Markov Decision Processes

- **Markov Decision Processes**

MDP contains some important elements such as environment, state, action, reward, policy and value. Given the state in the environment, the agent needs to select the action to maximize the long-term reward. So given the MDP problem, we can use dynamic algorithms to get the optimal based on a set of states/rewards.

- **Value Iteration**

In value iteration, we can start with arbitrary utilities and update the utilities based on the neighbors. The rewards propagate through the neighbors. By updating the estimated value function, we compute the optimal value function until convergence. We take maximum over all possible actions in the iterations.

- **Policy Iteration**

Policy maps states to actions. In policy iteration, we need to choose an arbitrary policy in the beginning, and we do not care about whether the initial policy is optimal or not. Following the policy will result in a utility which can help us figure out how to improve the policy by finding the actions which maximize the utilities. Then we can iteratively evaluate the policy and improve it until the policy is converged, meaning the policy will not improve anymore.

- **Q Learning**

Q Learning is a model-free reinforcement learning algorithm. It uses the data about the world including the reward and new state to evaluate the Bellman equation. Q learning is a off-policy learner so it learns the optimal policy value independently from the agent's actions. This method assigns the Q values to all states and then the agent will visit each state and re-assess Q values based on immediate and delayed rewards. So it is trying to find a balance between exploitation and exploration. In other words, Q learning can execute on immediate reward and also explores the delayed rewards.

MDP problem – Frozen Lake

- **Problem Description**

In the frozen lake MDP problem, an agent needs to start from a starting point S and tries to reach the goal point G. In this environment, if the agent walks on the frozen lake area (marked as F), it is safe. Otherwise, the agent might fall into the hole (marked as H) and end the game. The reward is 1 if the agent reaches the goal and 0 otherwise. The purpose is to maximize the expected rewards by finding a walkable path to the goal instead of falling into the freezing water.

- **Why Interesting?**

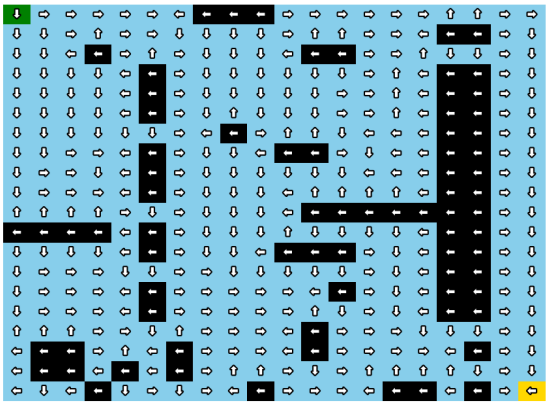
I think this MDP problem is interesting because it has both goal and hole in the given environment to affect the agent's decision. Frozen lake is a discrete and finite MDP so the convergence is guaranteed. In this problem, an agent will control a 20x20 grid world (See the grid in the right side). It allows the agent to choose 4 kinds of actions in each state: Up, Down, Left and Right. The movement direction is uncertain. The agent needs to observe the environment and leverage the rewards to take next action. So, it is a good example to use reinforcement learning to solve the problem with "large" number of states.

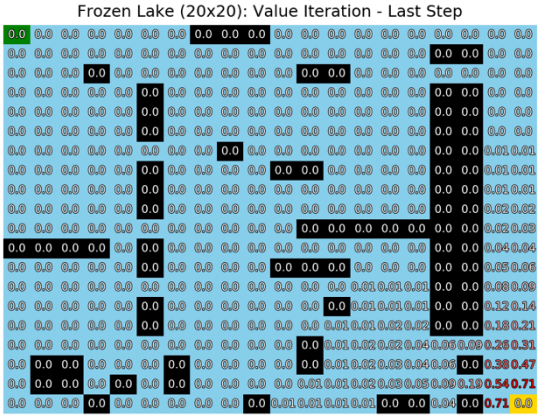
```
"SFFFFFFHHHFFFFFFFFF",
"FFFFFFFFFFFFFFFHHFF",
"FFFHFFFFFFFHHFFFFFF",
"FFFFFHFFFFFFFFFHHFF",
"FFFFFHFFFFFFFFFFHHF",
"FFFFFHFFFFFFFFFFHHFF",
"FFFFFFFFHFFFFFFFHHFF",
"FFFFFHFFFFHHFFFFHHFF",
"FFFFFHFFFFFFFFFFHHFF",
"FFFFFHFFFFFFFFFFHHFF",
"FFFFFFFFFFFHHHHHHHFF",
"HHHHFHFFFFFFFFFFHHFF",
"FFFFFHFFFFHHHFFFHHFF",
"FFFFFFFFFFFFFFFFHHFF",
"FFFFFHFFFFFFHFFFHHFF",
"FFFFFHFFFFFFFFFFHHFF",
"FFFFFFFFFFFHFFFFFFFF",
"FHHFFFHFFFFHFFFFFHFF",
"FHHFHFHFFFFFFFFFFFFF",
"FFFHFFFFFHFFFFHHFHFG"
```

