CS 312 : Artificial Intelligence Laboratory   Date : April 27, 2020
Task 8 : Reinforcement Learning

Lokesh Kumar Nirania
170010009

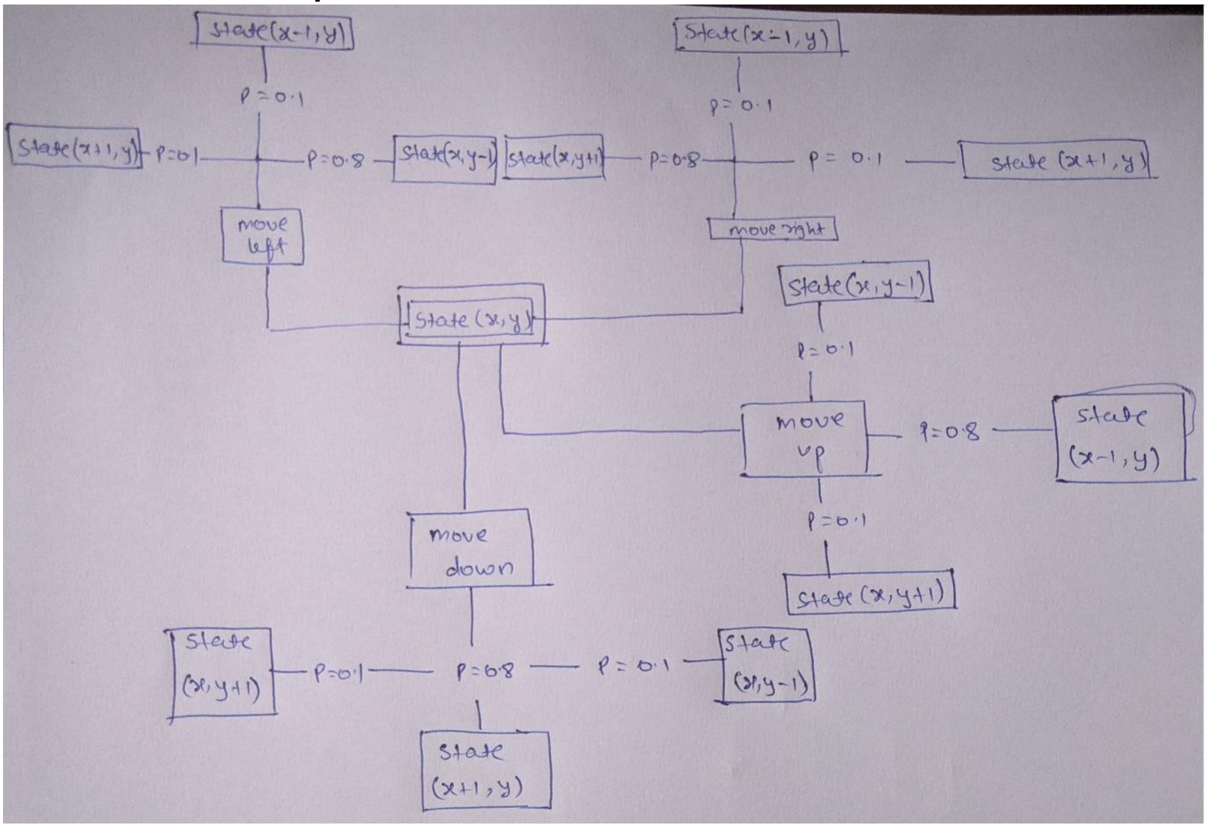
MDP Description :-

H : The grid has 4x4 i.e. 16 distinct states.
S : The 'Start State' is 1x0 cell, cell 3x1 is referred to as 'Bad State' and the cell 3x3 is referred to as 'Goal State'.
A : The agent can perform any of the four actions UP, DOWN, LEFT and RIGHT.
P : The Grid is a stochastic environment. Probability to move in any direction is 1/4. but Stochastic Prob are
Gird environment with following stochastic property:

Agent Action	Possible Actions	Probability
UP	UP, RIGHT, LEFT	0.8, 0.1, 0.1
DOWN	DOWN, RIGHT, LEFT	0.8, 0.1, 0.1
LEFT	LEFT, UP, DOWN	0.8, 0.1, 0.1
RIGHT	RIGHT, UP, DOWN	0.8, 0.1, 0.1

R : The agent receives -1 reward for every state other than the Bad State and Goal State. The agent receives -70 in the Bad State whereas it receives 100 in the Goal State.

State Transition Graph :-



Optimal Policy :-

We can use breadth-first search to find shortest path avoiding the bad state to verify the optimality

Experimental Results :-

If we follow stochastic rules (intention of moving up doesn't always result in moving up)

Gamma = 0.5

```
Value Iteration
episodes  8
[['↑' '→' '↓' '↓']
 ['←' '↑' '↓' '↓']
 ['←' '→' '→' '↓']
 ['→' '→' '→' '↑']]
[[0 1.62 6.3 14.45]
 [0 0.46 16.23 36.02]
 [0 15.87 38.52 86.03]
 [11.05 28.74 86.03 0]]

['←', [1, 0], -1]
['←', [1, 0], -1]
['←', [2, 0], -1]
['←', [2, 0], -1]
['←', [2, 0], -1]
['←', [2, 0], -1]
['←', [2, 0], -1]
['←', [2, 0], -1]
['→', [3, 0], -1]
['→', [3, 1], -1]
['→', [3, 2], -1]
['→', [2, 2], -1]
['↓', [2, 3], 100]
['G', [3, 3], 0]
---------------------------------
```

```
Policy Iteration
episodes 30
[['→' '→' '→' '↓']
 ['↑' '↑' '↓' '↓']
 ['←' '→' '→' '↓']
 ['↓' '→' '→' 'G']]
[[0.34 2.71 8.06 14.51]
 [-0.91 -0.6 16.03 44.61]
 [-1.96 5.14 36.16 98.31]
 [-0.88 11.75 93.75 0]]

['↑', [1, 0], -1]
['→', [0, 0], -1]
['→', [0, 1], -1]
['→', [0, 2], -1]
['↓', [1, 2], -1]
['→', [2, 2], -1]
['↓', [2, 3], 100]
['G', [3, 3], 0]
--------------------------
```

