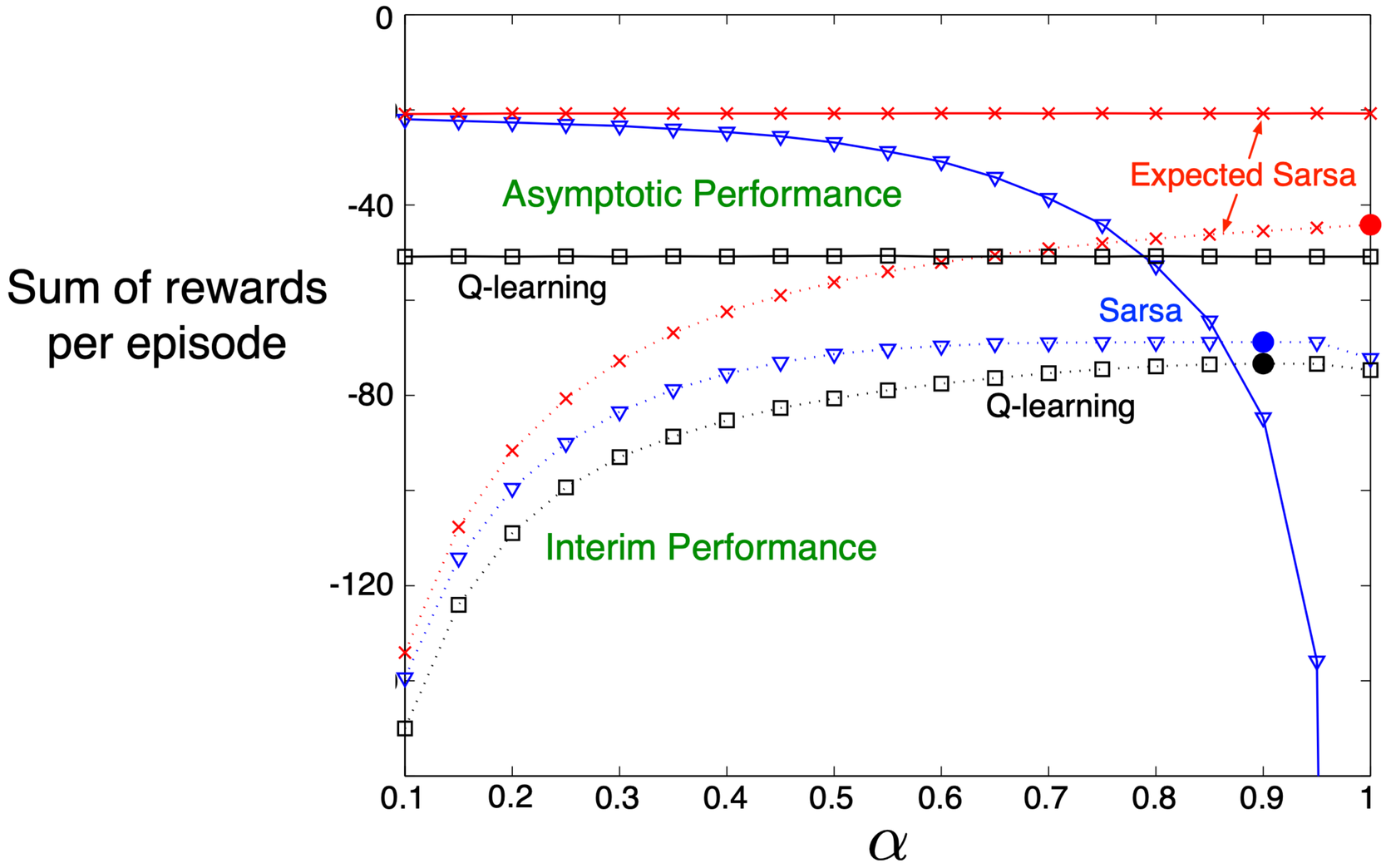
Choosing an RL algorithm!

- Ask yourself the following questions:
  - Do I have access to the model: p(s',r|s,a)?
  - Is changing the policy during an episode important for good performance?
    - When would this not be the case….
  - Will my initial estimate of the value function, Q_0, be really terrible?
  - Do I need the optimal policy or near-optimal good enough?
  - *Are we measuring performance online or offline?* <- *lets come back to this one*

How you implement the algorithm is often the most important choice

- Let’s say we decide we want to use Expected Sarsa
- How should we:
  - Decide on the target policy? (Greedy, E-Greedy)
  - Decide on the behavior policy?
  - Epsilon (maybe we have one for behavior and one for target) … maybe we use a schedule?
  - How do we initialize Q_0?
  - How do we set alpha…or change it with time?
- Do we care about the rewards the agent accumulates during learning, or just the final policy?

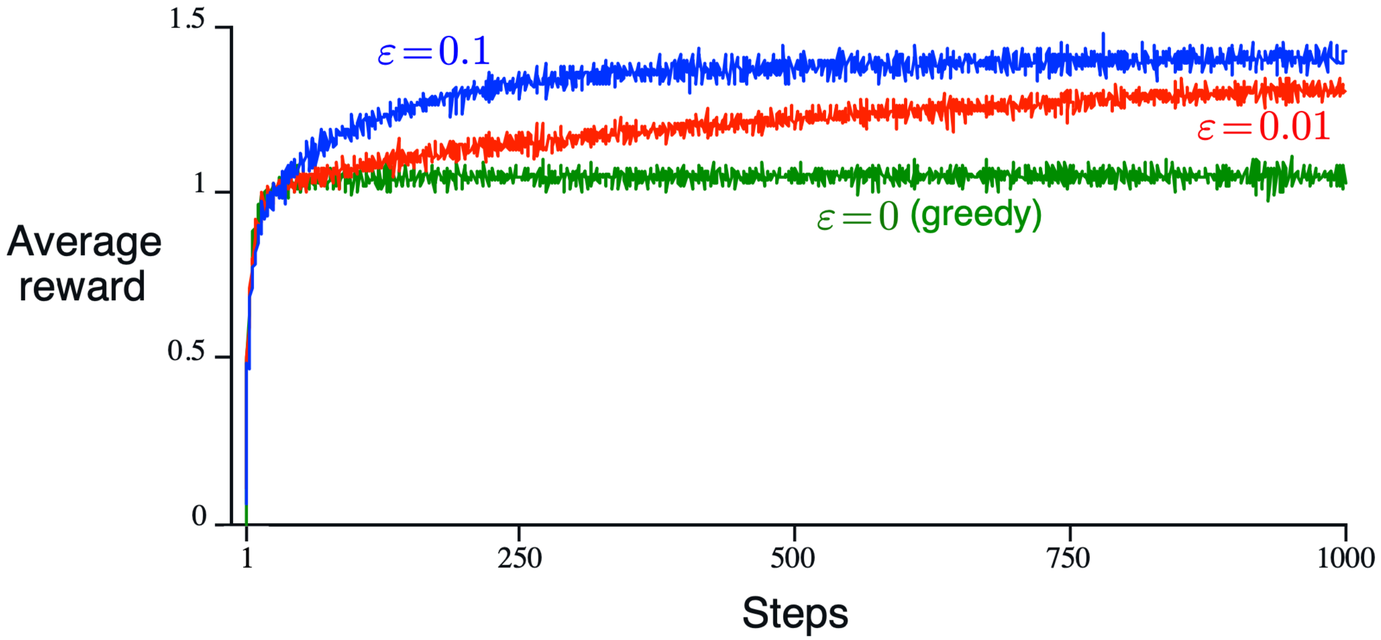
How you implement the algorithm is often the most important choice



