Online Planning in MDPs: Monte Carlo Methods

Tom Silver

Machine Learning for Robot Planning

Princeton University

Fall 2025

Recap & Preview

- Last time: started *online planning* for MDPs
  - Current state known
  - Agent “in the wild”
  - Interleaving planning and execution

Recap & Preview

- Last time: started *online planning* for MDPs
  - Current state known
  - Agent "in the wild"
  - Interleaving planning and execution
- Considered *reachability* and *heuristics*
  - Expectimax search exploits reachability
  - Leaf heuristic evaluation, RTDP, determinization use heuristics

Recap & Preview

- Last time: started *online planning* for MDPs
  - Current state known
  - Agent “in the wild”
  - Interleaving planning and execution
- Considered *reachability* and *heuristics*
  - Expectimax search exploits reachability
  - Leaf heuristic evaluation, RTDP, determinization use heuristics
- Some MDPs are still too hard!
  - One hard case: *very large transition distributions*
  - Another hard case: *long horizons and sparse rewards*

MDPs with Very Large Transition Distributions

Recall Bellman backups:

- Given state s, for each a, *for each possible next state* s', update $V(s)$.

When number of possible next states is large, Bellman backups will be slow.

MDPs with Very Large Transition Distributions

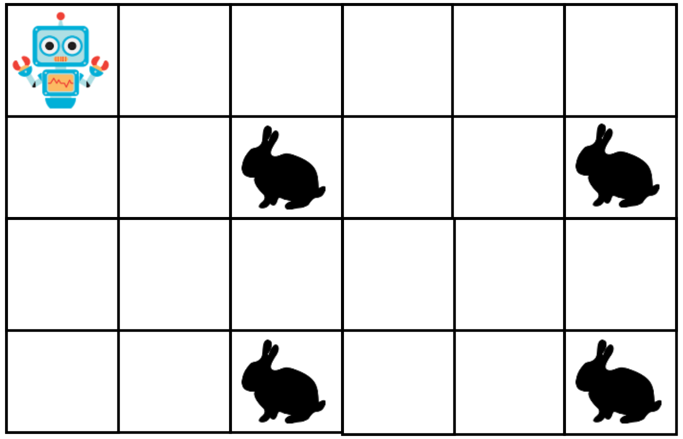
Recall Bellman backups:

- Given state s, for each a, *for each possible next state* s', update $V(s)$.

When number of possible next states is large, Bellman backups will be slow.

Examples:

1. "Chase" with multiple bunnies (or Pacman with ghosts)
2. Server farm; any server might fail with small probability
3. Pathological MDP, small probability of transitioning anywhere



MDPs with Very Large Transition Distributions

Recall Bellman backups:

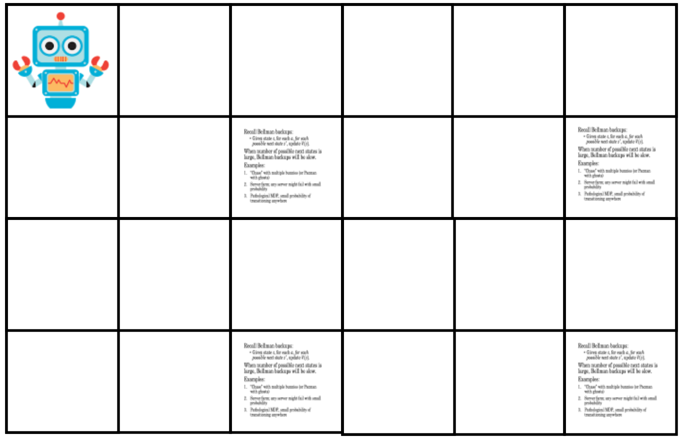
- Given state s, for each a, *for each possible next state* s', update $V(s)$.

When number of possible next states is large, Bellman backups will be slow.

Examples:

1. “Chase” with multiple bunnies (or Pacman with ghosts)
2. Server farm; any server might fail with small probability
3. Pathological MDP, small probability of transitioning anywhere

Almost all methods we’ve seen use Bellman backups. What’s the exception?



Simulator Access to MDPs

- Possible next states may be too big to enumerate.
  - Example: server farm with 100 servers, 2^{100} next possible states

Simulator Access to MDPs

- Possible next states may be too big to enumerate.
  - Example: server farm with 100 servers, 2^{100} next possible states
- Even if we can't enumerate, it may be possible to efficiently sample next states from the transition model, given s and a
  - Example: flip a coin 100 times to sample a next state

Simulator Access to MDPs

- Possible next states may be too big to enumerate.
  - Example: server farm with 100 servers, 2^{100} next possible states
- Even if we can't enumerate, it may be possible to efficiently sample next states from the transition model, given s and a
  - Example: flip a coin 100 times to sample a next state

- **Simulator access** (a.k.a. *generative access*) to an MDP:

We can only sample $s' \sim P(\cdot \mid s, a)$.

Simulator Access to MDPs

- Possible next states may be too big to enumerate.
  - Example: server farm with 100 servers, 2^{100} next possible states
- Even if we can't enumerate, it may be possible to efficiently sample next states from the transition model, given s and a
  - Example: flip a coin 100 times to sample a next state
- **Simulator access** (a.k.a. *generative access*) to an MDP:

We can only sample $s' \sim P(\cdot \mid s, a)$.

We're going to need some new planning algorithms…

Monte Carlo Bellman Backups

- Idea: replace full Bellman backup with **Monte Carlo (MC) Bellman backup**, which samples next states instead.

Monte Carlo Bellman Backups

- Idea: replace full Bellman backup with **Monte Carlo (MC) Bellman backup**, which samples next states instead.
- Another view: we're approximating the transition distribution with a sampling distribution

Monte Carlo Bellman Backups

- Idea: replace full Bellman backup with **Monte Carlo (MC) Bellman backup**, which samples next states instead.
- Another view: we're approximating the transition distribution with a sampling distribution

MonteCarloBellmanBackup$(s, \mathtt{V}, \mathcal{S}, \mathcal{A}, P, R, \gamma, w)$

$\mathtt{vs} = -\infty$ // New estimate for $V(s)$
for each $\mathtt{a} \in \mathcal{A}$
    $\mathtt{qsa} = 0$ // New estimate for $Q(s, a)$
    repeat w times
        $\mathtt{ns} \sim P(\cdot \mid \mathtt{s}, \mathtt{a})$ // Simulator access only
        $\mathtt{qsa} = \mathtt{qsa} + \frac{1}{w}(R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \mathtt{V}[\mathtt{ns}])$
    $\mathtt{vs} = \max(\mathtt{vs}, \mathtt{qsa})$
return $\mathtt{vs}$

Finite horizon case is analogous

Sparse Sampling

- **Sparse sampling** = Expectimax + MC Bellman backups

Sparse Sampling

- **Sparse sampling** = Expectimax + MC Bellman backups
- Nice property: can get optimality guarantees that depend only on w and H, *not* on $|\mathcal{S}|$ (Kearns, Mansour, and Ng 1999).

SPARSESAMPLING$(s_0, \mathcal{S}, \mathcal{A}, P, R, H, w)$

// a.k.a. MONTECARLOEXPECTIMAXSEARCH
return $\operatorname{argmax}_{\mathsf{a}} \mathrm{Q}(s_0, \mathsf{a}, 0, \mathcal{S}, \mathcal{A}, P, R, H, w)$

$\mathrm{Q}(\mathsf{s}, \mathsf{a}, \mathsf{t}, \mathcal{S}, \mathcal{A}, P, R, H, w)$

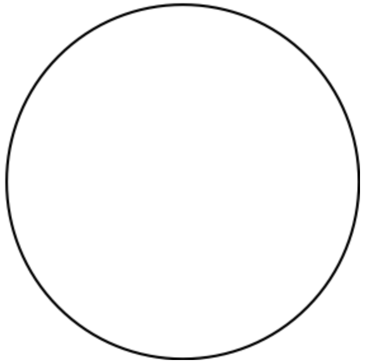
    qsa $= 0$
    repeat w **times**
        ns $\sim P(\cdot \mid \mathsf{s}, \mathsf{a})$ // Simulator access only
        $\mathsf{v}_{ns} = \mathrm{V}(\mathsf{s}, \mathsf{t}, \mathcal{S}, \mathcal{A}, P, R, H, w)$
        qsa $=$ qsa $+ \frac{1}{w}(R(\mathsf{s}, \mathsf{a}, \mathsf{ns}) + \gamma \mathsf{v}_{ns})$
    return qsa

$\mathrm{V}(\mathsf{s}, \mathsf{t}, \mathcal{S}, \mathcal{A}, P, R, H, w)$

if $t = H$
    return 0
return $\max_{\mathsf{a}} \mathrm{Q}(\mathsf{s}, \mathsf{a}, \mathsf{t}, \mathcal{S}, \mathcal{A}, P, R, H, w)$

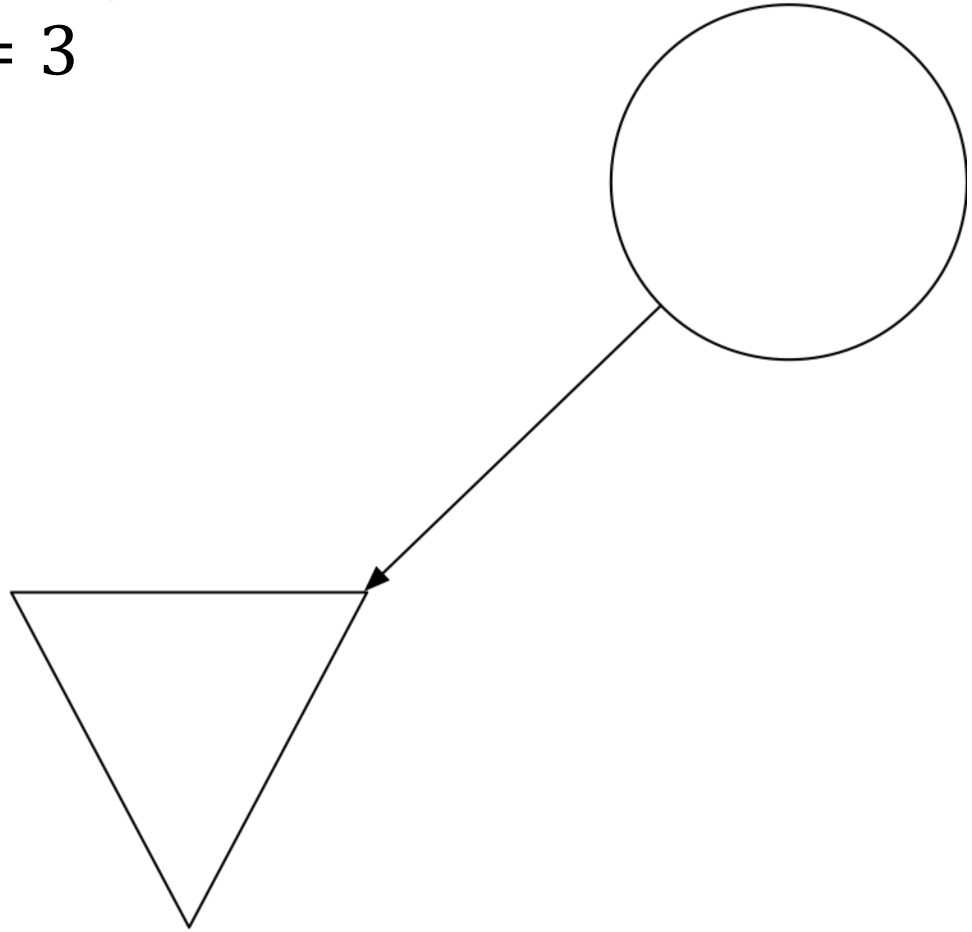
Sparse sampling

$H = 2, w = 3$



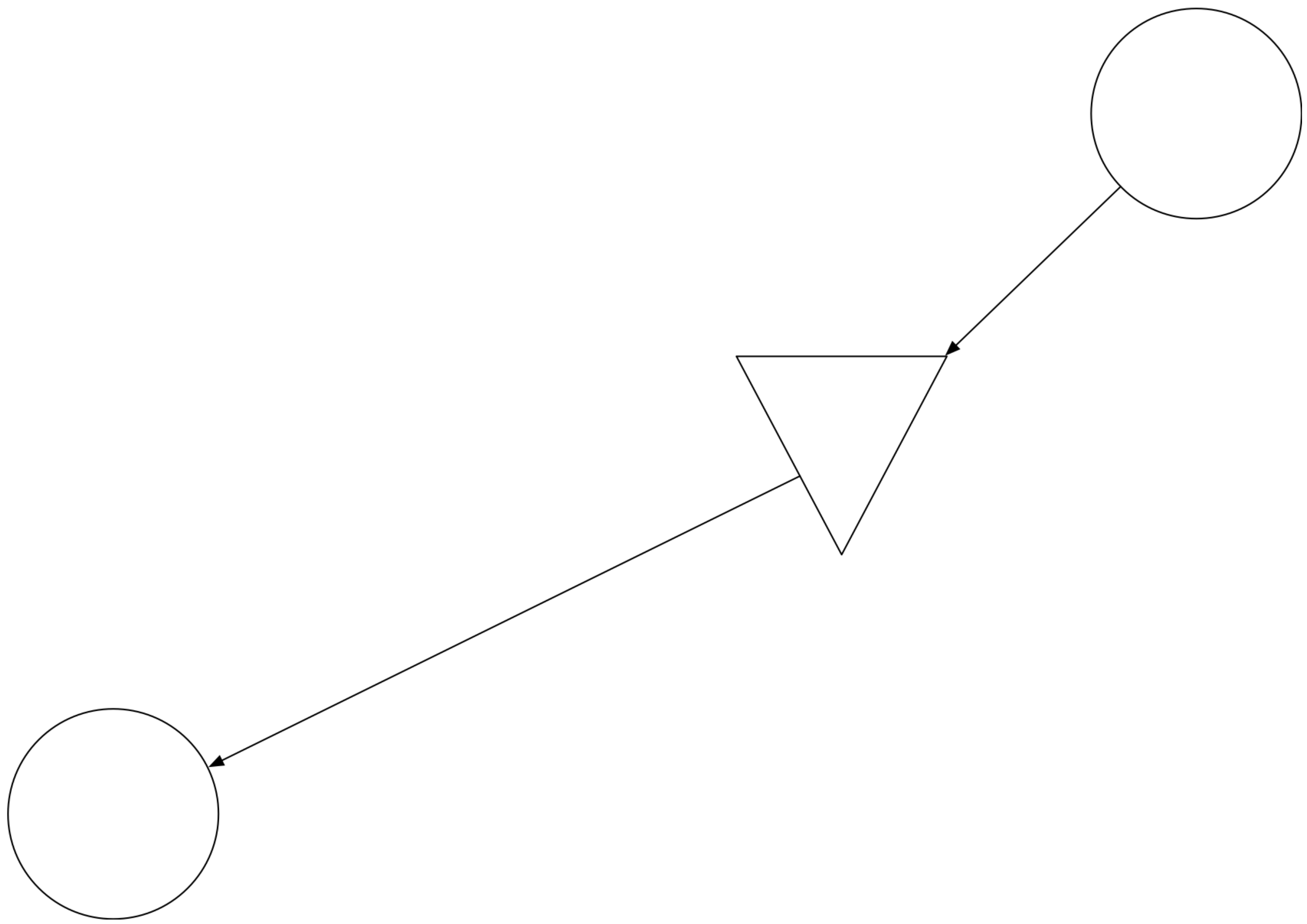
Sparse sampling

$H = 2, w = 3$



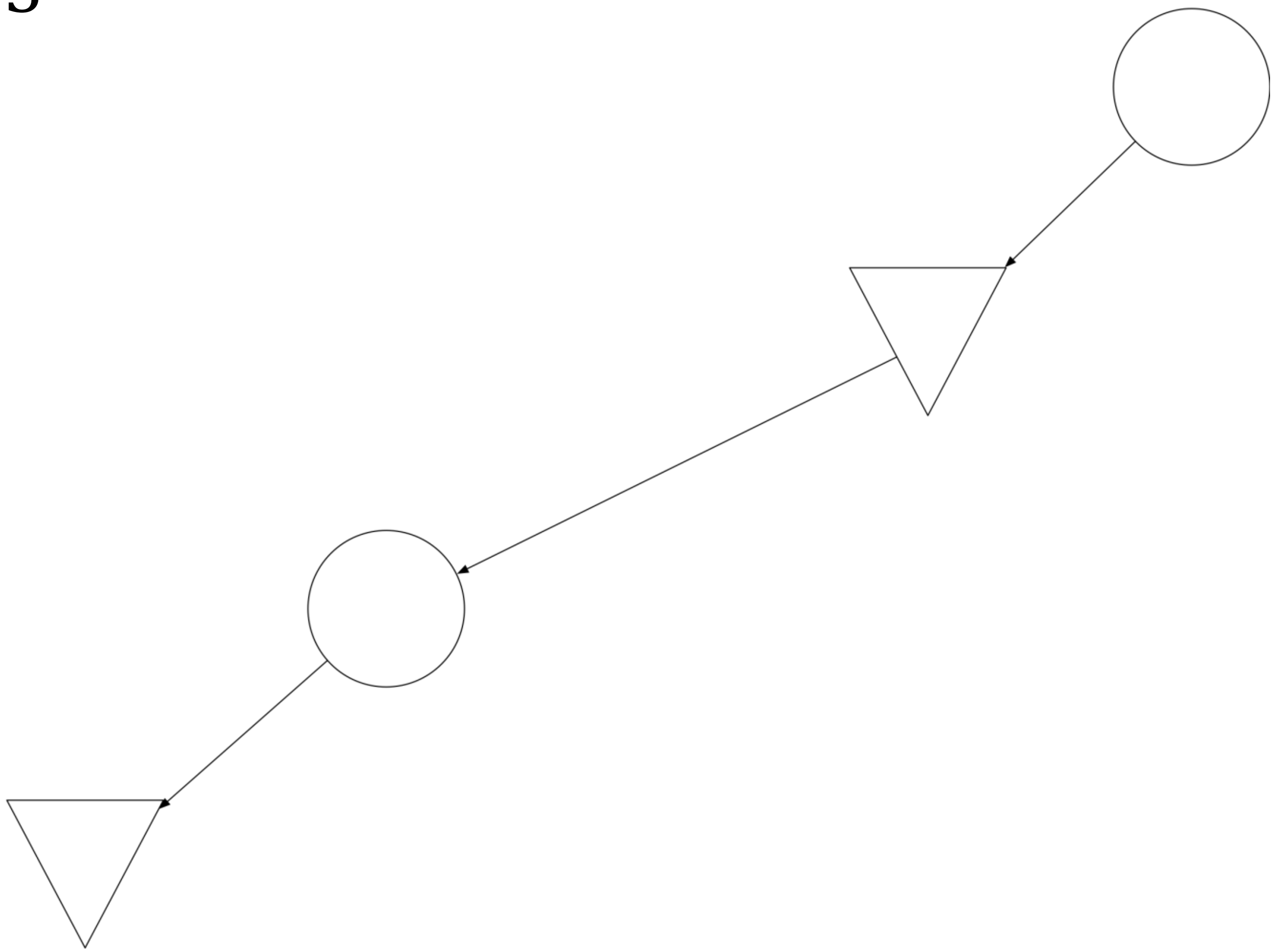
Sparse sampling

$H = 2, w = 3$

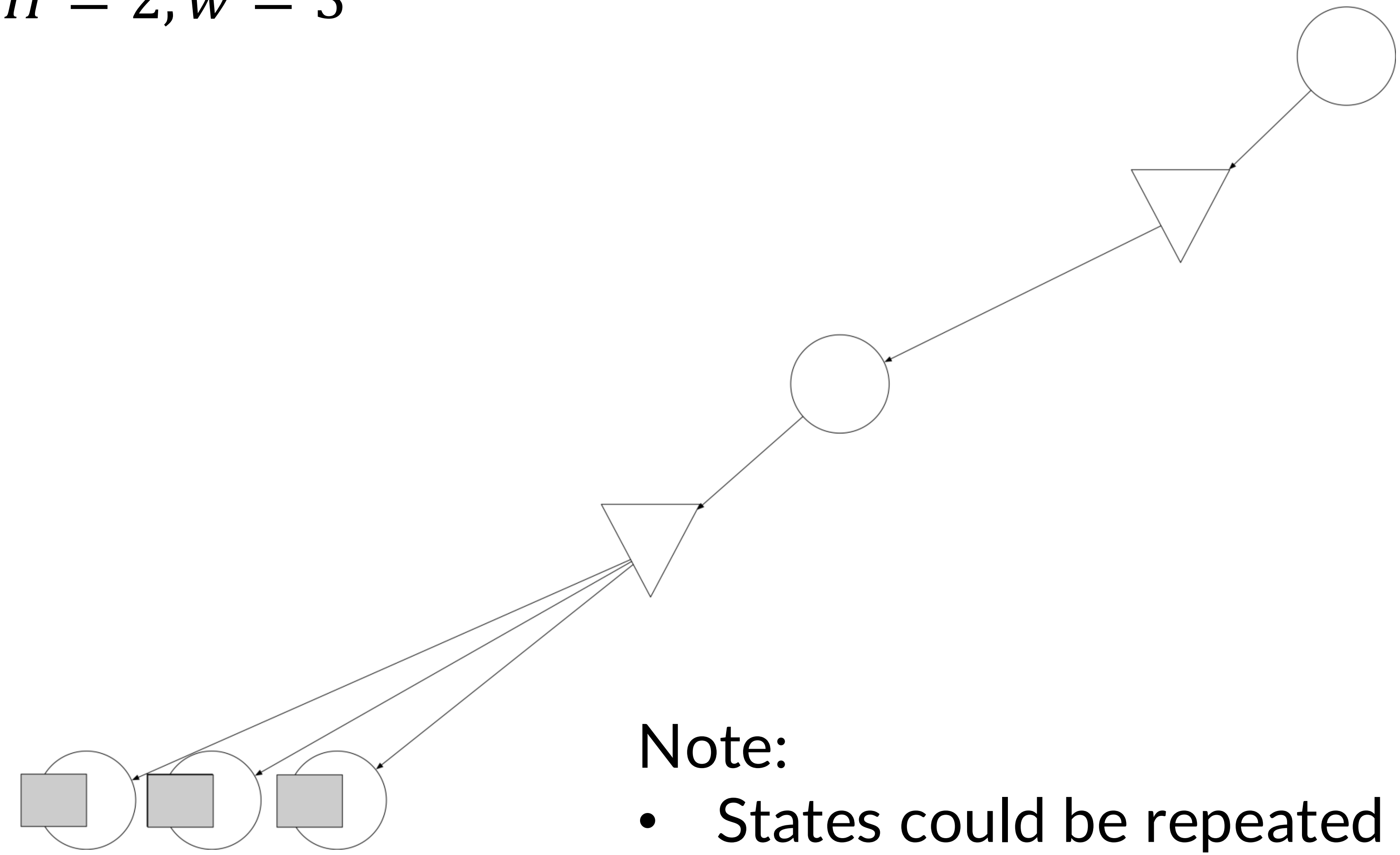


Sparse sampling

$$H = 2, w = 3$$



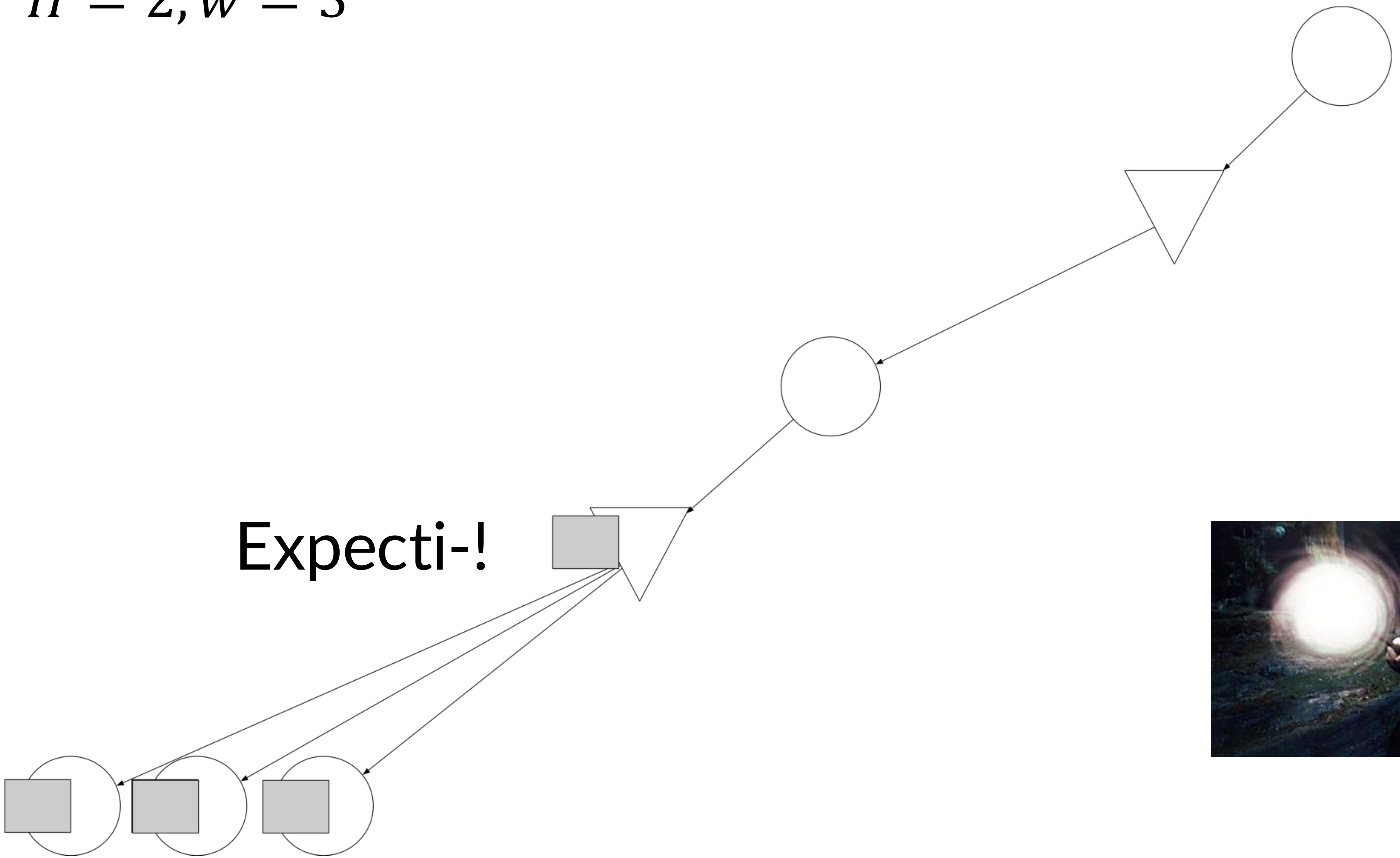
Sparse sampling
$H = 2, w = 3$



Note:
- States could be repeated
- Actual # successors could be >> 3

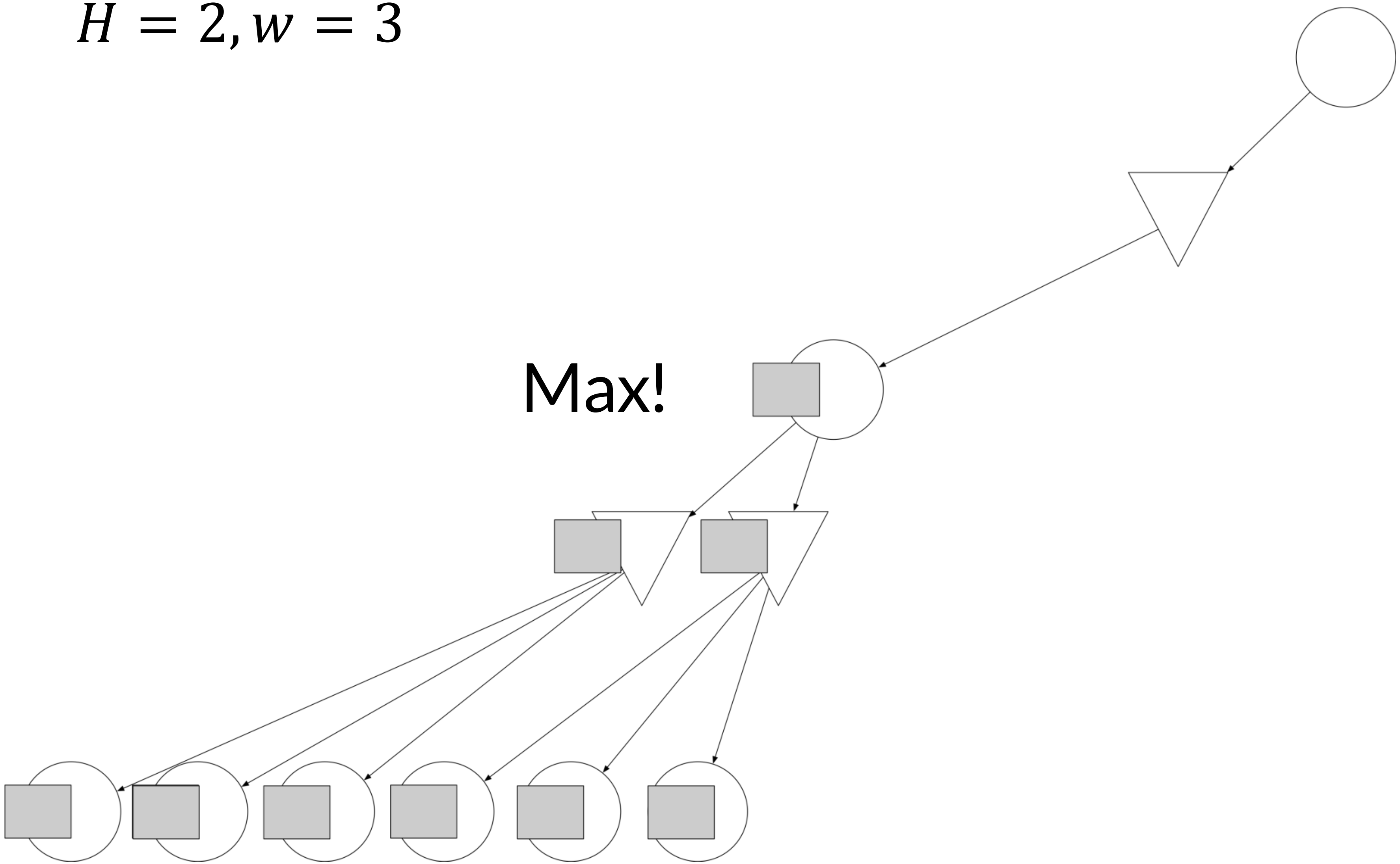
Sparse sampling

$H = 2, w = 3$



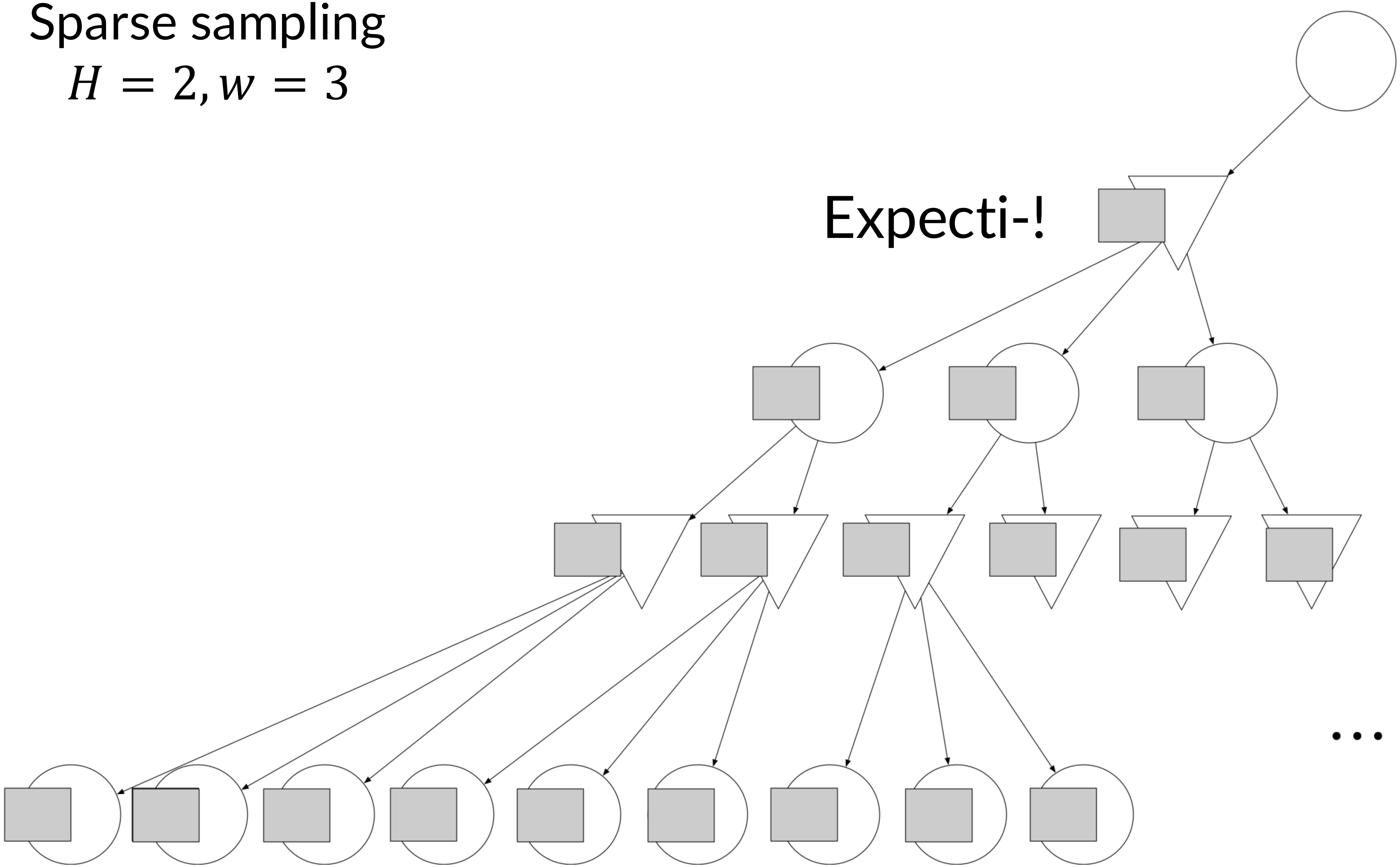
Expecti-!

