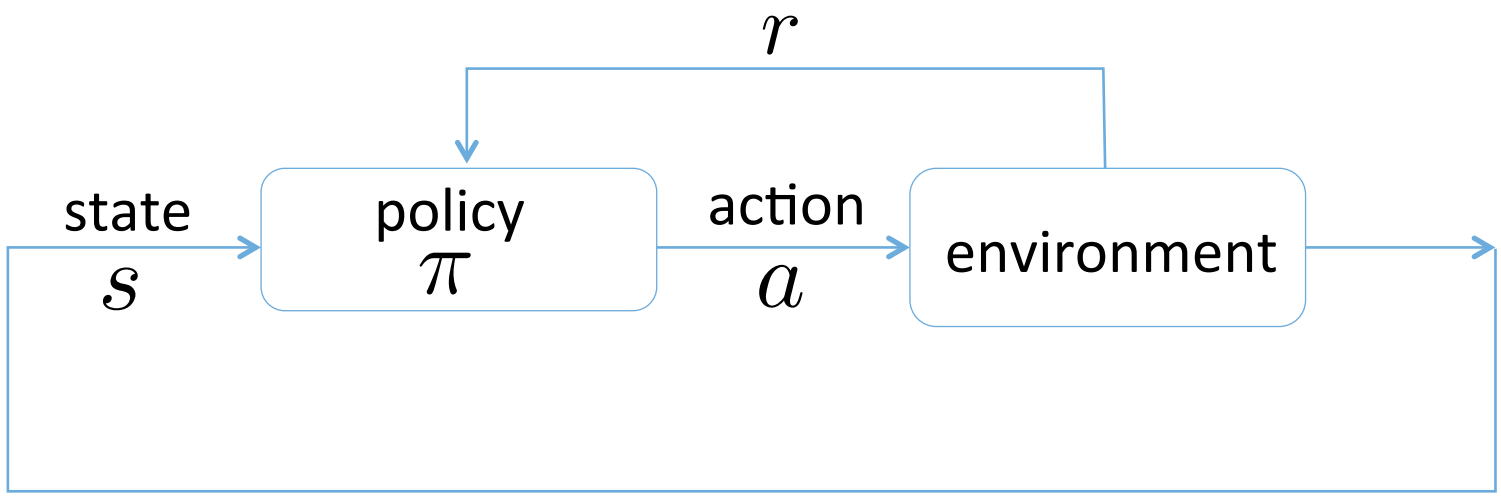
RL Basics

Reinforcement Learning – basics



maximize $G_t = r_{t+1} + \gamma r_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$

"return", cumulative discounted rewards

A Simple Solution to Learning a Policy: Copy an expert "Imitation Learning"

Observe what an expert does, and try to copy it:

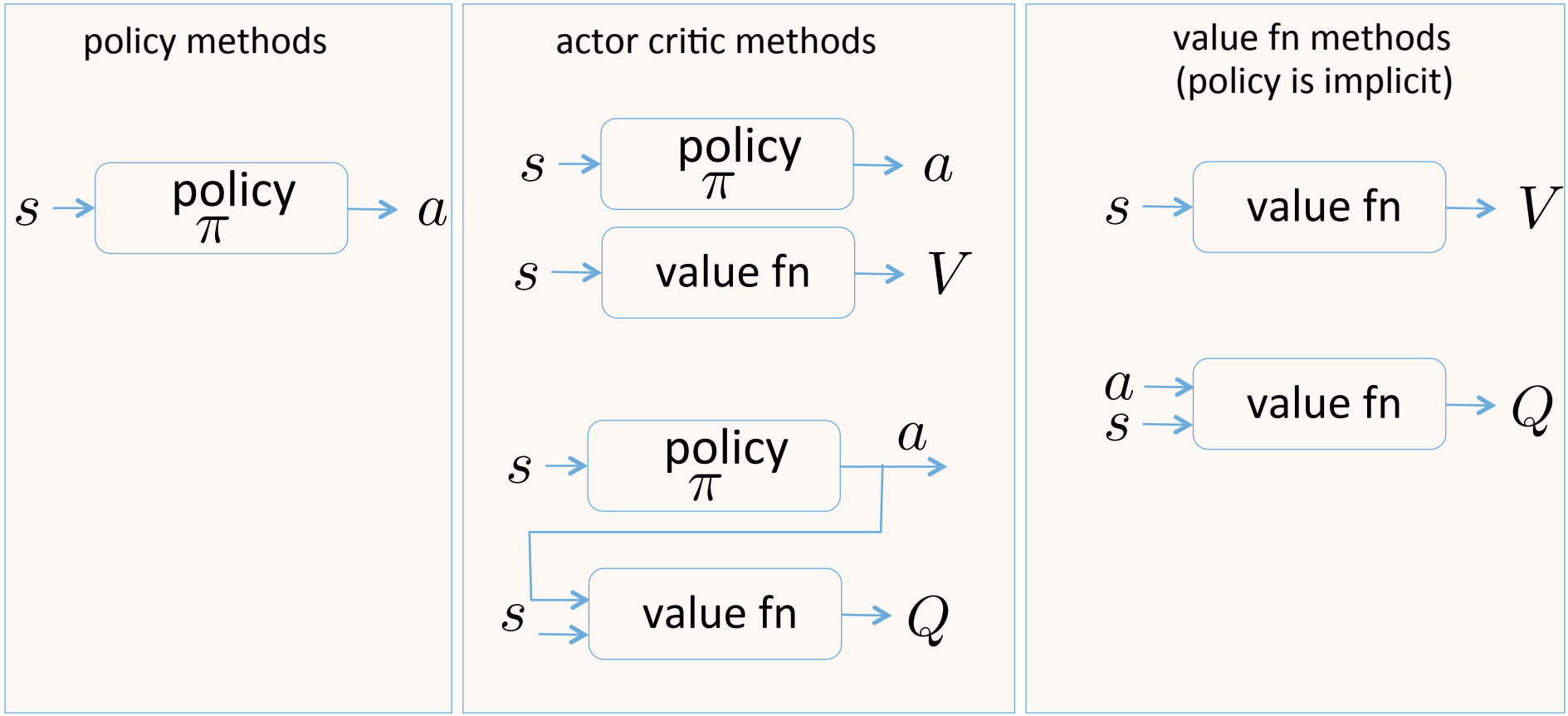
$$\{(s_i, a_i)\}$$

$$\Downarrow$$

$$a = \pi(s)$$

- requires an expert
- learned policy is unaware of the actual task (!)
- suffers from compounding errors over time
  - can be mitigated by collecting data during perturbations
- transfer between two Deep-NN policies:
  "policy distillation" "policy cloning"

