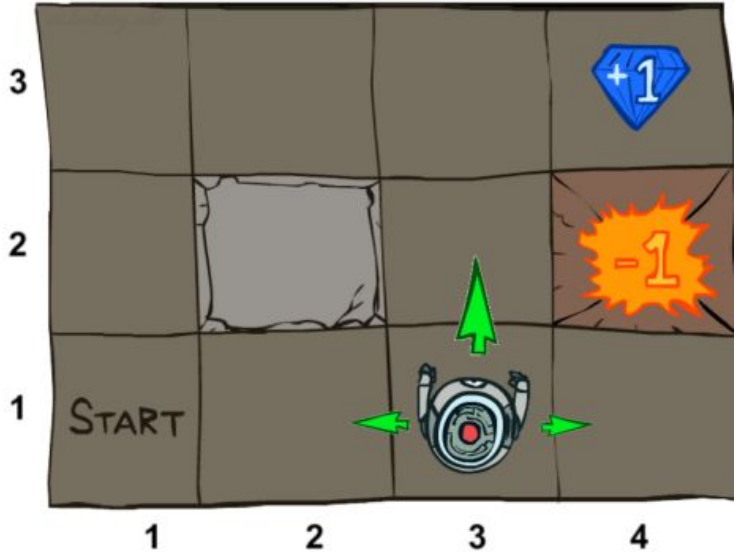
Markov Decision Process I

CSE 4617: Artificial Intelligence

Isham Tashdeed
Lecturer, CSE

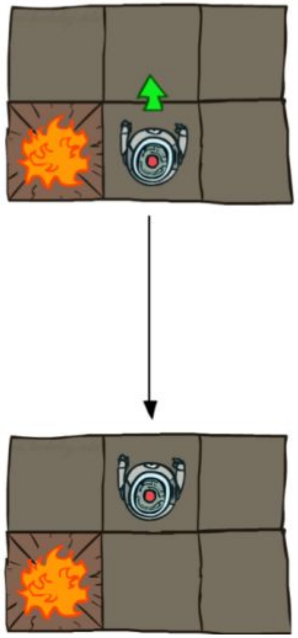
Non-deterministic Search Problems: Grid World

- The agent lives in a grid based world where walls may block the agent's path
- Actions do not always go as planned
  - 80% of the time the intended action is taken, unless a wall is blocking the path
  - Remaining 20% of the time, the agent will do something different than the intended action
- The agent receives rewards each time step
  - A big reward/punishment at the end
  - Small reward/punishment at each time step
- The goal is to maximize the sum of rewards

