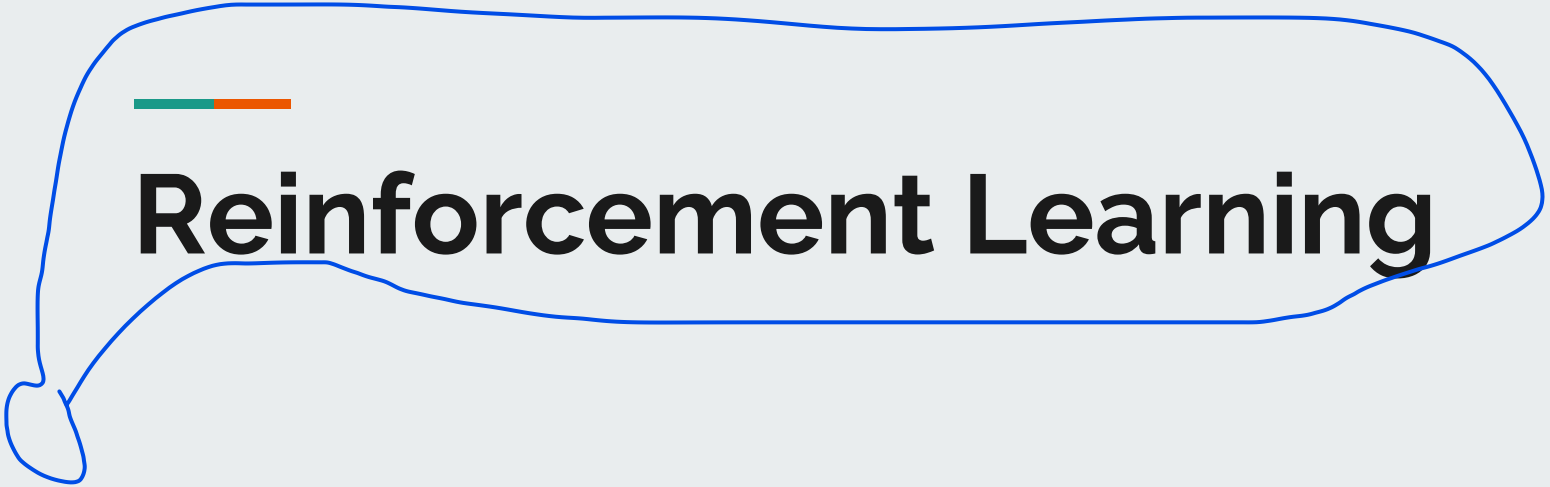
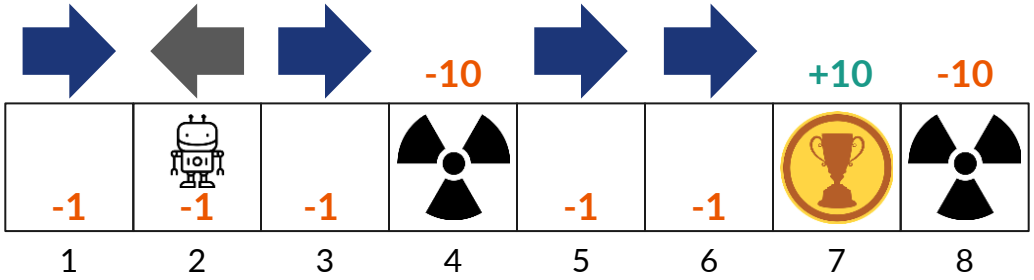


Reinforcement Learning

Chapter 21



- The agent interacts with the environment for 3 episodes.
- Each episode start from a random state.
- The state sequence for each episode is shown in Table 1.

E1	2 → 1 → 3 → 5 → 7
E2	1 → 2 → 1 → 3 → 4
E3	5 → 6 → 7

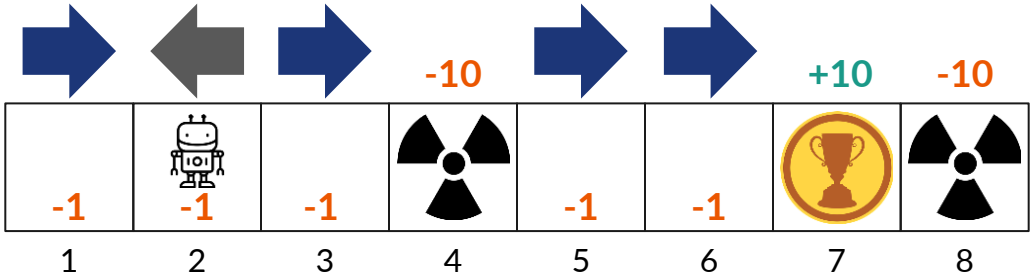
Table 1

Using three episodes shown in table 1, do the following:

- Apply “Direct Utility Estimation”.
- Following “Adaptive Dynamic Programming”, estimate the transition model then write down the linear equations to evaluate the policy after receiving all the percept.
- Assuming all utilities are initialized to 0, apply “TD-Learning” on the following episodes.

Assume all cells (except terminals) have -1 reward, and assume that the discount factor is 1.

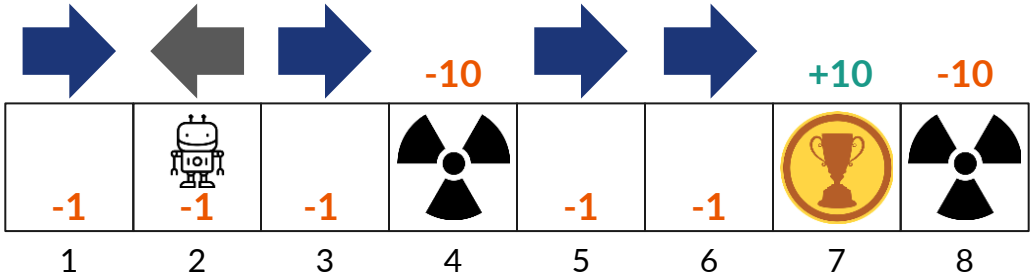
Direct Utility Estimation:



Rollout	2	1	3	5	7
Reward	-1	-1	-1	-1	+10
Acc. Reward	6	7	8	9	10

State	1	2	3	4	5	6	7	8
Acc. Reward Sum	7	6	8	0	9	0	10	0
Visits	1	1	1	0	1	0	1	0
Expected Utility	7	6	8	-	9	-	10	-

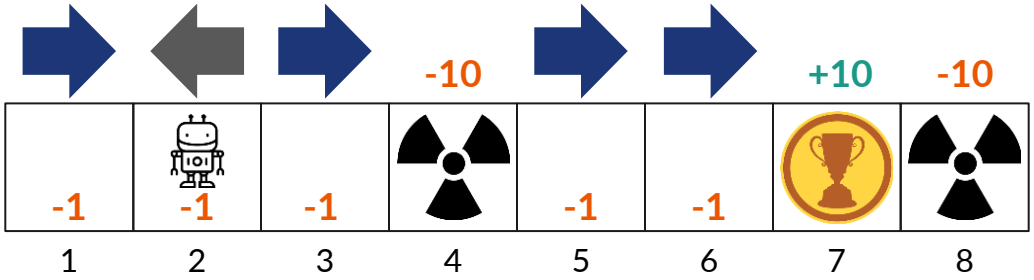
Direct Utility Estimation:



Rollout	1	2	1	3	4
Reward	-1	-1	-1	-1	-10
Acc. Reward	-14	-13	-12	-11	-10

State	1	2	3	4	5	6	7	8
Acc. Reward Sum	-18	-7	-3	-10	9	0	10	0
Visits	3	2	2	1	1	0	1	0
Expected Utility	-6	-3.5	-1.5	-10	9	-	10	-

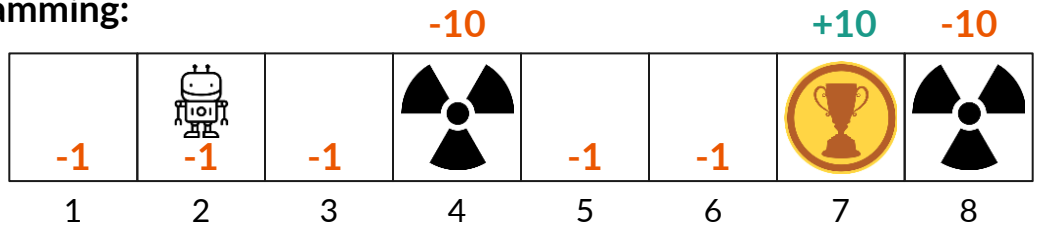
Direct Utility Estimation:



Rollout	5	6	7		
Reward	-1	-1	+10		
Acc. Reward	8	9	10		

State	1	2	3	4	5	6	7	8
Acc. Reward Sum	-18	-7	-3	-10	17	9	20	0
Visits	3	2	2	1	2	1	2	0
Expected Utility	-6	-3.5	-1.5	-10	8.5	9	10	-

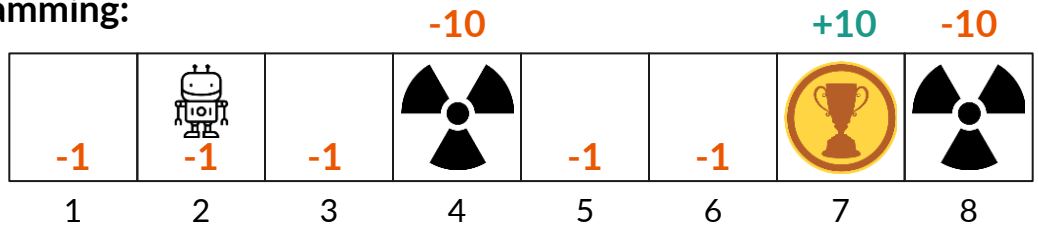
Adaptive Dynamic Programming:



E1: 2 → 1 → 3 → 5 → 7

State	Transition Model	Reward
1	3 with p=1/1	-1
2	1 with p=1/1	-1
3	5 with p=1/1	-1
4		0
5	7 with p=1/1	-1
6		0
7	Terminal	+10
8		0

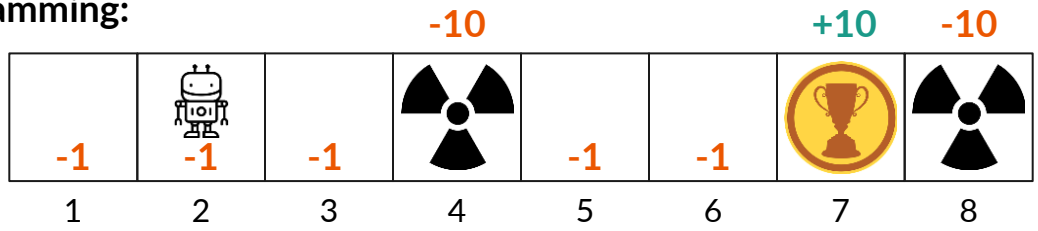
Adaptive Dynamic Programming:



E2: 1 → 2 → 1 → 3 → 4

State	Transition Model	Reward
1	3 with p=2/3, 2 with p=1/3	-1
2	1 with p=2/2	-1
3	5 with p=1/2, 4 with p=1/2	-1
4	Terminal	-10
5	7 with p=1/1	-1
6		0
7	Terminal	+10
8		0

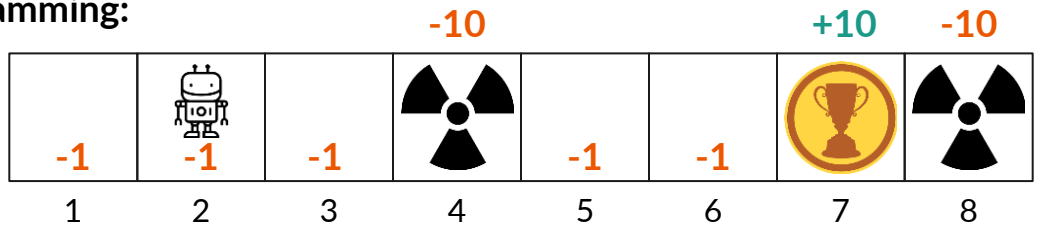
Adaptive Dynamic Programming:



E3: 5 → 6 → 7

State	Transition Model	Reward
1	3 with p=2/3, 2 with p=1/3	-1
2	1 with p=2/2	-1
3	5 with p=1/2, 4 with p=1/2	-1
4	Terminal	-10
5	7 with p=1/2, 6 with p=1/2	-1
6	7 with p=1/1	-1
7	Terminal	+10
8		0

Adaptive Dynamic Programming:



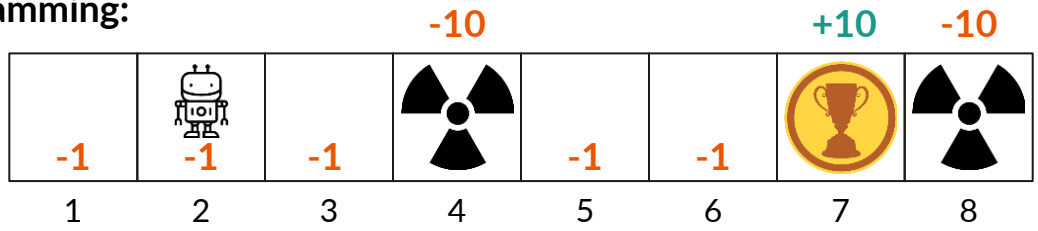
$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s')$$

Equations

U(1) = -1 + (2/3) * U(3) + (1/3) * U(2)
U(2) = -1 + (2/2) * U(1)
U(3) = -1 + (1/2) * U(5) + (1/2) * U(4)
U(4) = -10
U(5) = -1 + (1/2) * U(7) + (1/2) * U(6)
U(6) = -1 + (1/1) * U(7)
U(7) = 10

State	Transition Model	Reward
1	3 with p=2/3, 2 with p=1/3	-1
2	1 with p=2/2	-1
3	5 with p=1/2, 4 with p=1/2	-1
4	Terminal	-10
5	7 with p=1/2, 6 with p=1/2	-1
6	7 with p=1/1	-1
7	Terminal	+10
8		0

Adaptive Dynamic Programming:



$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s')$$

Equations

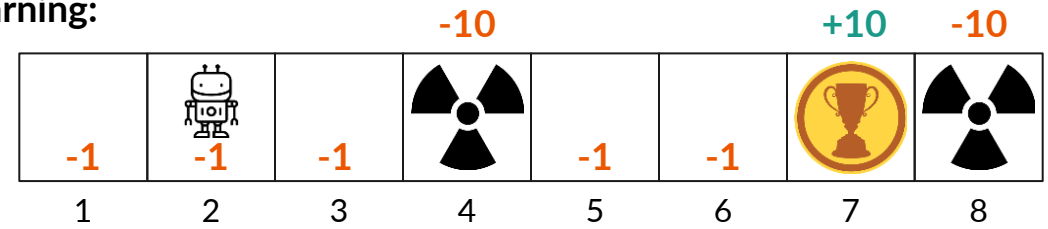
U(1) = -1 + (2/3) * U(3) + (1/3) * U(2)
U(2) = -1 + (2/2) * U(1)
U(3) = -1 + (1/2) * U(5) + (1/2) * U(4)
U(4) = -10
U(5) = -1 + (1/2) * U(7) + (1/2) * U(6)
U(6) = -1 + (1/1) * U(7)
U(7) = 10

Solution

U(1) = -3.75
U(2) = -4.75
U(3) = -1.75
U(4) = -10
U(5) = 8.5
U(6) = 9
U(7) = 10

Temporal Difference Learning:

Let alpha = 0.1



E1: 2 → 1 → 3 → 5 → 7

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s))$$

U(2) ← U(2) + 0.1(R(2) + 1*U(1) - U(2))
← 0 + 0.1(-1 + 1*0 - 0) ← -0.1

U(1) ← 0 + 0.1(-1 + 1*0 - 0) ← -0.1

U(3) ← 0 + 0.1(-1 + 1*0 - 0) ← -0.1

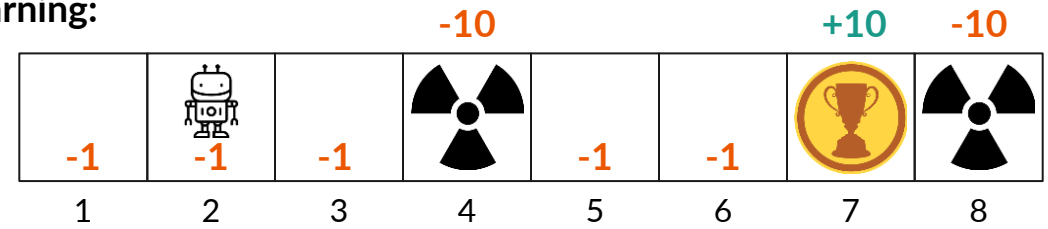
U(5) ← 0 + 0.1(-1 + 1*10 - 0) ← 0.9

U(7) ← 10

State	1	2	3	4	5	6	7	8
Expected Utility	-0.1	-0.1	-0.1	-	0.9	-	10	-

Temporal Difference Learning:

Let alpha = 0.1



E2: 1 → 2 → 1 → 3 → 4

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma\, U^{\pi}(s') - U^{\pi}(s))$$

U(1) ← -0.1 + 0.1(-1 + 1*-0.1 - -0.1) ← -0.2

U(2) ← -0.1 + 0.1(-1 + 1*-0.2 - -0.1) ← -0.21

U(1) ← -0.2 + 0.1(-1 + 1*-0.1 - -0.2) ← -0.29

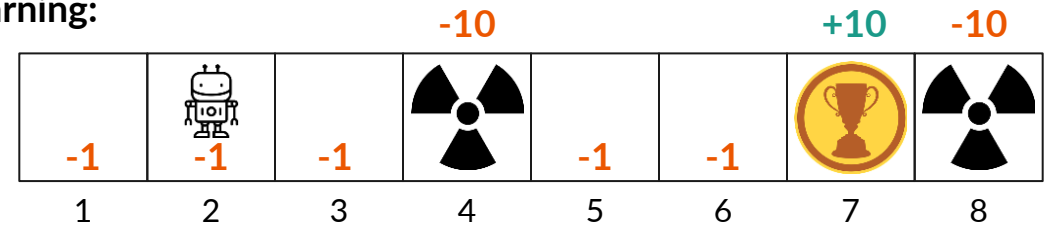
U(3) ← -0.1 + 0.1(-1 + 1*-10 - -0.1) ← -1.19

U(4) ← -10

State	1	2	3	4	5	6	7	8
Expected Utility	-0.29	-0.21	-1.19	-10	0.9	-	10	-

Temporal Difference Learning:

Let alpha = 0.1



E3: 5 → 6 → 7

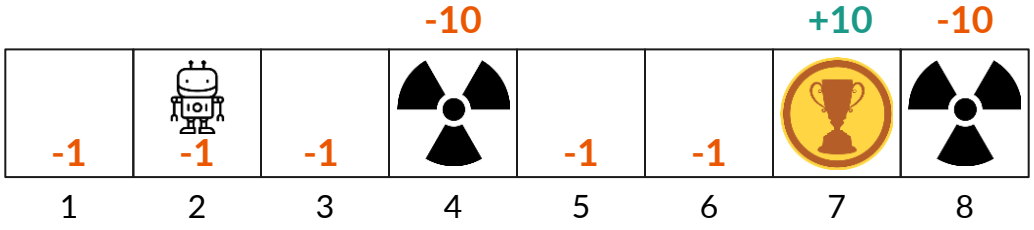
$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma\, U^{\pi}(s') - U^{\pi}(s))$$

U(5) ← 0.9 + 0.1(-1 + 1*0 - 0.9) ← 0.71

U(6) ← 0 + 0.1(-1 + 1*10 - 0) ← -0.9

U(7) ← 10

State	1	2	3	4	5	6	7	8
Expected Utility	-0.29	-0.21	-1.19	-10	0.71	-0.9	10	-



Assume each episode start from at state 2.

