Frozen Lake: Reinforcement Learning Model based on stochastic Q-learning

Table of Contents

Abstract

The goal of this assignment is to implement Q-learning using a Q-table. Q-learning is an off policy learning method, which learns from the environment and comes up with a policy, that helps the agent to reach the goal. The first phase of the assignment gives a baseline implementation of reinforcement learning model , with a clear and step by step implementation of Q learning. The final phase of the report involves an implementation of Q learning using both methods. The environments namely FrozenLake-v0 is generated with the Openaigym, which is a toolkit for developing and comparing reinforcement learning algorithms.

Problem Statement

Developing a reinforcement learning agent model that automatically learns to play Frozen Lake game and acheive the best possible score over time.

Game Introduction

The agent should start from 'S' and move across to reach the frisbee(goal 'G'). While taking steps, there are holes 'H' which lead to end of the game or Frozen ground 'F' which are safe to pass. The agent should choose path intelligently from start 'S' to goal 'G' to gain a reward.

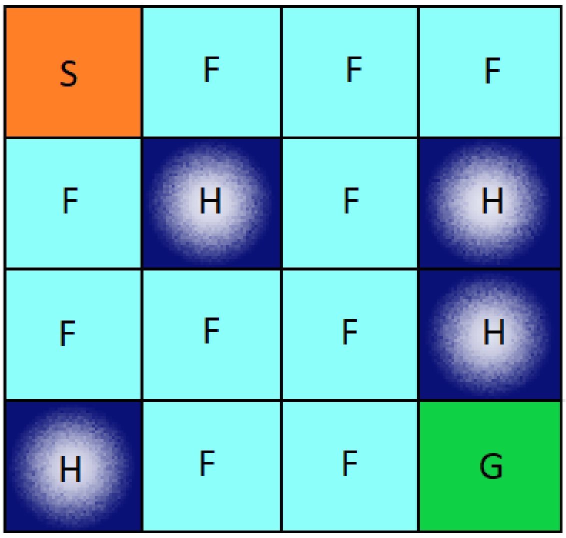

Fig. Frozen Lake game view

Actions and states in the game

There are 4 actions and 16 states in the game.

States (4 in each stage):

- SFFF
- FHFH
- FFFH
- HFFG

S - start state
F - frozen
H - Hole

G - Goal, Frisbee

Actions: Left, Right, Up and Down

Rewards

1. S - start state (safe, reward = 0)
2. F - frozen (safe, reward = 0)
3. H - Hole (reward = -1, terminate the episode)
4. G - Goal (reward = 1, terminate the episode)

Solution approach

First the agent is trained over 10,000 episodes in which we update the Q-table (containing the action values of the steps taken) using the Bellman's equation and use it as a reference to play the game. The maximum number of steps taken during an episode will remain fixed.

The terms in the Bellman's equation are pre-defined which are discounting rate, learning rate and the reward, state, action values are obtained from the environment during the episode based on the action chosen. The Q-table is updated with the result of Bellman's equation.

An exploration variable 'epsilon' is also chosen to define the exploration ratio and how it should reduce over time is defined using the exploration equation.

Finally the results are evaluated with respect to average steps taken in all the episodes(when the agent plays the game) and the score over time(when agent was trained to play the game).

Building Reinforcement Learning Model

Importing Libraries

In [1]:

```
import gym
import numpy as np
import random
```

Obtaining Frozen Lake environment from OpenGym AI

In [2]:

```
env = gym.make("FrozenLake-v0")
env.render()
```

```
SFFF
FHFH
FFFH
HFFG
```

Defining action and state sizes as per the Frozen Lake environment

In [3]:

```
action_size = env.action_space.n
state_size = env.observation_space.n
print ("Environemnt properties","\n")
print("Action size: ", action_size, "\n", "State size: ", state_size)
```

```
Environemnt properties

Action size:  4
 State size:  16
```

Defining a baseline performance

The parameters for the Bellman's equation for setting the baseline performance are listed below.

Step 1: Initializing the parameters

Initially the Q-table is started with zero matrix of size (4,16)

In [88]:

```
qtable = np.zeros((state_size, action_size))
print("Q-table: ","\n")
print(qtable)
```

```
Q-table:

[[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.]
 [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.]]
```

later the values in the Q-table are updated using the Bellman's equation which is

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{r_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

In [4]:

```
total_episodes = 5000        # Total episodes
alpha = 0.7              # Learning rate
max_steps = 99               # Max steps per episode
gamma = 0.8                  # Discounting rate

# Exploration parameters
epsilon = 1.0                # Exploration rate
max_epsilon = 1.0            # Exploration probability at start
min_epsilon = 0.01           # Minimum exploration probability
decay_rate = 0.01
```

Step 2: Training the agent

The agent is trained over 5000 episodes initially to benchmark as a baseline performance. Below are the steps run for training the agent.

1. For each episode, and in each step of the episode, generate a random number from a uniform distribution and if the number is greater than epsilon(exploration rate) then the agent choses the action that has maximum action value (future rewards).
2. Based on the action chosen, the agent receives a rewards as a value and the value is fed into the Bellman's equation and the Q-table is updated repeatedly based on the reward.
3. After each episode, the epsilon is reduced at an exponential rate as per the equation with the decay-rate defined.
4. Finally, at the end of all episodes the Q-table will have all the action values required to play the game.

In [5]:

```
# List of rewards
rewards = []

# 2 For life or until learning is stopped
for episode in range(total_episodes):
    # Reset the environment
    state = env.reset()
    step = 0
    done = False
    total_rewards = 0

    for step in range(max_steps):
        # 3. Choose an action a in the current world state (s)
        ## First we randomize a number
        exp_exp_tradeoff = random.uniform(0, 1)

        ## If this number > greater than epsilon --> exploitation (taking the bi
ggest Q value for this state)
        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state,:])

        # Else doing a random choice --> exploration
        else:
            action = env.action_space.sample()

        # Take the action (a) and observe the outcome state(s') and reward (r)
        new_state, reward, done, info = env.step(action)

        # Update Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') - Q(s,a)]
        # qtable[new_state,:] : all the actions we can take from new state
        qtable[state, action] = qtable[state, action] + alpha * (reward + gamma
* np.max(qtable[new_state, :]) - qtable[state, action])

        total_rewards += reward

        # Our new state is state
        state = new_state

        # If done (if we're dead) : finish episode
        if done == True:
            break

    # Reduce epsilon (because we need less and less exploration)
    epsilon = min_epsilon + (max_epsilon - min_epsilon)*np.exp(-decay_rate*episo
de)
    rewards.append(total_rewards)

print("****** Baseline performance of Frozen Lake agent ******\n")
print("Total episodes: ", total_episodes )
print("Learning rate: ", alpha)
print("Discounting rate: ", gamma,"\n")
print ("Score over time: " +  str(sum(rewards)/total_episodes)+"\n")
print(qtable,"\n")
print("Epsilon at maximum steps: ", epsilon)
```

```
****** Baseline performance of Frozen Lake agent ******

Total episodes:  5000
Learning rate:  0.7
Discounting rate:  0.8

Score over time: 0.321

[[2.01868798e-03 1.83881566e-03 2.77875168e-03 1.95429521e-03]
 [3.58470115e-04 8.88737740e-05 8.40307187e-04 1.94744987e-03]
 [1.37296322e-04 7.19128296e-04 8.42765935e-04 1.52908203e-04]
 [3.26880176e-05 2.01969030e-04 6.66864346e-05 5.54878723e-04]
 [1.08662775e-02 1.64942211e-03 1.37605798e-03 8.26887789e-04]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [8.95707868e-05 4.53410148e-04 1.98593111e-03 2.24638971e-06]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [2.75003151e-03 3.19647729e-03 2.30344232e-03 6.43107599e-02]
 [2.29726973e-02 2.09576238e-02 1.57425451e-02 6.44327905e-03]
 [3.78144999e-02 3.63525260e-02 9.87841987e-03 8.16926083e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [2.07557635e-02 9.82723363e-02 5.90263506e-01 9.72860703e-03]
 [5.06205387e-02 5.63086410e-02 9.13673690e-01 5.54839838e-02]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

Epsilon at maximum steps:  0.01
```

The score over time of 0.3062 is achieved using Q-learning. This is set as "Baseline performance" of the model.