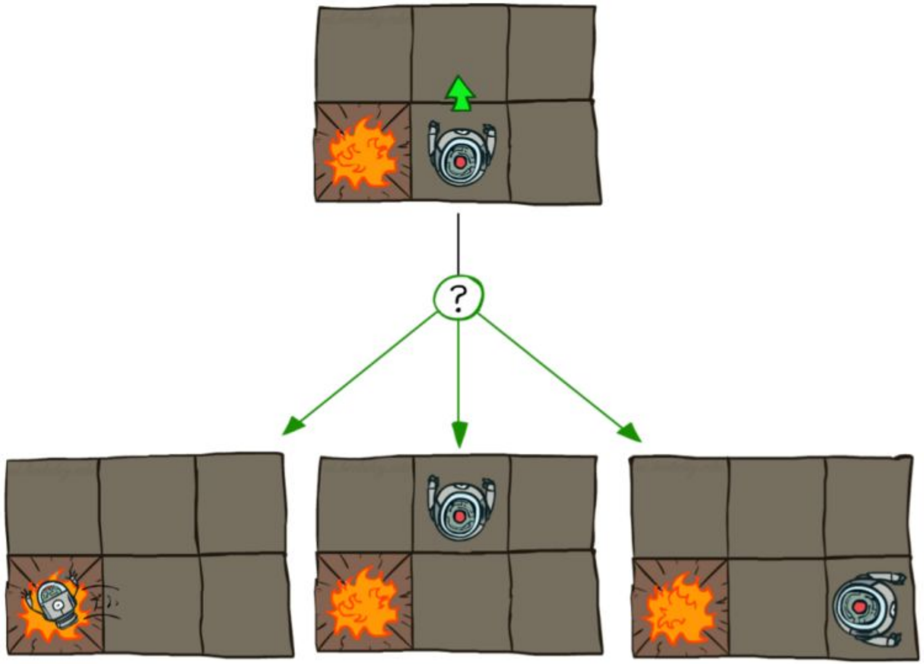


Grid World Actions

Deterministic Grid World



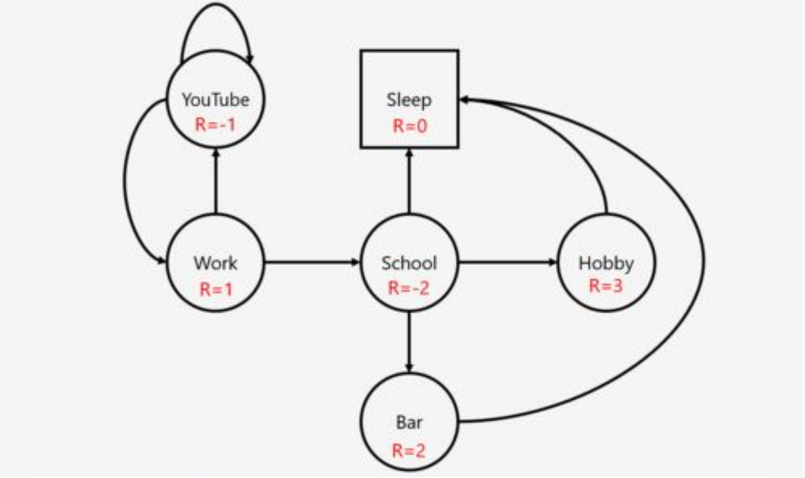
Stochastic Grid World

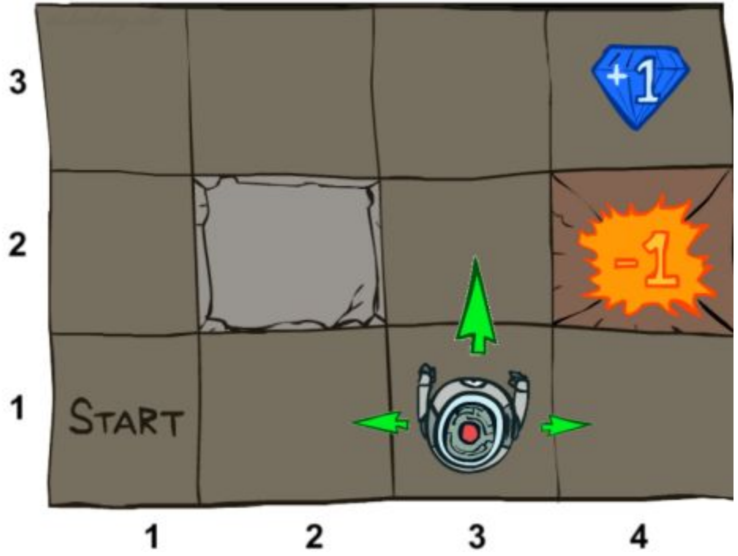


Formalizing MDPs

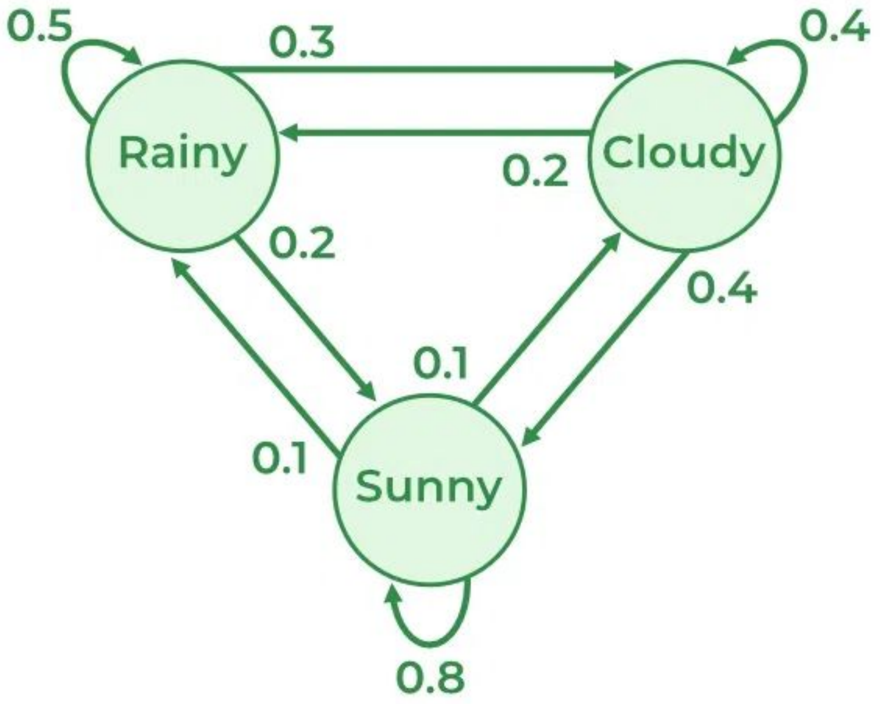
An MDP is defined by:

- Set of states $s \in S$
- Set of actions $a \in A$
- Transition function $T(s, a, s')$
    - Probability that taking action a from state s will take the agent to s'
    - $P(s' \mid s, a)$
- Reward function $R(s, a, s')$, or $R(s)$, or $R(s')$
- Start state
- Terminal State → Sometimes not present





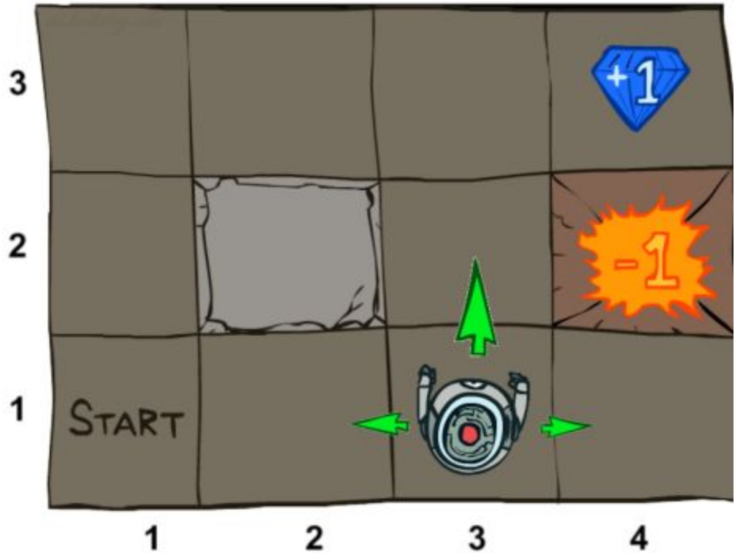
Markov Processes



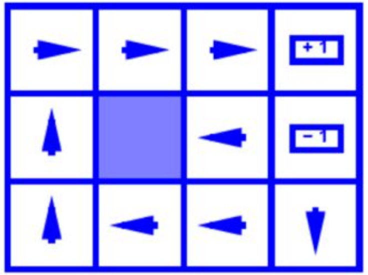
Policies

Policies

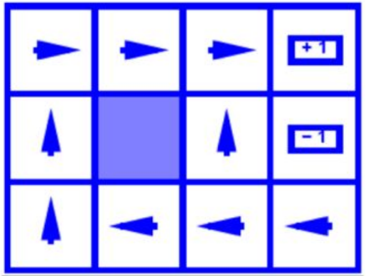
- In deterministic single-agent search problems, we wanted an optimal plan, or sequence of actions, from start to a goal
- We want an optimal policy $\pi^* : S \rightarrow A$
  - A policy π gives an action for each state
  - An optimal policy π^* is one that maximizes expected utility
- What is the difference between a policy and a plan?
  - What does expectimax calculate?