Assume all cells (except terminals) have -1 reward, and assume that the discount factor is 1.

Since we are going to sample the rollout by our self, we need the transition model:

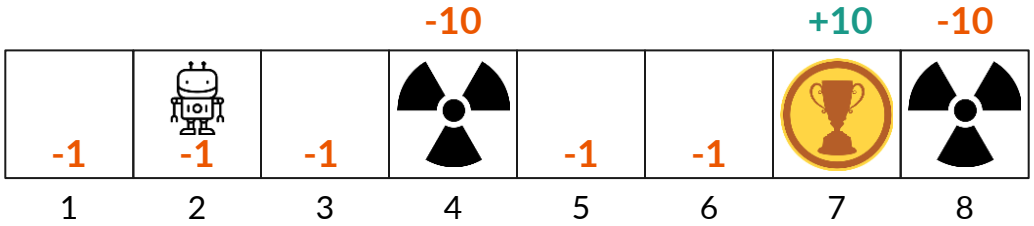
- Action L: 100% Move left one step
- Action R: 20% Move right one step, 80% Move right 2 steps.

Note: In real application, we rarely know the transition model. Instead, we sample the transition by interacting with an environment.

- **Apply Tabular Q-Learning to the aforementioned environment problem. Assume that the policy will pick L if the Q-values is indecisive.**

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

2, L → 1

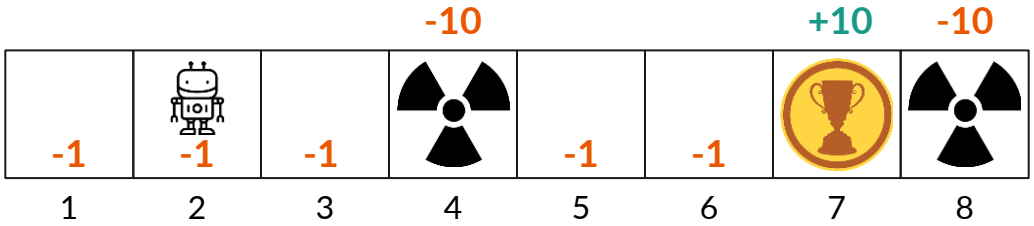
$$Q(2, L) \leftarrow Q(2,L) + 0.1(\ R(2) + 1*\max(Q(1,L),\ Q(1,R)) - Q(2,L))$$

$$\leftarrow 0 + 0.1(\ -1 + 1*\max(0,\ 0) - 0) \quad \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	0	-0.1	0	0	0	0	0	0
Q(State, R)	0	0	0		0	0		
Policy	L	R	L	L	L	L	L	L

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

1, L → 1

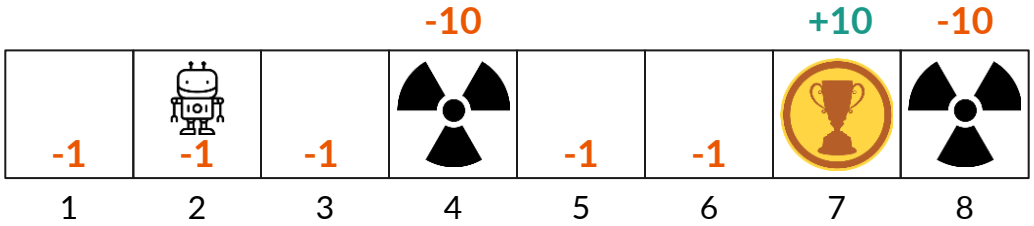
$$Q(1, L) \leftarrow Q(1,L) + 0.1(\ R(1) + 1*\max(Q(1,L),\ Q(1,R)) - Q(1,L))$$

$$\leftarrow 0 + 0.1(\ -1 + 1*\max(0,\ 0) - 0) \quad \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.1	0	0	0	0	0	0
Q(State, R)	0	0	0		0	0		
Policy	R	R	L	-	L	L	-	-

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

1, R → 2

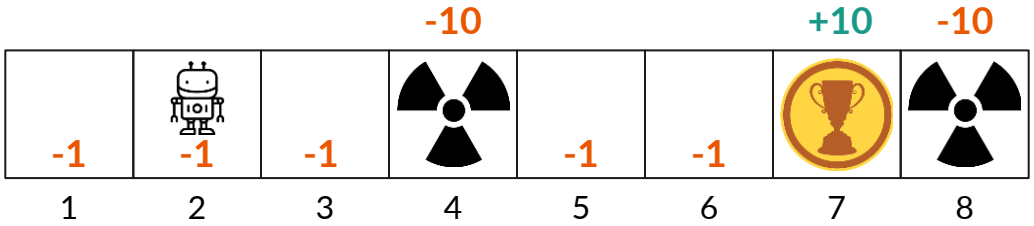
$$Q(1, L) \leftarrow Q(1,L) + 0.1(\ R(1) + 1*\max(Q(2,L),\ Q(2,R)) - Q(1,L))$$

$$\leftarrow 0 + 0.1(\ -1 + 1*\max(-0.1,\ 0) - 0) \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.1	0	0	0	0	0	0
Q(State, R)	-0.1	0	0		0	0		
Policy	L	R	L	-	L	L	-	-

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

2, R → 3

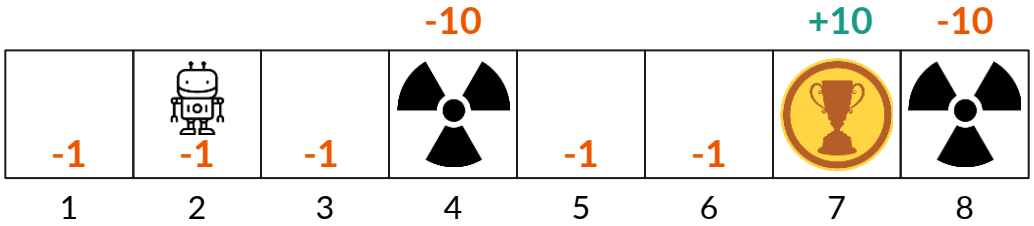
$$Q(2, R) \leftarrow Q(2,R) + 0.1(R(2) + 1*\max(Q(3,L), Q(3,R)) - Q(2,R))$$

$$\leftarrow 0 + 0.1(-1 + 1*\max(0, 0) - 0) \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.1	0	0	0	0	0	0
Q(State, R)	-0.1	-0.1	0		0	0		
Policy	L	L	L	-	L	L	-	-

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

3, L → 2

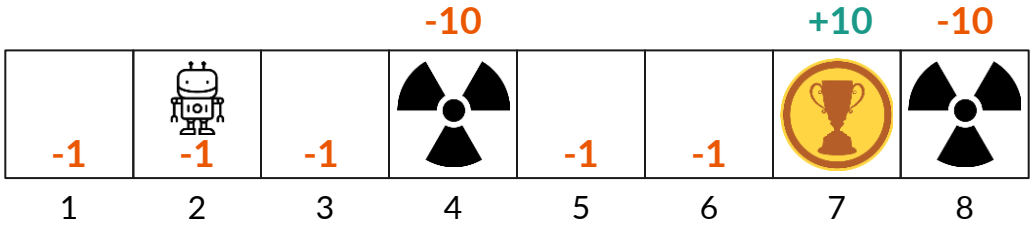
$$Q(3, L) \leftarrow Q(3,L) + 0.1(R(3) + 1*\max(Q(2,L), Q(2,R)) - Q(3,L))$$

$$\leftarrow 0 + 0.1(-1 + 1*\max(-0.1, -0.1) - 0) \leftarrow -0.11$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.1	-0.11	0	0	0	0	0
Q(State, R)	-0.1	-0.1	0		0	0		
Policy	L	L	R	-	L	L	-	-

