گزارشکارپروژه ۴

۱) احتمال مشاهده

```python
def getObservationProb(self, noisyDistance, pacmanPosition, ghostPosition, jailPosition):
    """
    Return the probability P(noisyDistance | pacmanPosition, ghostPosition).
    """
    "*** YOUR CODE HERE ***"
    if ghostPosition == jailPosition:
        if noisyDistance is None:
            return 1
        else:
            return 0
    if noisyDistance is None:
        return 0
    return busters.getObservationProbability(noisyDistance, manhattanDistance(ghostPosition, pacmanPosition))
```

توضیحات : دراین تابع قصد داریم با توجه به noisyDistance که ازسنسورشنوایی کسب شده است، احتمال وجود روح درموقعیت ghostPosition را بدست آوریم .
اگرnoisyDistance مقدارNone باشد وموقعیتی که می خواهیم احتمال وجود روح درآن را بدست آوریم موقعیت زندان باشد، احتمال برابر با ۱ است ولی اگرموقعیتی که می خواهیم احتمال وجود روح درآن را بدست آوریم موقعیت زندان نباشد، احتمال ۰ است .

حال اگرnoisyDistance مقدارNone نباشد وموقعیتی که می خواهیم احتمال وجود روح درآن را بدست آوریم موقعیت زندان باشد، احتمال ۰ است ولی اگرموقعیتی که می خواهیم احتمال وجود روح درآن را بدست آوریم موقعیت زندان نباشد، احتمال وجود روح درآن موقعیت با توجه به فاصله آن موقعیت تا موقعیت پکمن ومقدارnoisyDistance با استفاده ازتابع getObservationProbability بدست می آید.

خروجی تست ها

```
Starting on 1-25 at 12:14:06

Question q1
===========
*** PASS: test_cases\q1\1-ObsProb.test
***      PASS

### Question q1: 2/2 ###


Finished at 12:14:06

Provisional grades
==================
Question q1: 2/2
------------------
Total: 2/2
```

۲) مشاهده استنتاج دقیق

```
def observeUpdate(self, observation, gameState):
    """
    Update beliefs based on the distance observation and Pacman's position.

    The observation is the noisy Manhattan distance to the ghost you are
    tracking.

    self.allPositions is a list of the possible ghost positions, including
    the jail position. You should only consider positions that are in
    self.allPositions.

    The update model is not entirely stationary: it may depend on Pacman's
    current position. However, this is not a problem, as Pacman's current
    position is known.
    """
    "*** YOUR CODE HERE ***"
    allGhostPositions = self.allPositions
    beliefs = self.beliefs
    pacmanPosition = gameState.getPacmanPosition()
    jailPosition = self.getJailPosition()
    for position in allGhostPositions:
        prob = self.getObservationProb(observation, pacmanPosition, position, jailPosition)
        prior = beliefs[position]
        beliefs[position] = prob * prior
    beliefs.normalize()
```

توضیحات : دراین تابع قصد داریم با توجه به noisyDistance که ازسنسورشنوایی کسب شده است، احتمال وجود روح درتمام موقعیت هایی که ممکن است روح درآن جا باشد راupdate کنیم. برای هریک از موقعیت های احتمالی میزان احتمال را باتوجه به noisyDistance از طریق تابع getObservationProb بدست می آوریم ومیزان احتمال وجود روح درآن موقعیت را برابر با حاصل ضرب مقدارقبلی احتمال آن و مقداراحتمال بدست آمده ازgetObservationProb قرارمی دهیم .(به عنوان مثال اگرتا قبل از کسب noisyDistance جدید تصورمی کردیم با احتمال ۰/۶ روح دریک موقعیت است و noisyDistance جدید به ما می گوید احتمال روح دراین موقعیت ۰/۸ است ، احتمال وجود وجود روح درآن موقعیت را ۰/۴۸ درنظرمی گیریم).