Sparse sampling
$H = 2, w = 3$

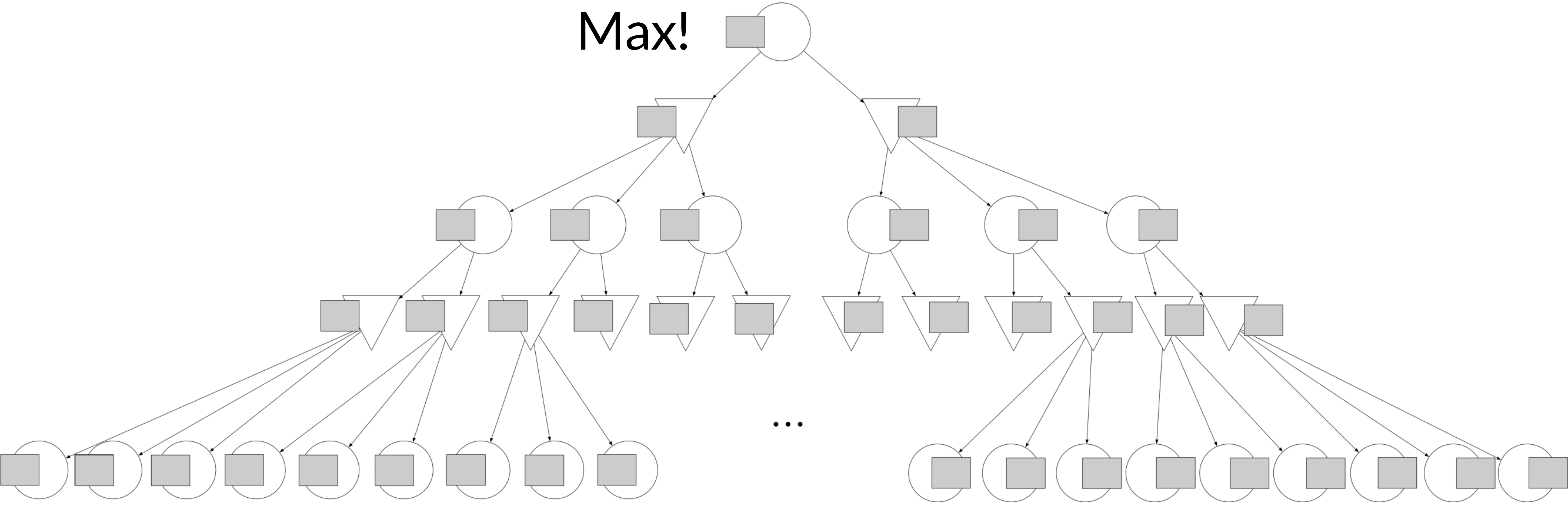


Sparse sampling

$H = 2, w = 3$



Sparse sampling
$H = 2, w = 3$



Limitations of Sparse Sampling

- Recall limitation of expectimax: exhaustive AODAG building
- Sparse sampling is similarly exhaustive
  - It does not use reward info at all in building the AODAG

Limitations of Sparse Sampling

- Recall limitation of expectimax: exhaustive AODAG building
- Sparse sampling is similarly exhaustive
  - It does not use reward info at all in building the AODAG
- RTDP was better: it built out using value estimates
- But RTDP still performed exhaustive Bellman backups

Limitations of Sparse Sampling

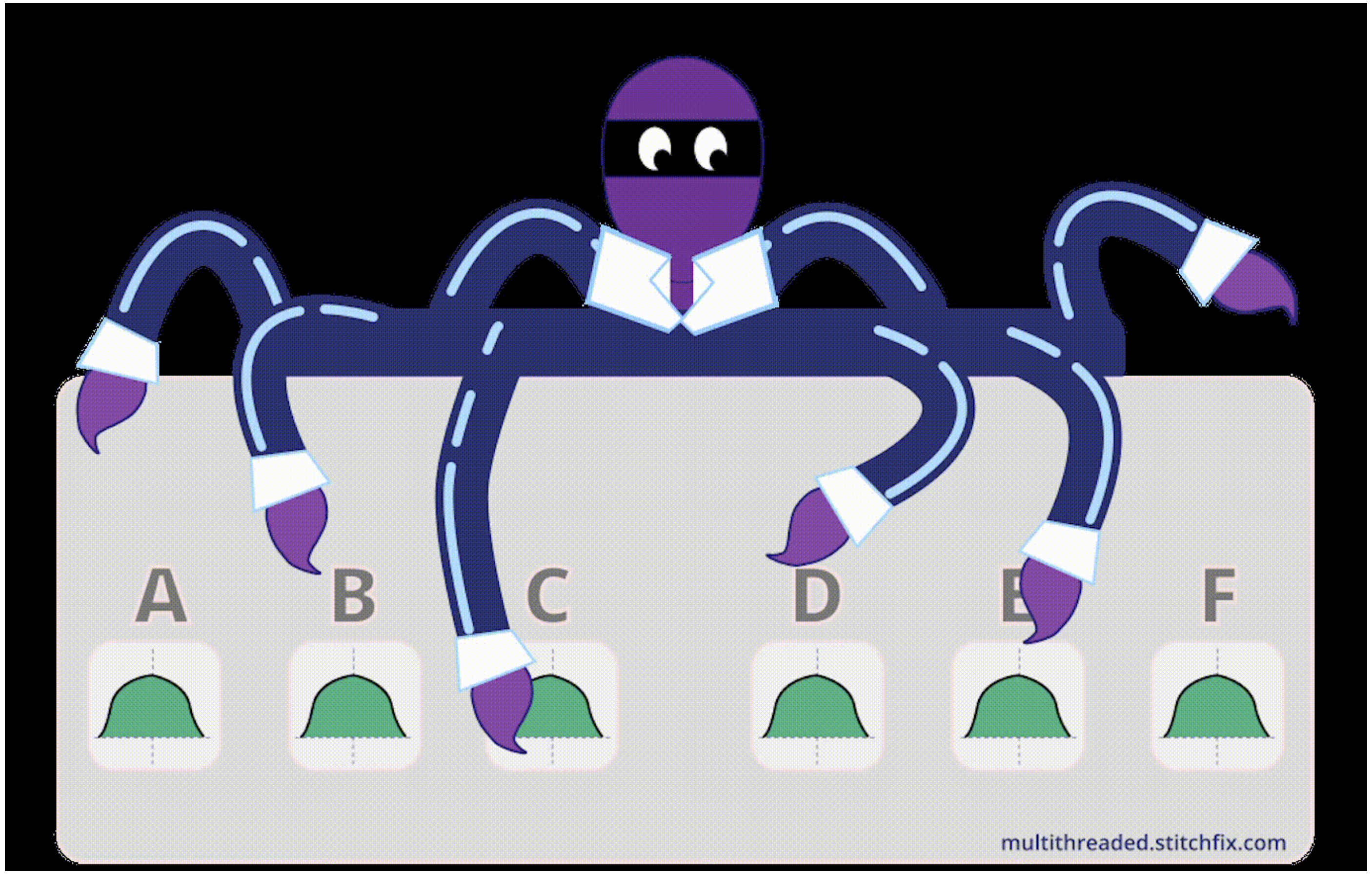
- Recall limitation of expectimax: exhaustive AODAG building
- Sparse sampling is similarly exhaustive
  - It does not use reward info at all in building the AODAG
- RTDP was better: it built out using value estimates
- But RTDP still performed exhaustive Bellman backups
- Moreover, RTDP may be a little “too greedy”
  - Always expands AODAG according to current best estimate
  - Does not *explore* parts of AODAG where estimates are uncertain

Limitations of Sparse Sampling

- Recall limitation of expectimax: exhaustive AODAG building
- Sparse sampling is similarly exhaustive
  - It does not use reward info at all in building the AODAG
- RTDP was better: it built out using value estimates
- But RTDP still performed exhaustive Bellman backups
- Moreover, RTDP may be a little "too greedy"
  - Always expands AODAG according to current best estimate
  - Does not *explore* parts of AODAG where estimates are uncertain

Let's study this in special case: $H = 1$.

Multi-armed Bandits (MAB)



https://multithreaded.stitchfix.com/blog/2020/08/05/bandits/

Multi-armed Bandits (MAB)

Consider finite horizon MDP, $H = 1$. Simulator access only.

And just one fixed initial state.

Multi-armed Bandits (MAB)

Consider finite horizon MDP, $H = 1$. Simulator access only.

Multi-armed Bandits (MAB): Repeat M times:

1. Select $a \in \mathcal{A}$

Multi-armed Bandits (MAB)

Consider finite horizon MDP, $H = 1$. Simulator access only.

Multi-armed Bandits (MAB): Repeat M times:

1. Select $a \in \mathcal{A}$
2. Receive sample $s' \in P(\cdot |\, s, a)$

Multi-armed Bandits (MAB)

Consider finite horizon MDP, $H = 1$. Simulator access only.

Multi-armed Bandits (MAB): Repeat M times:

1. Select $a \in \mathcal{A}$
2. Receive sample $s' \in P(\cdot | s, a)$
3. Observe reward $R(s, a, s')$

Multi-armed Bandits (MAB)

Consider finite horizon MDP, $H = 1$. Simulator access only.

Multi-armed Bandits (MAB): Repeat M times:

1. Select $a \in \mathcal{A}$ How? This is the challenge.
2. Receive sample $s' \in P(\cdot | s, a)$
3. Observe reward $R(s, a, s')$

Multi-armed Bandits (MAB)

Consider finite horizon MDP, $H = 1$. Simulator access only.

Multi-armed Bandits (MAB): Repeat M times:

1. Select $a \in \mathcal{A}$

How? This is the challenge.

2. Receive sample $s' \in P(\cdot \mid s, a)$
3. Observe reward $R(s, a, s')$

What's the objective?

MAB: A Tale of Two Settings

Objective: minimize *regret*

MAB: A Tale of Two Settings

Objective: minimize *regret*

Simple Regret

- After M samples, take one final action and receive r_{M+1}.
- Simple regret: $r^*_{M+1} - r_{M+1}$ where r^*_{M+1} is best possible under clairvoyant policy.

MAB: A Tale of Two Settings

Objective: minimize *regret*

Simple Regret

- After M samples, take one final action and receive r_{M+1}.
- Simple regret: $r^*_{M+1} - r_{M+1}$ where r^*_{M+1} is best possible under clairvoyant policy.
- Don't care about $r_1, \ldots, r_M$.
- Just want informative data.
- A.k.a. *selection problem*.

MAB: A Tale of Two Settings

Objective: minimize *regret*

Simple Regret

- After M samples, take one final action and receive r_{M+1}.
- Simple regret: $r^*_{M+1} - r_{M+1}$ where r^*_{M+1} is best possible under clairvoyant policy.
- Don't care about $r_1, \ldots, r_M$.
- Just want informative data.
- A.k.a. *selection problem*.

Cumulative Regret

- Cumulative regret: $r^*_1 + \cdots + r^*_M - (r_1 + \cdots + r_M)$.

MAB: A Tale of Two Settings

Objective: minimize *regret*

Simple Regret

- After M samples, take one final action and receive r_{M+1}.
- Simple regret: $r^*_{M+1} - r_{M+1}$ where r^*_{M+1} is best possible under clairvoyant policy.
- Don't care about $r_1, \ldots, r_M$.
- Just want informative data.
- A.k.a. *selection problem*.

Cumulative Regret

- Cumulative regret: $r^*_1 + \cdots + r^*_M - (r_1 + \cdots + r_M)$.
- *Exploration-exploitation*: at each step, should we select action believed to be best (exploit) or try one we're uncertain about (explore)?

MAB: A Tale of Two Settings

Objective: minimize *regret*

Simple Regret

- After M samples, take one final action and receive r_{M+1}.
- Simple regret: $r^*_{M+1} - r_{M+1}$ where r^*_{M+1} is best possible under clairvoyant policy.
- Don't care about $r_1, \ldots, r_M$.
- Just want informative data.
- A.k.a. *selection problem*.

Cumulative Regret

- Cumulative regret: $r_1^* + \cdots + r_M^* \; - (r_1 + \cdots + r_M)$.
- *Exploration-exploitation*: at each step, should we select action believed to be best (exploit) or try one we're uncertain about (explore)?

When would each make more sense?

Strategies for MAB

- Most strategies maintain sample estimate of Q function:

$$\hat{Q}(s, a) = \frac{1}{|\mathcal{I}_a|} \sum_{i \in \mathcal{I}_a} r_i$$

s not important here, but will be later

where $\mathcal{I}_a$ is the set of step indices where action a was selected.

Strategies for MAB

- Most strategies maintain sample estimate of Q function:

$$\hat{Q}(s,a) = \frac{1}{|\mathcal{J}_a|} \sum_{i \in \mathcal{J}_a} r_i$$

s not important here, but will be later

where $\mathcal{J}_a$ is the set of step indices where action a was selected.

- Notation: $N(s,a) = |\mathcal{J}_a|$.

Number of times we have tried a

Strategies for MAB

- Most strategies maintain sample estimate of Q function:

$$\hat{Q}(s, a) = \frac{1}{|\mathcal{J}_a|} \sum_{i \in \mathcal{J}_a} r_i$$

s not important here, but will be later

where $\mathcal{J}_a$ is the set of step indices where action a was selected.

- Notation: $N(s, a) = |\mathcal{J}_a|$.

Number of times we have tried a

Why might "always select $\mathrm{argmax}_\mathrm{a}\ \hat{Q}(s, a)$" be a suboptimal strategy?

Strategies for MAB: ϵ-greedy

Epsilon-greedy strategy

- With probability ϵ, select random action (explore)
- Otherwise, select $\mathrm{argmax_a}\ \hat{Q}(s, a)$ (exploit)

If there's an action that has never been tried ($N(s\ , a) = 0$), select it.

Strategies for MAB: UCB

Upper confidence bounds (UCB)

Main idea: *optimism in the face of uncertainty*.

- New restaurant in town! I don’t know if it’s good, but optimistically, it might be fantastic! Let’s eat.
- New course offering! I don’t know if it’s good, but optimistically, it might be. Let’s take it!

Why is optimism in the face of uncertainty a good principle?

- If your optimistic predictions are correct, you'll be thrilled!
- If they're not, you will quickly discover that you were wrong from the new data.
- Contrast with pessimism.

Being Optimistic with Confidence Bounds

- Suppose I believe that with 95% probability, $\hat{Q}(s, a_1)$ is between -1.25 and 4.75.

- Optimism in the face of uncertainty says: it's *plausibly possible* that $\hat{Q}(s, a_1)$ = 4.75, so I'm going to assume that it is.

Being Optimistic with Confidence Bounds

- Suppose I believe that with 95% probability, $\hat{Q}(s, a_1)$ is between -1.25 and 4.75.

- Optimism in the face of uncertainty says: it's *plausibly possible* that $\hat{Q}(s, a_1)$ = 4.75, so I'm going to assume that it is.

- I also think that with 95% probability, $-3.0 \ \leq \hat{Q}(s, a_2) \leq 5.0$.

- Optimistically, $\hat{Q}(s, a_2) > \hat{Q}(s, a_1)$. So, I'll choose a_2!

Recipe for Deriving UCB Algorithms

Recipe for Deriving UCB Algorithms

- For each action $a \in \mathcal{A}$, define random variables $X_a^1, \dots X_a^n$ where X_a^i represents the reward for the i^{th} try of action a.

- Note that these X_a^i are i.i.d. with distribution $R(s, a, S')$, where $S' \sim P(s' \mid s, a)$, which has mean $Q(s, a)$.

Recipe for Deriving UCB Algorithms

- For each action $a \in \mathcal{A}$, define random variables $X_a^1, \dots X_a^n$ where X_a^i represents the reward for the i^{th} try of action a.
- Note that these X_a^i are i.i.d. with distribution $R(s, a, S')$, where $S' \sim P(s' \mid s, a)$, which has mean $Q(s, a)$.
- Let $\hat{X}_a^n = \frac{1}{n} \sum_{i=1}^n X_a^i$. Represents $\hat{Q}(s, a)$ after n tries of a.

Recipe for Deriving UCB Algorithms

- For each action $a \in \mathcal{A}$, define random variables $X_a^1, \dots X_a^n$ where X_a^i represents the reward for the i^{th} try of action a.

- Note that these X_a^i are i.i.d. with distribution $R(s, a, S')$, where $S' \sim P(s' \mid s, a)$, which has mean $Q(s, a)$.

- Let $\hat{X}_a^n = \frac{1}{n} \sum_{i=1}^{n} X_a^i$. Represents $\hat{Q}(s, a)$ after n tries of a.

- Make some assumptions about the distribution (e.g., it is subgaussian) and use some concentration bounds (e.g, Chebyshev) to derive an inequality like...

Recipe for Deriving UCB Algorithms

$$P\left(Q(s,a) \geq \hat{X}_a^n + \sqrt{\frac{2\log\left(\frac{1}{\delta}\right)}{n}}\right) \leq \delta \qquad \text{For any } \delta \in (0,1)$$

Recipe for Deriving UCB Algorithms

$$P\left(Q(s,a) \geq \hat{X}_a^n + \sqrt{\frac{2\log\left(\frac{1}{\delta}\right)}{n}}\right) \leq \delta \qquad \text{For any } \delta \in (0,1)$$

After n tries of action a, I can be sure, with $(1-\delta)$ probability, that my estimate of the value of a is within a constant from the true value.

Recipe for Deriving UCB Algorithms

$$P\left(Q(s,a) \geq \hat{X}_a^n + \sqrt{\frac{2\log\left(\frac{1}{\delta}\right)}{n}}\right) \leq \delta \qquad \text{For any } \delta \in (0,1)$$

After n tries of action a, I can be sure, with $(1-\delta)$ probability, that my estimate of the value of a is within a constant from the true value.

Given a desired confidence level, like $(1-\delta)$ = 0.95, the most optimistic plausible estimate of the true value is $\hat{X}_a^n + constant$.

Recipe for Deriving UCB Algorithms

As n gets larger, or as $1 - \delta$ gets larger, bound gets tighter.

$$P\left(Q(s,a) \geq \hat{X}_a^n + \sqrt{\frac{2\log\left(\frac{1}{\delta}\right)}{n}}\right) \leq \delta \quad \text{For any } \delta \in (0,1)$$

After n tries of action a, I can be sure, with $(1-\delta)$ probability, that my estimate of the value of a is within a constant from the true value.

Given a desired confidence level, like $(1-\delta)$ = 0.95, the most optimistic plausible estimate of the true value is $\hat{X}_a^n + \mathit{constant}$.

Strategies for MAB: UCB

Upper confidence bounds (UCB)

Note resemblance to concentration bounds!

Idea: construct confidence intervals for $\hat{Q}$, then be *optimistic in the face of uncertainty.*

$$\text{At step } m\text{, select: } \operatorname{argmax}_{\mathrm{a}} \left[\hat{Q}(s,a) + \frac{\phi(m)}{\sqrt{N(s,a)}}\right]$$

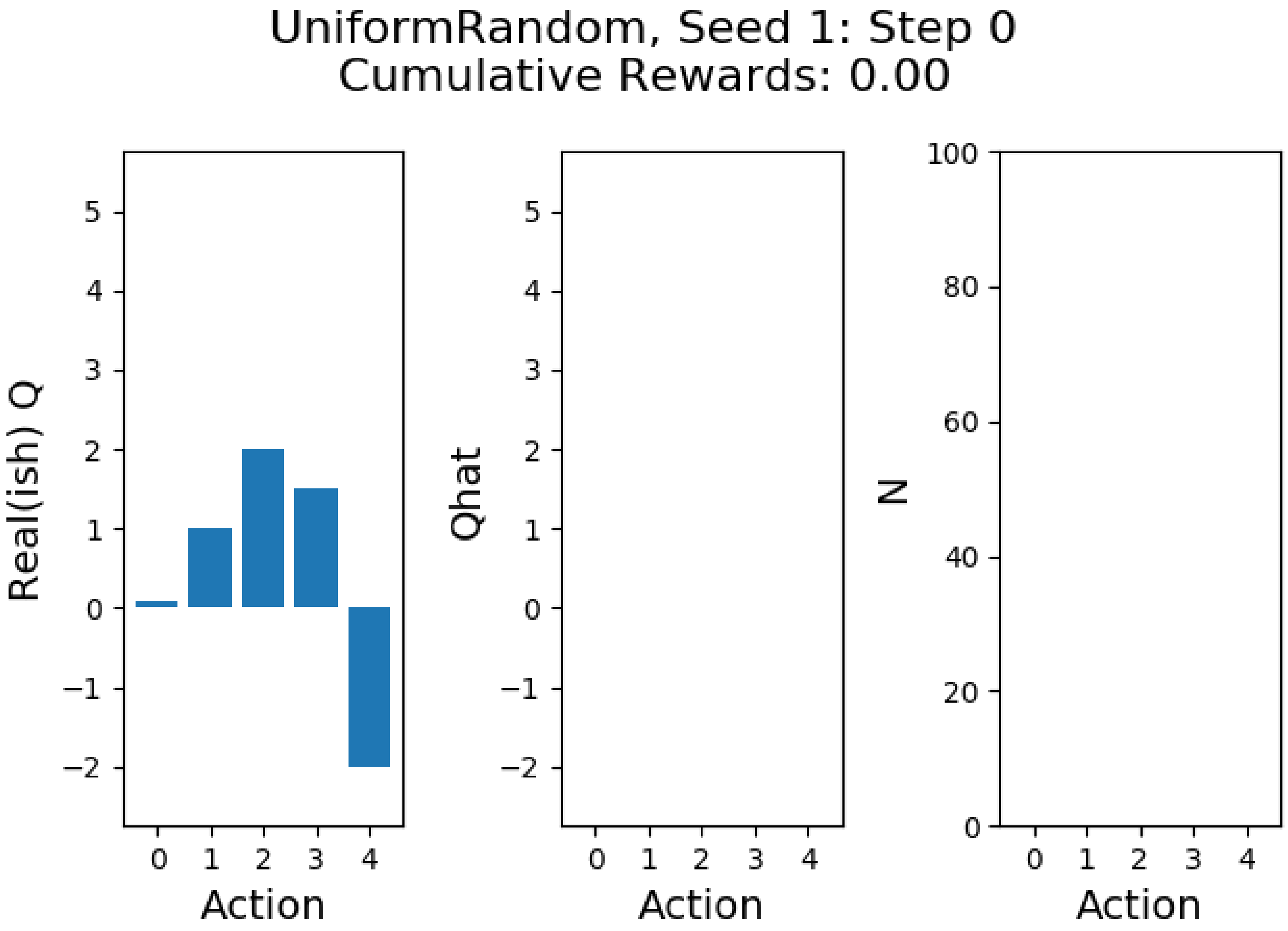
where ϕ can be several functions; often $\phi(m) = c\sqrt{\log(m)}$ for a hyperparameter c.

Intuition: as number of tries increases, shift from exploration to exploitation.

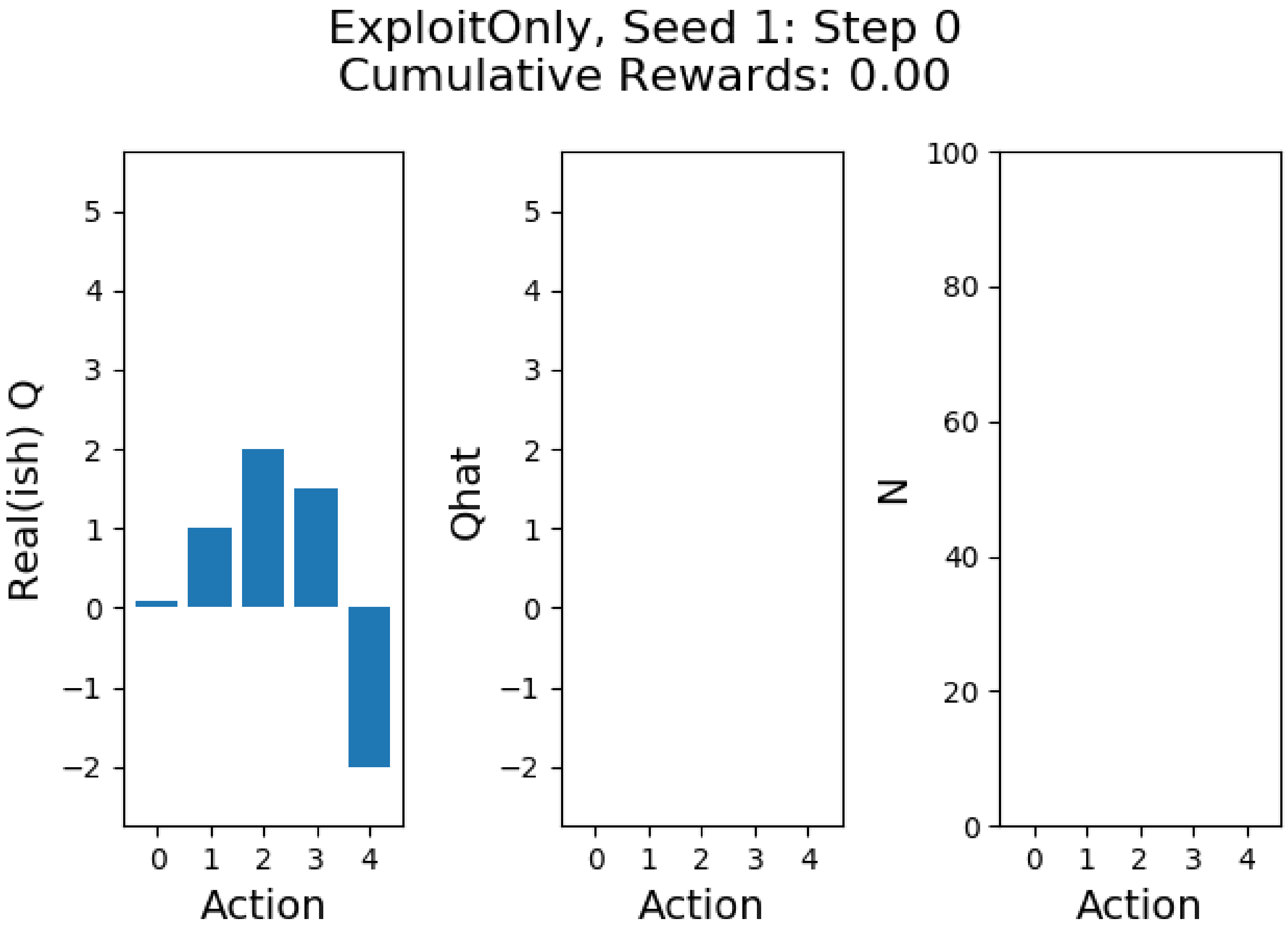
Strategies for MAB: UCB

Upper confidence bounds (UCB)

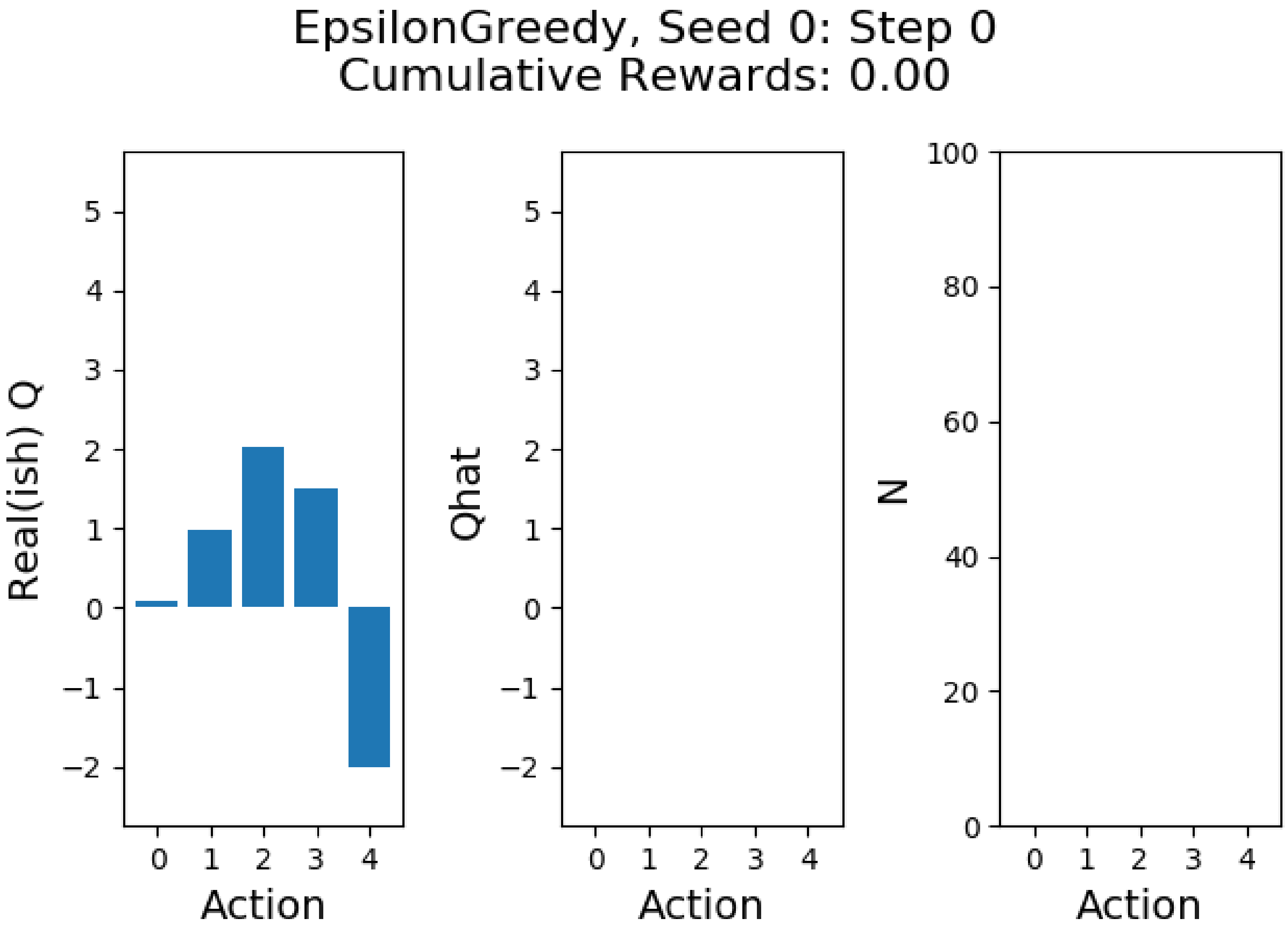
- UCB attains optimal cumulative regret (Lai & Robbins 1985)
- It does not attain optimal simple regret (Bubeck et al. 2010)
- But it's widely used in planning contexts nonetheless, and works well in practice

