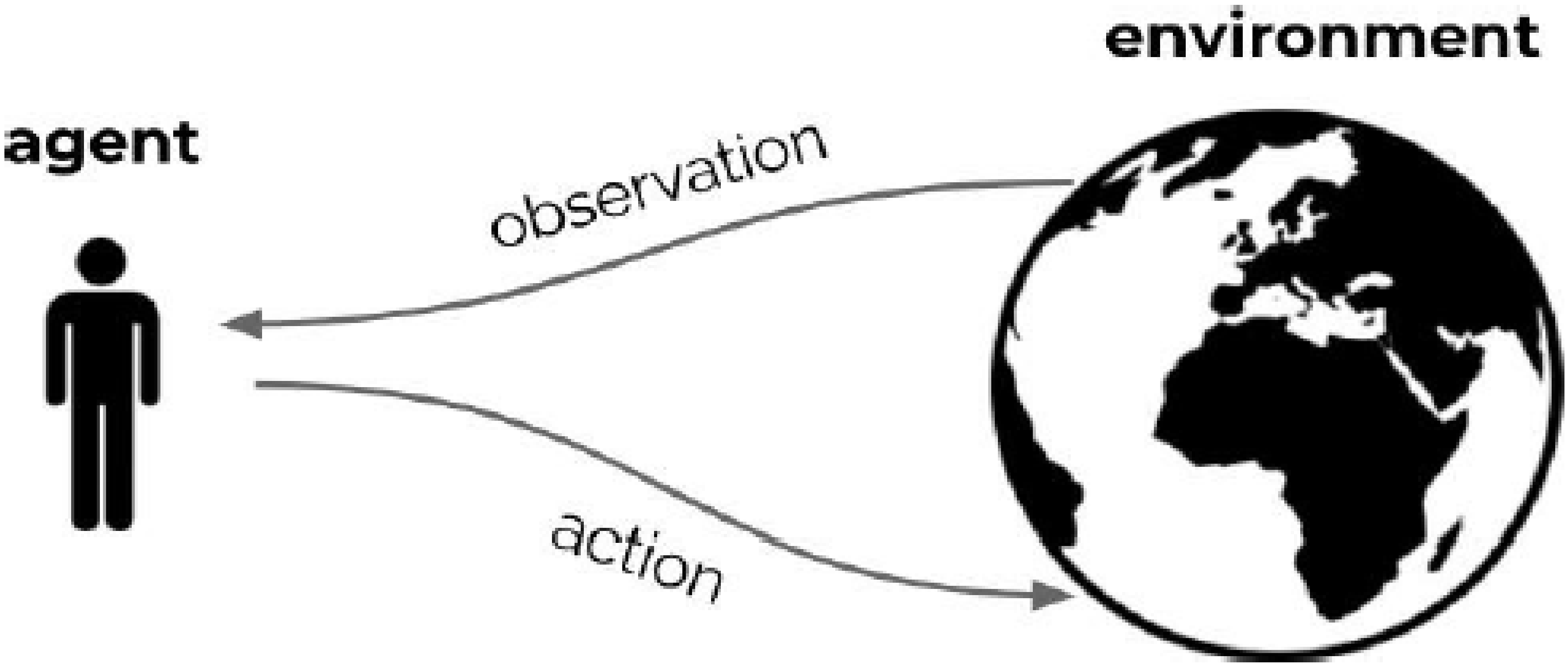
Reinforcement learning



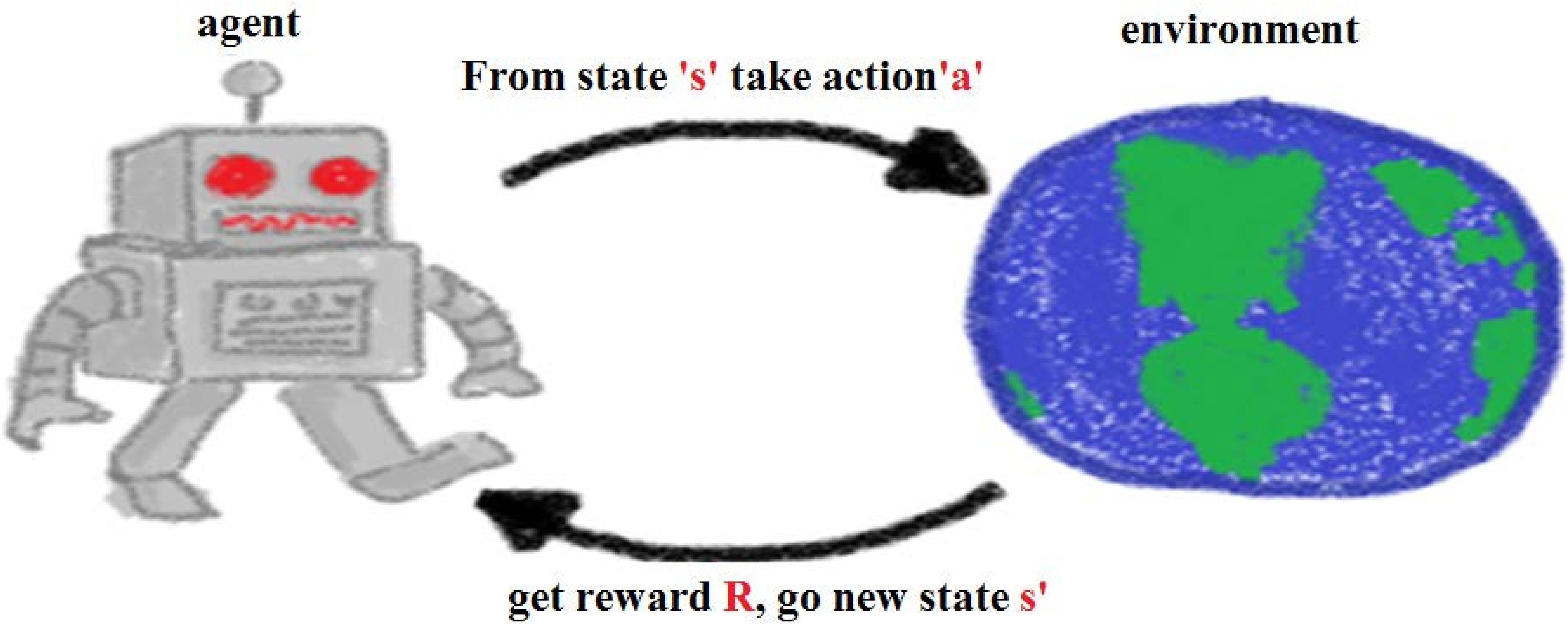
The term *reinforcement learning* comes from cognitive science and psychology and it describes the learning system of carrot and stick

Based on a motivational tactic that uses a reward and punishment system to encourage improved performance/behavior

learning by means of good or bad experience,

- **Reinforcement Learning (RL)**
- **RL is learning through direct experimentation.**
- **It does not assume the existence of a teacher that provides 'training examples' upon which learning of a task takes place**
- **Instead, in RL experience is the only teacher**

Reinforcement learning



Agent receives no examples and starts with no model of the environment.