- Each point on this plot represents a different choice of: <target policy, alpha, performance_measure>

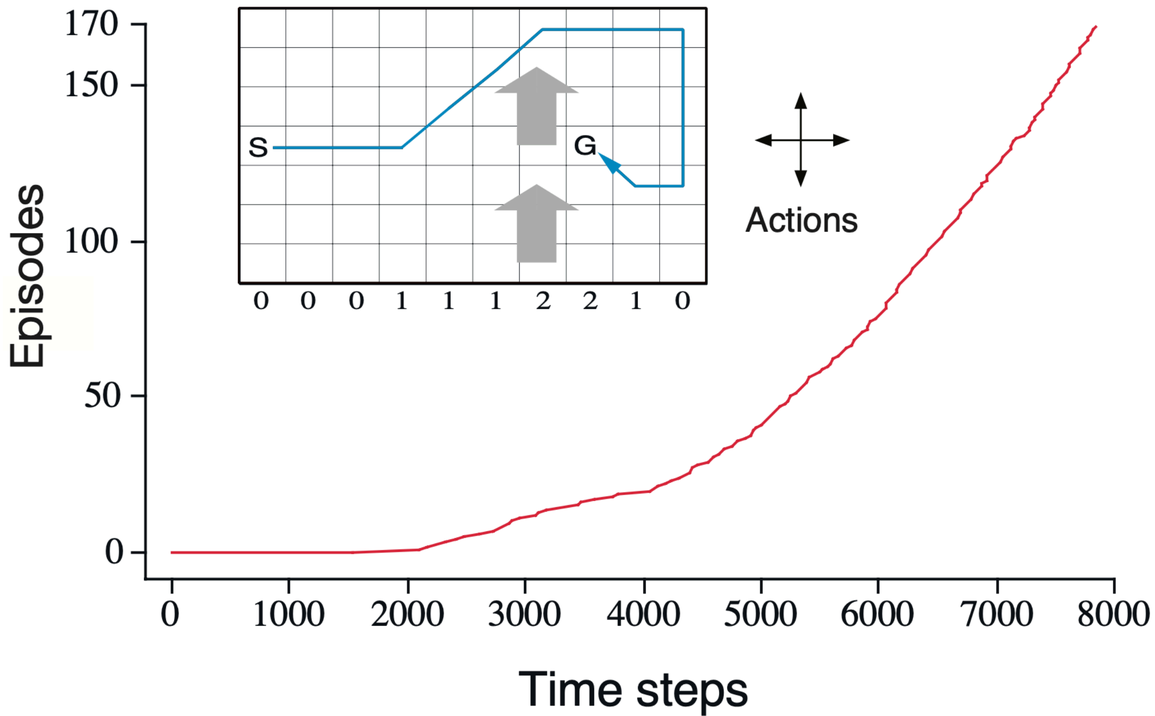
Online vs Offline performance

- Online is easy! It is what we have been looking at all along.
  - We measure performance as the agent interacts and learns in the environment
  - **In bandits we measured the reward on every time step**



Online vs Offline performance

- Online is easy! It is what we have been looking at all along.
  - We measure performance as the agent interacts and learns in the environment
  - **With Sarsa we counted (accumulated) the number of episodes complete over time, while learning**



Q-learning learns the optimal policy, but takes actions according to an e-greedy policy

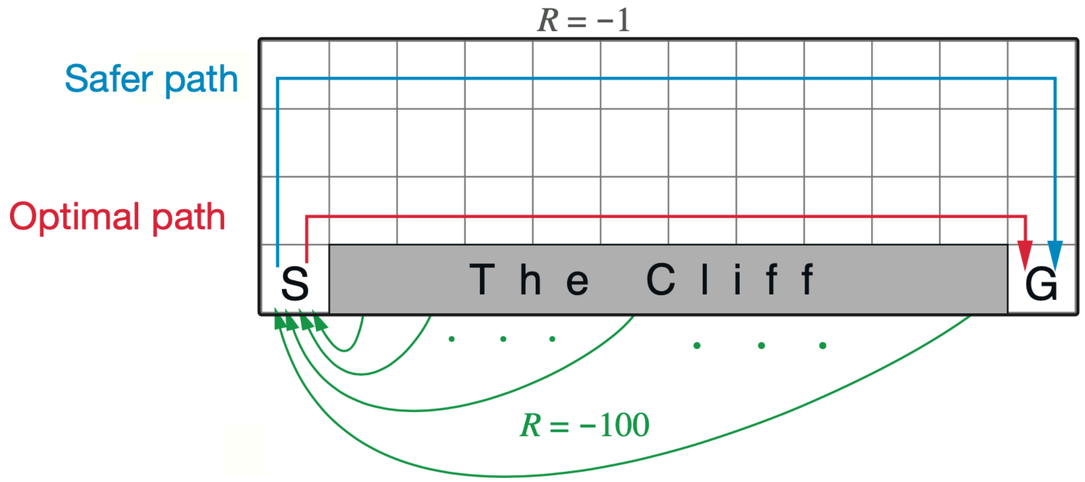
- On every time-step Q-learning either takes the best action according to Q, or it explores with probability epsilon!
  - That means every so often Q-learning does a random action
- By measuring the performance of Q-learning while its following epsilon-greedy, we are evaluating Q-learning online**—while its learning and interacting**

