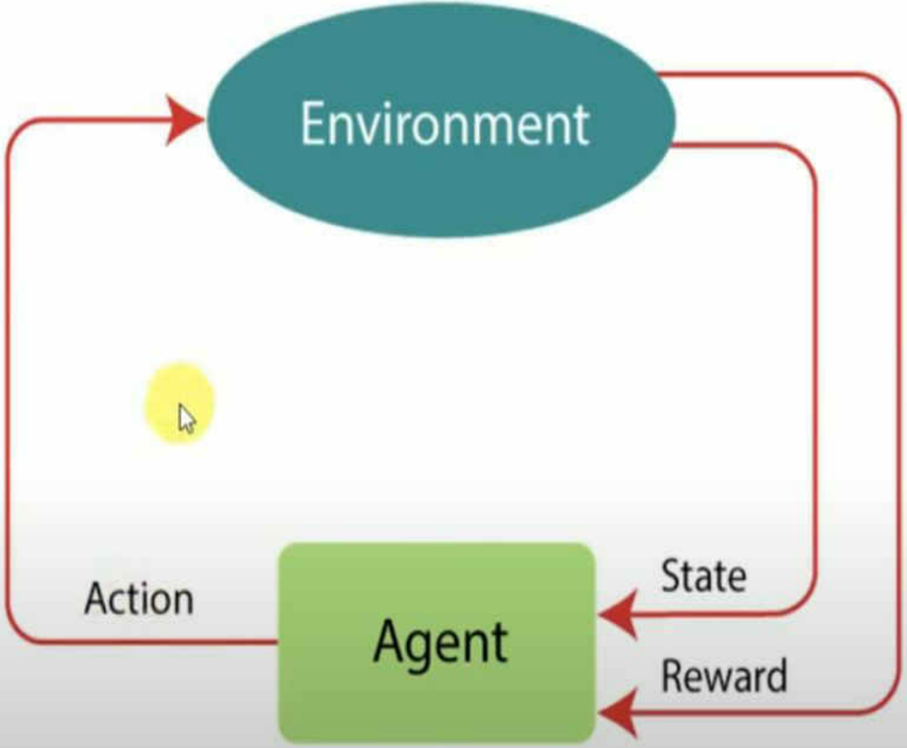
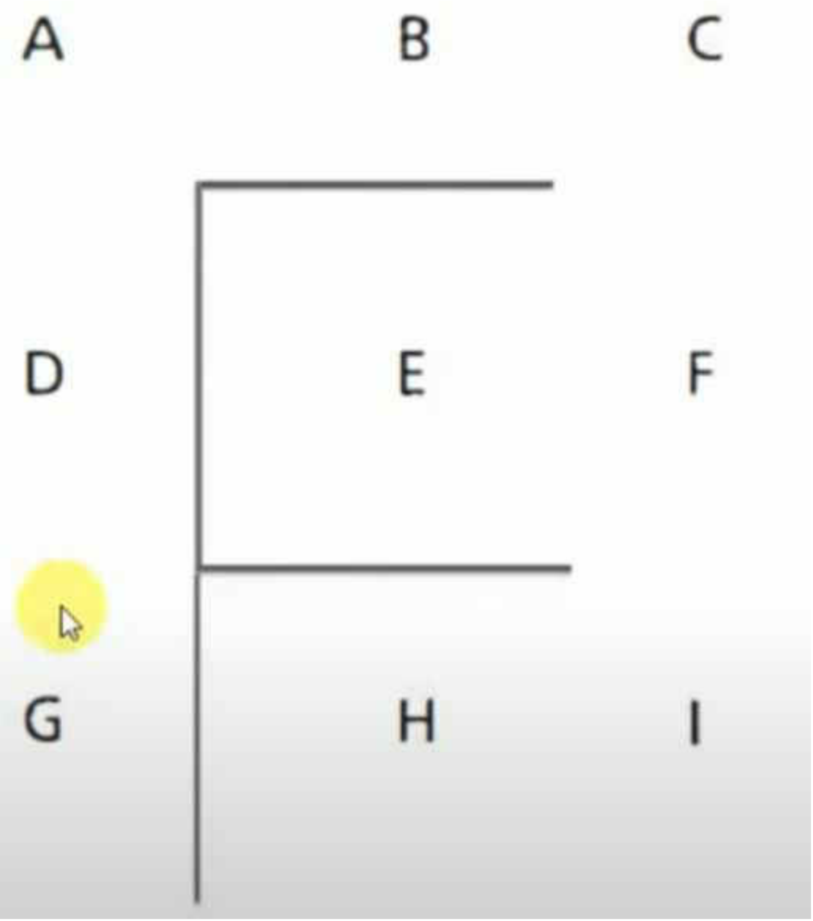
- Reinforcement Learning is a feedback-based Machine learning Approach here an agent learns to which actions to perform by looking at the environment and the results of actions.
- For each correct action, the agent gets positive feedback, and for each incorrect action, the agent gets negative feedback or penalty.



