Maze-Q learning

過程

步驟一: 調整 Reward

```
if A == "right":
        if S == GOAL:
                S_ = "terminal"
                R = 100
        elif S % N_STATES_x == N_STATES_x - 1 or (S + 1) in walls:
                S_ = S
                R = -10
        else:
                S_ = S + 1
                R = -1
elif A == "left":
        if S % N_STATES_x == 0 or (S - 1) in walls:
                S_ = S
                R = -10
        else:
                S_ = S - 1
                R = -1
elif A == "up":
        if S < N_STATES_x or (S - N_STATES_x) in walls:
                S_ = S
                R = -10
        else:
                S_ = S - N_STATES_x
                R = -1
elif A == "down":
        if S >= (N_STATES_y - 1) * N_STATES_x or (S + N_STATES_x) in walls:
                S_ = S
                R = -10
        else:
                S_ = S + N_STATES_x
                R = -1
```

首先我將 Reward 改變為如果碰到障礙物或超出範圍(即違反規則)的話-10 分，

往上下左右移動-1 分(防止逗留)，達到終點 100

步驟二: 採用 Double DQN 架構

```
q_table = build_q_table(N_STATES_x, N_STATES_y, ACTIONS)
q_table2 = build_q_table(N_STATES_x, N_STATES_y, ACTIONS)
```

```
if S_ != "terminal":
        q_target = G + (GAMMA ** N_STEP) * q_table2.iloc[S_, :].max()
else:
        q_target = G
        is_terminated = True
```

```
q_table2 = q_table.copy()
```

我建立了兩個 Q-tables，一個用於選擇動作，另一個用於估計動作的價值。在每個時間步，使用第二個 Q-table 來估計下一個狀態的最大價值，並更新第一個 Q-table。在每個 episode 結束時，將第一個 Q-table 的值複製到第二個 Q-table 中。

步驟三:調整 EPSILON ALPHA GAMMA

Qtable 開頭、結尾

```
Q-table:

(<built-in function all>,           left       right         up       down
0   -18.929230   -9.950055 -18.915841  -9.943879
1    -9.949679   -9.949947 -18.758792 -18.775660
2    -9.950004   -9.949925 -18.745449 -18.772260
3    -9.949940  -18.814216 -18.695478  -9.949923
4     0.000000    0.000000   0.000000   0.000000
5     0.000000    0.000000   0.000000   0.000000
6   -16.536799  -17.835812 -16.685078  -9.816857
7     0.000000    0.000000   0.000000   0.000000
8   -17.302724  -16.706303 -17.759781  -9.815346
9     0.000000    0.000000   0.000000   0.000000
10   -8.550174   -7.606266 -10.227331  -9.236128
11   -7.782307    0.184495 -14.270478  -7.796950
12   -4.120636    3.373431  -8.398714  -1.559280
13    1.217895    4.859368  -6.937371  -1.055781
14    2.322784    6.510410  -5.078180  -4.721531
15    3.553446    8.344900  -3.965859  -0.251591
```

```
214    0.000000    0.000000    0.000000    0.000000
215  -17.670538   -7.798131   -9.133698  -18.153484
216   -8.515946   -7.543319  -17.507296  -17.422407
217   -8.720585   -7.261805  -17.065222  -17.473843
218   -8.491585   -6.951420   -9.159172  -16.890864
219   -8.156842   -6.608123   -8.381054  -16.868773
220   -7.577527   -6.227838   -7.475915  -16.474832
221   -7.739798   -8.044976   -5.806178  -15.849961
222   -8.871492   -9.025037   -5.743782  -16.856868
223   -8.993328   -8.991073   -8.991179  -17.016571
224   -8.968097   -8.966125   -8.967514  -17.191961
225   -8.956430  -15.554257   -8.950324  -16.803100
226    0.000000    0.000000    0.000000    0.000000
227   -1.000000   -0.296200   -1.000000   -1.000000
228   -0.288910   -1.000000   -0.297010   -1.000000
229    0.000000    0.000000    0.000000    0.000000
230   75.812873  100.000000   76.395456   67.983810)
```

最佳結果

1039 步 5 個寶箱

```
['Episode 24: total_steps=910']
Episode 23: SCORE=5
```

心得

在原本的 Reward 設定上，原本以為沒有甚麼不對勁的，但是之後思考在移動時沒有碰到障礙物會+3 在另一方面是否也代表走越多步 Reward 越多，果不其然在改為負數後真的大大縮減了總步數，不過步數最後維持在 400 左右，接著我建立另一個 q-table2 來估計 q-value，以避免讓隨時變動的 q-table1 估計 q-value，這又讓我的步數大大縮減到 100 左右，接著我嘗試調整探索率、學習率來改善結果，但也沒有特別顯著的作用，最終結果顯示最少步的回合得到的寶相最少，步數為 1000 或以上寶相數十分可觀。

Colab Link

https://colab.research.google.com/drive/1NKkoZoPHYFK6wWqYzqii9KFBFKd

oG5qq?usp=share_link