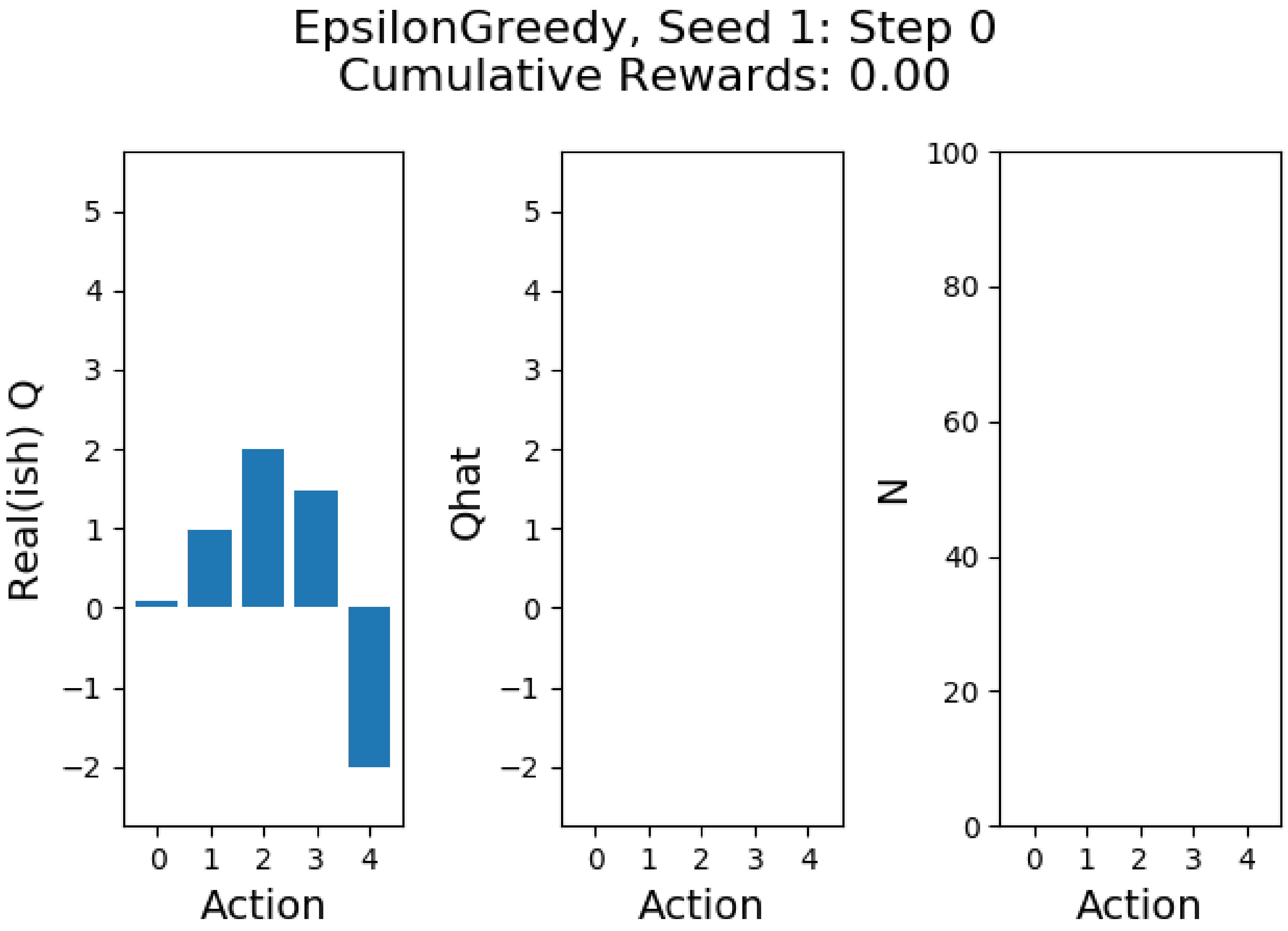


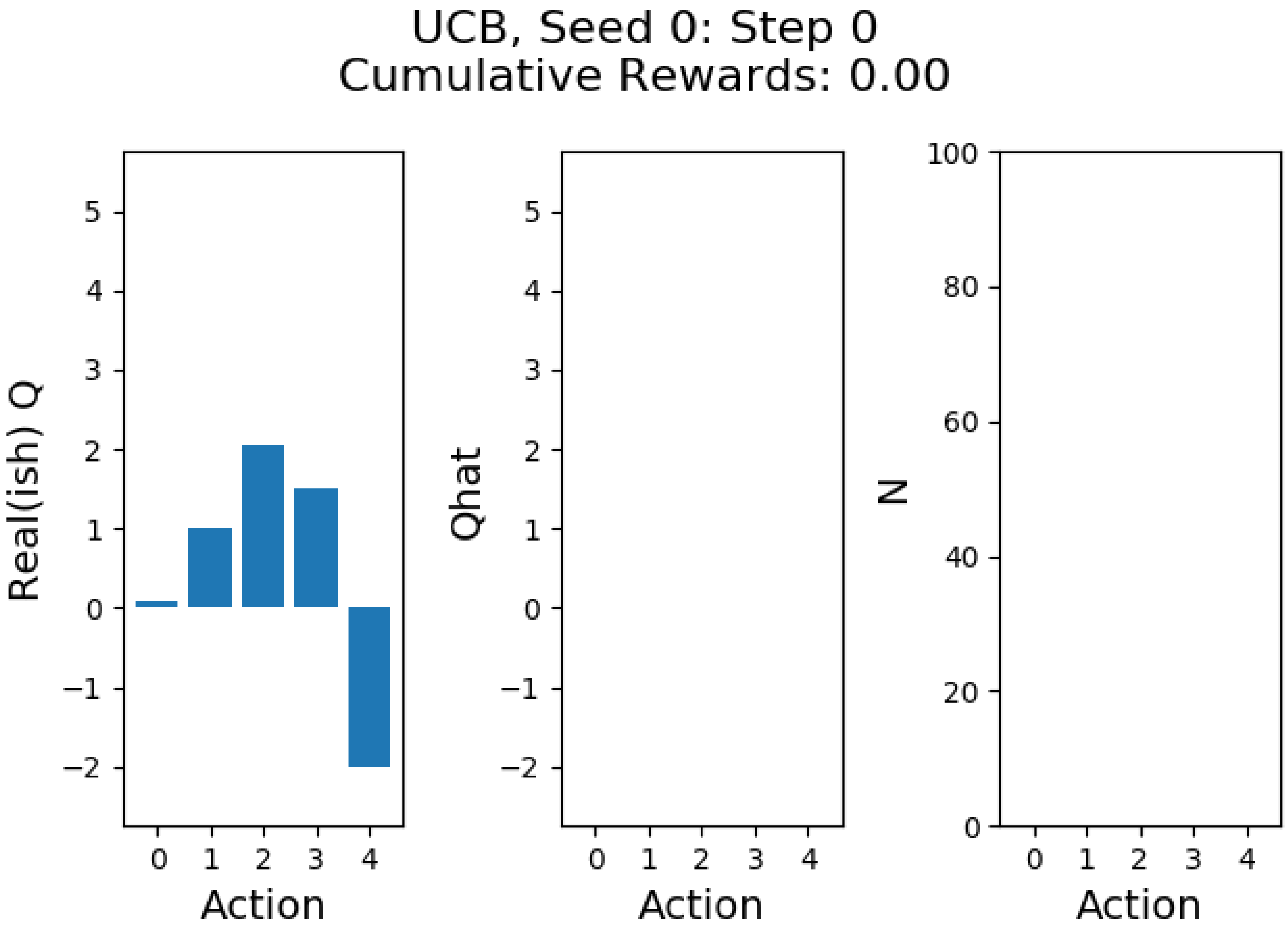
UniformRandom, Seed 1: Step 0
Cumulative Rewards: 0.00
Real(ish) Q
Qhat
N
Action
Action
Action
0
1
2
3
4
5
−1
−2
20
40
60
80
100

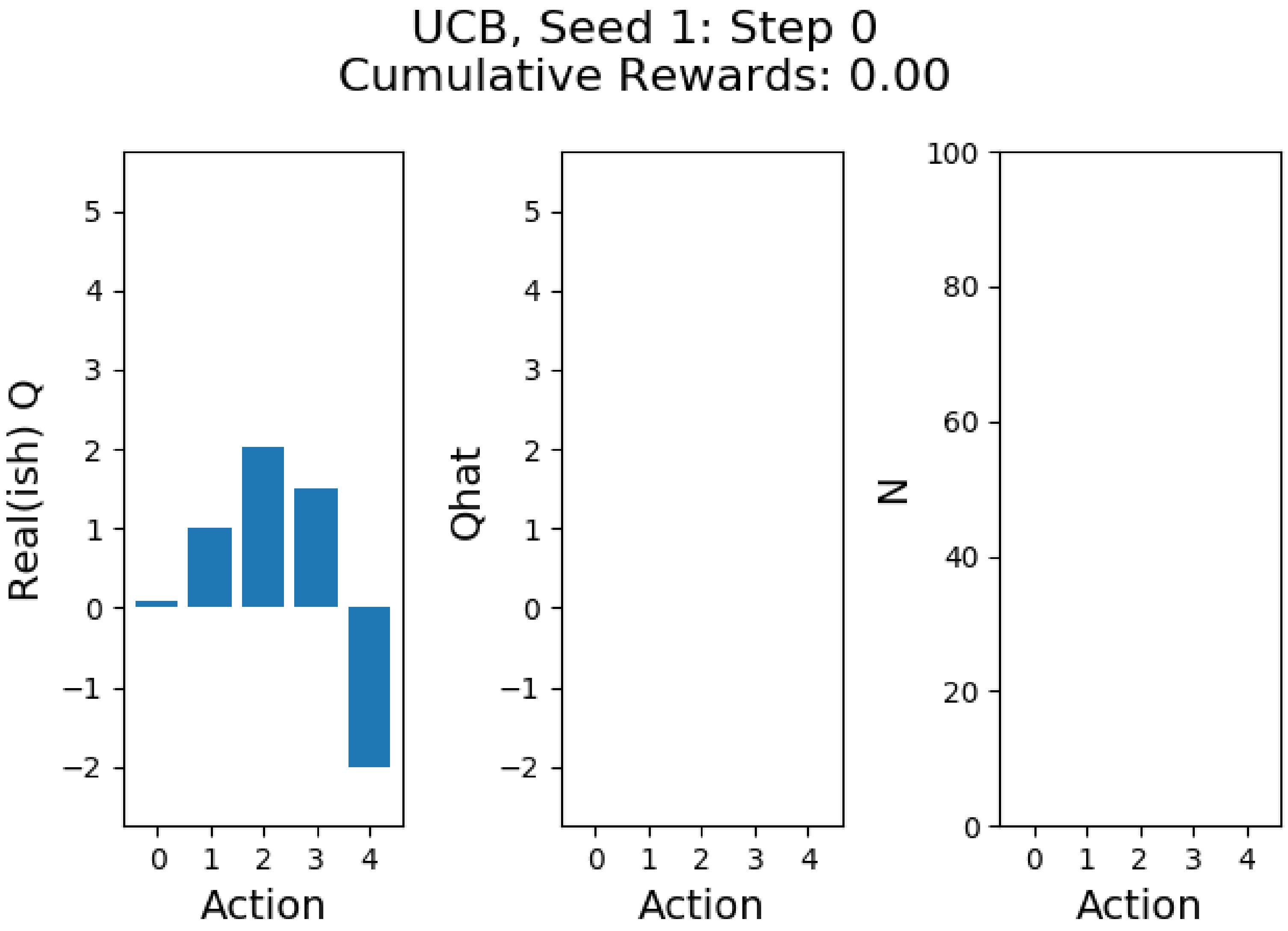


ExploitOnly, Seed 1: Step 0
Cumulative Rewards: 0.00
Real(ish) Q
Qhat
N
Action
Action
Action


EpsilonGreedy, Seed 0: Step 0
Cumulative Rewards: 0.00
Real(ish) Q
Qhat
N
Action
Action
Action
5
4
3
2
1
0
−1
−2
0 1 2 3 4
100
80
60
40
20
0


EpsilonGreedy, Seed 1: Step 0
Cumulative Rewards: 0.00
Real(ish) Q
Qhat
N
Action
Action
Action
5
4
3
2
1
0
−1
−2
100
80
60
40
20
0
0 1 2 3 4


UCB, Seed 0: Step 0
Cumulative Rewards: 0.00
Real(ish) Q
Qhat
N
Action
Action
Action
5
4
3
2
1
0
−1
−2
100
80
60
40
20
0
0 1 2 3 4


UCB, Seed 1: Step 0
Cumulative Rewards: 0.00
Real(ish) Q
Qhat
N
Action
Action
Action
5
4
3
2
1
0
−1
−2
100
80
60
40
20
0
0 1 2 3 4

	Uniform Random	Exploit Only	Epsilon Greedy	UCB
Mean **Cumulative** Reward	55.70	141.98	134.75	149.70
Std **Cumulative** Reward	22.94	44.09	37.66	36.23
Mean **Final** Reward	1.98	1.22	1.72	2.34
Std **Final** Reward	3.12	1.84	3.01	3.51

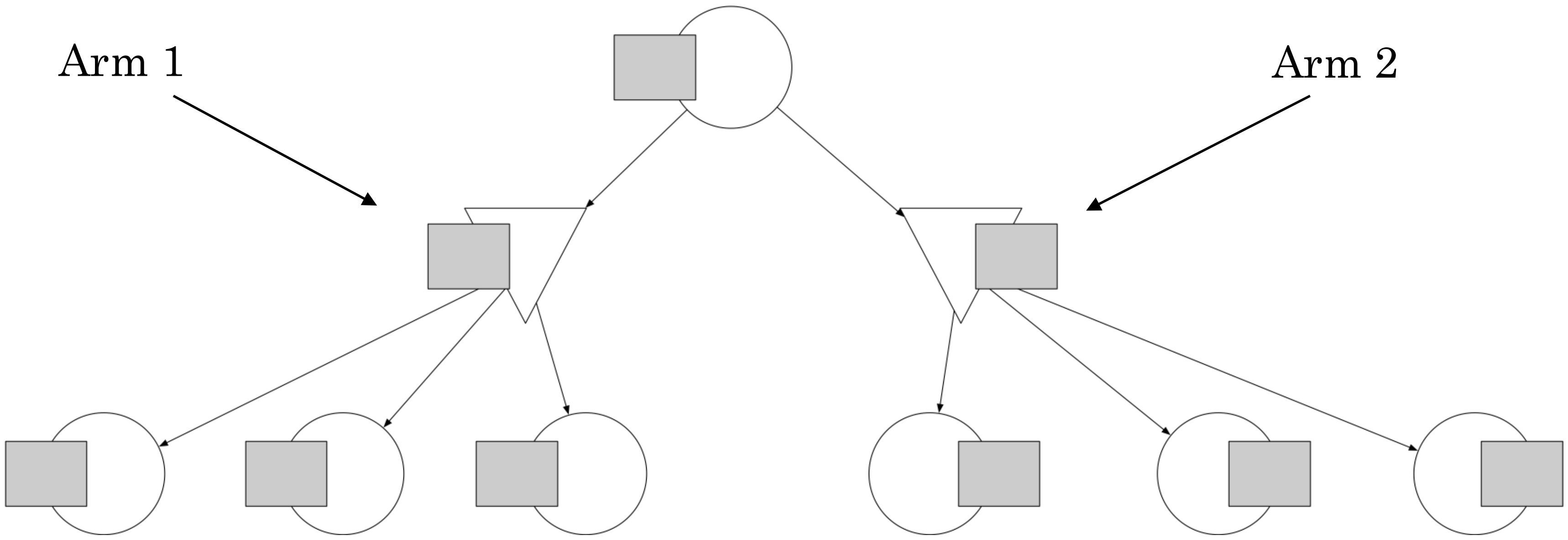
	Uniform Random	Exploit Only	Epsilon Greedy	UCB
Mean **Cumulative** Reward	55.70	141.98	134.75	149.70
Std **Cumulative** Reward	22.94	44.09	37.66	36.23
Mean **Final** Reward	1.98	1.22	1.72	2.34
Std **Final** Reward	3.12	1.84	3.01	3.51

Lots more on Bandits:

- "Bandit Algorithms." Lattimore & Szepesvari (2020). Free online.
- https://banditalgs.com/

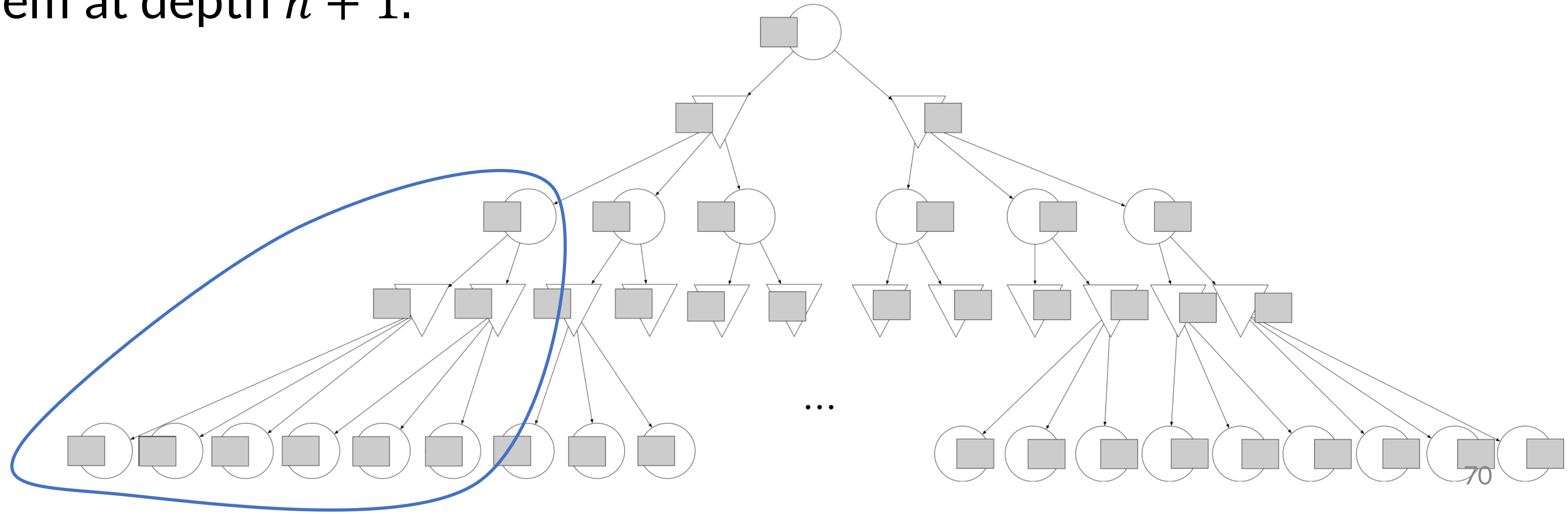
Recursive Bandits

- Sparse sampling with $H = 1 \approx$ "Uniform Random" for MAB.



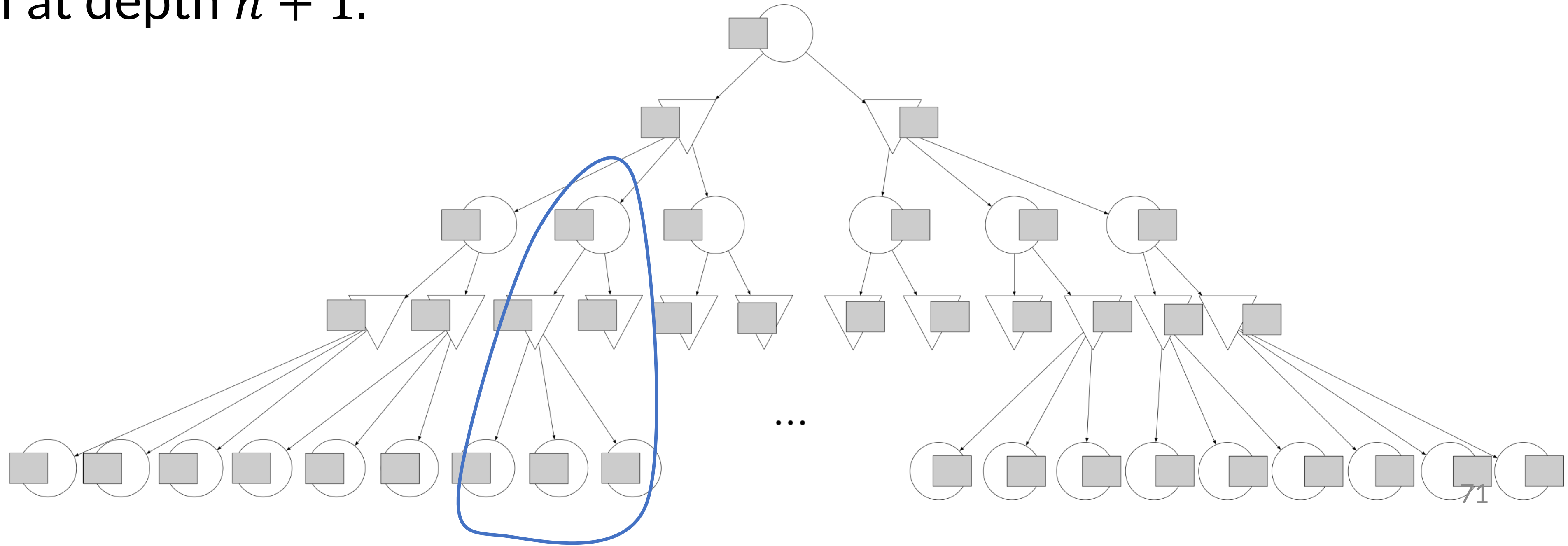
Recursive Bandits

- Sparse sampling with $H = 1 \approx$ “Uniform Random” for MAB.
- Sparse sampling with $H > 1 \approx$ naïve solution to *recursive bandit problem.*
  - To determine the “reward” for taking action at depth h, first solve MAB problem at depth $h + 1$.



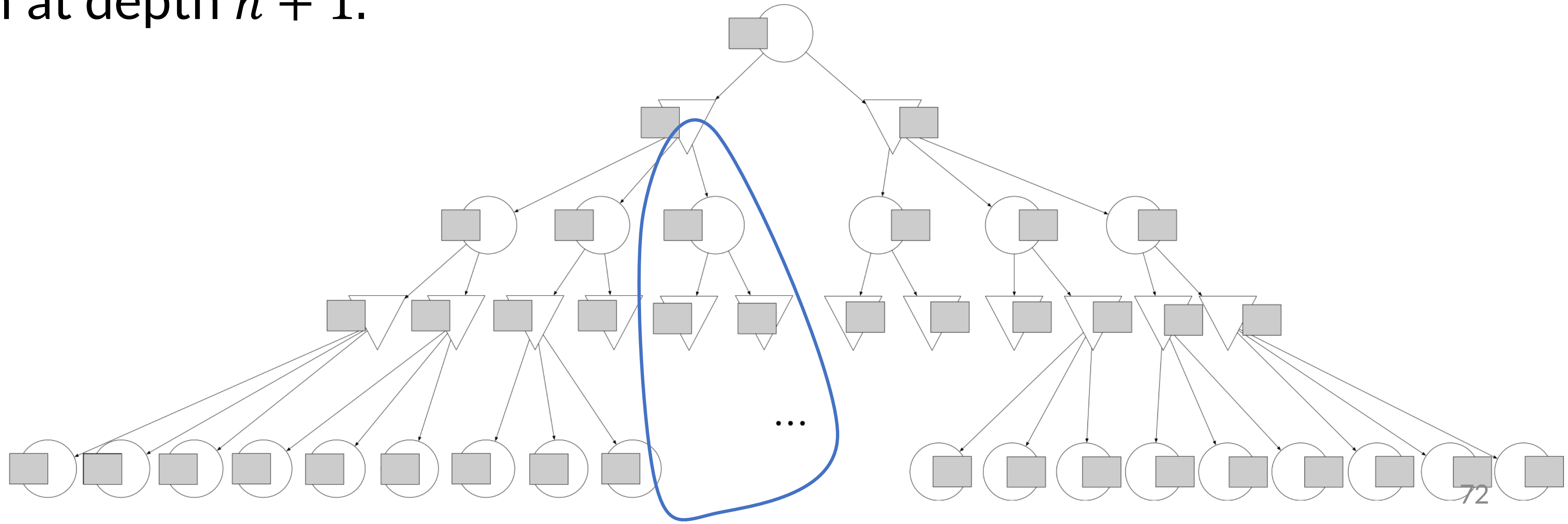
Recursive Bandits

- Sparse sampling with $H = 1 \approx$ "Uniform Random" for MAB.
- Sparse sampling with $H > 1 \approx$ naïve solution to *recursive bandit problem.*
  - To determine the "reward" for taking action at depth h, first solve MAB problem at depth $h + 1$.



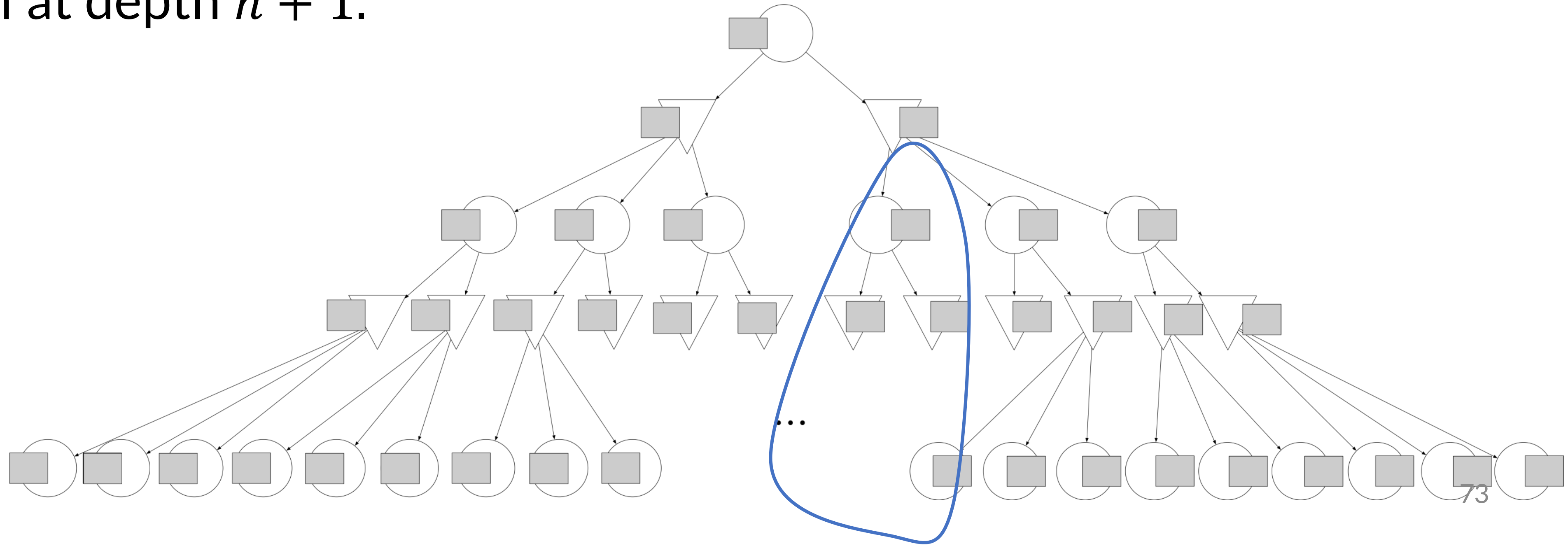
Recursive Bandits

- Sparse sampling with $H = 1 \approx$ "Uniform Random" for MAB.
- Sparse sampling with $H > 1 \approx$ naïve solution to *recursive bandit problem.*
  - To determine the "reward" for taking action at depth h, first solve MAB problem at depth $h + 1$.



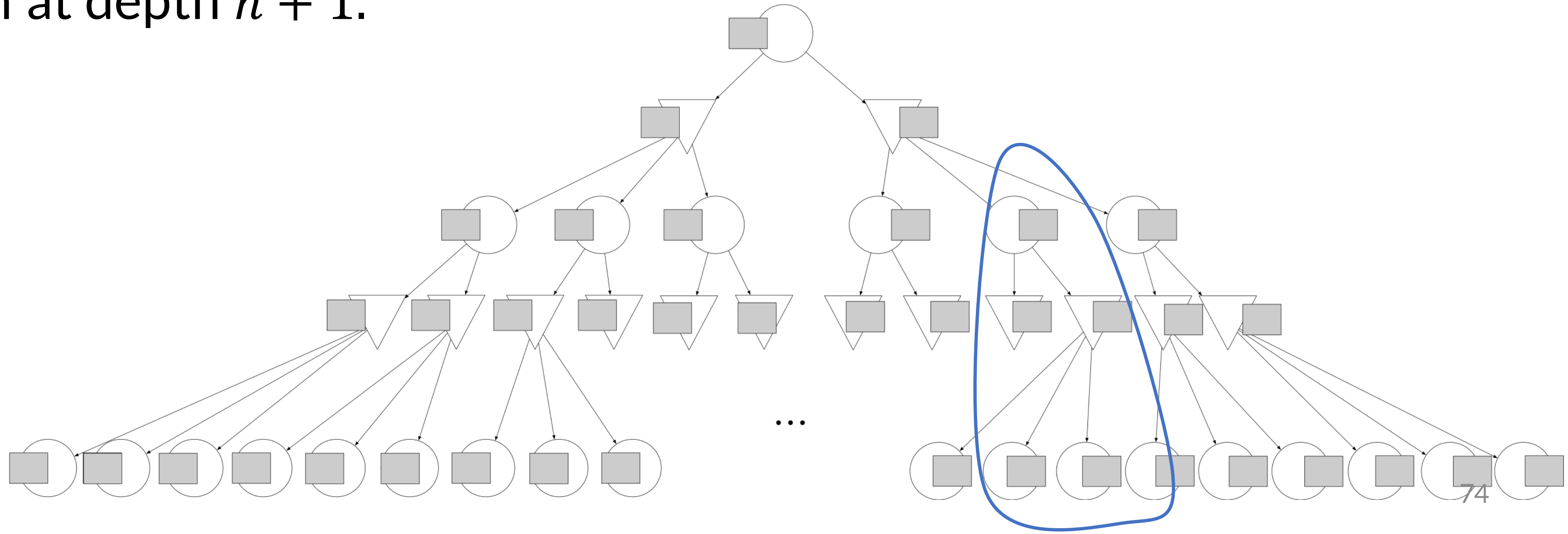
Recursive Bandits

- Sparse sampling with $H = 1 \approx$ “Uniform Random” for MAB.
- Sparse sampling with $H > 1 \approx$ naïve solution to *recursive bandit problem*.
    - To determine the “reward” for taking action at depth h, first solve MAB problem at depth $h + 1$.



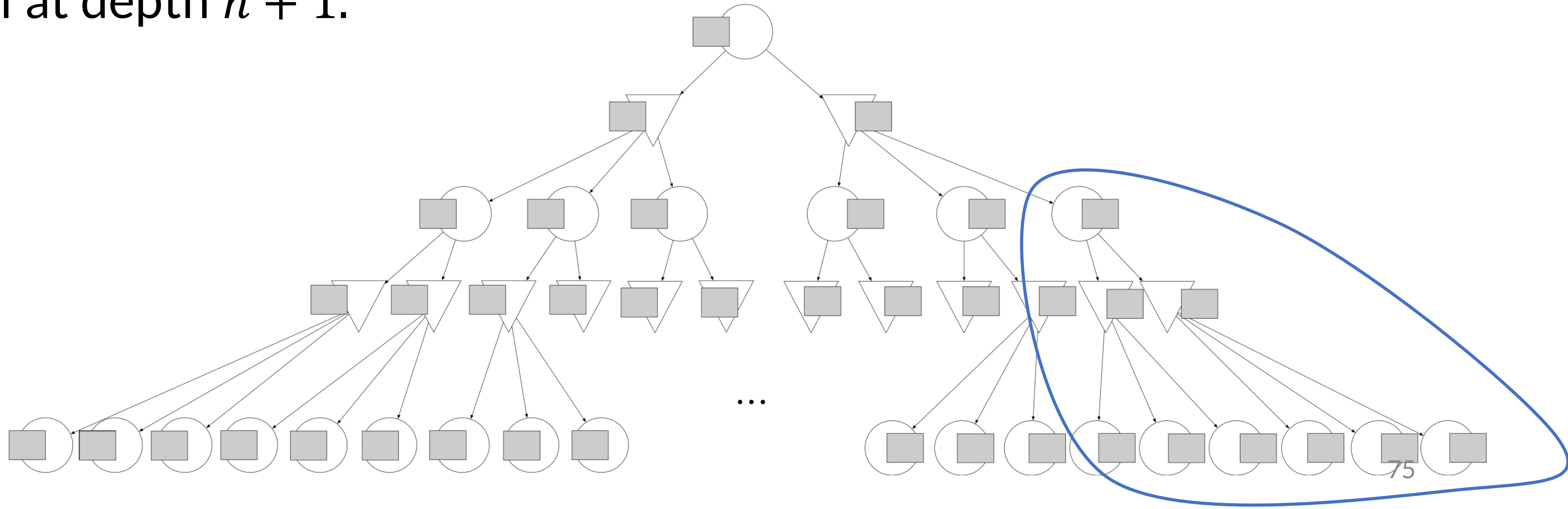
Recursive Bandits

- Sparse sampling with $H = 1 \approx$ “Uniform Random” for MAB.
- Sparse sampling with $H > 1 \approx$ naïve solution to *recursive bandit problem.*
  - To determine the “reward” for taking action at depth h, first solve MAB problem at depth $h + 1$.