Step 3: Making the agent play the game

Now, the agent is ready for playing the game. Steps executed by the below code for playing the game:

1. For 50 episodes or chances, the agents plays the Frozen Lake game.
2. In each episode, the specific action in the particular state is chosen in the Q-table that has the highest expected future value.
3. In each episode, the last action chosen is printed and the last state of the rendered is printed to understand whether it is 'Hole' or 'Goal'.

In [6]:

```
env.reset()
episode_steps = []
for episode in range(50):
    state = env.reset()
    step = 0
    done = False
    print("****************************************************")
    print("EPISODE ", episode)

    for step in range(max_steps):

        # Take the action (index) that have the maximum expected future reward g
iven that state
        action = np.argmax(qtable[state,:])

        new_state, reward, done, info = env.step(action)

        if done:
            # Here, we decide to only print the last state (to see if our agent
 is on the goal or fall into an hole)
            env.render()

            # We print the number of step it took.
            print("Number of steps", step)
            break
        state = new_state
    episode_steps.append(step)
env.close()
```

```
**************************************************
EPISODE  0
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
**************************************************
EPISODE  1
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
**************************************************
EPISODE  2
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 46
**************************************************
EPISODE  3
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 3
**************************************************
EPISODE  4
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
**************************************************
EPISODE  5
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
**************************************************
EPISODE  6
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 19
**************************************************
EPISODE  7
  (Right)
SFFF
FHFH
```

```
FFFH
HFFG
Number of steps 9
***************************************************
EPISODE  8
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
***************************************************
EPISODE  9
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 12
***************************************************
EPISODE  10
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 40
***************************************************
EPISODE  11
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
***************************************************
EPISODE  12
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
***************************************************
EPISODE  13
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 21
***************************************************
EPISODE  14
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 27
***************************************************
EPISODE  15
```

```
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
****************************************************
EPISODE  16
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 15
****************************************************
EPISODE  17
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 21
****************************************************
EPISODE  18
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 59
****************************************************
EPISODE  19
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 12
****************************************************
EPISODE  20
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 69
****************************************************
EPISODE  21
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 15
****************************************************
EPISODE  22
  (Right)
SFFF
FHFH
FFFH
HFFG
```

```
Number of steps 22
****************************************************
EPISODE  23
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 40
****************************************************
EPISODE  24
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 24
****************************************************
EPISODE  25
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
****************************************************
EPISODE  26
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
****************************************************
EPISODE  27
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
****************************************************
EPISODE  28
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 42
****************************************************
EPISODE  29
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
****************************************************
EPISODE  30
  (Right)
SFFF
```

```
FHFH
FFFH
HFFG
Number of steps 27
****************************************************
EPISODE  31
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
****************************************************
EPISODE  32
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
****************************************************
EPISODE  33
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 28
****************************************************
EPISODE  34
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
****************************************************
EPISODE  35
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 38
****************************************************
EPISODE  36
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 15
****************************************************
EPISODE  37
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 25
****************************************************
```

```
EPISODE  38
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 3
***************************************************
EPISODE  39
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 12
***************************************************
EPISODE  40
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
***************************************************
EPISODE  41
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 39
***************************************************
EPISODE  42
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 27
***************************************************
EPISODE  43
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 23
***************************************************
EPISODE  44
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 21
***************************************************
EPISODE  45
  (Right)
SFFF
FHFH
FFFH
```

```
HFFG
Number of steps 36
****************************************************
EPISODE  46
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 12
****************************************************
EPISODE  47
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 45
****************************************************
EPISODE  48
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 20
****************************************************
EPISODE  49
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
```

In [7]:

```
print(np.mean(episode_steps))
```

```
20.78
```

Tuning with hyperparameters

Testing with different hyperparameters (set 1)

The hyperparameters, alpha and gamma are changed to evaluate the change in baseline performance

In [38]:

```
qtable = np.zeros((state_size, action_size))
total_episodes = 5000        # Total episodes
alpha = 0.75           # Learning rate
max_steps = 99                # Max steps per episode
gamma = 0.85                  # Discounting rate

# Exploration parameters
epsilon = 1.0                 # Exploration rate
max_epsilon = 1.0             # Exploration probability at start
min_epsilon = 0.01            # Minimum exploration probability 
decay_rate = 0.01
```

In [45]:

```
# List of rewards
rewards = []

# 2 For life or until learning is stopped
for episode in range(total_episodes):
    # Reset the environment
    state = env.reset()
    step = 0
    done = False
    total_rewards = 0

    for step in range(max_steps):
        # 3. Choose an action a in the current world state (s)
        ## First we randomize a number
        exp_exp_tradeoff = random.uniform(0, 1)

        ## If this number > greater than epsilon --> exploitation (taking the bi
ggest Q value for this state)
        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state,:])

        # Else doing a random choice --> exploration
        else:
            action = env.action_space.sample()

        # Take the action (a) and observe the outcome state(s') and reward (r)
        new_state, reward, done, info = env.step(action)

        # Update Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') - Q(s,a)]
        # qtable[new_state,:] : all the actions we can take from new state
        qtable[state, action] = qtable[state, action] + alpha * (reward + gamma
* np.max(qtable[new_state, :]) - qtable[state, action])

        total_rewards += reward

        # Our new state is state
        state = new_state

        # If done (if we're dead) : finish episode
        if done == True:
            break

    # Reduce epsilon (because we need less and less exploration)
    epsilon = min_epsilon + (max_epsilon - min_epsilon)*np.exp(-decay_rate*episo
de)
    rewards.append(total_rewards)

print("******Performance of Frozen Lake agent at given hyperparameters******\n")
print("Total episodes: ", total_episodes )
print("Learning rate: ", alpha)
print("Discounting rate: ", gamma,"\n")
print ("Score over time: " +  str(sum(rewards)/total_episodes)+"\n")
print(qtable,"\n")
print("Epsilon at maximum steps: ", epsilon)
```

```
******Performance of Frozen Lake agent at given hyperparameters*****
*

Total episodes:  5000
Learning rate:  0.75
Discounting rate:  0.85

Score over time: 0.3394

[[1.13160078e-02 1.50824877e-03 1.38342705e-03 1.48666337e-03]
 [4.69567510e-04 1.92859777e-04 2.78123308e-04 2.78066509e-02]
 [6.82009032e-04 1.81063918e-04 2.56395844e-04 1.26152579e-02]
 [8.84992662e-04 7.44054542e-06 7.06267128e-05 7.53325585e-04]
 [3.65932089e-02 9.55403844e-04 3.78783202e-04 4.96932510e-04]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [6.06653021e-04 1.36825235e-05 2.04406503e-03 6.68053743e-08]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [2.77652663e-03 8.35450179e-04 5.92122799e-05 6.07809552e-02]
 [5.32533838e-03 1.78055707e-01 7.25641491e-03 1.10527660e-02]
 [5.37302806e-01 6.86925497e-04 2.56658627e-03 1.00407497e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [4.71663245e-04 4.17741634e-03 6.62523036e-01 1.26652604e-02]
 [6.30652167e-02 4.17437410e-02 9.61276811e-01 3.22237680e-02]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

Epsilon at maximum steps:  0.01
```