- What are the situations where reinforcement learning can be used?
- Consider the following grid game shown in Figure , where a robot can move.
- Assume the starting node is E and goal node is G, the game is about finding the shortest path from starting to goal state.

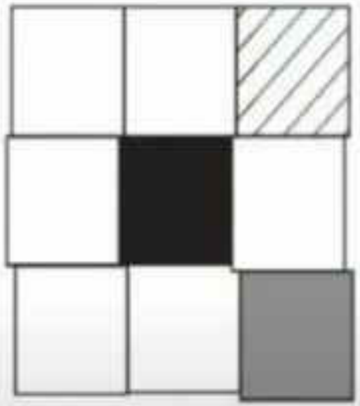


- The agent interacts with the environment and identifies the possible actions he can perform.
- The primary goal of an agent in reinforcement learning is to perform actions by looking at the environment and get the maximum positive rewards.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- Since there is no labeled data, so the agent is bound to learn by its experience only.
- Reinforcement Learning is used to solve specific type of problem where decision making is sequential, and the goal is long-term, such as game-playing, robotics, etc.

- There are two types of reinforcement learning - **positive and negative.**
- **Positive reinforcement learning** is a recurrence of behaviour due to positive rewards.
- Rewards increase strength and the frequency of a specific behaviour.
- This encourages to execute similar actions that yield maximum reward.
- Similarly, in **negative reinforcement learning**, negative rewards are used as a deterrent to weaken the behaviour and to avoid it.
- Rewards decreases strength and the frequency of a specific behaviour.