Agent gets feedback through rewards, or reinforcement.

Markov Decision Processes (MDP) describe an environment for reinforcement learning.

In MPDs the "present state & action" pair holds all the information required for determining future state

Markov Decision Process (MDP)

- At each discrete time point
  - **Agent** observes state s_t and chooses *action* a_t *(according to some probability distribution)*
  - Receives *reward* r_t from the environment and the *state changes* to s_{t+1}
- **Markov assumption:**

  $r_t = r(s_t, a_t) \quad s_{t+1} = \delta(s_t, a_t)$

  r_t and s_{t+1} depend only on the *current* state and action

A fundamental issue in RL algorithms is the balance/trade-off between Exploration of the environment and Exploitation of information already obtained by the agent.

Exploitation:

Use current knowledge to take the optimal action

Exploration:

Take possibly suboptimal action to gather more information about the environment

Exploitation vs Exploration

exploitation(greedy approach) : Taking the action which the agent estimates to be the best at the current moment.

the agent is exploiting its current knowledge about the reward structure of the environment to act.

The opposite approach to greedy selection is to simply always take a random action- exploration.

- Restaurant Selection
  - Exploitation: Go to favorite restaurant
  - Exploration: Try a new restaurant

- Online Banner Advertisements
  - Exploitation: Show the most successful advert
  - Exploration: Show a different advert

- Clinical Trial
  - Exploitation: Choose the best treatment so far
  - Exploration: Try a new treatment

- Game Playing
  - Exploitation: Play the move you believe is best
  - Exploration: Play an experimental move

Q-learning: learning the action-value function

Q-learning is about learning Q-values through observations.

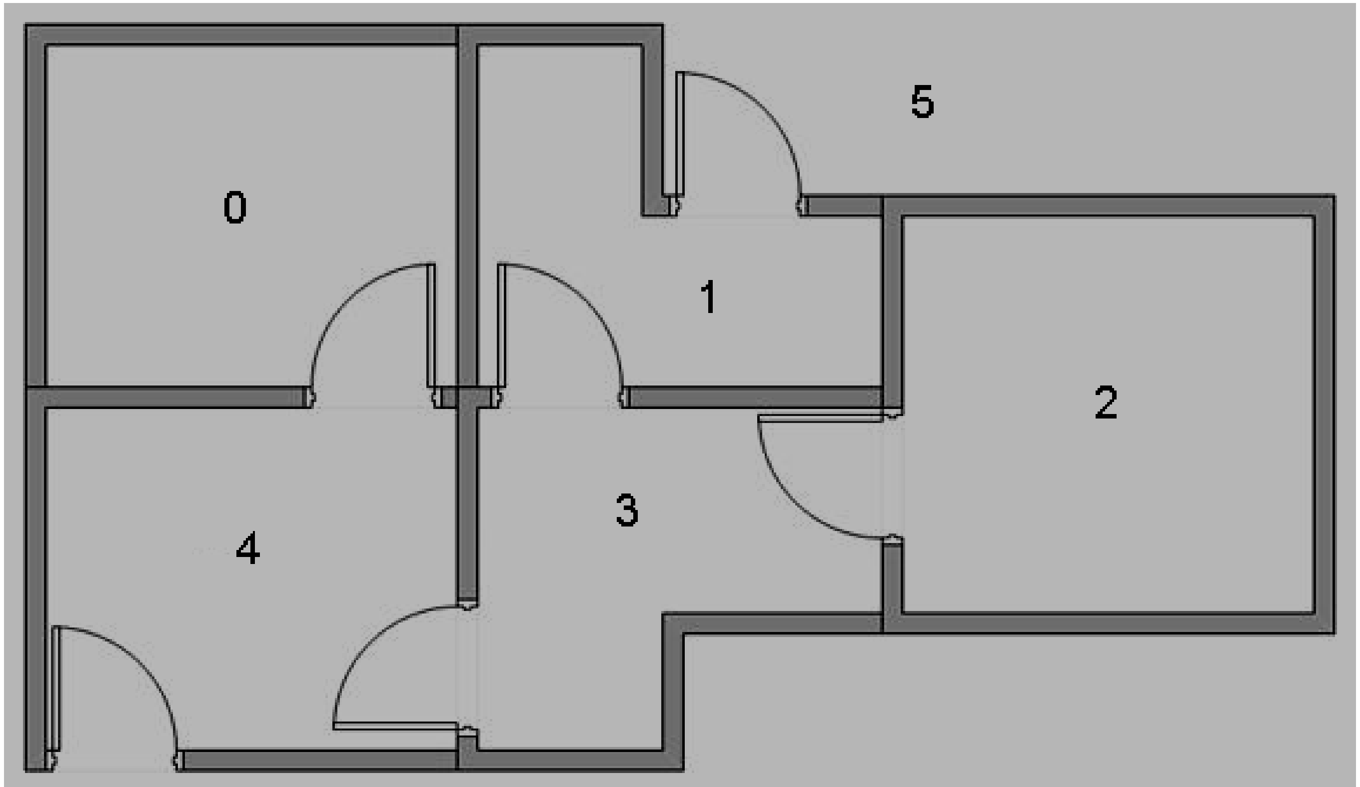
Q-learning is a model-free reinforcement learning technique.