Deep RL: key building blocks



Common Assumptions

- *episodic* tasks
  - repeated interactions with the world e.g., games, trials
  - begins from a standard starting state (or distribution of states)
  - ends when reaching terminal state, or a fixed time T
- *discounting:* γ
  - avoid infinite rewards when T=∞
  - summarize uncertainty abou the future
- *stochastic policies*: action probabilities $\pi(a|s)$ vs $a = \pi(s)$
  - "smoothes"the learning, e.g., probability of playing a card, steering left, etc
  - provides exploratory actions (for some algorithms)
  - usually switch to deterministic policy once learning is complete

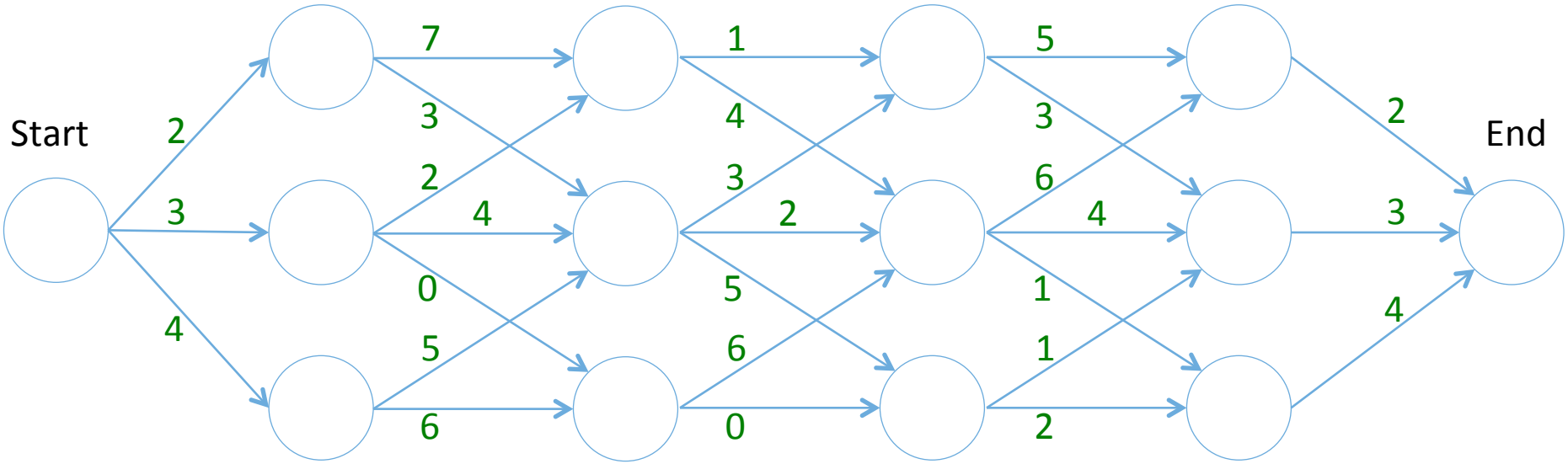
Common Assumptions

- Markov Property
  - the future only depends on the current state, not the history

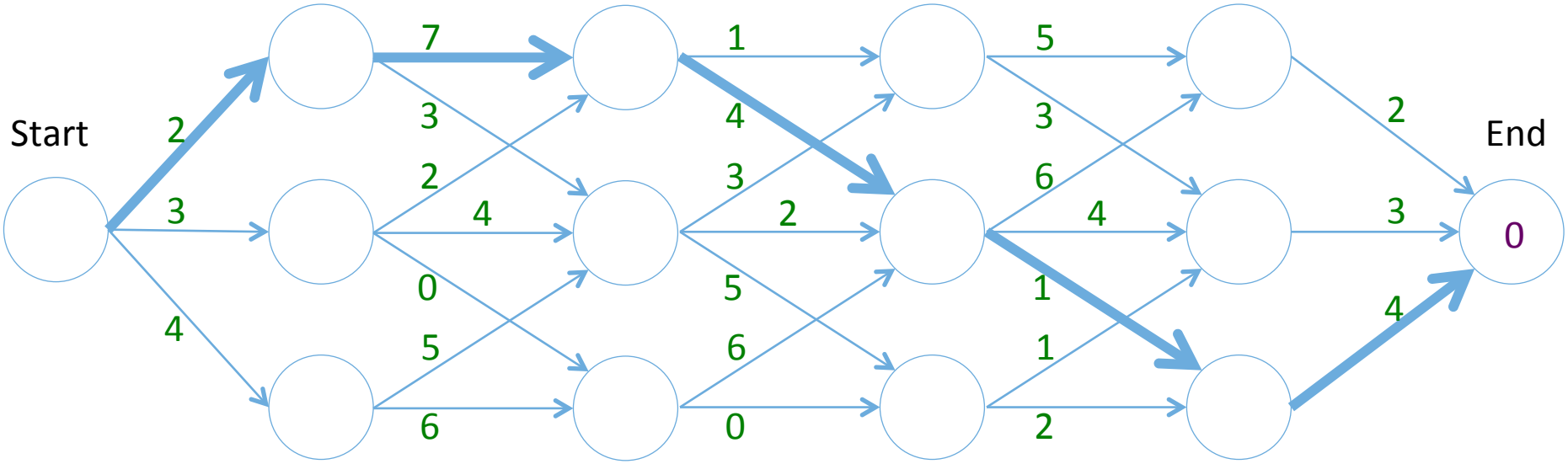
$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \ldots, S_t]$$

