گزارشکار پروژه ۳

۱) تکرار ارزش

```
84      def computeQValueFromValues(self, state, action):
85          """
86            Compute the Q-value of action in state from the
87            value function stored in self.values.
88          """
89          "*** YOUR CODE HERE ***"
90          Q_value = 0
91          for prob_state in self.mdp.getTransitionStatesAndProbs(state, action):
92              Q_value += prob_state[1] * (
93                      self.mdp.getReward(state, action, prob_state[0]) + self.discount * self.values[prob_state[0]])
94          return Q_value
```

توضیحات : دراین تابع قصد داریم Q_value را به ازای state وaction داده شده بدست آوریم. برای این کار از فرمول زیر استفاده می کنیم(خطوط ۹۱ و۹۲) :

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

در انتها مقدارQ_value را برمی گردانیم . نکته قابل ذکر این است که prob_state ای که درخط ۹۱ وجود دارد یک آرایه ۲ بعدی است که خانه اول آن state احتمالی و خانه دوم آن احتمال رفتن به آن state است.

```
61      def runValueIteration(self):
62          # Write value iteration code here
63          """*** YOUR CODE HERE ***"""
64          for i in range(self.iterations):
65              temp = self.values.copy()
66              for state in self.mdp.getStates():
67                  if self.mdp.isTerminal(state):
68                      temp[state] = 0
69                  else:
70                      Q_max = -float('inf')
71                      possible_actions = self.mdp.getPossibleActions(state)
72                      for possible_action in possible_actions:
73                          Q_max = max(Q_max, self.computeQValueFromValues(state, possible_action))
74                      if Q_max > -float('inf'):
75                          temp[state] = Q_max
76              self.values = temp
```

توضیحات : دراین تابع قصد داریم value هر state را تا عمق ۱۰۰ بدست آوریم . برای بدست آوردن value هر state ابتدا چک می کنیم که اگرterminal state بود مقدار ۰ بگیرد درغیر این صورت بین همه Q هایی که state آن ، state ای است که

می خواهیم مقدار آن را update کنیم و action آن یکی از action های مجازstate مورد بررسی است ماکسیمم مقداررا به عنوان value جدید آن state درنظرمی گیریم .

نکته قابل ذکراین است تا قبل از پایان هرiteration ، state ها با value های قدیم state های دیگر کار می کنند ودرانتهای کار value ها update می شوند.

```
97      def computeActionFromValues(self, state):
98          """
99            The policy is the best action in the given state
100           according to the values currently stored in self.values.
101
102           You may break ties any way you see fit.  Note that if
103           there are no legal actions, which is the case at the
104           terminal state, you should return None.
105         """
106         "*** YOUR CODE HERE ***"
107         if not self.mdp.isTerminal(state):
108             possible_actions = self.mdp.getPossibleActions(state)
109             Q_max = -float('inf')
110             best_action = None
111             for possible_action in possible_actions:
112                 current_Q = self.computeQValueFromValues(state, possible_action)
113                 if current_Q > Q_max:
114                     Q_max = current_Q
115                     best_action = possible_action
116             return best_action
```

توضیحات : در این تابع قصد داریم بهترین action را برای هر state بدست آوریم . برای این کار ابتدا Q_value مربوط به آن state و هرaction مجاز را درنظرگرفته و action ای که منجر به بدست آمدن بیشترین Q_value شده را برمی گردانیم.

درواقع ازفرمول زیر استفاده کرده ایم :

$$\pi\ (s) = \arg\max_{a} Q\ (s, a)$$

خروجی تست ها :

```
Starting on 1-19 at 22:06:45

Question q1
==========

*** PASS: test_cases\q1\1-tinygrid.test
*** PASS: test_cases\q1\2-tinygrid-noisy.test
*** PASS: test_cases\q1\3-bridge.test
*** PASS: test_cases\q1\4-discountgrid.test

### Question q1: 4/4 ###


Finished at 22:06:46

Provisional grades
==================
Question q1: 4/4
------------------
Total: 4/4
```

۲) تجزیه وتحلیل عبور از پل

گزارش: دلیل انتخاب این مقادیر را به زبان ساده و به صورت شهودی توضیح دهید.

