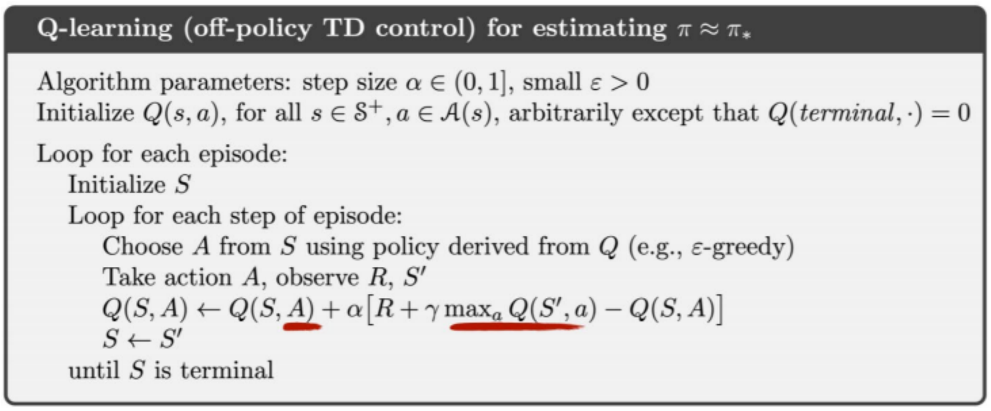
Reinforcement Learning Assignment Report

G2202226E

Part1. Q-Learning & implementation



```
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize $S$
    Loop for each step of episode:
        Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
        Take action $A$, observe $R$, $S'$
        $Q(S,A) \leftarrow Q(S,\underline{A}) + \alpha[R + \gamma \underline{\max_a Q(S',a)} - Q(S,A)]$
        $S \leftarrow S'$
    until $S$ is terminal
```

Fig.1. The steps of the q-learning algorithm(from lecture slide)

I chose the Q-Learning algorithm and implemented it on the basics of the *q-learning.py* file. Differently from other algorithms, Q-learning just needs a one-step trajectory. Also, for q-learning, the estimations are all state-action pairs. A *q-value* can be seen as the quality of an action taken from its state.

Given two paths in a game, we always want to choose the path with the maximum scores to win the game. As the equation from the figure above, after choosing and taking action A, there would be a reward. But also as an off-policy algorithm, the actual action of next_state is unnecessary, it always selects the action with the highest Q-value, the policy here just determines the visited and updated state and its corresponding action and does nothing else. In the same game case I mentioned, to win the game and also enlarge the reward, we have to choose the next action with maximum reward under the state next_state, then the Bellman equation comes:

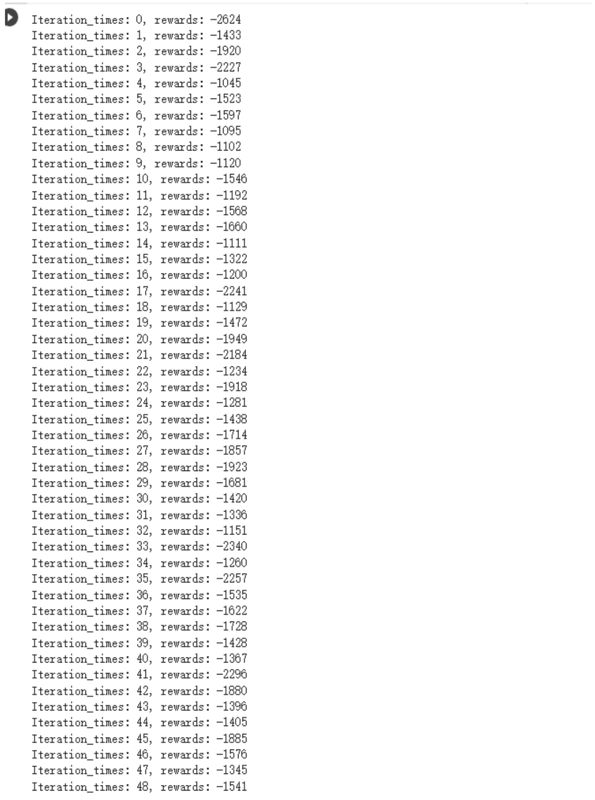
$$R + \gamma \underline{\max_a Q(S', a)}$$

It introduces gamma, which is a discount factor of the reward. For the state, we have now, the maximum reward should be the latest maximum reward plus the maximum reward of the future state, which is next_state. the Learning rate controls the updating from the current Q-value to the next Q-value. All Q-values are stored in the Q-table with row-states and column-actions style.