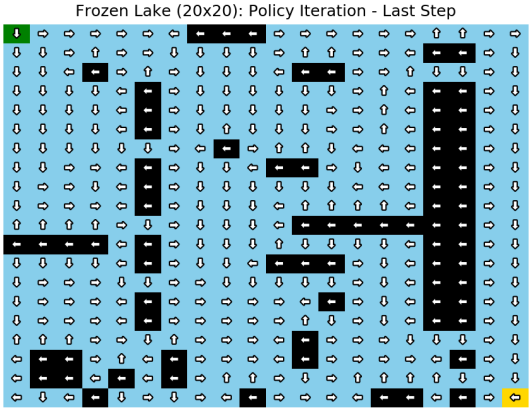
- **Value Iteration & Policy Iteration**

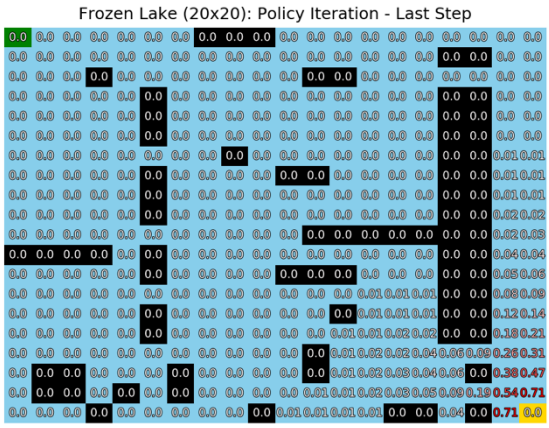
Based on the graphs below, we can see in the last step, both value iteration and policy iteration method show the similar optimal path from the starting point to the goal and compute the similar utilities in each state. So I can say they converge to the same answer. I think it's because both algorithms implicitly update the policy and state value in each iteration so they can find the optimal answer. They are guaranteed to converge to an optimal policy at the end.

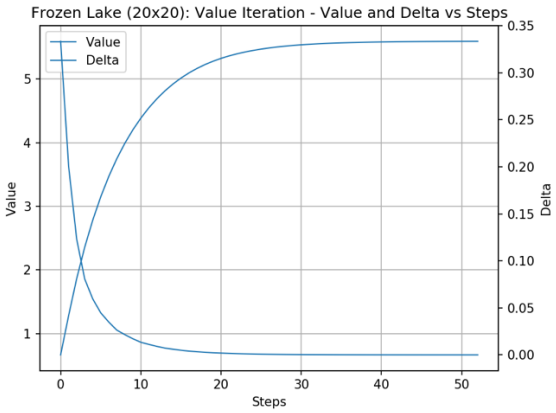
In the "value vs step" graphs, I can see the value curve goes up and then slow down. When the value does not increase anymore, it means the values converge. Value iteration takes around 40-50 steps to converge, and policy iteration takes 3 steps to converge. So, policy iteration takes few iterations and run fast. Since the value iteration process need to run through all possible actions at once to find the maximum action value and it uses non-linear equation, value iteration is computationally expensive than policy iteration and need more time to converge. Even though policy iteration needs 2 phases including policy evaluation and policy improvement, it remove "maximum" function in the algorithm so it uses linear algebra to do the computation, which involves less computation work.

