# Introduction to Reinforcement Learning

(1) MDP (Markov Decision Process)
& basic conceptions of RL (reinforcement learning)
(2) Bellman Equation
(3) Value iteration
(4) Q-learning & SARSA

(1)MDP (Markov Decision Process)
& basic conceptions of RL (reinforcement learning)

# The formal mathematical model:

State set: $S$

Action set: $A$

State transition model: $P(s_{t+1}|s_t, a_t)$

-- markov assumption: transition probability only depends on $s_t$ and $a_t$, and not previous actions or states.

Reward function: $R(s_t, a_t, s_{t+1})$

Policy: $\pi(s)$

-- define the action probability on state s

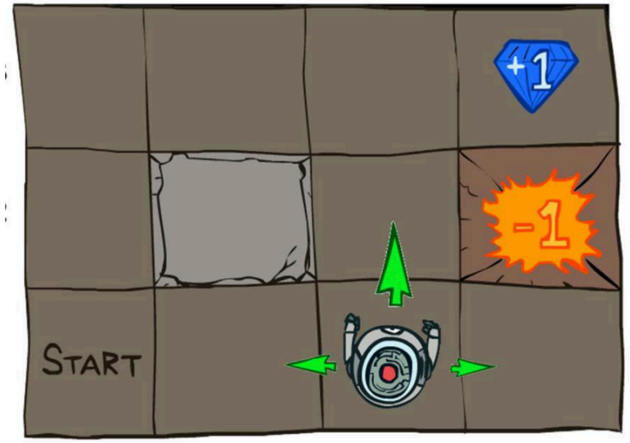
Reward decay: $\gamma$ (we care more about reward in

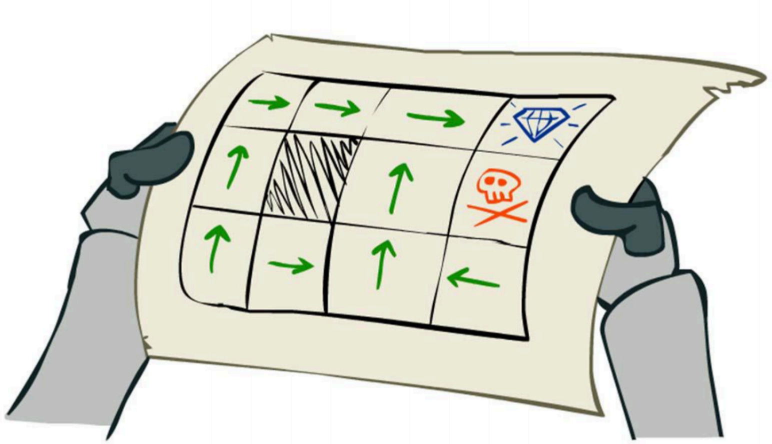
Total reward: $\mathrm{U}(s_0, a_0, s_1, a_1, s_2, \dots, s_{T-1}, a_{T-1}, s_T)$

$=\sum_{t=0}^{t=T-1} \gamma^t \times R(s_t, a_t, s_{t+1})$.

# Example of MDP: Grid World

Robot on a grid; goal: find the best policy



Source: P. Abbeel and D. Klein

$S$ : the location of the robot, we have 11 states (we cannot move to the rock)

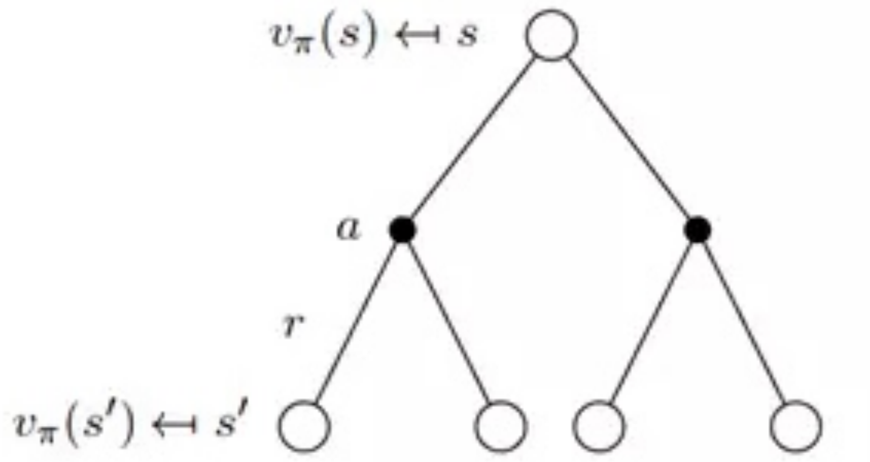
$A$: possible commands, we have 4 commands, up down right left.