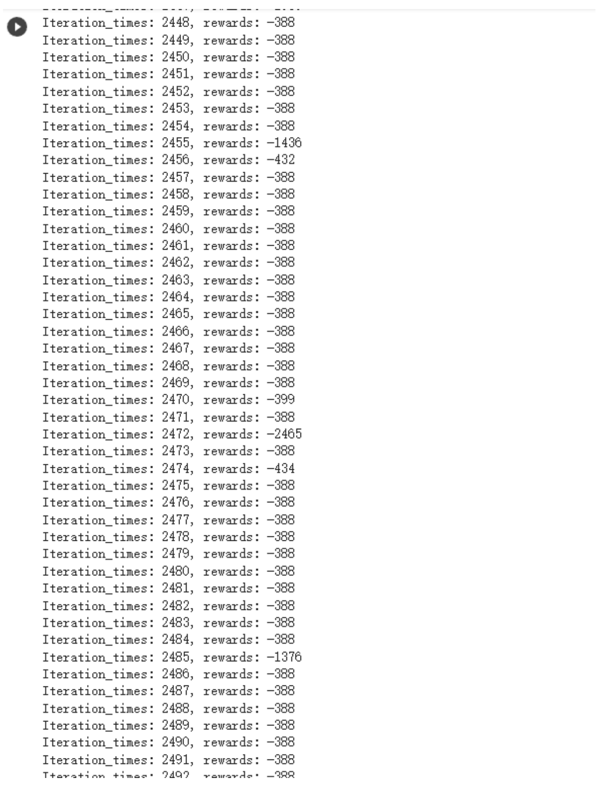
The steps of Q-learning are shown above. After iterating, the Q-values converge, and that is what model we want. For this assignment, all rewards are given as a negative value, so the final value should be closer to 0.

For the implementation of this part, i first got the target Q-values for the next state, this was done by getting all possible actions. And then used the updating q-value function to get the final Q. In this updating formula, the Bellman was included by using *np.max()* function. And the others were implemented by the given parameters.

Part.2 Learning Progress



```
Iteration_times: 0, rewards: -2624
Iteration_times: 1, rewards: -1433
Iteration_times: 2, rewards: -1920
Iteration_times: 3, rewards: -2227
Iteration_times: 4, rewards: -1045
Iteration_times: 5, rewards: -1523
Iteration_times: 6, rewards: -1597
Iteration_times: 7, rewards: -1095
Iteration_times: 8, rewards: -1102
Iteration_times: 9, rewards: -1120
Iteration_times: 10, rewards: -1546
Iteration_times: 11, rewards: -1192
Iteration_times: 12, rewards: -1568
Iteration_times: 13, rewards: -1660
Iteration_times: 14, rewards: -1111
Iteration_times: 15, rewards: -1322
Iteration_times: 16, rewards: -1200
Iteration_times: 17, rewards: -2241
Iteration_times: 18, rewards: -1129
Iteration_times: 19, rewards: -1472
Iteration_times: 20, rewards: -1949
Iteration_times: 21, rewards: -2184
Iteration_times: 22, rewards: -1234
Iteration_times: 23, rewards: -1918
Iteration_times: 24, rewards: -1281
Iteration_times: 25, rewards: -1438
Iteration_times: 26, rewards: -1714
Iteration_times: 27, rewards: -1857
Iteration_times: 28, rewards: -1923
Iteration_times: 29, rewards: -1681
Iteration_times: 30, rewards: -1420
Iteration_times: 31, rewards: -1336
Iteration_times: 32, rewards: -1151
Iteration_times: 33, rewards: -2340
Iteration_times: 34, rewards: -1260
Iteration_times: 35, rewards: -2257
Iteration_times: 36, rewards: -1535
Iteration_times: 37, rewards: -1622
Iteration_times: 38, rewards: -1728
Iteration_times: 39, rewards: -1428
Iteration_times: 40, rewards: -1367
Iteration_times: 41, rewards: -2296
Iteration_times: 42, rewards: -1880
Iteration_times: 43, rewards: -1396
Iteration_times: 44, rewards: -1405
Iteration_times: 45, rewards: -1885
Iteration_times: 46, rewards: -1576
Iteration_times: 47, rewards: -1345
Iteration_times: 48, rewards: -1541
```