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

2, L → 1

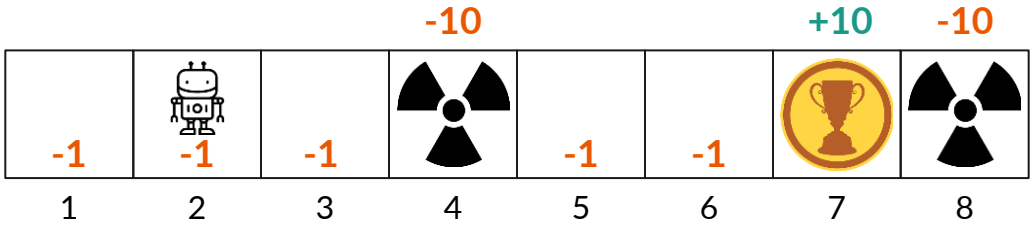
$$Q(2, L) \leftarrow Q(2,L) + 0.1(\ R(2) + 1*\max(Q(1,L),\ Q(1,R)) - Q(2,L))$$

$$\leftarrow -0.1 + 0.1(\ -1 + 1*\max(-0.1,\ -0.1) - -0.1) \leftarrow -0.2$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.2	-0.11	0	0	0	0	0
Q(State, R)	-0.1	-0.1	0		0	0		
Policy	L	R	R	-	L	L	-	-

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

1, L → 1

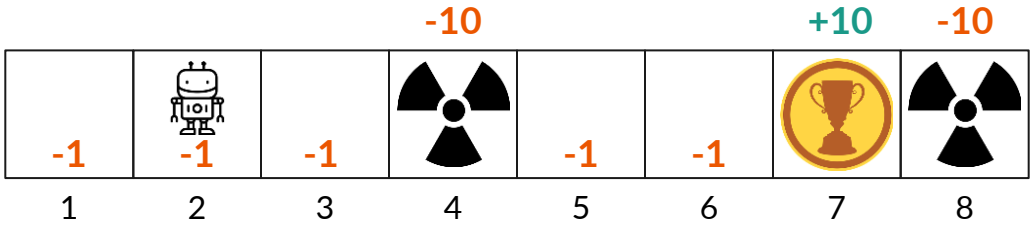
$$Q(1, L) \leftarrow Q(1,L) + 0.1(\ R(1) + 1*\max(Q(1,L),\ Q(1,R)) - Q(1,L))$$

$$\leftarrow -0.1 + 0.1(\ -1 + 1*\max(-0.1,\ -0.1) - -0.1) \leftarrow -0.2$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.2	-0.2	-0.11	0	0	0	0	0
Q(State, R)	-0.1	-0.1	0		0	0		
Policy	R	R	R	-	L	L	-	-

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

1, R → 3

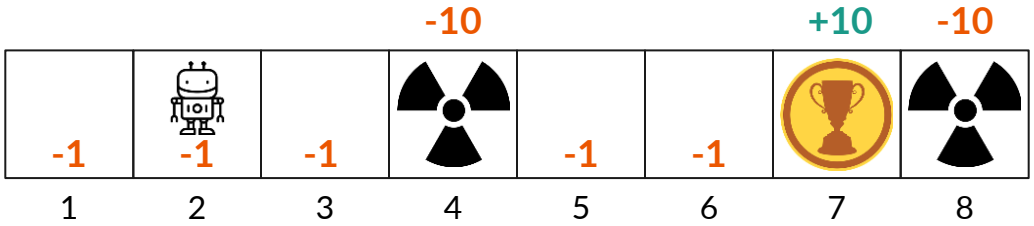
$$Q(1, R) \leftarrow Q(1,R) + 0.1(R(1) + 1*\max(Q(3,L), Q(3,R)) - Q(1,R))$$

$$\leftarrow -0.1 + 0.1(-1 + 1*\max(-0.11, 0) - -0.1) \leftarrow -0.19$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.21	-0.2	-0.11	0	0	0	0	0
Q(State, R)	-0.19	-0.1	0		0	0		
Policy	R	R	R	-	L	L	-	-

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

3, R → 5

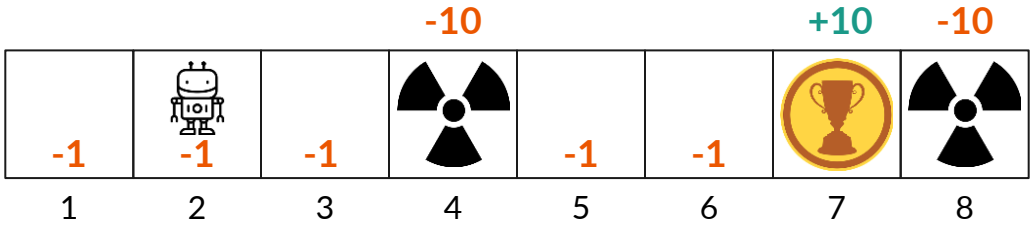
$$Q(3, R) \leftarrow Q(3,R) + 0.1(R(3) + 1*\max(Q(5,L), Q(5,R)) - Q(3,R))$$

$$\leftarrow -0 + 0.1(-1 + 1*\max(0, 0) - -0) \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.21	-0.2	-0.11	0	0	0	0	0
Q(State, R)	-0.19	-0.1	-0.1		0	0		
Policy	R	R	R	-	L	L	-	-

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

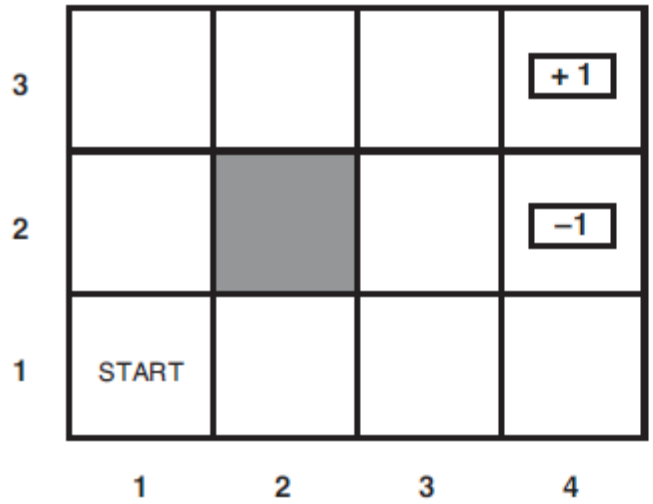
5, L → 4

Episode End

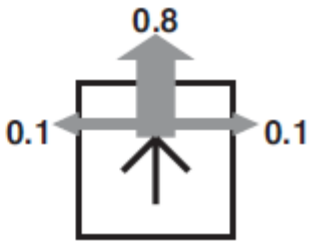
```
Q(5, L) ← Q(5,L) + 0.1( R(5) + 1*Q(4,_) - Q(5,L))

        ← -0 + 0.1( -1 + 1*-10 - -0)  ← -1.1
```

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.21	-0.2	-0.11	-10	-1.1	0	0	0
Q(State, R)	-0.19	-0.1	-0.1		0	0		
Policy	R	R	R	-	R	L	-	-

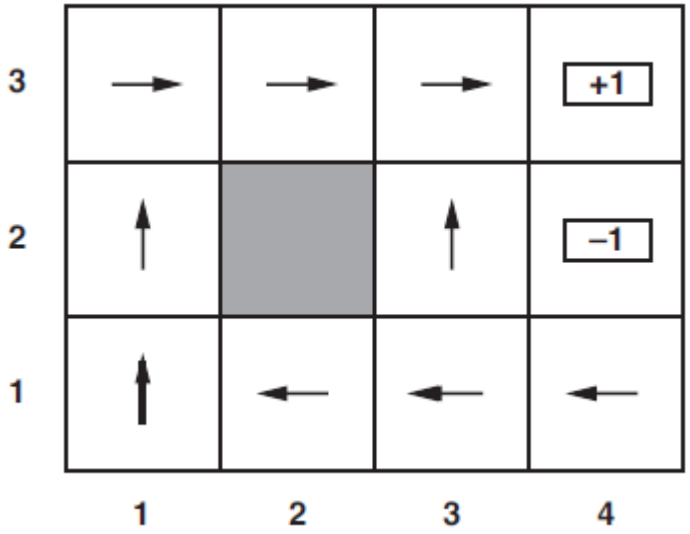
Another Problem



(a)



(b)



The agent plays 3 episodes in the environment starting from a random state in each episode

.

The rollouts were as follows:

E1: (1,1) → (1,2) → (1,3) → (2,3) → (3,3) → (4,3)

E2: (3,2) → (3,3) → (3,2) → (3,3) → (4,3)

E3: (2,3) → (3,3) → (3,2) → (4,3)

Use these 3 rollouts, do the following:

1. Apply “Direct Utility Estimation”.
2. Following “Adaptive Dynamic Programming”, estimate the transition model then write down the equations to evaluate the policy after receiving all the percept.
3. Assuming all utilities are initialized to 0, apply “TD-Learning” on the following rollouts.

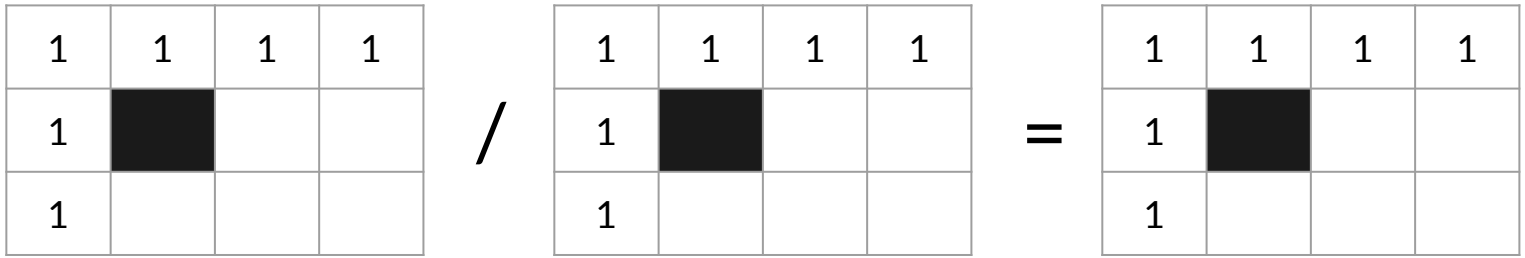
Assume all cells (except terminals) have 0 reward, and assume that the discount factor is 1.

