Reinforcement Learning

MUKESH KUMAR

AGENDA

- What is Reinforcement Learning
- Core Concepts of RL
- Types of RL
- Q-Learning Algorithm
- Mouse Maze Problem
- Q-Learning in OpenAI Gym – Smart Taxi
- SARSA
- Policy Gradient Theory

Introduction to Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with an environment to maximize a reward.

Introduction to Reinforcement Learning

Definition: Learning by interacting with an environment to maximize cumulative reward.

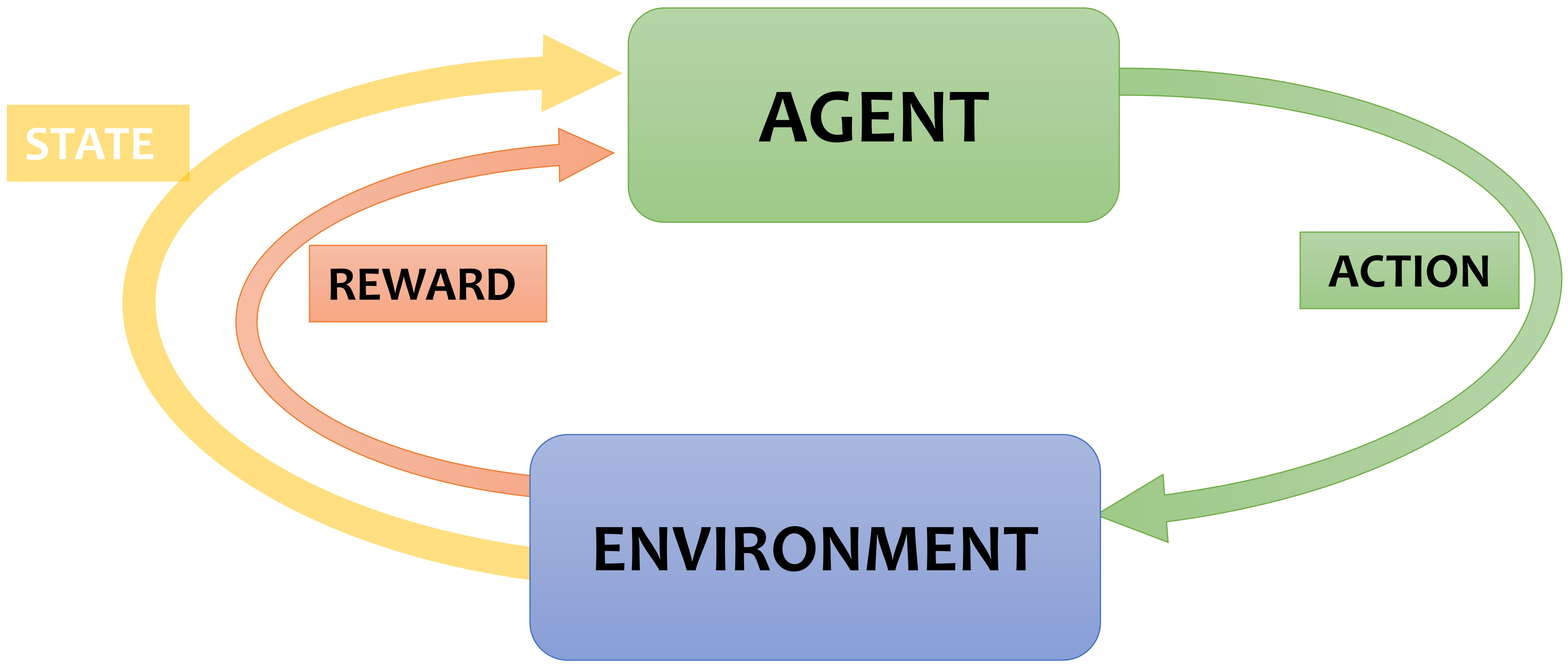
Key Concepts:

- **Agent:** The learner or decision-maker.
- **Environment:** The world the agent interacts with.
- **Action:** A move the agent makes.
- **State:** The current situation of the agent.
- **Reward:** Feedback signal from the environment.