- Markov Decision Process (MDP) vs
- Partially Observable Markov Decision Process (POMDP)
  - state needs to be estimated; 'belief state'

Laying the Foundations:
A Simple Deterministic Example



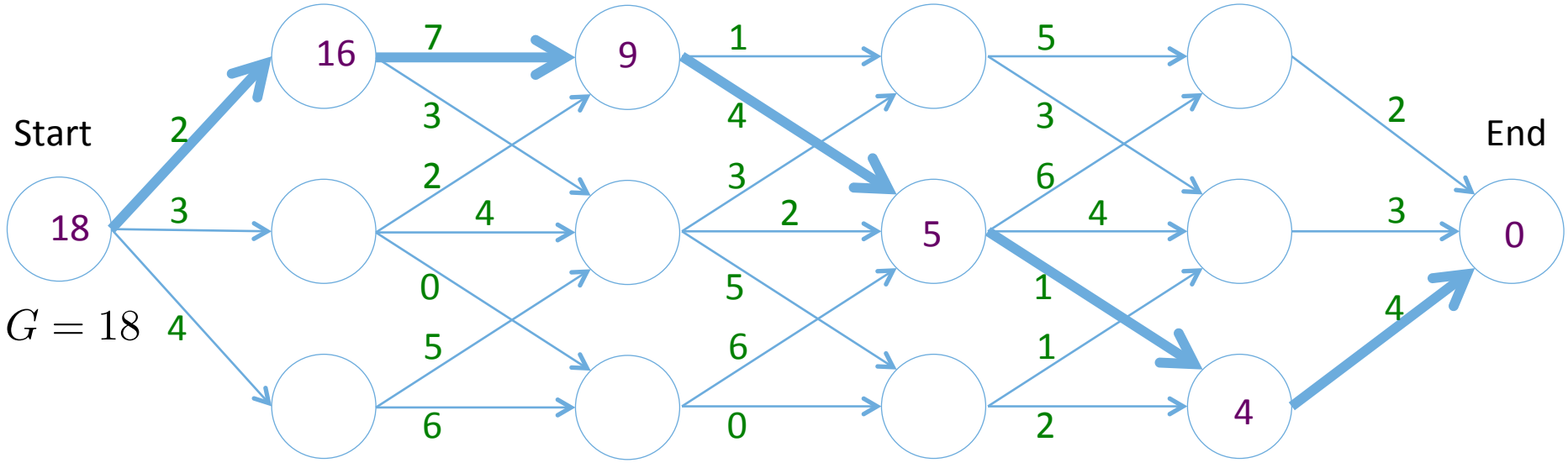
Returns for a given sequence of decisions



$$G_t = r_{t+1} + \gamma r_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Above we have $\gamma = 1$

Returns for a given sequence of decisions

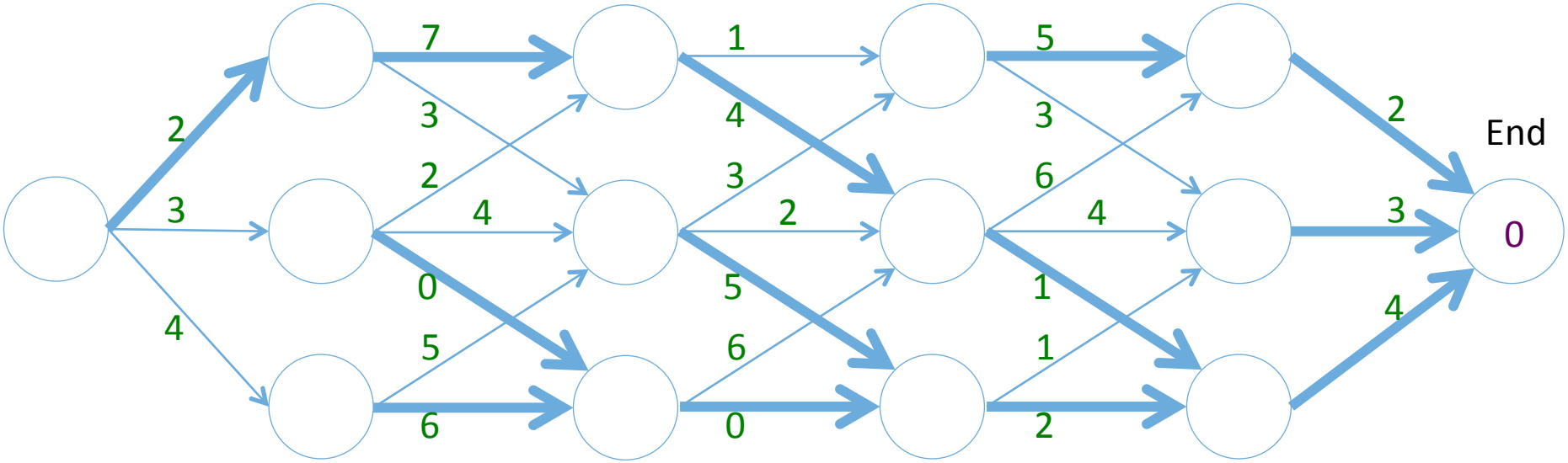


$$G_t = r_{t+1} + \gamma r_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$ Above we have $\gamma = 1$

$$\gamma = 0.9 : \; G(\text{start}) = 2 + (0.9)7 + (0.9)^2 4 + (0.9)^3 1 + (0.9)^4 4$$