Direct Utility Estimation

Episode 1

Step	1	2	3	4	5	6
State	(1,1)	(1,2)	(1,3)	(2,3)	(3,3)	(4,3)
Reward	0	0	0	0	0	1
Future Reward	1	1	1	1	1	1

Future Reward (State) = Reward (State) + discount * Future Reward (Next State)

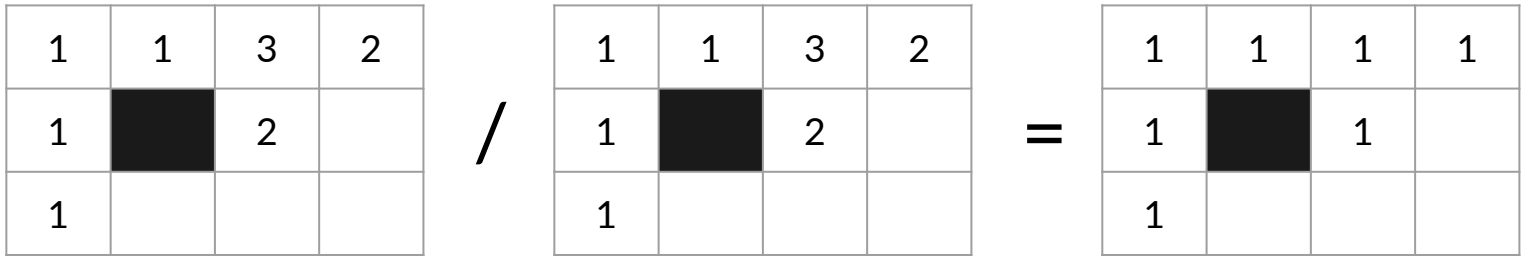


Direct Utility Estimation

Episode 2

Step	1	2	3	4	5
State	(3,2)	(3,3)	(3,2)	(3,3)	(4,3)
Reward	0	0	0	0	1
Future Reward	1	1	1	1	1

Future Reward (State) = Reward (State) + discount * Future Reward (Next State)



Direct Utility Estimation

Episode 3

Step	1	2	3	4
State	(2,3)	(3,3)	(3,2)	(4,2)
Reward	0	0	0	-1
Future Reward	-1	-1	-1	-1

Future Reward (State) = Reward (State) + discount * Future Reward (Next State)

1	0	2	2
1		1	-1
1			

Future Reward Sum

/

1	2	4	2
1		3	1
1			

Visit Count

=

RESULT

1	0	0.5	1
1		0.3	-1
1			

Estimated Utility

Adaptive Dynamic Programming

Transition Model Estimation

E1: (1,1) → (1,2) → (1,3) → (2,3) → (3,3) → (4,3)

State	Transition Model	Reward
(1,1)	(1,2) with p=1/1	0
(1,2)	(1,3) with p=1/1	0
(1,3)	(2,3) with p=1/1	0
(2,3)	(3,3) with p=1/1	0
(3,3)	(4,3) with p=1/1	0
(4,3)	Terminal	1

Adaptive Dynamic Programming

Transition Model Estimation

E2: (3,2) → (3,3) → (3,2) → (3,3) → (4,3)

State	Transition Model	Reward
(1,1)	(1,2) with p=1/1	0
(1,2)	(1,3) with p=1/1	0
(1,3)	(2,3) with p=1/1	0
(2,3)	(3,3) with p=1/1	0
(3,2)	(3,3) with p=2/2	0
(3,3)	(4,3) with p=2/3 & (3,2) with p=1/3	0
(4,3)	Terminal	1

Adaptive Dynamic Programming

Transition Model Estimation

E3: (2,3) → (3,3) → (3,2) → (4,2)

State	Transition Model	Reward
(1,1)	(1,2) with p=1/1	0
(1,2)	(1,3) with p=1/1	0
(1,3)	(2,3) with p=1/1	0
(2,3)	(3,3) with p=2/2	0
(3,2)	(3,3) with p=2/3 & (4,2) with p=1/3	0
(3,3)	(4,3) with p=2/4 & (3,2) with p=2/4	0
(4,3)	Terminal	1
(4,2)	Terminal	-1

Adaptive Dynamic Programming

Utility Estimation

$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s')$$

Equations

U(1,1) = U(1,2)
U(1,2) = U(1,3)
U(1,3) = U(2,3)
U(2,3) = U(3,3)
U(3,2) = (2/3) * U(3,3) + (1/3) * U(4,2)
U(3,3) = (2/4) * U(4,3) + (2/4) * U(3,2)
U(4,3) = 1
U(4,2) = -1

These equations are linear so they can be easily solved to get the estimated utilities.

State	Transition Model	Reward
(1,1)	(1,2) with p=1/1	0
(1,2)	(1,3) with p=1/1	0
(1,3)	(2,3) with p=1/1	0
(2,3)	(3,3) with p=2/2	0
(3,2)	(3,3) with p=2/3 & (4,2) with p=1/3	0
(3,3)	(4,3) with p=2/4 & (3,2) with p=2/4	0
(4,3)	Terminal	1
(4,2)	Terminal	-1

Temporal Difference Learning

Assume alpha = 0.1 (learning rate)

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma\, U^{\pi}(s') - U^{\pi}(s))$$

E1: (1,1) → (1,2) → (1,3) → (2,3) → (3,3) → (4,3)

U(1,1) ← U(1,1) + 0.1(R(1,1) + 1*U(1,2) - U(1,1))
← 0 + 0.1(0 + 1*0 - 0) ← 0

U(1,2) ← 0 + 0.1(0 + 1*0 - 0) ← 0

U(1,3) ← 0 + 0.1(0 + 1*0 - 0) ← 0

U(2,3) ← 0 + 0.1(0 + 1*0 - 0) ← 0

U(3,3) ← 0 + 0.1(0 + 1*1 - 0) ← 0.1

U(4,3) ← 1

0	0	0.1	1
0		0	0
0	0	0	0

Estimated Utility

I am treating the terminal state in a special manner here. As soon as I find a terminal state, I update its utility to be its reward since R(s) is deterministic given the state so I don't need to use the td-update. We could still choose to apply the rule (U(4,3) will be 0.1 and U(3,3) will be 0) but the convergence will be slower. These are assumptions and could be handled differently since it is not specified in the book.

Temporal Difference Learning

Assume alpha = 0.1 (learning rate)

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma\, U^{\pi}(s') - U^{\pi}(s))$$

E2: (3,2) → (3,3) → (3,2) → (3,3) → (4,3)

U(3,2) ← 0 + 0.1(0 + 1*0.1 - 0) ← 0.01

U(3,3) ← 0.1 + 0.1(0 + 1*0.01 - 0.1) ← 0.091

U(3,2) ← 0.01 + 0.1(0 + 1*0.091 - 0.01) ← 0.0181

U(3,3) ← 0.091 + 0.1(0 + 1*1 - 0.091) ← 0.1819 U(4,3) ← 1

TA Note: Numbers calculated in a hurry so there may be some mistakes.

0	0	0.1819	1
0		0.0181	0
0	0	0	0

Estimated Utility

Temporal Difference Learning

Assume alpha = 0.1 (learning rate)

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha(R(s) + \gamma\, U^{\pi}(s') - U^{\pi}(s))$$

E3: (2,3) → (3,3) → (3,2) → (4,2)

U(2,3) ← 0 + 0.1(0 + 1*0.1819 - 0) ← 0.01819

U(3,3) ← 0.1819 + 0.1(0 + 1*0.0181 - 0.1819) ← 0.16552

U(3,2) ← 0.0181 + 0.1(0 + 1*-1 - 0.0181) ← -0.08371

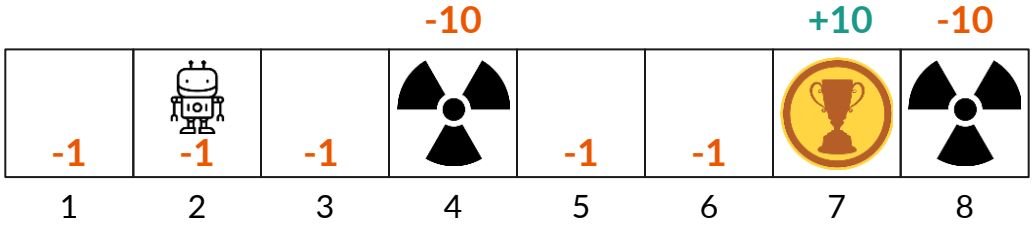
U(4,2) ← -1

0	0.01819	0.16552	1
0		-0.08371	-1
0	0	0	0

Estimated Utility

RESULT

Continued



Assume each episode start from at state 2.

Assume all cells (except terminals) have -1 reward, and assume that the discount factor is 1.