Goal:

Learn a **policy** (strategy) that maximizes the **cumulative reward** over time.

Architecture of Reinforcement Learning



Where is Reinforcement Learning Used?

Games & Simulation:

- AlphaGo (beating world champions at Go)
- OpenAI Five (Dota 2)
- Chess and Atari games

Robotics:

- Robotic arms for picking/sorting
- Industrial automation
- Bipedal/humanoid robot walking

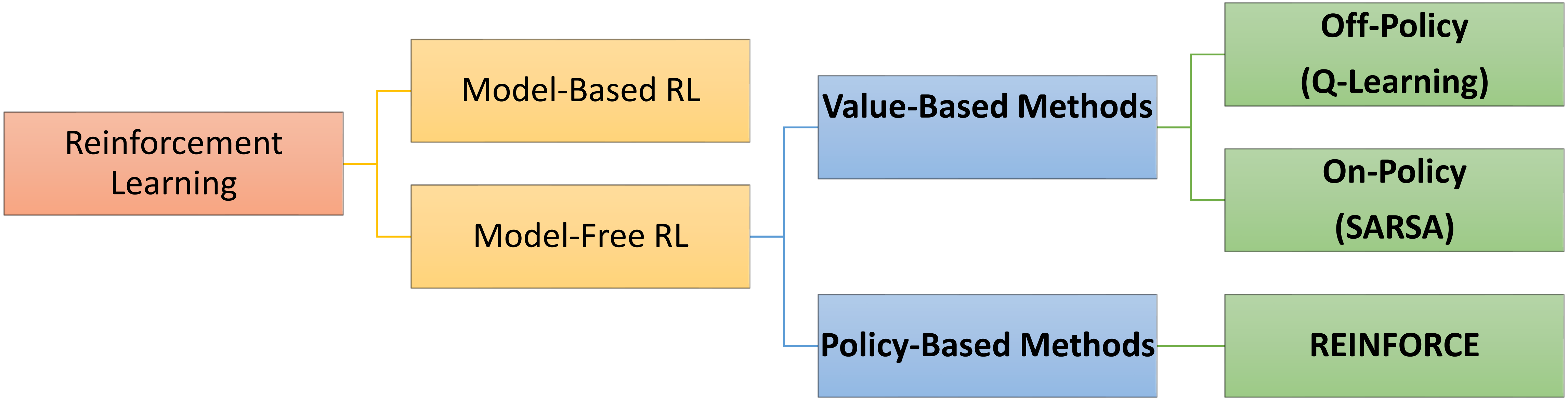
Where is Reinforcement Learning Used?

Autonomous Driving:

- Lane following
- Obstacle avoidance
- Decision-making in traffic

Operations & Control:

- Elevator scheduling
- HVAC systems
- Power grid optimization


Reinforcement Learning
Model-Based RL
Model-Free RL
Value-Based Methods
Policy-Based Methods
Off-Policy
(Q-Learning)
On-Policy
(SARSA)
REINFORCE

Key Components of Reinforcement Learning

Agent

- The decision maker
- Learns from experience and chooses actions

Environment

- Everything the agent interacts with
- Provides feedback via rewards

Key Components of Reinforcement Learning

State (s)

- A snapshot of the environment at a given time
- Example: Position of a car in a driving simulation

Action (a)