Returns for a given policy π

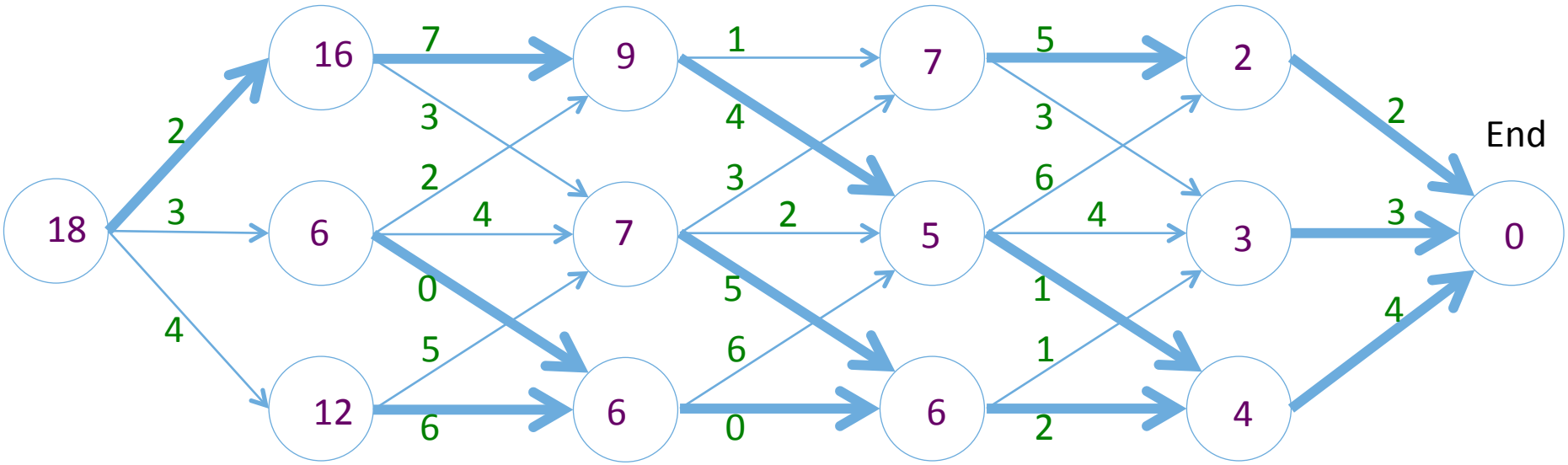
Assume that we begin in a random state, for this example, *"exploring start"* and therefore we care about the returns from all states.



$$\gamma = 1$$

$$G_t = r_{t+1} + \gamma r_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

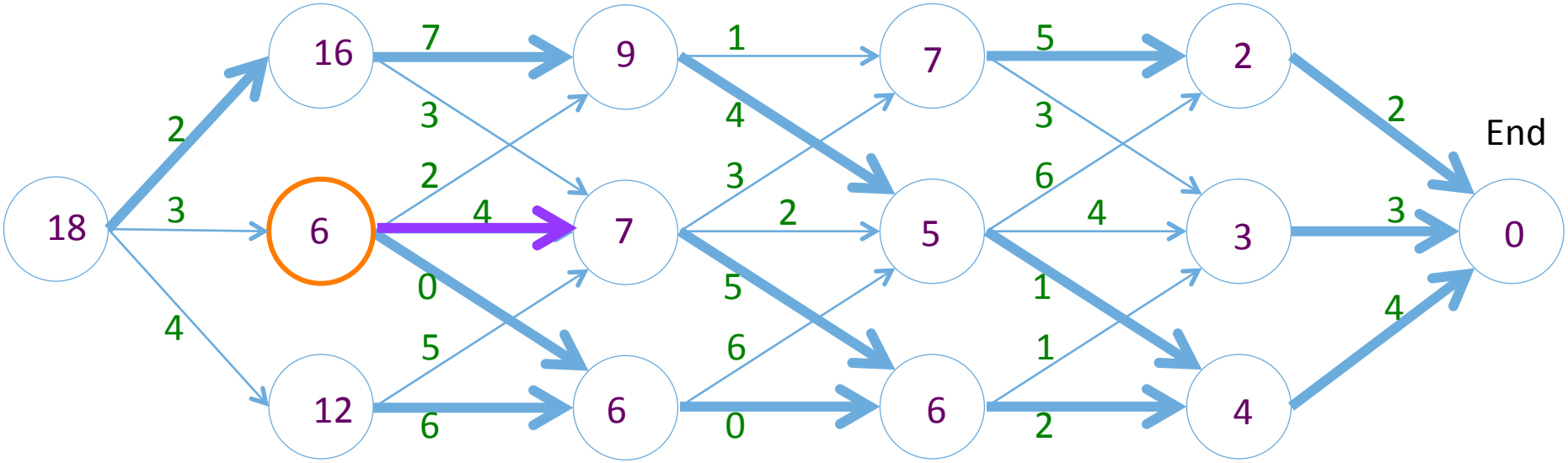
Returns for a given policy π



$$\gamma = 1$$

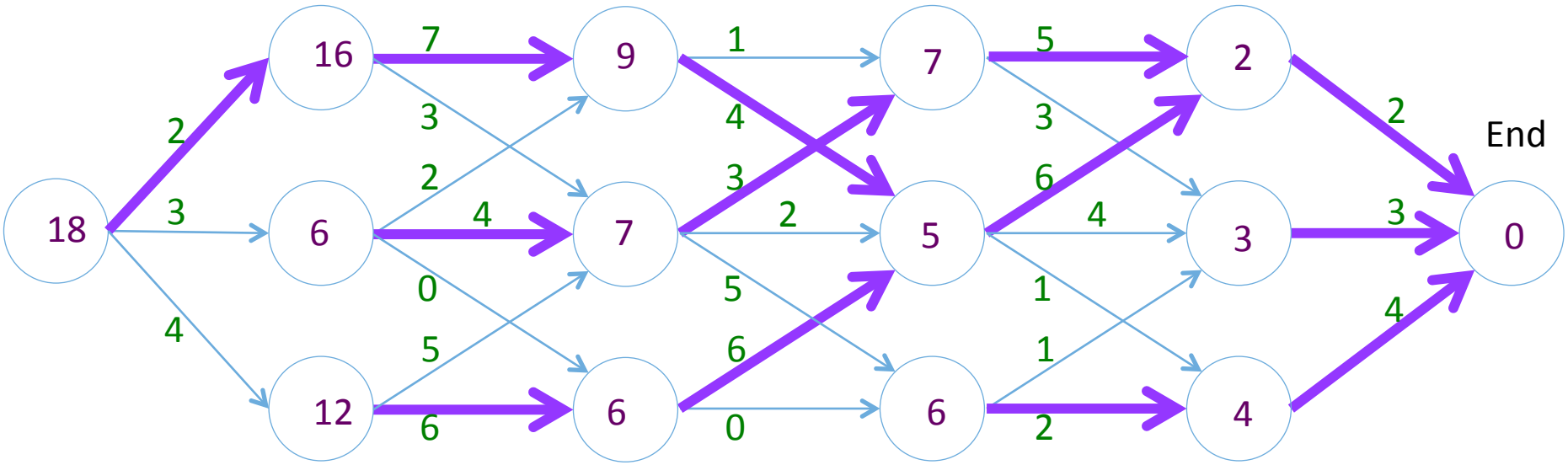
$$G_t = r_{t+1} + \gamma r_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Given the known returns, now improve the policy