خروجی تست ها

```
Starting on 1-25 at 12:28:04

Question q2
===========
*** q2) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases\q2\1-ExactUpdate.test
*** q2) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases\q2\2-ExactUpdate.test
*** q2) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases\q2\3-ExactUpdate.test
*** q2) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases\q2\4-ExactUpdate.test

### Question q2: 3/3 ###


Finished at 12:28:04

Provisional grades
==================
Question q2: 3/3
------------------
Total: 3/3
```

۳) استنتاج دقیق با گذشت زمان

```
318     def elapseTime(self, gameState):
319         """
320         Predict beliefs in response to a time step passing from the current
321         state.
322
323         The transition model is not entirely stationary: it may depend on
324         Pacman's current position. However, this is not a problem, as Pacman's
325         current position is known.
326         """
327         "*** YOUR CODE HERE ***"
328         beliefs = self.beliefs
329         GhostsPositions = self.allPositions
330
331         result = DiscreteDistribution()
332         for prvPositions in GhostsPositions:
333             newPosDist = self.getPositionDistribution(gameState, prvPositions)
334             for newPosition, probability in newPosDist.items():
335                 result[newPosition] += beliefs[prvPositions] * probability
336         self.beliefs = result
```

توضیحات : دراین تابع قصد داریم با گذشت یک واحد زمانی احتمال وجود روح درهریک ازlegalPosition هارا update کنیم . هربارفرض می کنیم روح به طورقطعی دریکی ازlegalPosition ها هست واحتمال وجود روح درهریک ازlegalPosition هارا با استفاده از تابع getPositionDistribution در زمان جدید بدست می آوریم .حال مقدار احتمال وجود روح درهریک ازموقعیت ها را با حاصل ضرب احتمال بدست آمده واحتمال واقعی وجود روح در موقعیت اولیه ای که درنظرگرفته بودیم، جمع می کنیم(به عنوان مثال اگراحتمال وجود روح دریک موقعیت ۰/۶ است وما با فرض اینکه روح درهمین موقعیت است مقداراحتمال موقعیت سمت راست آن را ۰/۲ پیش بینی کرده ایم، می توان گفت روح درزمان بعدی با احتمال ۰/۱۲ درآن موقعیت سمت راستی است).

خروجی تست ها

```
Starting on 1-25 at 12:42:56

Question q3
===========
*** q3) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases\q3\1-ExactPredict.test
*** q3) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases\q3\2-ExactPredict.test
*** q3) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases\q3\3-ExactPredict.test
*** q3) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases\q3\4-ExactPredict.test

### Question q3: 3/3 ###


Finished at 12:43:04

Provisional grades
==================
Question q3: 3/3
------------------
Total: 3/3
```

۴) استنتاج دقیق با تست کامل

```python
def chooseAction(self, gameState):
    """
    First computes the most likely position of each ghost that has
    not yet been captured, then chooses an action that brings
    Pacman closest to the closest ghost (according to mazeDistance!).
    """
    pacmanPosition = gameState.getPacmanPosition()
    legal = [a for a in gameState.getLegalPacmanActions()]
    livingGhosts = gameState.getLivingGhosts()
    livingGhostPositionDistributions = \
        [beliefs for i, beliefs in enumerate(self.ghostBeliefs)
         if livingGhosts[i + 1]]
    "*** YOUR CODE HERE ***"
    minimumDistanceToGhosts = float('inf')
    closestPosition = None
    for ghostPositionDistribution in livingGhostPositionDistributions:
        maxProbablePosition = ghostPositionDistribution.argMax()
        distance = self.distancer.getDistance(pacmanPosition, maxProbablePosition)
        if minimumDistanceToGhosts == float('inf') or distance < minimumDistanceToGhosts:
            minimumDistanceToGhosts = distance
            closestPosition = maxProbablePosition

    minimumDistanceToClosestGhost = float('inf')
    bestAction = None
    for action in legal:
        newPosition = Actions.getSuccessor(pacmanPosition, action)
        distance = self.distancer.getDistance(newPosition, closestPosition)
        if minimumDistanceToClosestGhost == float('inf') or distance < minimumDistanceToClosestGhost:
            minimumDistanceToClosestGhost = distance
            bestAction = action
    return bestAction
```

توضیحات : دراین تابع قصد داریم بهترین action ای را انتخاب کنیم که پکمن را به موقعیتی دارای دوویژگی بیشترین احتمال وجود روح درآن و نزدیک ترین به پکمن ببرد.