- A decision taken by the agent
- Example: Move left, accelerate, pick object

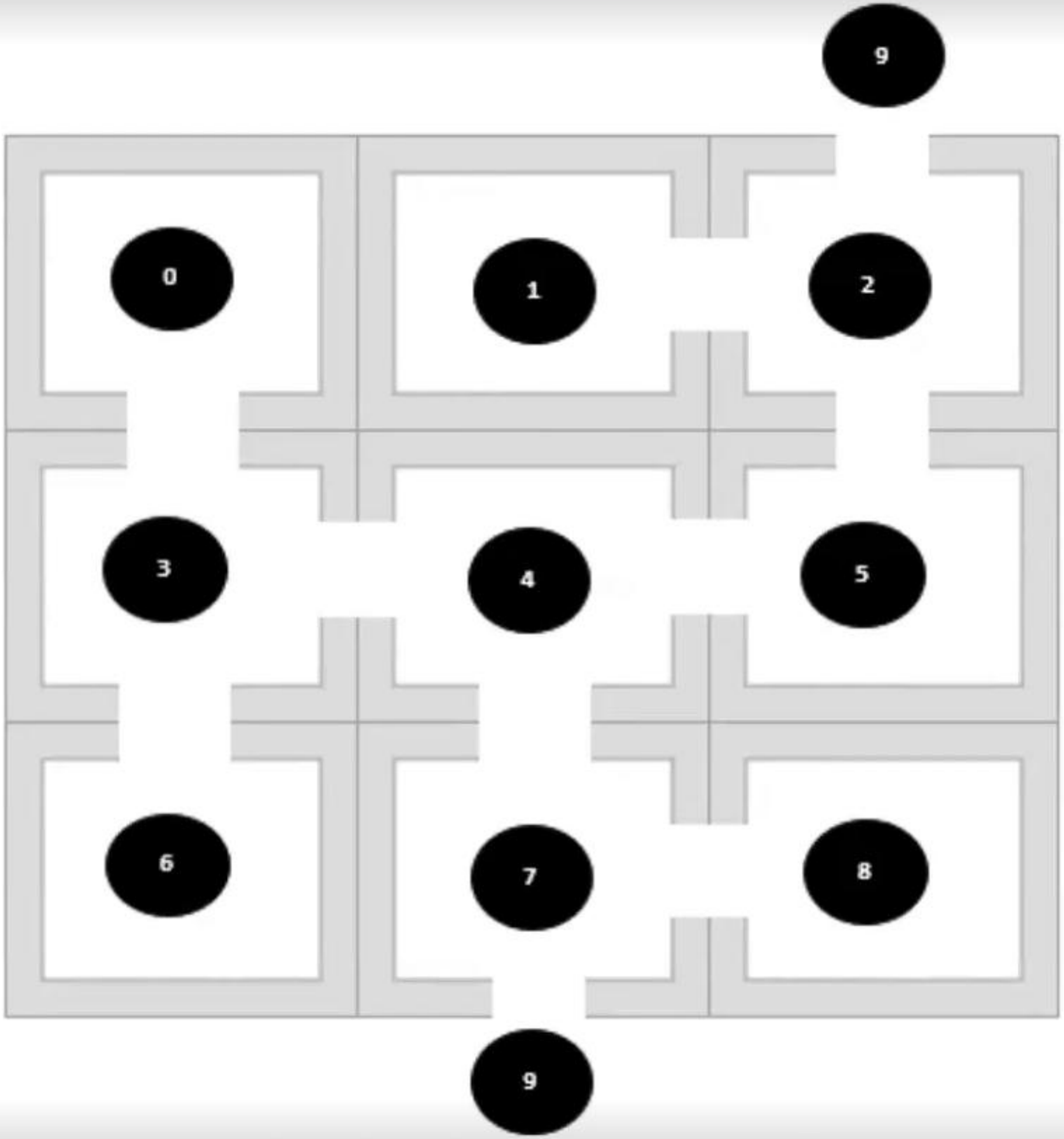
Key Components of Reinforcement Learning

Reward (r)

- A numerical signal from the environment
- Positive for good actions, negative for bad ones
- Drives learning and behavior