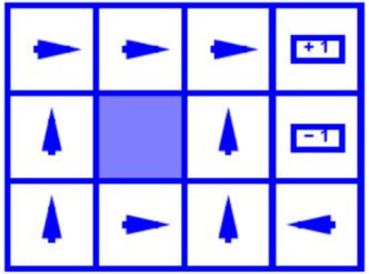
Optimal Policies



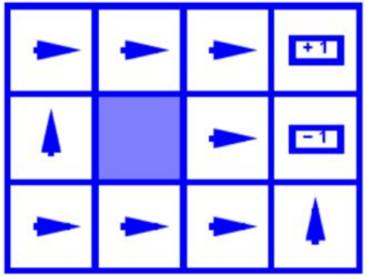
R(s) = -0.01



R(s) = -0.03



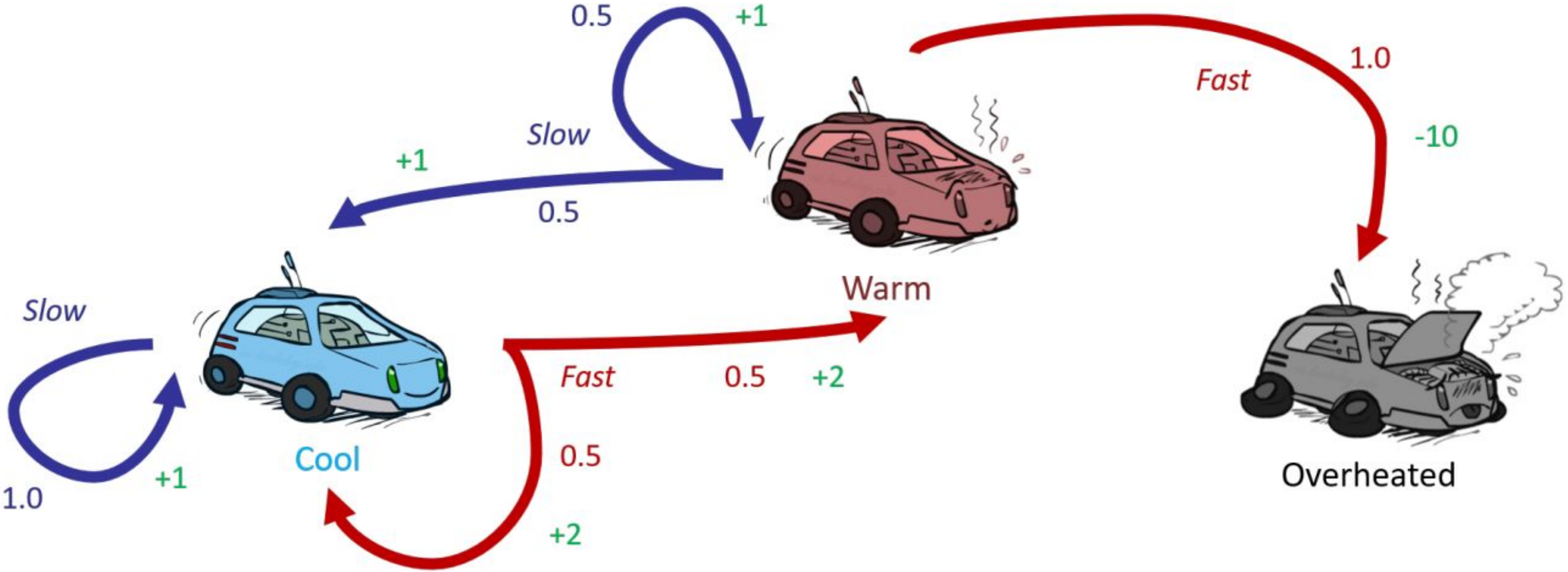
R(s) = -0.4



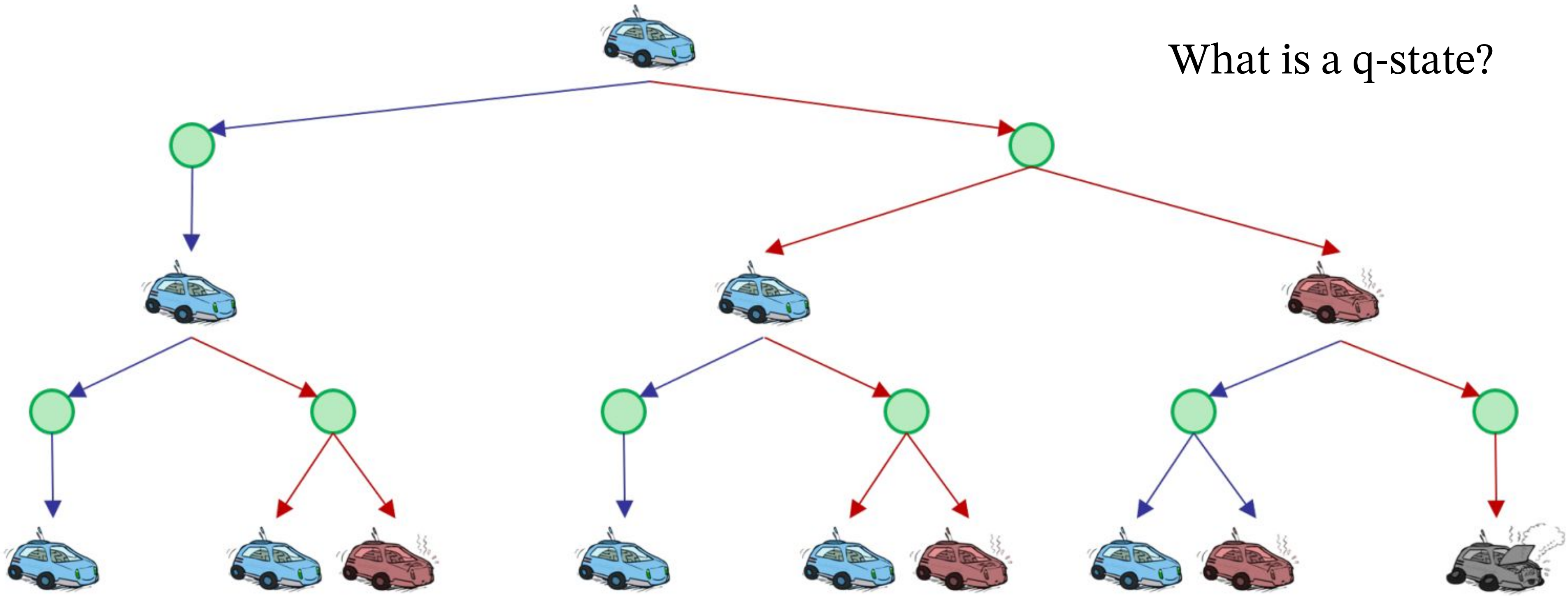
R(s) = -2.0

Racing

- There are three states: Cool, Warm, and Overheated
- There are two actions: Slow, Fast



Racing Search Tree



What is a q-state?

Utilities of Sequence

Utilities of Sequence

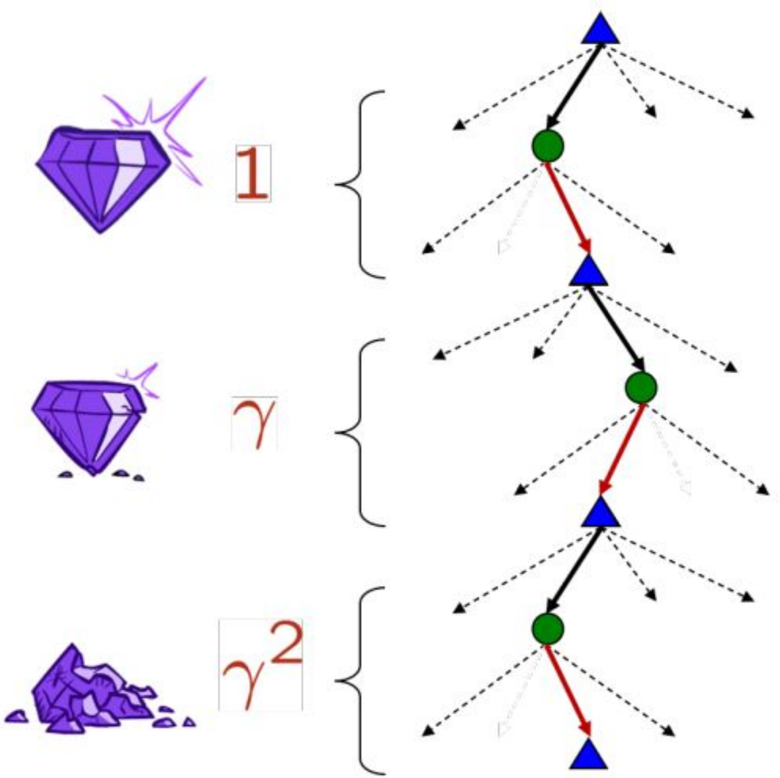
What preferences should an agent have when:

- Given the choice of more rewards or less rewards?
- Given the choice of getting the rewards early of getting them later?