In [46]:

```
env.reset()
episode_steps = []
for episode in range(50):
    state = env.reset()
    step = 0
    done = False
    print("****************************************************")
    print("EPISODE ", episode)

    for step in range(max_steps):

        # Take the action (index) that have the maximum expected future reward g
iven that state
        action = np.argmax(qtable[state,:])

        new_state, reward, done, info = env.step(action)

        if done:
            # Here, we decide to only print the last state (to see if our agent
 is on the goal or fall into an hole)
            env.render()

            # We print the number of step it took.
            print("Number of steps", step)
            break
        state = new_state
    episode_steps.append(step)
env.close()
```

```
****************************************************
EPISODE  0
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 62
****************************************************
EPISODE  1
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 36
****************************************************
EPISODE  2
****************************************************
EPISODE  3
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 27
****************************************************
EPISODE  4
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 21
****************************************************
EPISODE  5
****************************************************
EPISODE  6
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 60
****************************************************
EPISODE  7
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
****************************************************
EPISODE  8
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 27
****************************************************
```

```
EPISODE  9
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
**************************************************
EPISODE  10
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 12
**************************************************
EPISODE  11
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
**************************************************
EPISODE  12
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 19
**************************************************
EPISODE  13
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 12
**************************************************
EPISODE  14
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 90
**************************************************
EPISODE  15
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 57
**************************************************
EPISODE  16
**************************************************
EPISODE  17
  (Left)
SFFF
```

```
FHFH
FFFH
HFFG
Number of steps 32
***************************************************
EPISODE  18
***************************************************
EPISODE  19
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
***************************************************
EPISODE  20
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 25
***************************************************
EPISODE  21
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 64
***************************************************
EPISODE  22
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 16
***************************************************
EPISODE  23
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 28
***************************************************
EPISODE  24
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 54
***************************************************
EPISODE  25
  (Right)
SFFF
FHFH
FFFH
HFFG
```

```
Number of steps 24
****************************************************
EPISODE  26
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 21
****************************************************
EPISODE  27
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 66
****************************************************
EPISODE  28
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
****************************************************
EPISODE  29
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 74
****************************************************
EPISODE  30
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 61
****************************************************
EPISODE  31
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 28
****************************************************
EPISODE  32
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 22
****************************************************
EPISODE  33
  (Right)
SFFF
```

```
FHFH
FFFH
HFFG
Number of steps 16
****************************************************
EPISODE  34
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 21
****************************************************
EPISODE  35
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 33
****************************************************
EPISODE  36
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 47
****************************************************
EPISODE  37
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 37
****************************************************
EPISODE  38
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 39
****************************************************
EPISODE  39
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
****************************************************
EPISODE  40
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
```

```
EPISODE  41
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 48
**************************************************
EPISODE  42
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 26
**************************************************
EPISODE  43
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 29
**************************************************
EPISODE  44
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 36
**************************************************
EPISODE  45
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 29
**************************************************
EPISODE  46
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 26
**************************************************
EPISODE  47
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 98
**************************************************
EPISODE  48
  (Right)
SFFF
FHFH
FFFH
```

```
HFFG
Number of steps 22
****************************************************
EPISODE  49
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
```

In [47]:

```
print(np.mean(episode_steps))
```

```
38.88
```

Testing with different hyperparameters (set 2)

The hyperparameters, alpha and gamma are changed to evaluate the change in baseline performance

In [48]:

```
qtable = np.zeros((state_size, action_size))
total_episodes = 5000        # Total episodes
alpha = 0.8           # Learning rate
max_steps = 99                # Max steps per episode
gamma = 0.9                  # Discounting rate

# Exploration parameters
epsilon = 1.0                 # Exploration rate
max_epsilon = 1.0             # Exploration probability at start
min_epsilon = 0.01            # Minimum exploration probability
decay_rate = 0.01
```

In [49]:

```
# List of rewards
rewards = []

# 2 For life or until learning is stopped
for episode in range(total_episodes):
    # Reset the environment
    state = env.reset()
    step = 0
    done = False
    total_rewards = 0

    for step in range(max_steps):
        # 3. Choose an action a in the current world state (s)
        ## First we randomize a number
        exp_exp_tradeoff = random.uniform(0, 1)

        ## If this number > greater than epsilon --> exploitation (taking the bi
ggest Q value for this state)
        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state,:])

        # Else doing a random choice --> exploration
        else:
            action = env.action_space.sample()

        # Take the action (a) and observe the outcome state(s') and reward (r)
        new_state, reward, done, info = env.step(action)

        # Update Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') - Q(s,a)]
        # qtable[new_state,:] : all the actions we can take from new state
        qtable[state, action] = qtable[state, action] + alpha * (reward + gamma
* np.max(qtable[new_state, :]) - qtable[state, action])

        total_rewards += reward

        # Our new state is state
        state = new_state

        # If done (if we're dead) : finish episode
        if done == True:
            break

    # Reduce epsilon (because we need less and less exploration)
    epsilon = min_epsilon + (max_epsilon - min_epsilon)*np.exp(-decay_rate*episo
de)
    rewards.append(total_rewards)

print("******Performance of Frozen Lake agent at given hyperparameters******\n")
print("Total episodes: ", total_episodes )
print("Learning rate: ", alpha)
print("Discounting rate: ", gamma,"\n")
print ("Score over time: " +  str(sum(rewards)/total_episodes)+"\n")
print(qtable,"\n")
print("Epsilon at maximum steps: ", epsilon)
```

```
******Performance of Frozen Lake agent at given hyperparameters*****
*

Total episodes:  5000
Learning rate:  0.8
Discounting rate:  0.9

Score over time: 0.412

[[4.19579971e-03 4.79686919e-02 4.17395556e-03 3.45251385e-02]
 [1.15564406e-03 9.90869412e-04 2.60558183e-05 6.49700166e-02]
 [5.25989338e-03 9.04894061e-02 1.51259971e-03 6.46026658e-04]
 [2.29185933e-02 5.26476265e-06 2.21085249e-03 2.15189542e-02]
 [4.87802653e-02 1.74782518e-02 7.74458436e-03 3.84230598e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.35941697e-06 2.29432615e-10 2.38298754e-02 3.77839587e-09]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [2.15271380e-02 1.10587738e-02 1.96082680e-03 2.15926744e-01]
 [9.33867175e-03 3.34334736e-01 5.67540697e-05 7.59446817e-03]
 [5.31205024e-01 4.42216269e-03 4.34607292e-04 9.39520516e-05]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [8.15731494e-03 1.04999196e-02 7.20868124e-01 1.89721660e-02]
 [1.65532966e-01 1.72397048e-01 1.69949941e-01 9.78180994e-01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

Epsilon at maximum steps:  0.01
```

In [50]:

```
env.reset()
episode_steps = []
for episode in range(50):
    state = env.reset()
    step = 0
    done = False
    print("****************************************************")
    print("EPISODE ", episode)

    for step in range(max_steps):

        # Take the action (index) that have the maximum expected future reward g
iven that state
        action = np.argmax(qtable[state,:])

        new_state, reward, done, info = env.step(action)

        if done:
            # Here, we decide to only print the last state (to see if our agent
 is on the goal or fall into an hole)
            env.render()

            # We print the number of step it took.
            print("Number of steps", step)
            break
        state = new_state
    episode_steps.append(step)
env.close()
```

```
***************************************************
EPISODE  0
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 22
***************************************************
EPISODE  1
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 4
***************************************************
EPISODE  2
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 23
***************************************************
EPISODE  3
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 43
***************************************************
EPISODE  4
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
***************************************************
EPISODE  5
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
***************************************************
EPISODE  6
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 25
***************************************************
EPISODE  7
  (Right)
SFFF
FHFH
```

```
FFFH
HFFG
Number of steps 42
***************************************************
EPISODE  8
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 20
***************************************************
EPISODE  9
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 48
***************************************************
EPISODE  10
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 7
***************************************************
EPISODE  11
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
***************************************************
EPISODE  12
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 22
***************************************************
EPISODE  13
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
***************************************************
EPISODE  14
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 53
***************************************************
EPISODE  15
```

```
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 32
**************************************************
EPISODE  16
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
**************************************************
EPISODE  17
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 16
**************************************************
EPISODE  18
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 13
**************************************************
EPISODE  19
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 40
**************************************************
EPISODE  20
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 80
**************************************************
EPISODE  21
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
**************************************************
EPISODE  22
  (Right)
SFFF
FHFH
FFFH
HFFG
```

```
Number of steps 35
**************************************************
EPISODE  23
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 7
**************************************************
EPISODE  24
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 55
**************************************************
EPISODE  25
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
**************************************************
EPISODE  26
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 13
**************************************************
EPISODE  27
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 89
**************************************************
EPISODE  28
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
**************************************************
EPISODE  29
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
**************************************************
EPISODE  30
  (Right)
SFFF
```

```
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
EPISODE  31
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
****************************************************
EPISODE  32
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
EPISODE  33
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 47
****************************************************
EPISODE  34
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 44
****************************************************
EPISODE  35
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 24
****************************************************
EPISODE  36
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 13
****************************************************
EPISODE  37
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 64
****************************************************
```

```
EPISODE  38
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
**************************************************
EPISODE  39
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
**************************************************
EPISODE  40
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 29
**************************************************
EPISODE  41
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 16
**************************************************
EPISODE  42
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 64
**************************************************
EPISODE  43
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
**************************************************
EPISODE  44
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
**************************************************
EPISODE  45
  (Right)
SFFF
FHFH
FFFH
```

```
HFFG
Number of steps 14
****************************************************
EPISODE  46
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
****************************************************
EPISODE  47
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
****************************************************
EPISODE  48
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 29
****************************************************
EPISODE  49
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 16
```