Mouse Maze

Problem Introduction


9
0
1
2
3
4
5
6
7
8
9

- There are totally 10 States = 0-9

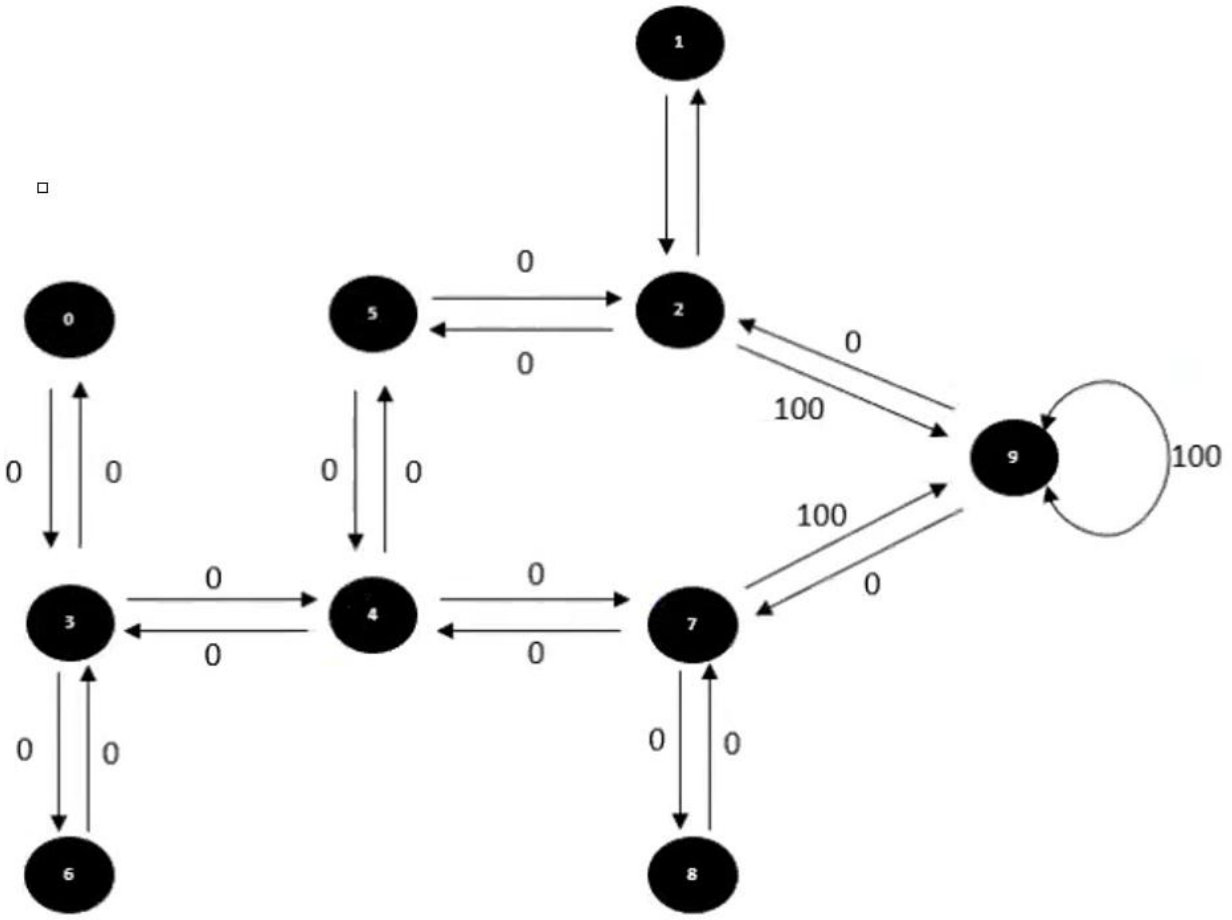
- For each state what are the possible actions and reward for each action(reward can be positive or negative)

- Goal: Mouse has to exit the maze

Setting up the Rewards

- Reward is a scalar quantity
  - Positive
  - Negative
  - Neutral

For mouse maze , we only have positive rewards, there is not Negative rewards



Initializing Matrices

- Q-Matrix
- R-Matrix
- State-Action Reward Matrix

- What are the different starts in mouse maze problem?

- What are the actions possible in mouse maze problems??