در اینجا باید noise را تا مقدار تقریباً صفر کاهش دهیم تا احتمال انحراف عامل به طرفین و افتادن در state های با پاداش منفی به ۰ میل کند .

خروجی تست ها :

```
Starting on 1-19 at 22:19:06

Question q2
==========

*** PASS: test_cases\q2\1-bridge-grid.test

### Question q2: 1/1 ###


Finished at 22:19:06

Provisional grades
==================
Question q2: 1/1
------------------
Total: 1/1
```

۳) سیاست ها

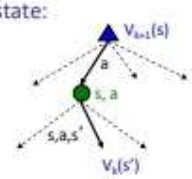
گزارش:آیا استفاده از الگوریتم تکرار ارزش تحت هر شرایطی به همگرایی می انجامد؟

بله . با توجه به اسلایدهای درس .

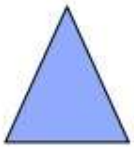
Value Iteration

- Start with $V_0(s) = 0$: no time steps left means an expected reward sum of zero
- Given vector of $V_k(s)$ values, do one ply of expectimax from each state:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$



- Repeat until convergence
- Complexity of each iteration: $O(S^2A)$
- Theorem: will converge to unique optimal values
  - Basic idea: approximations get refined towards optimal values
  - Policy may converge long before values do



```
Starting on 1-19 at 22:34:04

Question q3
===========

*** PASS: test_cases\q3\1-question-3.1.test
*** PASS: test_cases\q3\2-question-3.2.test
*** PASS: test_cases\q3\3-question-3.3.test
*** PASS: test_cases\q3\4-question-3.4.test
*** PASS: test_cases\q3\5-question-3.5.test

### Question q3: 5/5 ###


Finished at 22:34:04

Provisional grades
==================
Question q3: 5/5
------------------
Total: 5/5
```

۴) تکرار ارزش ناهمزمان

```python
def runValueIteration(self):
    states = self.mdp.getStates()
    for i in range(self.iterations):
        if self.mdp.isTerminal(states[i % len(states)]):
            continue
        Q_max = -float('inf')
        for possible_action in self.mdp.getPossibleActions(states[i % len(states)]):
            Q_max = max(Q_max, self.computeQValueFromValues(states[i % len(states)], possible_action))
        if Q_max > -float('inf'):
            self.values[states[i % len(states)]] = Q_max
```

توضیحات : دراین تابع نیزمانند تابع بخش ۱ قصد داریم value هر state را تا عمق معینی بدست آوریم با این تفاوت که دراینجا value هر state درجا update شده وstate های دیگر با این مقدار جدید کارمی کنند.

نکته قبل ذکراین است که value مربوط به state ها به ترتیب و درقالب یک cycle، update می شوند.

سوال: روش های بروزرسانی ای که در بخش اول بروزرسانی با استفاده از(batch)ودر این بخش (بروزرسانی به صورت تکی) پیاده کرده اید را با یکدیگر مقایسه کنید. (یک نکته مثبت و یک نکته منفی برای هرکدام)

درروش غیرهمگام ما کم تر از روشbatch عمیق می شویم چون به تعداد iterations/len(states) برای هر state عمیق می شویم ولی درروش batch به تعداد iterations و این باعث می شود مقداری که درروش batch بدست می آوریم با احتمال بالایی دقیق تر باشد اما از طرفی بدی روش batch این است که از مقادیر قدیم هر state استفاده می کند و همین طور کندتراست و دیرتر value ها converge می شوند.