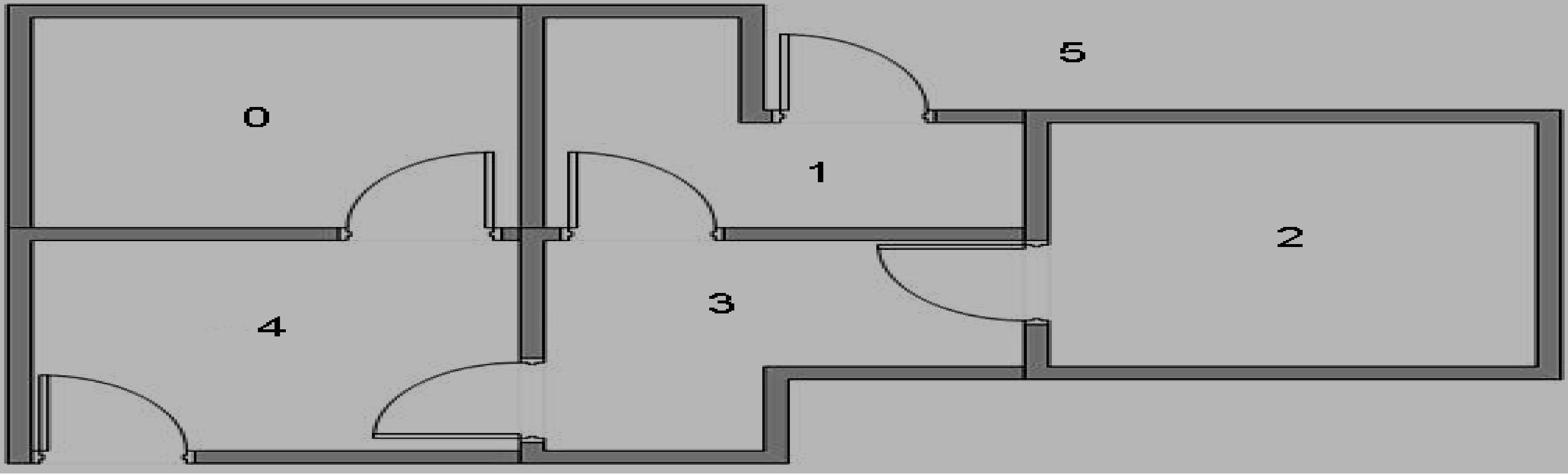
The ‘Q’ in Q-learning stands for quality.

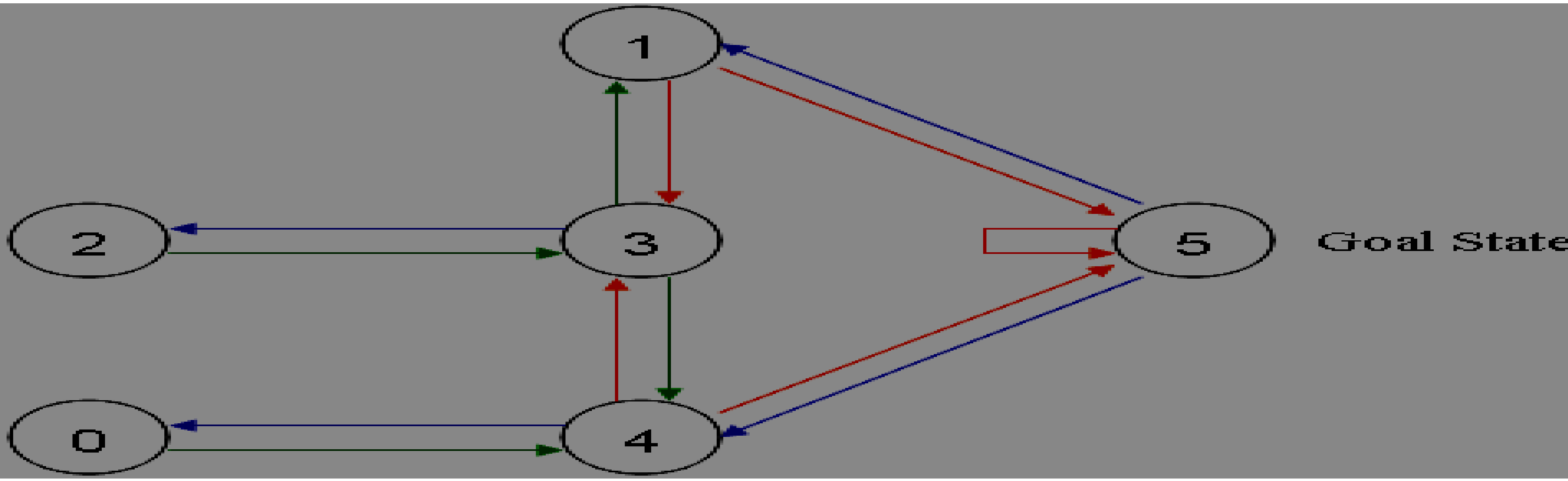
Quality in this case represents how useful a given action is in gaining some future reward.

Q Learning Example

- 5 rooms in a building connected by doors
- The outside of the building can be thought of as one big room (5).
- Doors 1 and 4 lead from building to room 5 (GOAL-outside).