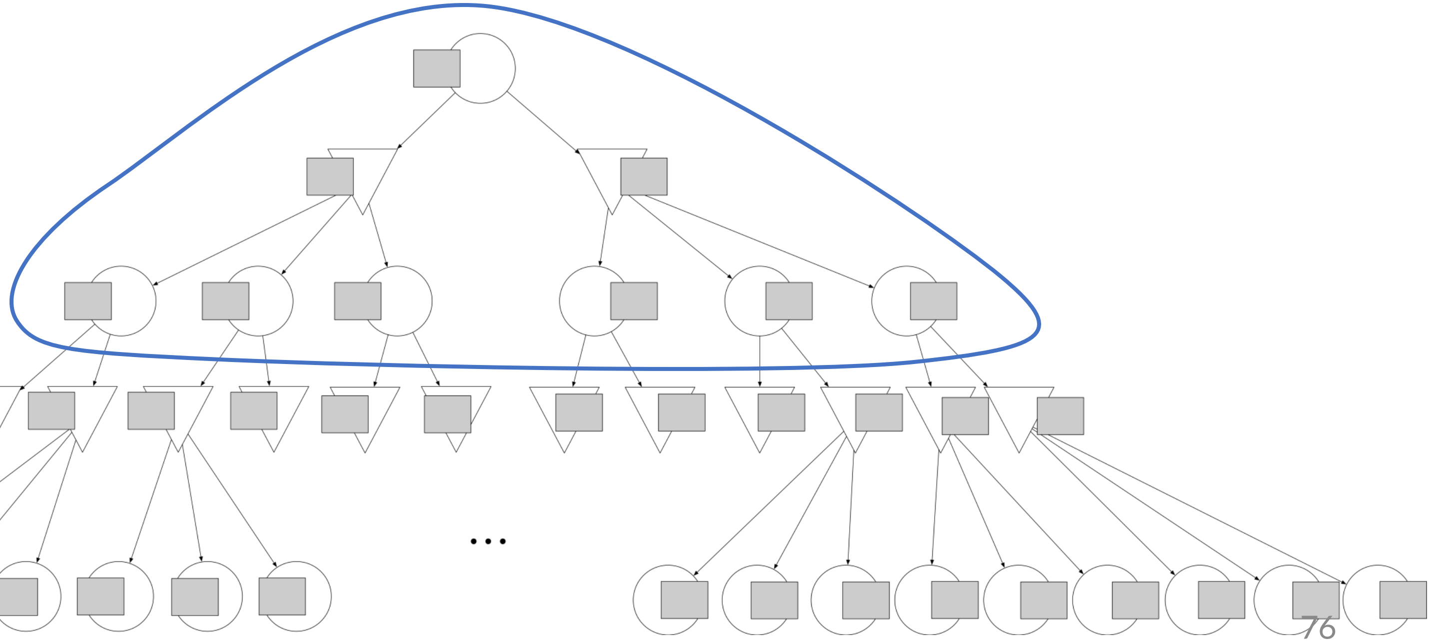
Recursive Bandits

- Sparse sampling with $H = 1 \approx$ “Uniform Random” for MAB.
- Sparse sampling with $H > 1 \approx$ naïve solution to *recursive bandit problem.*
  - To determine the “reward” for taking action at depth h, first solve MAB problem at depth $h + 1$.



Recursive Bandits

- Sparse sampling with $H = 1 \approx$ "Uniform Random" for MAB.
- Sparse sampling with $H > 1 \approx$ naïve solution to *recursive bandit problem.*
  - To determine the "reward" for taking action at depth h, first solve MAB problem at depth $h + 1$.



Recursive Bandits

- Sparse sampling with $H = 1 \approx$ "Uniform Random" for MAB.
- Sparse sampling with $H > 1 \approx$ naïve solution to *recursive bandit problem.*
  - To determine the "reward" for taking action at depth h, first solve MAB problem at depth $h + 1$.
- Could we recursively apply bandit approaches like UCB within sparse sampling?
  - Sure!

Regret in Recursive Bandits

- At the root, it’s clear that we care about *simple* regret, rather than *cumulative* regret.

Regret in Recursive Bandits

- At the root, it's clear that we care about *simple* regret, rather than *cumulative* regret.
- However, beyond the root, the story is less clear [1].
- Each non-root state node s needs to figure out *both* the best action to take at s, *and* the value $V(s)$, for use by ancestors.

[1] "Simple Regret Optimization in Online Planning for Markov Decision Processes." Feldman & Domshlak (2012).

Regret in Recursive Bandits

- At the root, it's clear that we care about *simple* regret, rather than *cumulative* regret.
- However, beyond the root, the story is less clear [1].
- Each non-root state node s needs to figure out *both* the best action to take at s, *and* the value $V(s)$, for use by ancestors.
- These are somewhat competing: if all I need is to check that a is best, it could make sense to thoroughly check other actions, making sure they're not better.
- But if what I need is $V(s) = Q(s, a)$, then I need more a samples.

[1] "Simple Regret Optimization in Online Planning for Markov Decision Processes." Feldman & Domshlak (2012).

Regret in Recursive Bandits

- At the root, it’s clear that we care about *simple* regret, rather than *cumulative* regret.
- However, beyond the root, the story is less clear [1].
- Each non-root state node s needs to figure out *both* the best action to take at s, *and* the value $V(s)$, for use by ancestors.
- These are somewhat competing: if all I need is to check that a is best, it could make sense to thoroughly check other actions, making sure they’re not better.
- But if what I need is $V(s) = Q(s, a)$, then I need more a samples.
- For this reason, some works (e.g. [2]) advocate using one strategy at/near the root, and a different strategy elsewhere.

[1] “Simple Regret Optimization in Online Planning for Markov Decision Processes.” Feldman & Domshlak (2012).
[2] “Minimizing Simple and Cumulative Regret in Monte-Carlo Tree Search.” Pepels et al. (2014).

Limitation of Sparse Sampling + UCB

- Even with a smarter bandits strategy, sparse sampling suffers from poor *anytime performance*
  - Anytime performance: evaluation of the best policy found for any given computational budget (e.g., wall clock time)

Limitation of Sparse Sampling + UCB

- Even with a smarter bandits strategy, sparse sampling suffers from poor *anytime performance*
  - Anytime performance: evaluation of the best policy found for any given computational budget (e.g., wall clock time)
- In general, sparse sampling completely evaluates each subtree before returning to the parent.

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS)

Brings together many of the ideas we have seen:

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS)

Brings together many of the ideas we have seen:

1. Restricting to *reachable* states by building out AODAG

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS)

Brings together many of the ideas we have seen:

1. Restricting to *reachable* states by building out AODAG
2. Using *heuristics* to evaluate leaves

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS)

Brings together many of the ideas we have seen:

1. Restricting to *reachable* states by building out AODAG
2. Using *heuristics* to evaluate leaves
3. Sparse sampling of transition model

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS)

Brings together many of the ideas we have seen:

1. Restricting to *reachable* states by building out AODAG
2. Using *heuristics* to evaluate leaves
3. Sparse sampling of transition model
4. MAB exploration techniques

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS)

Brings together many of the ideas we have seen:

1. Restricting to *reachable* states by building out AODAG
2. Using *heuristics* to evaluate leaves
3. Sparse sampling of transition model
4. MAB exploration techniques
5. Expanding AODAG gradually

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS)

Brings together many of the ideas we have seen:

1. Restricting to *reachable* states by building out AODAG
2. Using *heuristics* to evaluate leaves
3. Sparse sampling of transition model
4. MAB exploration techniques
5. Expanding AODAG gradually

One new idea: estimating heuristics with *rollouts*.

Estimating Heuristics with Rollouts

- To get cheap heuristic for a state, MCTS uses **rollouts**.
- A rollout is a trajectory of states, actions, and rewards sampled from the MDP with a policy π_{rollout}.

Estimating Heuristics with Rollouts

- To get cheap heuristic for a state, MCTS uses **rollouts**.
- A rollout is a trajectory of states, actions, and rewards sampled from the MDP with a policy π_{rollout}.
- Estimated value is average of cumulative rollout rewards
  - With temporal discounting applied as needed

Estimating Heuristics with Rollouts

- To get cheap heuristic for a state, MCTS uses **rollouts**.
- A rollout is a trajectory of states, actions, and rewards sampled from the MDP with a policy π_{rollout}.
- Estimated value is average of cumulative rollout rewards
  - With temporal discounting applied as needed
- Common choice of π_{rollout} is random action selection
  - Domain-specific knowledge or machine learning can also be used

Estimating Heuristics with Rollouts

- To get cheap heuristic for a state, MCTS uses **rollouts**.
- A rollout is a trajectory of states, actions, and rewards sampled from the MDP with a policy π_{rollout}.
- Estimated value is average of cumulative rollout rewards
  - With temporal discounting applied as needed
- Common choice of π_{rollout} is random action selection
  - Domain-specific knowledge or machine learning can also be used

When would rollouts give good or bad heuristic estimates?

Monte Carlo Tree Search (MCTS)

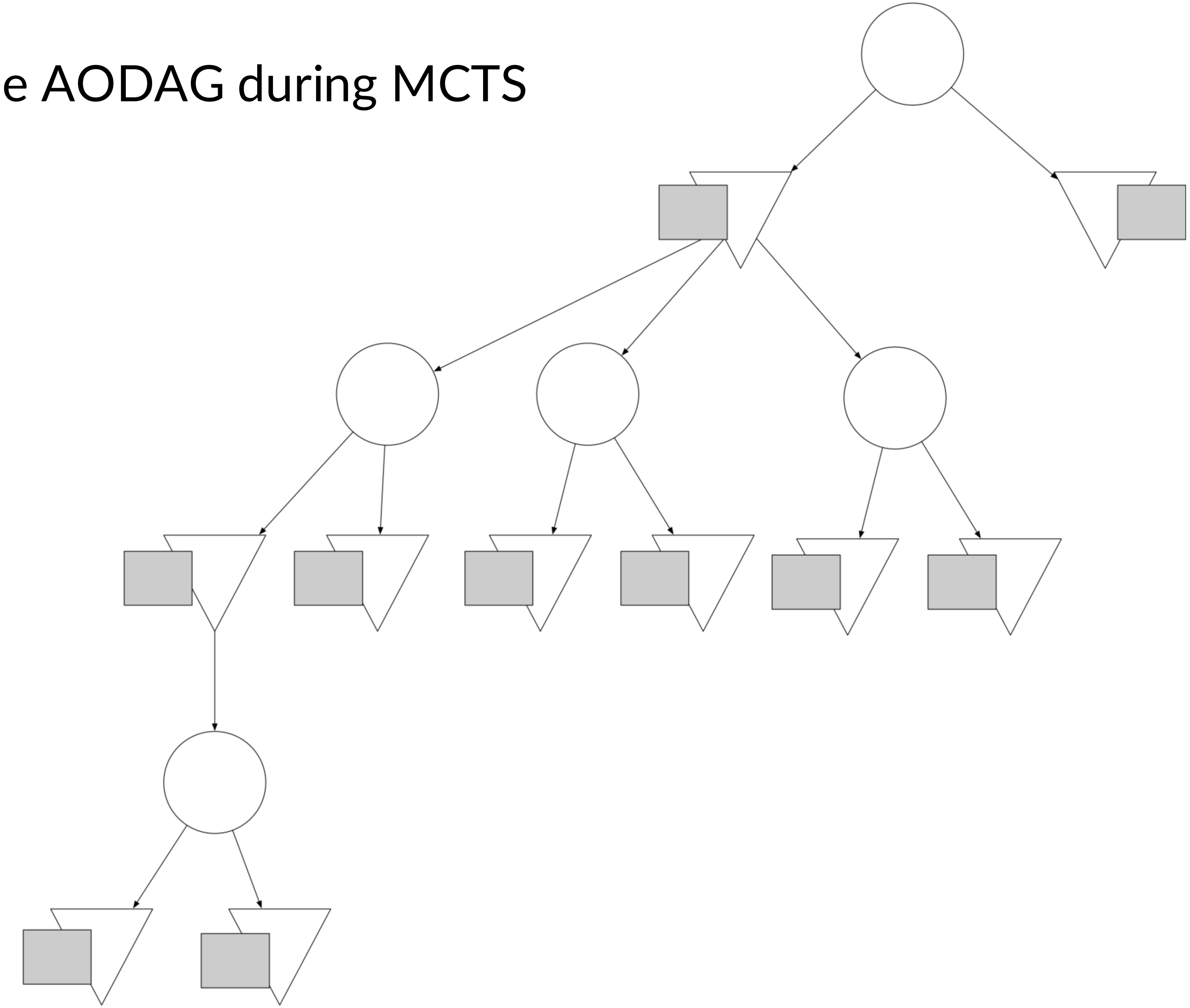
MCTS starts by initializing AODAG, $\hat{Q}$, and N.

Monte Carlo Tree Search (MCTS)

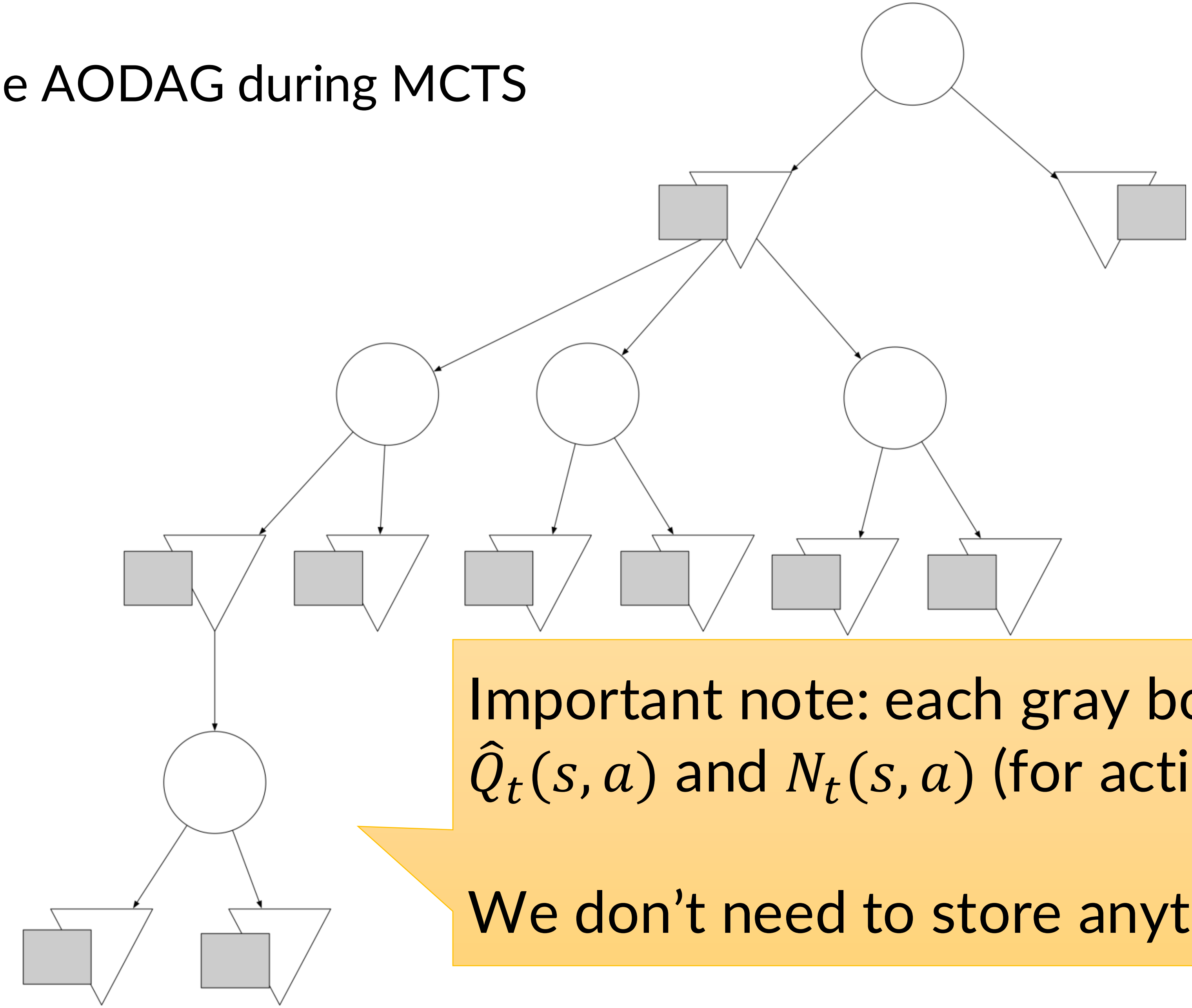
MCTS starts by initializing AODAG, $\hat{Q}$, and N.

Unlike expectimax / sparse sampling, but *like* RTDP, we're going to maintain and update $\hat{Q}$ for nodes in the AODAG, rather than calculating them once and for all.

Example AODAG during MCTS



Example AODAG during MCTS



Important note: each gray box now includes *both* $\hat{Q}_t(s, a)$ and $N_t(s, a)$ (for action nodes).

We don't need to store anything at state nodes.

Monte Carlo Tree Search (MCTS)

MCTS starts by initializing AODAG, $\hat{Q}$, and N.

Then, repeat until time runs out:

Monte Carlo Tree Search (MCTS)

MCTS starts by initializing AODAG, $\hat{Q}$, and N.

Then, repeat until time runs out:

1. **Selection**: Pick a leaf (action) node to explore.

Monte Carlo Tree Search (MCTS)

MCTS starts by initializing AODAG, $\hat{Q}$, and N.

Then, repeat until time runs out:

1. **Selection**: Pick a leaf (action) node to explore.

<u>Pick the leaf as follows:</u>

1. Start at the root
2. Select an action (using *tree policy*)
3. Sample a next state
4. Repeat 2-3 until a leaf is reached

Monte Carlo Tree Search (MCTS)

MCTS starts by initializing AODAG, $\hat{Q}$, and N.

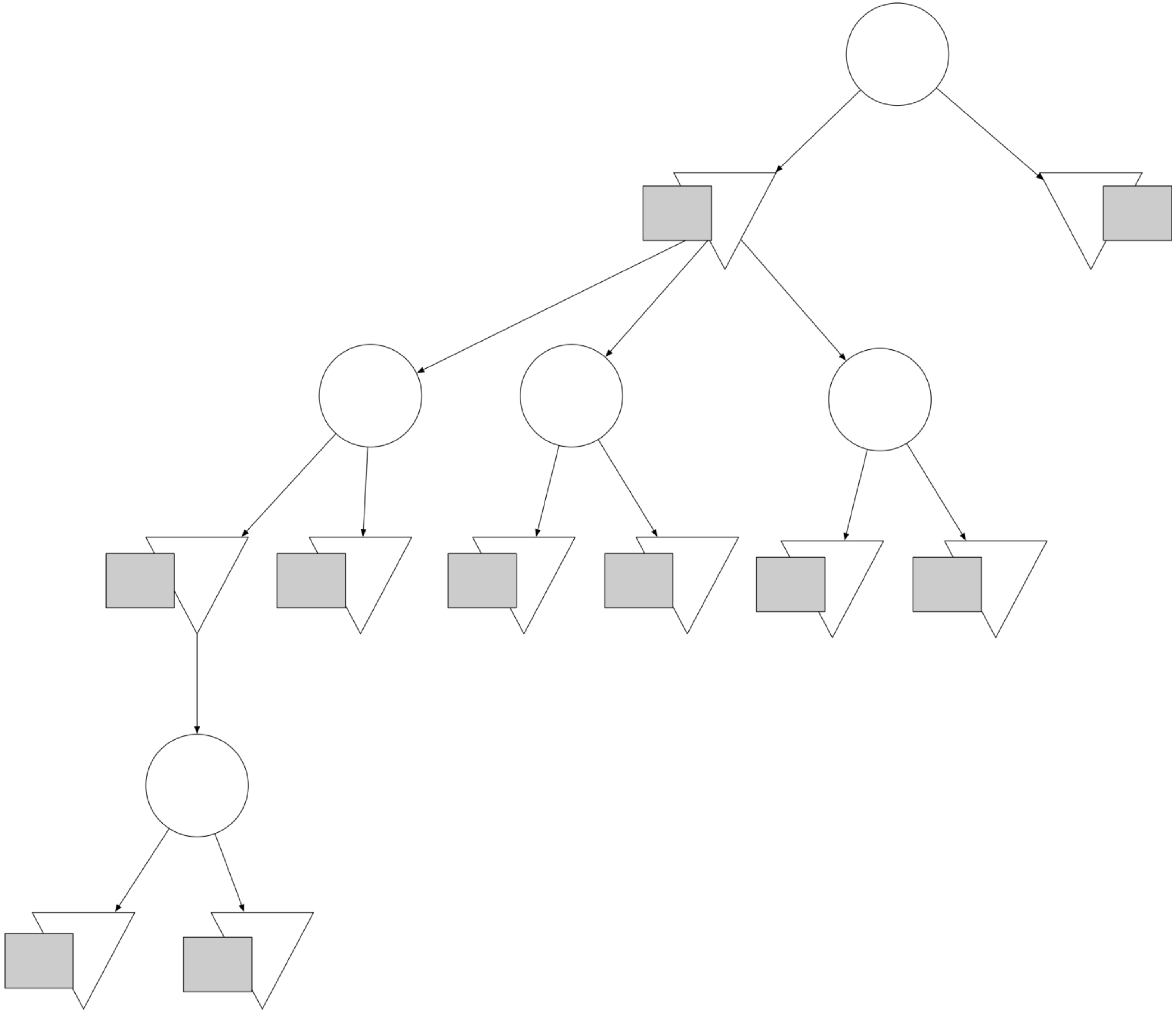
Then, repeat until time runs out:

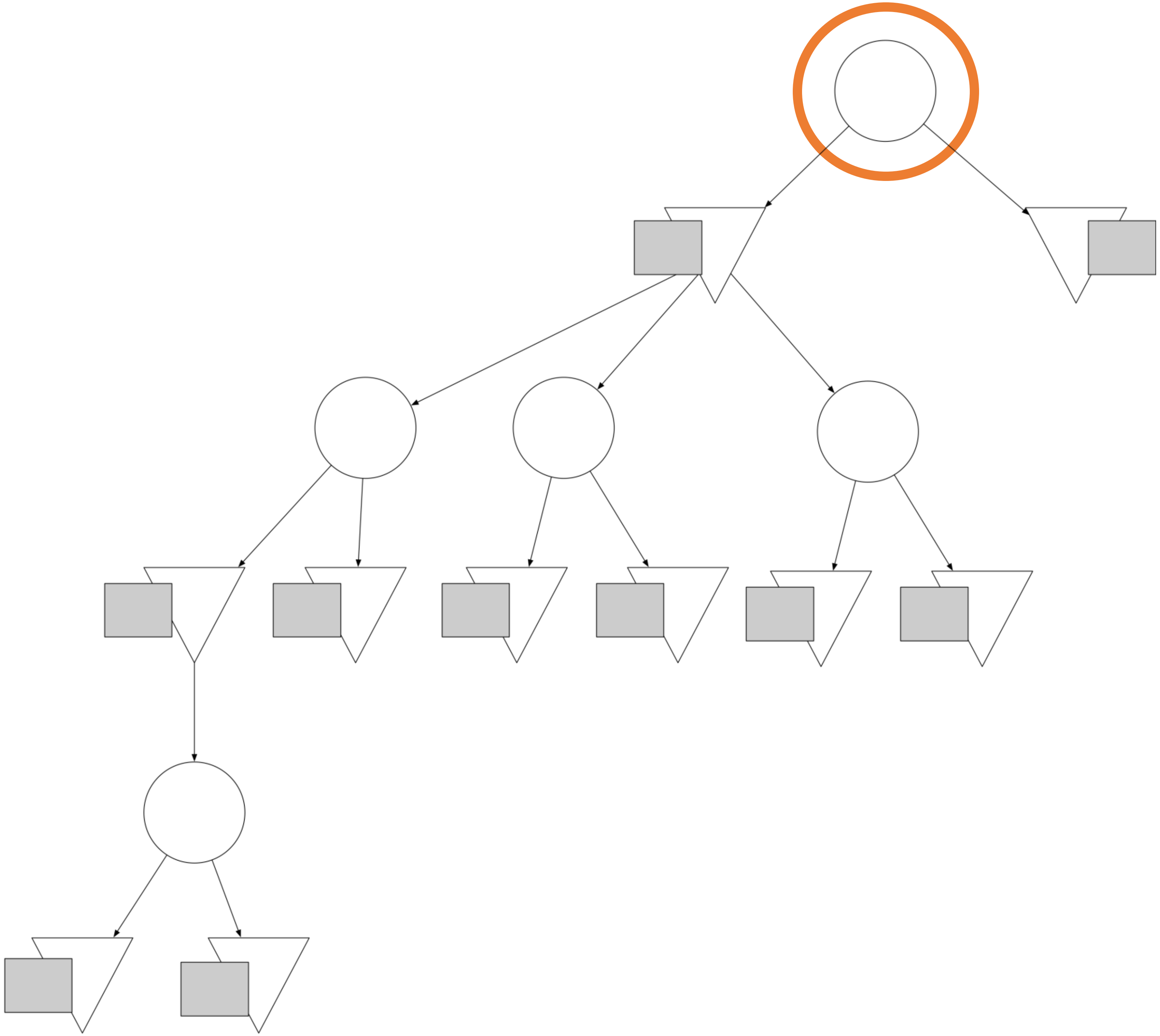
1. **Selection**: Pick a leaf (action) node to explore.

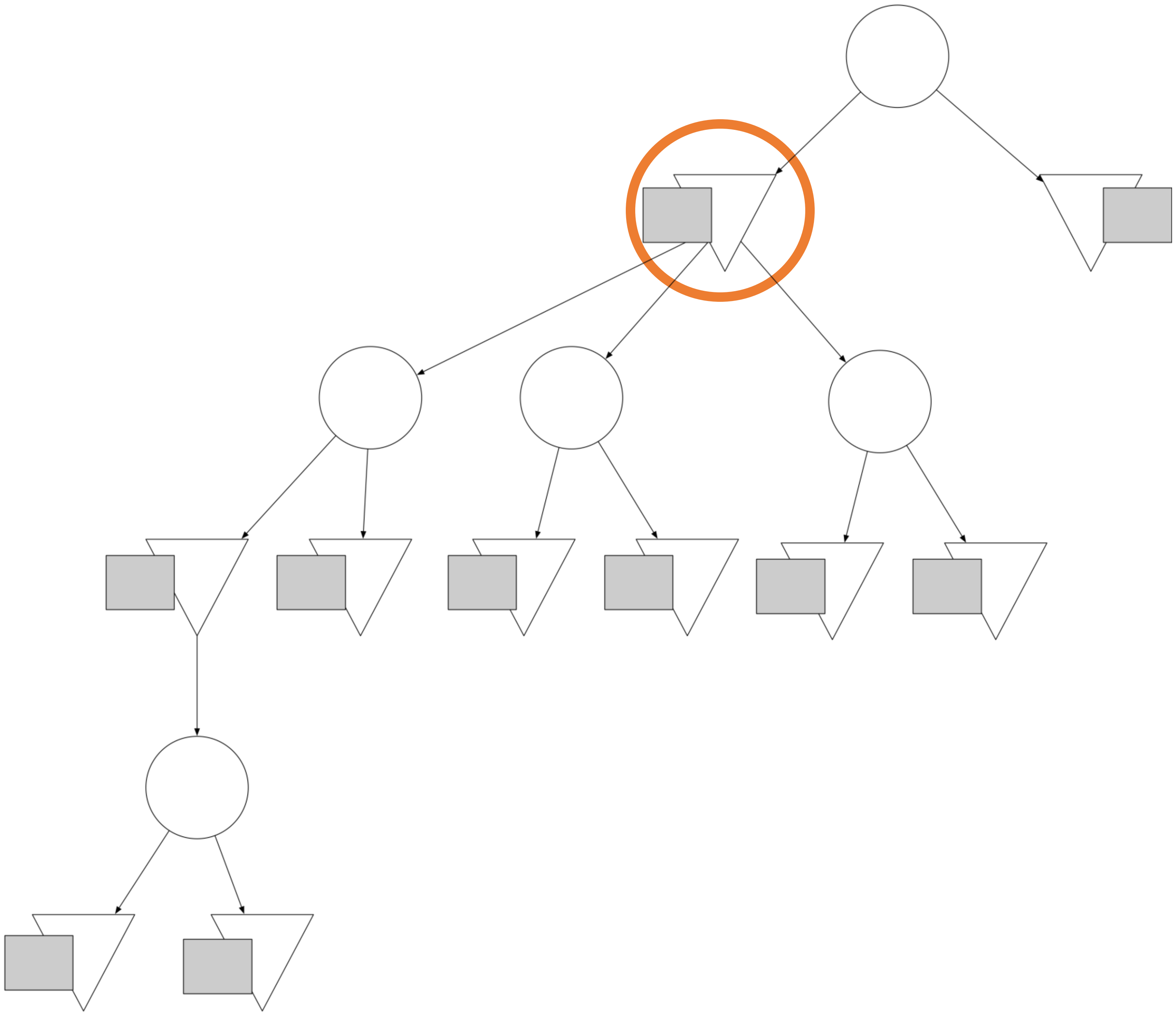
Pick the leaf as follows:

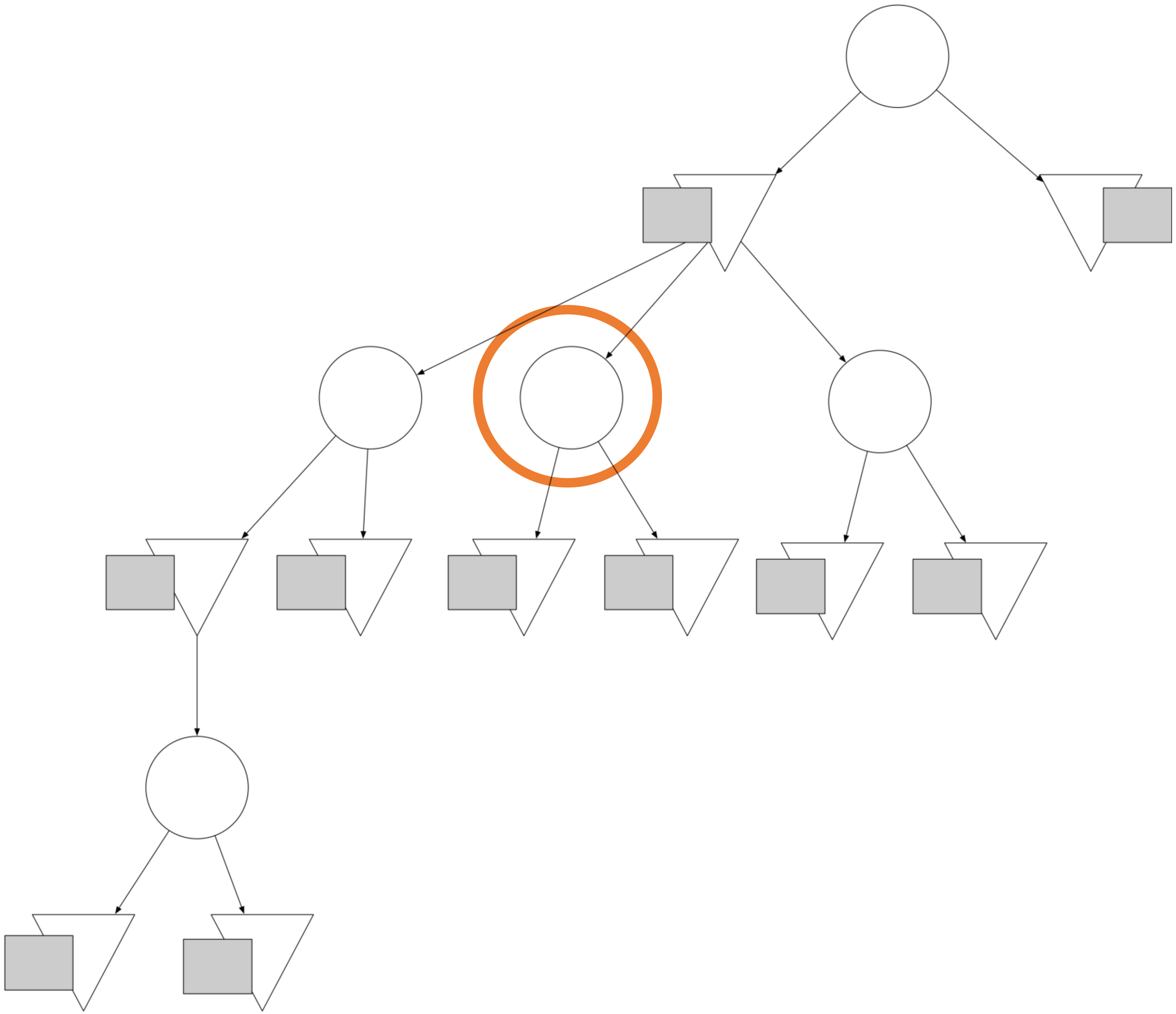
1. Start at the root
2. Select an action (using *tree policy*)
3. Sample a next state
4. Repeat 2-3 until a leaf is reached