$P(s_{t+1}|s_t, a_t)$: the robot will move to the other directions with 10% probability

$R(s_t, a_t, s_{t+1}) = the\ cost\ of\ the\ grid$

*Find the best policy* $\pi(s)$ to maximize total reward $R(s_0, a_0, s_1, \ldots, s_{T-1}, a_{T-1}, s_T)$

# (2) Bellman Equation




Value function:
describes the expected <u>future</u> value after a state $s_t$ or an action $a_t$

- State-Value Function:

$$v_\pi(s_t) = E[U(s_t, a_t, s_{t+1}, \dots)|s_t]$$

- Action-Value Function:

$$q(s_t, a_t) = E[U(s_t, a_t, s_{t+1}, \dots)|s_t, a_t]$$

- recursive formula for State-Value Function:

$$v_\pi(s_t) = \sum_{a_t \in A} P(a_t|\pi, s_t) \times \sum_{s_{t+1} \in S} P(s_{t+1}|s_t, a_t) \times (R(s_t, a_t, s_{t+1}) + \gamma \times v_\pi(s_{t+1}))$$
$$= \sum_{a_t \in A} P(a_t|\pi, s_t) \times q(s_t, a_t)$$

- recursive formula for Action-Value Function: Quiz 1

# (3) Value iteration

## optimal state-value function

$$v_\pi(s_t) = \sum_{a_t \in A} P(a_t|\pi, s_t) \times \sum_{s_{t+1} \in S} P(\, s_{t+1}|s_t, a_t) \times (R(s_t, a_t, s_{t+1}) + \gamma \times v_\pi(s_{t+1}))$$

$$v_{\pi^*}(s_t) = \max_{a_t \in A} \sum_{s_{t+1} \in S} P(s_{t+1}|s_t, a_t) \times (R(s_t, a_t, s_{t+1}) + \gamma \times v_{\pi^*}(s_{t+1}))$$

## value iteration algorithm

For all states, random initialize $v_0(s)$

for each iteration:

  for all states:

$$v_{k+1}(s) = \max_{a \in A} \sum_{s\prime \in S} P(s'|s, a) \times (R(s, a, s') + v_k(s'))$$

# (4) Q-learning & SARSA

# What if we know nothing about $P(s_{t+1}|s_t, a_t)$ ?

$$v_\pi(s_t) = \sum_{a_t \in A} P(a_t|\pi, s_t) \times \sum_{s_{t+1} \in S} P(s_{t+1}|s_t, a_t) \times (R(s_t, a_t, s_{t+1}) + v_\pi(s_{t+1}))$$

We reserve a table of $Q(s, a)$

For each iteration:

We start from the start state $s_0$, go through a path to the terminal state $s_T$

We update parts of $Q(s, a)$ based on the path $s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T$

# Q-learning

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., $\epsilon$-greedy)
    Take action $A$, observe $R$, $S'$
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', \boxed{a}) - Q(S, A)\big]$
    $S \leftarrow S'$
  until $S$ is terminal