Gamma = 0.9

```
Value Iteration
episodes  13
[['→' '→' '↓' '↓']
 ['→' '→' '↓' '↓']
 ['↓' '→' '→' '↓']
 ['→' '→' '→' '↑']]
[[45.19 52.69 61.19 69.69]
 [46.48 54.35 71.07 81.81]
 [47.56 70.39 83.08 95.91]
 [62.28 74.16 95.91 0]]

['→', [1, 0], -1]
['→', [1, 1], -70]
['→', [2, 1], -1]
['→', [2, 2], -1]
['↓', [2, 3], 100]
['G', [3, 3], 0]
--------------------------------
```

```
Policy Iteration
episodes 30
[['↓' '→' '→' '↓']
 ['→' '↑' '↓' '↓']
 ['←' '→' '→' '↓']
 ['↓' '→' '→' 'G']]
[[3.1 11.72 14.34 34.14]
 [9.46 10.86 27.74 70.69]
 [-2.92 20.79 58.22 98.16]
 [-1.43 62.87 97.58 0]]

['→', [1, 0], -1]
['↑', [1, 1], -1]
['→', [0, 1], -1]
['→', [0, 2], -1]
['↓', [0, 3], -1]
['↓', [1, 3], -1]
['↓', [2, 3], 100]
['G', [3, 3], 0]
------------------------------
```

Purely Deterministic

Gamma = 0.5

```
Value Iteration
episodes  4
[['↓' '↓' '↓' '↓']
 ['↓' '→' '↓' '↓']
 ['↓' '↓' '↓' '↓']
 ['→' '→' '→' '↑']]
[[1.19 4.38 10.75 23.5]
 [4.38 10.75 23.5 49.0]
 [10.75 23.5 49.0 100.0]
 [23.5 49.0 100.0 0]]

['↓', [1, 0], -1]
['↓', [2, 0], -1]
['→', [3, 0], -1]
['→', [3, 1], -1]
['→', [3, 2], 100]
['G', [3, 3], 0]
-------------------------------

Policy Iteration
episodes 30
[['→' '→' '↓' '←']
 ['→' '→' '→' '↓']
 ['↑' '↑' '↑' '↓']
 ['↓' '→' '→' 'G']]
[[-1.36 0.17 7.84 0.25]
 [4.37 10.75 23.5 49.0]
 [-0.76 -0.5 4.23 100.0]
 [-1.16 11.62 75.0 0]]

['→', [1, 0], -1]
['→', [1, 1], -1]
['→', [1, 2], -1]
['↓', [1, 3], -1]
['↓', [2, 3], 100]
['G', [3, 3], 0]
-------------------------------
```

Gamma = 0.9

```
Value Iteration
episodes  4
[['↓' '↓' '↓' '↓']
 ['↓' '→' '↓' '↓']
 ['↓' '↓' '↓' '↓']
 ['→' '→' '→' '↑']]
[[54.95 62.17 70.19 79.1]
 [62.17 70.19 79.1 89.0]
 [70.19 79.1 89.0 100.0]
 [79.1 89.0 100.0 0]]

['↓', [1, 0], -1]
['↓', [2, 0], -1]
['→', [3, 0], -1]
['→', [3, 1], -1]
['→', [3, 2], 100]
['G', [3, 3], 0]
-------------------------------

Policy Iteration
episodes 30
[['→' '←' '↓' '←']
 ['→' '→' '↓' '↓']
 ['↓' '↑' '→' '↓']
 ['→' '→' '→' 'G']]
[[-2.05 -1.79 7.38 -1.3]
 [62.15 70.18 79.1 32.88]
 [2.35 21.81 89.0 100.0]
 [23.58 63.72 98.44 0]]

['→', [1, 0], -1]
['→', [1, 1], -1]
['↓', [1, 2], -1]
['→', [2, 2], -1]
['↓', [2, 3], 100]
['G', [3, 3], 0]
-------------------------------
```

Comparison of Policy and Value Iteration :-

1. **Policy iteration** includes: **policy evaluation** + **policy improvement**, and the two are repeated iteratively until policy converges.
2. **Value iteration** includes: **finding optimal value function** + one **policy extraction**. There is no repeat of the two because once the value function is optimal, then the policy out of it should also be optimal (i.e. converged).

3. **Finding optimal value function** can also be seen as a combination of policy improvement (due to max) and truncated policy evaluation (the reassignment of v_(s) after just one sweep of all states regardless of convergence).
4. The algorithms for **policy evaluation** and **finding optimal value function** are highly similar except for a max operation (as highlighted)
5. Similarly, the key step to **policy improvement** and **policy extraction** are identical except the former involves a stability check.

Conclusion

In my experience, *policy iteration* is faster than *value iteration*, as a policy converges more quickly than a value function.