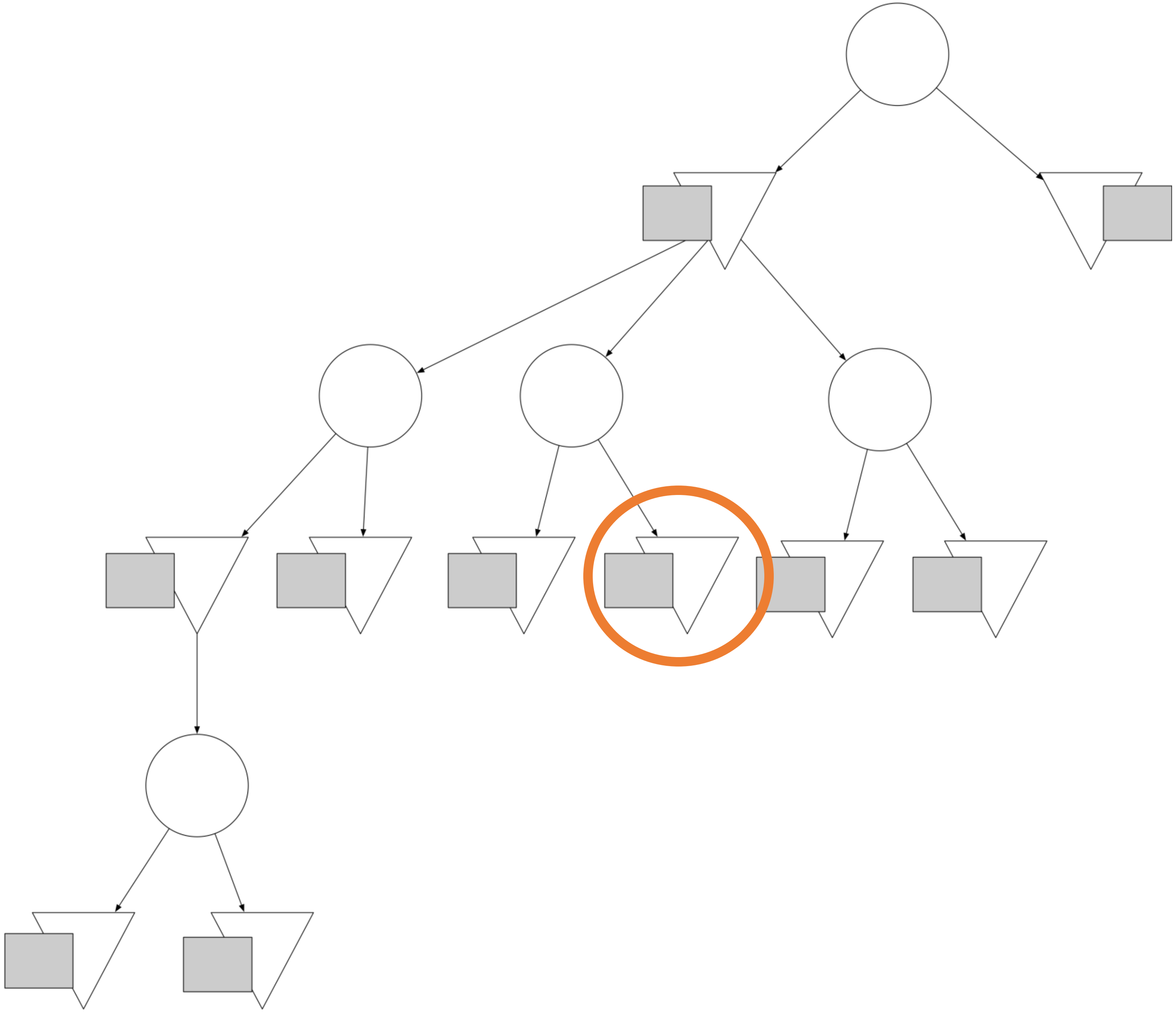
Use MAB ideas









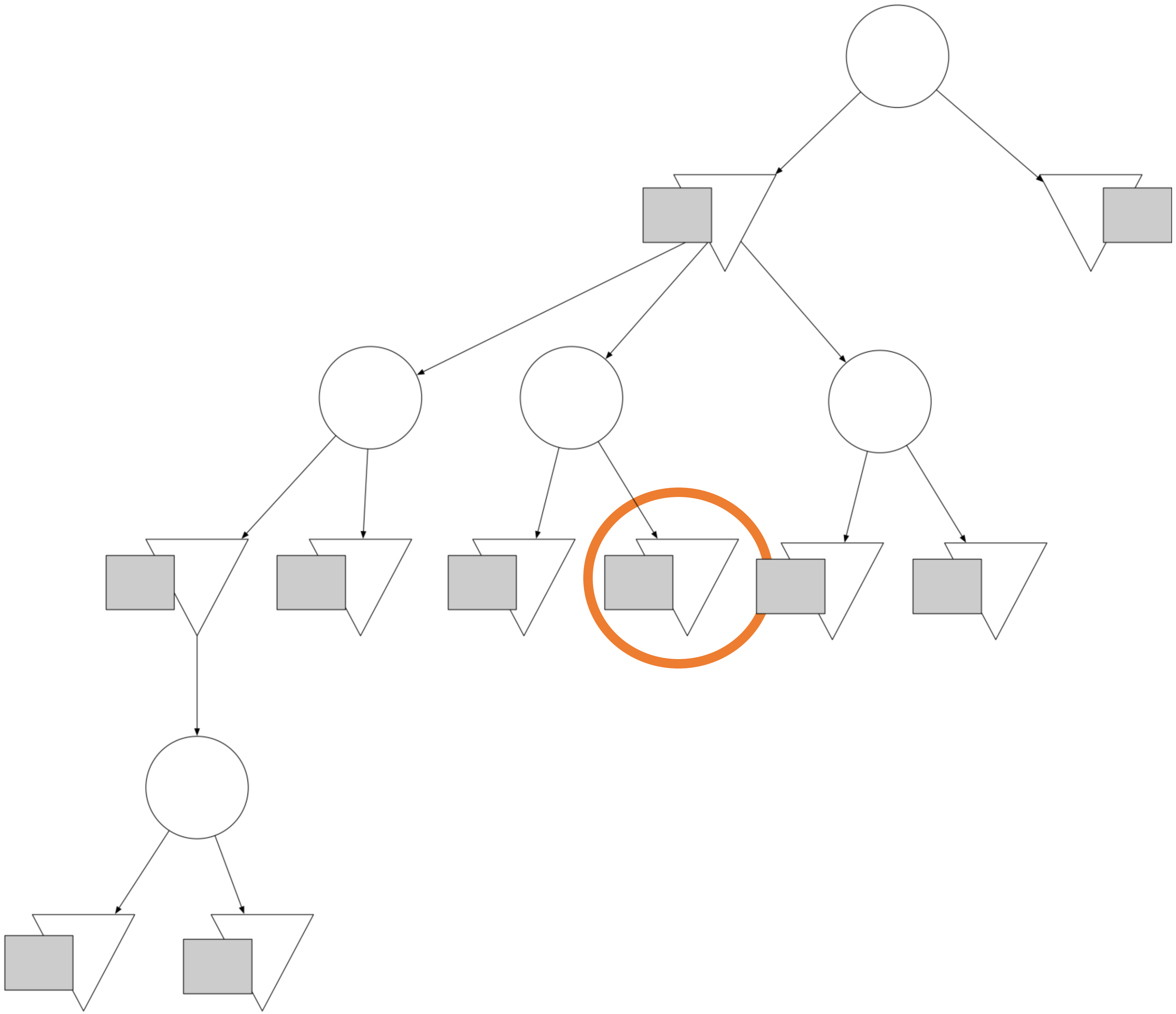


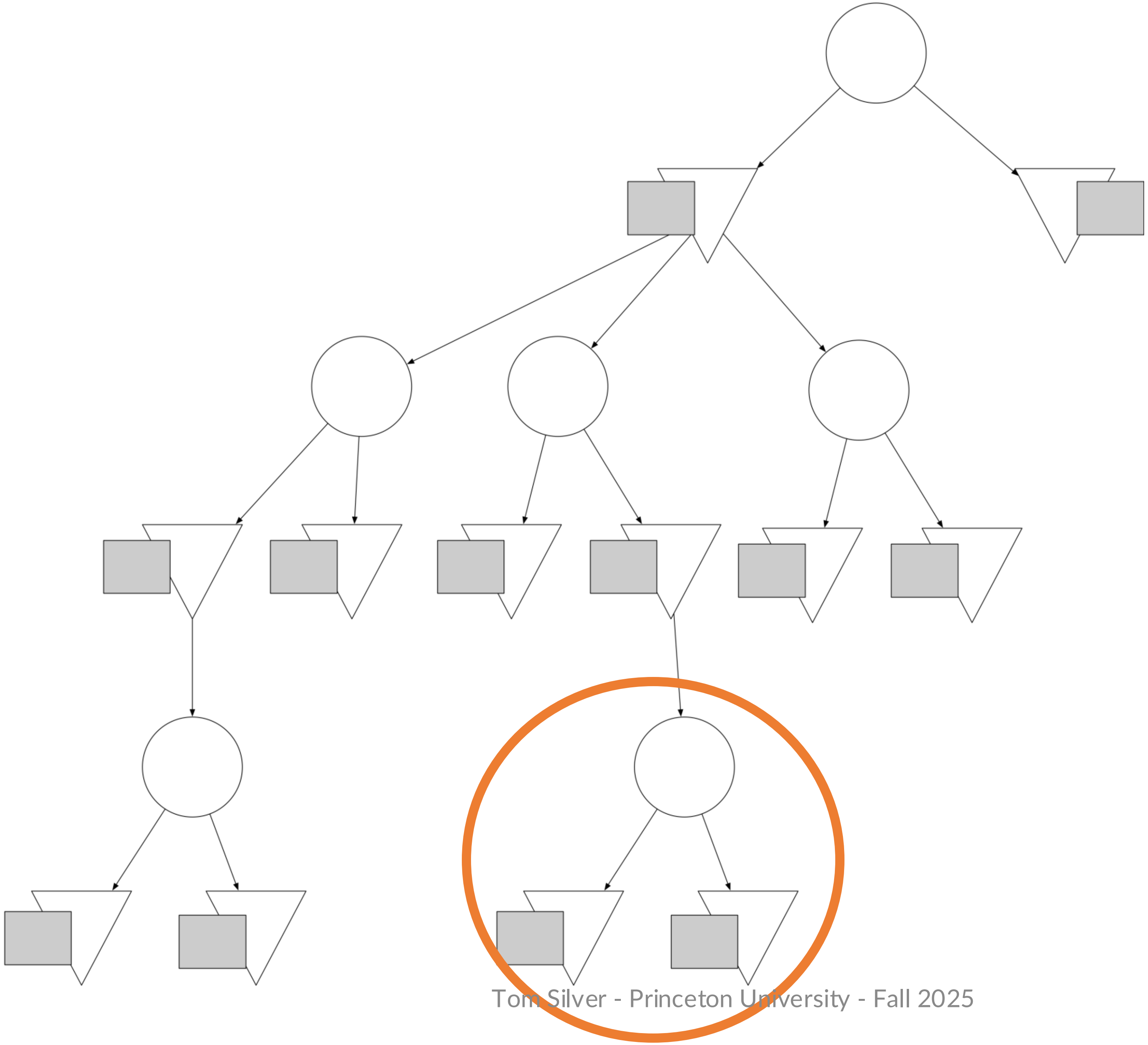
Monte Carlo Tree Search (MCTS)

MCTS starts by initializing AODAG, $\hat{Q}$, and N.

Then, repeat until time runs out:

1. **Selection**: Pick a leaf (action) node to explore.
2. **Expansion:** Sample a next state. Create a new state node and new child action nodes, one per possible action.



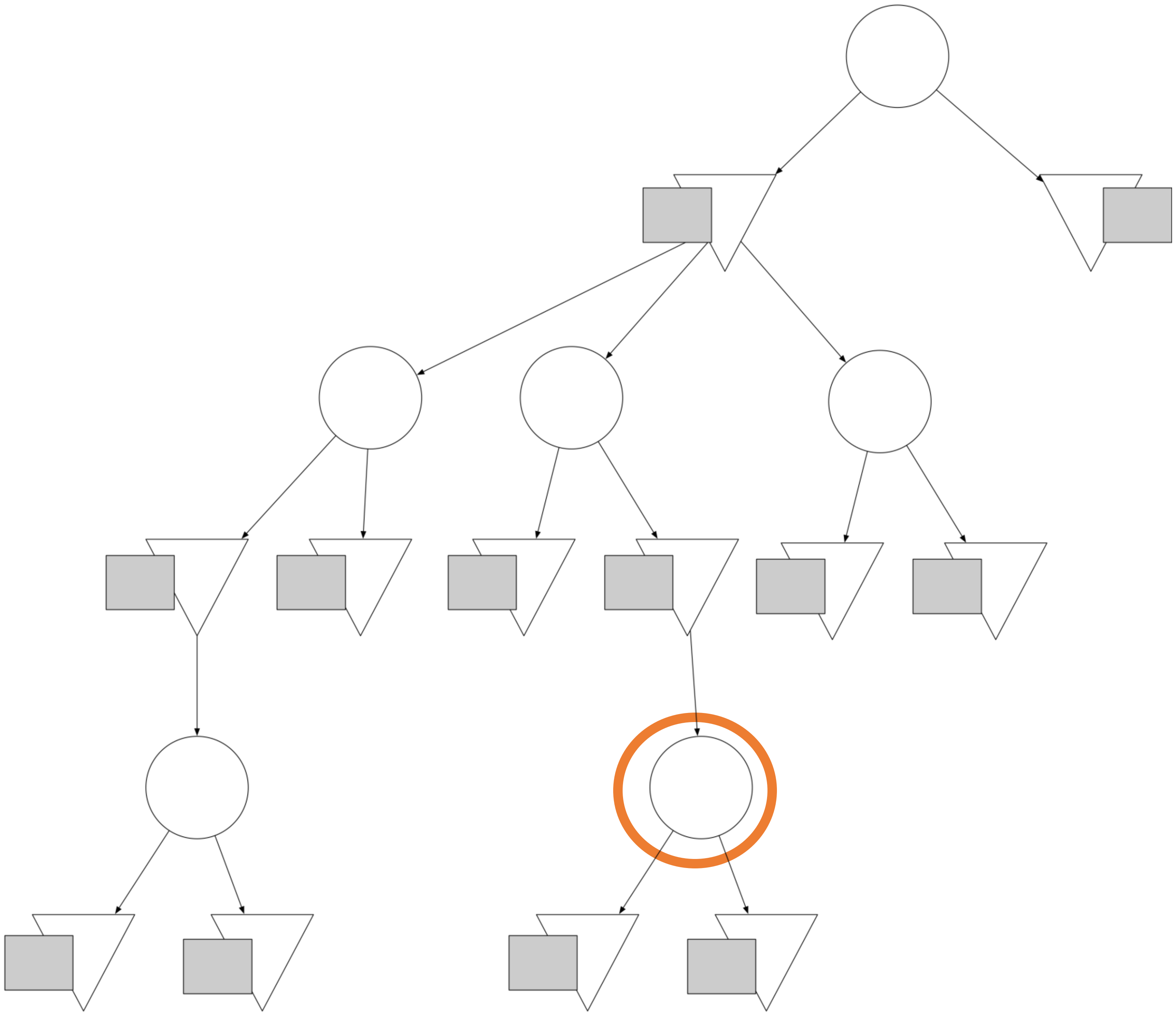


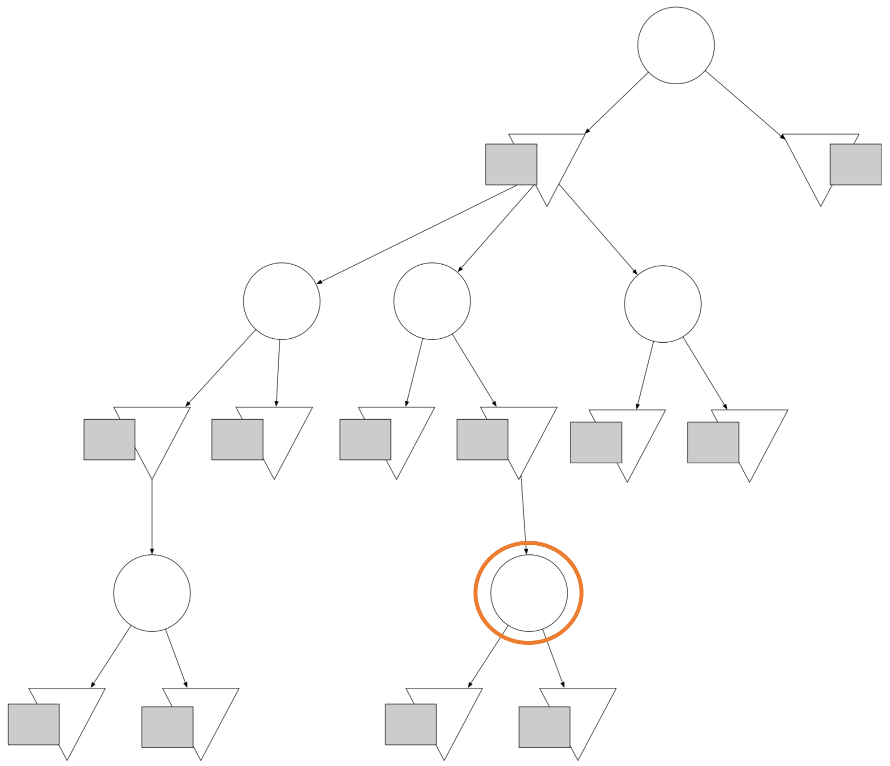
Monte Carlo Tree Search (MCTS)

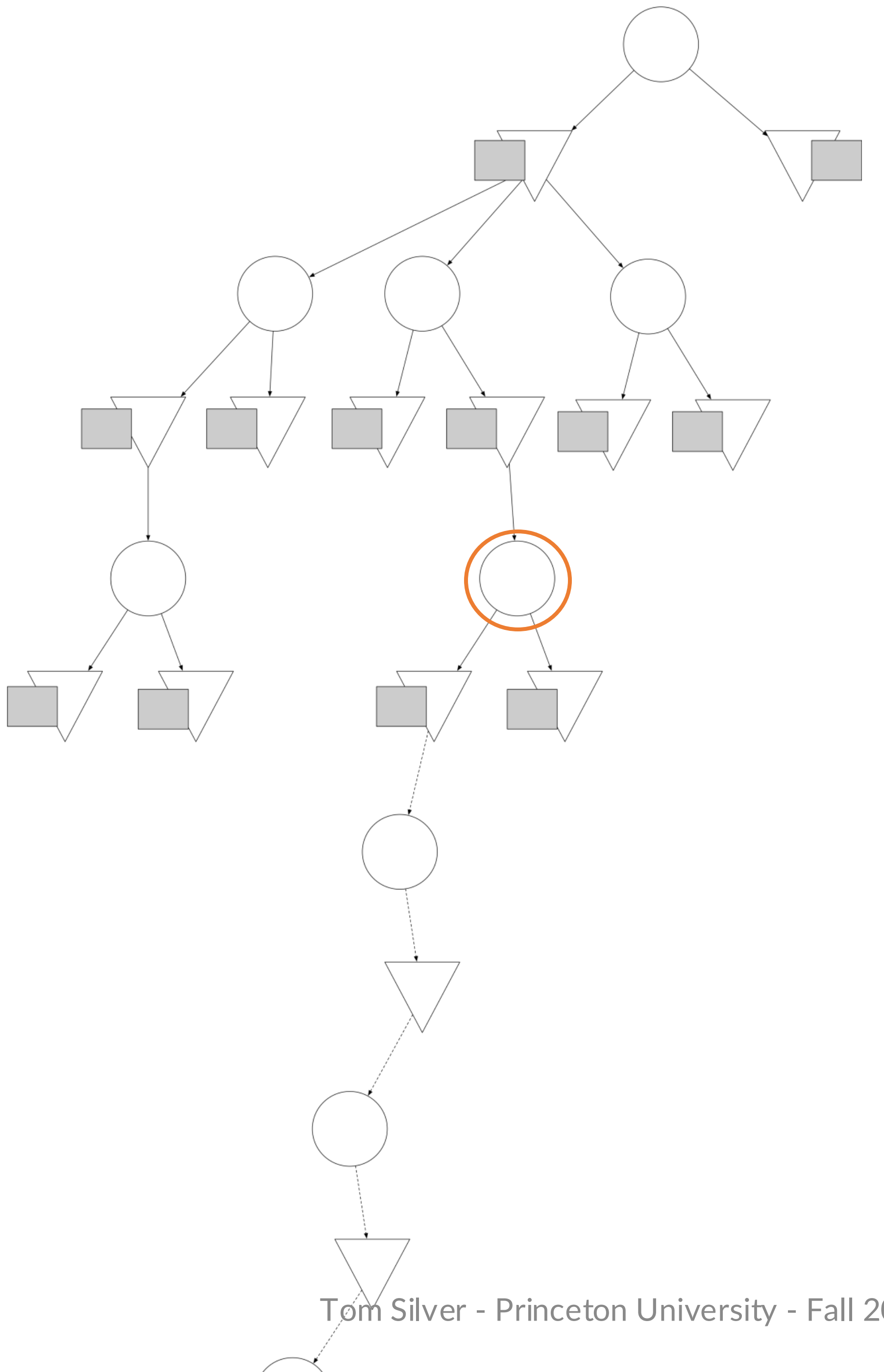
MCTS starts by initializing AODAG, $\hat{Q}$, and N.

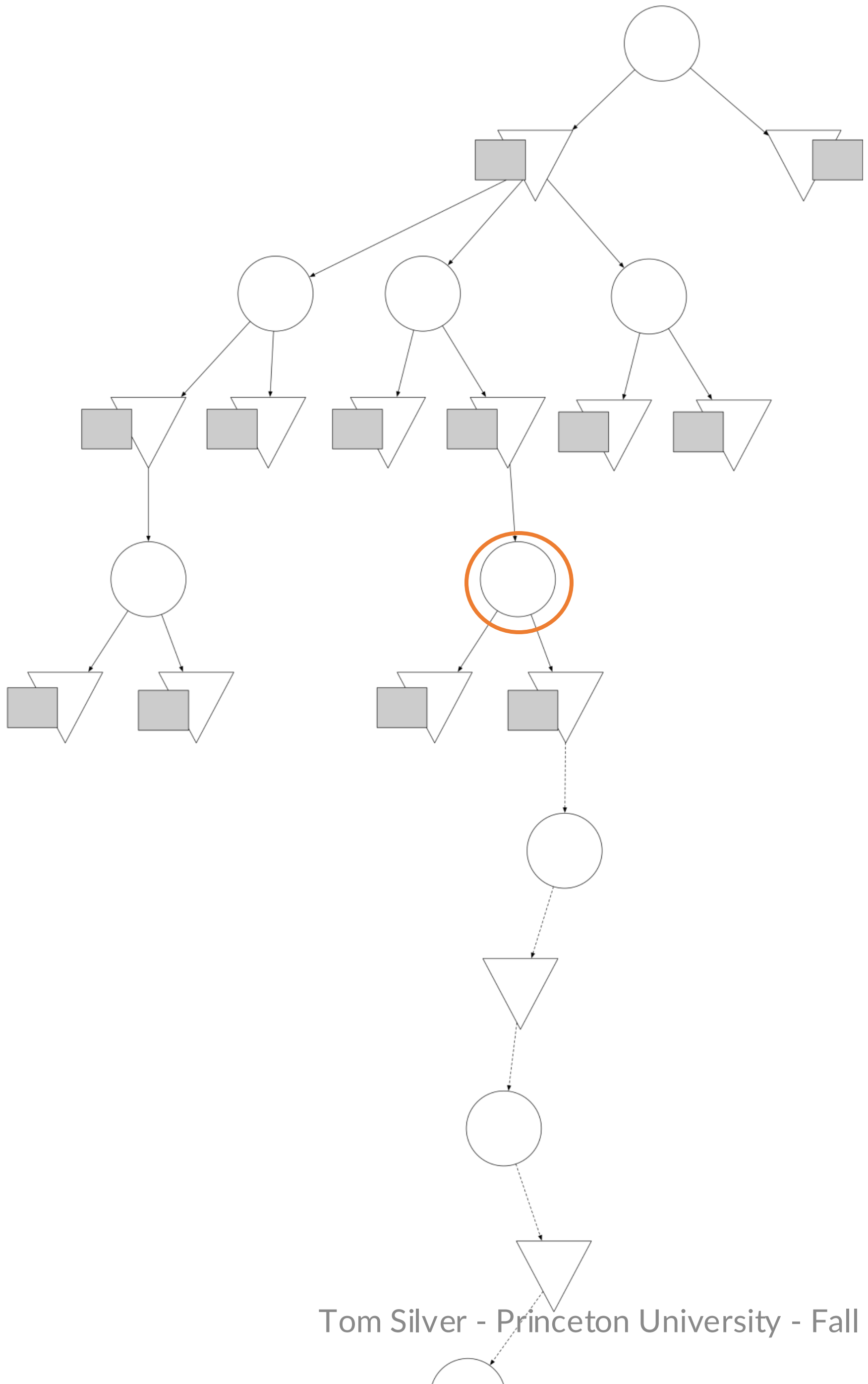
Then, repeat until time runs out:

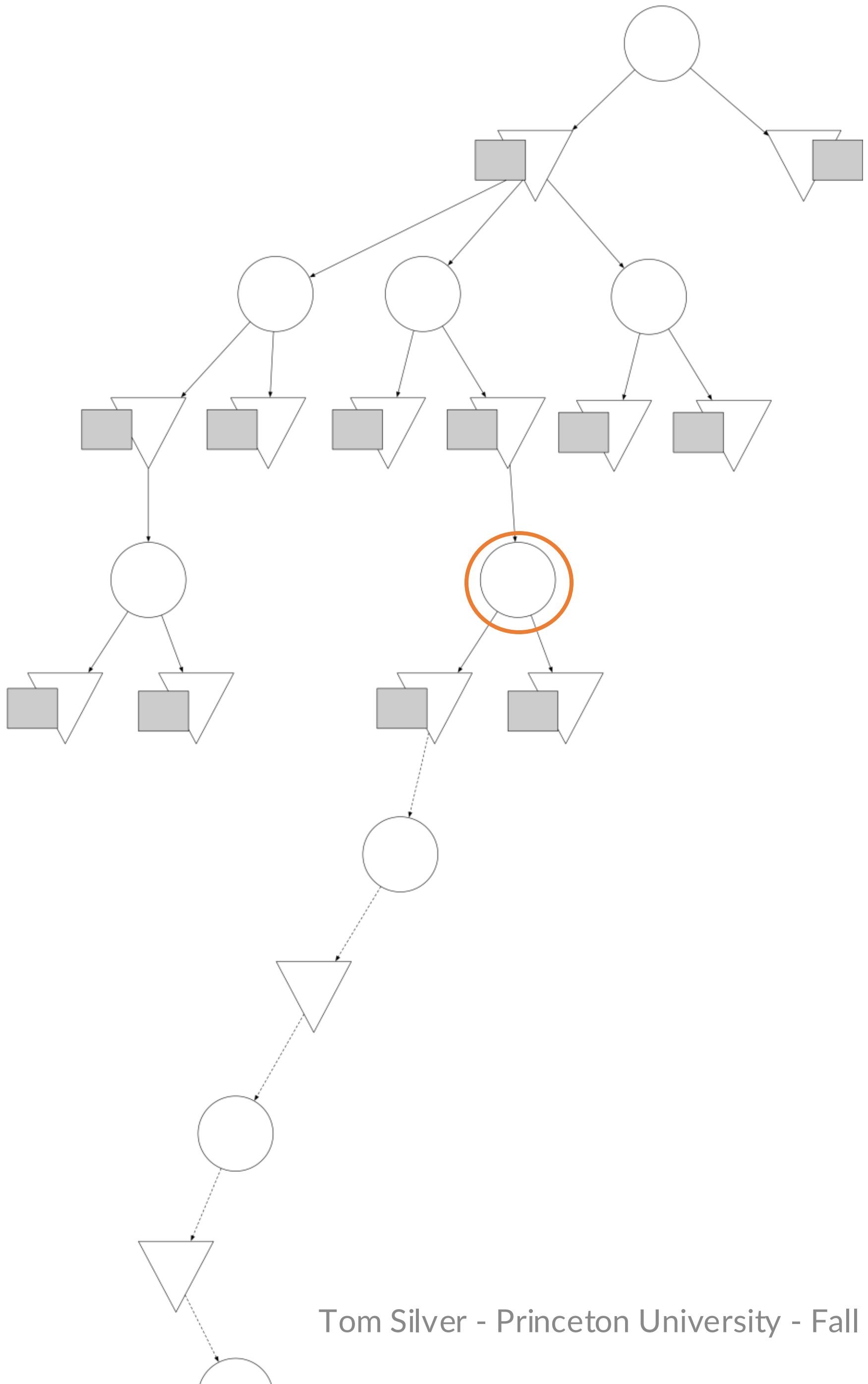
1. **Selection**: Pick a leaf (action) node to explore.
2. **Expansion:** Sample a next state. Create a new state node and new child action nodes, one per possible action.
3. **Simulation**: Calculate a heuristic value for the new state node using *rollouts*.

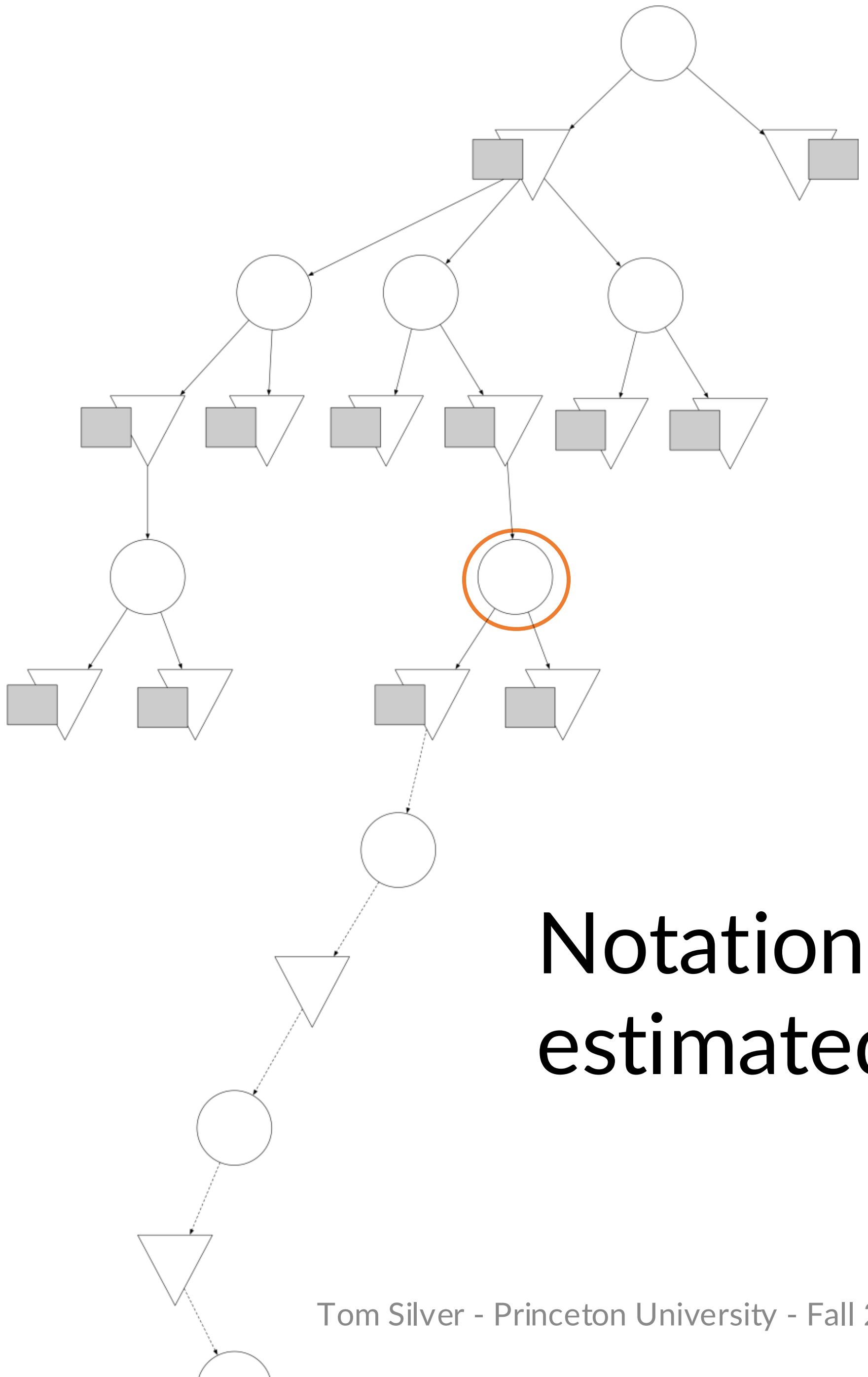












Notation: let ρ denote the estimated heuristic from rollouts.

Monte Carlo Tree Search (MCTS)

MCTS starts by initializing AODAG, $\hat{Q}$, and N.

Then, repeat until time runs out:

1. **Selection**: Pick a leaf (action) node to explore.
2. **Expansion:** Sample a next state. Create a new state node and new child action nodes, one per possible action.
3. **Simulation**: Calculate a heuristic value for the new state node using *rollouts*.

But if you have a heuristic, maybe use that instead!
But, good to be *admissible*.

Monte Carlo Tree Search (MCTS)

MCTS starts by initializing AODAG, $\hat{Q}$, and N.