Since we are going to sample the rollout by our self, we need the transition model:

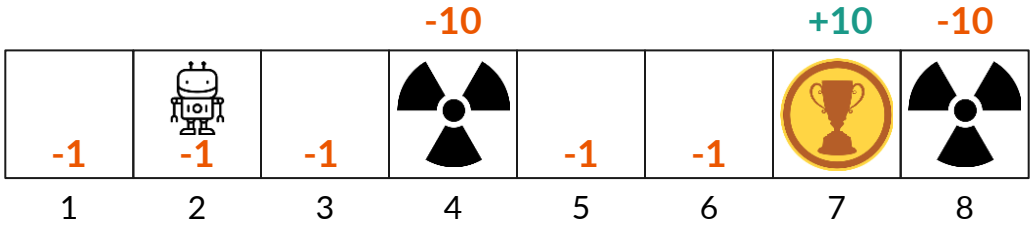
- Action L: 100% Move left one step
- Action R: 20% Move right one step, 80% Move right 2 steps.

Note: In real application, we rarely know the transition model. Instead, we sample the transition by interacting with an environment.

- **Apply Tabular SARSA to the aforementioned environment problem. Assume that the policy will pick L if the Q-values is indecisive.**

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

2, L → 1, L

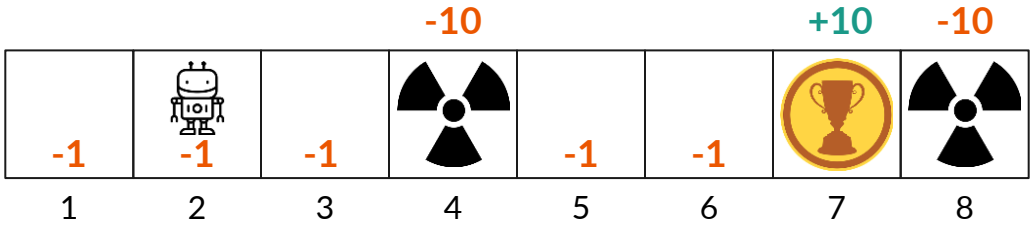
$$Q(2, L) \leftarrow Q(2,L) + 0.1(\ R(2) + 1*Q(1,L) - Q(2,L))$$

$$\leftarrow 0 + 0.1(\ -1 + 1*0 - 0) \ \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	0	-0.1	0	0	0	0	0	0
Q(State, R)	0	0	0		0	0		
Policy	L	R	L	L	L	L	L	L

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

1, L → 1, L

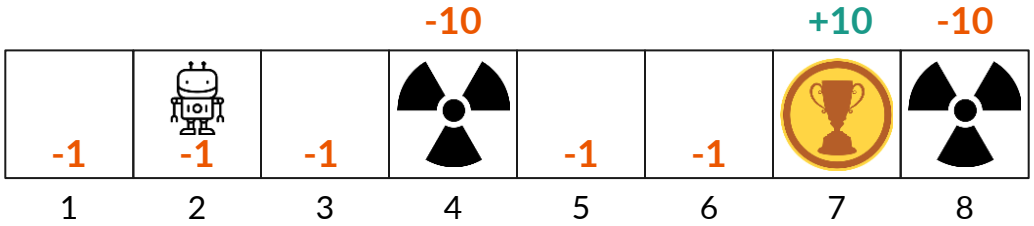
$$Q(1, L) \leftarrow Q(1,L) + 0.1(\ R(1) + 1*Q(1,L) - Q(1,L))$$

$$\leftarrow 0 + 0.1(\ -1 + 1*0 - 0) \ \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.1	0	0	0	0	0	0
Q(State, R)	0	0	0		0	0		
Policy	R	R	L	-	L	L	-	-

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

1, L → 1, R

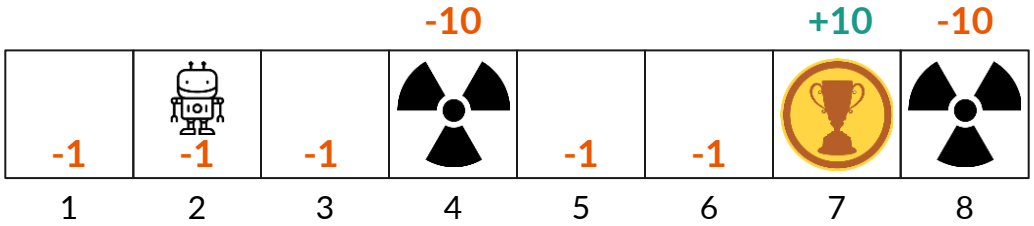
$$Q(1, L) \leftarrow Q(1,L) + 0.1(R(1) + 1*Q(1,R) - Q(1,L))$$

$$\leftarrow 0 + 0.1(-1 + 1*0 - 0) \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.1	0	0	0	0	0	0
Q(State, R)	0	0	0		0	0		
Policy	R	R	L	-	L	L	-	-

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

1,R → 2,R

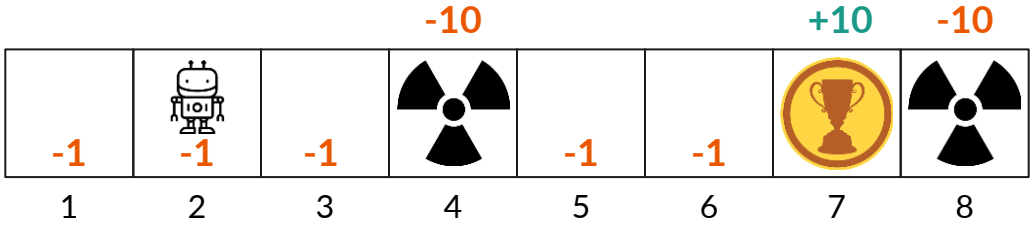
$$Q(1, L) \leftarrow Q(1,L) + 0.1(\ R(1) + 1*Q(2,R) - Q(1,L))$$

$$\leftarrow 0 + 0.1(\ -1 + 1*0 - 0) \ \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.1	0	0	0	0	0	0
Q(State, R)	-0.1	0	0		0	0		
Policy	L	R	L	-	L	L	-	-

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

2, R → 3, L

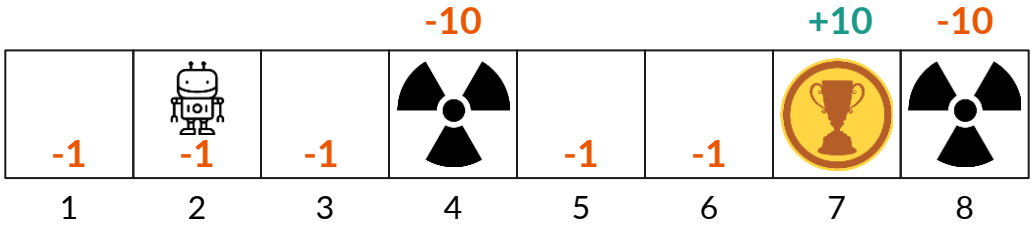
$$Q(2,\ R) \leftarrow Q(2,R) + 0.1(\ R(2) + 1*Q(3,L) - Q(2,R))$$

$$\leftarrow 0 + 0.1(\ -1 + 1*0 - 0) \ \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.1	0	0	0	0	0	0
Q(State, R)	-0.1	-0.1	0		0	0		
Policy	L	L	L	-	L	L	-	-

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

3,L → 2,L

$$Q(3, L) \leftarrow Q(3,L) + 0.1(\, R(3) + 1*Q(2,L) - Q(3,L))$$

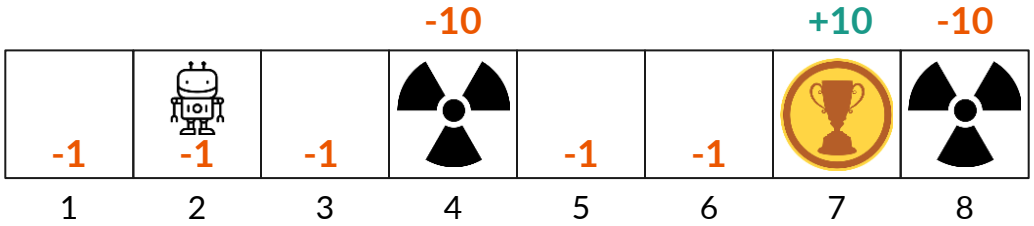
$$\leftarrow 0 + 0.1(\, -1 + 1*-0.1 - 0) \leftarrow -0.11$$

→

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.1	-0.11	0	0	0	0	0
Q(State, R)	-0.1	-0.1	0		0	0		
Policy	L	L	R	-	L	L	-	-

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

2, L → 1, L

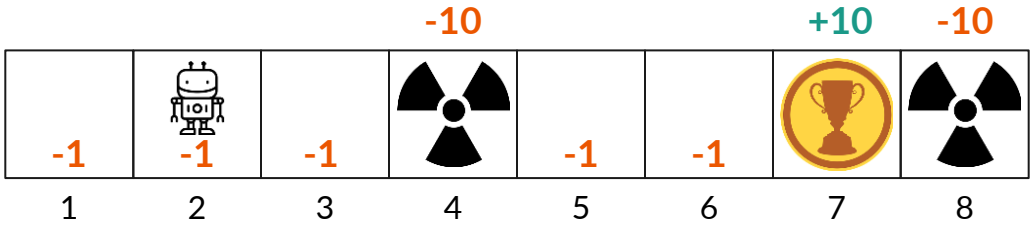
$$Q(2, L) \leftarrow Q(2,L) + 0.1(\, R(2) + 1*Q(1,L) - Q(2,L))$$

$$\leftarrow -0.1 + 0.1(\, -1 + 1*-0.1 - -0.1) \leftarrow -0.2$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.1	-0.2	-0.11	0	0	0	0	0
Q(State, R)	-0.1	-0.1	0		0	0		
Policy	L	R	R	-	L	L	-	-