For an agent:

- It is reasonable to maximize the sum of rewards
- It is also reasonable to prefer instant rewards rather than delayed rewards

Discounting



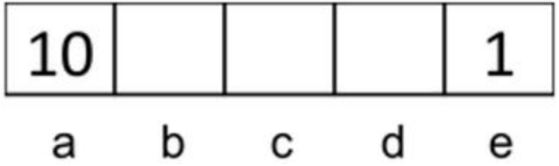
- At each time step, we multiply the rewards with a certain value γ
- Sooner rewards probably do have higher utility than later rewards
- Helps the algorithm converge

If $\gamma = 0.5$, which order is better?

- U ([1, 2, 3])
- U ([3, 2, 1])

For how long can we delay the rewards?

Stationary Preferences

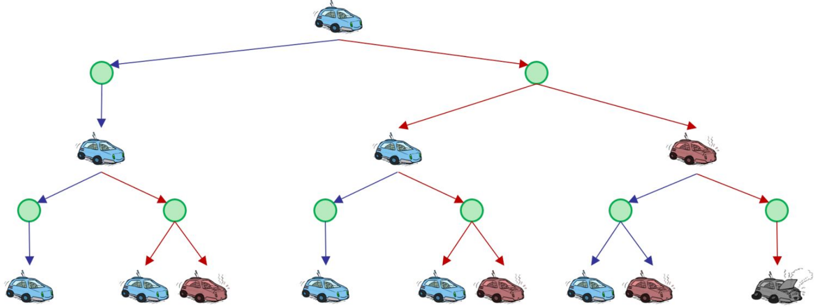


- If we assume stationary preferences:
  - $[a_1, a_2, a_3, \ldots] \succ [b_1, b_2, b_3, \ldots]$
  - $[r, a_1, a_2, a_3, \ldots] \succ [r, b_1, b_2, b_3, \ldots]$
- There are only two ways to define utilities
  - Additive $\rightarrow$ $U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2, \ldots$
  - Discounted $\rightarrow$ $U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2, \ldots$

For the deterministic grid world:

- What is the optimal policy for $\gamma = 1$?
- What is the optimal policy for $\gamma = 0.1$?
- For what value of γ, West and East are equally good when in state d?

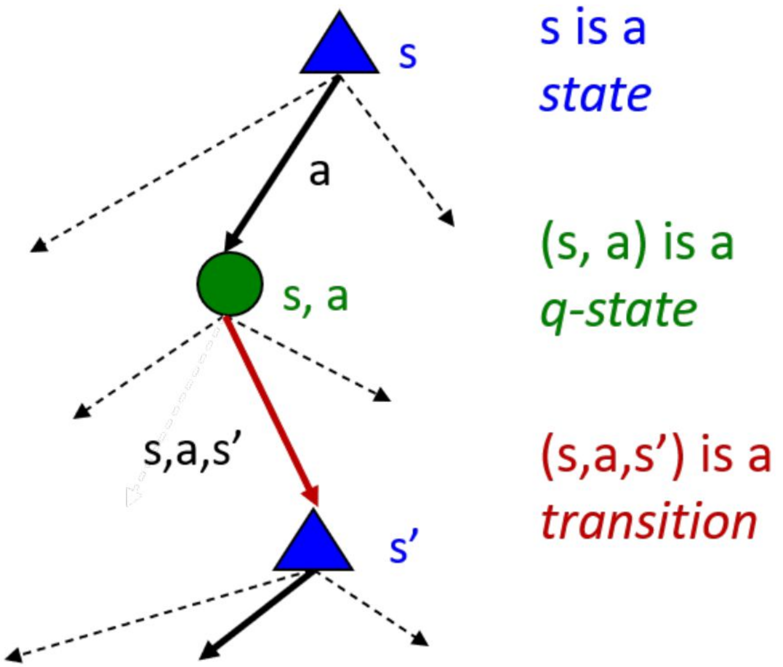
Infinite Utilities



- What if the game lasts forever?
- Keep a finite horizon → Similar to depth-limited search
  - Terminate after T steps → "Life" of the agent
  - Can give rise to non-stationary policies → π depends on the steps left
  - Similar to how drastic things can happen in sports when the game is near termination!
- Discounting → $0 < \gamma < 1$
  - Sum of a decreasing infinite series is finite
  - Smaller γ means smaller horizon
- Absorbing state
  - For every policy, a terminal state will eventually be reached
  - Similar to overheating or finding the terminal state

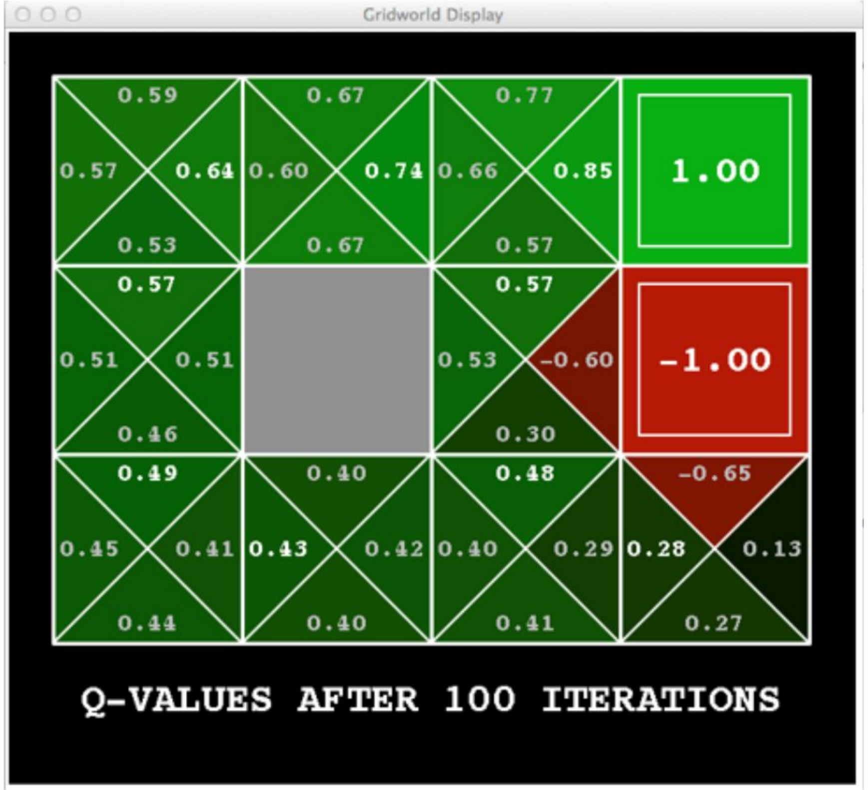
Optimal Quantities

- The value/utility of a state s
  - $V^*(s) \rightarrow$ Expected utility of starting from s and acting optimally
- The value/utility of a *q-state*(s, a)
  - $Q^*(s,a) \rightarrow$ Expected utility of starting from s, having taken action a and acting optimally
- The optimal policy
  - $\pi^*(s) \rightarrow$ Optimal action from state s