Then, repeat until time runs out:

1. **Selection**: Pick a leaf (action) node to explore.
2. **Expansion:** Sample a next state. Create a new state node and new child action nodes, one per possible action.
3. **Simulation**: Calculate a heuristic value for the new state node using *rollouts*.
4. **Backpropagation:** Update $\hat{Q}$ and N for the selected state and action and all ancestors.

Monte Carlo Tree Search (MCTS)

MCTS starts by initializing AODAG, $\hat{Q}$, and N.

Then, repeat until time runs out:

1. **Selection**: Pick a leaf (action) node to explore.
2. **Expansion:** Sample a next state. Create a new state node and new child action nodes, one per possible action.
3. **Simulation**: Calculate a heuristic value for the new state node using *rollouts*.
4. **Backpropagation:** Update $\hat{Q}$ and N for the selected state and action and all ancestors.

Not neural network backprop!

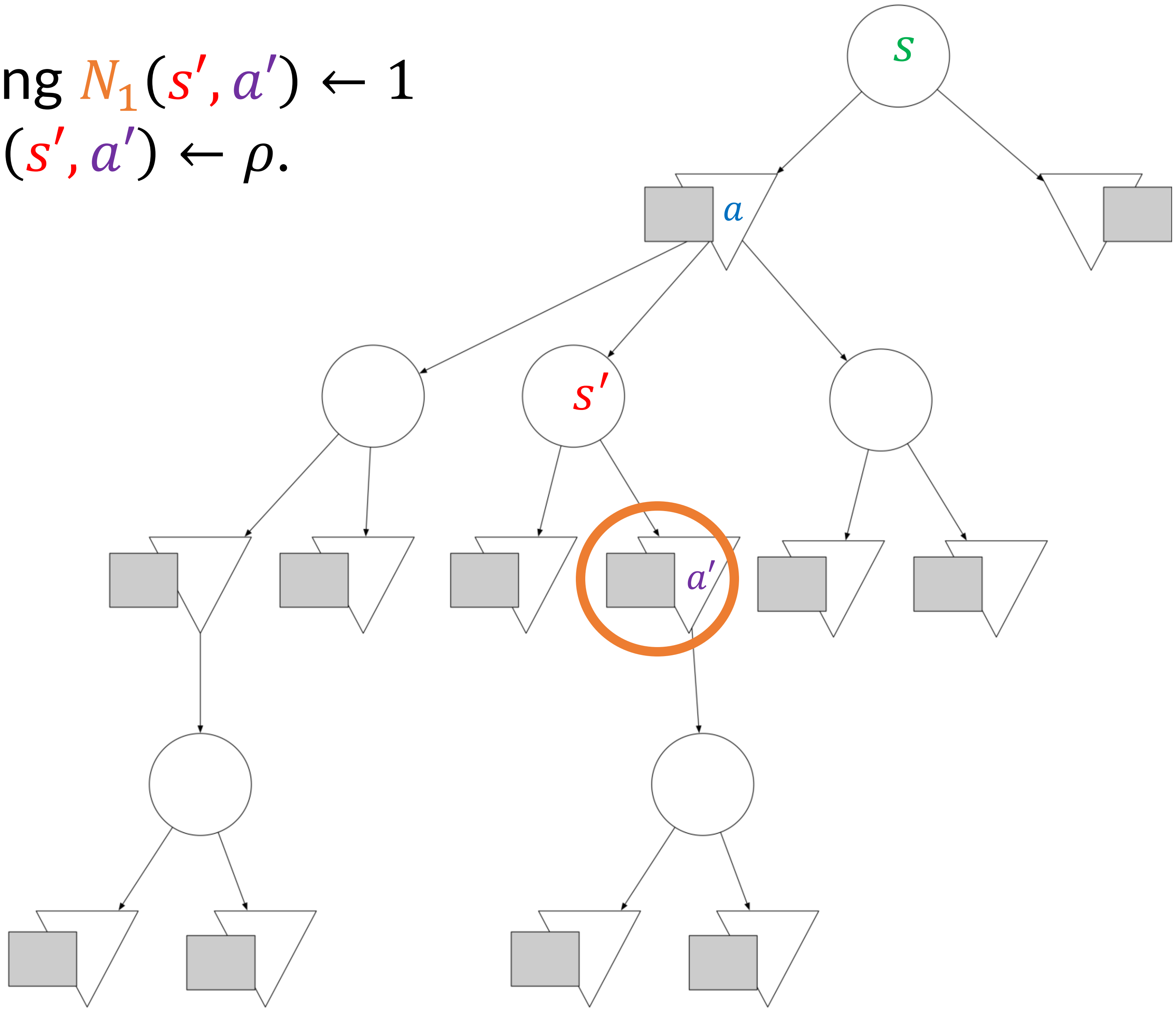
MCTS Backpropagation

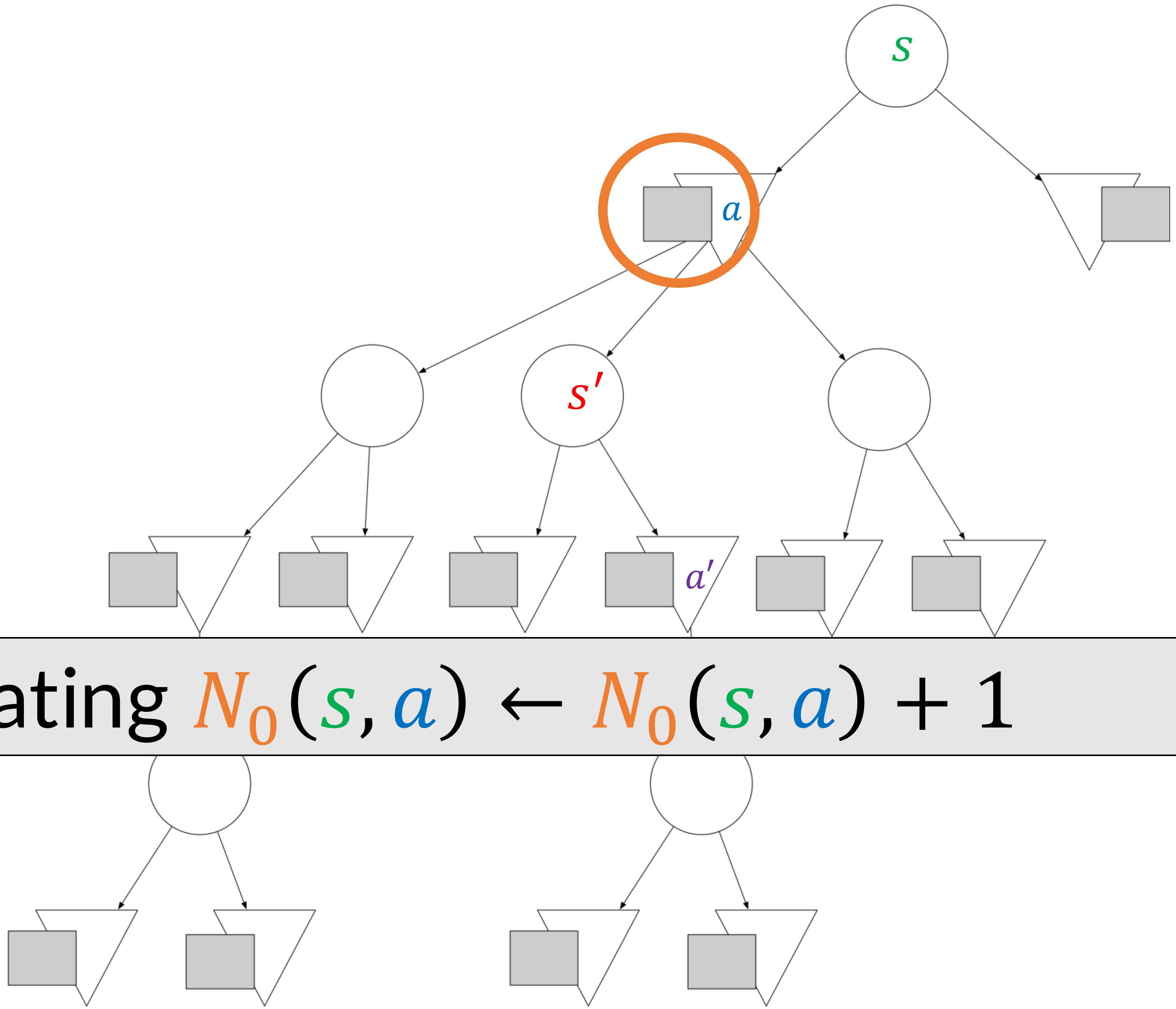
- $\widehat{Q_t}(s, a)$ will be the average of all cumulative rewards seen during planning, when starting at s at time t and taking a.

- And, $N_t(s, a)$ should be the visitation counts.

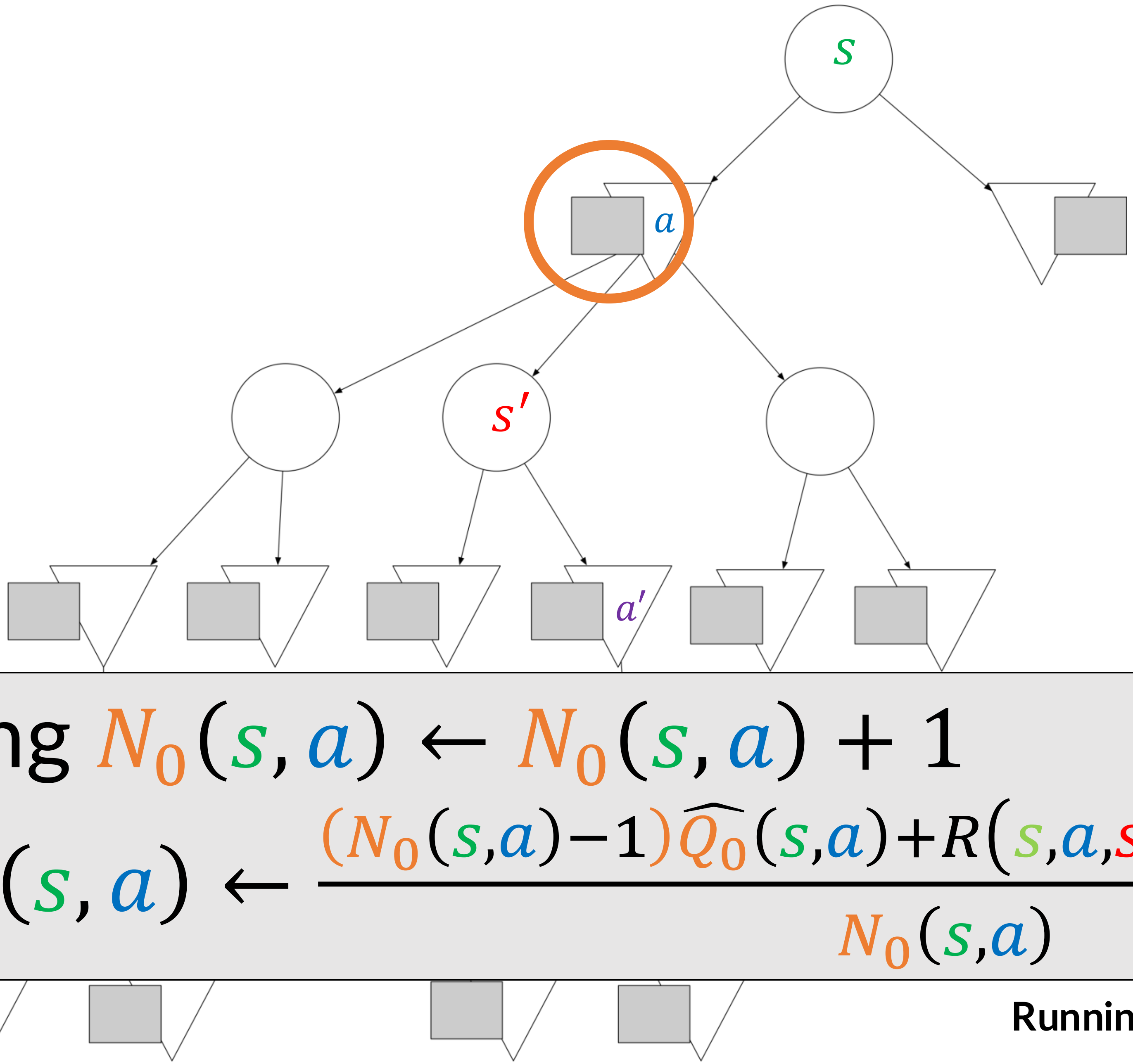
- Backpropagation: given one new trajectory, update $\hat{Q}, N$.

Updating $N_1(s', a') \leftarrow 1$ and $\widehat{Q_1}(s', a') \leftarrow \rho$.





Updating $N_0(s, a) \leftarrow N_0(s, a) + 1$



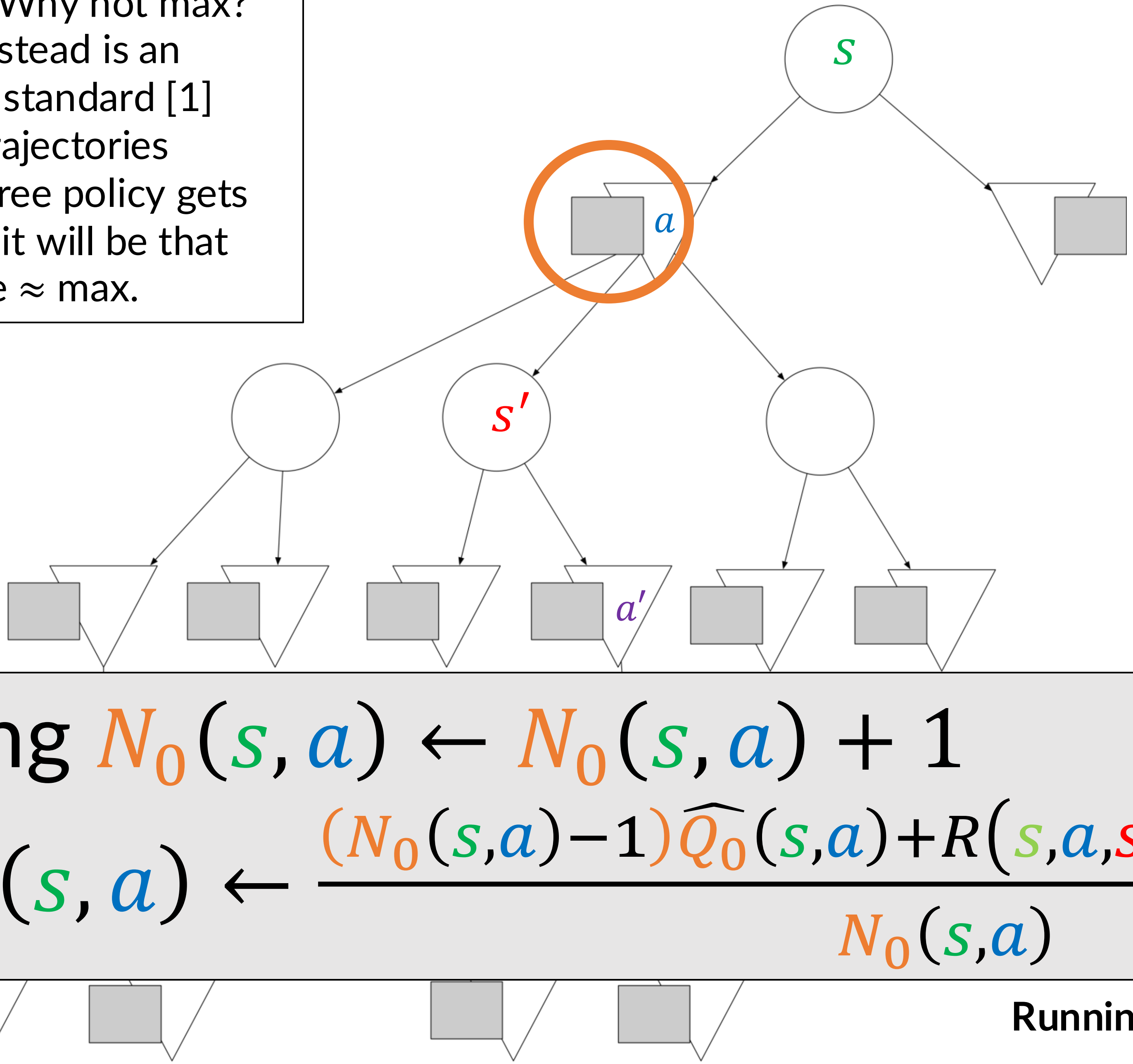
Updating $N_0(s,a) \leftarrow N_0(s,a) + 1$

and $\widehat{Q_0}(s,a) \leftarrow \dfrac{(N_0(s,a)-1)\widehat{Q_0}(s,a)+R(s,a,s')+\gamma\widehat{Q_1}(s',a')}{N_0(s,a)}$

Running average!

Running average? Why not max?

- Taking a max instead is an option, but less standard [1]
- As number of trajectories increases, and tree policy gets more exploit-y, it will be that running average ≈ max.



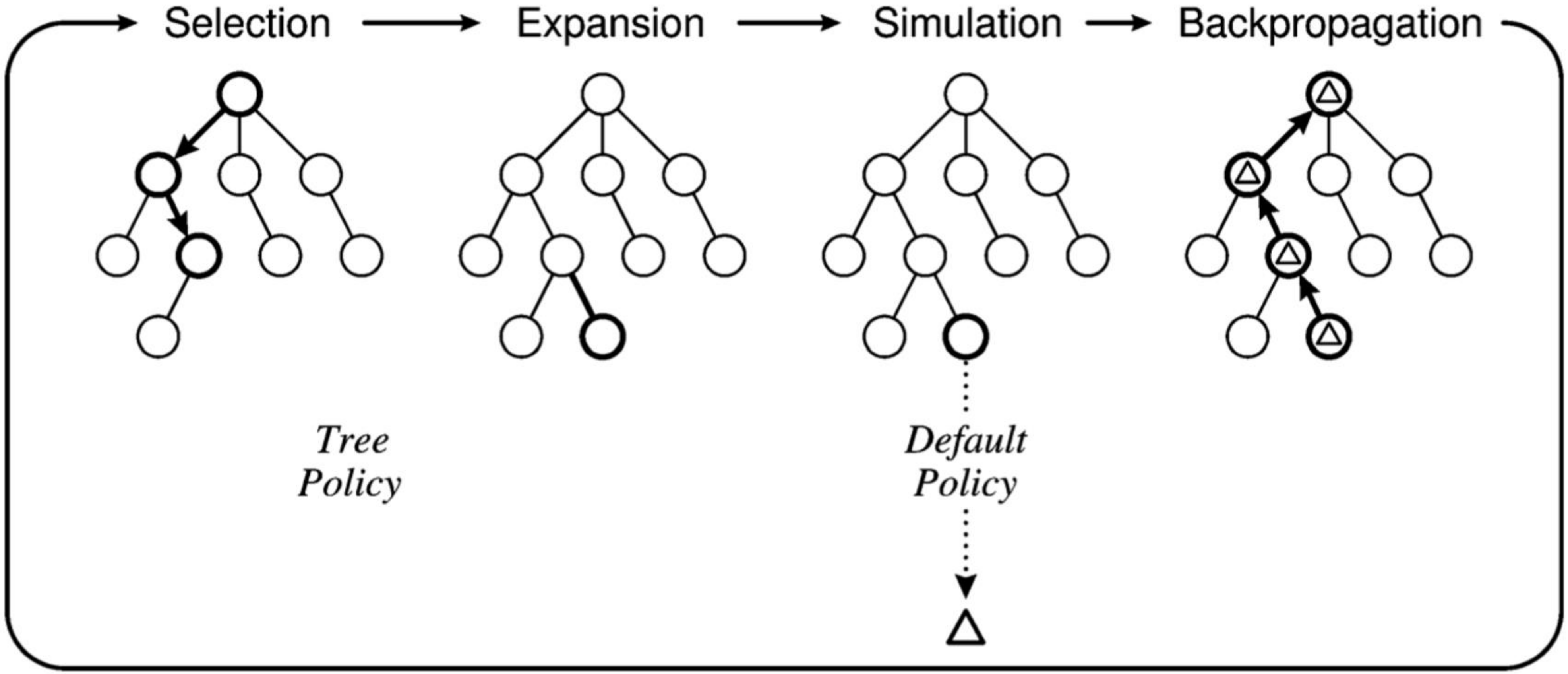
Updating $N_0(s,a) \leftarrow N_0(s,a) + 1$

and $\widehat{Q_0}(s,a) \leftarrow \dfrac{(N_0(s,a)-1)\widehat{Q_0}(s,a)+R(s,a,s')+\gamma\widehat{Q_1}(s',a')}{N_0(s,a)}$

Running average!

[1] https://www.cs.utexas.edu/~pstone/Papers/bib2html-links/ICML16-khandelwal.slides.pdf

MCTS Summary



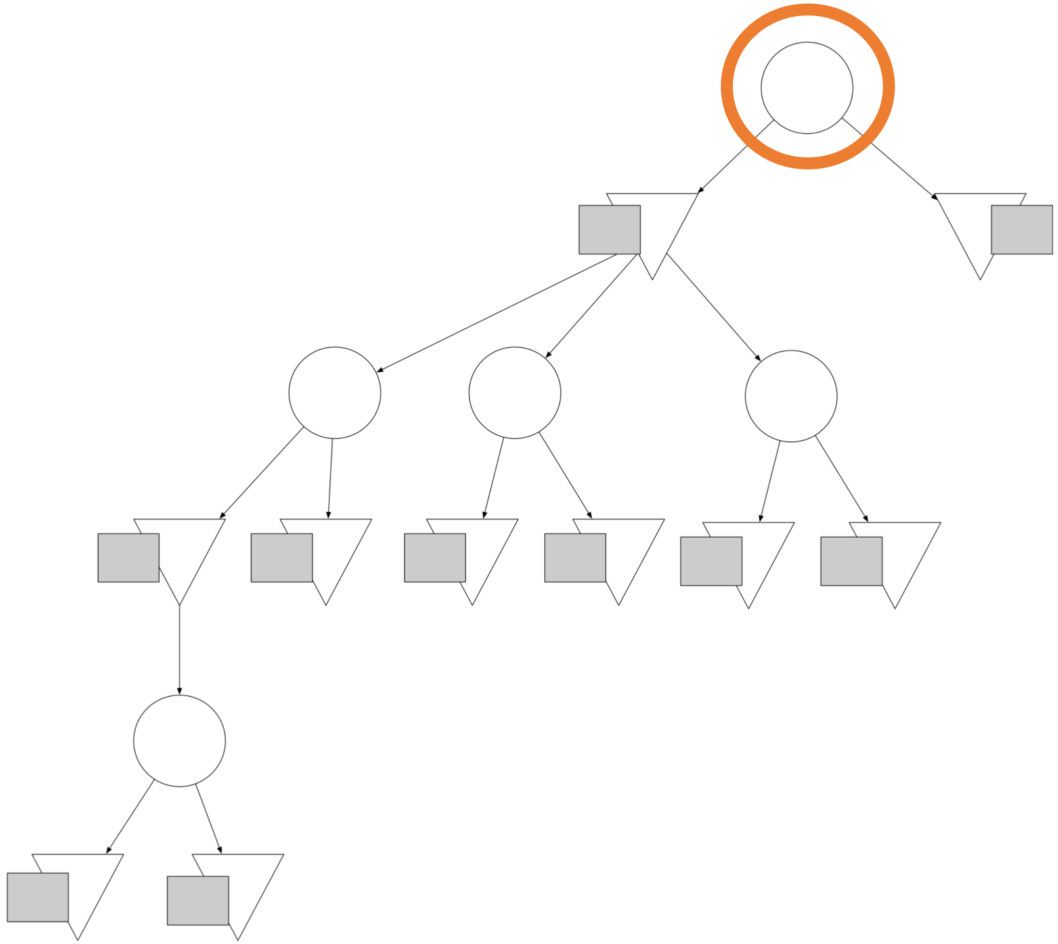
“A Survey of Monte Carlo Tree Search Methods.” Browne et al. (2012).

MCTS$(s_0, \mathcal{S}, \mathcal{A}, P, R, \gamma)$

`Q` $=$ `dict()` // Estimate for $Q_t(s, a)$
`N` $=$ `dict()` // Visitation counts $N_t(s, a)$
repeat until time runs out
    Simulate$(s_0, \mathcal{S}, \mathcal{A}, P, R, \gamma, \texttt{Q}, \texttt{N}, 0)$ // Updates `Q` and `N`
return $\text{argmax}_a\, \texttt{Q}(0, s_0, a)$

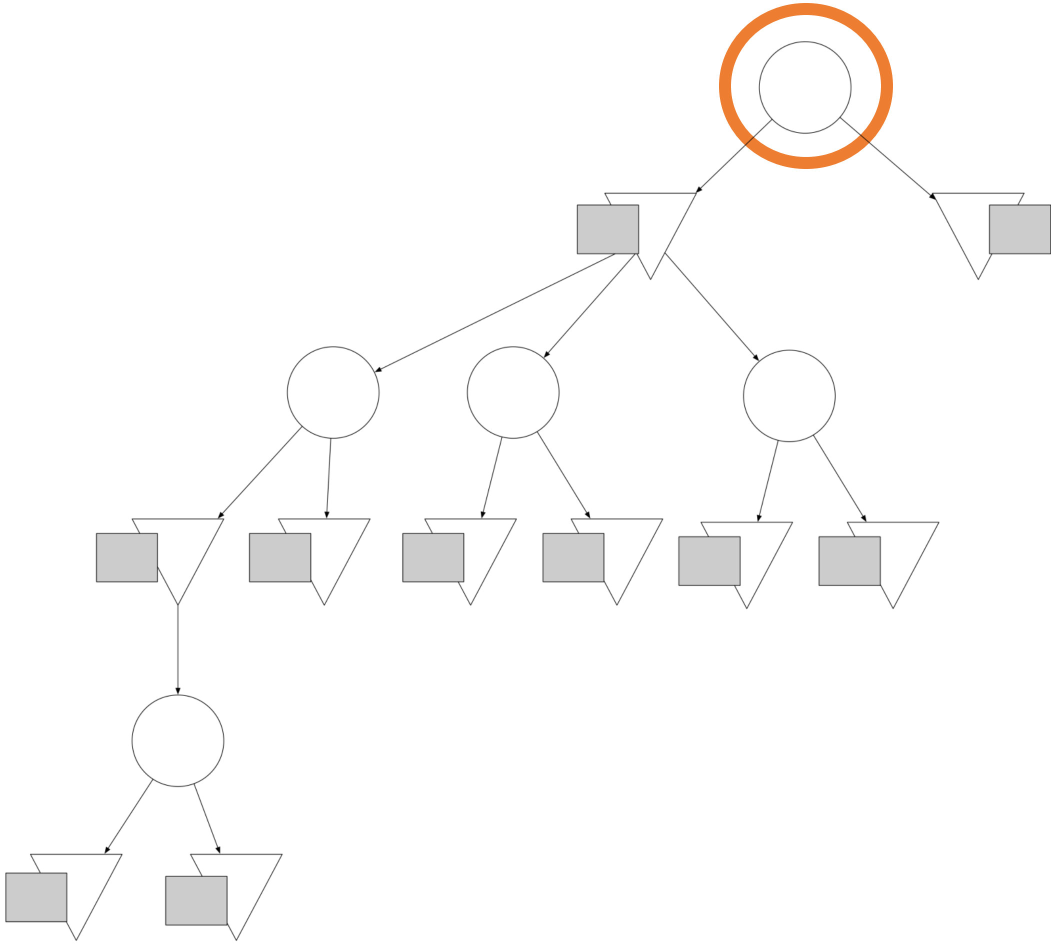
$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
    for $\mathtt{a} \in \mathcal{A}$
        $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
        $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



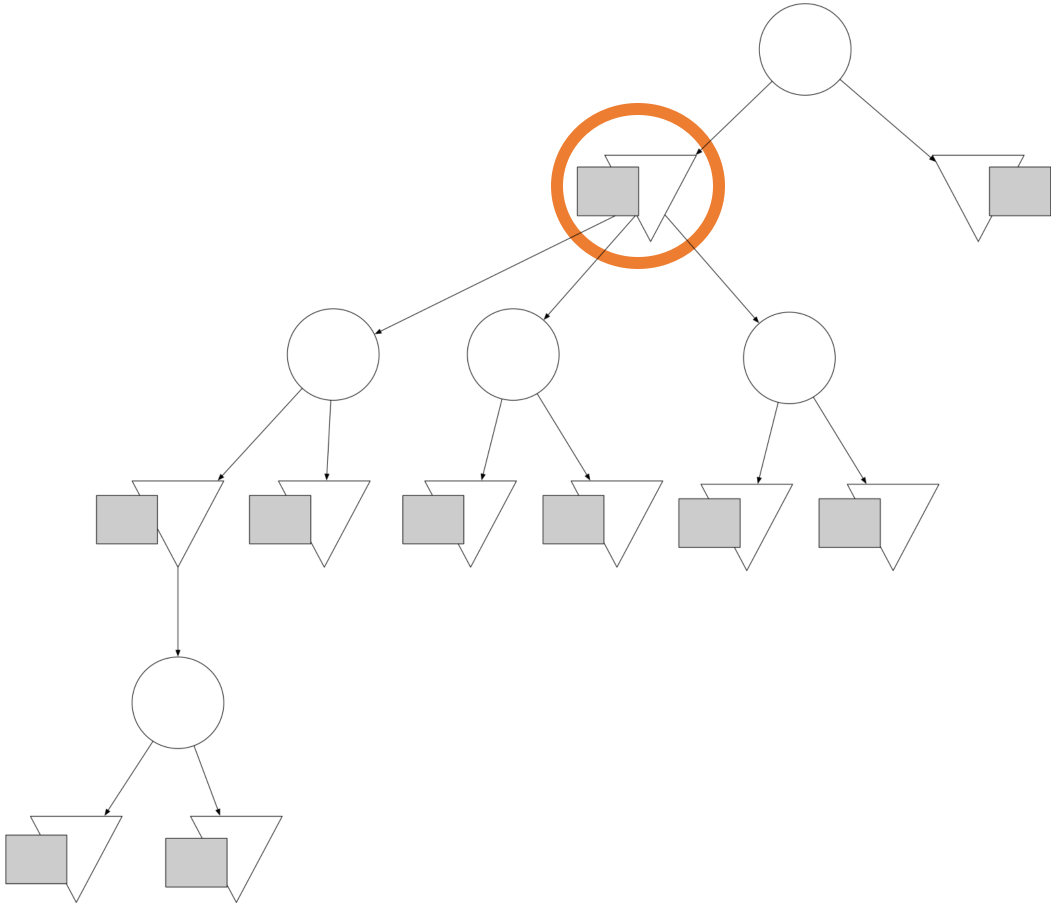
$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



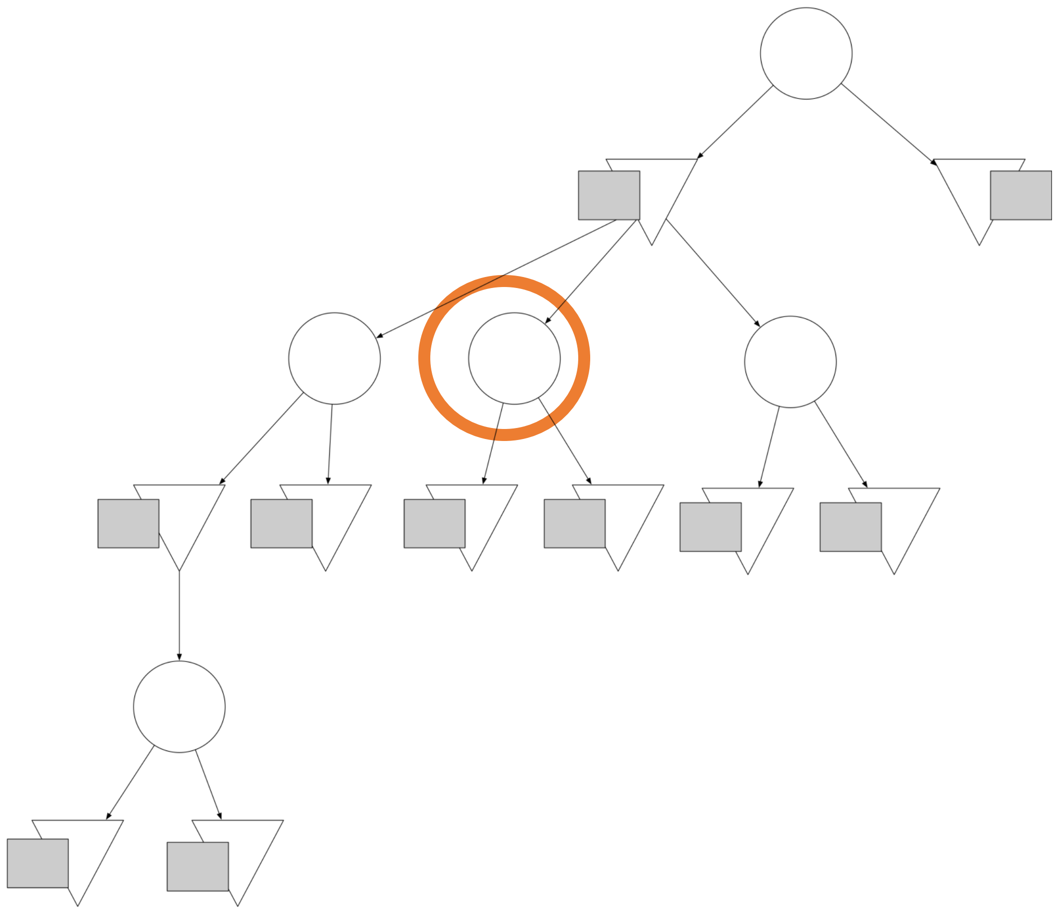
$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$ **False**
    for $\mathtt{a} \in \mathcal{A}$
        $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
        $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