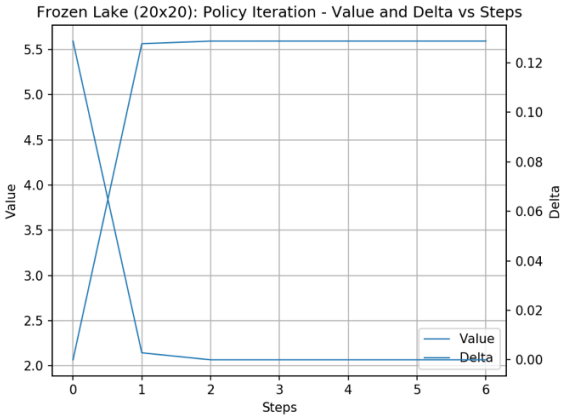






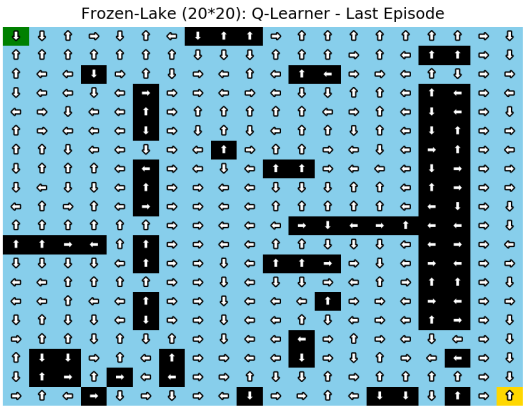


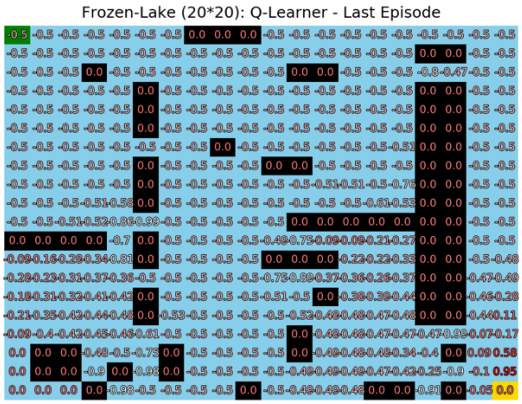


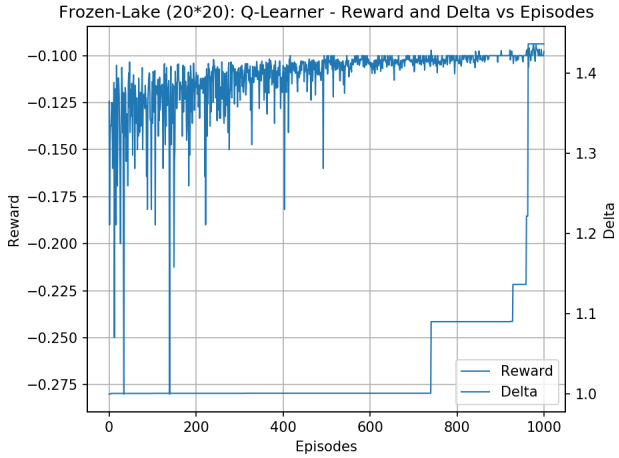


- **Q Learning**

Based on the graphs below, it is interesting to see Q learning function can find a path to the goal but does not converge very well. The Q values in the graphs are estimates of the sum of future rewards. It can give us the idea about how much reward we can get until the game ends. In the last graph, I can see the reward fluctuate a lot. Even after 800 iterations, the reward values become a little stable but still does not converge very well. I think it is because all Q values are initialized as 0 in the beginning and any taken actions might lead to a negative Q values. Then the decision-making system will explore a new state. Therefore, the long-time exploration process will consider many other actions (instead of experienced actions) in many states and make it hard to converge.