Gridworld Values

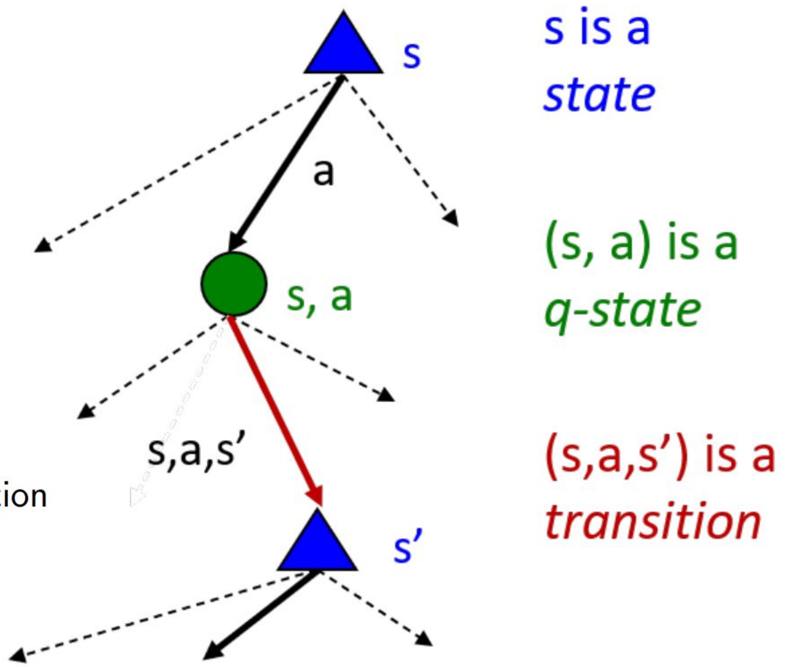
Gridworld Values



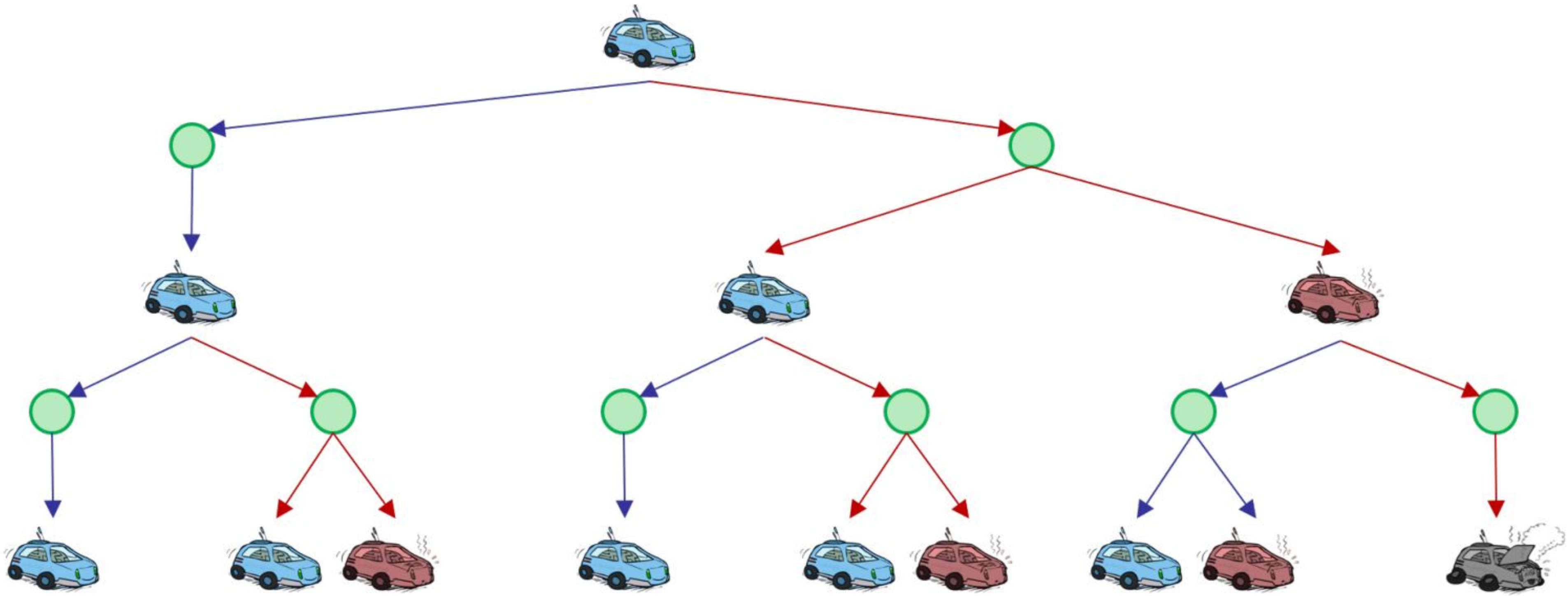
Values of States

To solve an MDP → We need to compute the value (expectimax) of a state (for all states to get an optimal policy)

