Q1: (a) $V_*(s) = \max_a q_*(s,a)$

(b) $q_*(s,a) = \sum_{s',r} p(s',r|s,a)(r + \gamma V_*(s'))$

(c) $\pi_*(s) = \arg\max_a q_*(s,a)$

(d) $\pi_*(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)(r + \gamma V_*(s'))$

(e) $\begin{cases} p(s'|s,a) = \sum_r p(s',r|s,a) \\ r(s,a) = \sum_{s',r} p(s',r|s,a) \times r. \end{cases}$

$V_\pi(s) = \sum_a \pi(a|s)\left[r(s,a) + \sum_{s'} p(s'|s,a) \times \gamma V_\pi(s')\right]$

$V_*(s) = \max_a \left[r(s,a) + \sum_{s'} p(s'|s,a) \times \gamma V_*(s')\right]$

$q_\pi(s,a) = r(s,a) + \sum_{s'} p(s'|s,a) \times \gamma \sum_{a'} \pi(a'|s) \times q(s',a')$

$q_*(s,a) = r(s,a) + \sum_{s'} p(s'|s,a) \times \max_a q_*(s',a')$

Q2: (a) Since the condition for the policy iteration algorithm to stop optimizing is that the policy $\pi(s)$ no longer changes (meaning that the optimal policy is obtained), but for real use we care more about the results rather than the policy, once we got a result that is good enough for use to use (even if the policy is different but the result no longer changes), we can stop optimizing the policy.

Therefore, we need to change the judgment condition in the pseudocode to determine whether the policy should be optimized by the result obtained from different policies.

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      old-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V(s')]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Change the yellow part to: $q_{\text{old-action}}(s,a) \neq q_\pi(s,a)$

(b) Value iteration does not have similiar problems, because value iteration does not only focus on the policy like the policy iteration, but on the value. The policy is optimal as long as the result is optimal.

Q3: (a). 1. Initialization:

$Q(s,a) \in \mathbb{R}$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$.

2. Policy Evaluation:

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in S$

$a \leftarrow \pi(s)$

$q \leftarrow Q(s,a)$

$Q(s,a) \leftarrow \sum_{s',r} p(s', r \mid s, a)[r + \gamma \sum_{a'} \pi(a' \mid s') Q(s', a')]$

$\Delta \leftarrow \max(\Delta, |q - Q(s,a)|)$

Until $\Delta < \theta$ (a small positive number ...)

3. Policy Improvement:

policy-stable $\leftarrow$ True

For each $s \in S$:

old-action $\leftarrow \pi(s)$

$\pi(s) \leftarrow \underset{a}{\text{argmax}}\, q(s,a)$

If $\sum_{s',r} p(s',r|s,\pi(s))[r+\gamma \sum_{a'} \pi(a'|s') q(a',s')] \neq$

$\sum_{s',r} p(s',r|s,\text{old\_action})[r+\gamma \sum_{a'} \pi(a'|s') \times q(a',s')]$.

Then policy-stable $\leftarrow$ False

If policy-stable, then stop and return $Q \approx q_*$, and $\pi \approx \pi_*$; else, go to 2.

(b). $\theta > 0$

Initialize $Q(s,a)$, for all $s \in S^+$, arbitrarily except that $Q(\text{terminal}, a)$.

Loop:

$\Delta \leftarrow 0$

Loop for each $s \in S$:

$a \leftarrow \underset{a}{\text{argmax}}\, q(s,a)$

$q \leftarrow Q(s,a)$

$Q(s,a) \leftarrow \sum_{s',r} p(s',r|s,a)[r+\gamma \max_{a} q(s',a')]$

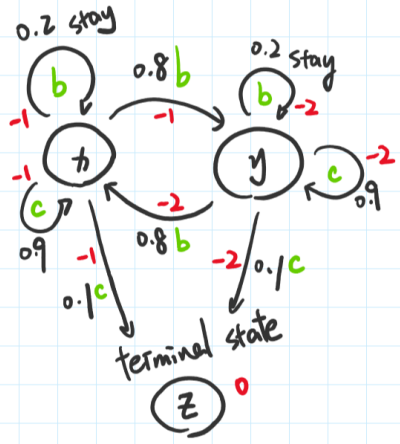
$\Delta \leftarrow \max(\Delta, |q-Q(s,a)|)$