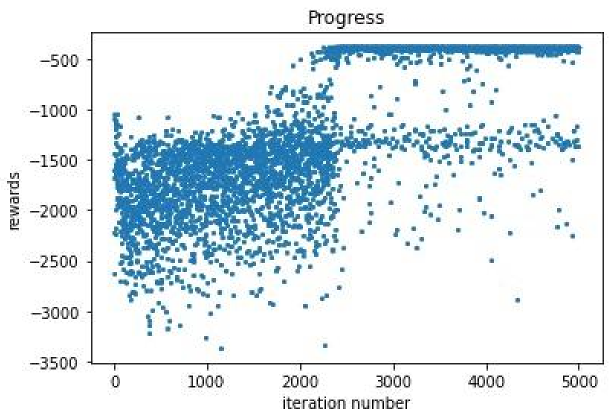


```
Iteration_times: 2448, rewards: -388
Iteration_times: 2449, rewards: -388
Iteration_times: 2450, rewards: -388
Iteration_times: 2451, rewards: -388
Iteration_times: 2452, rewards: -388
Iteration_times: 2453, rewards: -388
Iteration_times: 2454, rewards: -388
Iteration_times: 2455, rewards: -1436
Iteration_times: 2456, rewards: -432
Iteration_times: 2457, rewards: -388
Iteration_times: 2458, rewards: -388
Iteration_times: 2459, rewards: -388
Iteration_times: 2460, rewards: -388
Iteration_times: 2461, rewards: -388
Iteration_times: 2462, rewards: -388
Iteration_times: 2463, rewards: -388
Iteration_times: 2464, rewards: -388
Iteration_times: 2465, rewards: -388
Iteration_times: 2466, rewards: -388
Iteration_times: 2467, rewards: -388
Iteration_times: 2468, rewards: -388
Iteration_times: 2469, rewards: -388
Iteration_times: 2470, rewards: -399
Iteration_times: 2471, rewards: -388
Iteration_times: 2472, rewards: -2465
Iteration_times: 2473, rewards: -388
Iteration_times: 2474, rewards: -434
Iteration_times: 2475, rewards: -388
Iteration_times: 2476, rewards: -388
Iteration_times: 2477, rewards: -388
Iteration_times: 2478, rewards: -388
Iteration_times: 2479, rewards: -388
Iteration_times: 2480, rewards: -388
Iteration_times: 2481, rewards: -388
Iteration_times: 2482, rewards: -388
Iteration_times: 2483, rewards: -388
Iteration_times: 2484, rewards: -388
Iteration_times: 2485, rewards: -1376
Iteration_times: 2486, rewards: -388
Iteration_times: 2487, rewards: -388
Iteration_times: 2488, rewards: -388
Iteration_times: 2489, rewards: -388
Iteration_times: 2490, rewards: -388
Iteration_times: 2491, rewards: -388
Iteration_times: 2492, rewards: -388
```

Fig.2 shows 2 parts of the results after each iteration from learning.

The first image in Fig.2 shows the calculation of the starting part, and the second image shows the almost final part when the iteration number was set to 2500. From the results, we can see that the results converge to the value of -388, and on the behalf of the learning progress, we can predict when the iteration number becomes larger, the converge will perform better. So I adjust the iteration number to 5000 and plot the scatter figure of all rewards:

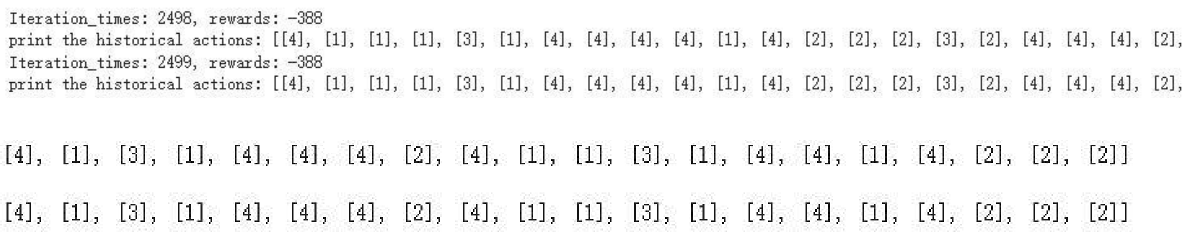
```
Iteration_times: 4999, rewards: -388
```



As it shows, although there are still some values not nearby or at -388 after learning, about 90% are in the right position. The remaining 10% are obviously to see cause they are dotted and some of them are still having a bad reward. And the good values all gather together and then give a whole area, which makes up of many small dots with only size 5(usually 20), so even it looks small but contains about 90% of the results.