In [51]:

```
print(np.mean(episode_steps))
```

```
24.74
```

Testing with different hyperparameters (set 3)

The hyperparameters, epsilon and decay rate are changed to evaluate the change in baseline performance

In [52]:

```
qtable = np.zeros((state_size, action_size))
total_episodes = 5000        # Total episodes
alpha = 0.7               # Learning rate
max_steps = 99                # Max steps per episode
gamma = 0.8                  # Discounting rate

# Exploration parameters
epsilon = 0.9                 # Exploration rate
max_epsilon = 1.0             # Exploration probability at start
min_epsilon = 0.01            # Minimum exploration probability
decay_rate = 0.0075
```

In [53]:

```
# List of rewards
rewards = []

# 2 For life or until learning is stopped
for episode in range(total_episodes):
    # Reset the environment
    state = env.reset()
    step = 0
    done = False
    total_rewards = 0

    for step in range(max_steps):
        # 3. Choose an action a in the current world state (s)
        ## First we randomize a number
        exp_exp_tradeoff = random.uniform(0, 1)

        ## If this number > greater than epsilon --> exploitation (taking the bi
ggest Q value for this state)
        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state,:])

        # Else doing a random choice --> exploration
        else:
            action = env.action_space.sample()

        # Take the action (a) and observe the outcome state(s') and reward (r)
        new_state, reward, done, info = env.step(action)

        # Update Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') - Q(s,a)]
        # qtable[new_state,:] : all the actions we can take from new state
        qtable[state, action] = qtable[state, action] + alpha * (reward + gamma
* np.max(qtable[new_state, :]) - qtable[state, action])

        total_rewards += reward

        # Our new state is state
        state = new_state

        # If done (if we're dead) : finish episode
        if done == True:
            break

    # Reduce epsilon (because we need less and less exploration)
    epsilon = min_epsilon + (max_epsilon - min_epsilon)*np.exp(-decay_rate*episo
de)
    rewards.append(total_rewards)

print("******Performance of Frozen Lake agent at given hyperparameters******\n")
print("Total episodes: ", total_episodes )
print("Learning rate: ", alpha)
print("Discounting rate: ", gamma,"\n")
print ("Score over time: " +  str(sum(rewards)/total_episodes)+"\n")
print(qtable,"\n")
print("Epsilon at maximum steps: ", epsilon)
```

```
******Performance of Frozen Lake agent at given hyperparameters*****
*

Total episodes:  5000
Learning rate:  0.7
Discounting rate:  0.8

Score over time: 0.293

[[2.46591893e-03 1.27173914e-02 1.43329347e-02 5.76927960e-04]
 [1.68677139e-05 1.54512132e-04 8.83079117e-04 2.15058730e-02]
 [8.02604119e-02 4.30267679e-04 4.65529422e-05 1.50596490e-02]
 [2.95549758e-05 4.70940679e-05 2.54030966e-05 1.91780046e-02]
 [2.78508895e-02 2.17250252e-04 9.46577657e-03 7.30857425e-05]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.54498518e-01 1.95619494e-07 1.39166865e-04 3.08870430e-06]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [9.55038846e-04 9.09176005e-05 2.22041974e-02 8.55500529e-02]
 [1.48457496e-01 2.50422413e-01 1.59340501e-03 6.19490169e-03]
 [1.82500570e-01 1.75719493e-01 1.76832450e-03 9.28656741e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [2.70989556e-03 3.79872467e-04 6.22528082e-01 1.08407775e-03]
 [1.89521200e-01 2.45786790e-01 9.93993855e-01 7.14529128e-02]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

Epsilon at maximum steps:  0.010000000000000052
```

In [54]:

```
env.reset()
episode_steps = []
for episode in range(50):
    state = env.reset()
    step = 0
    done = False
    print("****************************************************")
    print("EPISODE ", episode)

    for step in range(max_steps):

        # Take the action (index) that have the maximum expected future reward g
iven that state
        action = np.argmax(qtable[state,:])

        new_state, reward, done, info = env.step(action)

        if done:
            # Here, we decide to only print the last state (to see if our agent
 is on the goal or fall into an hole)
            env.render()

            # We print the number of step it took.
            print("Number of steps", step)
            break
        state = new_state
    episode_steps.append(step)
env.close()
```

```
***************************************************
EPISODE  0
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 33
***************************************************
EPISODE  1
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 12
***************************************************
EPISODE  2
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 58
***************************************************
EPISODE  3
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 24
***************************************************
EPISODE  4
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 42
***************************************************
EPISODE  5
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 23
***************************************************
EPISODE  6
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
***************************************************
EPISODE  7
  (Right)
SFFF
FHFH
```

```
FFFH
HFFG
Number of steps 36
**************************************************
EPISODE  8
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 43
**************************************************
EPISODE  9
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 57
**************************************************
EPISODE  10
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 29
**************************************************
EPISODE  11
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 25
**************************************************
EPISODE  12
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 25
**************************************************
EPISODE  13
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 20
**************************************************
EPISODE  14
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
**************************************************
EPISODE  15
```

```
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 28
****************************************************
EPISODE  16
****************************************************
EPISODE  17
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 13
****************************************************
EPISODE  18
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
****************************************************
EPISODE  19
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 28
****************************************************
EPISODE  20
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
EPISODE  21
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
****************************************************
EPISODE  22
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 41
****************************************************
EPISODE  23
  (Left)
SFFF
FHFH
```

```
FFFH
HFFG
Number of steps 15
**************************************************
EPISODE  24
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 21
**************************************************
EPISODE  25
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 69
**************************************************
EPISODE  26
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 44
**************************************************
EPISODE  27
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
**************************************************
EPISODE  28
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 27
**************************************************
EPISODE  29
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 20
**************************************************
EPISODE  30
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
**************************************************
EPISODE  31
```

```
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 16
****************************************************
EPISODE  32
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
****************************************************
EPISODE  33
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 54
****************************************************
EPISODE  34
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
****************************************************
EPISODE  35
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 34
****************************************************
EPISODE  36
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
****************************************************
EPISODE  37
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 42
****************************************************
EPISODE  38
  (Left)
SFFF
FHFH
FFFH
HFFG
```

```
Number of steps 10
***************************************************
EPISODE  39
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
***************************************************
EPISODE  40
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 36
***************************************************
EPISODE  41
***************************************************
EPISODE  42
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 19
***************************************************
EPISODE  43
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 21
***************************************************
EPISODE  44
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 44
***************************************************
EPISODE  45
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 49
***************************************************
EPISODE  46
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 61
***************************************************
EPISODE  47
```

```
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 31
****************************************************
EPISODE  48
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
****************************************************
EPISODE  49
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 13
```

In [55]:

```
print(np.mean(episode_steps))
```

```
30.0
```

Testing with different hyperparameters (set 4)