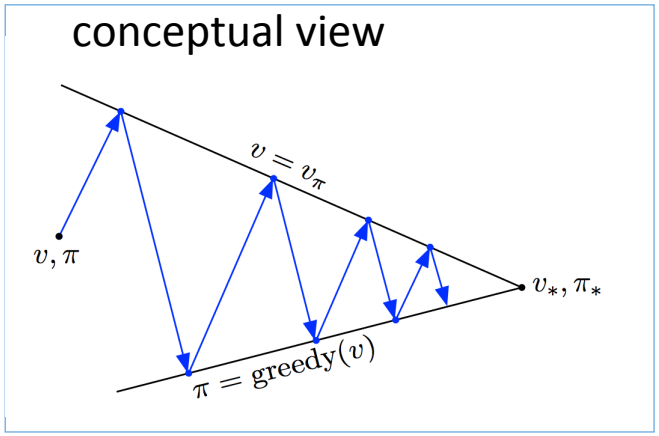
$$\pi(s) \leftarrow \arg\max \sum_{s',r} [r(s,a) + \gamma V(s')] \qquad \gamma = 1$$

Final improved policy



Terminology

- *Generalized Policy Iteration*



E: policy evaluation

$\pi \rightleftarrows V_\pi$

I: policy improvement

optimal policy,
optimal value fn

V_*, π_*

$$\pi_0 \xrightarrow{\text{evaluation}} V_{\pi_0} \xrightarrow{\text{improve}} \pi_1 \xrightarrow{\text{evaluation}} V_{\pi_1} \xrightarrow{\text{improve}} \pi_2 \xrightarrow{\text{evaluation}} \cdots \xrightarrow{\text{improve}} \pi_* \xrightarrow{\text{evaluation}} V_*$$

Terminology

- *state value* function V(s): $V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$
  - **expected** returns for a given state, under a given policy "how good is a state?"
  - for our deterministic example, $V_\pi(s) = G_\pi(s)$
- *value iteration, with bootstrapping*:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots$$
$$G_t = r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \ldots)$$
$$G_t = r_{t+1} + \gamma G_{t+1}$$

similarly for the value functions

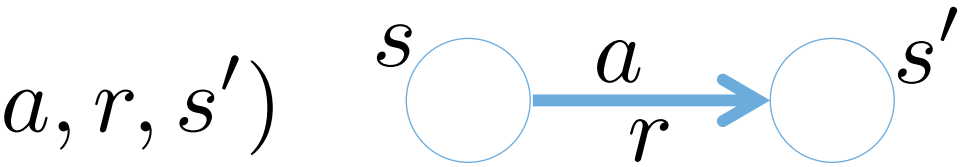
for each iteration k
  for each state s
    $V_{k+1}(s) = r + \gamma V_k(s')$

  - iterative value estimates depend on previous value estimates (!)
- *prioritized sweeping*: order updates based on size of expected change in V

Terminology

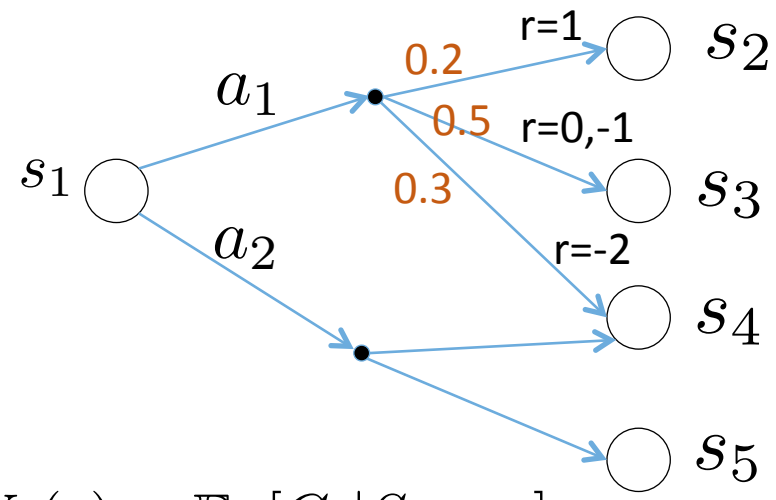
- *discount factor* $0 \leq \gamma \leq 1$
  - allows for continuing episodes, i.e., infinite length, which would otherwise produce a potentially infinite return
  - probability of episode continuing at any given time
  - preference for current rewards over future rewards, given that there may be more uncertainty about the future rewards
- *experience tuples*

$$(s, a, r, s')$$



  - from state s, we took action a, and received reward r and ended in state s'

Stochastic Environments



$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

$$V_{\pi,k+1}(s) = \sum_{s',r} p(s', r|s, a)[r + \gamma V_{\pi,k}(s')]$$

dynamics function: $p(s', r|s_1, a_1)$
"state transition model"

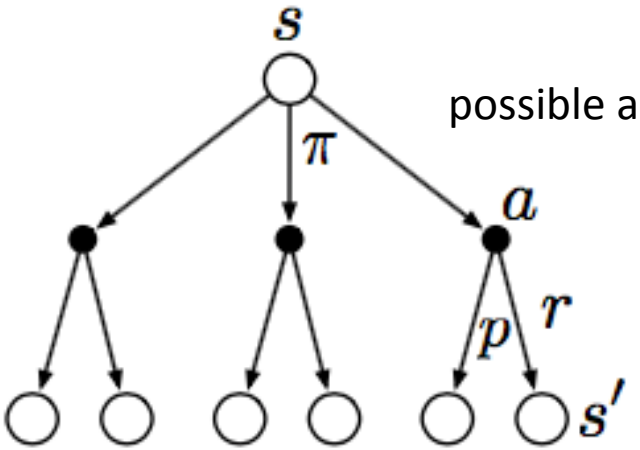
when in state s_1 and taking action a_1
what is the probability of ending up
in state s' and receiving reward r