Until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that

$\pi(s) = \underset{a}{\text{argmax}} \sum_{s',r} p(s',r|s,a)[r+\gamma q(s',\pi(s'))]$

Q4 :



(a) I think we can be sure that moving to z is the best action for both states x and y. But for state y, moving to x is also a better action than staying, because it reduces the amount of negative future reward gained. Although policy b reduces negative future reward by a smaller amount than policy c for y, it makes it more likely than policy c that negative future reward will be reduced. Therefore, I think policy c should be the optimal policy for state x, and policy b may be the optimal policy for state y.

(b). ① Policy evaluation.

$$V_\pi(s) = \sum_a \pi(a|s) \left[\sum_{s',r} p(s',r|s,a)(r + \gamma V_\pi(s')) \right]$$

Initial policy has action c in both states :

$$\pi(x) = c, \quad \pi(y) = c$$

$$\begin{cases} V_{\pi c}(x) = 0.1 \times (-1+0) + 0.9 \times (-1 + \gamma V_{\pi c}(x)) \\ V_{\pi c}(y) = 0.1 \times (-2+0) + 0.9 \times (-2 + \gamma V_{\pi c}(y)) \end{cases}$$

$$\Rightarrow \begin{cases} V_{\pi c}(x) = -\frac{1}{1-0.9\gamma} \\ V_{\pi c}(y) = -\frac{2}{1-0.9\gamma} \end{cases}$$

For $\gamma = 1$. $\begin{cases} V_{\pi c}(x) = -10 \\ V_{\pi c}(y) = -20. \end{cases}$

Policy improvement.

When we apply policy b.

$$\begin{cases} V_{\pi b}(x) = 0.2\times(-1+\gamma V_{\pi c}(x)) + 0.8\times(-1+\gamma V_{\pi c}(y)) \\ V_{\pi b}(y) = 0.2\times(-2+\gamma V_{\pi c}(y)) + 0.8\times(-2+\gamma V_{\pi c}(x)) \end{cases}$$

$$\Rightarrow \text{For } \gamma=1 \begin{cases} V_{\pi b}(x) = -19 < V_{\pi c}(x) = -10 \\ V_{\pi b}(y) = -14 > V_{\pi c}(y) = -20. \end{cases}$$

Thus. $\pi'(x) \to c$, $\pi'(y) \to b$.

$\pi'(x) \neq \pi(x)$, $\pi'(y) \neq \pi(y)$. It doesn't meet the condition of policy-stable, so return to policy evaluation.

② Policy evaluation.

When we apply policy as last improvement.

$$\begin{cases} V_{\pi c}(x) = 0.1\times(-1+0) + 0.9\times(-1+V_{\pi c}(x)) \\ V_{\pi b}(y) = 0.2\times(-1+\gamma V_{\pi b}(y)) + 0.8\times(-2+\gamma V_{\pi c}(x)) \end{cases}$$

$$\Rightarrow \text{For } \gamma=1 \begin{cases} V_{\pi c}(x) = -10 \\ V_{\pi b}(y) = -12.5 \end{cases}$$

Policy improvement

When we apply different policy again.

$$\begin{cases} V_{\pi'b}(x) = 0.2 \times (-1 + \gamma V_{\pi'c}(x)) + 0.8 \times (-1 + \gamma V_{\pi'b}(y)) \\ V_{\pi'c}(y) = 0.1 \times (-2 + 0) + 0.9 \times (-2 + \gamma V_{\pi'b}(y)) \end{cases}$$

For $\gamma = 1$. $\begin{cases} V_{\pi'b}(x) = -13 < V_{\pi'c}(x) = -10 \\ V_{\pi'c}(y) = -13.25 < V_{\pi'b}(y) = -12.5 \end{cases}$

Thus, $\pi''(x) \to c$, $\pi''(y) \to b$.