The hyperparameters, epsilon and decay rate are changed to evaluate the change in baseline performance

In [67]:

```
qtable = np.zeros((state_size, action_size))
total_episodes = 5000        # Total episodes
alpha = 0.7               # Learning rate
max_steps = 99                # Max steps per episode
gamma = 0.8                  # Discounting rate

# Exploration parameters
epsilon = 0.8                 # Exploration rate
max_epsilon = 1.0             # Exploration probability at start
min_epsilon = 0.01            # Minimum exploration probability
decay_rate = 0.005
```

In [68]:

```
# List of rewards
rewards = []

# 2 For life or until learning is stopped
for episode in range(total_episodes):
    # Reset the environment
    state = env.reset()
    step = 0
    done = False
    total_rewards = 0

    for step in range(max_steps):
        # 3. Choose an action a in the current world state (s)
        ## First we randomize a number
        exp_exp_tradeoff = random.uniform(0, 1)

        ## If this number > greater than epsilon --> exploitation (taking the bi
ggest Q value for this state)
        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state,:])

        # Else doing a random choice --> exploration
        else:
            action = env.action_space.sample()

        # Take the action (a) and observe the outcome state(s') and reward (r)
        new_state, reward, done, info = env.step(action)

        # Update Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') - Q(s,a)]
        # qtable[new_state,:] : all the actions we can take from new state
        qtable[state, action] = qtable[state, action] + alpha * (reward + gamma
* np.max(qtable[new_state, :]) - qtable[state, action])

        total_rewards += reward

        # Our new state is state
        state = new_state

        # If done (if we're dead) : finish episode
        if done == True:
            break

    # Reduce epsilon (because we need less and less exploration)
    epsilon = min_epsilon + (max_epsilon - min_epsilon)*np.exp(-decay_rate*episo
de)
    rewards.append(total_rewards)

print("******Performance of Frozen Lake agent at given hyperparameters******\n")
print("Total episodes: ", total_episodes )
print("Learning rate: ", alpha)
print("Discounting rate: ", gamma,"\n")
print ("Score over time: " +  str(sum(rewards)/total_episodes)+"\n")
print(qtable,"\n")
print("Epsilon at maximum steps: ", epsilon)
```

```
******Performance of Frozen Lake agent at given hyperparameters*****
*

Total episodes:  5000
Learning rate:  0.7
Discounting rate:  0.8

Score over time: 0.2738

[[3.32596938e-04 2.76879939e-04 3.75150322e-03 3.10437671e-04]
 [1.28398189e-03 1.95585622e-04 1.66492183e-04 3.85425278e-03]
 [3.78565798e-03 3.21847329e-03 5.72972227e-04 6.85931289e-05]
 [1.20497562e-05 2.16542260e-05 1.84136772e-05 3.13883956e-03]
 [9.21004581e-02 1.75528957e-05 2.78671261e-04 1.31928576e-02]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.47381805e-02 1.02361982e-06 1.55233130e-05 1.76147344e-05]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [9.01712278e-04 1.34987038e-03 1.02822133e-03 1.16595749e-01]
 [3.45396178e-03 3.96192216e-01 3.45368609e-04 1.21009108e-02]
 [1.96460108e-01 4.96482487e-03 2.42965123e-03 9.72948467e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [7.88153671e-03 1.15052760e-02 7.47503370e-01 9.70988274e-03]
 [2.34856443e-02 3.34143416e-02 4.25134287e-02 8.01332492e-01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

Epsilon at maximum steps:  0.010000000013817982
```

In [69]:

```
env.reset()
episode_steps = []
for episode in range(50):
    state = env.reset()
    step = 0
    done = False
    print("****************************************************")
    print("EPISODE ", episode)

    for step in range(max_steps):

        # Take the action (index) that have the maximum expected future reward g
iven that state
        action = np.argmax(qtable[state,:])

        new_state, reward, done, info = env.step(action)

        if done:
            # Here, we decide to only print the last state (to see if our agent
 is on the goal or fall into an hole)
            env.render()

            # We print the number of step it took.
            print("Number of steps", step)
            break
        state = new_state
    episode_steps.append(step)
env.close()
```

```
****************************************************
EPISODE  0
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
****************************************************
EPISODE  1
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 29
****************************************************
EPISODE  2
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 23
****************************************************
EPISODE  3
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 28
****************************************************
EPISODE  4
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 29
****************************************************
EPISODE  5
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 12
****************************************************
EPISODE  6
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
****************************************************
EPISODE  7
  (Left)
SFFF
FHFH
```

```
FFFH
HFFG
Number of steps 13
****************************************************
EPISODE  8
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 26
****************************************************
EPISODE  9
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 23
****************************************************
EPISODE  10
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 28
****************************************************
EPISODE  11
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
****************************************************
EPISODE  12
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 3
****************************************************
EPISODE  13
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
****************************************************
EPISODE  14
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
****************************************************
EPISODE  15
```

```
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
***************************************************
EPISODE  16
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 56
***************************************************
EPISODE  17
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 58
***************************************************
EPISODE  18
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
***************************************************
EPISODE  19
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 70
***************************************************
EPISODE  20
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 59
***************************************************
EPISODE  21
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 48
***************************************************
EPISODE  22
  (Left)
SFFF
FHFH
FFFH
HFFG
```

```
Number of steps 23
****************************************************
EPISODE  23
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 45
****************************************************
EPISODE  24
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 63
****************************************************
EPISODE  25
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 24
****************************************************
EPISODE  26
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 31
****************************************************
EPISODE  27
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
****************************************************
EPISODE  28
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 53
****************************************************
EPISODE  29
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
EPISODE  30
  (Left)
SFFF
```

```
FHFH
FFFH
HFFG
Number of steps 21
****************************************************
EPISODE  31
****************************************************
EPISODE  32
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
****************************************************
EPISODE  33
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
****************************************************
EPISODE  34
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 94
****************************************************
EPISODE  35
****************************************************
EPISODE  36
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 36
****************************************************
EPISODE  37
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 15
****************************************************
EPISODE  38
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 40
****************************************************
EPISODE  39
  (Left)
SFFF
FHFH
```

```
FFFH
HFFG
Number of steps 90
****************************************************
EPISODE  40
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 21
****************************************************
EPISODE  41
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 69
****************************************************
EPISODE  42
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
EPISODE  43
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
****************************************************
EPISODE  44
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 49
****************************************************
EPISODE  45
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 75
****************************************************
EPISODE  46
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 24
****************************************************
EPISODE  47
```

```
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 62
****************************************************
EPISODE  48
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
****************************************************
EPISODE  49
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 50
```

In [70]:

```
print(np.mean(episode_steps))
```

```
34.32
```

Changing policy to 'Mean' of the action values at that state and observing the performance

In [74]:

```
qtable = np.zeros((state_size, action_size))
total_episodes = 10000        # Total episodes
alpha = 0.7              # Learning rate
max_steps = 99                # Max steps per episode
gamma = 0.8                  # Discounting rate

# Exploration parameters
epsilon = 1.0                 # Exploration rate
max_epsilon = 1.0             # Exploration probability at start
min_epsilon = 0.01            # Minimum exploration probability
decay_rate = 0.01
```

