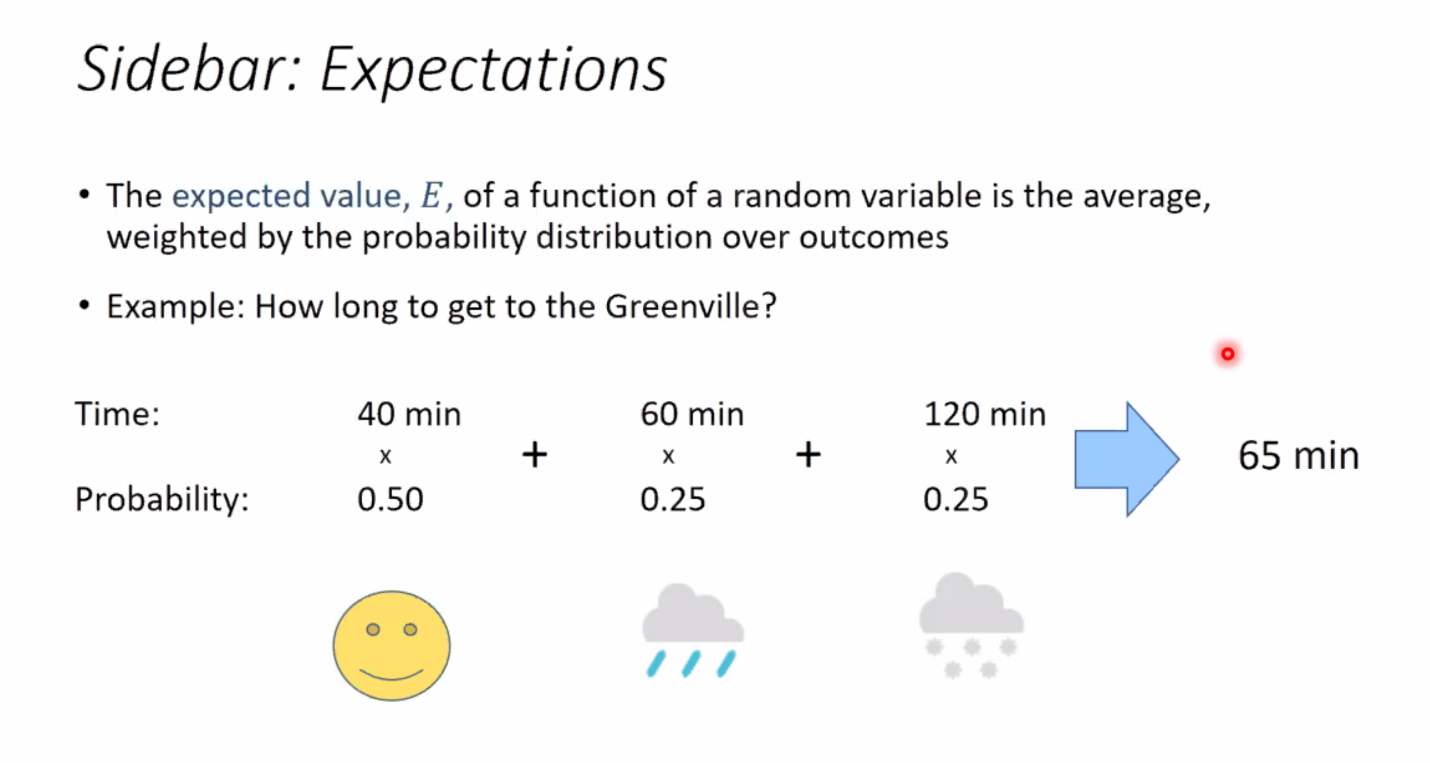
# Probabilities

* A random variable, X, represents an event whose outcome is unknown
* P(-) is called probability distribution (assigns weights to outcomes)
  + Random variable: X = weather tomorrow
  + Outcomes: X in {sunshine, light rain, thunderstorms}
  + Distribution: P(X=sunshine) = 0.5, P(X=rain) = 0.25, P(X=snow) = 25
* P(x) >= 0
* Sums of all probabilities = X
* With more evidence, probabilities may change
  + P(T=rain) = 0.25, P(T=rain | Neighbor carriers umbrella) = 0.50