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

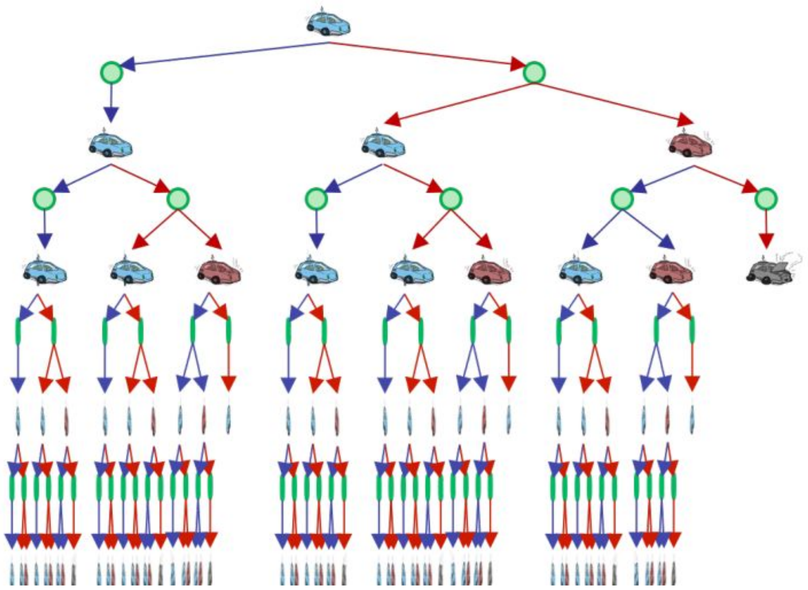


Racing Search Tree



Racing Search Tree

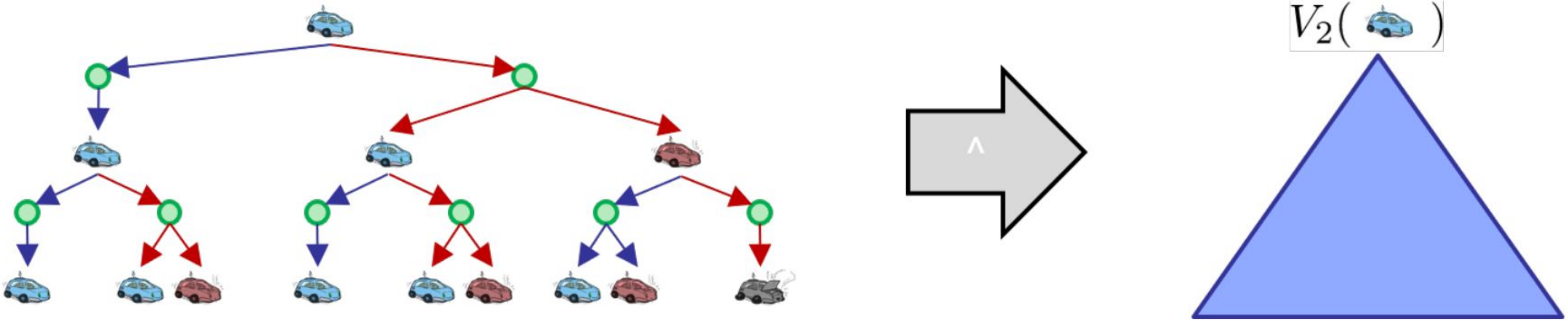
- Problem → Repeating states
- Solution → Caching
- Problem → Tree is infinite
- Solution → Depth-limited, Also because of γ, deeper states don't matter that much



We design Value Iteration Algorithm to address these issues from a new angle

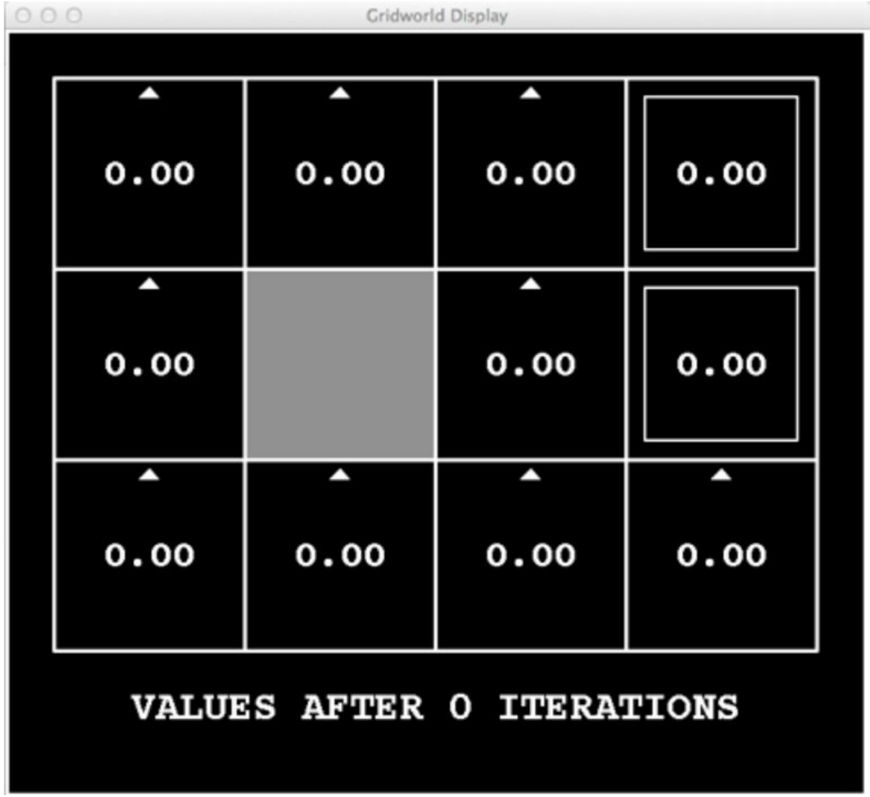
Value Iteration Algorithm

- Main idea → Time limited values
  - Start from the bottom
- $V_k(s)$ → The optimal value/utility of the state s if the game ends in k time steps
- It is similar to a k-depth expectimax search starting from state s