In [77]:

```
# List of rewards
rewards = []

# 2 For life or until learning is stopped
for episode in range(total_episodes):
    # Reset the environment
    state = env.reset()
    step = 0
    done = False
    total_rewards = 0

    for step in range(max_steps):
        # 3. Choose an action a in the current world state (s)
        ## First we randomize a number
        exp_exp_tradeoff = random.uniform(0, 1)

        ## If this number > greater than epsilon --> exploitation (taking the bi
ggest Q value for this state)
        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state,:])

        # Else doing a random choice --> exploration
        else:
            action = env.action_space.sample()

        # Take the action (a) and observe the outcome state(s') and reward (r)
        new_state, reward, done, info = env.step(action)

        # Update Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') - Q(s,a)]
        # qtable[new_state,:] : all the actions we can take from new state
        qtable[state, action] = qtable[state, action] + alpha * (reward + gamma
* np.mean(qtable[new_state, :]) - qtable[state, action])

        total_rewards += reward

        # Our new state is state
        state = new_state

        # If done (if we're dead) : finish episode
        if done == True:
            break

    # Reduce epsilon (because we need less and less exploration)
    epsilon = min_epsilon + (max_epsilon - min_epsilon)*np.exp(-decay_rate*episo
de)
    rewards.append(total_rewards)

print("******Performance of Frozen Lake agent with Mean of action values as the
 policy******\n")
print("Total episodes: ", total_episodes )
print("Learning rate: ", alpha)
print("Discounting rate: ", gamma)
print("Decay rate: ", decay_rate,"\n")
print ("Score over time: " +  str(sum(rewards)/total_episodes)+"\n")
print(qtable,"\n")
print("Epsilon at maximum steps: ", epsilon)
```

```
******Performance of Frozen Lake agent with Mean of action values as
the policy******

Total episodes:  10000
Learning rate:  0.7
Discounting rate:  0.8
Decay rate:  0.01

Score over time: 0.096

[[4.98243206e-11 6.88566746e-08 2.97705443e-10 1.58714057e-09]
 [1.38760635e-12 4.96834519e-10 1.86815630e-11 1.04298665e-08]
 [4.19965186e-11 5.67008113e-11 5.05512767e-08 3.89020911e-11]
 [3.93373830e-14 1.04225875e-12 2.00110277e-13 2.37802592e-08]
 [5.41031511e-08 2.12188990e-11 2.19384442e-08 7.28067814e-08]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [6.47383329e-10 1.27563565e-12 6.31201289e-08 1.10587017e-12]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [4.80034388e-11 1.48157899e-11 6.42321977e-10 1.34441053e-05]
 [9.56334382e-09 1.37933971e-02 4.04121202e-08 4.91458630e-08]
 [2.95085268e-02 3.37155174e-07 7.76470948e-09 3.34344676e-07]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.08201289e-07 1.09707624e-07 1.17157260e-07 8.58117495e-03]
 [4.52642396e-05 4.03550455e-05 5.31297201e-05 7.87358599e-01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

Epsilon at maximum steps:  0.01
```

In [78]:

```
env.reset()
episode_steps = []
for episode in range(50):
    state = env.reset()
    step = 0
    done = False
    print("****************************************************")
    print("EPISODE ", episode)

    for step in range(max_steps):

        # Take the action (index) that have the maximum expected future reward g
iven that state
        action = np.argmax(qtable[state,:])

        new_state, reward, done, info = env.step(action)

        if done:
            # Here, we decide to only print the last state (to see if our agent
 is on the goal or fall into an hole)
            env.render()

            # We print the number of step it took.
            print("Number of steps", step)
            break
        state = new_state
    episode_steps.append(step)
env.close()
```

```
***************************************************
EPISODE  0
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
***************************************************
EPISODE  1
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 1
***************************************************
EPISODE  2
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
***************************************************
EPISODE  3
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 15
***************************************************
EPISODE  4
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
***************************************************
EPISODE  5
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 4
***************************************************
EPISODE  6
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
***************************************************
EPISODE  7
  (Right)
SFFF
FHFH
```

```
FFFH
HFFG
Number of steps 6
****************************************************
EPISODE  8
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 2
****************************************************
EPISODE  9
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 13
****************************************************
EPISODE  10
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
****************************************************
EPISODE  11
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 1
****************************************************
EPISODE  12
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 4
****************************************************
EPISODE  13
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
****************************************************
EPISODE  14
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
EPISODE  15
```

```
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 3
**************************************************
EPISODE  16
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
**************************************************
EPISODE  17
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 1
**************************************************
EPISODE  18
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 1
**************************************************
EPISODE  19
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
**************************************************
EPISODE  20
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
**************************************************
EPISODE  21
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 3
**************************************************
EPISODE  22
  (Right)
SFFF
FHFH
FFFH
HFFG
```

```
Number of steps 18
****************************************************
EPISODE  23
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
EPISODE  24
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
EPISODE  25
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
****************************************************
EPISODE  26
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
****************************************************
EPISODE  27
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 13
****************************************************
EPISODE  28
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
****************************************************
EPISODE  29
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
****************************************************
EPISODE  30
  (Right)
SFFF
```

```
FHFH
FFFH
HFFG
Number of steps 19
*****************************************************
EPISODE  31
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 2
*****************************************************
EPISODE  32
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
*****************************************************
EPISODE  33
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
*****************************************************
EPISODE  34
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 26
*****************************************************
EPISODE  35
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 2
*****************************************************
EPISODE  36
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
*****************************************************
EPISODE  37
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
*****************************************************
```

```
EPISODE  38
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 2
**************************************************
EPISODE  39
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 2
**************************************************
EPISODE  40
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 2
**************************************************
EPISODE  41
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 15
**************************************************
EPISODE  42
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 4
**************************************************
EPISODE  43
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 3
**************************************************
EPISODE  44
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 1
**************************************************
EPISODE  45
  (Up)
SFFF
FHFH
FFFH
```

```
HFFG
Number of steps 30
****************************************************
EPISODE  46
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 2
****************************************************
EPISODE  47
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 45
****************************************************
EPISODE  48
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 3
****************************************************
EPISODE  49
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 2
```

Changing policy to 'Min' of the action values at that state and observing the performance

In [79]:

```
qtable = np.zeros((state_size, action_size))
total_episodes = 10000        # Total episodes
alpha = 0.7             # Learning rate
max_steps = 99                # Max steps per episode
gamma = 0.8                  # Discounting rate

# Exploration parameters
epsilon = 1.0                 # Exploration rate
max_epsilon = 1.0             # Exploration probability at start
min_epsilon = 0.01            # Minimum exploration probability
decay_rate = 0.01
```

In [80]:

```
# List of rewards
rewards = []

# 2 For life or until learning is stopped
for episode in range(total_episodes):
    # Reset the environment
    state = env.reset()
    step = 0
    done = False
    total_rewards = 0

    for step in range(max_steps):
        # 3. Choose an action a in the current world state (s)
        ## First we randomize a number
        exp_exp_tradeoff = random.uniform(0, 1)

        ## If this number > greater than epsilon --> exploitation (taking the bi
ggest Q value for this state)
        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state,:])

        # Else doing a random choice --> exploration
        else:
            action = env.action_space.sample()

        # Take the action (a) and observe the outcome state(s') and reward (r)
        new_state, reward, done, info = env.step(action)

        # Update Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') - Q(s,a)]
        # qtable[new_state,:] : all the actions we can take from new state
        qtable[state, action] = qtable[state, action] + alpha * (reward + gamma
* np.mean(qtable[new_state, :]) - qtable[state, action])

        total_rewards += reward

        # Our new state is state
        state = new_state

        # If done (if we're dead) : finish episode
        if done == True:
            break

    # Reduce epsilon (because we need less and less exploration)
    epsilon = min_epsilon + (max_epsilon - min_epsilon)*np.exp(-decay_rate*episo
de)
    rewards.append(total_rewards)

print("******Performance of Frozen Lake agent with 'Min of action values' as the
policy******\n")
print("Total episodes: ", total_episodes )
print("Learning rate: ", alpha)
print("Discounting rate: ", gamma)
print("Decay rate: ", decay_rate,"\n")
print ("Score over time: " +  str(sum(rewards)/total_episodes)+"\n")
print(qtable,"\n")
print("Epsilon at maximum steps: ", epsilon)
```

```
******Performance of Frozen Lake agent with 'Min of action values' a
s the policy******

Total episodes:  10000
Learning rate:  0.7
Discounting rate:  0.8
Decay rate:  0.01

Score over time: 0.1121

[[1.11193686e-09 4.31928739e-09 3.68317764e-06 1.21978958e-09]
 [6.20996383e-11 3.54435590e-10 6.38903376e-11 4.17952057e-09]
 [5.39188343e-10 2.83724787e-04 8.99206882e-10 8.74871799e-10]
 [8.62342040e-13 8.73076680e-10 4.88486680e-14 5.09512959e-13]
 [2.97106227e-06 9.63996341e-10 2.67057056e-10 1.78026435e-10]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [2.72191984e-09 1.82393683e-04 1.30551725e-09 1.09391411e-09]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [5.87698329e-10 8.94037566e-08 1.34721467e-07 4.85730625e-06]
 [1.05884050e-06 1.67823779e-03 7.63705122e-06 8.56135672e-07]
 [3.57126218e-03 3.77257833e-07 5.05958549e-06 3.30531907e-04]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [5.33792132e-05 2.59274358e-07 3.79703642e-05 9.07215835e-07]
 [3.39952244e-04 3.81391577e-03 4.42339566e-04 4.95685659e-05]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

Epsilon at maximum steps:  0.01
```