Expectations

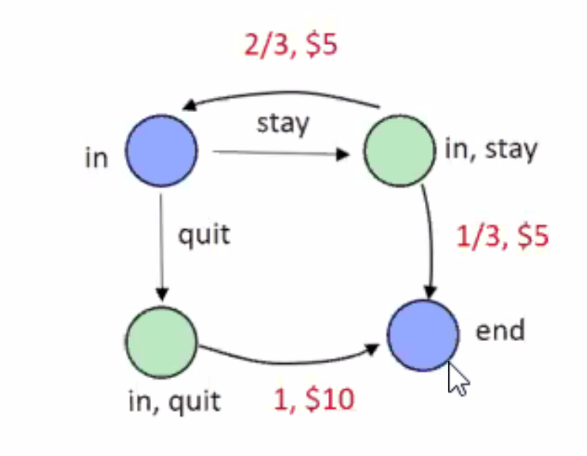
* The expected value, E, of a function of a random variable is the average, weighted by the probability distribution over outcomes
* How long to get to Greenville?



# Markov Decision Processes

Gridworld

* Noisy motion model
  + **80% of time, action N takes agent North (if no wall)**
  + **10% of the time, N takes agent west; 10% east**
  + If there is a wall in direction of the agent takes, the agent stays put
* Agent receives rewards each time step
  + Small “living” reward each step (can be negative or 0)
  + Big rewards come at the end (good or bad)
* Goal: Maximize sum of rewards

Simple game

* At each round
  + Option to stay or quit
  + If quit, game is ended and you get $10
  + If stay, you get $5 and then roll the dice
    - If result is 1 or 2, game is ended
    - Otherwise game contributes to the next round

Markov Decision Process

* Set of states S
* Set of actions A
* Transition function T(s, a, s’)
  + Also called model or dynamics
  + Sometimes P(s’|s, a)
* A reward function R(s, a, s’)
  + Sometimes just R(s) or R(s’)
* A start state s0 (maybe a terminal one as well)
* MDPs are non-deterministic search problems
* Discount factor /gamma
* Horizon H (can be infinite)
* Quantities so far:
  + \pi
    - Choice of action for each state
  + Utility

Markovian Assumption

* In markov decision processes, “Markov” means action outcomes depend only on the current states
* Eg 1 in Project 1, children of expanded state depended only on the current state (not the history)

Policy

* In deterministic search problems, we seek optimal sequences of actions from start to goal
* For MDPs, we seek an optimal policy /pi\*: S->A
  + A policy /pi gives an action for each state

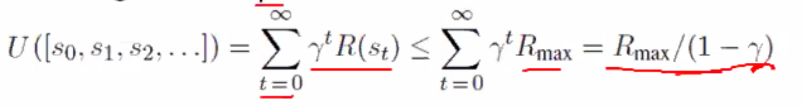
Evaluating a policy

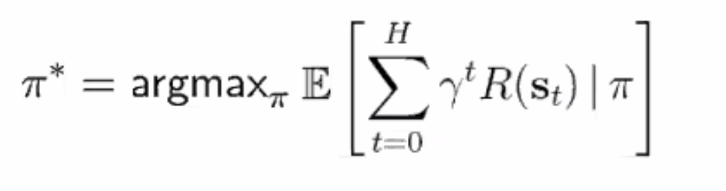
* Following a policy yields a random path
* The utility, U, of a policy is the (discounted) sum of the rewards along the path \
* Expected utility is the average of all possible paths
* An optimal policy, /pi\*, maximizes expected utility if followed

Discounting

* Reasonable to maximize sum of rewards
* Also reasonable to prefer rewards now to rewards later
* Solution: Values of rewards decay exponentially over time based on a discount factor 0 <= y <=
* Why?
  + Sooner rewards probably do have higher utility
  + Helps algorithms converge

Avoiding infinite rewards

* If game lasts forever, do we get infinite rewards?
* Solutions:
  + Finite horizon H
    - Termiate episodes after a fixed number of H steps
    - Gives nonstatioary policies (/pi depends on time left)
  + Discounting: Use 0 < \gamma < 1
  + Absorbing state: Guarantee that for every policy a terminal state will eventually be reached