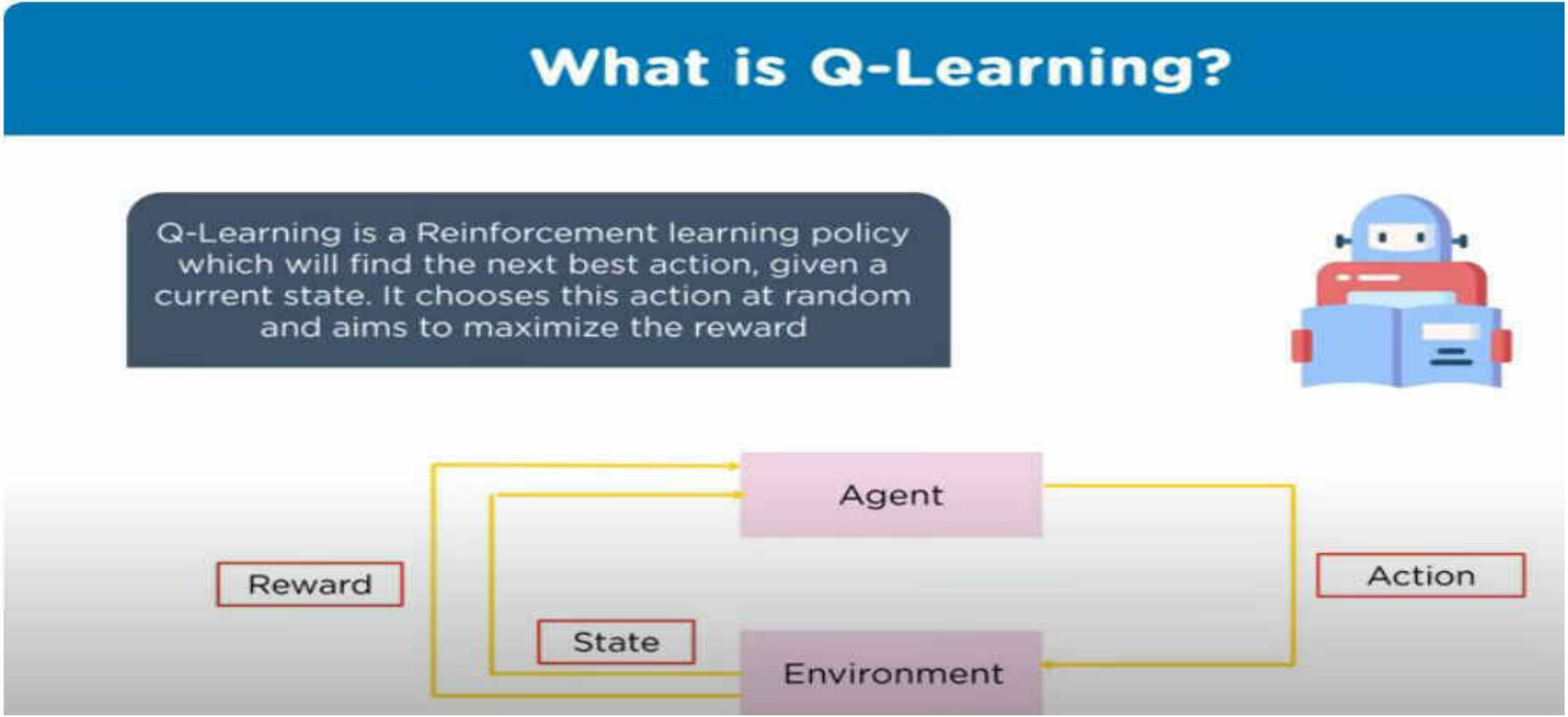
- In a **maze game**, there may a danger spot that may lead to loss.
- Negative rewards can be designed for such spots so that the agent does not visit that spot.
- Positive and negative rewards are simulated in reinforcement learning, say +10 for positive reward and -10 for some danger or negative reward.
- Reinforcement learning is an example of semi-supervised learning technique and is used to model sequential decision-making process.

- Consider another grid game as shown in Figure.
- In this grid game, the grey tile indicates the danger, black is a block and the tile with diagonal lines is the goal.
- The aim is to start, say from bottom-left grid, using the actions left, right, top and bottom to reach the goal state.
- Reinforcement learning is highly suitable for solving problems like this, especially the ones with uncertainty.



- Reinforcement is not suitable for environments where complete information is available.
- For example, the problems like object detection, face recognition, fraud detction can be better solved using a classifier than by reinforcement learning.

Q learning


What is Q-Learning?
Q-Learning is a Reinforcement learning policy which will find the next best action, given a current state. It chooses this action at random and aims to maximize the reward
Agent
Reward
Action
State
Environment

Q-learning is a machine learning approach that enables a model to iteratively learn and improve over time by taking the correct action. Q-learning is a type of reinforcement learning. With reinforcement learning, a machine learning model is trained to mimic the way animals or children learn.

How does Q-learning work?

Q-learning models operate in an iterative process that involves multiple components working together to help train a model. The iterative process involves the agent learning by exploring the environment and updating the model as the exploration continues. The multiple components of Q-learning include the following:

- **Agents.** The agent is the entity that acts and operates within an environment.
- **States.** The state is a variable that identifies the current position in an environment of an agent.
- **Actions.** The action is the agent's operation when it is in a specific state.
- **Rewards.** A foundational concept within reinforcement learning is the concept of providing either a positive or a negative response for the agent's actions.
- **Episodes.** An episode is when an agent can no longer take a new action and ends up terminating.
- **Q-values.** The Q-value is the metric used to measure an action at a particular state.

Q-Learning Algorithm - Reinforcement learning

Q learning algorithm

For each s, a initialize the table entry $\hat{Q}(s, a)$ to zero.