خروجی تست ها :

```
Starting on 1-19 at 23:03:50

Question q4
===========

*** PASS: test_cases\q4\1-tinygrid.test
*** PASS: test_cases\q4\2-tinygrid-noisy.test
*** PASS: test_cases\q4\3-bridge.test
*** PASS: test_cases\q4\4-discountgrid.test

### Question q4: 1/1 ###


Finished at 23:03:50

Provisional grades
==================
Question q4: 1/1
------------------
Total: 1/1
```

ه) تکرار ارزش اولویت بندی شده

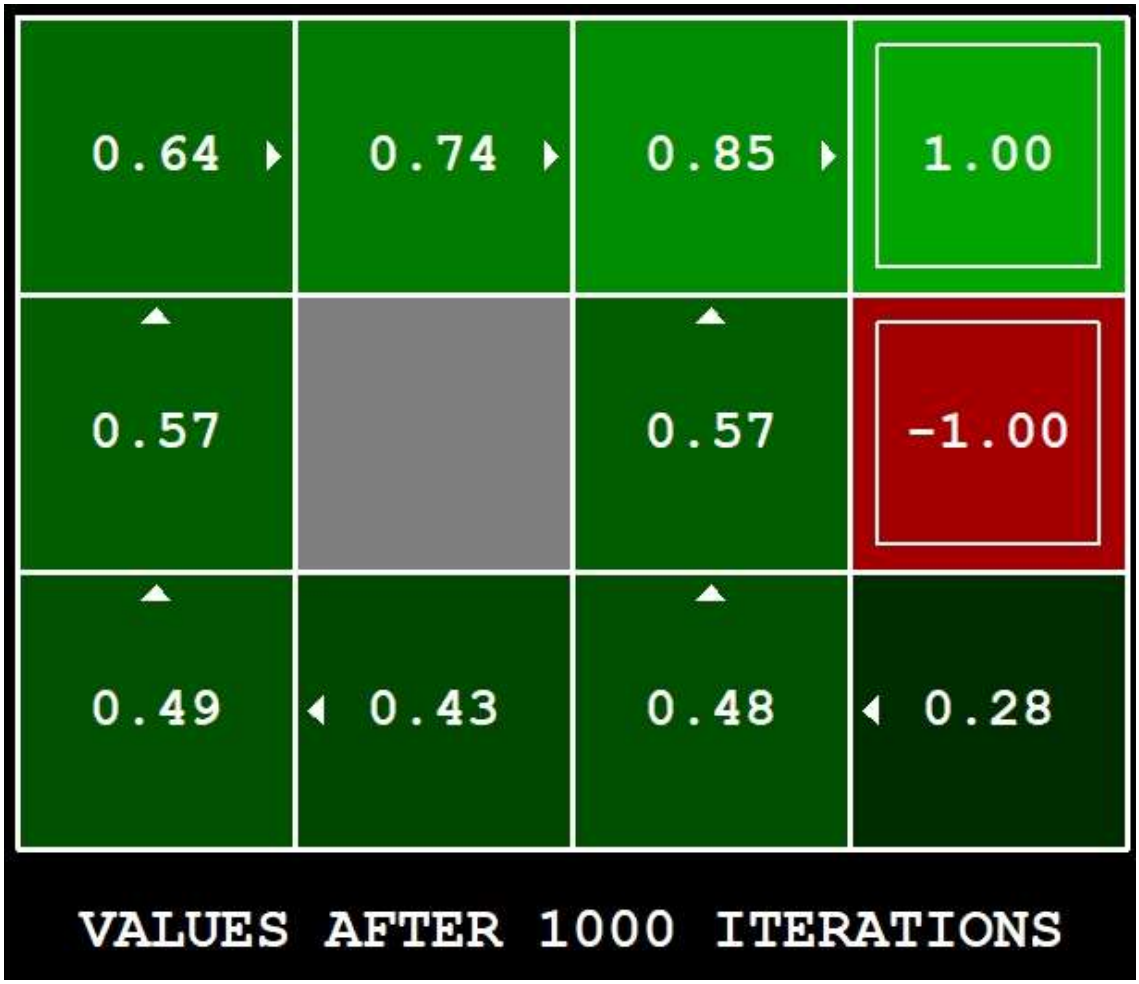
```
def runValueIteration(self):
    """*** YOUR CODE HERE ***"""
    predecessors = {}
    for state in self.mdp.getStates():
        predecessors[state] = self.findPredecessors(state)

    priorityQueue = util.PriorityQueue()
    for state in self.mdp.getStates():
        if not self.mdp.isTerminal(state):
            Q_max = -float('inf')
            for possible_action in self.mdp.getPossibleActions(state):
                Q_max = max(Q_max, self.computeQValueFromValues(state, possible_action))
            diff = abs(self.values[state] - Q_max)
            priorityQueue.update(state, -diff)

    for i in range(self.iterations):
        if not priorityQueue.isEmpty():
            state = priorityQueue.pop()

            if not self.mdp.isTerminal(state):
                possible_actions = self.mdp.getPossibleActions(state)
                Q_max= -float('inf')
                for possible_action in possible_actions:
                    Q_max = max(Q_max,self.computeQValueFromValues(state,possible_action))
                if Q_max > -float('inf'):
                    self.values[state] = Q_max

                for predecessor in list(predecessors[state]):
                    if not self.mdp.isTerminal(predecessor):
                        possible_actions = self.mdp.getPossibleActions(predecessor)
                        Q_max = - float('inf')
                        for possible_action in possible_actions:
                            Q_max = max(Q_max, self.computeQValueFromValues(predecessor, possible_action))
                        diff = abs(self.values[predecessor] - Q_max)
                        if diff > self.theta:
                            priorityQueue.update(predecessor, -diff)
        else:
            break
```

```
def findPredecessors(self, currentState):

    predecessors = set()

    if not self.mdp.isTerminal(currentState):
        for state in self.mdp.getStates():
            possible_actions = self.mdp.getPossibleActions(state)
            for possible_action in possible_actions:
                prob_states = self.mdp.getTransitionStatesAndProbs(state, possible_action)
                for prob_state in prob_states:
                    if prob_state[0] == currentState and prob_state[1] > 0:
                        predecessors.add(state)

    return predecessors
```