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

1, L → 1, L

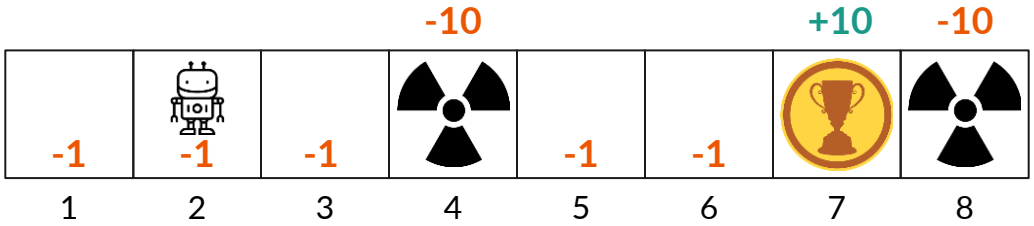
$$Q(1, L) \leftarrow Q(1,L) + 0.1(\ R(1) + 1*Q(1,L) - Q(1,L))$$

$$\leftarrow -0.1 + 0.1(\ -1 + 1*-0.1 - -0.1) \leftarrow -0.2$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.2	-0.2	-0.11	0	0	0	0	0
Q(State, R)	-0.1	-0.1	0		0	0		
Policy	R	R	R	-	L	L	-	-

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

1, L → 1, R

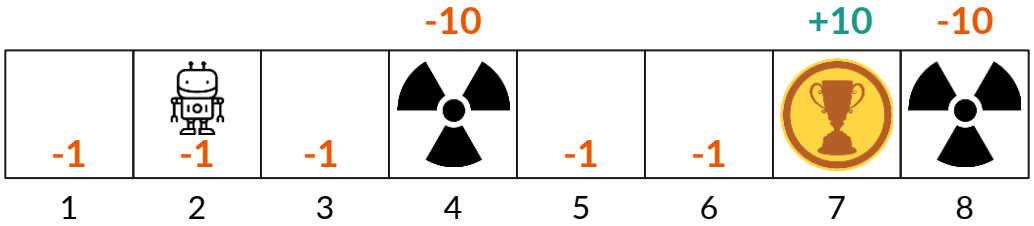
$$Q(1, L) \leftarrow Q(1,L) + 0.1(\, R(1) + 1*Q(1,R) - Q(1,L))$$

$$\leftarrow -0.1 + 0.1(\, -1 + 1*-0.1 - -0.1) \leftarrow -0.2$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.2	-0.2	-0.11	0	0	0	0	0
Q(State, R)	-0.1	-0.1	0		0	0		
Policy	R	R	R	-	L	L	-	-

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

`1,R → 3,R`

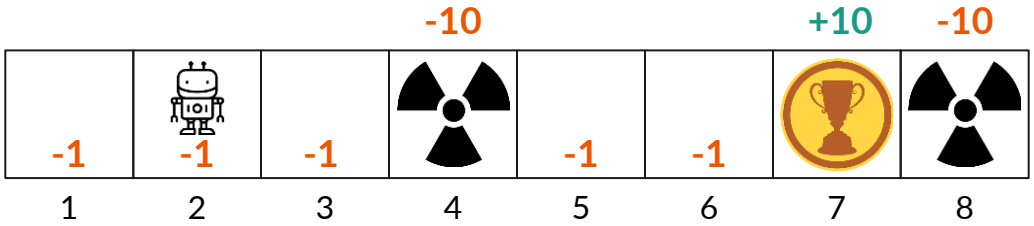
$$Q(1, R) \leftarrow Q(1,R) + 0.1(\, R(1) + 1*Q(3,R) - Q(1,R))$$

$$\leftarrow -0.1 + 0.1(\, -1 + 1*0 - -0.1) \leftarrow -0.19$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.21	-0.2	-0.11	0	0	0	0	0
Q(State, R)	-0.19	-0.1	0		0	0		
Policy	R	R	R	-	L	L	-	-

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

3,R → 5,L

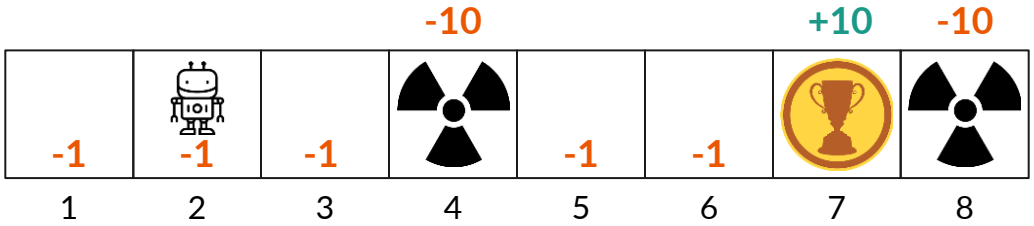
$$Q(3,\ R) \leftarrow Q(3,R) + 0.1(\ R(3) + 1*Q(5,L) - Q(3,R))$$

$$\leftarrow -0 + 0.1(\ -1 + 1*0 - -0) \leftarrow -0.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.21	-0.2	-0.11	0	0	0	0	0
Q(State, R)	-0.19	-0.1	-0.1		0	0		
Policy	R	R	R	-	L	L	-	-

SARSA:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma\, Q(s',a') - Q(s,a))$$

Transition

5, L → 4

Episode End

$$Q(5,\ L) \leftarrow Q(5,L) + 0.1(\ R(5) + 1*Q(4,\_) - Q(5,L))$$

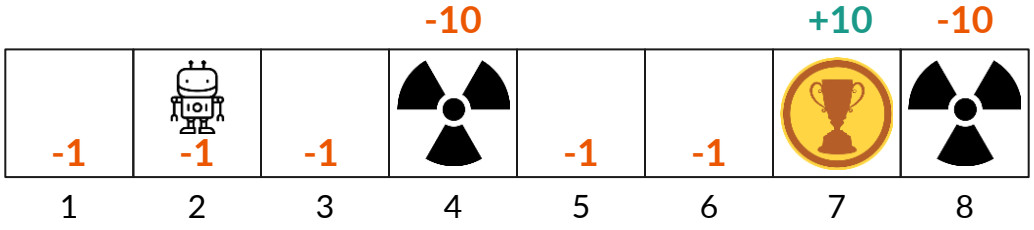
$$\leftarrow -0 + 0.1(\ -1 + 1*-10 - -0) \leftarrow -1.1$$

State	1	2	3	4	5	6	7	8
Q(State, L)	-0.21	-0.2	-0.11	-10	-1.1	0	0	0
Q(State, R)	-0.19	-0.1	-0.1		0	0		
Policy	R	R	R	-	R	L	-	-

21.4 Write out the parameter update equations for TD learning with

$$\hat{U}(x, y) = \theta_0 + \theta_1 x + \theta_2 y + \theta_3 \sqrt{(x - x_g)^2 + (y - y_g)^2} \, .$$

$$
\begin{aligned}
\theta_0 &\leftarrow \theta_0 + \alpha \left(u_j(s) - \hat{U}_\theta(s) \right) , \\
\theta_1 &\leftarrow \theta_1 + \alpha \left(u_j(s) - \hat{U}_\theta(s) \right) x , \\
\theta_2 &\leftarrow \theta_2 + \alpha \left(u_j(s) - \hat{U}_\theta(s) \right) y , \\
\theta_3 &\leftarrow \theta_3 + \alpha \left(u_j(s) - \hat{U}_\theta(s) \right) \sqrt{(x - x_g)^2 + (y - y_g)^2} .
\end{aligned}
$$



Assume each episode start from at state 2.

Assume all cells (except terminals) have -1 reward, and assume that the discount factor is 1.

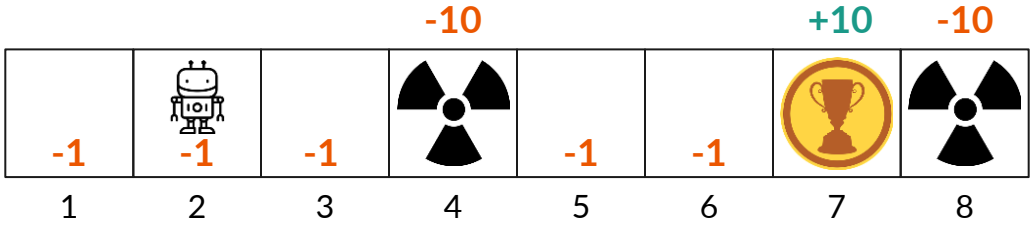
Since we are going to sample the rollout by our self, we need the transition model:

- Action L: 100% Move left one step
- Action R: 20% Move right one step, 80% Move right 2 steps.

Note: In real application, we rarely know the transition model. Instead, we sample the transition by interacting with an environment.

- **Apply Approximate Q-Learning to the aforementioned environment problem. Assume that the policy will pick L if the Q-values is indecisive**

$$Q(s, a) = W_{1a}x + W_{0a}$$

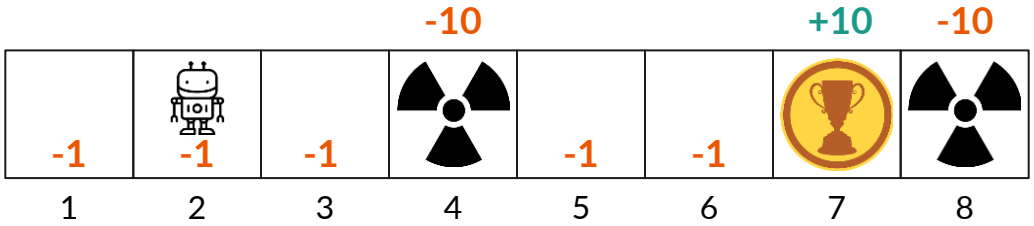


Assume each episode start from at state 2.