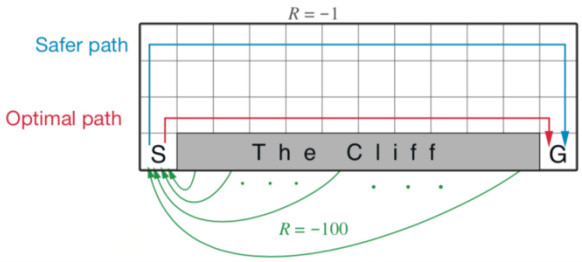




MDP Problem – Cliff Walking

- **Problem Description**

The agent starts from the starting point S and needs to go to the goal – Point G. The grey area is the cliff and the agent needs to avoid it. Stepping into the cliff region incur the negative reward of -100 and instantly send the agent back to the starting point S. The reward in other transition region is -1. Reaching the goal gives us 0 reward. Moving in 4 directions are allowed: Up, Down, Left and Right.

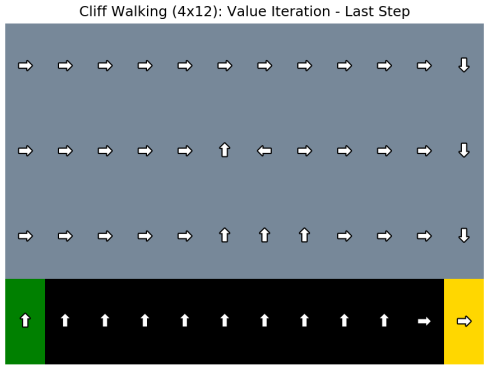


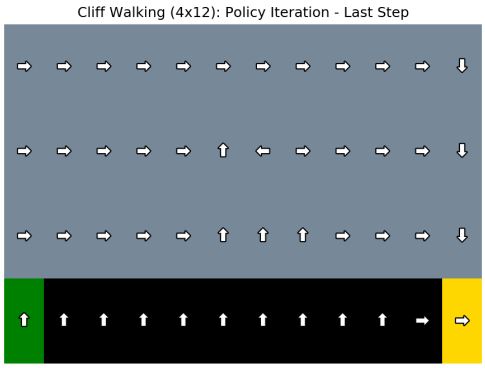
- **Why Interesting?**

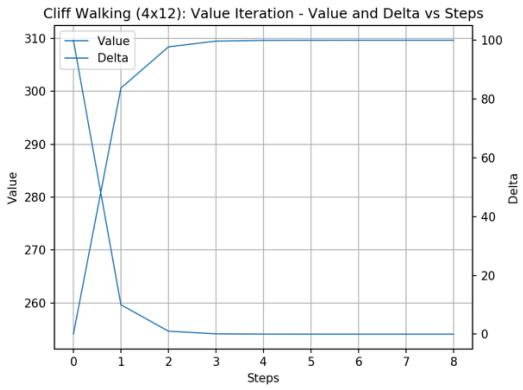
I think this MDP problem is interesting because the rewards in this case are all negative. So in order to maximize the sum of the future rewards (rewards should be closest to 0), the agent needs to be in a rush, meaning we need to find a way to reach out the goal as soon as possible. In this problem, I use 4x12 grid to make it have “small” number of states.

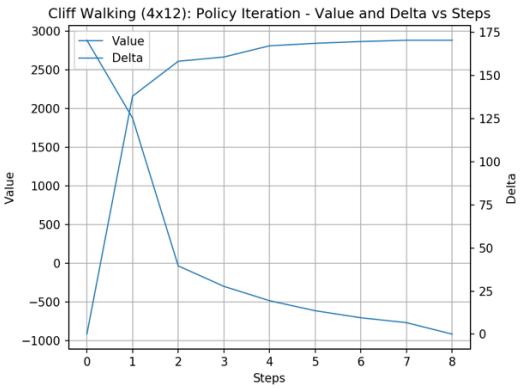
- **Value Iteration & Policy Iteration**

In the graphs below, I can see that these 2 methods result in same answers. In value iteration, it converges in step 3. In policy iteration, it converges in step 5. So these 2 methods almost converges at the same time. I think it’s because this cliff walking problem involves less states (4x12=48 states). So both value iteration and policy iteration can do a good job of finding the optimal way and also converge quickly.



