Assignment 3

61075029H Jun Yu SHEN 沈峻宇

1. MC-Epsilon greedy and MC-Exploring
    a. MC-Exploring Start
    Max iteration = 20000
    Gamma = 0.9
    Horizon = 3
    The results obtained using the above parameters are as follows

```
Before (random policy), T = Target, W = Wall
-------------------------------------------------
| T      | left   | right  | left   | left   | left   |
-------------------------------------------------
| right  | right  | down   | up     | W      | up     |
-------------------------------------------------
| down   | W      | up     | W      | down   | left   |
-------------------------------------------------
| right  | right  | up     | left   | right  | left   |
-------------------------------------------------

Optimal policy, T = Target, W = Wall
-------------------------------------------------
| T      | left   | left   | left   | left   | down   |
-------------------------------------------------
| up     | up     | left   | up     | W      | up     |
-------------------------------------------------
| up     | W      | down   | W      | right  | down   |
-------------------------------------------------
| up     | right  | up     | right  | right  | left   |
-------------------------------------------------
```

    b. MC-Epsilon greedy without Exploring Starts(On-policy)
    The only way to avoid the assumption that exploring starts is to ensure that all actions can be selected.

On-policy first-visit MC control (for ε-soft policies)

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \leftarrow$ arbitrary
  $Returns(s,a) \leftarrow$ empty list
  $\pi(a|s) \leftarrow$ an arbitrary ε-soft policy

Repeat forever:
  (a) Generate an episode using π
  (b) For each pair s,a appearing in the episode:
    $G \leftarrow$ return following the first occurrence of s,a
    Append G to $Returns(s,a)$
    $Q(s,a) \leftarrow \text{average}(Returns(s,a))$
  (c) For each s in the episode:
    $A^* \leftarrow \arg\max_a Q(s,a)$
    For all $a \in \mathcal{A}(s)$:

$$\pi(a|s) \leftarrow \begin{cases} 1-\varepsilon+\varepsilon/|\mathcal{A}(s)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq A^* \end{cases}$$

With probability ε, the current action with the largest action value estimate is selected, while with probability 1-ε, an action is randomly selected from all actions at random.
If there are multiple actions to choose from, you can use the following formula to calculate the probability and then select.

$$\pi(\alpha|s) \leftarrow \begin{cases} 1-\varepsilon+\dfrac{\varepsilon}{|\mathrm{A}(s)|} \\ \dfrac{\varepsilon}{|\mathrm{A}(s)|} \end{cases}$$

$$|\mathrm{A}(s)| = number\ of\ actions$$

The code is shown in the figure below and is selected according to the odds calculated by epsilon:

```
PolicyProbility = np.ones(len(valid_actions)) * self.epsilon / len(valid_actions)
PolicyProbility[np.argmax(Q_value)] += 1 - self.epsilon
# print("--------")
# print(valid_actions, PolicyProbility)
# print(valid_actions[np.random.choice(np.arange(len(valid_actions)), p = PolicyProbility)])
return valid_actions[np.random.choice(np.arange(len(valid_actions)), p = PolicyProbility)]
```

Max iteration = 20000
Gamma = 0.9
Horizon = 3
Epsilon = 0.2
The results obtained using the above parameters are as follows:

```
Before (random policy), T = Target, W = Wall
---------------------------------------------------
| T      | right  | right  | right  | left   | down   |
---------------------------------------------------
| right  | up     | up     | up     | W      | up     |
---------------------------------------------------
| up     | W      | up     | W      | right  | up     |
---------------------------------------------------
| right  | right  | right  | right  | up     | up     |
---------------------------------------------------

Optimal policy, T = Target, W = Wall
---------------------------------------------------
| T      | left   | left   | left   | left   | left   |
---------------------------------------------------
| up     | left   | up     | left   | W      | up     |
---------------------------------------------------
| up     | W      | up     | W      | right  | up     |
---------------------------------------------------
| up     | left   | up     | right  | left   | left   |
---------------------------------------------------
```

By comparing the two results, we can see that in the state of small "Horizon", **"without ExploringStarts"** is better because it can explore and exploration with epsilon.

2. Fly~~
   a. Add new action fly

```
self.grid_world = [[  "T",   "s1",   "s2",   "s3",   "s4",   "s5"],
                   [ "s6",   "s7",   "s8",   "s9",    "W",  "s10"],
                   ["s11",    "W",  "s12",    "W",  "s13",  "s14"],
                   ["s15",  "s16",  "s17",  "s18",  "s19",  "s20"]] #T: Target, W: Wall
self.action_to_number = {"up": 0, "right":1, "down":2, "left":3, "fly":4}    #行為
self.action_dict = {"up": [-1,0], "right": [0, 1], "down": [1,0], "left":[0,-1], "fly":[10,10]}
self.direction_dict = {0: "up", 1:"right", 2:"down", 3:"left", 4:"fly"}
self.invalid_start = ["T", "W"]
```

Movements for two special coordinates

```
def transfer_state(self, state_coordinates, action): #Input(state, action), Output(next state)

    current_state_coordinates = state_coordinates
    #Action for fly only on coordinate s13, s9
    if self.action_to_number[action] == 4:
        if (state_coordinates == [2,4]).all(): #s13 to s1
            return np.array([0,1])
        elif (state_coordinates == [1,3]).all():    #s9 to s6
            return np.array([1,0])
        else:
            return current_state_coordinates
    next_state_coordinates = state_coordinates + self.action_dict[action]
```

   b. Monte Carlo Algorithm addfly
   Same parameters as the first question.

```
Before (random policy), T = Target, W = Wall
-------------------------------------------------------------
| T       | left    | right   | left    | left    | left    |
-------------------------------------------------------------
| right   | right   | down    | up      | W       | up      |
-------------------------------------------------------------
| down    | W       | up      | W       | fly     | down    |
-------------------------------------------------------------
| right   | right   | up      | left    | right   | left    |
-------------------------------------------------------------

Optimal policy, T = Target, W = Wall
-------------------------------------------------------------
| T       | left    | left    | left    | left    | left    |
-------------------------------------------------------------
| up      | left    | left    | fly     | W       | up      |
-------------------------------------------------------------
| up      | W       | down    | W       | fly     | left    |
-------------------------------------------------------------
| up      | left    | right   | right   | up      | left    |
-------------------------------------------------------------
```

   c. Monte Carlo Algorithm addfly without Exploring Starts

Same parameters as the first question.

```
Before (random policy), T = Target, W = Wall
-------------------------------------------------------
| T       | right  | right  | right  | right  | down   |
-------------------------------------------------------
| up      | up     | up     | up     | W      | up     |
-------------------------------------------------------
| up      | W      | up     | W      | right  | up     |
-------------------------------------------------------
| up      | right  | up     | right  | up     | up     |
-------------------------------------------------------

Optimal policy, T = Target, W = Wall
-------------------------------------------------------
| T       | left   | left   | left   | left   | down   |
-------------------------------------------------------
| up      | up     | up     | fly    | W      | up     |
-------------------------------------------------------
| up      | W      | up     | W      | fly    | left   |
-------------------------------------------------------
| up      | left   | up     | right  | up     | left   |
-------------------------------------------------------
```