Blackjack Homework

Problem1: Value Iteration

Following the operation from lecture6.1:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')]$$

We'll get:

$$\mathrm{U}_0 = 0 \quad 0 \quad 0 \quad 0 \quad 0$$
$$\mathrm{U}_1 = 0 \quad 15 \quad -5 \quad 26.5 \quad 0$$
$$\mathrm{U}_2 = 0 \quad 14 \quad 13.45 \quad 23 \quad 0$$

$$\pi_1 = NA \leftarrow \quad \leftarrow\rightarrow \quad \rightarrow NA$$
$$\pi_2 = NA \leftarrow \quad \rightarrow \quad \rightarrow NA$$

Problem2: Transforming MDPs

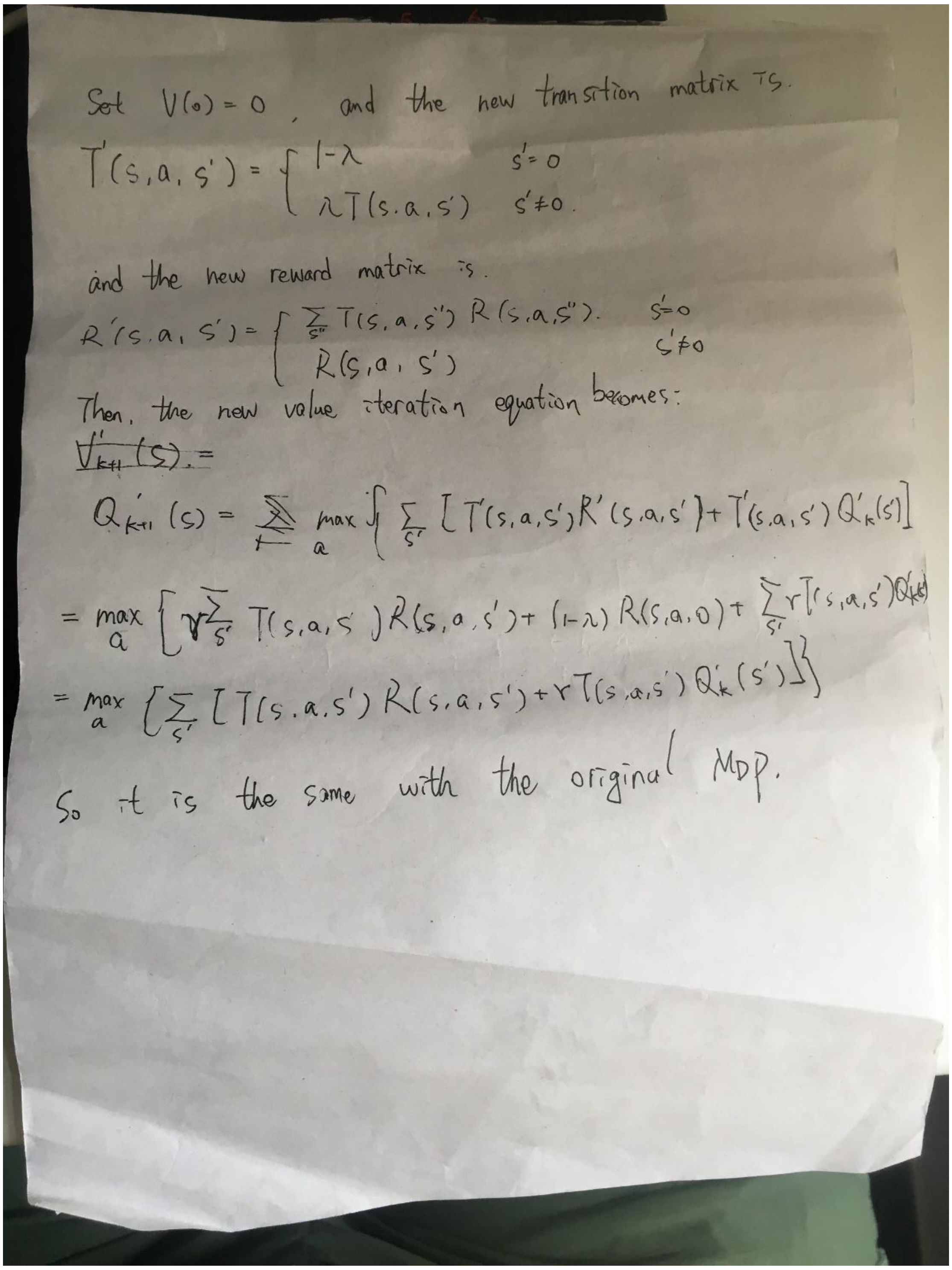
2b:

We can topological sort all the states, then determines the value of these nodes by this order.
Pseudocode: (Recursive)

```
def findV(state):
    if state.isEnd():
        return 0
    actions=getActions(state)
    V(state)= $\max_{actions(s)} \sum_{s'} T(s,a,s')[Reward(s,a,s') + \gamma$ findV(s')]
```

This Version of findV function do not save all values of all state, but we can easily do that with only a slight change.

2c: see the picture below



Set $V(0) = 0$, and the new transition matrix is.

$$T'(s,a,s') = \begin{cases} 1-\lambda & s'=0 \\ \lambda T(s,a,s') & s' \neq 0. \end{cases}$$

and the new reward matrix is.

$$R'(s,a,s') = \begin{cases} \sum_{s''} T(s,a,s'') R(s,a,s''). & s'=0 \\ R(s,a,s') & s' \neq 0 \end{cases}$$

Then, the new value iteration equation becomes:

~~$V'_{k+1}(s). =$~~

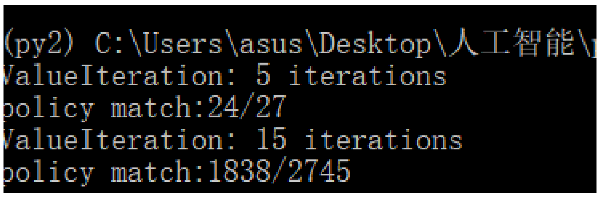
$$Q'_{k+1}(s) = \cancel{\sum_{t}} \max_a \left\{ \sum_{s'} \left[T'(s,a,s') R'(s,a,s') + T'(s,a,s') Q'_k(s') \right] \right.$$

$$= \max_a \left[\gamma \sum_{s'} T(s,a,s') R(s,a,s') + (1-\lambda) R(s,a,0) + \sum_{s'} \gamma T(s,a,s') Q'_k(s') \right.$$

$$= \max_a \left\{ \sum_{s'} \left[T(s,a,s') R(s,a,s') + \gamma T(s,a,s') Q'_k(s') \right] \right\}$$

So it is the same with the original MDP.

4b:

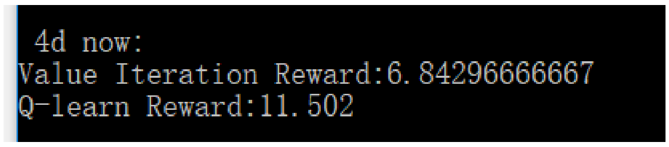


```
(py2) C:\Users\asus\Desktop\人工智能\
ValueIteration: 5 iterations
policy match:24/27
ValueIteration: 15 iterations
policy match:1838/2745
```

My code of problem 4b is written in submission.py. We can find that for small MDP, in the total 27 states, there are 24 match states, and for large MDP, in the total 2745 states, there is only 1838 match, which is about 67%. The reason is written in the text book. The function approximation algorithm is not an exact algorithm, they just use weighted features to approximate the true Q-value. However, we cannot know the true value in a very difficult problem, such as chess games. And this kind of approximation is limited by feature extraction. Therefore, we cannot always choose the same policy as Value Iteration.

4d:

We run the experiment and then get this result:



```
 4d now:
Value Iteration Reward:6.84296666667
Q-learn Reward:11.502
```

We will find that Q-learn achieves a quite stronger result than Value Iteration. Since the value Iteration is just “remember” the policy of each states. It cannot handle problem with a little bit change. The policy which best for original MDP does not work on new problem. However, Q-learn can adapt this change.