درابتدابه ازای هریک از روح ها موقعیتی که بیشترین احتمال وجود آن درآن است را پیدا می کنیم وبین این موقعیت های با احتمال بیشینه ،موقعیتی را که به موقعیت پکمن نزدیک ترین است انتخاب می کنیم . سپس بین تمام action های مجازپکمن، action ای را که آن را به موقعیت بدست آمده درقسمت قبل بیشتر نزدیک می کند، انتخاب می کنیم.

خروجی تست ها

```
Starting on 1-25 at 13:03:31

Question q4
===========
*** q4) Exact inference full test: 0 inference errors.
*** PASS: test_cases\q4\1-ExactFull.test
*** q4) Exact inference full test: 0 inference errors.
*** PASS: test_cases\q4\2-ExactFull.test
ExactInference
[Distancer]: Switching to maze distances
Average Score: 763.3
Scores:        778, 769, 759, 761, 776, 761, 758, 753, 763, 755
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** Won 10 out of 10 games. Average score: 763.300000 ***
*** smallHunt) Games won on q4 with score above 700: 10/10
*** PASS: test_cases\q4\3-gameScoreTest.test

### Question q4: 2/2 ###


Finished at 13:03:44

Provisional grades
==================
Question q4: 2/2
------------------
Total: 2/2
```

۵) تقریب استنتاج اولیه و باور ها

```python
    def initializeUniformly(self, gameState):
        """
        Initialize a list of particles. Use self.numParticles for the number of
        particles. Use self.legalPositions for the legal board positions where
        a particle could be located. Particles should be evenly (not randomly)
        distributed across positions in order to ensure a uniform prior. Use
        self.particles for the list of particles.
        """
        self.particles = []
        "*** YOUR CODE HERE ***"
        particlesNumber = self.numParticles
        legalPositions = self.legalPositions

        for _ in range(int(particlesNumber / len(legalPositions))):
            for legalPosition in legalPositions:
                self.particles.append(legalPosition)
```

توضیحات : دراین تابع قصد داریم درابتدا توزیع particle ها بین موقعیت های مجازبه طور یکنواخت باشد . برای این کاربه عنوان مثال اگر۳۶ تا particle و ۹ تا legalPosition داشته باشیم ۴ بارمرحله زیررا تکرارمیکنیم :

دراین مرحله هریک ازlegalPosition ها رابه تعداد ۱باربه particle ها اضافه می کنیم.

```
419     def getBeliefDistribution(self):
420         """
421         Return the agent's current belief state, a distribution over ghost
422         locations conditioned on all evidence and time passage. This method
423         essentially converts a list of particles into a belief distribution.
424
425         This function should return a normalized distribution.
426         """
427         "*** YOUR CODE HERE ***"
428         distribution = DiscreteDistribution()
429         particles = self.particles
430         for particle in particles:
431             distribution[particle] += 1
432         distribution.normalize()
433         return distribution
```

توضیحات : دراین تابع قصد داریم میزان احتمال وجود روح درهریک ازlegalPosition هارا بدست آوریم .برای این کار ابتدا تعداد تکرارهرlegalPosition درparticles را بدست آورده و به کمک تابع normalize احتمال وجود روح درهریک ازآن هارا محاسبه می کنیم.

خروجی تست ها

```
Starting on 1-25 at 13:42:41

Question q5
===========
*** q5) Particle filter initialization test: 0 inference errors.
*** PASS: test_cases\q5\1-ParticleInit.test

### Question q5: 2/2 ###


Finished at 13:42:46

Provisional grades
==================
Question q5: 2/2
------------------
Total: 2/2
```

۶) مشاهده استنتاج تقریبی

```python
def observeUpdate(self, observation, gameState):
    """
    Update beliefs based on the distance observation and Pacman's position.

    The observation is the noisy Manhattan distance to the ghost you are
    tracking.

    There is one special case that a correct implementation must handle.
    When all particles receive zero weight, the list of particles should
    be reinitialized by calling initializeUniformly. The total method of
    the DiscreteDistribution may be useful.
    """
    "*** YOUR CODE HERE ***"
    distribution = DiscreteDistribution()
    is_all_zero = True
    particles = self.particles
    numParticles = self.numParticles
    pacmanPos = gameState.getPacmanPosition()
    jailPos = self.getJailPosition()

    for particle in particles:
        distribution[particle] += self.getObservationProb(observation, pacmanPos, particle, jailPos)
    values = distribution.values()
    for value in values:
        if value != 0:
            is_all_zero = False

    if not is_all_zero:
        self.particles = [distribution.sample() for none in range(0, numParticles)]
    else:
        self.initializeUniformly(gameState)
```