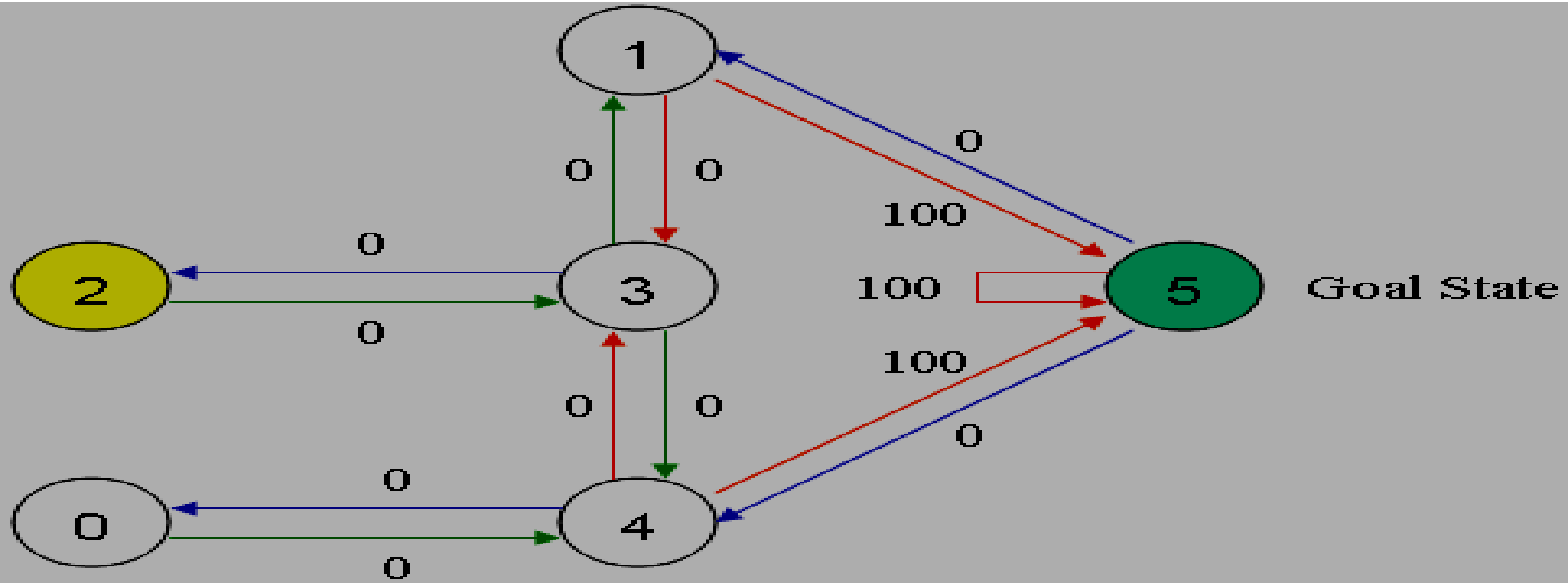




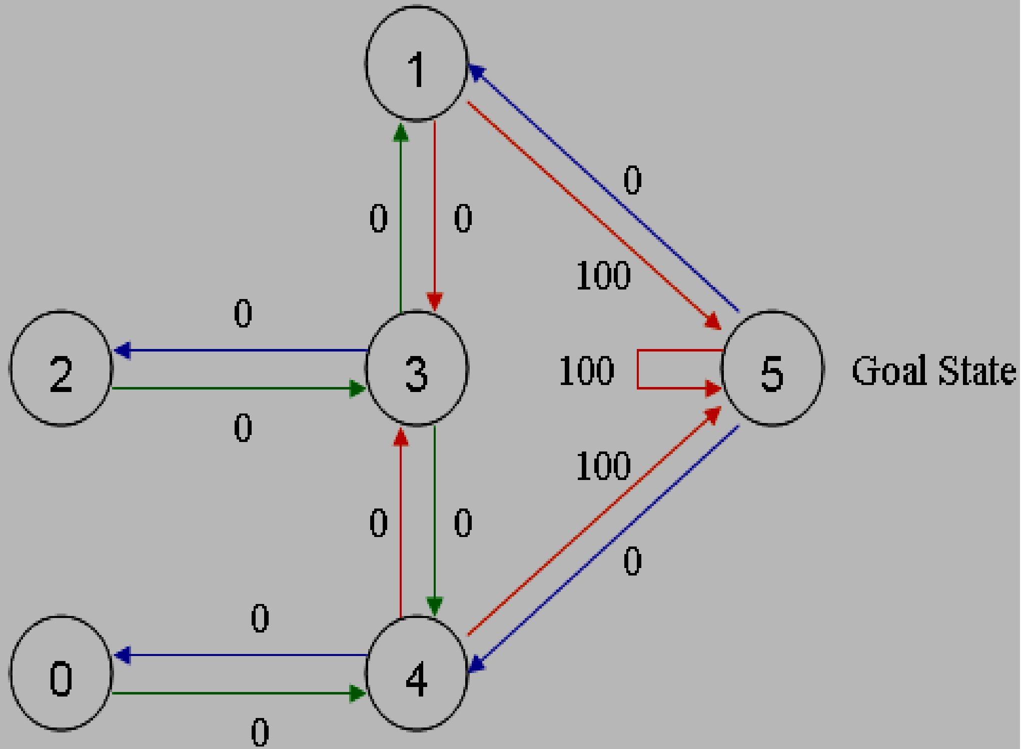
Each room is a node **Each door is a link.**

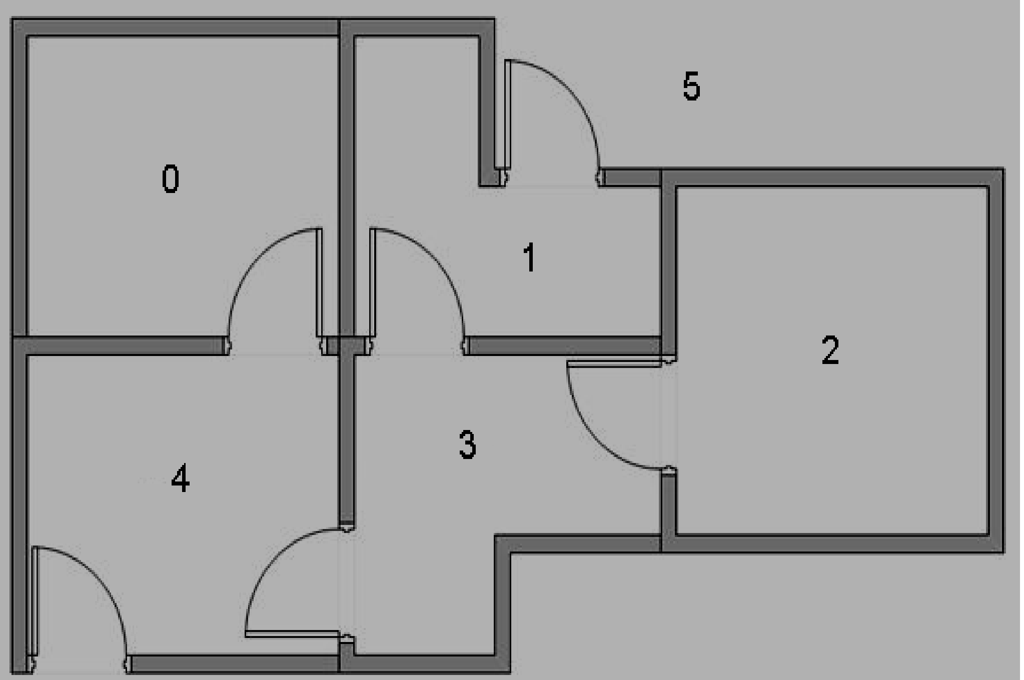
- **Goal room number =5**
- **Room 5 loops back to itself .**

- Each Room, including outside (room no 5), is called a **"state"**
- The agent's movement from one room to another is an **"action"**



- **In diagram, a "state" is depicted as a node, while "action" is represented by the arrows.**