$p(s', r\|s_1, a_1)$	next state	reward
0.2	s_2	1
0.4	s_3	0
0.1	s_3	-1
0.3	s_4	-2

sums to 1

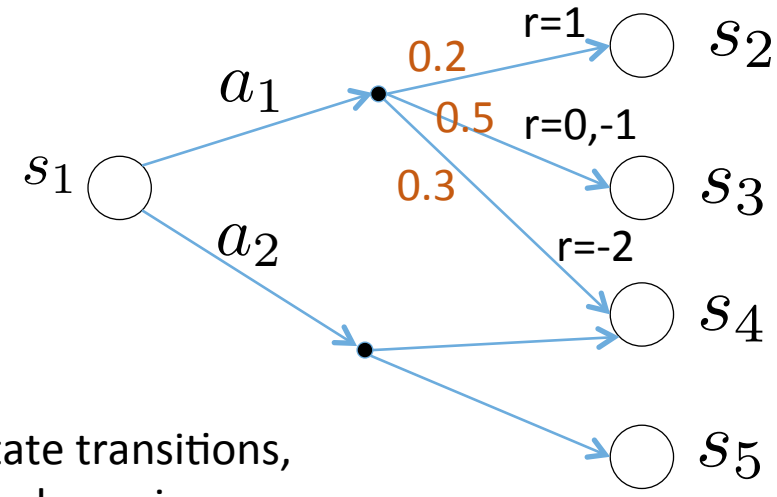
The value function provides the **expected returns**

Common “backup diagram”



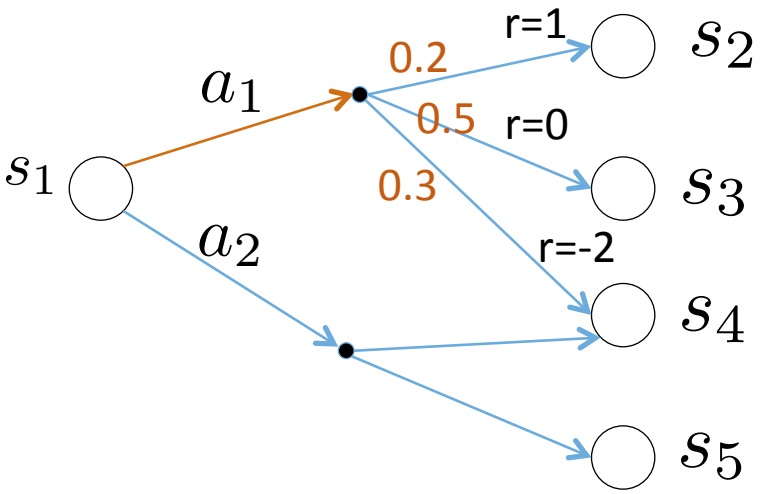
possible actions

distribution of state transitions,
due to stochastic dynamics,
with unique rewards

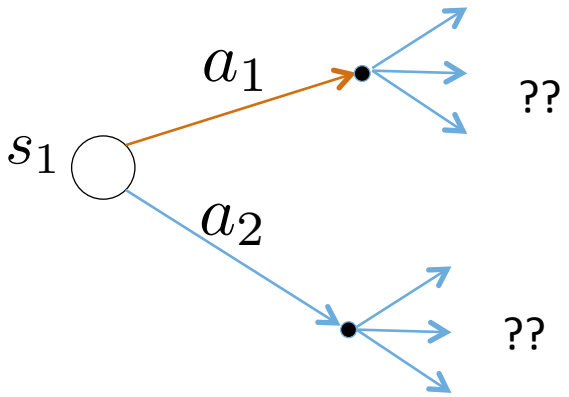


[Sutton, p81]

Model-based



Model-free



$$V_{\pi,k+1}(s) = \sum_{s',r} p(s', r|s, a)[r + \gamma V_{\pi,k}(s')]$$

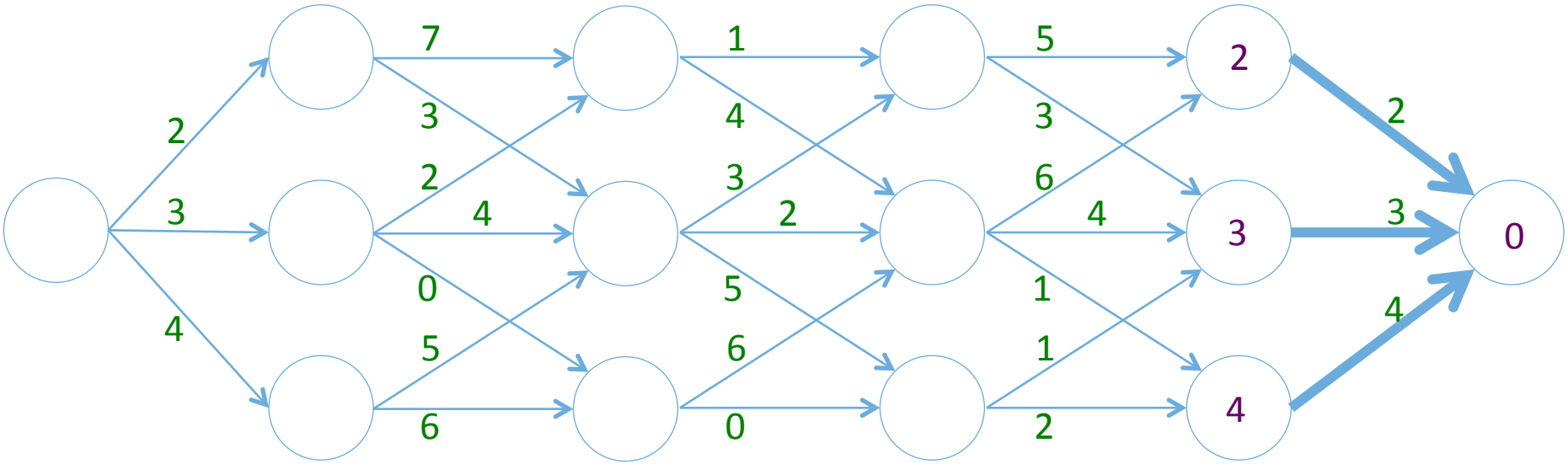
$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

model-based: all state-transitions and rewards are known in advance, follow some known distribution