Part3. Results & Final Value

As shown in Fig.2, the final sum of rewards is -388, by printing all previous actions, we can get the final actions are:



```
Iteration_times: 2498, rewards: -388
print the historical actions: [[4], [1], [1], [1], [3], [1], [4], [4], [4], [4], [1], [4], [2], [2], [2], [3], [2], [4], [4], [4], [2],
Iteration_times: 2499, rewards: -388
print the historical actions: [[4], [1], [1], [1], [3], [1], [4], [4], [4], [4], [1], [4], [2], [2], [2], [3], [2], [4], [4], [4], [2],
```

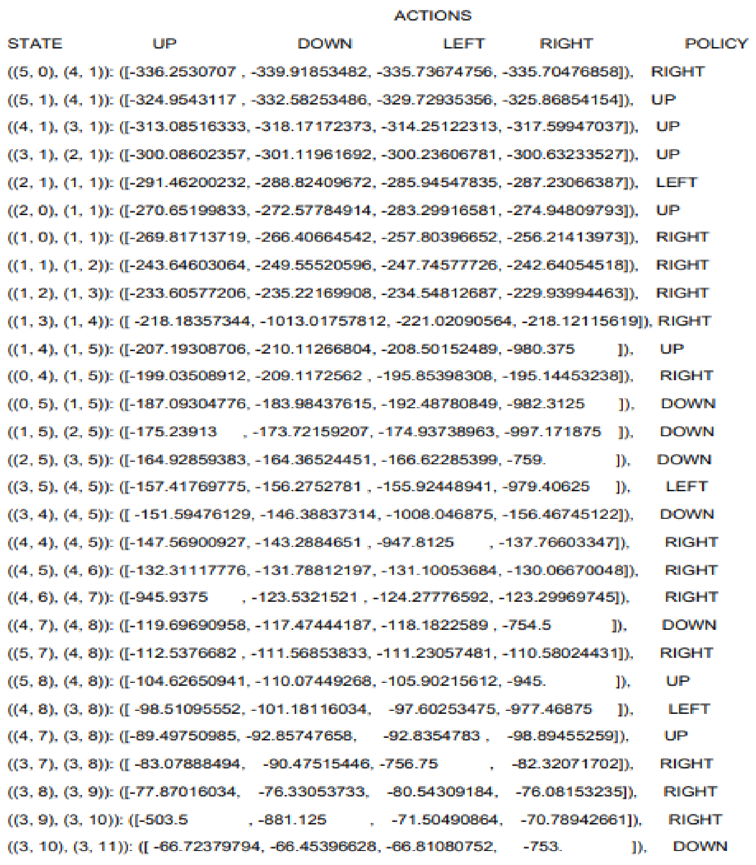
```
[4], [1], [3], [1], [4], [4], [4], [2], [4], [1], [1], [3], [1], [4], [4], [1], [4], [2], [2], [2]]

[4], [1], [3], [1], [4], [4], [4], [2], [4], [1], [1], [3], [1], [4], [4], [1], [4], [2], [2], [2]]
```

Fig 3. Two sets of actions with a final value of -388. ‘[1] [2] [3] [4]’ represent actions ‘[up, down, left, right]’.

From the above two sets of actions, we can know that after training, the actions become the same, and the final sum of rewards is -388.