Online vs Offline performance

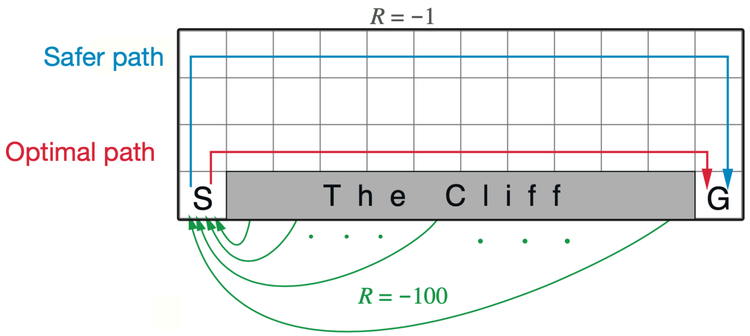
- Offline is the one that is different!
- We **do not** measure performance as the agent interacts and learns in the environment
  - We don’t record the rewards or number of steps while the agent is learning
- There are two phases that we switch between during the experiment:
  - **Learning** (updating the value function) **and taking actions**: no perf evaluation
  - **Testing**: learning is disabled, we evaluate the current policy \pi: record performance

Revisiting the cliff world

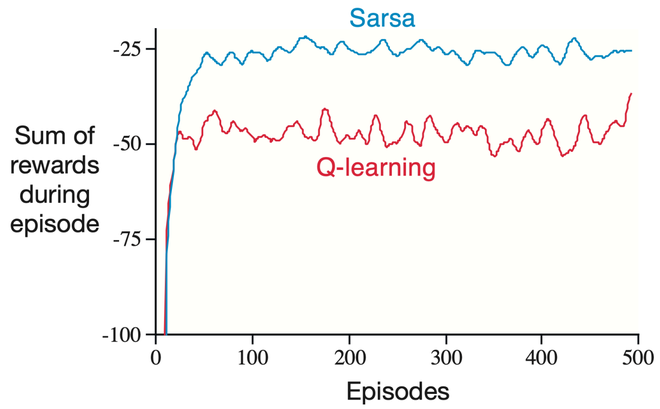
- Im we run Q-learning for a very very long time, such that it converges to \pi*
  - Q also converges to q*
- Then we run an episode, where actions are selected using epsilon greedy
  - A_t = epsilon_greedy(Q)
- What is argmax_a Q(S_{t+1}, a)?
- What will the rewards look like when we run epsilon_greedy(Q)?



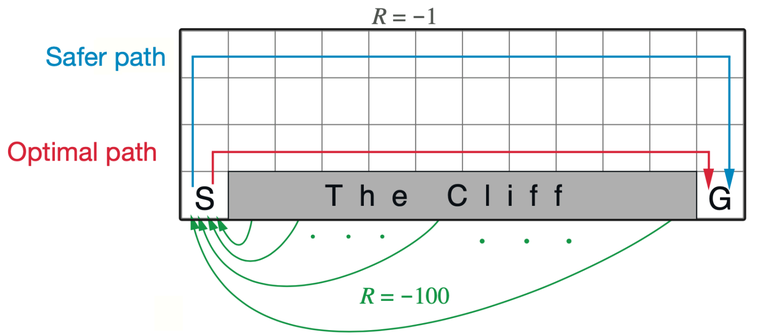
Revisiting the cliff world



- Imagine we run Q-learning for a few episodes with **A_t = epsilon_greedy(Q)**
- Then we run an episode a **test episode**, where actions are selected:
  - A_t = greedy(Q) and alpha=0 (learning disabled)
- What will the rewards look like?



Offline performance



- Let Q-learning continue learning for 100 more episodes with **A_t = epsilon_greedy(Q)**
- Then we run an episode a **test episode**:
  A_t = greedy(Q) and alpha=0 (learning disabled)
- What will the rewards look like when we run greedy(Q)?
- This is measuring offline performance
- The rewards during learning episodes, where we use epsilon-greedy, don’t count!

Worksheet questions

Q1

In Monte Carlo control, we required that every state-action pair be visited infinitely often. One way this can be guaranteed is by using exploring starts. Can we use exploring starts for Sarsa? Further, we have talked about using Sarsa with an ϵ-greedy policy. Can we use Monte Carlo with an ϵ-greedy policy? Does this ensure sufficient exploration?

Q2

- Exercise 6.12 Suppose action selection is greedy. Is Q-learning then exactly the same algorithm as Sarsa? Will they make exactly the same action selections and weight updates?

- Exercise 6.12 Suppose action selection is greedy. Is Q-learning then exactly the same algorithm as Sarsa? Will they make exactly the same action selections and weight updates?

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

- Imagine the following transition: <**S_t = X**, R_{t+1}, A_t, **S_{t+1} = X**>
  AND, that the update to Q(X, .) changed the max action is state X

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

- Imagine the following transition: <**S_t = X**, R_{t+1}, A_t, **S_{t+1} = X**>
  AND, that the update to Q(X, .) changed the max action is state X

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

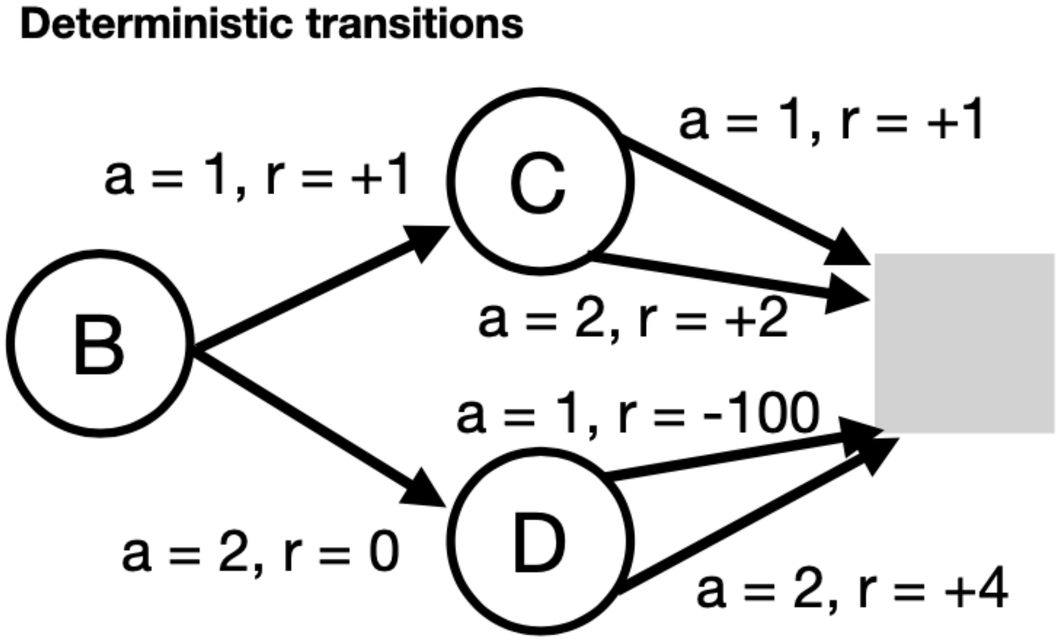
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