# SARSA

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

# Difference between Q-learning and SARSA

Diagram of Q-learning : An off-policy TD control algortithm

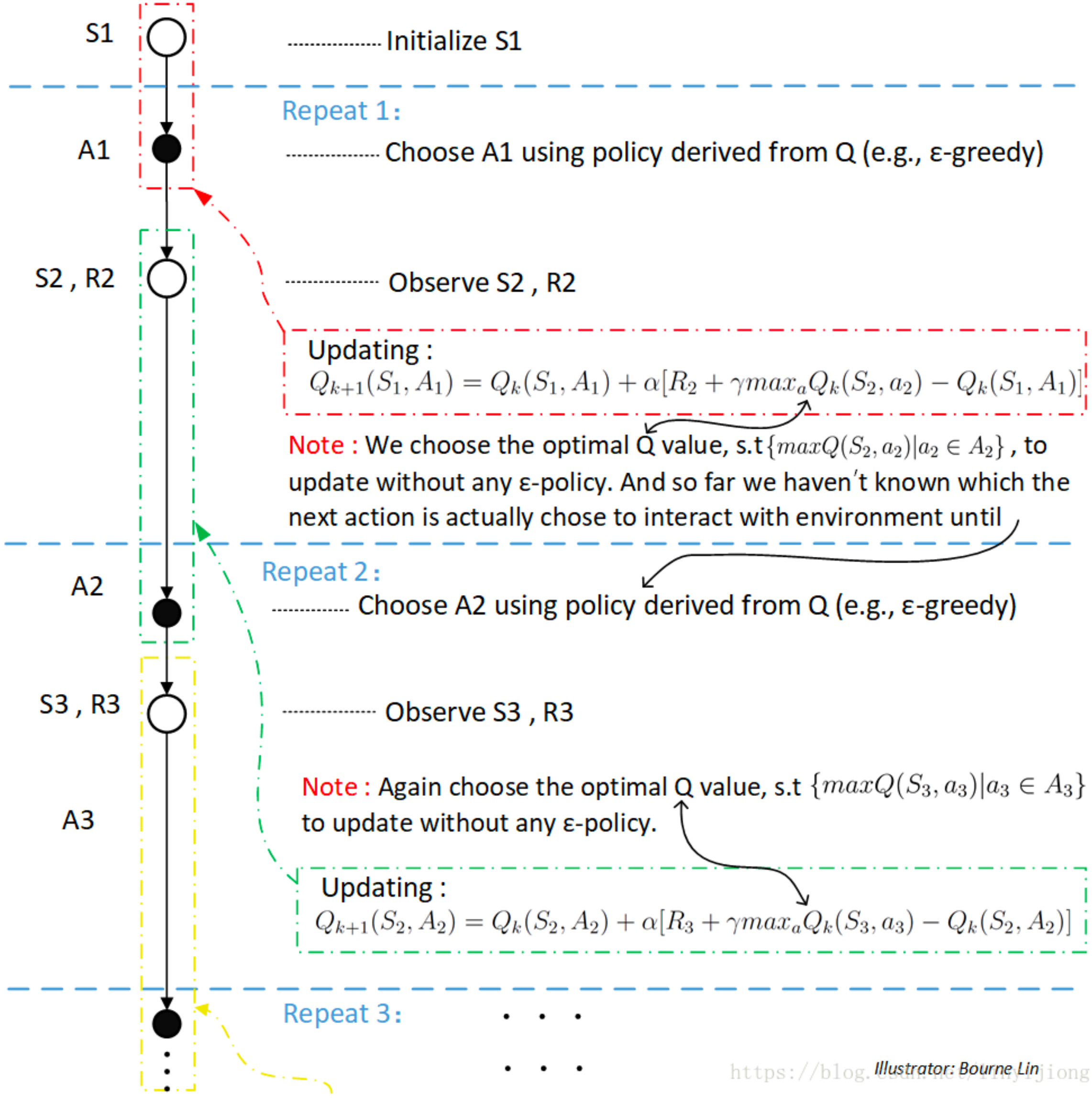
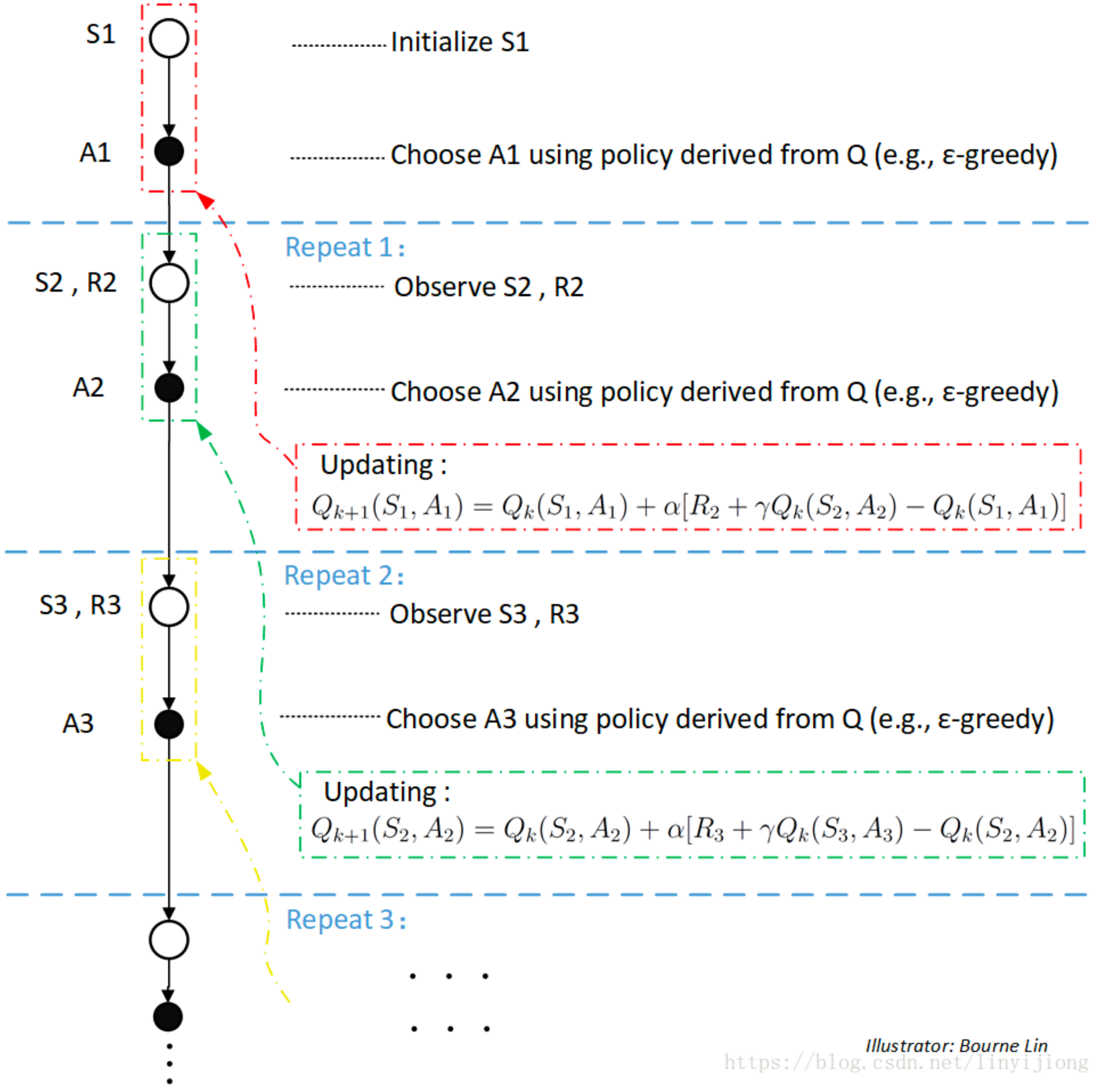