توضیحات : مطابق الگوریتم داخل پروژه این قسمت را پیاده سازي کرده ایم.

خروجی تست ها :

```
Starting on 1-19 at 23:15:37

Question q5
==========

*** PASS: test_cases\q5\1-tinygrid.test
*** PASS: test_cases\q5\2-tinygrid-noisy.test
*** PASS: test_cases\q5\3-bridge.test
*** PASS: test_cases\q5\4-discountgrid.test

### Question q5: 3/3 ###


Finished at 23:15:37

Provisional grades
==================
Question q5: 3/3
------------------
Total: 3/3
```



۶) یادگیری Q

```
42      def __init__(self, **args):
43          "You can initialize Q-values here..."
44          ReinforcementAgent.__init__(self, **args)
45
46          "*** YOUR CODE HERE ***"
47          self.QValues = util.Counter()
48
49      def getQValue(self, state, action):
50          """
51            Returns Q(state,action)
52            Should return 0.0 if we have never seen a state
53            or the Q node value otherwise
54          """
55          "*** YOUR CODE HERE ***"
56          return self.QValues[(state, action)]
57          # util.raiseNotDefined()
```

توضیحات : ابتدا مقدارvalue ها را در init تعریف مي کنیم. سپس در getQValue با دادن state و action مقدار value را مي گیریم.

```
59      def computeValueFromQValues(self, state):
60          """
61            Returns max_action Q(state,action)
62            where the max is over legal actions.  Note that if
63            there are no legal actions, which is the case at the
64            terminal state, you should return a value of 0.0.
65          """
66          "*** YOUR CODE HERE ***"
67          legal_actions = self.getLegalActions(state)
68          Q_max = -float('inf')
69          for legal_action in legal_actions:
70              Q_max = max(Q_max, self.getQValue(state, legal_action))
71
72          if Q_max == -float('inf'):
73              Q_max = 0
74          return Q_max
```

توضیحات : دراین تابع Q_max مربوط به هر state را با بدست آورن Q(s,a) به ازای هرaction مجاز، برمی گردانیم .

```
77      def computeActionFromQValues(self, state):
78          """
79            Compute the best action to take in a state.  Note that if there
80            are no legal actions, which is the case at the terminal state,
81            you should return None.
82          """
83          "*** YOUR CODE HERE ***"
84          legal_actions = self.getLegalActions(state)
85          best_actions = []
86          if len(legal_actions) == 0:
87              return None
88          value = self.computeValueFromQValues(state)
89
90          for legal_action in legal_actions:
91              if self.getQValue(state, legal_action) == value:
92                  best_actions.append(legal_action)
93
94          return random.choice(best_actions)
```