Reward Matrix

- 10 States possible = 0-9
- 4 actions possible = up, down, right, left

	up	down	left	right
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				

For state 0

- From 0 it can only go to 3 , only down action is possible

- From State 1, only right is possible

	up	down	left	right
0		0		
1				0
2				
3				
4				
5				
6				
7				
8				
9				

Another approach of creating Reward Matrix

- State vs State

We are going to use this one for

Mouse maze problem

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

Sate Vs State Reward Matrix

- From 0 state it can move to only 3
- From 1 state it can move only to 2
- From 2 state it can go to 1,5 or 9

- This is what we call reward matrix, what are the possible actions and corresponding rewards

	0	1	2	3	4	5	6	7	8	9
0				0						
1			0							
2		0				0				100
3										
4										
5										
6										
7										
8										
9										

Reward Matrix

- This is the final Reward matrix , -1 here is not reward it's an invalid option for python program

-1	-1	-1	0	-1	-1	-1	-1	-1	-1
-1	-1	0	-1	-1	-1	-1	-1	-1	-1
-1	0	-1	-1	-1	0	-1	-1	-1	100
0	-1	-1	-1	0	-1	0	-1	-1	-1
-1	-1	-1	0	-1	0	-1	0	-1	-1
-1	-1	0	-1	0	-1	-1	-1	-1	-1
-1	-1	-1	0	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	0	-1	-1	-1	0	100
-1	-1	-1	-1	-1	-1	-1	0	-1	-1
-1	-1	0	-1	-1	-1	-1	0	-1	100

Q-Matrix (Quality Matrix)

- It's replica of reward matrix except it has all values initialized to zeros.

- Values will be updated based on actions agents take

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Step1: Calculation

- Lets say we dorp the mouse at 2
- From 2 : 1, 5, 9
- Say it goes to 9
- $Q(2,9) = R(2,9) + 0.7 (Q(9,2), Q(9,9), Q(9,7)) – Q(2,9)$
- $Q(2,9) = 100$