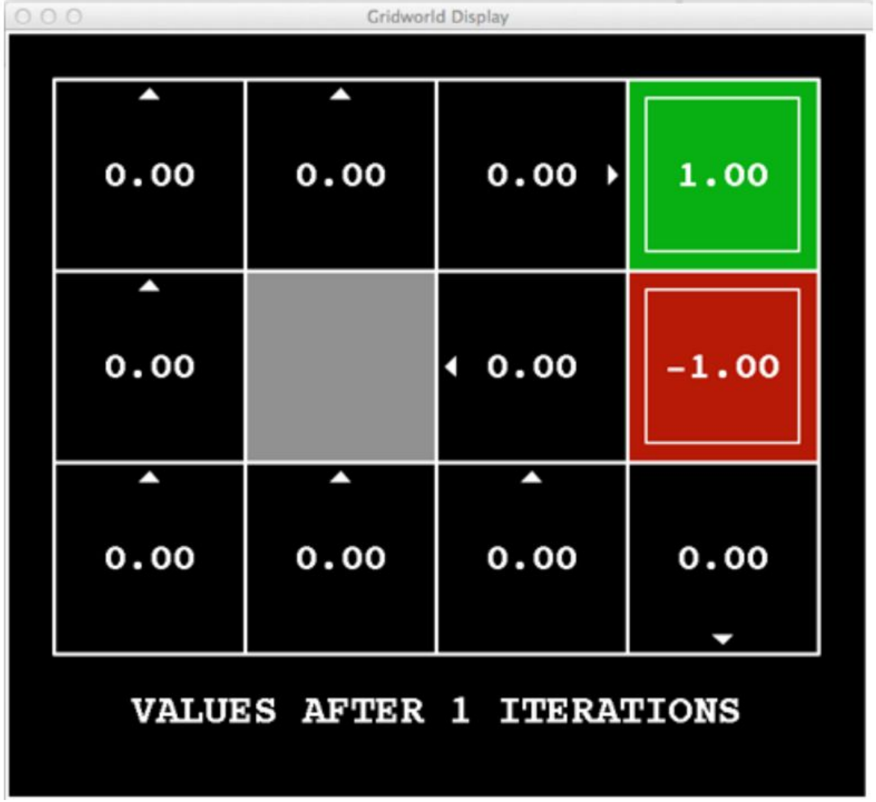
Gridworld Value Iteration

$k = 0$



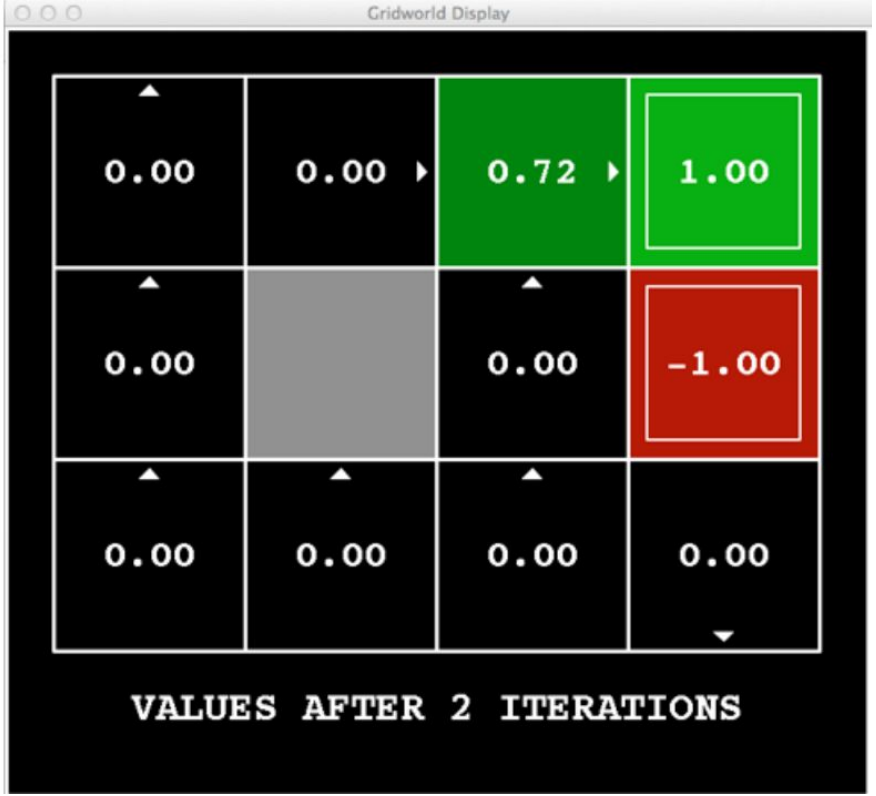
Gridworld Value Iteration

$k = 1$



Gridworld Value Iteration

$k = 2$

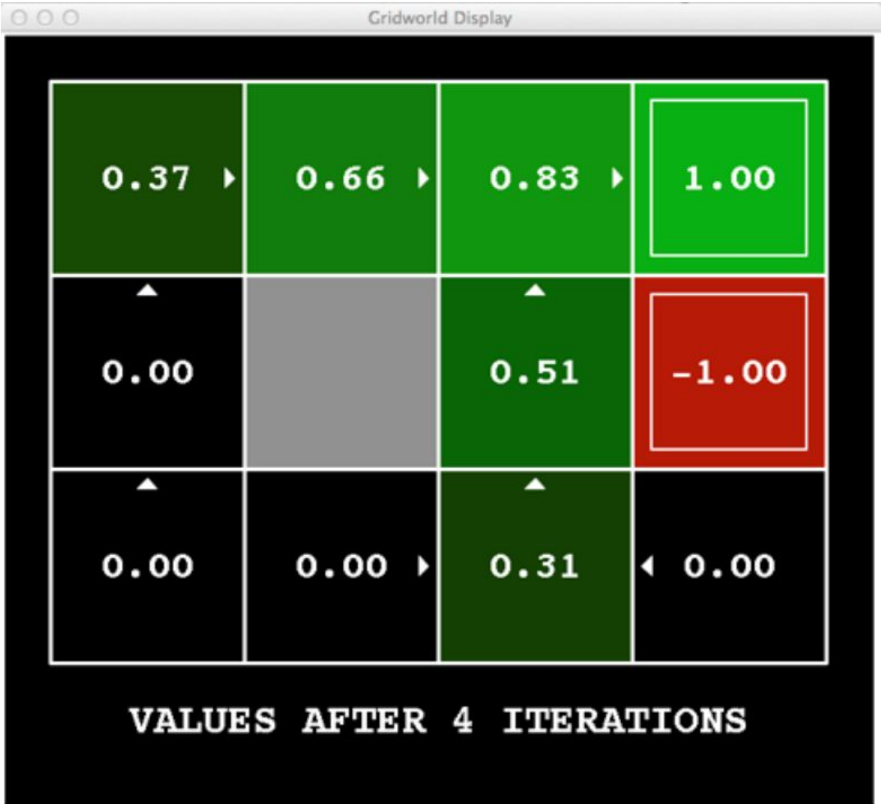


Gridworld Value Iteration

k = 3

Gridworld Value Iteration

$k = 4$

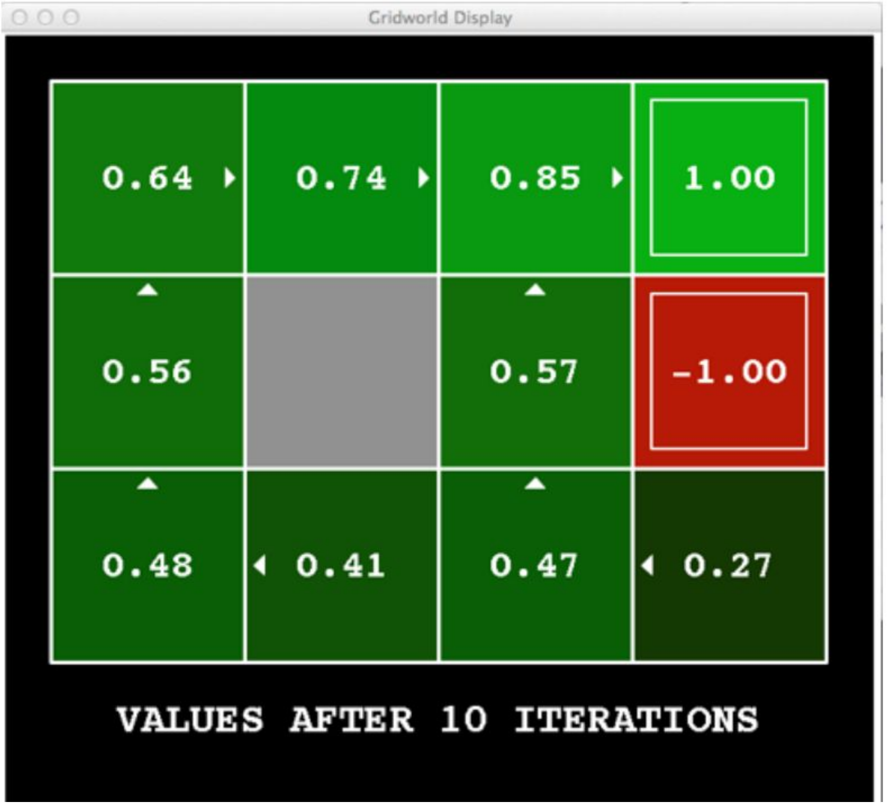


Gridworld Value Iteration

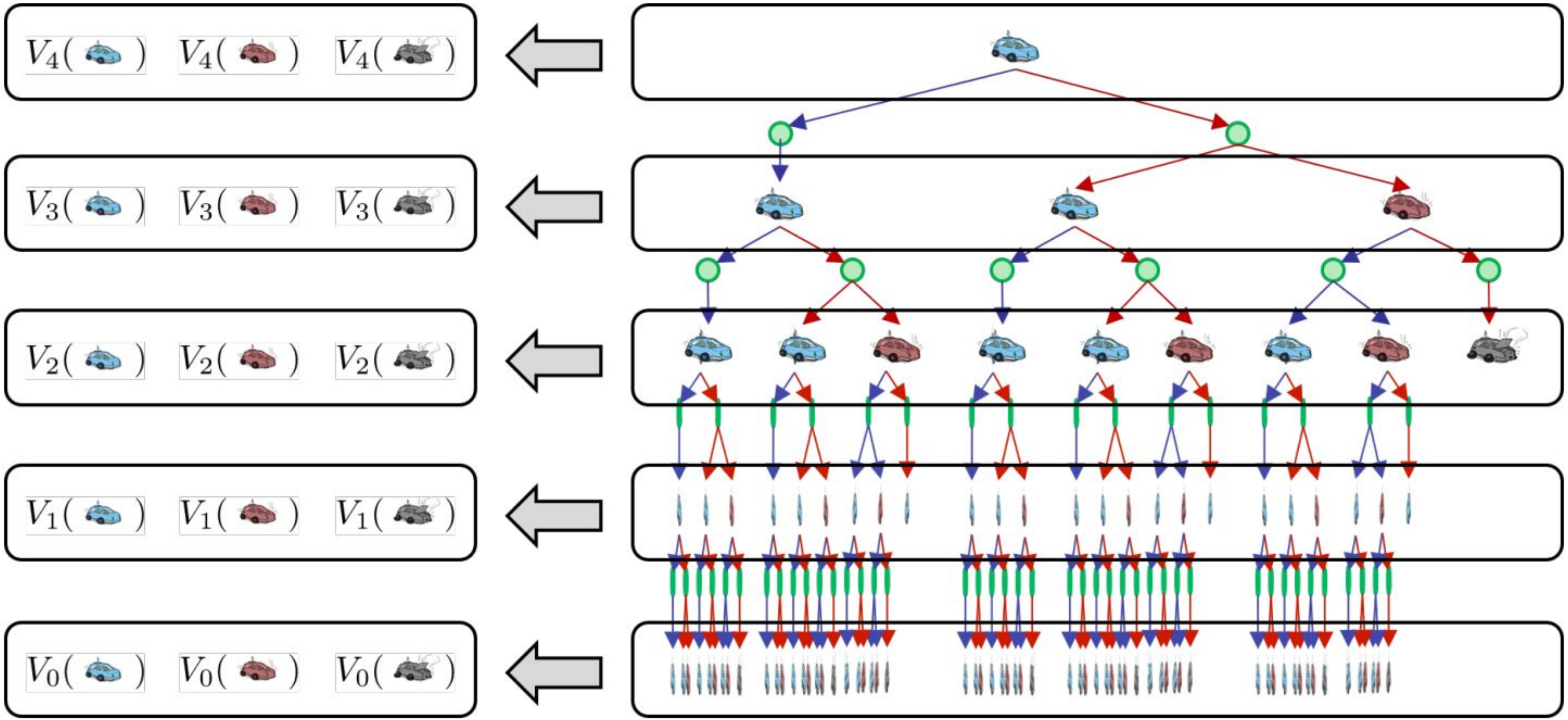
k = 5

Gridworld Value Iteration

$k = 10$



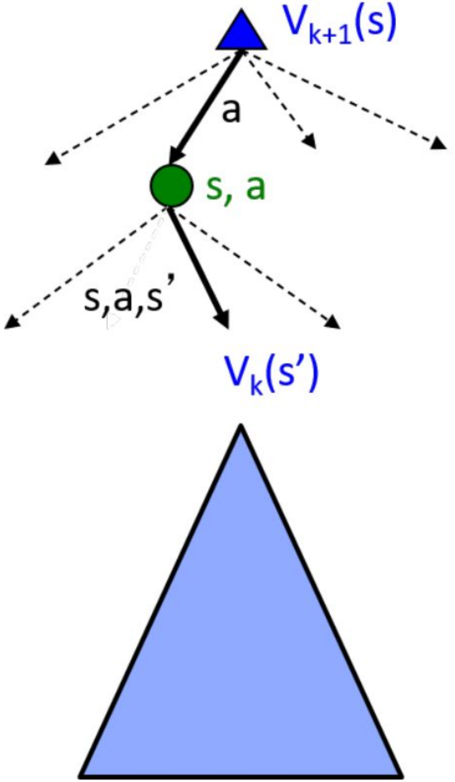
Computing Time Limited Values



Value Iteration