And the final table with these actions:



```
                                            ACTIONS
STATE            UP                DOWN              LEFT         RIGHT            POLICY
((5, 0), (4, 1)): ([-336.2530707 , -339.91853482, -335.73674756, -335.70476858]),   RIGHT
((5, 1), (4, 1)): ([-324.9543117 , -332.58253486, -329.72935356, -325.86854154]),   UP
((4, 1), (3, 1)): ([-313.08516333, -318.17172373, -314.25122313, -317.59947037]),   UP
((3, 1), (2, 1)): ([-300.08602357, -301.11961692, -300.23606781, -300.63233527]),   UP
((2, 1), (1, 1)): ([-291.46200232, -288.82409672, -285.94547835, -287.23066387]),   LEFT
((2, 0), (1, 1)): ([-270.65199833, -272.57784914, -283.29916581, -274.94809793]),   UP
((1, 0), (1, 1)): ([-269.81713719, -266.40664542, -257.80396652, -256.21413973]),   RIGHT
((1, 1), (1, 2)): ([-243.64603064, -249.55520596, -247.74577726, -242.64054518]),   RIGHT
((1, 2), (1, 3)): ([-233.60577206, -235.22169908, -234.54812687, -229.93994463]),   RIGHT
((1, 3), (1, 4)): ([ -218.18357344, -1013.01757812, -221.02090564, -218.12115619]), RIGHT
((1, 4), (1, 5)): ([-207.19308706, -210.11266804, -208.50152489, -980.375       ]),   UP
((0, 4), (1, 5)): ([-199.03508912, -209.1172562 , -195.85398308, -195.14453238]),    RIGHT
((0, 5), (1, 5)): ([-187.09304776, -183.98437615, -192.48780849, -982.3125      ]),   DOWN
((1, 5), (2, 5)): ([-175.23913    , -173.72159207, -174.93738963, -997.171875  ]),   DOWN
((2, 5), (3, 5)): ([-164.92859383, -164.36524451, -166.62285399, -759.          ]),   DOWN
((3, 5), (4, 5)): ([-157.41769775, -156.2752781 , -155.92448941, -979.40625    ]),    LEFT
((3, 4), (4, 5)): ([ -151.59476129, -146.38837314, -1008.046875, -156.46745122]),    DOWN
((4, 4), (4, 5)): ([-147.56900927, -143.2884651 , -947.8125      , -137.76603347]),    RIGHT
((4, 5), (4, 6)): ([-132.31117776, -131.78812197, -131.10053684, -130.06670048]),    RIGHT
((4, 6), (4, 7)): ([-945.9375      , -123.5321521 , -124.27776592, -123.29969745]),    RIGHT
((4, 7), (4, 8)): ([-119.69690958, -117.47444187, -118.1822589 , -754.5          ]),   DOWN
((5, 7), (4, 8)): ([-112.5376682 , -111.56853833, -111.23057481, -110.58024431]),    RIGHT
((5, 8), (4, 8)): ([-104.62650941, -110.07449268, -105.90215612, -945.          ]),    UP
((4, 8), (3, 8)): ([ -98.51095552, -101.18116034,   -97.60253475, -977.46875    ]),    LEFT
((4, 7), (3, 8)): ([-89.49750985, -92.85747658,     -92.8354783 ,   -98.89455259]),    UP
((3, 7), (3, 8)): ([ -83.07888494,   -90.47515446, -756.75          ,   -82.32071702]),   RIGHT
((3, 8), (3, 9)): ([-77.87016034,   -76.33053733,   -80.54309184,   -76.08153235]),    RIGHT
((3, 9), (3, 10)): ([-503.5           , -881.125        ,   -71.50490864,   -70.78942661]),   RIGHT
((3, 10), (3, 11)): ([ -66.72379794, -66.45396628, -66.81080752,     -753.           ]),    DOWN
```

```
((4, 10), (3, 11)): ([ -63.55477939,   -61.09665654, -503.5          , -61.0646124 ]),      RIGHT
((4, 11), (3, 11)): ([ -56.63092162,   -56.89554441,   -58.54457991, -754.5          ]),   UP
((3, 11), (2, 11)): ([ -51.14234507,   -51.25799982,   -51.46069209, -755.25         ]),   UP
((2, 11), (1, 11)): ([ -47.43891641,   -45.43394765,   -44.58822734, -504.           ]),   LEFT
((2, 10), (1, 11)): ([ -36.95780539,   -38.73659392, -504.5         ,   -44.03167707]),    UP
((1, 10), (1, 11)): ([ -31.63177101,   -32.18576736, -756.          ,   -30.26040949]),    RIGHT
((1, 11), (1, 12)): ([-24.65135077, -24.94316352,   -27.88076861,   -24.50546413]),       RIGHT
((1, 12), (1, 13)): ([ -19.70248902, -503.          ,   -20.93182649,   -23.28033748]),    UP
((0, 12), (1, 13)): ([-17.27661857, -20.20931354, -17.12807567,      -13.840898   ]),     RIGHT
((0, 13), (1, 13)): ([-12.86198592,   -8.9302      , -17.11679167,    -9.92524969]),       DOWN
((1, 13), (2, 13)): ([-7.9313125,       -4.98       , -8.548825 ,          -7.450125 ]),    DOWN
((2, 13), (3, 13)): ([   -3.      ,          -2.     ,       -502.      ,            -2.9925]),   DOWN
((3, 13), (4, 13)): ([0.              ,            0.,                   0.,                  0.])
```