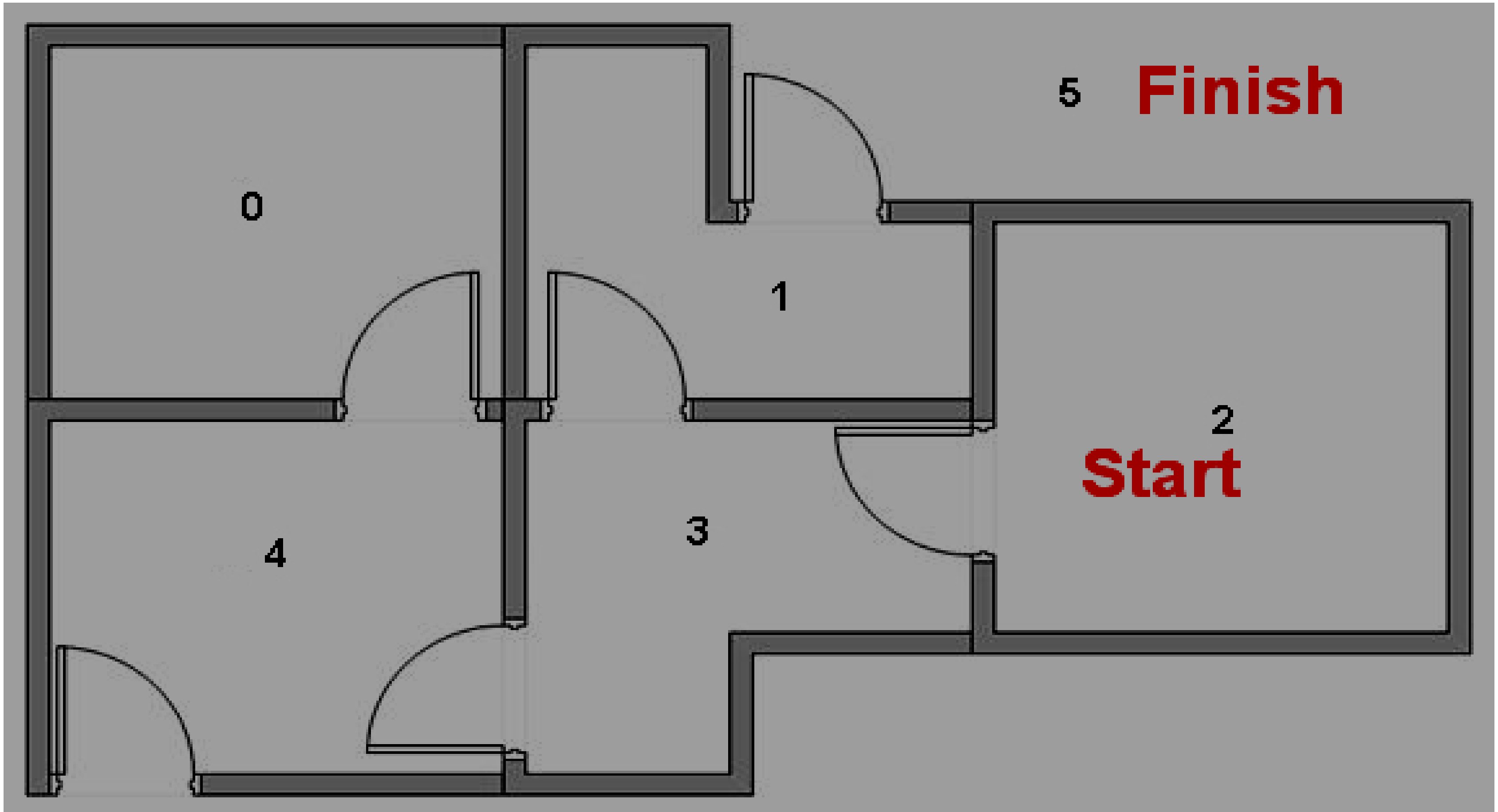


- **Each arrow contains an instant reward value,**
- **A reward value is associated to each door**
- **The doors that lead immediately to the goal have an instant reward of 100**
- **Otherwise reward = 0**

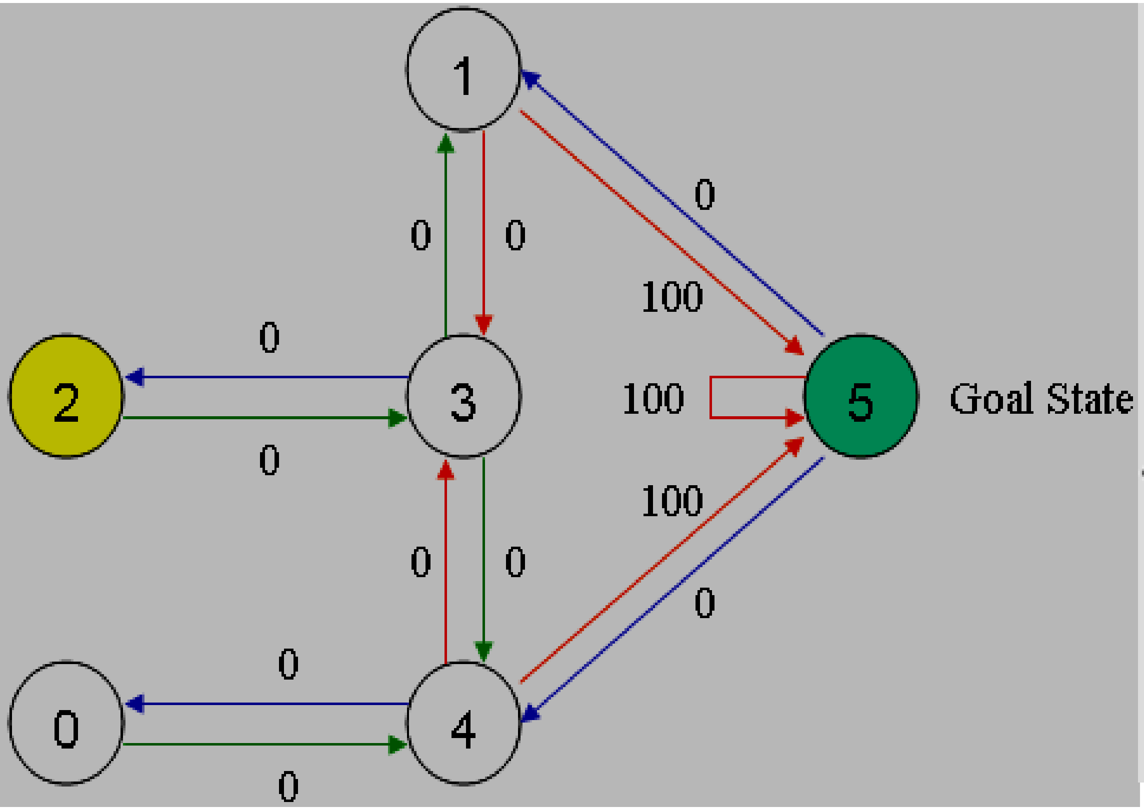
In Q-learning, goal is to reach the state with the highest reward, so that if the agent arrives at the goal, it will remain there forever, called an "absorbing goal".