Diagram of Sarsa : An on-policy TD control algorithm



# Quiz 2

- We start from s0, choose a0; we get reward 1 and then we get to s2, and choose a1.
- Please update the Q(s0,a0) based on the current Q table and the movement above, using SARSA and Q-learning
- $\gamma = 0.5, \alpha$=0.5

| | a0 | a1 | a2 |
|---|---|---|---|
| s0 | 1 | 2 | 3 |
| s1 | 2 | 3 | 1 |
| s2 | 3 | 1 | 2 |

| | a0 | a1 | a2 |
|---|---|---|---|
| s0 | 1 | 2 | 3 |
| s1 | 2 | 3 | 1 |
| s2 | 3(Q-learning) | 1(SARSA) | 2 |

s0, a0, reward=1, s2, a1

$$Q\_learning: Q(s0, a0) = 1 + 0.5 \times [1 + 0.5 \times 3 - 1] = 1.75$$

$$SARSA: Q(s0, a0) = 1 + 0.5 \times [1 + 0.5 \times 1 - 1] = 1.25$$

# Quiz 3

In the grid world below, the green area is grass, and every square you move on it will deduct 1 point, and the black area will deduct 100 points and return to the starting point S (Start). We hope to learn a path with the highest score to reach the terminal T (Terminal), the terminal will give 100 points. Use SARSA and Q-learning respectively for learning. The results are shown in the figure, and the red is the optimal path of the corresponding algorithm. Which one is the result of SARSA?