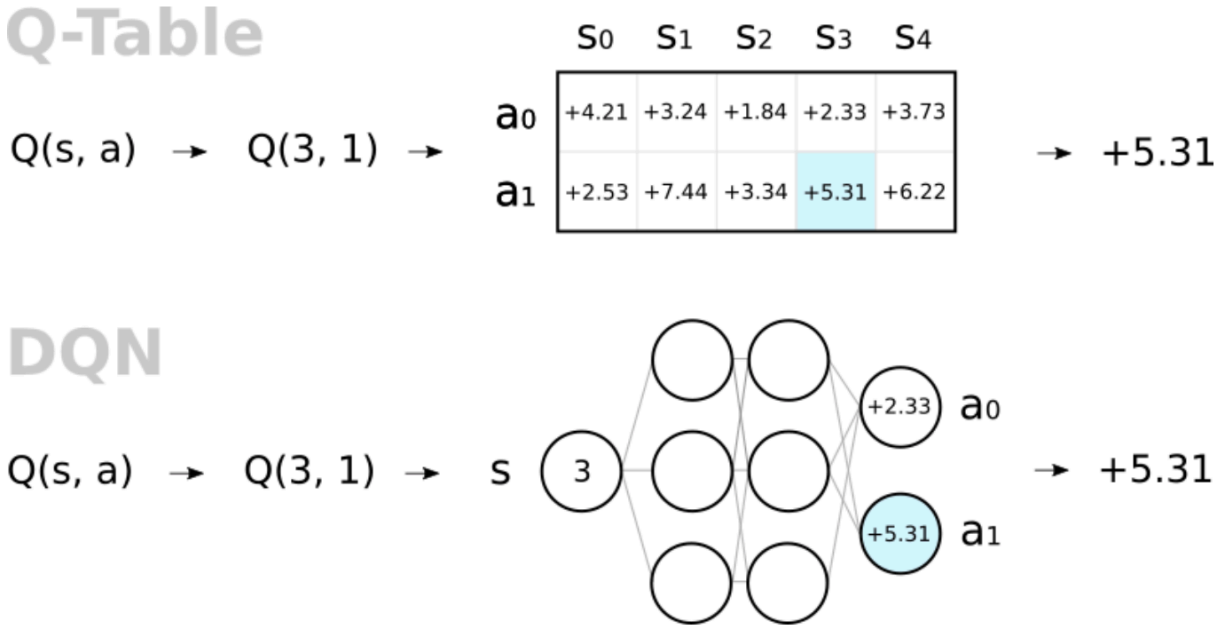
1. The advantage of MDP solver is that this solver takes less time for spaces or iterations. Moreover, it is difficult to know if we get the optimal policy or if more iteration would help by Q learning. However, we can be more confident about that the policy get by MDP would be most likely the optimal policy.
2. If we are not taking an "explore" action and only follow the best actions, it would speed up because if we are doing the best actions, we will get to gold states easily. However, the drawbacks are that if we initialize the Q-value for some states highly due to the randomness. The states only look bad when their Q-values for all directions are all bad. However, for a state, if action "up" and "left" are both good actions, but action "left" is a little better. In this case, if it initializes with "up" and get a good value, it seems less likely to correct it to action "left" and its Q-value would influence other states' Q-value. To get fast convergence, the initial values should be as close as possible to the final values; otherwise, it would take a longer time to correct it.
3. I would change the states int nodes in the tree. As shown in the picture below, for a state Q(3,1), it would be the node of s in third branch and the value of take action a1. Thus, in our case, a state would be at position [1,3] and take an action of "down", so this state would be at the branch [1,3] and at the action node "DOWN" whose value is 0 in my code.



(Picture is from https://towardsdatascience.com/reinforcement-learning-tutorial-part-3-basic-deep-q-learning-186164c3bf4)