Assignment 3

1. Sarsa Algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Sarsa algorithm is for on-policy learning. In each step, the agent would produce state S' for next time according to A and S. And what's different is that Sarsa would also produce the corresponding action A' using greedy strategy. The greedy choose strategy is as follows:

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

In my implementation, total time of iteration is 1000, $\alpha = 0.5, \gamma = 1, \varepsilon = 0./0.1/0.5$. Each iteration starts from [4, 0] and ends when reaching [4,12]. Reward for each step would be -1 unless:

1. stepping into cliff area R=-100, state returns to [4, 0].
2. Reaching terminal point [4,12], R=100.

$\varepsilon = 0.5$, it choose the safe path.

```
Sarsa algorithm:
['→', '→', '→', '→', '→', '→', '→', '→', '→', '→', '→', '↓']
['↑', '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']
```

$\varepsilon = 0.1$, it choose the path between safe and optimal.

```
Sarsa algorithm:
['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']
```

$\varepsilon = 0$, it choose the optimal path.

```
Sarsa algorithm:
['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']
```

The smaller ε is, the possibility of choosing a none-optimal action is smaller, so sarsa is more likely to choose a optimal path. And with possibility of choosing none-optimal action, with fear of going into cliff area, it tends to choose a safer path.

2. Q-Learning

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$;
  until S is terminal

The difference between Q-learning and sarsa is the way of updating Q value. In Q-learning, greedy algorithm is used to choose an action and update Q-value while the real action to execute is still choosed by ε-greedy algorithm. In comparison, sarsa choose the same action in updating Q-value and execution, which determines by ε-greedy algorithm. Therefore, Q-learning is an off-policy algorithm.

With ε being 0, 0.1 and 0.5, the result remains the same. So Q-Learning would always choose the optimal path.

```
Q learning algorithm:
['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']
```

Because in Q-Learning, Q-value is always updated using the optimal choise. So ε could only determine the states in each episode, but has no effect on the policy.