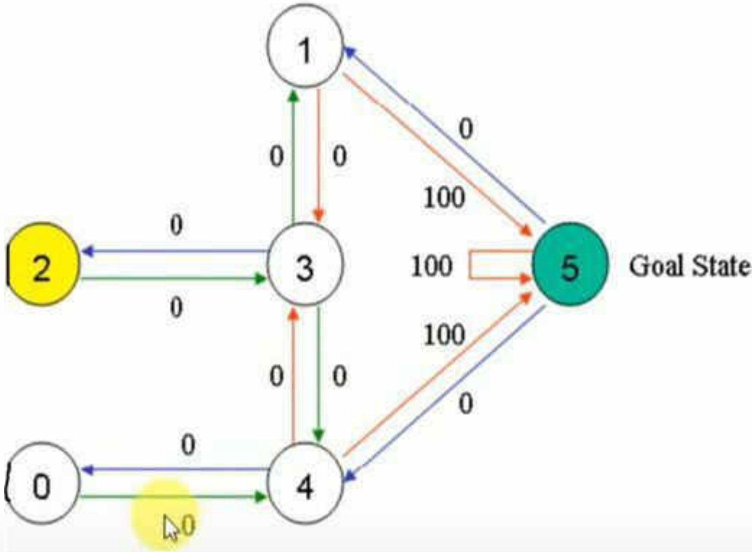
Observe the current state s

Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$



Q learning algorithm

For each s, a initialize the table entry $\hat{Q}(s,a)$ to zero

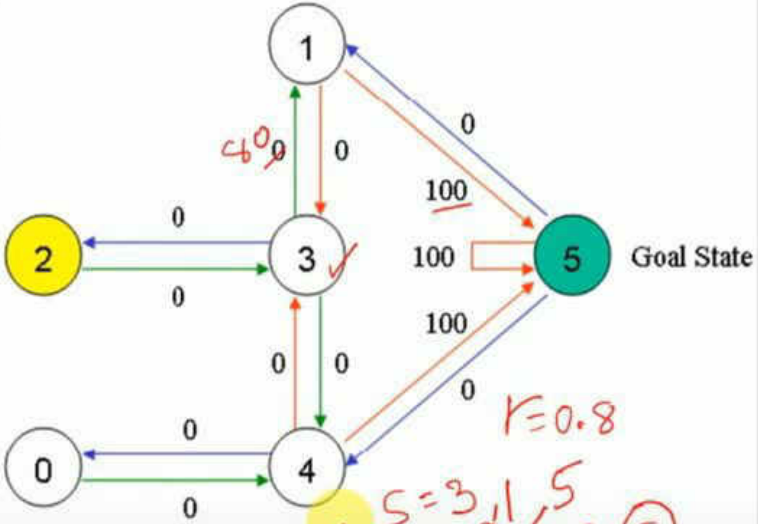
Observe the current state s

Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s,a)$ as follows:

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

- $s \leftarrow s'$



$\gamma = 0.8$

$s = 3, 1, 5$

$a = 2, 1, 4, 3, 5$

$r = 0, 100$

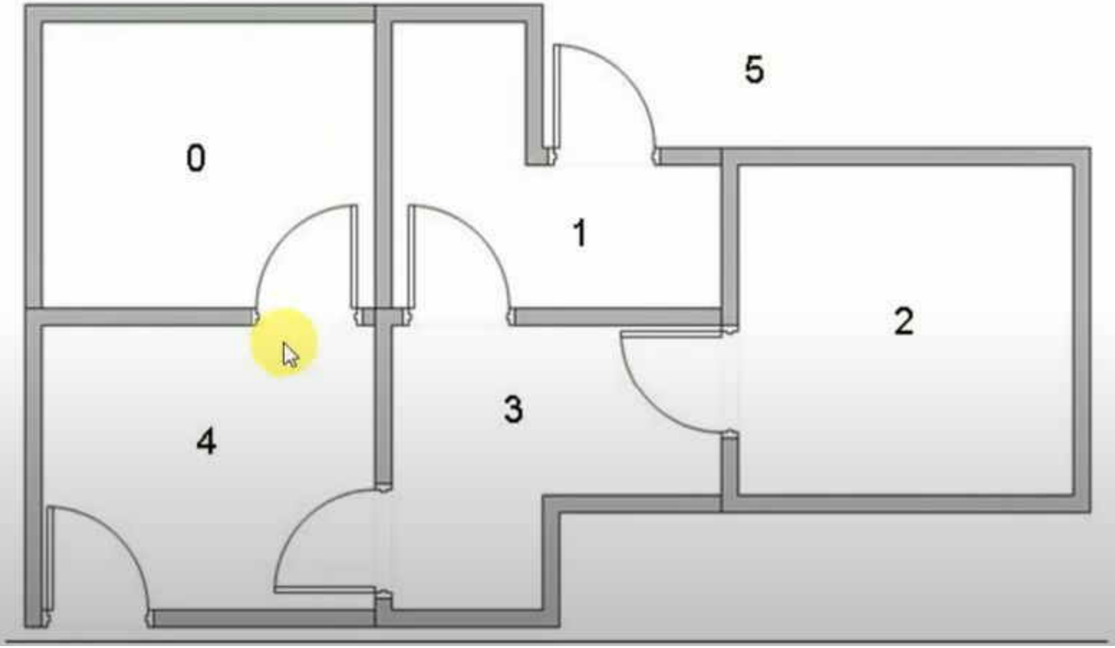
$s' = 1, 5$

$\hat{Q}(3,1) = 0 + 0.8 \max(0, 100) = 80$

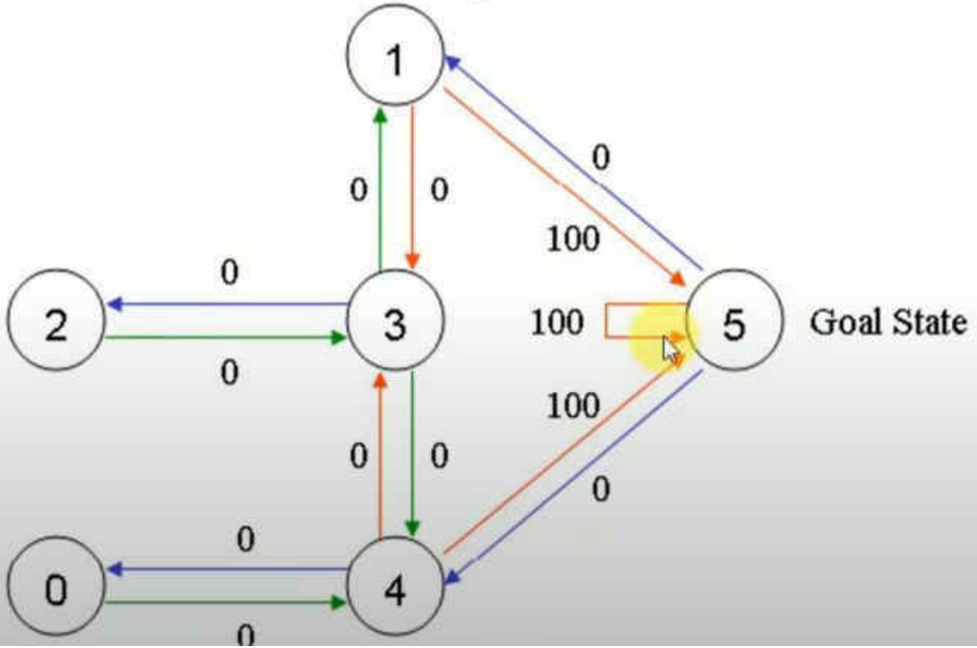
$\hat{Q}(1,5) = 100 + 0.8 \max(0, 0) = 100$