$\pi''(x) = \pi'(x)$, $\pi''(y) = \pi'(y)$, it meets the condition of policy-stable, so stop and return:

$$\begin{cases} V_*(x) = -10 \\ V_*(y) = -12.5 \end{cases} \qquad \begin{cases} \pi_*(x) = c \\ \pi_*(y) = b. \end{cases}$$

(c). ① Policy evaluation.

$$V_\pi(s) = \sum_a \pi(a|s) \left[\sum_{s',r} p(s',r|s,a)(r + \gamma V_\pi(s')) \right]$$

Initial policy has action b in both states:

$$\pi(x) = b \quad \pi(y) = b.$$

$$\begin{cases} V_{\pi b}(x) = 0.2 \times (-1 + \gamma V_{\pi b}(x)) + 0.8 \times (-1 + \gamma V_{\pi b}(y)) \\ V_{\pi b}(y) = 0.2 \times (-2 + \gamma V_{\pi b}(y)) + 0.8 \times (-2 + \gamma V_{\pi b}(x)) \end{cases}$$

For $\gamma = 1$: The equation does not hold and the result cannot be caculated.

For $\gamma \neq 1$:

$$\begin{cases} V_{\pi b}(y) = \dfrac{-2 + 0.4\gamma}{1 - 0.4\gamma - 0.6\gamma^2} \\ V_{\pi b}(x) = \dfrac{-1 + 0.8\gamma}{1 - 0.2\gamma} \times \dfrac{-2 + 0.4\gamma}{1 + 0.4\gamma - 0.6\gamma^2}. \end{cases}$$

Thus, I think discounting may have impact on the results, depending on the different discount factors. We need to analyze the optimal policy according to the specific discount factor.

Q5:

```
Vs(a) =  [[-1.85380443 -1.34185832 -1.22622928 -1.42007309 -1.97241846]
 [-0.96965064 -0.43208514 -0.35180898 -0.58272448 -1.18027658]
 [ 0.05486787  0.74165922  0.67626363  0.36114423 -0.40025498]
 [ 1.52582318  2.99591435  2.2534199   1.91064941  0.55045095]
 [ 3.31359559  8.79292942  4.43113177  5.32556099  1.4955287 ]]
Vs(b) =
 [[21.9764967  24.41877948 21.97690153 19.41877948 17.47690153]
 [19.77884703 21.97690153 19.77921138 17.80129024 16.02116122]
 [17.80096232 19.77921138 17.80129024 16.02116122 14.4190451 ]
 [16.02086609 17.80129024 16.02116122 14.4190451  12.97714059]
 [14.41877948 16.02116122 14.4190451  12.97714059 11.67942653]]
pi(s)_b =
 [['right' 'up' 'left' 'up' 'left']
 ['up' 'up' 'up' 'left' 'left']
 ['up' 'up' 'up' 'up' 'up']
 ['up' 'up' 'up' 'up' 'up']
 ['up' 'up' 'up' 'up' 'up']]
Vs(c) =
 [[21.97748529 24.4194281  21.97748529 18.45018449 16.60516604]
 [19.77973676 21.97748529 19.77973676 17.80176308 16.02158677]
 [17.80176308 19.77973676 17.80176308 16.02158677 14.4194281 ]
 [16.02158677 17.80176308 16.02158677 14.4194281  12.97748529]
 [14.4194281  16.02158677 14.4194281  12.97748529 11.67973676]]
pi(s)_c =
 [['right' 'up' 'left' 'up' 'left']
 ['up' 'up' 'up' 'left' 'left']
 ['up' 'up' 'up' 'up' 'up']
 ['up' 'up' 'up' 'up' 'up']
 ['up' 'up' 'up' 'up' 'up']]
```