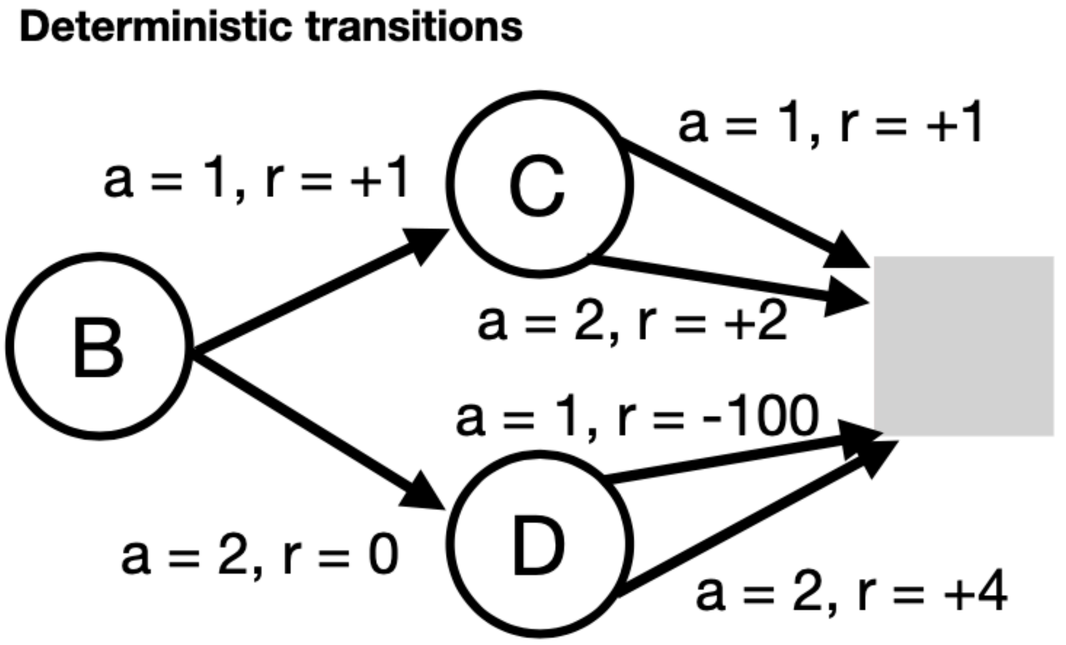
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Q3

Consider the following MDP, with three states B, C and D ($\mathcal{S} = \{B, C, D\}$), and 2 actions ($\mathcal{A} = \{1, 2\}$), with $\gamma = 1.0$. Assume the action values are initialized $Q(s, a) = 0 \; \forall \; s \in \mathcal{S}$ and $a \in \mathcal{A}$. The agent takes actions according to an ϵ-greedy policy with $\epsilon = 0.1$.



Q3

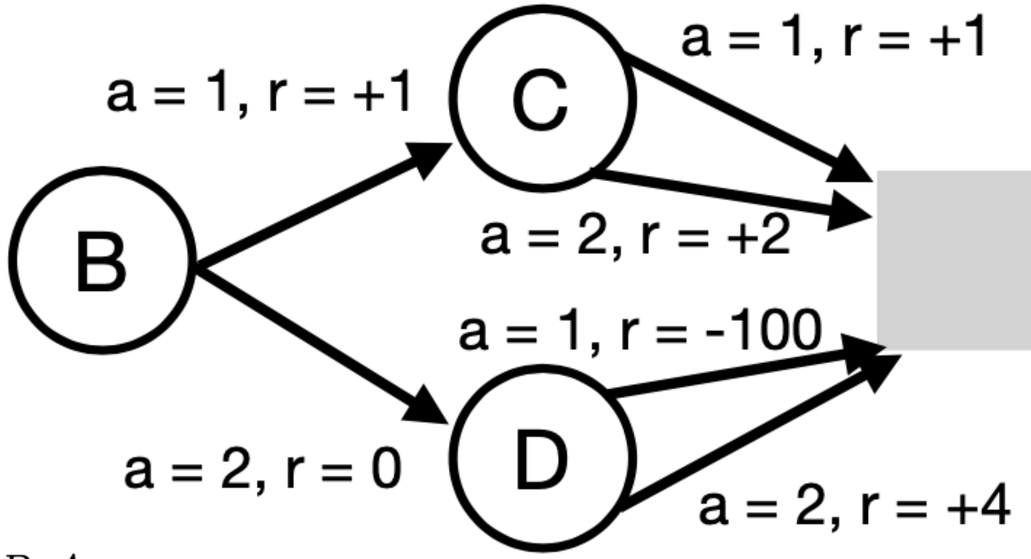


(a) What is the optimal policy for this MDP? What are the action-values corresponding to the optimal policy: $q^*(s, a)$?

- Work backwards and compute q* for each state and action:
  q*(C,1)=? q*(C,2)=? q*(D,1)=? q*(D,2)=? q*(B,1)=? q*(B,2)=?

- Use argmax of q* in each state to determine optimal policy
  pi(B)? pi(C)? pi(D)? How do we write it as probabilities ie pi(1|B)?

Q3

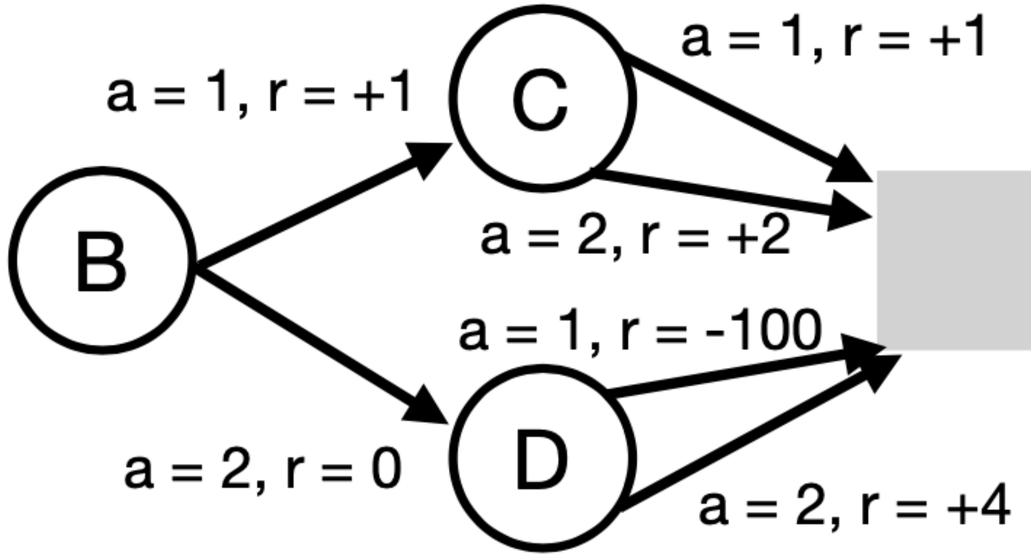


Imagine the agent experienced a single episode, and the following experience: $S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 2, R_2 = 4$. What are the Sarsa updates during this episode, assuming $\alpha = 0.1$? Start with state B, and compute and apply the Sarsa update. Then compute and apply the Sarsa update for the value of state D.