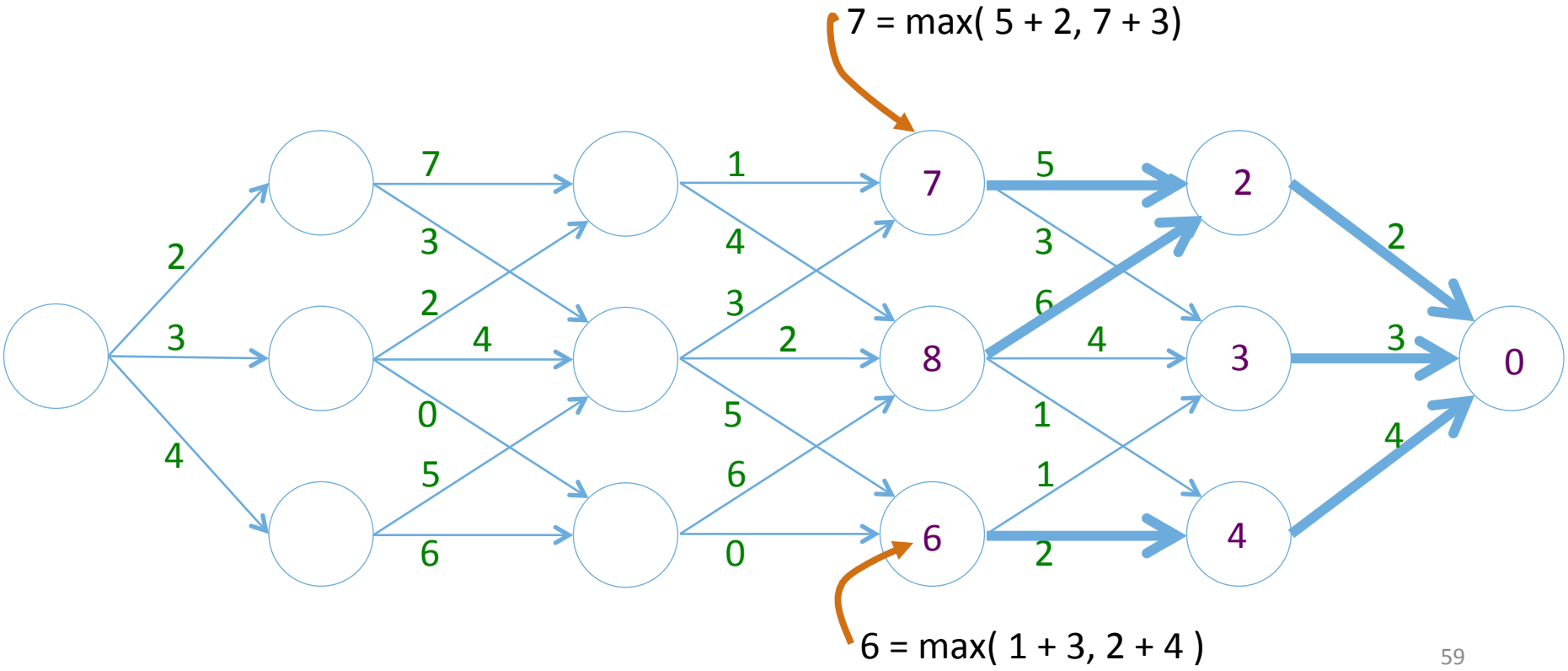
model-free: state-transitions and rewards are not known [therefore need to discover these thru experience!]

most basic RL: model-free); More advanced: model-based methods learn an approximate models for p()

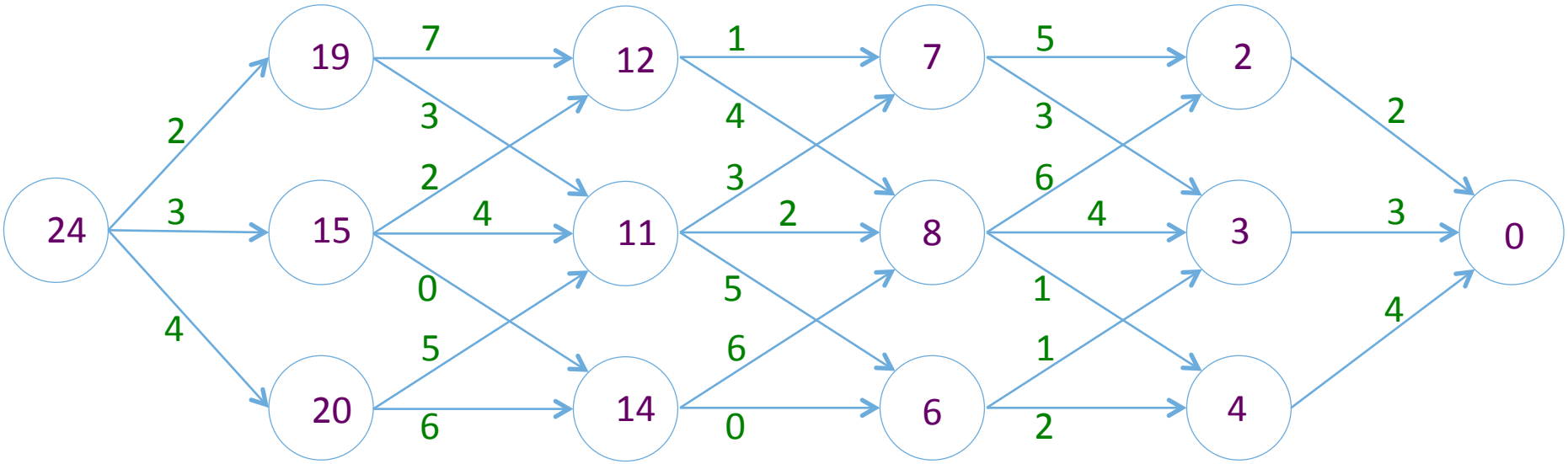
Determining the best returns for every state
(“Dynamic Programming” because the model is known)