توضیحات : دراین تابع قصد داریم باتوجه به observation (noisyDistance) ای که کسب شده است ، particle هارا update کنیم .درابتدا برای هریک ازlegalPostion ها احتمال وجود روح درآن legalPosition رابا استفاده از تابع getObservationProb بدست می آوریم . سپس اگر احتمال جدید وجود روح درهریک ازاین legalPosition ها ۰ بود particleها را از نو مقداردهی اولیه می کنیم و درغیراین صورت با استفاده ازاحتمالات جدید نمونه برداری کرده ولیست جدید particle هارا بدست می آوریم.

خروجی تست ها

```
Starting on 1-25 at 13:55:11

Question q6
===========
*** q6) Particle filter observe test: 0 inference errors.
*** PASS: test_cases\q6\1-ParticleUpdate.test
*** q6) Particle filter observe test: 0 inference errors.
*** PASS: test_cases\q6\2-ParticleUpdate.test
*** q6) Particle filter observe test: 0 inference errors.
*** PASS: test_cases\q6\3-ParticleUpdate.test
*** q6) Particle filter observe test: 0 inference errors.
*** PASS: test_cases\q6\4-ParticleUpdate.test
*** q6) successfully handled all weights = 0
*** PASS: test_cases\q6\5-ParticleUpdate.test
ParticleFilter
[Distancer]: Switching to maze distances
Average Score: 190.0
Scores:        196, 195, 195, 167, 198, 187, 197, 191, 186, 188
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** Won 10 out of 10 games. Average score: 190.000000 ***
*** oneHunt) Games won on q6 with score above 100: 10/10
*** PASS: test_cases\q6\6-ParticleUpdate.test

### Question q6: 3/3 ###


Finished at 13:55:21

Provisional grades
==================
Question q6: 3/3
------------------
Total: 3/3
```

۷) استنتاج تقریبی باگذشت زمان

```python
def elapseTime(self, gameState):
    """
    Sample each particle's next state based on its current state and the
    gameState.
    """
    "*** YOUR CODE HERE ***"
    copy = self.particles.copy()
    self.particles = []
    for particle in copy:
        self.particles.append(self.getPositionDistribution(gameState, particle).sample())
```

توضیحات : دراین تابع قصد داریم particle هارا با گذشت یک واحد زمانی update کنیم . این تابع دقیقا با همان رویکرد بخش ۳ پیاده سازی شده است اما دارای تفاوت اندکی است که درادامه گفته می شود .
فرض کنید دربخش ۳ هستیم و۲تا legalPosition داریم که احتمال وجود روح درهریک از آن ها ۰/۵ است . حال اگر فرض کنیم روح قطعا در legalPosition شماره ۱ است و با توجه به این فرض احتمال وجود روح درlegalPosition ها در زمان بعد را بدست آوریم و متوجه شویم با احتمال ۰/۶روح درlegalPosition شماره ۲ و با احتمال ۰/۴ درlegalPosition شماره ۱ است به این معنا است که احتمال واقعی وجود روح درlegalPosition شماره ۲ ، ۰/۳ است . حال اگر بخواهیم این مفهوم را درقالب particle بیان کنیم می گوییم درparticle ها باید ۱۰ تا particle ریخته شود که ۳ تای آن legalPositionشماره ۲ و ۷ تای آن legalPosition شماره ۱ است . به همین شکل مراحل بالا را با فرض اینکه روح قطعا درlegalPosition شماره ۲ است تکرار می کنیم وparticle های جدید را به particle های قبلی اضافه می کنیم .