توضیحات : در این تابع اگر action مجازي نداشتیم None برگردانده ودرغیراین صورت از میانaction هایی که منجر به رسیدن به Q_max شده اند یکی را به صورت رندوم انتخاب کرده و برمي گردانیم.

```
122     def update(self, state, action, nextState, reward):
123         """
124           The parent class calls this to observe a
125           state = action => nextState and reward transition.
126           You should do your Q-Value update here
127
128           NOTE: You should never call this function,
129           it will be called on your behalf
130         """
131         "*** YOUR CODE HERE ***"
132         self.QValues[(state, action)] = (1 - self.alpha) * self.QValues[(state, action)] + self.alpha * (
133                 reward + self.discount * self.getValue(nextState))
```

توضیحات : برای پیاده سازی این تابع از فرمول زیر استفاده می کنیم :

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\left[r + \gamma \max_{a'} Q(s',a')\right]$$

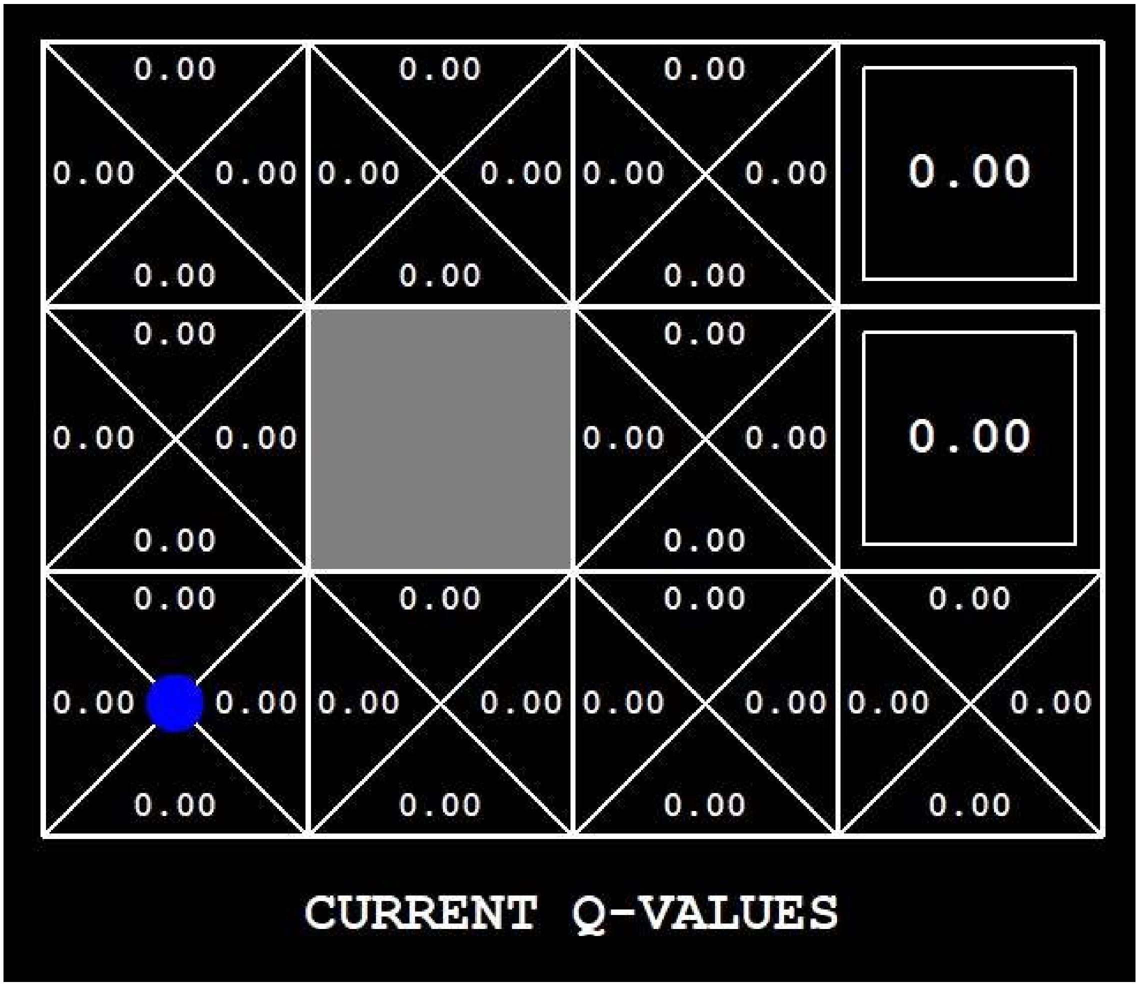
گزارش: توضیح دهیدکه اگر مقدار Q برای اقداماتی که عامل قبلا ندیده، بسیارکم یا بسیار زیاد باشد چه اتفاقی می افتد.