In [81]:

```
env.reset()
episode_steps = []
for episode in range(50):
    state = env.reset()
    step = 0
    done = False
    print("****************************************************")
    print("EPISODE ", episode)

    for step in range(max_steps):

        # Take the action (index) that have the maximum expected future reward g
iven that state
        action = np.argmax(qtable[state,:])

        new_state, reward, done, info = env.step(action)

        if done:
            # Here, we decide to only print the last state (to see if our agent
 is on the goal or fall into an hole)
            env.render()

            # We print the number of step it took.
            print("Number of steps", step)
            break
        state = new_state
    episode_steps.append(step)
env.close()
```

```
****************************************************
EPISODE  0
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 3
****************************************************
EPISODE  1
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 48
****************************************************
EPISODE  2
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
****************************************************
EPISODE  3
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 13
****************************************************
EPISODE  4
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 27
****************************************************
EPISODE  5
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
****************************************************
EPISODE  6
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 13
****************************************************
EPISODE  7
  (Down)
SFFF
FHFH
```

```
FFFH
HFFG
Number of steps 13
****************************************************
EPISODE  8
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
****************************************************
EPISODE  9
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
EPISODE  10
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
****************************************************
EPISODE  11
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 7
****************************************************
EPISODE  12
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 30
****************************************************
EPISODE  13
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 7
****************************************************
EPISODE  14
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 26
****************************************************
EPISODE  15
```

```
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
**************************************************
EPISODE  16
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 4
**************************************************
EPISODE  17
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 22
**************************************************
EPISODE  18
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
**************************************************
EPISODE  19
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 82
**************************************************
EPISODE  20
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 29
**************************************************
EPISODE  21
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 27
**************************************************
EPISODE  22
  (Down)
SFFF
FHFH
FFFH
HFFG
```

```
Number of steps 6
****************************************************
EPISODE  23
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 19
****************************************************
EPISODE  24
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
****************************************************
EPISODE  25
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 36
****************************************************
EPISODE  26
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 38
****************************************************
EPISODE  27
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 3
****************************************************
EPISODE  28
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 27
****************************************************
EPISODE  29
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 24
****************************************************
EPISODE  30
  (Down)
SFFF
```

```
FHFH
FFFH
HFFG
Number of steps 66
****************************************************
EPISODE  31
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 26
****************************************************
EPISODE  32
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
****************************************************
EPISODE  33
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 51
****************************************************
EPISODE  34
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
****************************************************
EPISODE  35
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
****************************************************
EPISODE  36
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 7
****************************************************
EPISODE  37
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
****************************************************
```

```
EPISODE  38
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 28
**************************************************
EPISODE  39
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 7
**************************************************
EPISODE  40
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 3
**************************************************
EPISODE  41
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
**************************************************
EPISODE  42
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 38
**************************************************
EPISODE  43
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
**************************************************
EPISODE  44
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 8
**************************************************
EPISODE  45
  (Down)
SFFF
FHFH
FFFH
```

```
HFFG
Number of steps 24
****************************************************
EPISODE  46
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 17
****************************************************
EPISODE  47
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 27
****************************************************
EPISODE  48
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 5
****************************************************
EPISODE  49
  (Left)
SFFF
FHFH
FFFH
HFFG
Number of steps 10
```

Improved performance results

In [82]:

```
qtable = np.zeros((state_size, action_size))
total_episodes = 40000        # Total episodes
alpha = 0.85           # Learning rate
max_steps = 99                # Max steps per episode
gamma = 0.95                  # Discounting rate

# Exploration parameters
epsilon = 1.0                 # Exploration rate
max_epsilon = 1.0             # Exploration probability at start
min_epsilon = 0.01            # Minimum exploration probability
decay_rate = 0.001
```

In [83]:

```
# List of rewards
rewards = []

# 2 For life or until learning is stopped
for episode in range(total_episodes):
    # Reset the environment
    state = env.reset()
    step = 0
    done = False
    total_rewards = 0

    for step in range(max_steps):
        # 3. Choose an action a in the current world state (s)
        ## First we randomize a number
        exp_exp_tradeoff = random.uniform(0, 1)

        ## If this number > greater than epsilon --> exploitation (taking the bi
ggest Q value for this state)
        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state,:])

        # Else doing a random choice --> exploration
        else:
            action = env.action_space.sample()

        # Take the action (a) and observe the outcome state(s') and reward (r)
        new_state, reward, done, info = env.step(action)

        # Update Q(s,a):= Q(s,a) + lr [R(s,a) + gamma * max Q(s',a') - Q(s,a)]
        # qtable[new_state,:] : all the actions we can take from new state
        qtable[state, action] = qtable[state, action] + alpha * (reward + gamma
* np.max(qtable[new_state, :]) - qtable[state, action])

        total_rewards += reward

        # Our new state is state
        state = new_state

        # If done (if we're dead) : finish episode
        if done == True:
            break

    # Reduce epsilon (because we need less and less exploration)
    epsilon = min_epsilon + (max_epsilon - min_epsilon)*np.exp(-decay_rate*episo
de)
    rewards.append(total_rewards)

print("****** Improved performance of Frozen Lake agent ******\n")
print("Total episodes: ", total_episodes )
print("Learning rate: ", alpha)
print("Discounting rate: ", gamma)
print("Decay rate: ", decay_rate,"\n")
print ("Score over time: " +  str(sum(rewards)/total_episodes)+"\n")
print(qtable,"\n")
print("Epsilon at maximum steps: ", epsilon)
```

```
****** Improved performance of Frozen Lake agent ******

Total episodes:  40000
Learning rate:  0.85
Discounting rate:  0.95
Decay rate:  0.001

Score over time: 0.450725

[[3.84899447e-02 4.48261812e-01 3.74964246e-02 3.82613384e-02]
 [3.91016382e-04 4.56372086e-03 6.24860963e-03 5.61429834e-02]
 [2.31659308e-03 6.72197764e-03 2.86539947e-03 2.90275529e-02]
 [1.88859043e-03 3.37481095e-03 1.62519184e-03 2.41124548e-02]
 [5.49727654e-01 3.00330164e-03 3.42980271e-02 1.17360806e-02]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [4.55684262e-04 7.01484813e-10 2.11124168e-05 1.06732982e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.19181272e-04 7.97120704e-03 7.11577540e-03 5.94555563e-01]
 [1.36466029e-06 1.09477649e-01 1.49040318e-02 1.28162595e-03]
 [8.08142483e-01 4.85343311e-05 1.99804114e-03 3.24233988e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [8.16671850e-02 1.28646901e-03 8.57887104e-01 3.08036930e-03]
 [8.81893765e-02 9.78922591e-01 1.34920677e-01 2.15466792e-01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

Epsilon at maximum steps:  0.010000000000000004
```