•*Imagine our agent as a dumb virtual robot that can learn through experience.*

•**The agent can pass from one room to another but has no knowledge of the environment, and doesn't know which sequence of doors lead to the outside.**



Reward Table, "matrix R".



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

-1 : No link between nodes [to distinguish between zero reward and no link]

0 : reward is zero

100 : reward is 100 if reaching to goal state

MAKE Reward matrix

Matrix "Q"

- **Matrix, "Q"** : **representing the memory of agent what the agent has learnt through experience.**
- **Rows** : **current state of the agent**
- **Columns** : **possible actions leading to the next state (the links between the nodes).**
- **Matrix Q is initialized to zero as:**

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

Q(state, action) =

R(state, action) + γ * Max[Q(next state, all actions)]

According to this formula, a value assigned to a specific element of matrix Q, is equal to the sum of the corresponding value in matrix R and the learning parameter Gamma, multiplied by the maximum value of Q for all possible actions in the next state.

($0 \leq \gamma \leq 1$) closer to zero => the agent will tend to consider only immediate rewards

closer to one => the agent will consider future rewards with greater weight, willing to delay the reward

Algorithm

The Q-Learning algorithm goes as follows:

1. Set the gamma parameter, and environment rewards in matrix R.

2. Initialize matrix Q to zero

3. For each episode:

Select a random initial state.

Do While the goal state hasn't been reached.

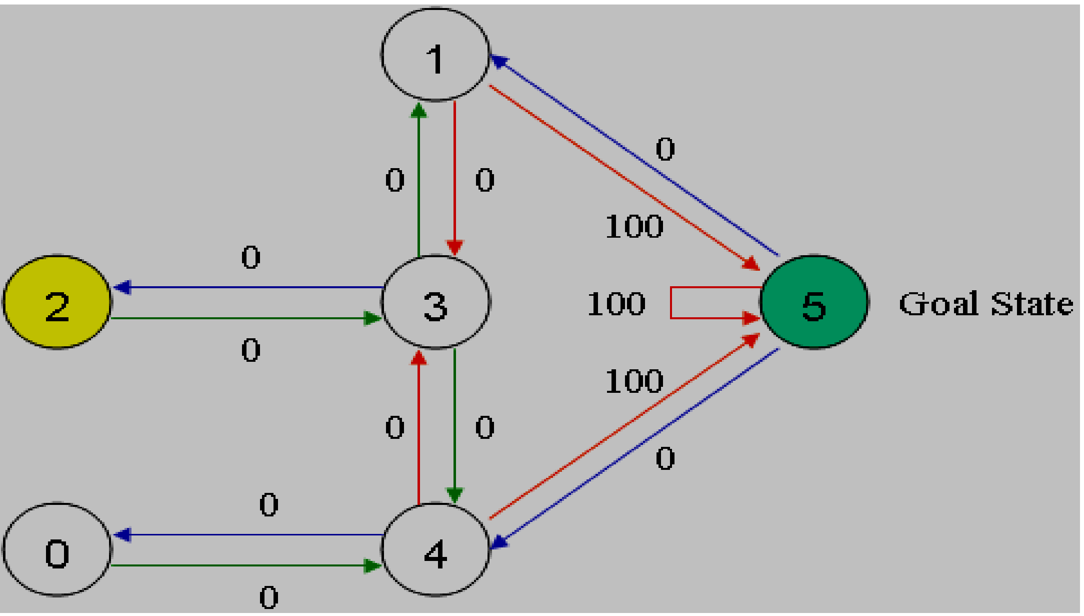
- **Select one among all possible actions for the current state.**
- **Using this possible action, consider going to the next state.**
- **Get maximum Q value for this next state based on all possible actions.**
- **Compute: Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]**
- **Set the next state as the current state.**

End Do

End For

- Set learning parameter Gamma = 0.8,
- **initial state as Room 1.**
- Initialize matrix Q as a zero matrix:

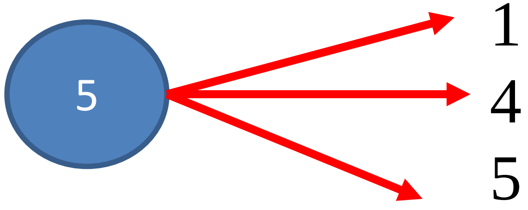
$$Q = \begin{array}{c|c} & \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 &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

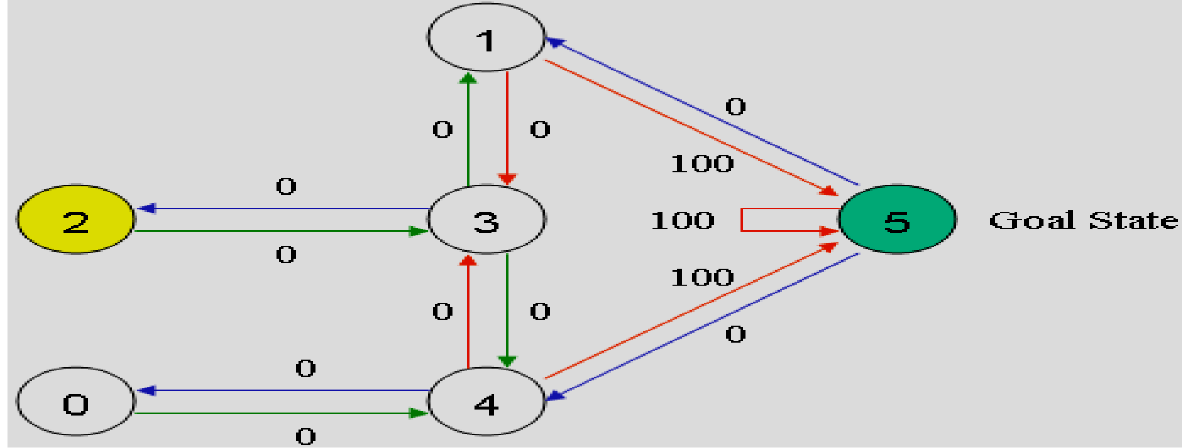


Action

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

By random selection, select to go to 5 as action





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

- **Since matrix Q is still initialized to zero, Q(5, 1), Q(5, 4), Q(5, 5), are all zero.**
- **The next state, 5, now becomes the current state.**
- **Because 5 is the goal state, finished one episode.**
- **agent's brain now contains an updated matrix Q as:**