اگرمقدارQ برای اقدامی بسیار زیاد باشد اما عامل به دلیل انتخاب رندوم بین اقداماتی که منجر به بدست آمدن Q_max می شوند ، این اقدام را انتخاب نکرده باشد با احتمال

(تعداد اکشن هایی که منجر به بدست آمدن Q_max می شوند)/۱ امکان انتخاب شدن این اقدام وجود دارد . اما اگر مقدارQ برای اقدامی بسیارکم باشد(به عنوان مثال ۰) و بقیه اقداماتی که عامل انجام داده منجر به Q صفر یا منفی شده باشد با احتمال (تعداد اکشن هایی که منجر به بدست آمدن Q = 0 می شوند)/۱ امکان انتخاب شدن این اقدام وجود دارد.

خروجی تست ها :

```
Starting on 1-20 at 0:02:20

Question q6
===========

*** PASS: test_cases\q6\1-tinygrid.test
*** PASS: test_cases\q6\2-tinygrid-noisy.test
*** PASS: test_cases\q6\3-bridge.test
*** PASS: test_cases\q6\4-discountgrid.test

### Question q6: 4/4 ###


Finished at 0:02:20

Provisional grades
==================
Question q6: 4/4
------------------
Total: 4/4
```

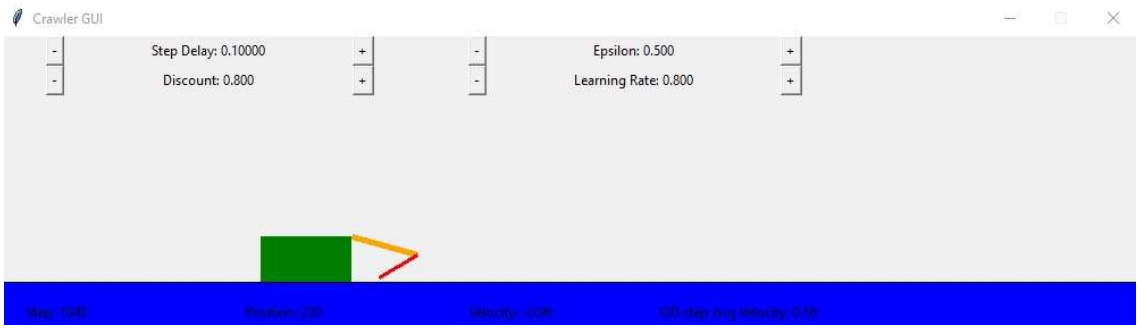

0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00 0.00
0.00 0.00
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00
CURRENT Q-VALUES