- Sarsa:
  Q(s,a) = Q(s,a) + alpha [R + \gamma Q(s',a') - Q(s,a)]
  Q(B,2) = Q(B,2) + alpha [R + \gamma Q(D,2) - Q(B,2)]
  Q(B,2) = 0 + 0.1 [0 + 1*0 - 0] = 0

  Q(s,a) = Q(s,a) + alpha [R + \gamma Q(s',a') - Q(s,a)]
  Q(D,2) = Q(D,2) + alpha [R + \gamma Q(T,.) - Q(D,2)]
  Q(D,2) = 0 + 0.1 [4 + 0 - 0] = 0.4

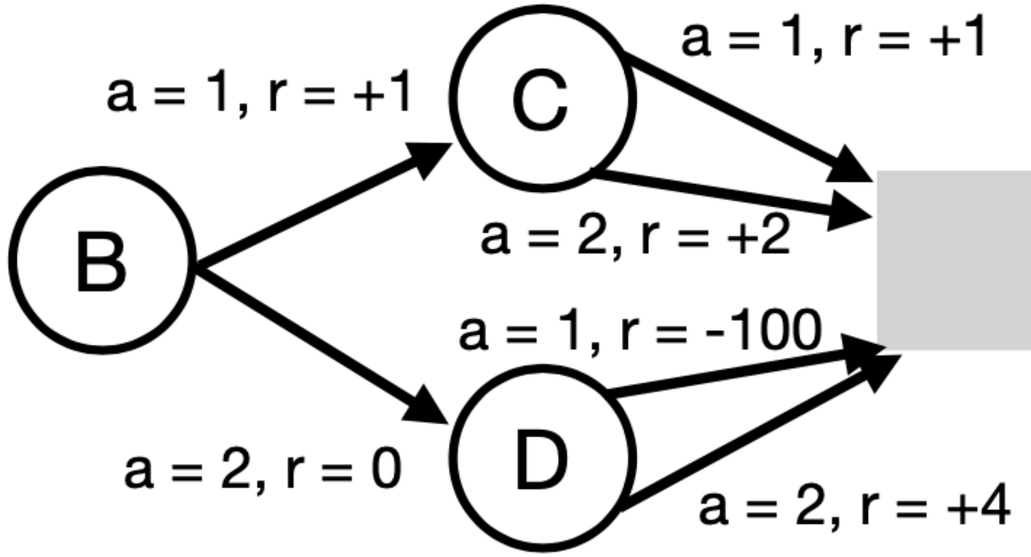
Q3



(b) Imagine the agent experienced a single episode, and the following experience: $S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 2, R_2 = 4$. What are the Sarsa updates during this episode, assuming $\alpha = 0.1$? Start with state B, and compute and apply the Sarsa update. Then compute and apply the Sarsa update for the value of state D.

(c) Using the sample episode above, compute the updates Q-learning would make, with $\alpha = 0.1$. Again start with state B, and then state D.

Q3



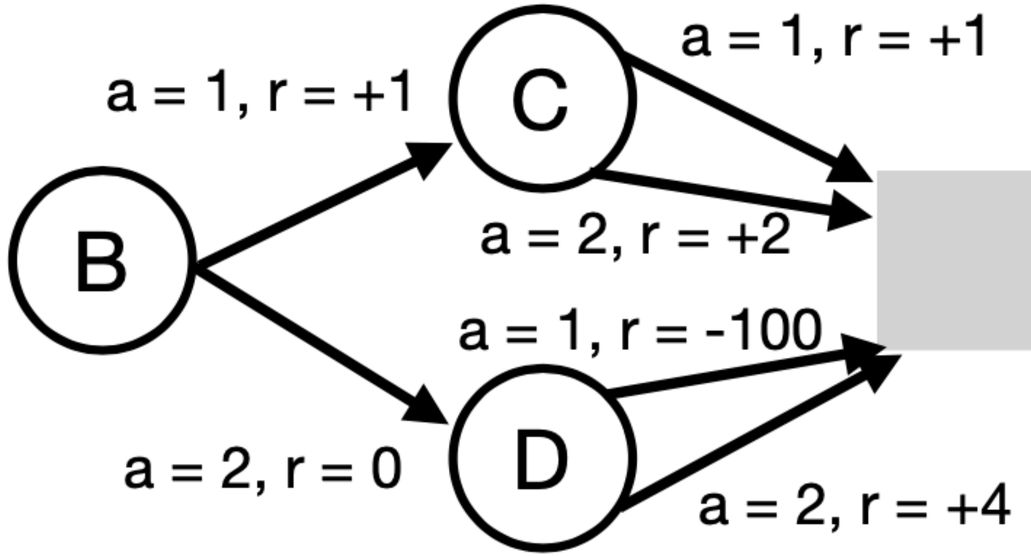
(c) Using the sample episode above, compute the updates Q-learning would make, with $\alpha = 0.1$. Again start with state B, and then state D.

Data: S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 2, R_2 = 4

- Q-learning:
  Q(s,a) = Q(s,a) + alpha [R + \gamma max_a'Q(s',a') - Q(s,a)]
  Q(B,2) = Q(B,2) + alpha [R + \gamma max_a' Q(D,a') - Q(B,2)]
  Q(B,2) = 0 + 0.1 [0 + 1*0 - 0] = 0

  Q(s,a) = Q(s,a) + alpha [R + \gamma max_a'Q(s',a') - Q(s,a)]
  Q(D,2) = Q(D,2) + alpha [R + \gamma max_a' Q(T,.) - Q(D,2)]
  Q(D,2) = 0 + 0.1 [4 + 0 - 0] = 0.4