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

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

- **For next episode, randomly chose initial state.**
- ***choose state 3 as initial state*.**
- **By random selection, *select state 1 as action*.**

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]

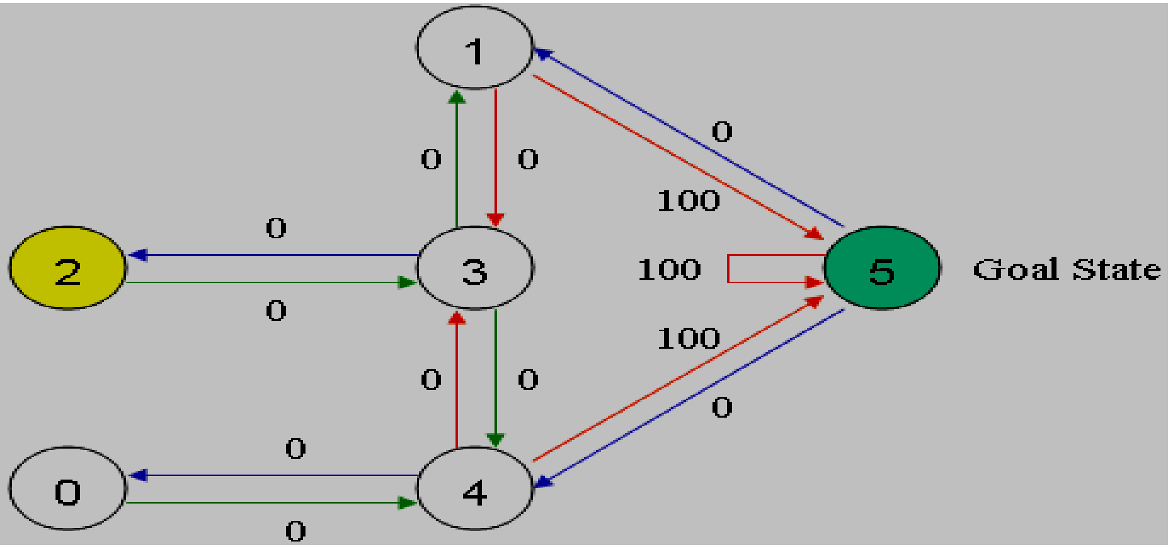
Q(3, 1) = R(3, 1) + 0.8 * Max[Q(1, 3), Q(1, 5)]

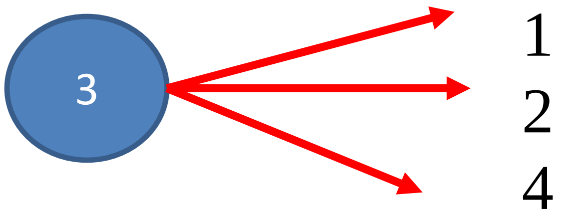
= 0 + 0.8 * Max(0, 100) = 80

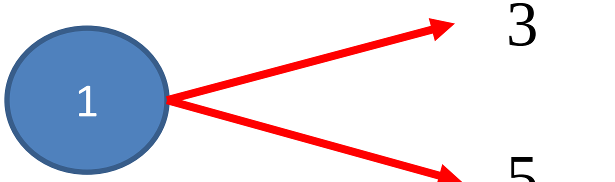
update matrix Q

.

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





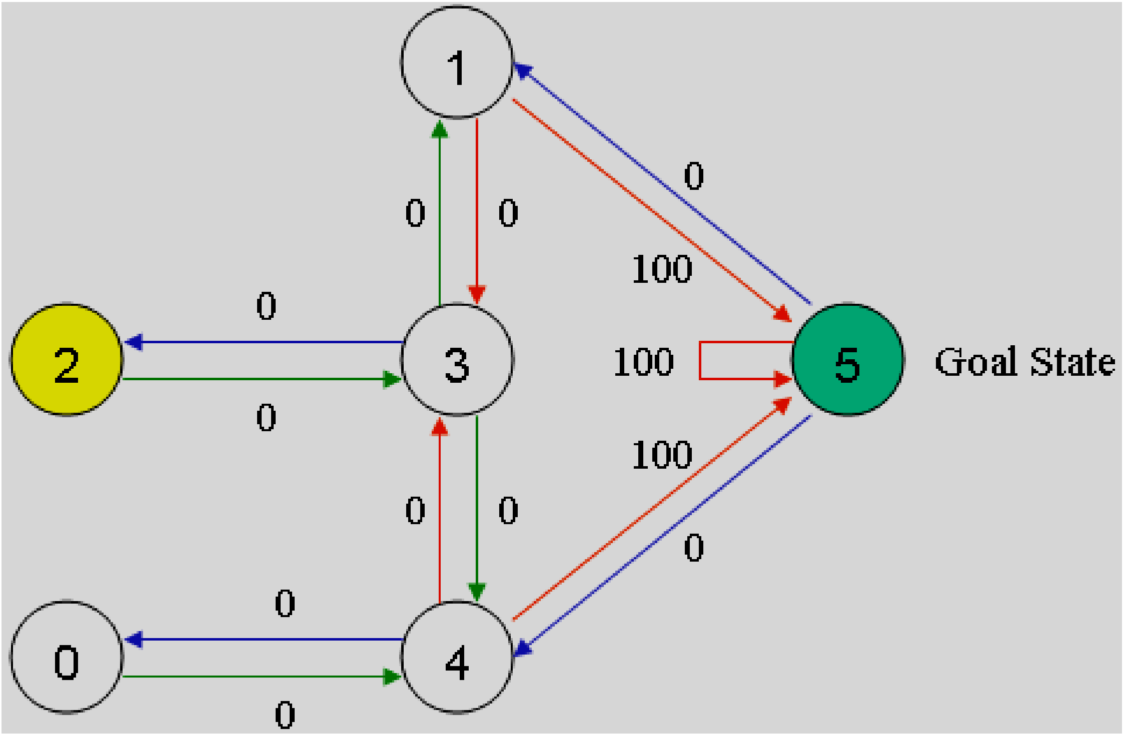


The matrix Q becomes

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

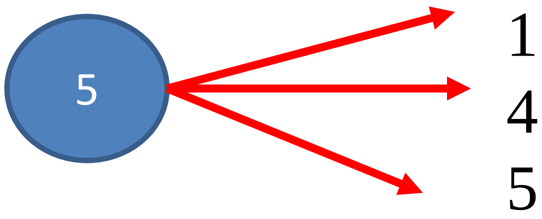
- **The next state, 1, now becomes the current state.**
- **Repeat the inner loop of the Q learning algorithm because state 1 is not the goal state[episode not over].**