“Bellman Backups”



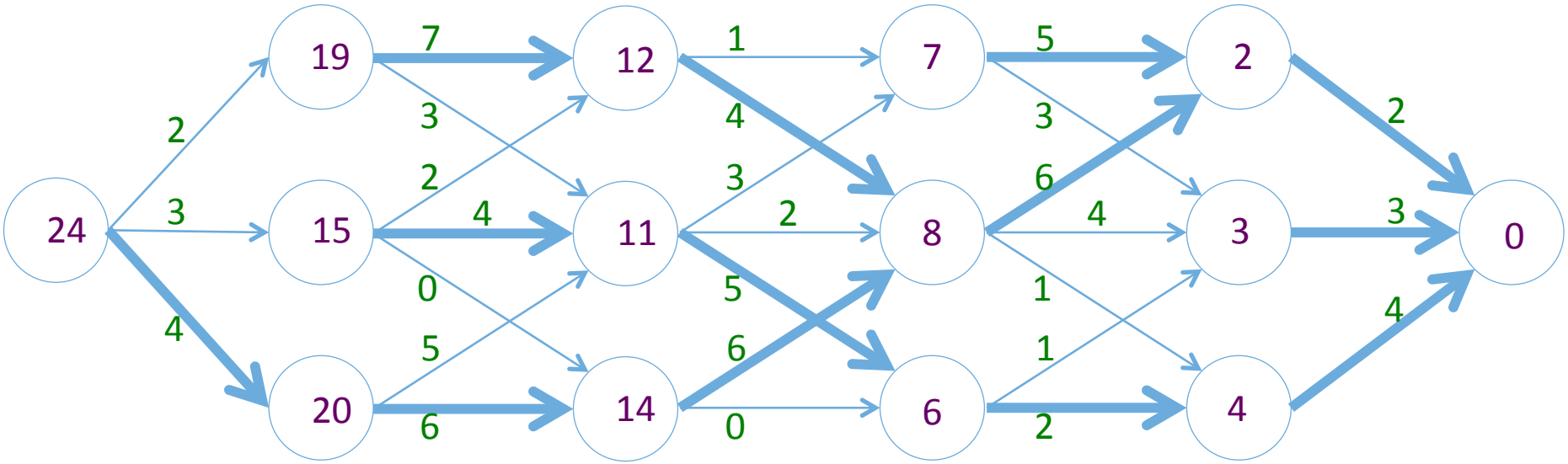
Final best returns for all states



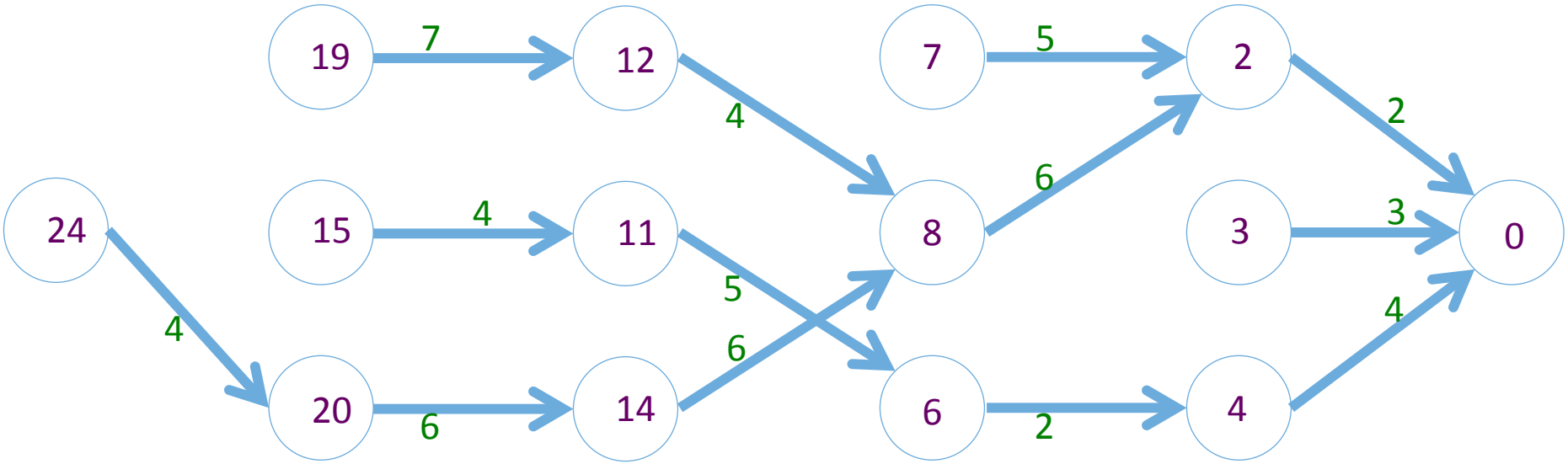
Optimal Policy (implicit in optimal value fn)

value fn update: $V_{k+1}(s) = \max_a (r + \gamma V_k(s'))$

related implicit policy: $\pi_{k+1}(s) = \arg\max_a (r + \gamma V_k(s'))$



Optimal Policy



More Definitions

- *Bellman backup*: a local iterative improvement of the value function
  - deterministic case: $V_{k+1}(s) = \max_a(r + \gamma V_k(s'))$
  - stochastic case: $V_{k+1}(s) = \max_a(\sum_{s',r} p(s', r|s, a)\,[r(s, a) + \gamma V_k(s')]$

Policy Objective Functions
(what is the overall thing that we wish to optimize?)

- episodic environments: use the start value

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

- continuing environments, e.g., learning how to walk
  - average value

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

  - average reward per time-step (largely equivalent; we'll typically use this)

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

for a stochastic policy

~[Silver]