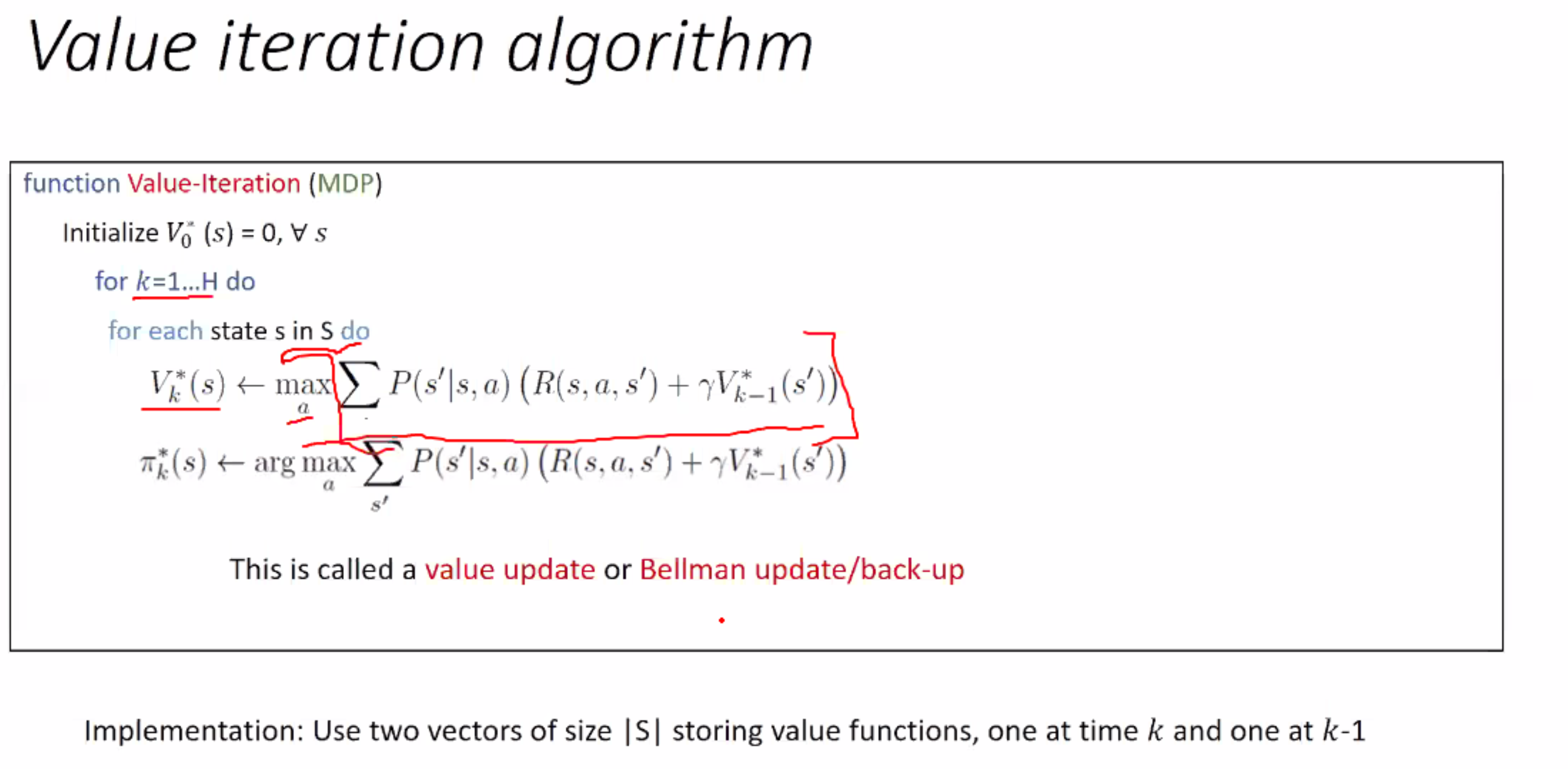
Solving MDPs

* Goal: Find optimal policy pi\*
* Exact methods based on dynamic programming
  + Value iteration
  + Policy iteration

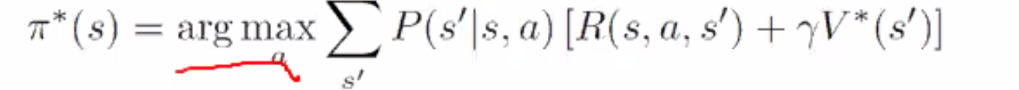
Optimal value function V\*

* V\*(s): Sum of discounted rewards starting in s
* Bellman equation: V\*(s) = maxa E[R(s,a,s’) + /gamma V\*(s`)\

Value iteration algorithm



Convergence of value iteration:

* Now we know how to act for ifinite horizon with discounted rwards
  + Run value iteration until convergence
  + This produces V\*, which in turn tells you how to act
* Policy may converge long before values do

Optimal utilities

* V\*(s) = expected utility starting in s and acting optimally
* Q\*(s, a) = expected utyility starting out