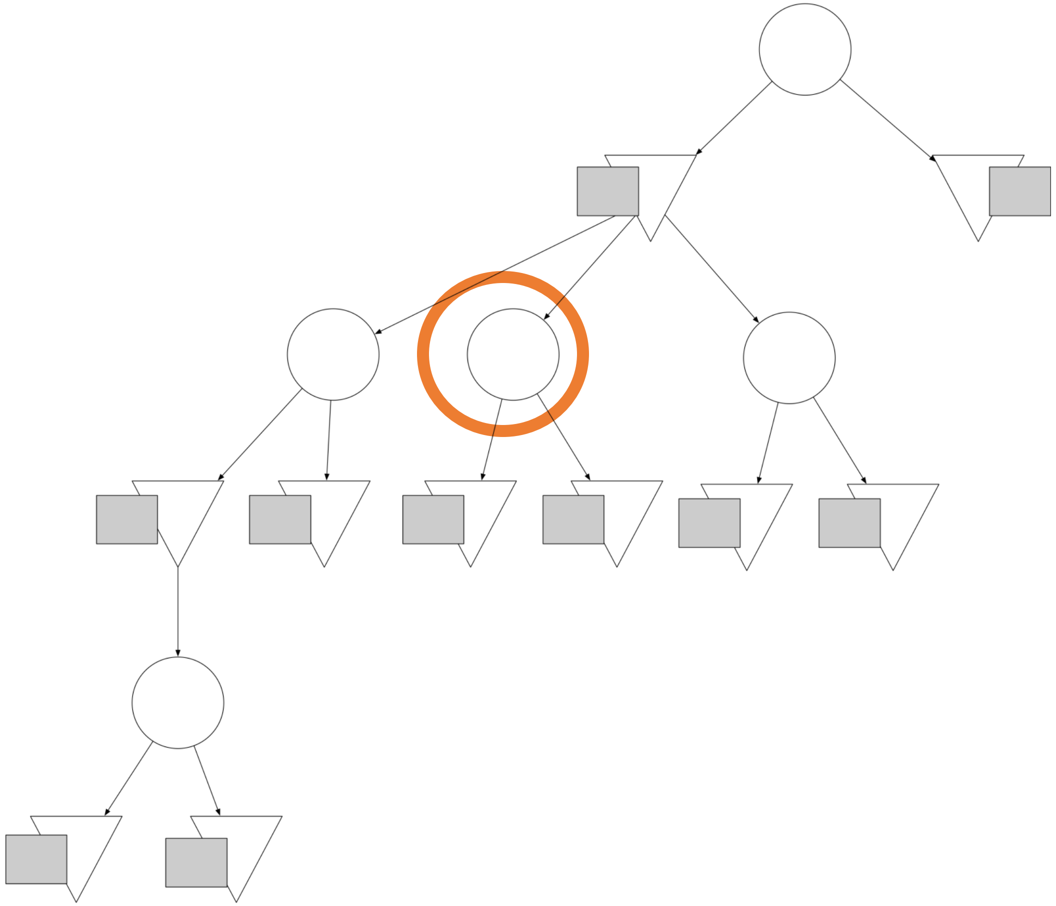
$\textsc{Simulate}(\texttt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \texttt{Q}, \texttt{N}, \texttt{t})$

// Base case: we've never visited this state at this depth before
if $(\texttt{t}, \texttt{s}, \texttt{a}) \notin \texttt{N}$ for an arbitrary $\texttt{a} \in \mathcal{A}$
    for $\texttt{a} \in \mathcal{A}$
        $\texttt{N}(\texttt{t}, \texttt{s}, \texttt{a}) = 0$
        $\texttt{Q}(\texttt{t}, \texttt{s}, \texttt{a}) = 0$
    return $\textsc{EstimateHeuristic}(\texttt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\boxed{\texttt{a} = \textsc{Explore}(\texttt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \texttt{Q}, \texttt{N}, \texttt{t})}$ // differs between MCTS algs
$\texttt{ns} \sim \mathrm{P}(\cdot \mid \texttt{s}, \texttt{a})$
$\texttt{qtsa} = R(\texttt{s}, \texttt{a}, \texttt{ns}) + \gamma \textsc{Simulate}(\texttt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \texttt{Q}, \texttt{N}, \texttt{t} + 1)$
$\texttt{N}(\texttt{t}, \texttt{s}, \texttt{a}) = \texttt{N}(\texttt{t}, \texttt{s}, \texttt{a}) + 1$
$\texttt{Q}(\texttt{t}, \texttt{s}, \texttt{a}) = \frac{(\texttt{N}(\texttt{t},\texttt{s},\texttt{a}) - 1)\texttt{Q}(\texttt{t},\texttt{s},\texttt{a}) + \texttt{qtsa}}{\texttt{N}(\texttt{t},\texttt{s},\texttt{a})}$
return $\texttt{Q}(\texttt{t}, \texttt{s}, \texttt{a})$



$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\boxed{\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})}$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before

if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$

 for $\mathtt{a} \in \mathcal{A}$

  $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$

  $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$

 return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts

$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs

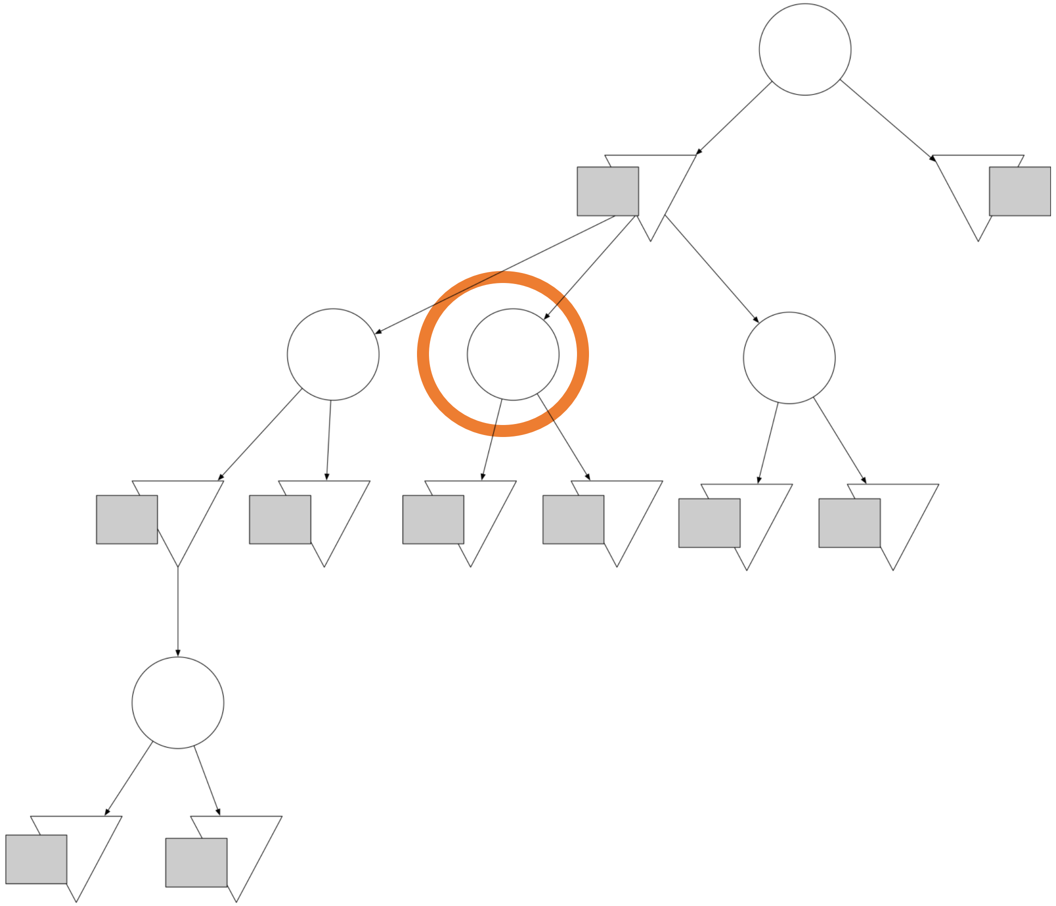
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$

$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \boxed{\textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t}+1)}$

$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$

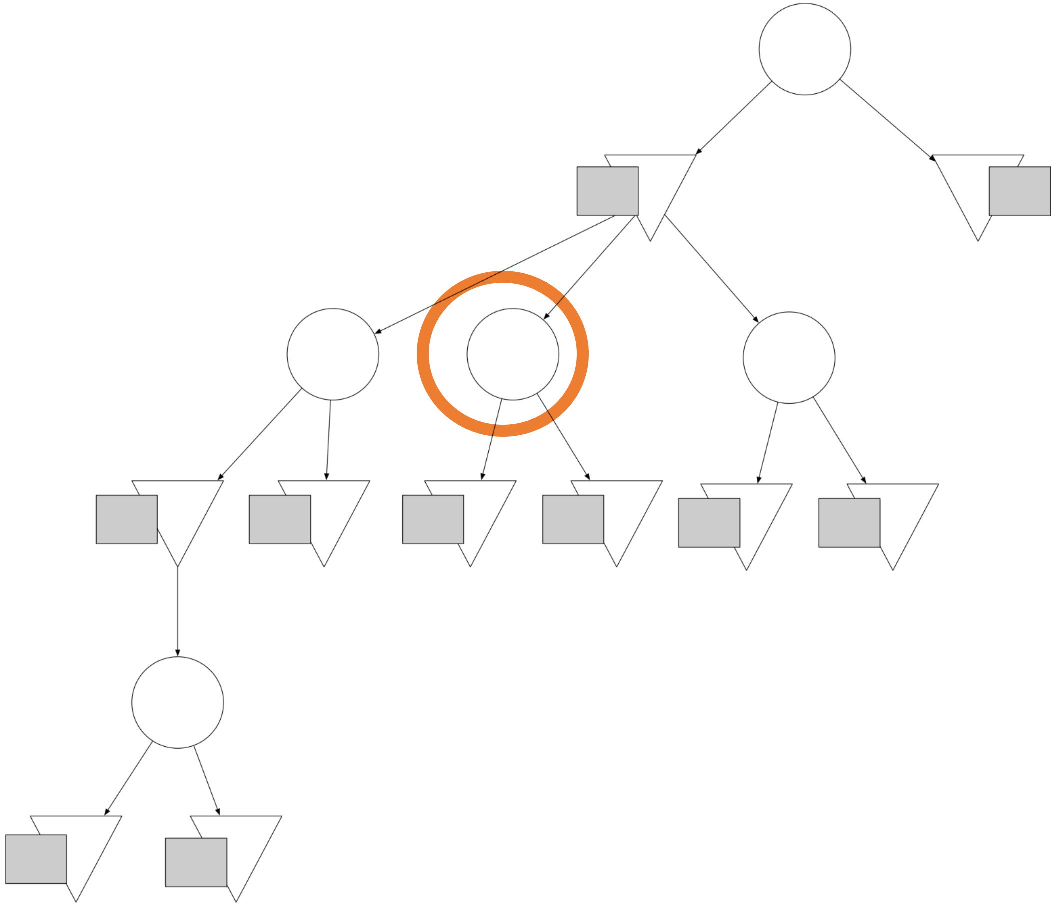
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})-1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a})+\mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$

return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



$\boxed{\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})}$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before

$\boxed{\textbf{if } (\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N} \text{ for an arbitrary } \mathtt{a} \in \mathcal{A}}$ **False**

    for $\mathtt{a} \in \mathcal{A}$

        $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$

        $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$

    return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts

$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs

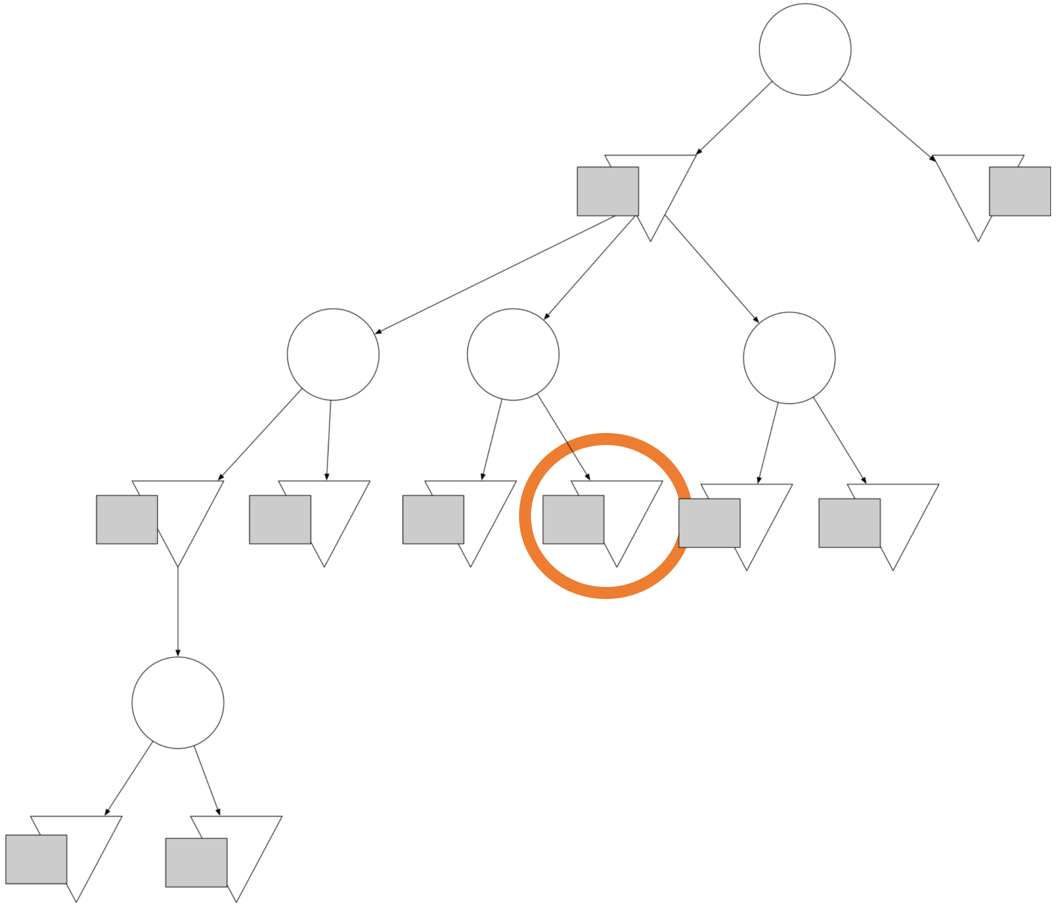
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$

$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$

$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$

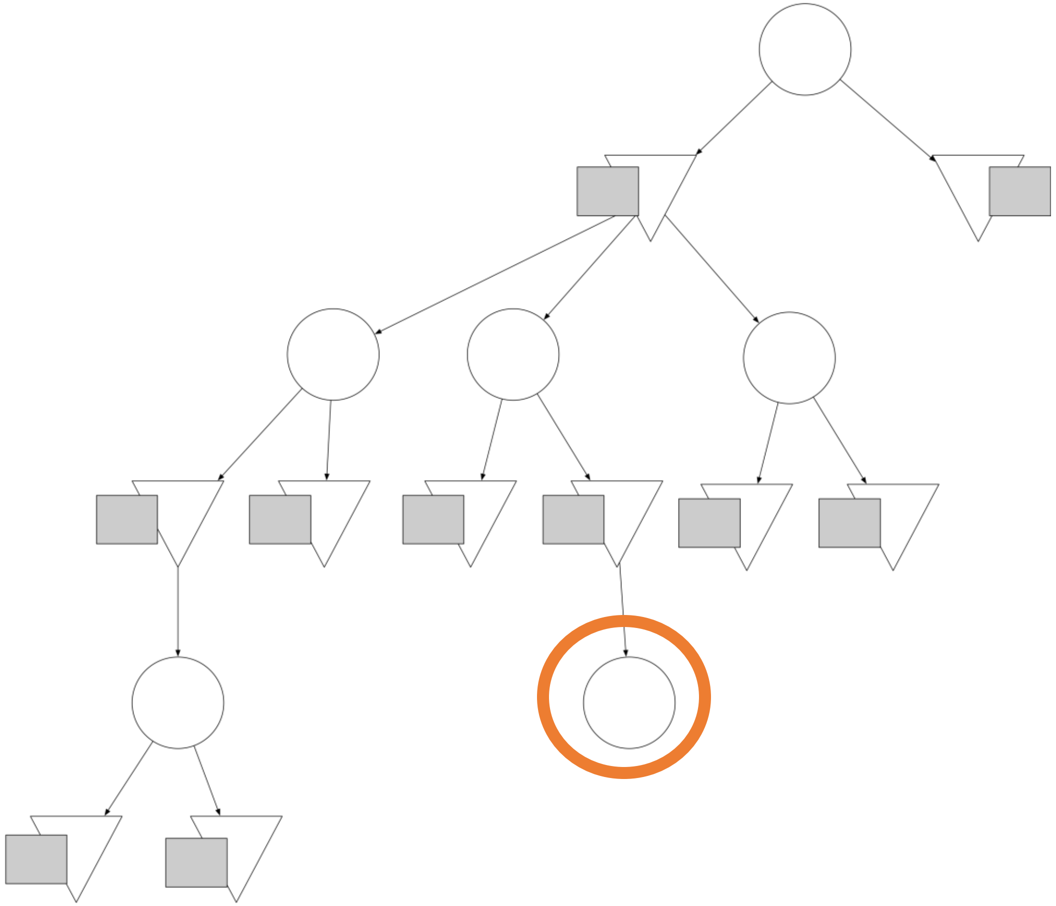
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$

return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



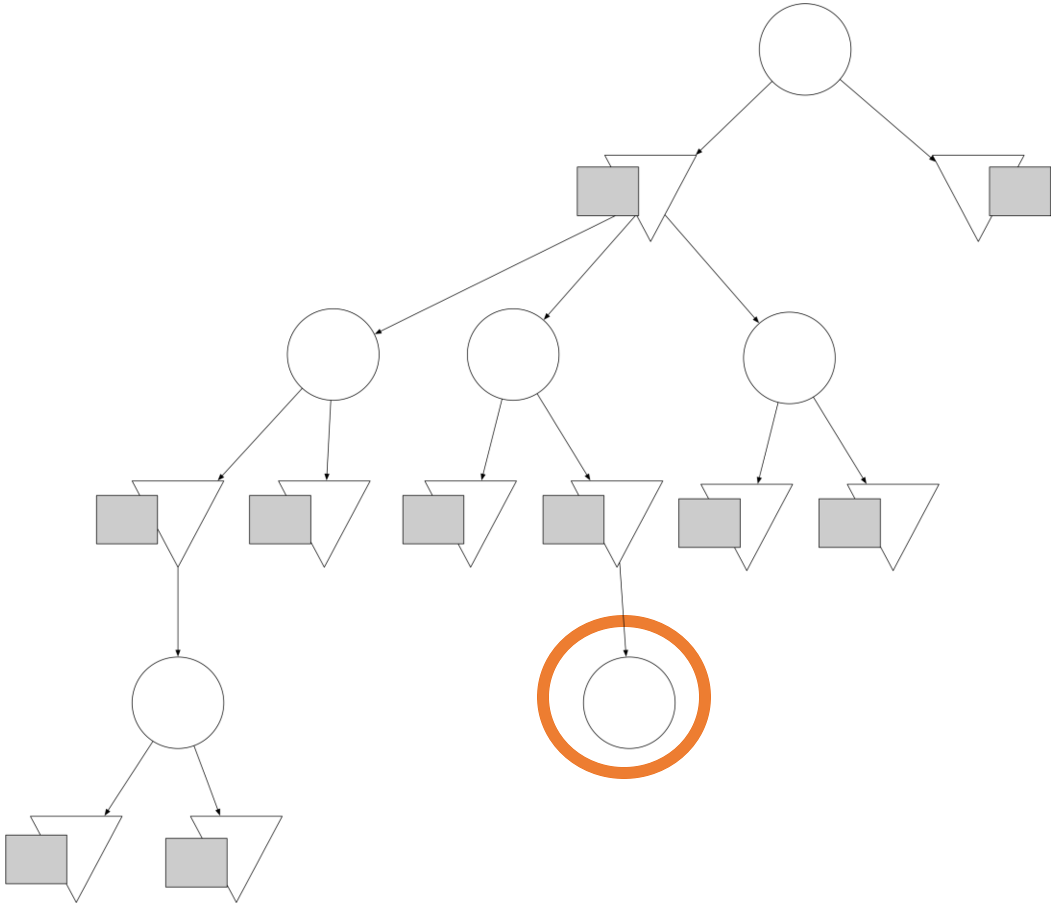
$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\boxed{\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})}$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



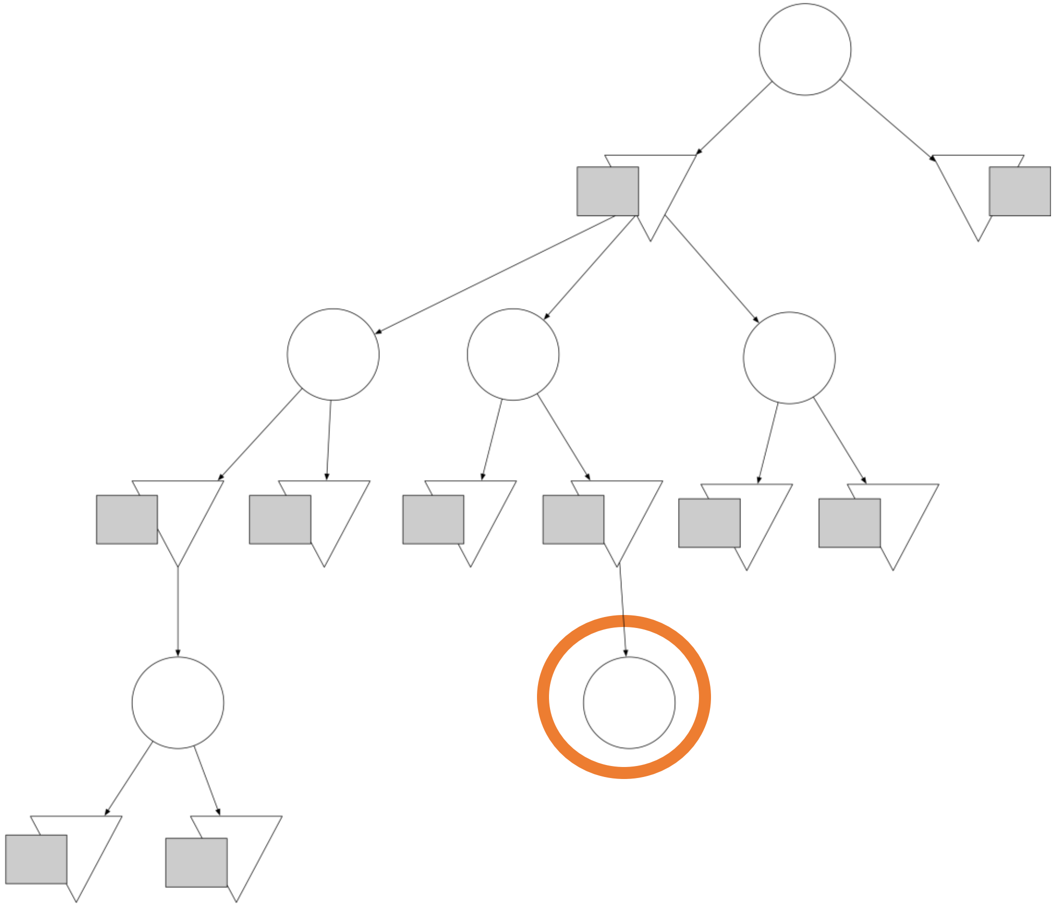
$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
    for $\mathtt{a} \in \mathcal{A}$
        $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
        $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\boxed{\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})}$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



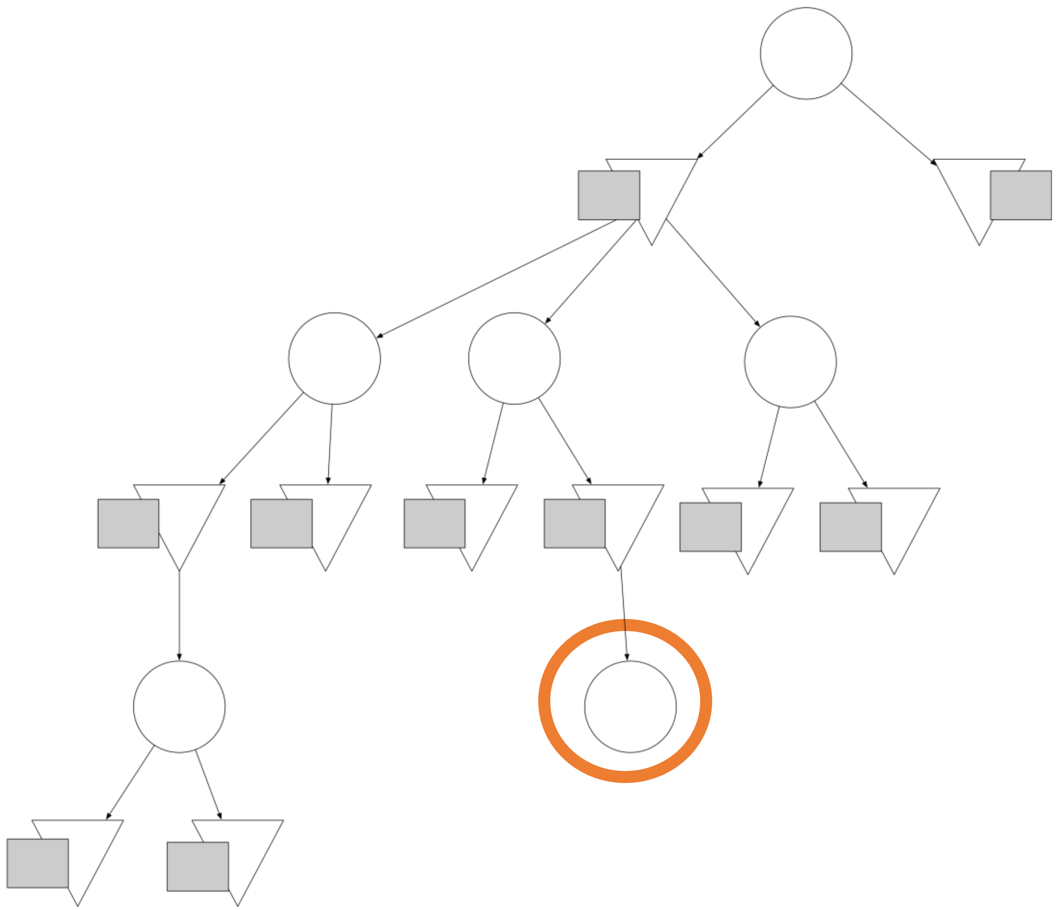
$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \boxed{\textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t}+1)}$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})-1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a})+\mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$

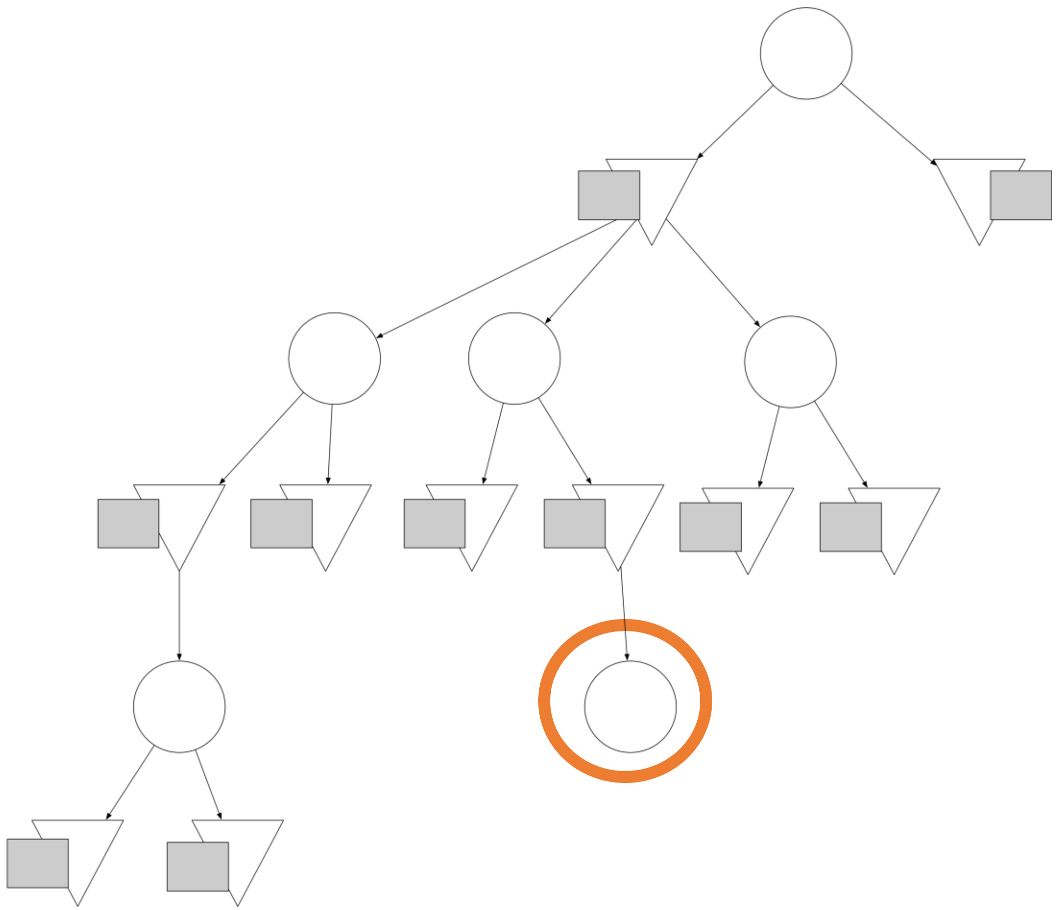


$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \boxed{\textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t}+1)}$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})-1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a})+\mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$

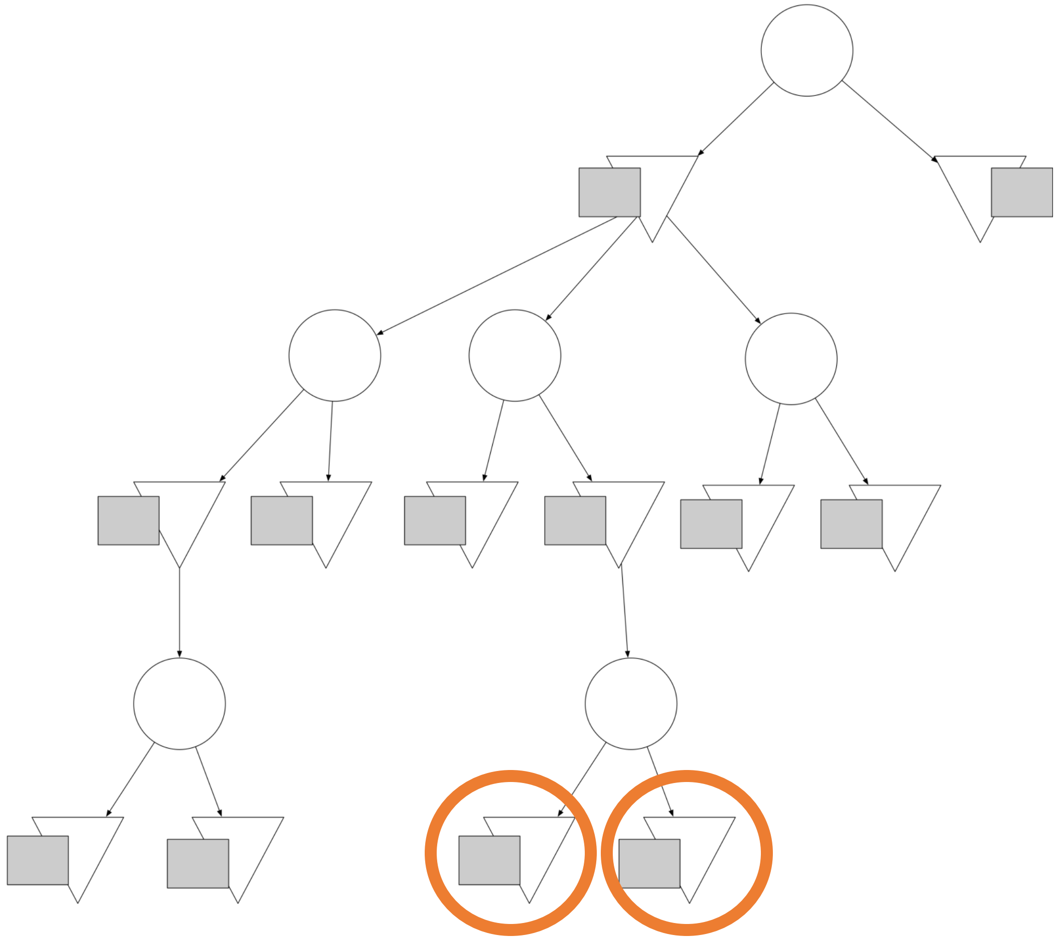


```
SIMULATE(s, 𝒮, 𝒜, P, R, γ, Q, N, t)
// Base case: we've never visited this state at this depth before
if (t, s, a) ∉ N for an arbitrary a ∈ 𝒜
    for a ∈ 𝒜
        N(t, s, a) = 0
        Q(t, s, a) = 0
    return ESTIMATEHEURISTIC(s, 𝒮, 𝒜, P, R, γ) // Run random rollouts
a = EXPLORE(s, 𝒮, 𝒜, P, R, γ, Q, N, t) // differs between MCTS algs
ns ~ P(· | s, a)
qtsa = R(s, a, ns) + γSIMULATE(ns, 𝒮, 𝒜, P, R, γ, Q, N, t + 1)
N(t, s, a) = N(t, s, a) + 1
Q(t, s, a) = ((N(t,s,a) − 1)Q(t,s,a) + qtsa) / N(t,s,a)
return Q(t, s, a)
```