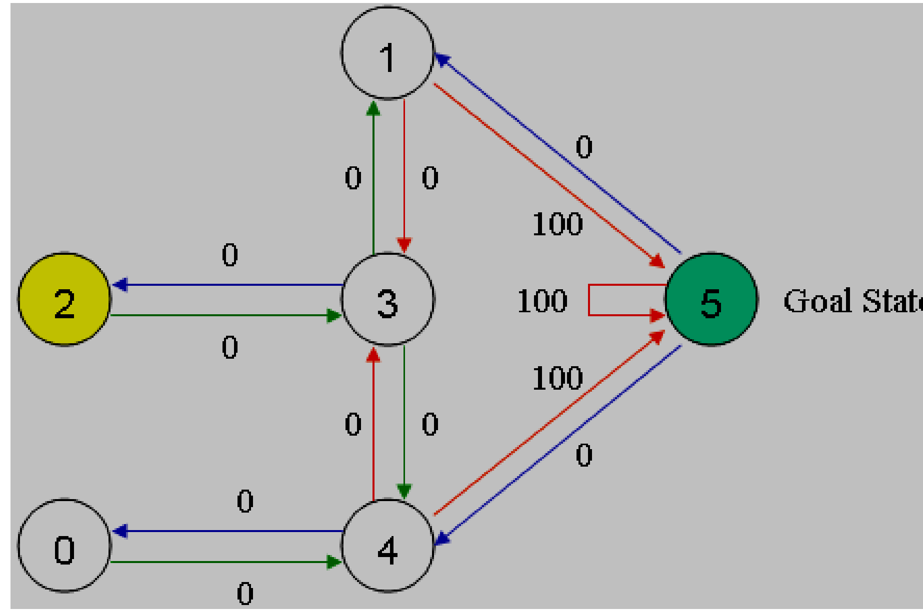
- **Set current state as 1**
- **two possible actions: go to state 3, or go to state 5.**
- **Randomly choose action state 5.**



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





- **Reached to goal state finish this episode.**
- **agent's brain now contain updated matrix Q as:**

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

- **Agent learns more through further episodes**
- **finally reach convergence values in matrix Q like:**

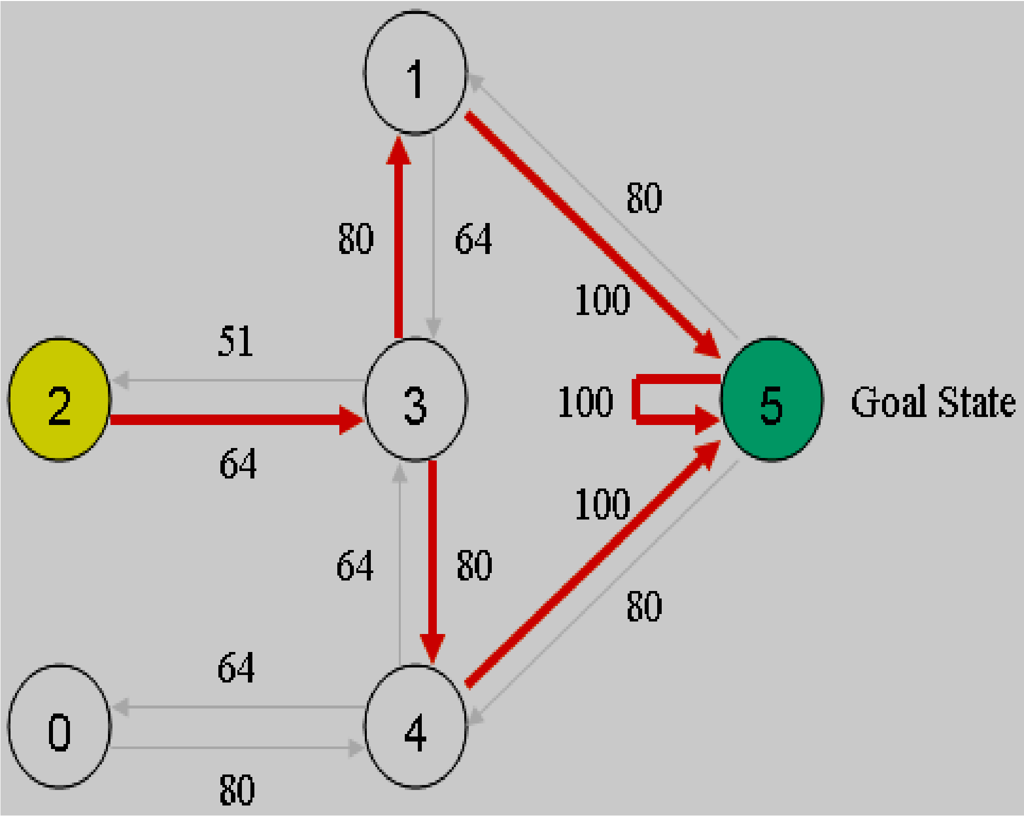
$$
Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 400 & 0 \\
0 & 0 & 0 & 320 & 0 & 500 \\
0 & 0 & 0 & 320 & 0 & 0 \\
0 & 400 & 256 & 0 & 400 & 0 \\
320 & 0 & 0 & 320 & 0 & 500 \\
0 & 400 & 0 & 0 & 400 & 500
\end{bmatrix}
\end{array}
$$

This matrix Q, can then be normalized (i.e.; converted to percentage) by dividing all non-zero entries by 5 :

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

- **Agent has learned the most optimal paths to the goal state through experience.**
- **Best sequences : the links with the highest values at each state.**

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



If we start at 2,
the best sequence is 2 - 3 - 1 – 5 or 2-3-4-5

DEEP Q LEARNING