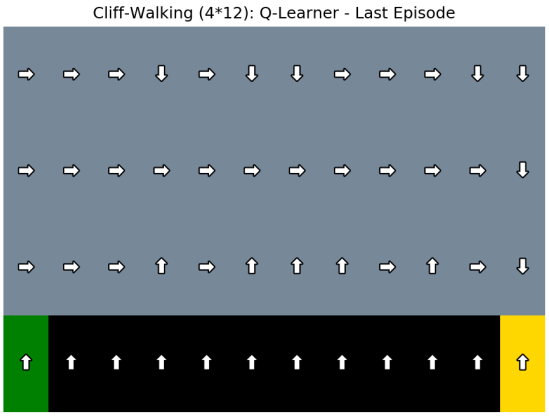


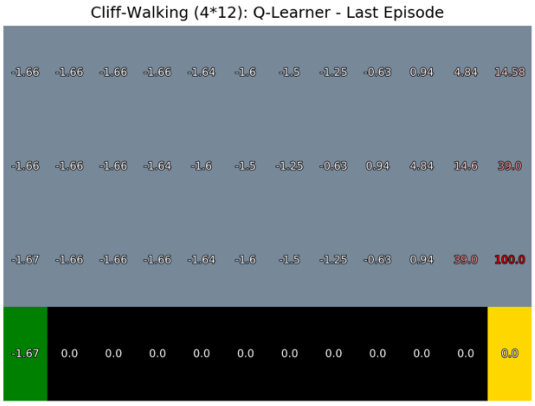




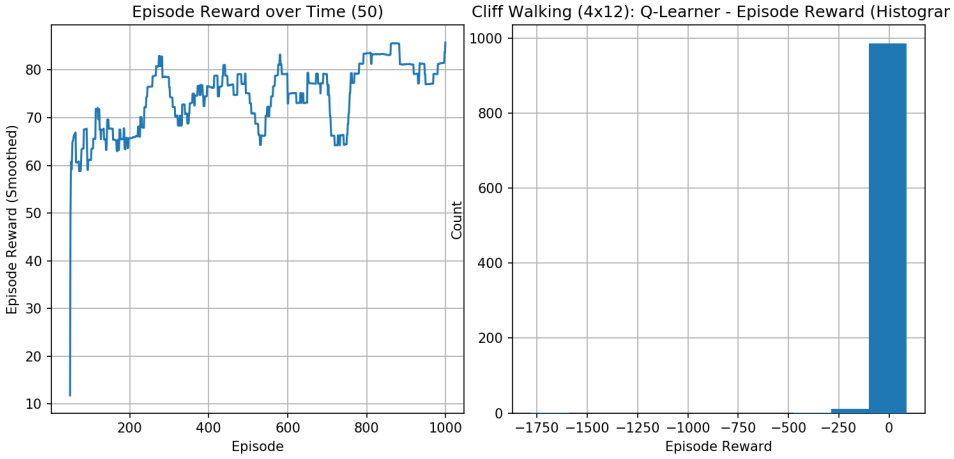
- **Q Learning**

Based on the last episode graph using Q learning algorithm, I can see the Q learning algorithm with exploration rate of 0.5 does find a path to reach out the goal. But if we think carefully, the states close to the edge are more easily to step into the cliff and then end the game. So some states on top of the cliff suggest the agent to go up instead of going right although going right can make it quickly reach the goal.





We initialize the process with random values. At each step, we gain information from the world. The information is used to update the values. In other words, we propagate the information about the action value one step at a time. But if we always take the “best” action we believe, we might act very greedily. To explore the world sufficiently, we need to pick the epsilon. If a random number is below epsilon, we can choose a random action. Otherwise, we can choose the action with highest value. By looking at the graphs below, I can see the reward values fluctuate a lot with the increase of episode. I think it’s because Q learning algorithm does the exploration so sometimes the reward will drop compared to the reward in previous episode.



- **Conclusion**

In general, value iteration and policy iteration can guarantee convergence. Value iteration algorithm is simpler but more expensive to compute and need more iterations to converge. Policy iteration algorithm is more complex but is cheaper to compute and requires fewer iterations to converge. For Q learning, it usually takes longer time to converge and computationally expensive. But it does not need Transition and Reward value and only uses the data

we learn from the world. With Q learning approach, all possible state-action pairs can be explored with infinite iterations.