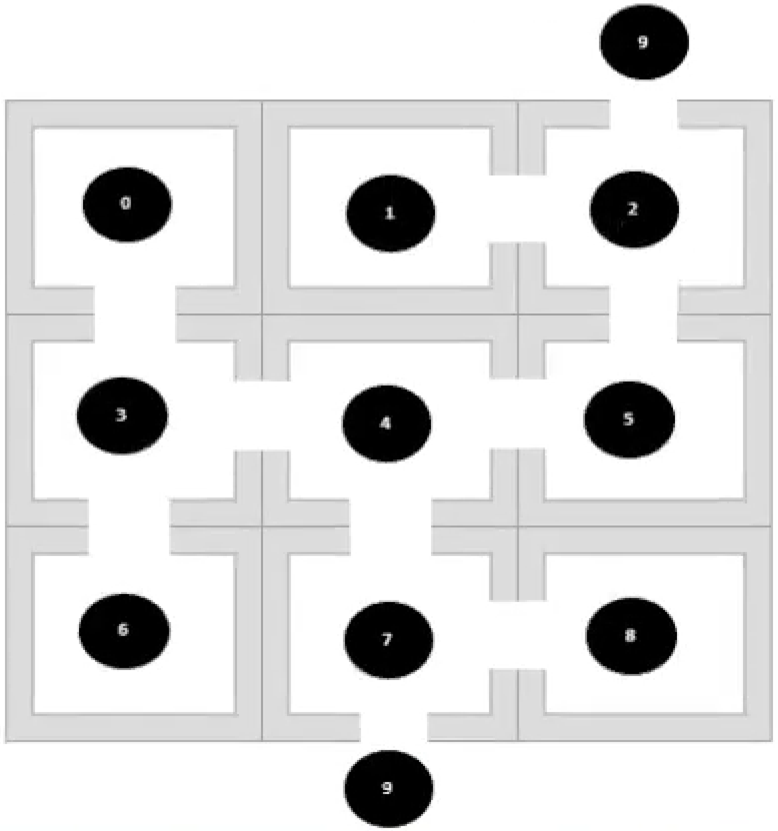
Updated Q-Matrix after step 1

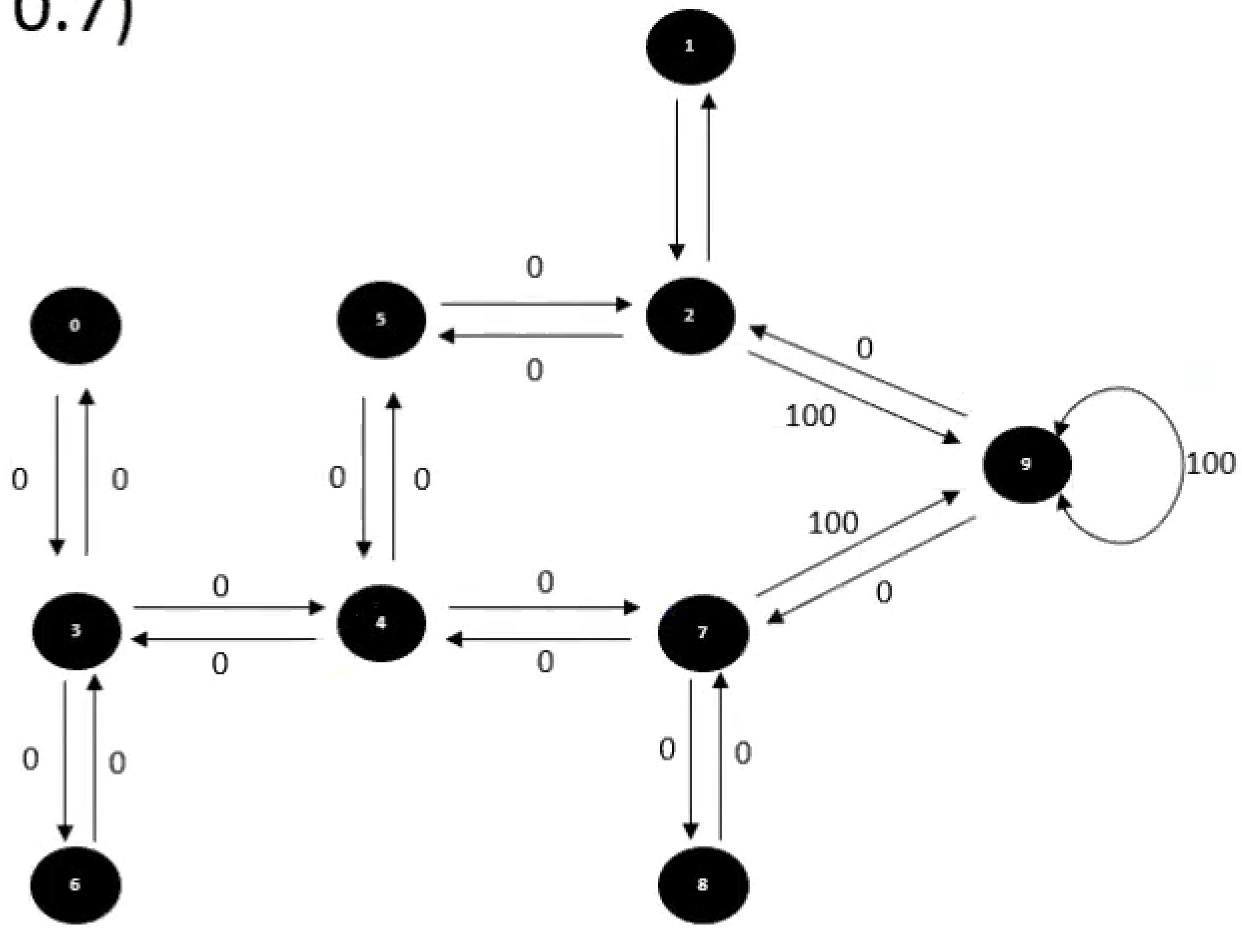
- When we repeate this process 1000 of times we will have a Q-matrix which will have optimum values in each cell
- We can then use this Q-Matrix to decide the next best state mouse should move in order to exit the maze in shortest possible path
- 0>3>4>7>9