۷) epsilon حریصانه

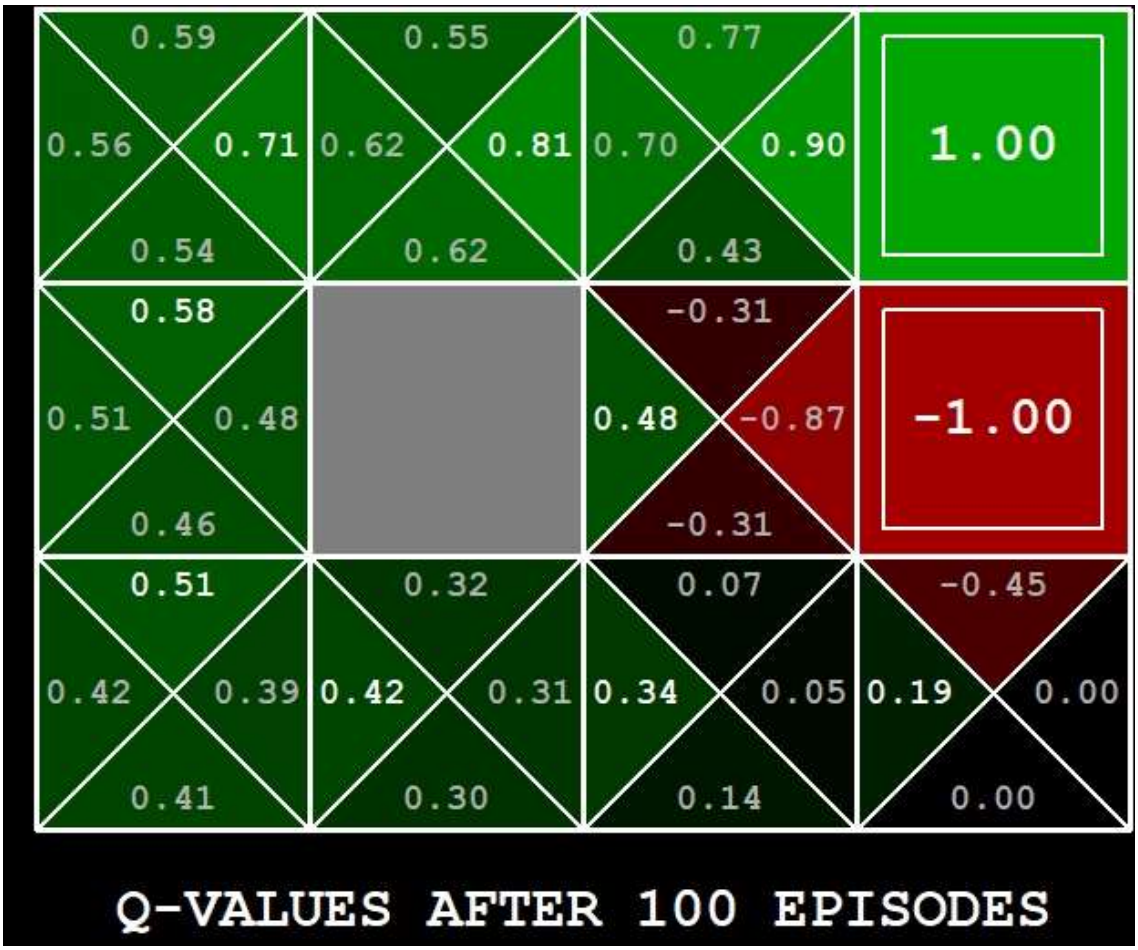
```python
def getAction(self, state):
    """
      Compute the action to take in the current state.  With
      probability self.epsilon, we should take a random action and
      take the best policy action otherwise.  Note that if there are
      no legal actions, which is the case at the terminal state, you
      should choose None as the action.

      HINT: You might want to use util.flipCoin(prob)
      HINT: To pick randomly from a list, use random.choice(list)
    """
    # Pick Action
    legalActions = self.getLegalActions(state)
    action = None
    "*** YOUR CODE HERE ***"
    if len(legalActions) > 0:
        if util.flipCoin(self.epsilon):
            action = random.choice(legalActions)
        else:
            action = self.computeActionFromQValues(state)

    return action
```

توضیحات : دراین تابع با احتمال Epsilon به صورت random یکی از action های مجاز را انتخاب می کنیم و با احتمال Epsilon – ۱ یکی از action های که منجر به Q_max می شوند.