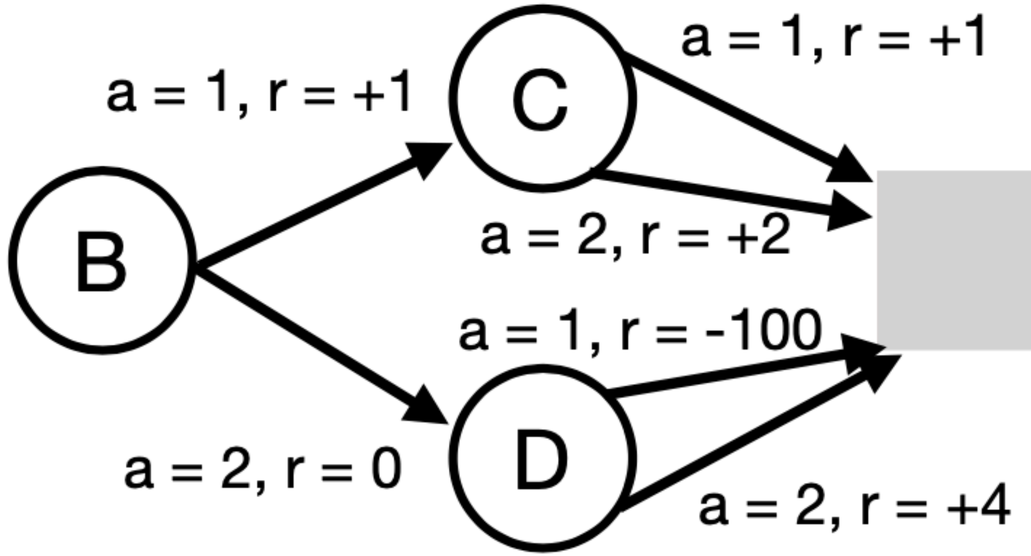
Q3



Let's consider one more episode: $S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100.$

- **What would the Q-learning updates be?**
  - Don't forget, Q(D,2) = 0.4 and the other values are equal to zero

Q3



- Q(B, 2) = 0; Q(D,2) = 0.4

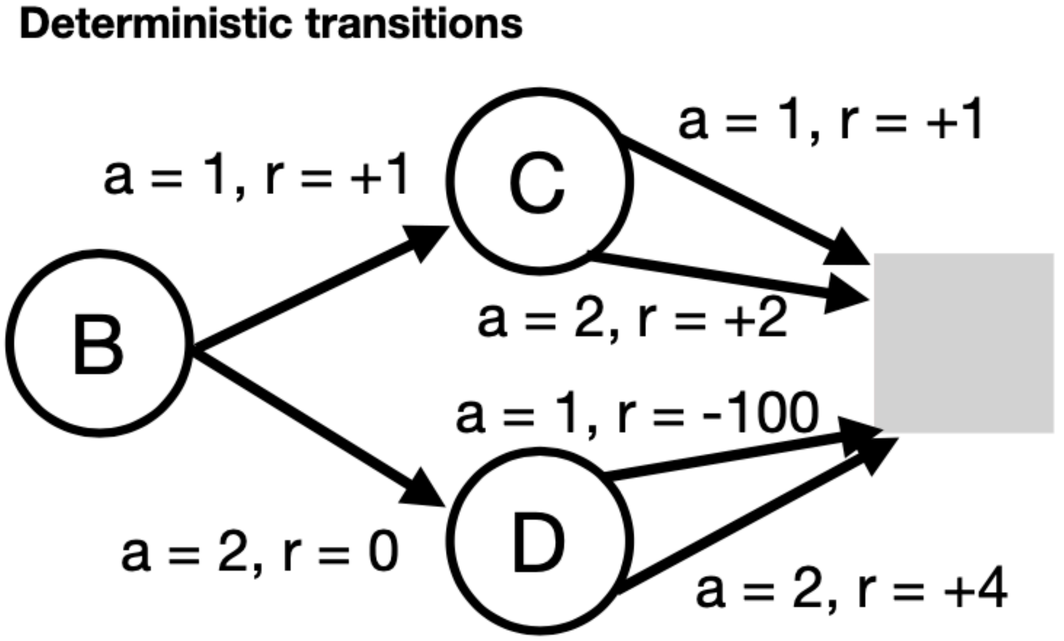
$$\max_{a' \in \{1,2\}} Q(D, a') = \max\{Q(D,1), Q(D,2)\}$$

Let's consider one more episode: $S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100.$

- Q-learning:
  Q(s,a) = Q(s,a) + alpha [R + \gamma max_a'Q(s',a') - Q(s,a)]
  Q(B,2) = Q(B,2) + alpha [R + \gamma **max_a' Q(D,a')** - Q(B,2)]
  Q(B,2) = Q(B,2) + alpha [R + \gamma **Q(D,2)** - Q(B,2)]
  Q(B,2) = 0 + 0.1 [0 + 1*0.4 - 0] = 0.04

  Q(s,a) = Q(s,a) + alpha [R + \gamma max_a'Q(s',a') - Q(s,a)]
  Q(D,1) = Q(D,1) + alpha [R + \gamma max_a' Q(T,.) - Q(D,1)]
  Q(D,1) = 0 + 0.1 [-100 + 0 - 0] = -10

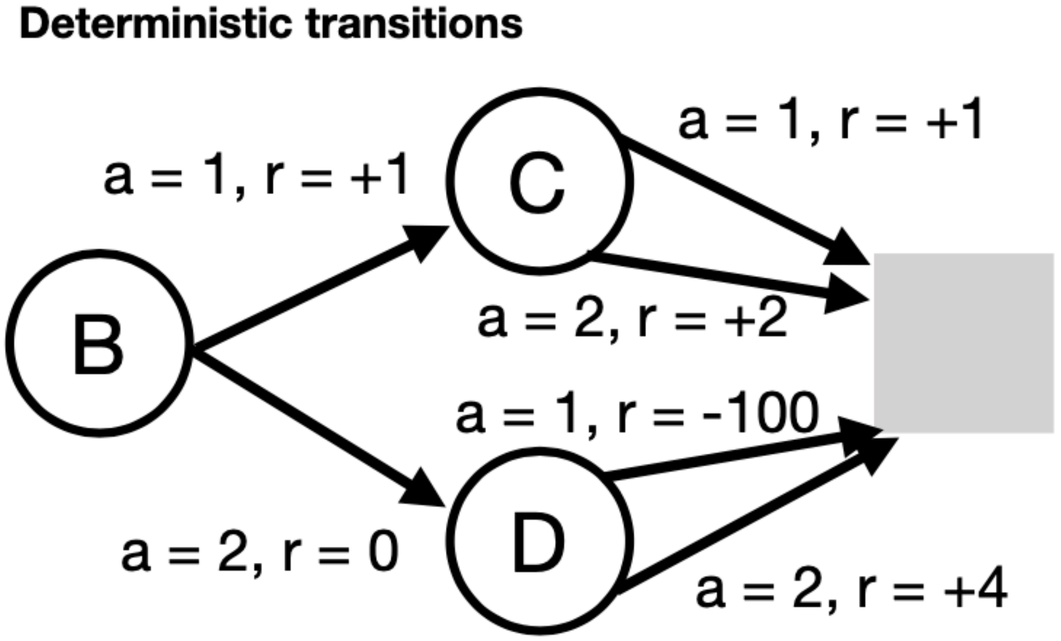
Q3



Let's consider one more episode: $S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100.$