Table.1 The V-table for actions with final reward, which equals -388.

Table 1 has the columns for ‘State’, ‘ACTIONS’, ‘POLICY’, and ‘ACTIONS’ contains all four actions: ‘UP’, ‘DOWN’, ‘LEFT’ and ‘RIGHT’. ‘STATE’ stands for the position of both Agent A and Box B. The values under four actions are their q-values for corresponding directions, and the policy indicates under this state, the best next action, and it is the least value among the four values. The full value of *self.Q* table with all states in every 1x1 grid can be checked in Appendix.

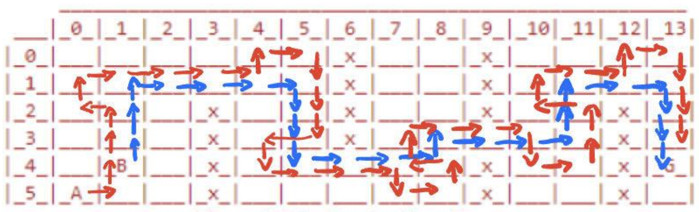


Figure 1. The BoxPushing Game.

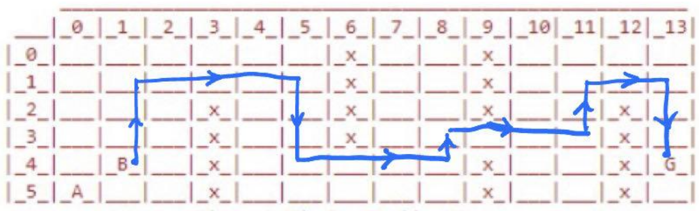


Figure 1. The BoxPushing Game.

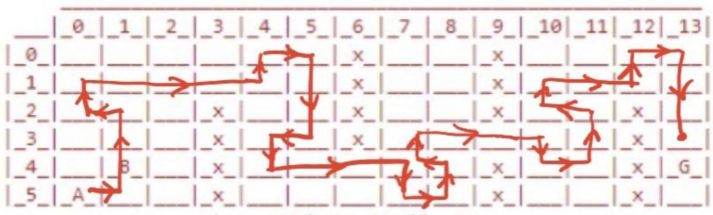


Figure 1. The BoxPushing Game.

Fig.4 Final trajectories

Fig. 4 shows the final results of the trajectories for Agent A and Box B, the first image shows for each state, the action for A and B with arrows for every small grid until the goal, with red line for Agent A and a blue line Box B, which regards to actions in *Fig.3*. The second image shows the full movement for Box B, and the final state is the goal. The third image shows the full movement of Agent A, the final state is one state before the goal after pushing Box B to the goal.

Conclusion

After implementation, as all results show, I got a good outcome. The final sum of rewards converged and got the value of -388. Also, the female route of the game was coming as an output, and all the states, actions and their corresponding policy matched.

From this assignment, I got a deeper understanding of Reinforce learning and q-learning. And the relationship between environment, agent, state, action, reward(sometimes penalty) and policy: the agent should expose to the environment and the situations that the agent is facing can be seen as states. Also, the agent performs an action to transition from one state to another state, a reward/penalty after the transition will be received by the agent. At last, the policy is the strategy of choosing an action given a state of best value.

Appendix:

colab:
https://colab.research.google.com/drive/1Qv53GNG3yV_mWe-hMNh3F5tVNG5umOjN?usp=sharing

The data can be checked in the 1.txt file(too big, over 40 pages).