In [84]:

```
env.reset()
episode_steps = []
for episode in range(50):
    state = env.reset()
    step = 0
    done = False
    print("****************************************************")
    print("EPISODE ", episode)

    for step in range(max_steps):

        # Take the action (index) that have the maximum expected future reward g
iven that state
        action = np.argmax(qtable[state,:])

        new_state, reward, done, info = env.step(action)

        if done:
            # Here, we decide to only print the last state (to see if our agent
 is on the goal or fall into an hole)
            env.render()

            # We print the number of step it took.
            print("Number of steps", step)
            break
        state = new_state
    episode_steps.append(step)
env.close()
```

```
****************************************************
EPISODE  0
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 26
****************************************************
EPISODE  1
****************************************************
EPISODE  2
****************************************************
EPISODE  3
****************************************************
EPISODE  4
****************************************************
EPISODE  5
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
****************************************************
EPISODE  6
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 74
****************************************************
EPISODE  7
****************************************************
EPISODE  8
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 18
****************************************************
EPISODE  9
****************************************************
EPISODE  10
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 35
****************************************************
EPISODE  11
****************************************************
EPISODE  12
****************************************************
EPISODE  13
  (Down)
SFFF
FHFH
```

```
FFFH
HFFG
Number of steps 95
****************************************************
EPISODE  14
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 84
****************************************************
EPISODE  15
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 26
****************************************************
EPISODE  16
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 39
****************************************************
EPISODE  17
****************************************************
EPISODE  18
****************************************************
EPISODE  19
****************************************************
EPISODE  20
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 33
****************************************************
EPISODE  21
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 15
****************************************************
EPISODE  22
****************************************************
EPISODE  23
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 34
****************************************************
EPISODE  24
```

```
****************************************************
EPISODE  25
****************************************************
EPISODE  26
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 31
****************************************************
EPISODE  27
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
****************************************************
EPISODE  28
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 20
****************************************************
EPISODE  29
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 70
****************************************************
EPISODE  30
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
****************************************************
EPISODE  31
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 6
****************************************************
EPISODE  32
****************************************************
EPISODE  33
****************************************************
EPISODE  34
  (Down)
SFFF
FHFH
FFFH
HFFG
```

```
Number of steps 12
****************************************************
EPISODE  35
****************************************************
EPISODE  36
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
****************************************************
EPISODE  37
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 38
****************************************************
EPISODE  38
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 64
****************************************************
EPISODE  39
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 35
****************************************************
EPISODE  40
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 53
****************************************************
EPISODE  41
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 12
****************************************************
EPISODE  42
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 24
****************************************************
EPISODE  43
```

```
  (Up)
SFFF
FHFH
FFFH
HFFG
Number of steps 7
****************************************************
EPISODE  44
****************************************************
EPISODE  45
****************************************************
EPISODE  46
****************************************************
EPISODE  47
****************************************************
EPISODE  48
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 14
****************************************************
EPISODE  49
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 9
```

In [85]:

```
print(np.mean(episode_steps))
```

```
59.78
```

Questions & Answers

1. Establish a baseline performance. How well did your RL Q-learning do on your problem?

```
Baseline performance - Score over time: 0.3412
```

2. What are the states, the actions and the size of the Q-table?

```
 There are 16 states and 4 actions.

 States (4 in each stage):
 1. SFFF
 2. FHFH
 3. FFFH
 4. HFFG

 S - start state
 F - frozen
 H - Hole
 G - Goal, Frisbee

Actions:
 Left
 Right
 Up
 Down

Size of Q-table:
 (4,16) - 4 actions and 16 states
```

- What are the rewards? Why did you choose them?

```
  S - start state (safe, reward = 0)
  F - frozen (safe, reward = 0)
  H - Hole (reward = -1, terminate the episode)
  G - Goal (reward = 1, terminate the episode)
```

- How did you choose alpha and gamma in the following equation?

```
  1. Alpha and gamma are chosen based on the weightage given to prior
 knowledge and future rewards respectively.
  2. For baseline performance alpha is given as 0.7 which means 70% we
ightage given to prior knowledge and gamma is chosen as 0.8 considerin
g 80% weightage to the future rewards at the state.
```

- Try at least one additional value for alpha and gamma. How did it change the baseline performance?

```
  1. When an increase of 0.05 to alpha and gamma from (0.7,0.8) to (0.
75,0.85), the baseline performance has increased by around 0.04 points
from 0.321 to 0.3394
  2. When an increase of 0.05 to alpha and gamma from (0.7,0.8) to (0.
8,0.90), the baseline performance has increased by around 0.09 points
 from 0.321 to 0.412
```

- Try a policy other than maxQ(s', a'). How did it change the baseline performance?

```
  1. For different policy which is meanQ(s',a') the performance is gre
atly reduced from 0.321 to 0.096.
  2. For different policy which is minQ(s',a') the performance is grea
tly reduced from 0.321 to 0.1121.
```

- How did you choose your decay rate and starting epsilon? Try at least one additional value for epsilon and the decay rate. How did it change the baseline performance? What is the value of epsilon when if you reach the max steps per episode?

```
  1. The decay rate and starting epsilon are considered to 0.01 and 1.
0 respectively. Intially the epsilon should be high to increase the po
ssibility of exploration and later it should reduce to increase the po
ssibility of exploitation.
  2. For a starting epsilon of 0.9, decay rate of 0.0075 the score ove
r time reduced from 0.321 to 0.293.
  3. For a starting epsilon of 0.8, decay rate of 0.005 the score over
time reduced from 0.321 to 0.2738.
  4. The value of epsilon at max steps is around 0.01.
```

- What is the average number of steps taken per episode?

```
  1. For the baseline performance of score over time 0.321, the averag
e steps taken are 18.86 or 19 approx.
  2. For the improved performance of score over time 0.450725, the ave
rage steps taken are 37.44 or 37 approx.
```

- Does Q-learning use value-based or policy-based iteration?

```
  Value-based because the Q-table is updated each time with the future
value of the action at that state.
```

- What is meant by expected lifetime value in the Bellman equation?

```
  Expected lifetime value in the Bellman's equation derived from the c
oncept of converging the future reward value while calculating the Q-v
alue function.
  The discounting rate is introduced to add weightage to the future re
ward (usually between 0 and 1) and get the value using the discounting
rate. The last term in the Bellman's equation refers to the expected l
ifetime value.
```

Conclusion

```
    1. Initially the baseline performance is set with certain hyperparamet
ers and the reinforcement learning model is trained using the Q-table. The
Q-table is updated using the Bellman's equation by giving weightage to the
future reward and prior knowledge.
    2. With hyperparameter tuning, the model is improved by changing numbe
r of training episodes, alpha(learning rate), gamma(discounting rate), sta
rting epsilon and the decay rate.
    3. It has been observed that the model performed poorly with 5000 epis
odes and later the model showed a bit improved when the alpha and gamma ar
e increased. However, the model required a big number of training episodes
to show significant. Hence, a set of 40000 episodes, (learning rate, disco
unting rate) as (0.85,0.95), starting epsilon as 1.0, decay rate as low as
0.001 resulted in an increased performance of 0.450725 which is an increas
e of 40% from the baseline performance.
    4. Meanwhile, it is also concluded that using different policy functio
ns like meanQ() and minQ() resulted in the decreased performance than the
 baseline.
    3. Later, the model is tested on a 50 episode test-bed and the wins an
d loses are observed.
    4. Finally, the model will result the maximum possible score over time
of '0.450725' that it has gained during the Q-learning and plays the game
 intelligently on it's own.
```

Author

```
Bhagyashri Rangnath Gundal (NUID: 001081806)
Master of Science in Information Systems
Northeastern University, Boston, MA
```

Citation

References:

[1] ADL (3 September 2018), "*An introduction to Q-Learning: reinforcement learning*" retrieved from https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/ (https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/)

[2] Adesh Gautam (9 July 2018),"*Introduction to Reinforcement Learning (Coding Q-Learning) — Part 3*" retrieved from https://medium.com/swlh/introduction-to-reinforcement-learning-coding-q-learning-part-3-9778366a41c0 (https://medium.com/swlh/introduction-to-reinforcement-learning-coding-q-learning-part-3-9778366a41c0)

[3] Richard S. Sutton and Andrew G. Barto (2018), "*Reinforcement Learning - An Introduction*".

Licensing