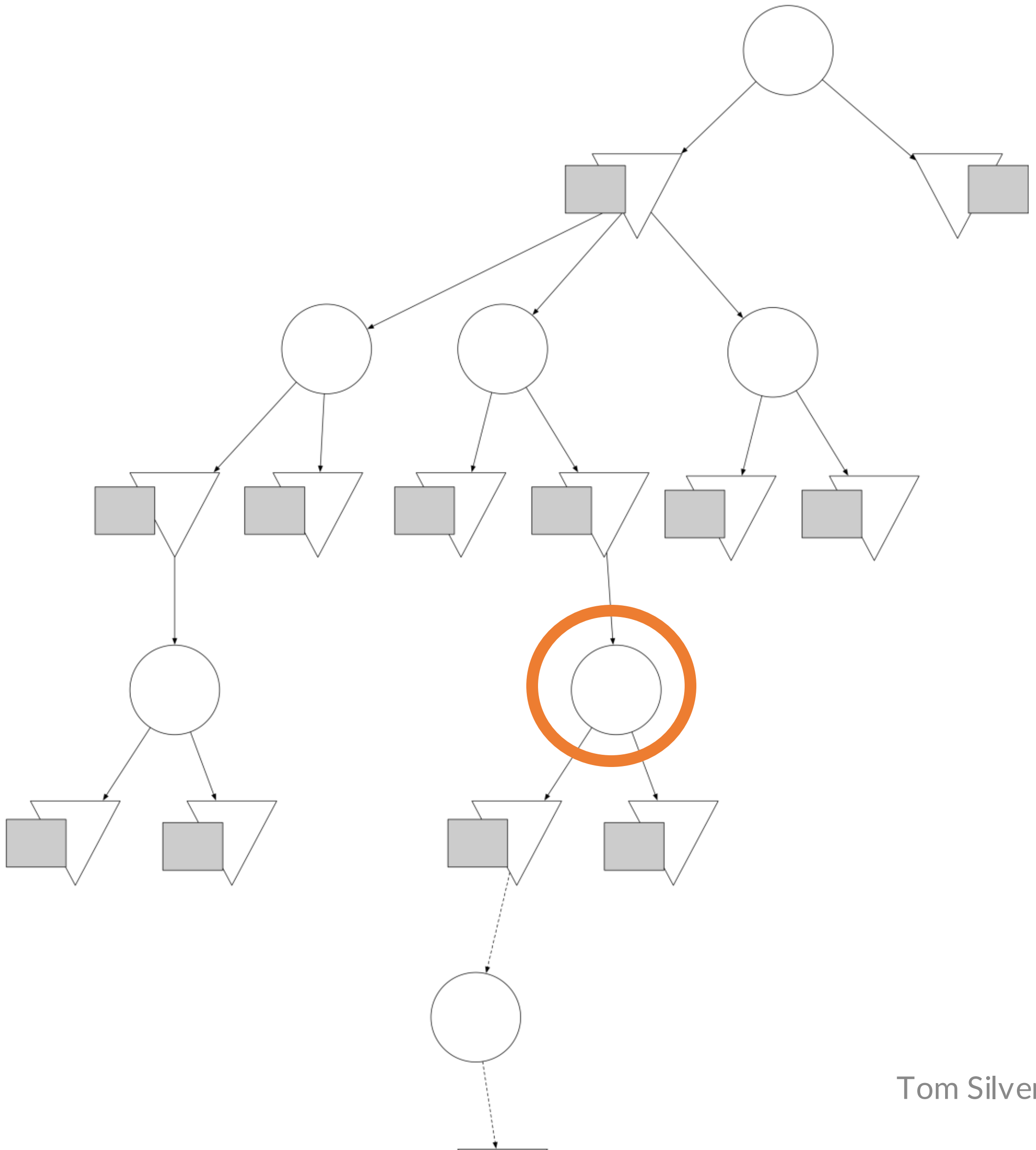
SIMULATE$(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$ **True**
    for $\mathtt{a} \in \mathcal{A}$
        $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
        $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    return ESTIMATEHEURISTIC$(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} =$ EXPLORE$(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma$SIMULATE$(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t}+1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})-1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a})+\mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



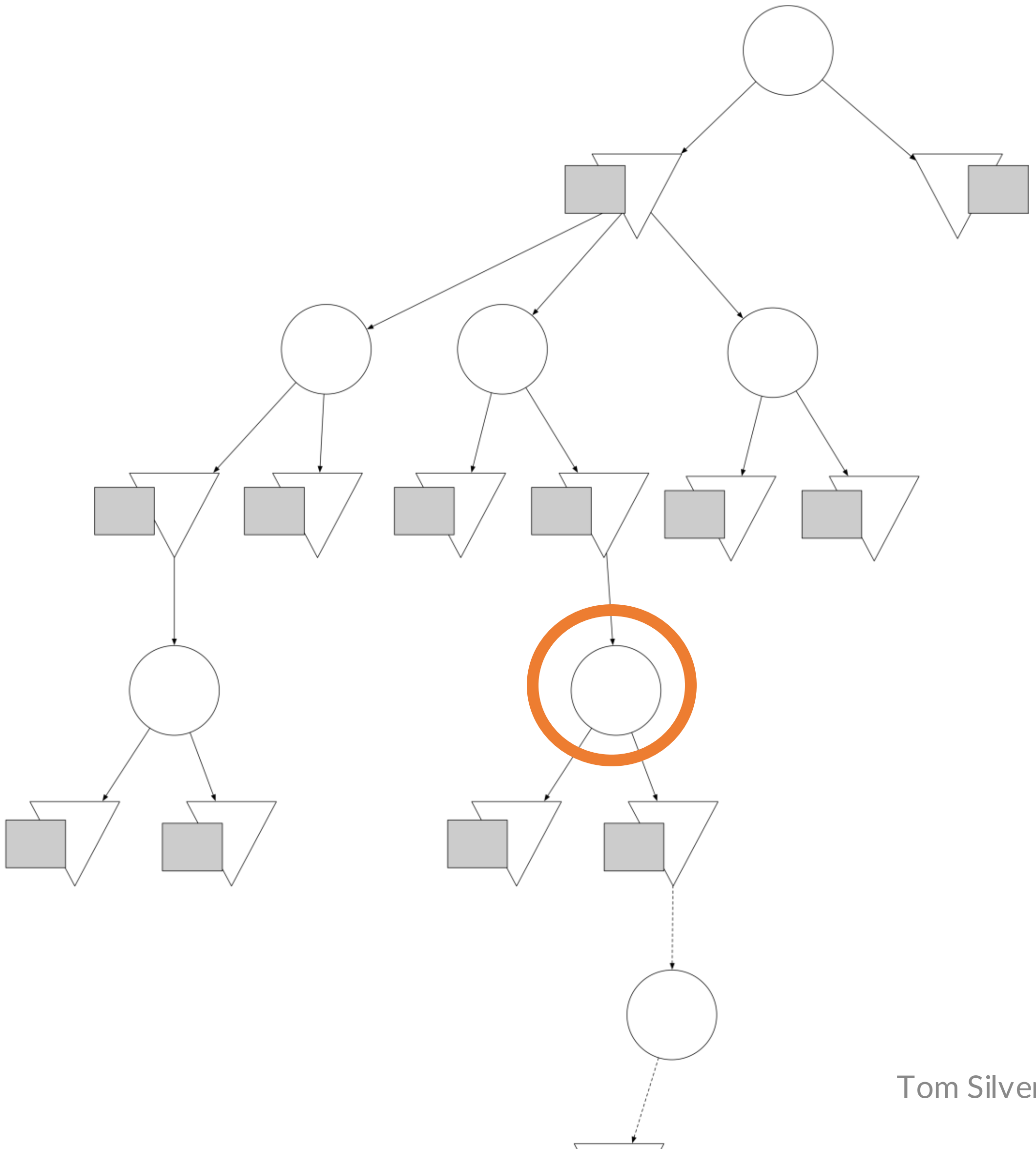
$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t}+1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})-1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a})+\mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



SIMULATE$(\texttt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \texttt{Q}, \texttt{N}, \texttt{t})$

// Base case: we've never visited this state at this depth before
if $(\texttt{t}, \texttt{s}, \texttt{a}) \notin \texttt{N}$ for an arbitrary $\texttt{a} \in \mathcal{A}$
    for $\texttt{a} \in \mathcal{A}$
        $\texttt{N}(\texttt{t}, \texttt{s}, \texttt{a}) = 0$
        $\texttt{Q}(\texttt{t}, \texttt{s}, \texttt{a}) = 0$
    return $\boxed{\text{ESTIMATEHEURISTIC}(\texttt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)}$ // Run random rollouts
$\texttt{a} = \text{EXPLORE}(\texttt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \texttt{Q}, \texttt{N}, \texttt{t})$ // differs between MCTS algs
$\texttt{ns} \sim \texttt{P}(\cdot \mid \texttt{s}, \texttt{a})$
$\texttt{qtsa} = R(\texttt{s}, \texttt{a}, \texttt{ns}) + \gamma\text{SIMULATE}(\texttt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \texttt{Q}, \texttt{N}, \texttt{t} + 1)$
$\texttt{N}(\texttt{t}, \texttt{s}, \texttt{a}) = \texttt{N}(\texttt{t}, \texttt{s}, \texttt{a}) + 1$
$\texttt{Q}(\texttt{t}, \texttt{s}, \texttt{a}) = \frac{(\texttt{N}(\texttt{t},\texttt{s},\texttt{a})-1)\texttt{Q}(\texttt{t},\texttt{s},\texttt{a})+\texttt{qtsa}}{\texttt{N}(\texttt{t},\texttt{s},\texttt{a})}$
return $\texttt{Q}(\texttt{t}, \texttt{s}, \texttt{a})$



$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before

if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$

  for $\mathtt{a} \in \mathcal{A}$

    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$

    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$

  return $\boxed{\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)}$ // Run random rollouts

$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs

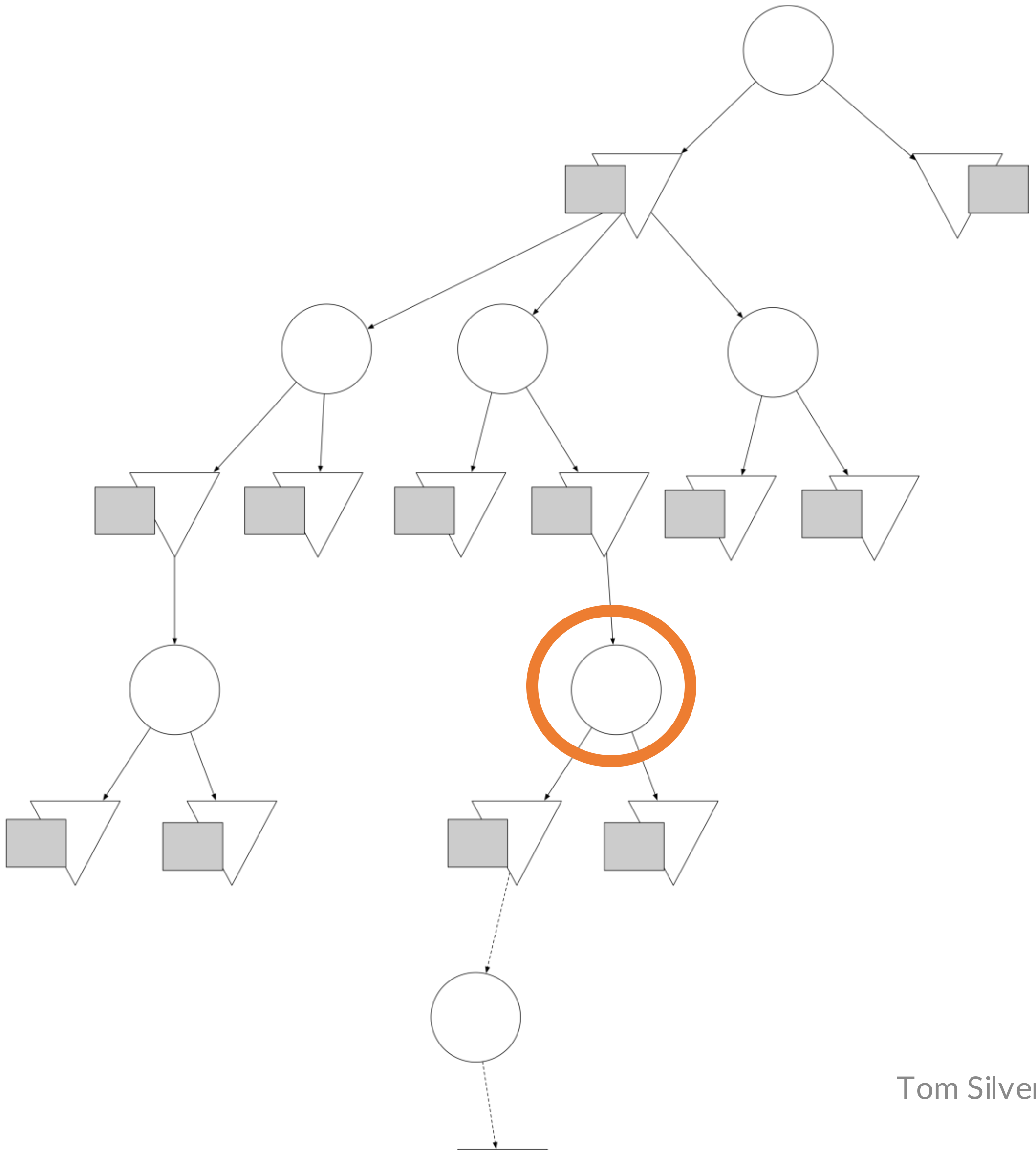
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$

$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$

$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$

$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$

return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before

if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$

    for $\mathtt{a} \in \mathcal{A}$

        $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$

        $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$

    return $\boxed{\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)}$ // Run random rollouts

$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs

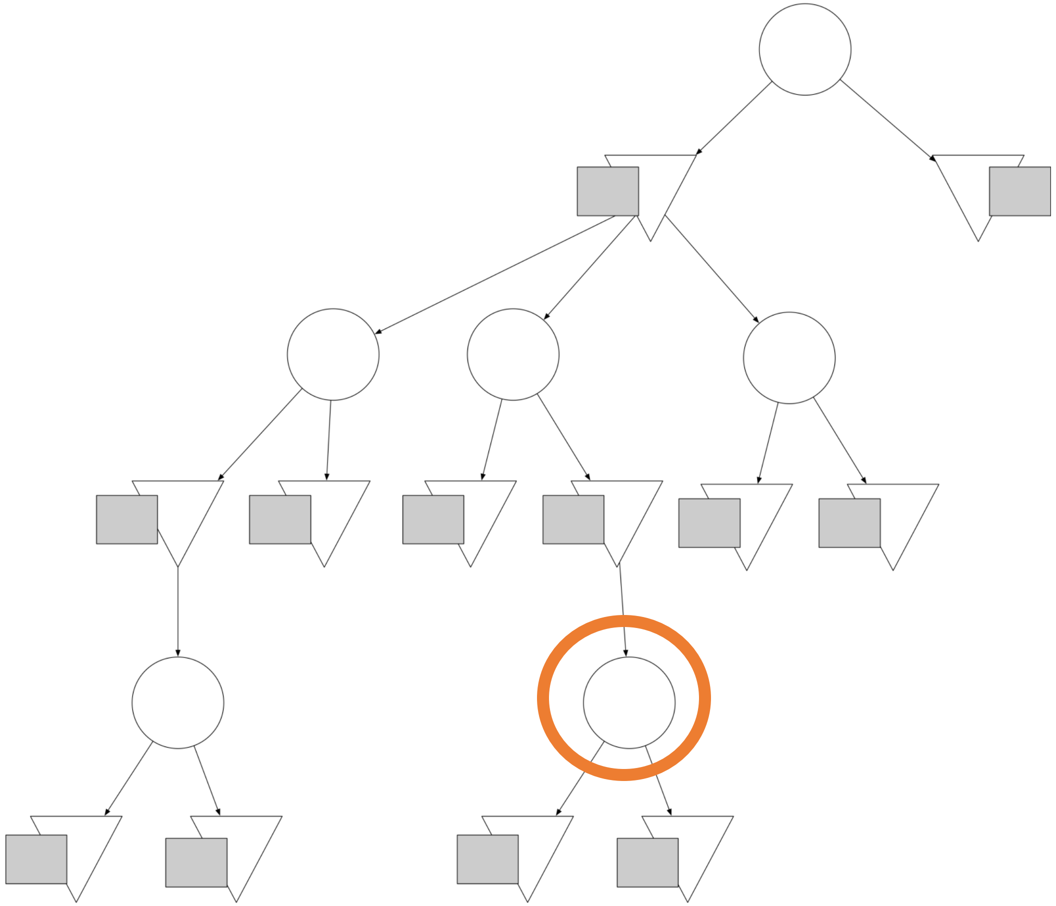
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$

$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$

$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$

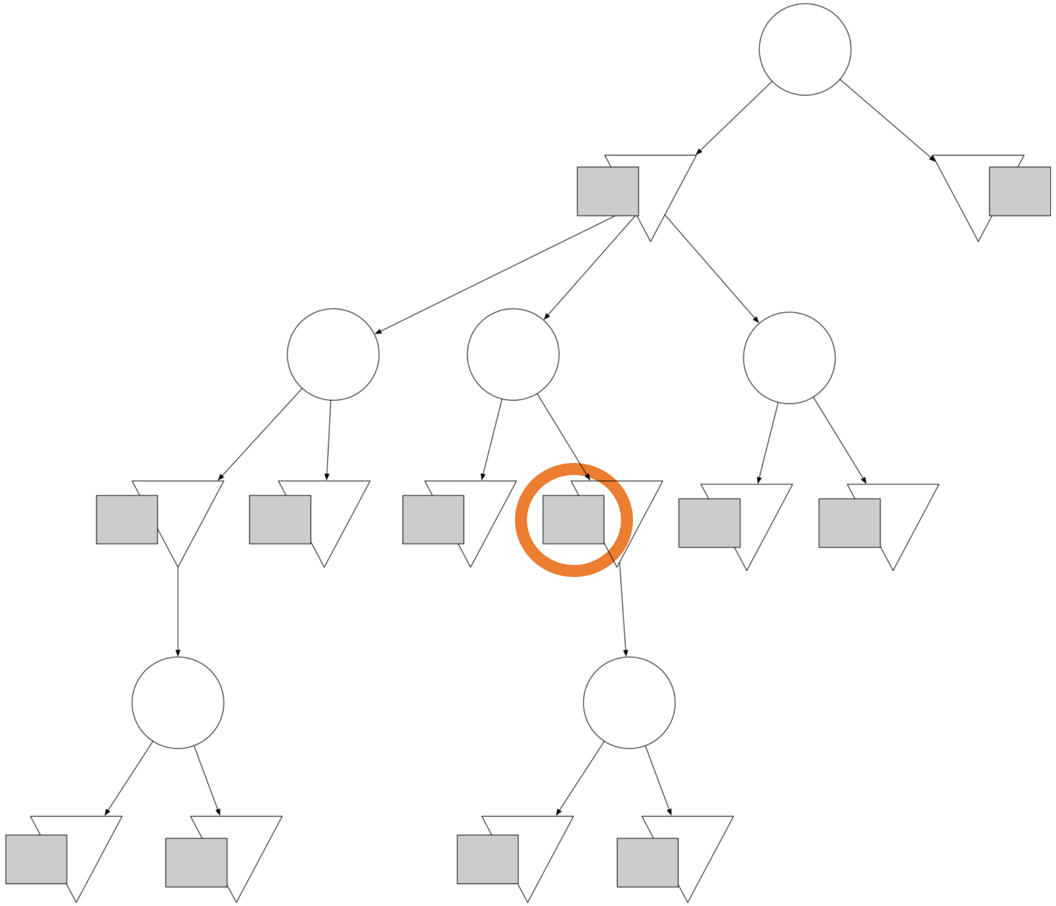
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$

return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



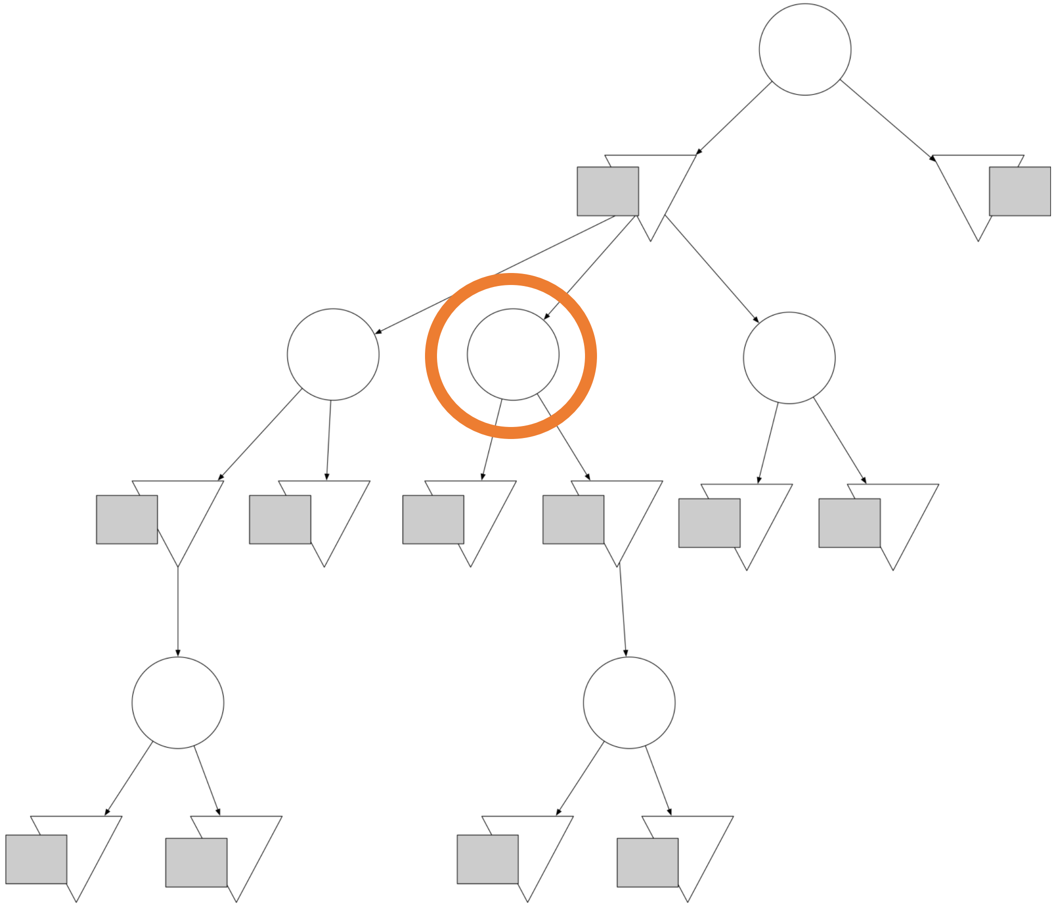
$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \boxed{\textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t}+1)}$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})-1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a})+\mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



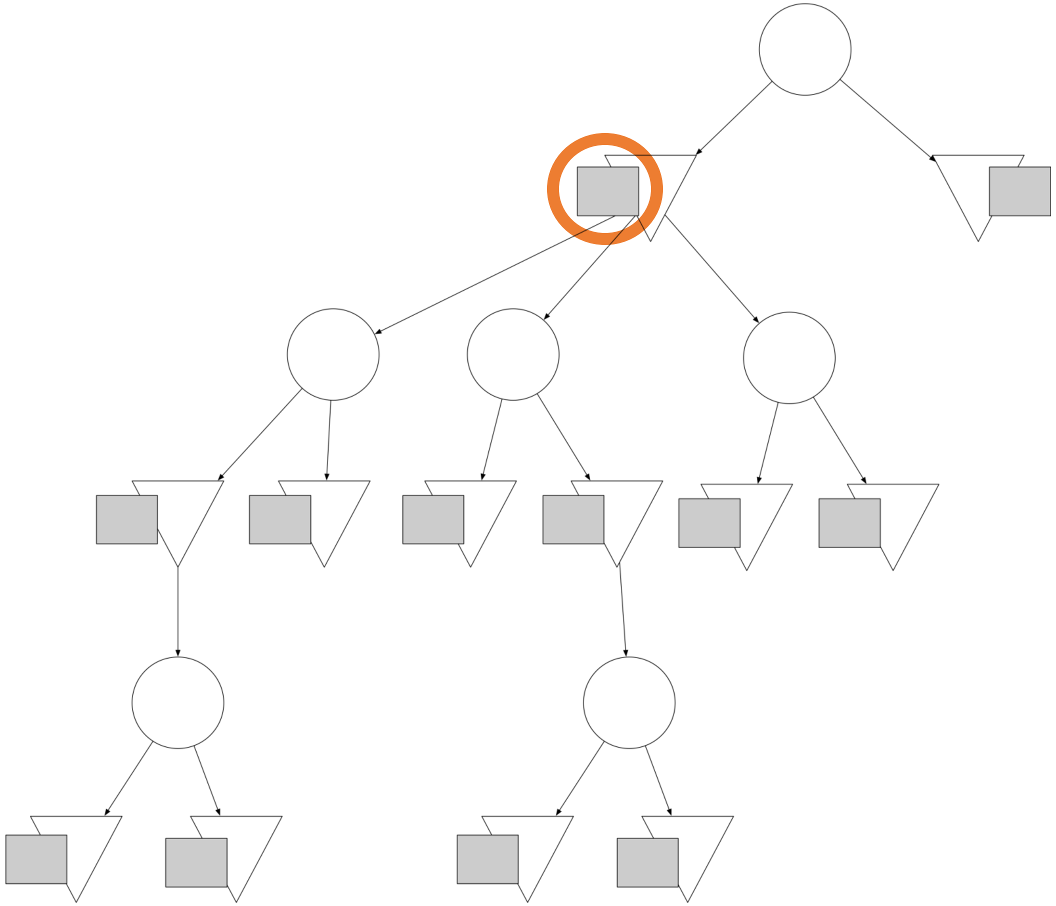
$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
    for $\mathtt{a} \in \mathcal{A}$
        $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
        $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



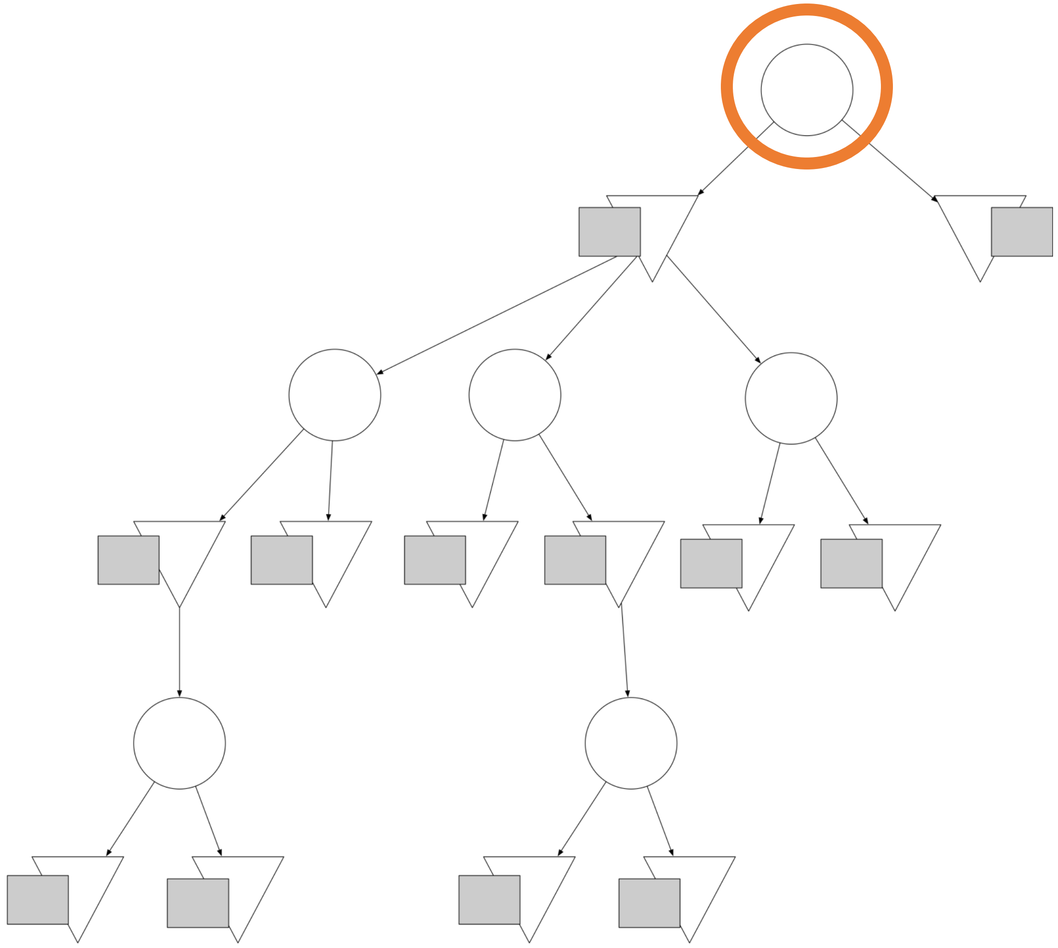
$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \boxed{\textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)}$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$



$\textsc{Simulate}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$

// Base case: we've never visited this state at this depth before
if $(\mathtt{t}, \mathtt{s}, \mathtt{a}) \notin \mathtt{N}$ for an arbitrary $\mathtt{a} \in \mathcal{A}$
  for $\mathtt{a} \in \mathcal{A}$
    $\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
    $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = 0$
  return $\textsc{EstimateHeuristic}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma)$ // Run random rollouts
$\mathtt{a} = \textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t})$ // differs between MCTS algs
$\mathtt{ns} \sim \mathrm{P}(\cdot \mid \mathtt{s}, \mathtt{a})$
$\mathtt{qtsa} = R(\mathtt{s}, \mathtt{a}, \mathtt{ns}) + \gamma \textsc{Simulate}(\mathtt{ns}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N}, \mathtt{t} + 1)$
$\mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \mathtt{N}(\mathtt{t}, \mathtt{s}, \mathtt{a}) + 1$
$\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a}) = \frac{(\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a}) - 1)\mathtt{Q}(\mathtt{t},\mathtt{s},\mathtt{a}) + \mathtt{qtsa}}{\mathtt{N}(\mathtt{t},\mathtt{s},\mathtt{a})}$
return $\mathtt{Q}(\mathtt{t}, \mathtt{s}, \mathtt{a})$

UCT: MCTS + UCB

- Probably the most popular algorithm in the MCTS family is **Upper Confidence Trees (UCT)**.
- UCT uses the exploration bonus from UCB to select actions.

$\textsc{Explore}(\mathtt{s}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathtt{Q}, \mathtt{N})$

1. $\mathtt{Ns} = \sum_{a \in \mathcal{A}} \mathtt{N}(\mathtt{s}, \mathtt{a})$
2. // c is the hyperparameter discussed in UCB slide
3. **return** $\operatorname{argmax}_{\mathtt{a} \in \mathcal{A}} \mathtt{Q}(\mathtt{s}, \mathtt{a}) + c\sqrt{\frac{\log \mathtt{Ns}}{\mathtt{N}(\mathtt{s},\mathtt{a})}}$

Summary

- Sparse sampling: expectimax search, but instead of full Bellman backups, use sampling to approximate
- Multi-armed bandits: select actions to minimize regret
- Monte Carlo Tree Search: sparse sampling + MAB exploration techniques + rollouts to estimate heuristics