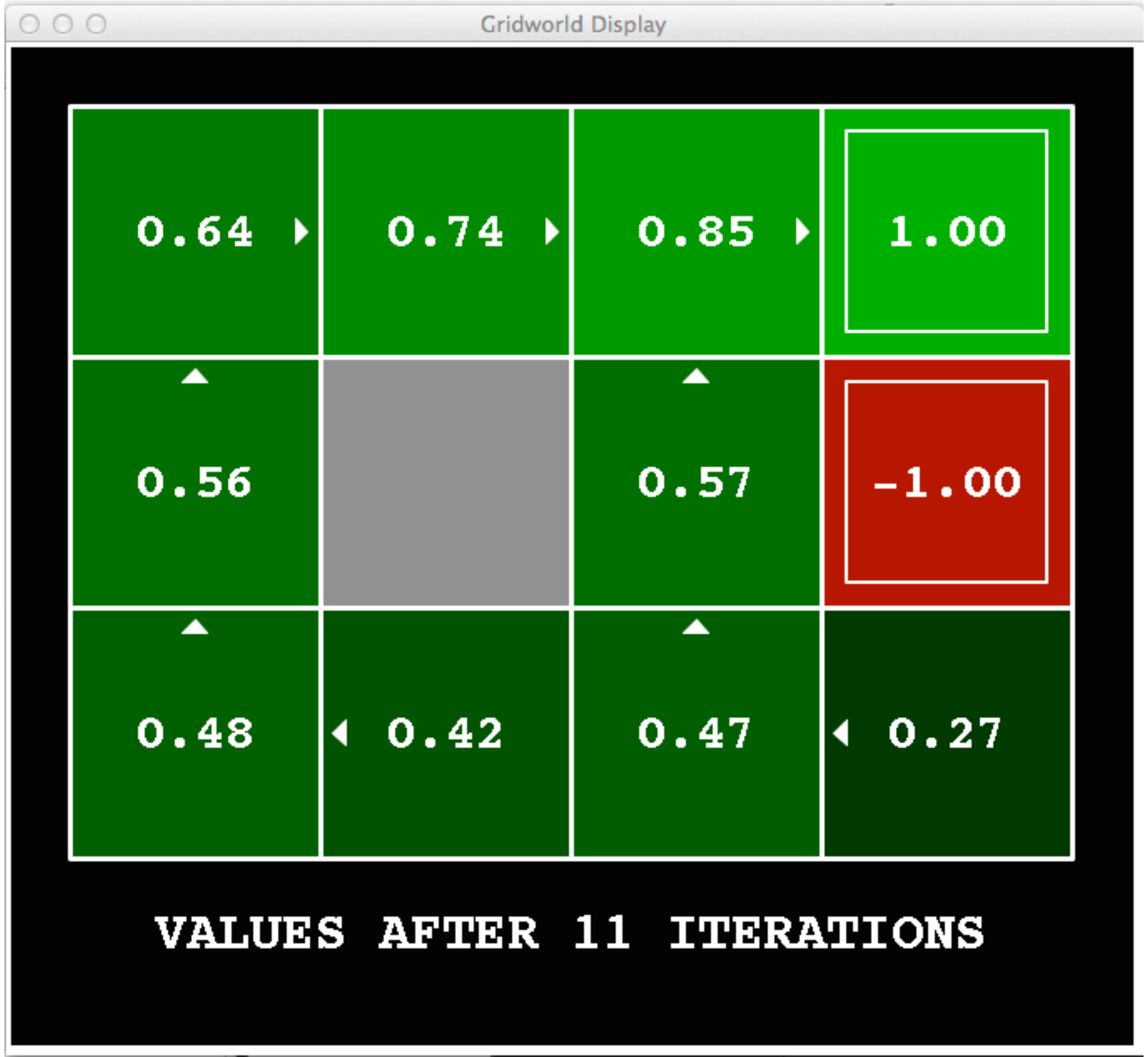
k=10




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

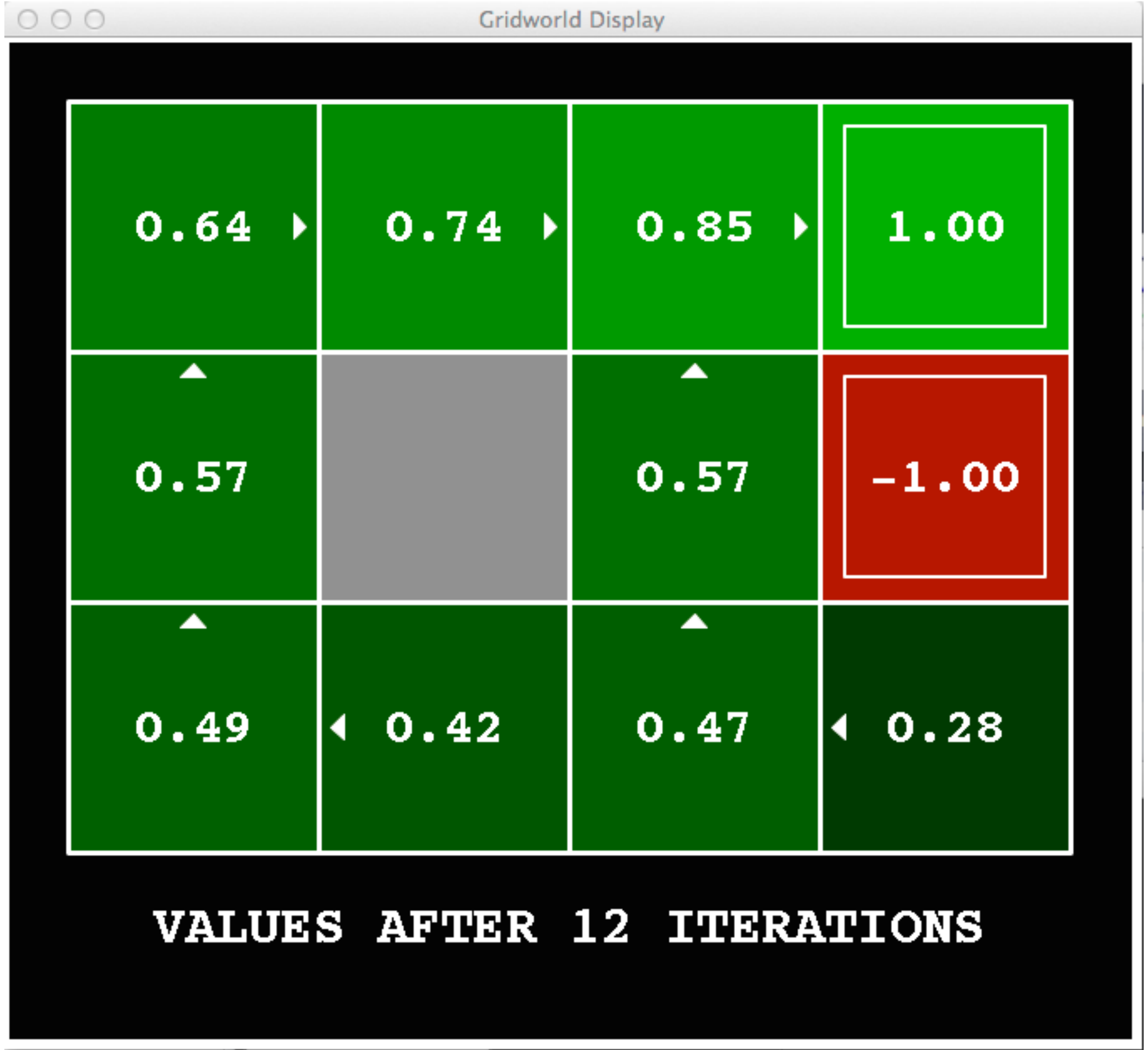
k=11




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

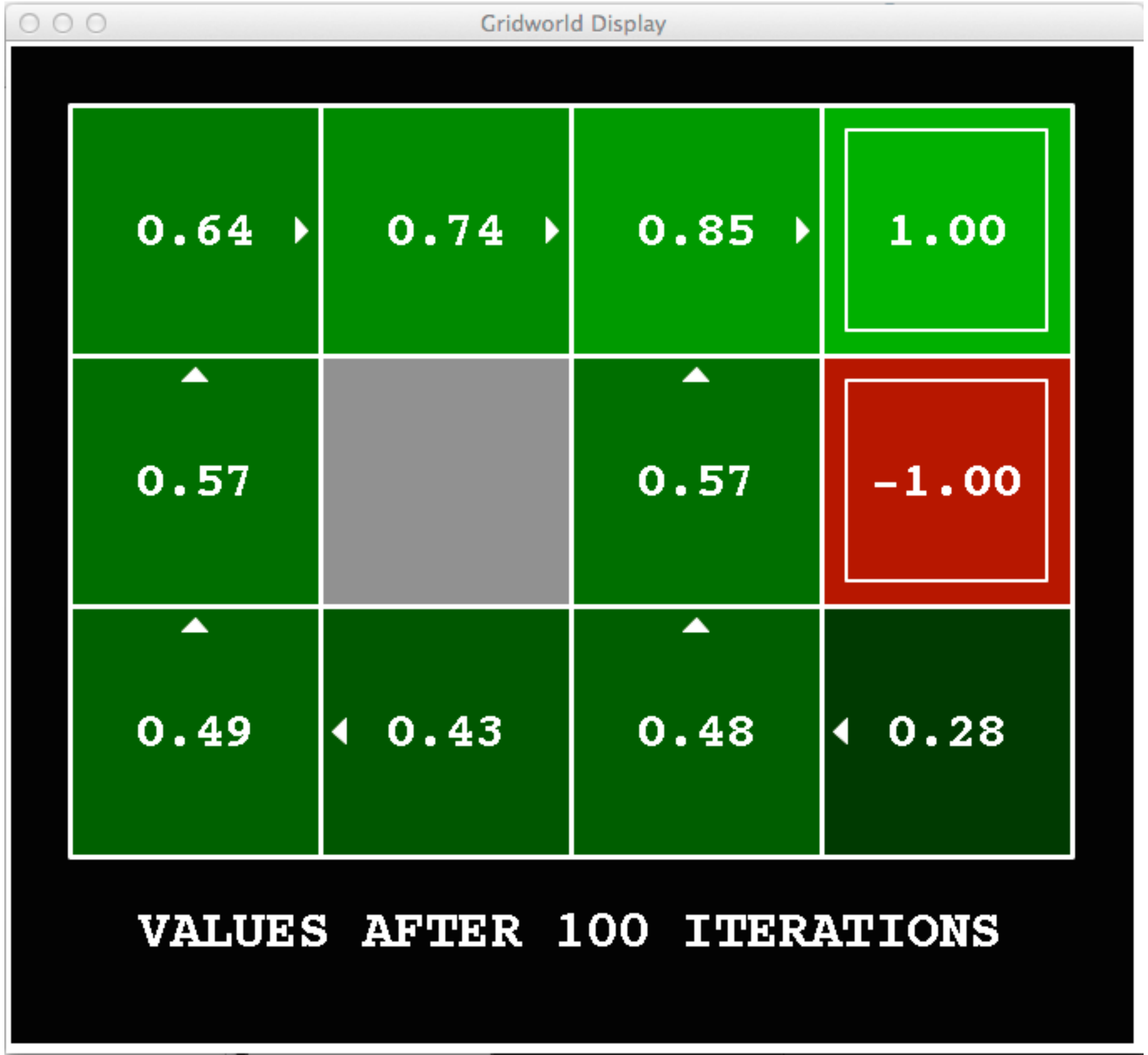
k=12




Noise = 0.2
Discount = 0.9
Living reward = 0

# Value Iteration

k=100



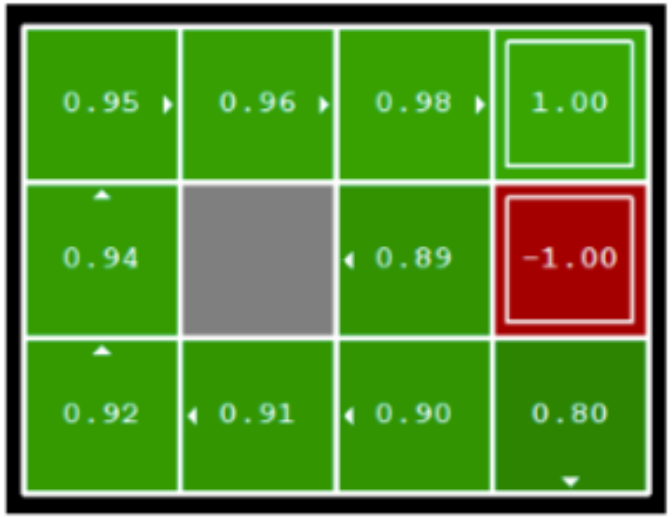

Noise = 0.2
Discount = 0.9
Living reward = 0

# How to extract the policy?

- **Assume that we have the optimal values V*(s)**
  - After applying the value iteration algorithm
- **Policy Extraction**
  - Extract the policy implied by the computed value function by (using 1-step look ahead).

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

Once we have the values computed, how to get the policy (arrows)?



# Asynchronous Value Iteration (Prioritized Sweeping)

- Value iteration so far assumes all states in $V_t$ are updated using values from prior value function $V_{t-1}$.
  - Requires keeping two functions
- Alternate approach is to keep a single value function estimate $\hat{V}$ and update states in order

$$\hat{V}(s) = R(s) + \gamma \max_a \sum_s p(s'|s,a)V(s')$$

- Sweeping through the states in order: Gauss-Seidel value iteration

- Can perform updates to a smaller number of states. Still the algorithm converges as long as a state is not starved.
- In essence backup with priority states whose successors change the most. Avoid backing up a state if the successors are not changing.
- After a backup update the priority queue.

# Policy Iteration vs. Value Iteration

- Value Iteration and Policy Iteration both compute the optimal values.
- Policy iteration may in many cases by more computationally expensive.
- For a finite policy space (i.e., discrete state, discrete actions), policy iteration is guaranteed to converge in a finite number of iterations.
  - Value iteration does not have the same guarantees.