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	100
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

- In Class Problem

Calculate the Q value, when the initial state = 3 and mouse moves to state 4 at Random. (Assume Gamma = 0.7)





Q-Learning Algorithm (Theory)

- **Q-Learning** is a model-free, value-based **Reinforcement Learning algorithm**
- that learns the **optimal action-selection policy** using **Q-values**.

- Learns how good an action is at a given state, without knowing the environment's
- dynamics.

Q-Learning Key Concepts

Q-Value (Q(s, a)):

• Estimates the **expected total reward** of taking action **a** in state **s** and following the optimal policy afterward.

Goal:

• Learn the optimal Q-values, denoted **Q***, so the agent can act greedily:

$\boldsymbol{\pi(s) = \mathrm{argmax}_a\, Q^*(s, a)}$

Q-Learning Update Rule

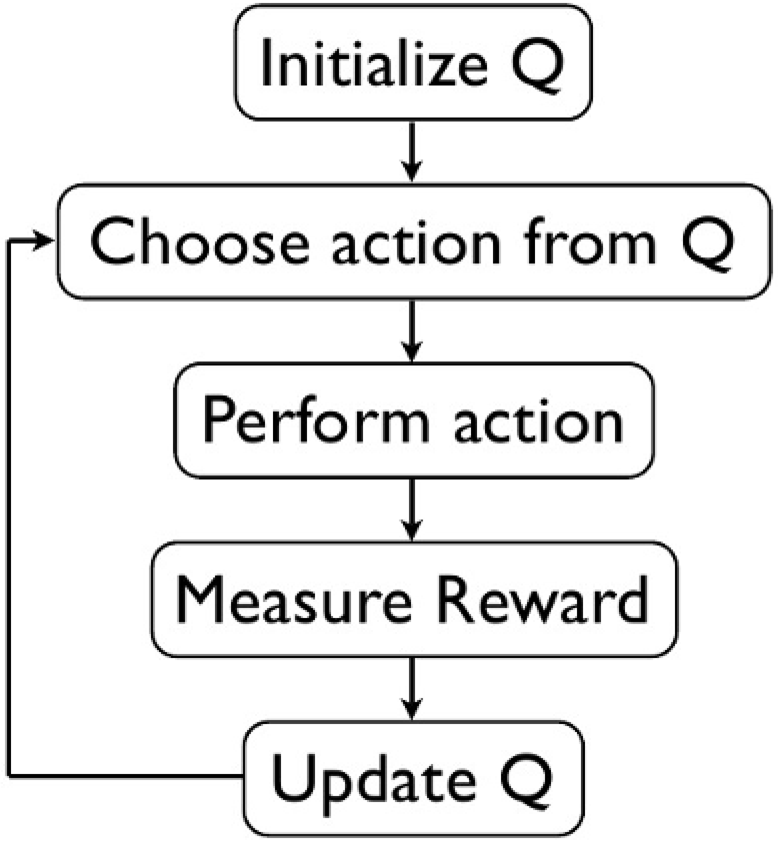
$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

- Where:
- **s** = current state
- **a** = action taken
- **r** = reward received
- **s'** = next state
- **a'** = possible next actions
- **α** = learning rate ($0 < \alpha \leq 1$)
- **γ** = discount factor ($0 \leq \gamma < 1$)

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \underbrace{\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]}_{\text{TD Error}=\delta}$$

Temporal Difference (TD)

Q-learning: Algorithm



Characteristics of Q-Learning

Model-Free

- Doesn't require knowledge of transition probabilities

Off-Policy

- Learns optimal policy independently of the actions taken

Convergence

- Converges to optimal Q-values under appropriate exploration and learning rate conditions