HW4 SUNID: BENMA

1a. $V_{opt}^{(t)}(s) \leftarrow \max_{a \in Actions(s)} \sum_{s'} T(s,a,s')[Reward(s,a,s') + \gamma V_{opt}^{(t-1)}(s')]$

Iter = 0, $V_{opt} = \underset{-2 \quad -1 \quad 0 \quad +1 \quad +2}{[0, 0, 0, 0, 0]}$

Iter = 1, $V_{opt}(s=-2) = 0$

$\left.\begin{array}{l} V_{opt}(s=-1,-1) = 0.8 \times (20+0) + 0.2 \times (-5+0) = 15 \\ V_{opt}(s=-1,+1) = 0.7 \times (20+0) + 0.3 \times (-5+0) = 12.5 \end{array}\right\} \Rightarrow_{max} = 15$

$V_{opt}(s=0,-1) = 0.8 \times (-5 + 1 \times 0) + 0.2 \times (-5 + 1 \times 0) = -5$

$V_{opt}(s=0,+1) = 0.7 \times (-5) + 0.3 \times (-5) = -5$

$\left.\begin{array}{l} V_{opt}(s=1,-1) = 0.8 \times (-5+0) + 0.2 \times (100+0) = 16 \\ V_{opt}(s=1,+1) = 0.7 \times (-5) + 0.3 \times 100 = 26.5 \end{array}\right\} \Rightarrow_{max} = 26.5$

$V_{opt}(s=+2) = 0$

$V_{opt}^{(1)}(s) = \underset{-2 \quad -1 \quad 0 \quad +1 \quad +2}{[0, 15, -5, 26.5, 0]}$

Iter = 2, $V_{opt}(s=-2) = 0$

$\left.\begin{array}{l} V_{opt}(s=-1,-1) = 0.8 \times (20+0) + 0.2 \times (-5-5) = 14 \\ V_{opt}(s=-1,+1) = 0.7 \times (20+0) + 0.3 \times (-5-5) = 11 \end{array}\right\} \Rightarrow_{max} = 14$

$\left.\begin{array}{l} V_{opt}(s=0,-1) = 0.8 \times (-5 + 15) + 0.2 \times (-5 + 26.5 \times 1) = 12.3 \\ V_{opt}(s=0,+1) = 0.3 \times (-5 + 26.5 \times 1) + 0.7 \times (-5 + 15 \times 1) = 13.45 \end{array}\right\} \Rightarrow_{max} = 13.45$

$\left.\begin{array}{l} V_{opt}(s=+1,-1) = 0.8 \times (-5 + (-5) \times 1) + 0.2 \times (100+0) = 12 \\ V_{opt}(s=+1,+1) = 0.7 \times (-5 + (-5) \times 1) + 0.3 \times (100+0) = 23 \end{array}\right\} \Rightarrow_{max} = 23$

$V_{opt}(s=2) = 0$

$V_{opt}^{(2)}(s) = \underset{-2 \quad -1 \quad 0 \quad +1 \quad +2}{[0, 14, 13.45, 23, 0]}$

1 b. $\pi_{opt}(s) = \arg\max_{a \in Action(s)} Q_{opt}(s,a)$

so base on $V_{opt}^{(2)}(s)$, $\pi_{opt} = [\underset{s=-1}{-1}, \underset{s=0}{+1}, \underset{s=1}{+1}]$

2 b. Because we have an acyclic MDP, so it ~~will~~ will not go back to a previous state, we can use recursion to find the optimal value via one pass like below:

def recursive (S):

$V_{opt} = \max_{a \in Actions(s)} \sum_{s'} T(s,a,s')[Reward(s,a,s') + \underline{\underline{recusive\,V(s)}}]$

2 c. $V_{opt}^{(t)} = \max_{a \in Actions(s)} \sum_{s'} T(s,a,s')[Reward(s,a,s') + \gamma V_{opt}^{(t-1)}(s')]$

since $\gamma' = 1$, ~~then we can define $Reward'(s,a,s') = \frac{1}{T(s,a,s')}$~~

$V'_{opt} = \max_{a \in Actions(s)} \sum_{s'} T'(s,a,s')[Reward'(s,a,s') + V_{opt}^{(t-1)}(s')]$

$= \max_{a \in Actions(s)} \sum_{s'} (T' R' + T' V_{opt}^{(t-1)})$

$V_{opt} = \max_{a \in Actions(s)} \sum_{s'} (TR + \gamma T V_{opt}^{(t-1)})$

If $V'_{opt} = V_{opt}$ for all $s \in States$ Then $Reward'(s,a,s') = \frac{T}{T'} Reward = 0$

$T'(s,a,s') = (1-\gamma)\, T(s,a,s')$

4b.

After running multiple value iterations between smallMDP and largeMDP, I find that largerMDP has larger discrepancies than the smallMDP. The reason is that the largeMDP has more state to explore than the smallMDP. The difference for the largeMDP is 1102 and smallMDP is 4. The percentage difference is 99.23%. Another issue is that the feature extractor extracted only state and action pairs.

4d.

The average reward from originalMDP is 12 and the average reward from new thresholdMDP is 10.

When value iteration find a policy of the originalMDP, it is to find the optimal policy according to the original threshold of 10 and if the threshold change to 15, it's still trying to avoid busting. The reason is that Q-Learning is offline policy which can find the new optimal policy to collect the maximum rewards but fixRLalgorithm is stateful, partial feedback, which cannot find optimal policy.

Therefore Q Learning is adaptable and can handle changes of thresholds.