- **What would the Sarsa updates be?**

Q3

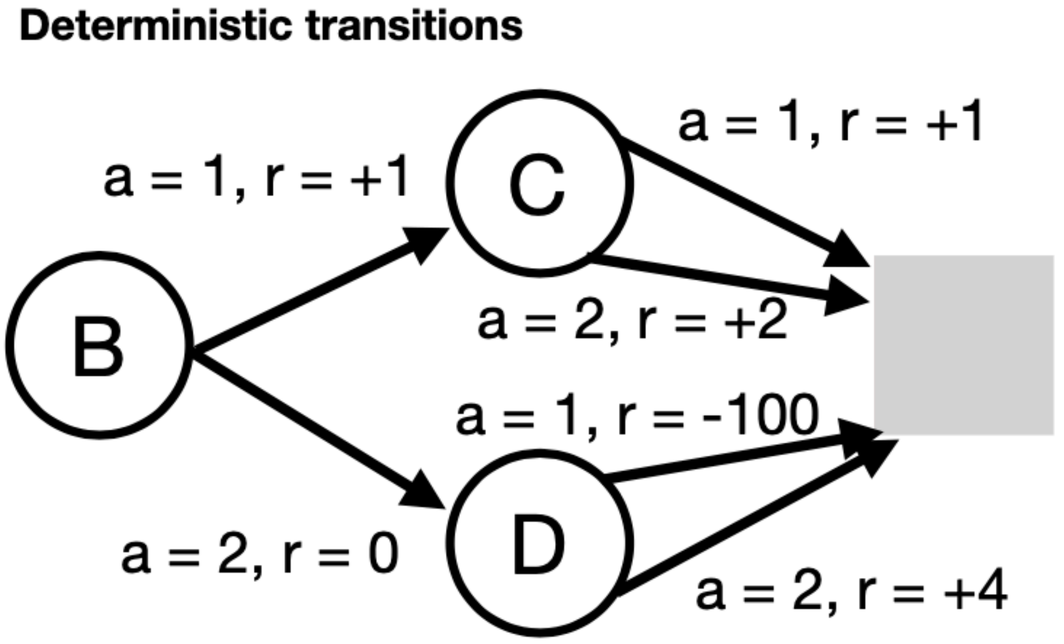


Let's consider one more episode: $S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100.$

- **What would the Sarsa updates be?**

- Sarsa: Q(B, 2) = 0; Q(D,1) = -10

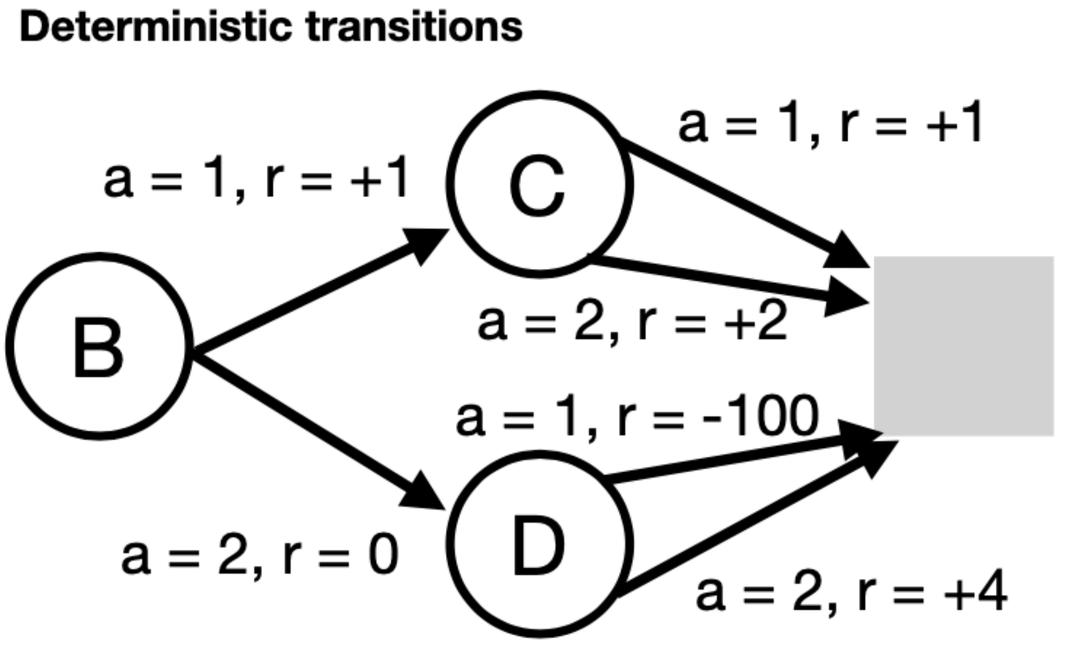
Q3



(d) Let's consider one more episode: $S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100$. What would the Sarsa updates be? And what would the Q-learning updates be?

(e) Assume you see one more episode, and it's the same one as in 1d. Once more update the action values, for Sarsa and Q-learning. What do you notice?

Q3

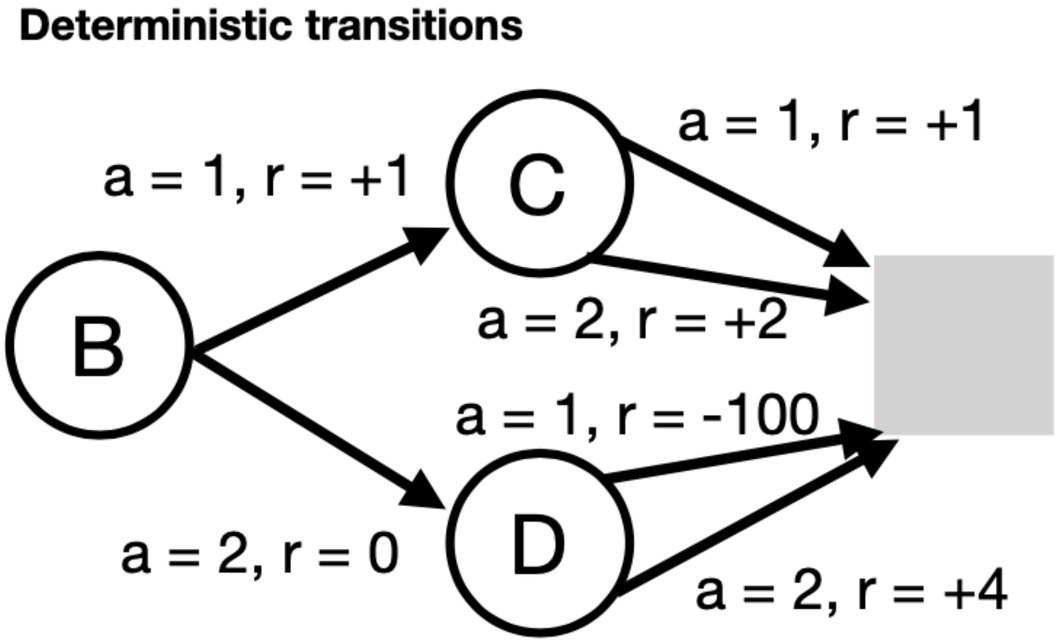


(d) Let's consider one more episode: $S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100$. What would the Sarsa updates be? And what would the Q-learning updates be?