Example

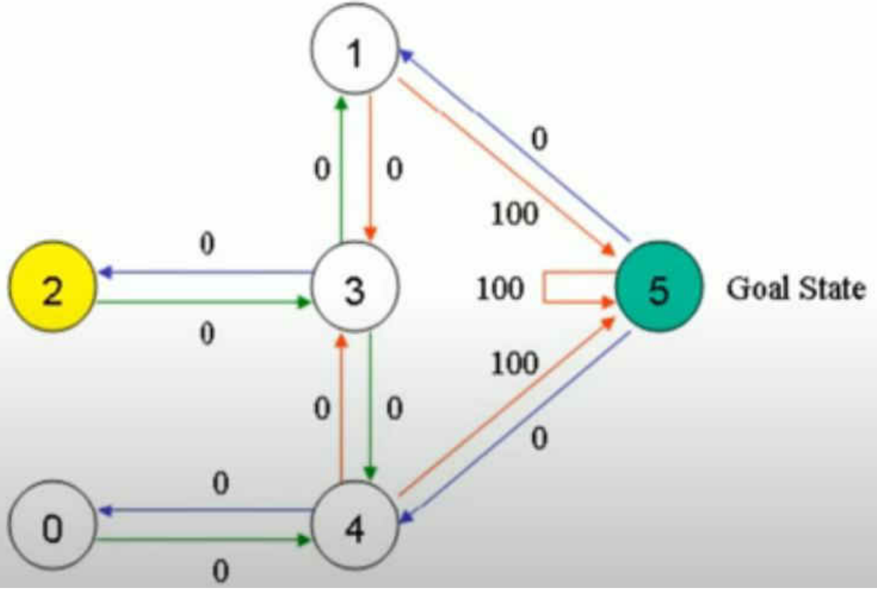
Suppose we have 5 rooms in a building connected by doors as shown in the figure below. We'll number each room 0 through 4. The outside of the building can be thought of as one big room (5). Notice that doors 1 and 4 lead into the building from room 5 (outside).



- The goal room is number 5
- The doors that lead immediately to the goal have an instant reward of 100. Other doors not directly connected to the target room have zero reward.
- Each arrow contains an instant reward value, as shown below:



- We can put the state diagram and the instant reward values into the following reward table, "matrix R". The -1's in the table represent null values (i.e.; where there isn't a link between nodes). For example, State 0 cannot go to State 1.



$$
R = \begin{array}{c} \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\text{Action} \\
\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix}
-1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & 0 & -1 & 100 \\
-1 & -1 & -1 & 0 & -1 & -1 \\
-1 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & -1 & 0 & -1 & 100 \\
-1 & 0 & -1 & -1 & 0 & 100
\end{bmatrix}
\end{array}
$$

- Learning rate = 0.8 and the initial state as Room 1.
- Initialize matrix Q as a zero matrix:

$$R = \begin{array}{c|c} & \textbf{Action} \\ \textbf{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{cc} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

- Look at the second row (state 1) of matrix R.
- There are two possible actions for the current state 1: go to state 3, or go to state 5.
- By random selection, we select to go to 5 as our action.

$$
R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}
$$

- Now let's imagine what would happen if our agent were in state 5.
- Look at the sixth row of the reward matrix R (i.e. state 5).
- It has 3 possible actions: go to state 1, 4 or 5.
- Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]
- Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100

$$
\begin{array}{cc}
 & \text{Action} \\
\text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array}
\end{array}
$$

$$
R = \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{bmatrix}
-1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & 0 & -1 & 100 \\
-1 & -1 & -1 & 0 & -1 & -1 \\
-1 & 0 & 0 & -1 & 0 & -1 \\
0 & -1 & -1 & 0 & -1 & 100 \\
-1 & 0 & -1 & -1 & 0 & 100
\end{bmatrix}
\qquad
Q = \begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 100 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

- The next state, 5, now becomes the current state.
- Because 5 is the goal state, we've finished one episode.
- Our agent's brain now contains an updated matrix Q as:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

- Now we imagine that we are in state 1 (next state).
- Look at the second row of reward matrix R (i.e. state 1).
- It has 2 possible actions: go to state 3 or state 5.
- Then, we compute the Q value:
- Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]
- Q(3, 1) = R(3, 1) + 0.8 * Max[Q(1, 3), Q(1, 5)] = 0 + 0.8 * Max(0, 100) = 80

$$
\begin{array}{ccc}
 & & \text{Action} \\
 & \text{State} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
R= & \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix}
\end{array}
$$

$$
\begin{array}{ccc}
 & & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
Q= & \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\end{array}
$$

- If our agent learns more through further episodes, it will finally reach convergence values in matrix Q like:

$$
Q=\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 80 & 0 \\
0 & 0 & 0 & 64 & 0 & 100 \\
0 & 0 & 0 & 64 & 0 & 0 \\
0 & 80 & 51 & 0 & 80 & 0 \\
64 & 0 & 0 & 64 & 0 & 100 \\
0 & 80 & 0 & 0 & 80 & 100
\end{bmatrix}
\end{array}
$$

- Tracing the best sequences of states is as simple as following the links with the highest values at each state.