State	W_1	W_0	1	2	3	4	5	6	7	8
Q(State, L)	0	0	0	0	0	0	0	0	0	0
Q(State, R)	0	0	0	0	0	0	0	0	0	0
Policy			L	R	L	L	L	L	L	L

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

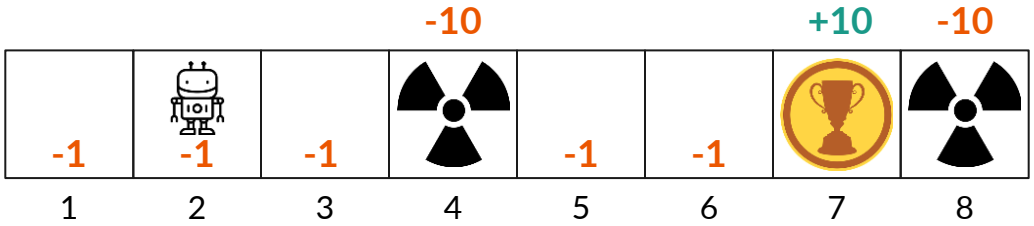
2, L → 1

```
Error ← R(2) + 1*max(Q(1,L), Q(1,R)) - Q(2,L)

      ← -1 + 1*max(0, 0) - 0 ← -1
W1 ← W1 + 0.1*Error*X ← 0 + 0.1*-1*2 ← -0.2
W0 ← W0 + 0.1*Error ← 0 + 0.1*-1 ← -0.1
```

State	W_1	W_0	1	2	3	4	5	6	7	8
Q(State, L)	-0.2	-0.1	-0.3	-0.5	-0.7	-0.9	-1.1	-1.3	-1.5	-1.7
Q(State, R)	0	0	0	0	0	0	0	0	0	0
Policy			L	R	L	L	L	L	L	L

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

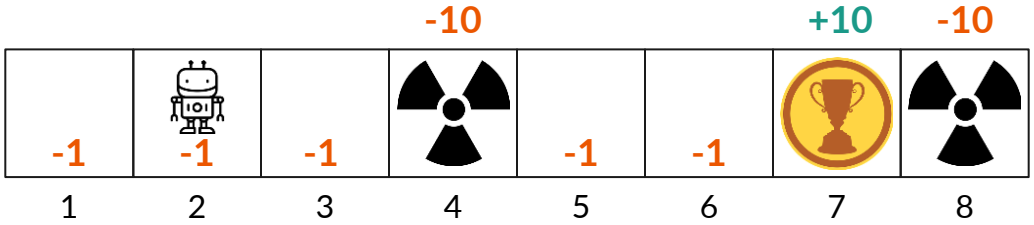
1, R → 2

```
Error ← R(1) + 1*max(Q(2,L), Q(2,R)) - Q(1,R)

      ← -1 + 1*max(0, 0) - 0 ← -1
W1 ← W1 + 0.1*Error*X ← 0 + 0.1*-1*1 ← -0.1
W0 ← W0 + 0.1*Error ← 0 + 0.1*-1 ← -0.1
```

State	W_1	W_0	1	2	3	4	5	6	7	8
Q(State, L)	-0.2	-0.1	-0.3	-0.5	-0.7	-0.9	-1.1	-1.3	-1.5	-1.7
Q(State, R)	-0.1	-0.1	-0.2	-0.3	-0.4	-0.5	-0.6	-0.6	-0.8	-0.9
Policy			R	R	R	R	R	R	R	R

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

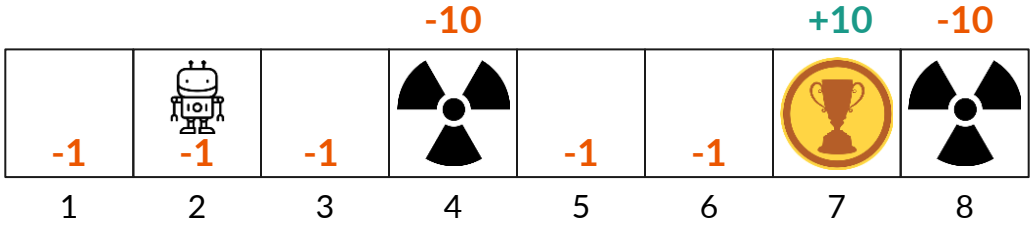
2, R → 3

```
Error ← R(2) + 1*max(Q(3,L), Q(3,R)) - Q(2,R)

      ← -1 + 1*max(-0.7, -0.4) - -0.3 ← -1.1
W1 ← W1 + 0.1*Error*X ← -0.1 + 0.1*-1.1*2 ← -0.32
W0 ← W0 + 0.1*Error ← -0.1 + 0.1*-1.1 ← -0.21
```

State	W_1	W_0	1	2	3	4	5	6	7	8
Q(State, L)	-0.2	-0.1	-0.3	-0.5	-0.7	-0.9	-1.1	-1.3	-1.5	-1.7
Q(State, R)	-0.32	-0.21	-0.53	-0.85	-1.17	-1.49	-1.81	-2.13	-2.45	-2.77
Policy			L	L	L	L	L	L	L	L

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

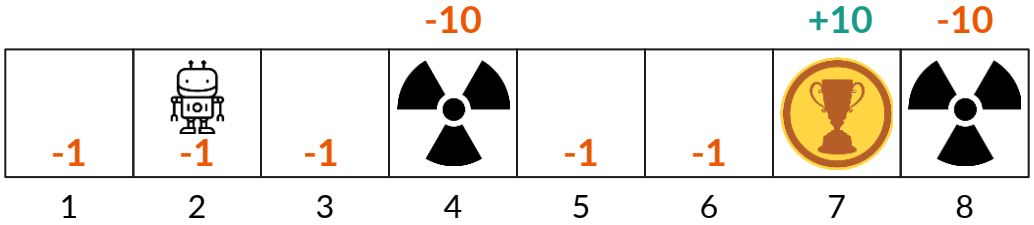
3, L → 2

```
Error ← R(3) + 1*max(Q(2,L), Q(2,R)) - Q(3,L)
      ← -1 + 1*max(-0.5, -0.85) - -0.7 ← -0.8
W1 ← W1 + 0.1*Error*X ← -0.2 + 0.1*-0.8*3 ← -0.44
W0 ← W0 + 0.1*Error ← -0.1 + 0.1*-0.8 ← -0.18
```

State	W_1	W_0	1	2	3	4	5	6	7	8
Q(State, L)	-0.44	-0.18	-0.62	-1.06	-1.5	-1.94	-2.38	-2.82	-3.26	-3.7
Q(State, R)	-0.32	-0.21	-0.53	-0.85	-1.17	-1.49	-1.81	-2.13	-2.45	-2.77
Policy			R	R	R	R	R	R	R	R

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

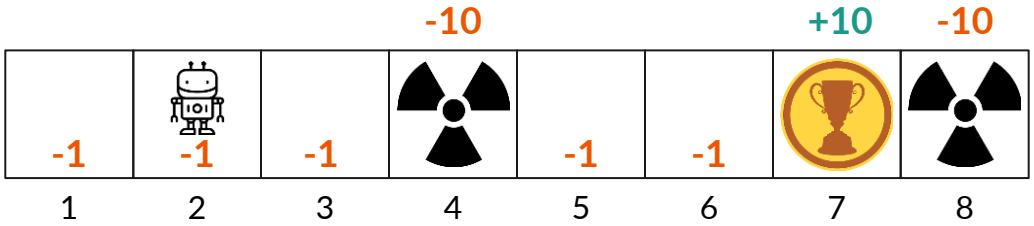
2, R → 4

```
Error ← R(2) + 1*max(Q(4,L), Q(4,R)) - Q(2,R)
      ← -1 + 1*max(-1.94, -1.49) - -0.85 ← -1.64
W1 ← W1 + 0.1*Error*X ← -0.32 + 0.1*-1.64*2 ← -0.648
W0 ← W0 + 0.1*Error ← -0.21 + 0.1*-1.64 ← -0.374
```

State	W_1	W_0	1	2	3	4	5	6	7	8
Q(State, L)	-0.44	-0.18	-0.62	-1.06	-1.5	-1.94	-2.38	-2.82	-3.26	-3.7
Q(State, R)	-0.648	-0.374	-1.022	-1.67	-2.318	-2.966	-3.614	-4.262	-4.91	-5.558
Policy			L	L	L	L	L	L	L	L

Q Learning:

Assume alpha = 0.1



$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Transition

4 (End)

```
Error ← R(4) - Q(4,L) ← -10 - -1.94 ← -8.06
W1 ← W1 + 0.1*Error*X ← -0.44 + 0.1*-8.06*4
W0 ← W0 + 0.1*Error ← -0.18 + 0.1*-8.06
Error ← R(4) - Q(4,R) ← -10 - -2.966 ← -7.034
W1 ← W1 + 0.1*Error*X ← -0.648 + 0.1*-7.034*4
W0 ← W0 + 0.1*Error ← -0.374 + 0.1*-7.034
```

Since 4 is a terminal state, we can do one of the following:
1- Assume the update will affect both action L & R which is what we did above.
2- Have separate weights for terminal states.
3- Change the representation of the MDP so that the reward function is R'(S,A,S') = R(S'). In such case the values for terminals "max(Q(s,a))" can be calculated to be 0 during the update.