(e) Assume you see one more episode, and it's the same one as in 1d. Once more update the action values, for Sarsa and Q-learning. What do you notice?

- Sarsa:
  Q(B, 2) = -1;
  Q(D,1) = -19
- Q-learning:
  Q(B, 2) = 0.076;
  Q(D,1) = -19

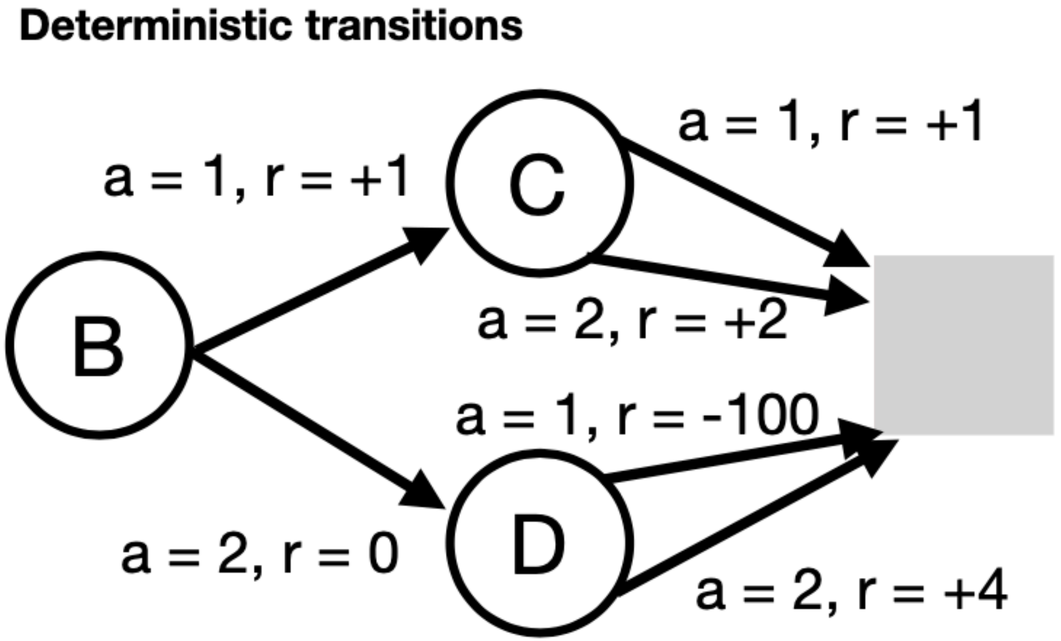
Q3



- **Do the values we are getting make sense?**
- Consider the optimal action values q*:
  q*(B,1) = 3, **q*(B,2) = 4**
  q*(C,1) = 1, q*(C,2) = 2
  q*(D,1) = -100, **q*(D,2) = 4**
- Recall we started with Q(s,a) = 0 forall s,a
- Consider the data we have seen:
  S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 2, R_2 = 4
  S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100
  S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100

- Sarsa:
  Q(B, 2) = 0, Q(D,1) = 0, Q(D,2) = 0.4
  Q(B, 2) = 0, Q(D,1) = -10, Q(D,2) = 0.4
  Q(B, 2) = -1, Q(D,1) = -19, Q(D,2) = 0.4

Q3



- **Do the values we are getting make sense?**
- Consider the optimal action values q*:
  q*(B,1) = 3, **q*(B,2) = 4**
  q*(C,1) = 1, q*(C,2) = 2
  q*(D,1) = -100, **q*(D,2) = 4**
- Recall we started with Q(s,a) = 0 forall s,a
- Consider the data we have seen:
  S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 2, R_2 = 4
  S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100
  S_0 = B, A_0 = 2, R_1 = 0, S_1 = D, A_1 = 1, R_2 = -100
- Which algorithm will converge to q*?

- Q-learning:
  Q(B, 2) = 0., Q(D,1) = 0, Q(D,2) = 0.4
  Q(B, 2) = 0.04, Q(D,1) = -10, Q(D,2) = 0.4
  Q(B, 2) = 0.076, Q(D,1) = -19, Q(D,2) = 0.4

Stochasticity and Variance

- “When comparing Sarsa and Q-learning in the cliff walking example, why does Sarsa learn the path that goes furthest from the cliff instead of a path that's only 1or 2 rows up from the cliff?”

- “How does sarsa fail to converge with higher values of alpha while expected sarsa stays constant for experiments such as the cliff walking experiment?”

- “If our rewards are stochastic how would q(s,a) change? I ask because while going through the videos and book our problems assume a constant reward per action but that's limiting so I'm curious how that might work.”

Challenge Question

5. In this question we compare the variance of the target for Sarsa and Expected Sarsa. Recall the update for Sarsa is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right]$$

and for Expected Sarsa is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|S_{t+1}) Q(S_{t+1}, a') - Q(S_t, A_t) \right].$$

(a) Start by comparing the part of the update that is different: $Q(S_{t+1}, A_{t+1})$ compared to $\sum_{a' \in \mathcal{A}} \pi(a'|S_{t+1}) Q(S_{t+1}, a')$. Write down the variance for these two terms, given $S_{t+1} = s'$.

$$\text{Var}(Q(s', A_{t+1})) \quad \text{and} \quad \text{Var}\left(\sum_{a' \in \mathcal{A}} \pi(a'|s') Q(s', a') \right)$$

Conclude that the variance is zero for Expected Sarsa, but likely non-zero for Sarsa. Notice that the only random variable is A_{t+1}, which is the action selected according to the target policy π with distribution $\pi(\cdot|S_{t+1})$.

(b) **Challenge Question:** Show that the variance of the Sarsa target is always greater than or equal to the variance of the Expected Sarsa target, given $S_t = s$ and $A_t = a$. Hint: use the Law of Total Variance, which states that for any two random variables X and Y, $\text{Var}(Y) = \mathbb{E}[\text{Var}(Y|X)] + \text{Var}(\mathbb{E}[Y|X])$. This law also applies to conditional distributions: for any random variables X, Y and Z, $\text{Var}(Y|Z) = \mathbb{E}[\text{Var}(Y|X, Z)|Z] + \text{Var}(\mathbb{E}[Y|X, Z]|Z)$.