- Start with $V_0(s) = 0$; no time steps left means an expected reward sum of zero
- Given a vector/array of $V_k(s)$ values, look for one level higher

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



- Repeat until convergence
- Complexity → O(S^2A)

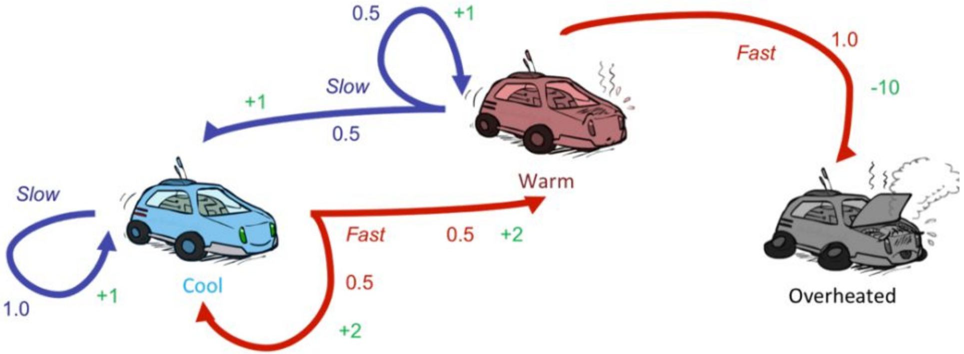
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V_k(s')\right]$$

Value Iteration

V_2

V_1

V_0

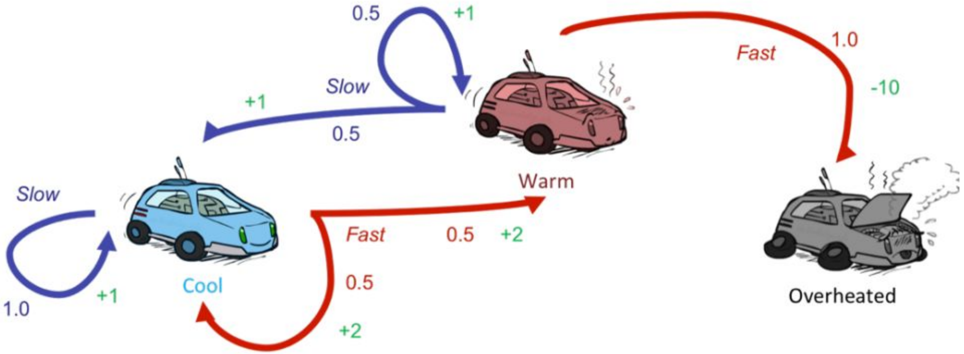


Assume no discount!

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

Value Iteration

V_2	3.5	2.5	0
V_1	2	1	0
V_0	0	0	0



Assume no discount!

Thank you