خروجی تست ها :



```
Starting on 1-20 at 0:28:24

Question q7
==========

*** PASS: test_cases\q7\1-tinygrid.test
*** PASS: test_cases\q7\2-tinygrid-noisy.test
*** PASS: test_cases\q7\3-bridge.test
*** PASS: test_cases\q7\4-discountgrid.test

### Question q7: 2/2 ###


Finished at 0:28:25

Provisional grades
==================
Question q7: 2/2
------------------
Total: 2/2
```



۸) بررسی دوباره عبور از پل

گزارش: به صورت ساده و شهودی توضیح دهیدکه باکم یا زیادکردن مقدار epsilonروند یادگیری عامل چگونه تغییر می کند.

هرچه مقدار epsilon کم باشد تلاش برای یادگیری محیط(explore) کم تر و بهره بردن از یادگیری تا بدین لحظه (exploit) بیشتر است و برعکس .

۹) پک من و Q-Learning

گزارش: تغییرات و فعالیت هایی که در این بخش انجام داده اید را توضیح دهید.

تغییراتی در این بخش انجام نشده است .

۱۰) یادگیری تقریبی Q

```
191     def getQValue(self, state, action):
192         """
193           Should return Q(state,action) = w * featureVector
194           where * is the dotProduct operator
195         """
196         "*** YOUR CODE HERE ***"
197         features_vector = self.featExtractor.getFeatures(state, action)
198         Q_value = 0
199         for feature in features_vector:
200             Q_value += self.weights[feature] * features_vector[feature]
201         return Q_value
```

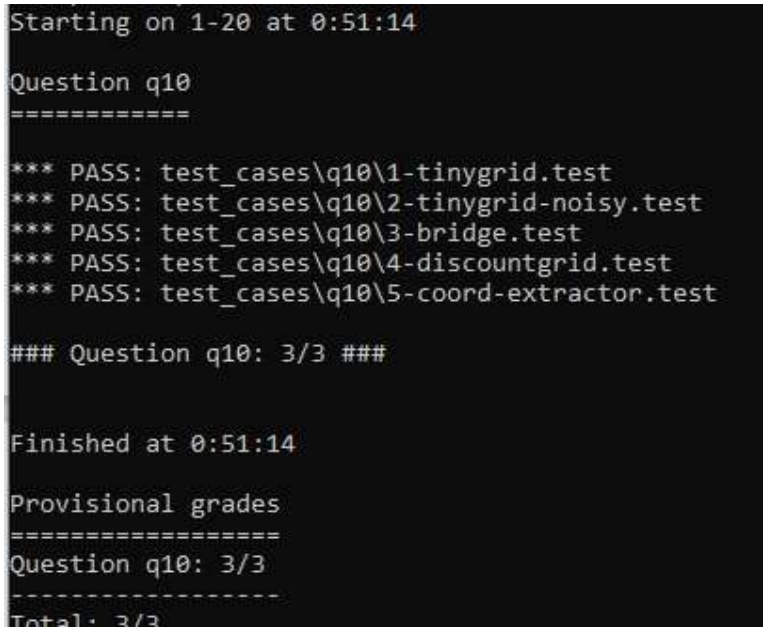
توضیحات : دراین تابع با استفاده از فرمول زیر مقدار Q(s,a) را تقریب می زنیم .

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

```
203     def update(self, state, action, nextState, reward):
204         """
205            Should update your weights based on transition
206         """
207         "*** YOUR CODE HERE ***"
208         features_vector = self.featExtractor.getFeatures(state, action)
209         diff = (reward + self.discount * self.getValue(nextState)) - self.getQValue(state, action)
210         for feature in features_vector:
211             self.weights[feature] += self.alpha * diff * features_vector[feature]
212
```

توضیحات : دراین تابع با استفاده از فرمول زیروزن ها را update می کنیم .

خروجی تست ها :


Starting on 1-20 at 0:51:14
Question q10
*** PASS: test_cases\q10\1-tinygrid.test
*** PASS: test_cases\q10\2-tinygrid-noisy.test
*** PASS: test_cases\q10\3-bridge.test
*** PASS: test_cases\q10\4-discountgrid.test
*** PASS: test_cases\q10\5-coord-extractor.test
Question q10: 3/3
Finished at 0:51:14
Provisional grades
Question q10: 3/3