خروجی تست ها

```
Starting on 1-25 at 14:10:44

Question q7
===========
*** q7) Particle filter full test: 0 inference errors.
*** PASS: test_cases\q7\1-ParticlePredict.test
*** q7) Particle filter full test: 0 inference errors.
*** PASS: test_cases\q7\2-ParticlePredict.test
*** q7) Particle filter full test: 0 inference errors.
*** PASS: test_cases\q7\3-ParticlePredict.test
*** q7) Particle filter full test: 0 inference errors.
*** PASS: test_cases\q7\4-ParticlePredict.test
*** q7) Particle filter full test: 0 inference errors.
*** PASS: test_cases\q7\5-ParticlePredict.test
ParticleFilter
[Distancer]: Switching to maze distances
Average Score: 376.8
Scores:        378, 374, 364, 385, 383
Win Rate:      5/5 (1.00)
Record:        Win, Win, Win, Win, Win
*** Won 5 out of 5 games. Average score: 376.800000 ***
*** smallHunt) Games won on q7 with score above 300: 5/5
*** PASS: test_cases\q7\6-ParticlePredict.test

### Question q7: 3/3 ###


Finished at 14:13:04

Provisional grades
==================
Question q7: 3/3
------------------
Total: 3/3
```

۸) مشاهدات مشترک فیلترذرات

```python
def initializeUniformly(self, gameState):
    """
    Initialize particles to be consistent with a uniform prior. Particles
    should be evenly distributed across positions in order to ensure a
    uniform prior.
    """
    self.particles = []
    "*** YOUR CODE HERE ***"
    legalPos = self.legalPositions
    numGhosts = self.numGhosts
    self.particles = []

    permutations = itertools.product(legalPos, repeat=numGhosts)
    particles = list(permutations)
    random.shuffle(particles)
    self.particles = particles
```

توضیحات : دراین تابع قصد داریم درابتدا توزیع particle ها بین موقعیت های مجازبه طور یکنواخت باشد.برای اینکار ابتدا با استفاده ازتابع product تمام جایگشت های ممکن برای حضورروح ها بدست می آید سپس با استفاده ازتابع shuffle ترتیب آن را طوری تغییرمی دهیم که توزیع particle ها در موقعیت های مجاز به طور یکنواخت باشد. ودرنهایت آن را به عنوان particles های اولیه درنظرمی گیریم .

خروجی تست ها

```
Starting on 1-25 at 15:32:57

Question q8
===========
*** q8) Joint particle filter initialization test: 0 inference errors.
*** PASS: test_cases\q8\1-JointParticleInit.test

### Question q8: 1/1 ###


Finished at 15:32:57

Provisional grades
==================
Question q8: 1/1
------------------
Total: 1/1
```

۹) مشاهدات مشترک فیلترذرات(بخش دوم)

```python
def observeUpdate(self, observation, gameState):
    """
    Update beliefs based on the distance observation and Pacman's position.

    The observation is the noisy Manhattan distances to all ghosts you
    are tracking.

    There is one special case that a correct implementation must handle.
    When all particles receive zero weight, the list of particles should
    be reinitialized by calling initializeUniformly. The total method of
    the DiscreteDistribution may be useful.
    """
    "*** YOUR CODE HERE ***"
    distribution = DiscreteDistribution()
    is_all_zero = True
    particles = self.particles
    pacmanPos = gameState.getPacmanPosition()
    jailPos = self.getJailPosition(self.numGhosts)

    for particle in particles:
        probability = 1
        for i in range(self.numGhosts):
            probability *= self.getObservationProb(observation[i], pacmanPos, particle[i], jailPos)
        distribution[particle] += probability

    values = distribution.values()
    for value in values:
        if value != 0:
            is_all_zero = False
    if not is_all_zero:
        self.particles = [distribution.sample() for none in range(self.numParticles)]
    else:
        self.initializeUniformly(gameState)
```

توضیحات : دراین تابع قصد داریم باتوجه به observation (noisyDistance) ای که کسب شده است ، particle هارا update کنیم .

** noisyDistance یک چندتایی با طول تعداد روح ها است **

درابتدا احتمال وجود روح های ۱ تا n درهرlegalPosition که n تایی است را برابر حاصل ضرب احتمالات بودن هریک ازروح های اندر[i]legalPosition با درنظرگرفتن اینکه noisyDistance[i] برای آن کسب شده است، قرار می دهیم . سپس اگر احتمال جدید وجود روح ها درهریک ازاین legalPosition های nتایی ۰ بود particleها را از نو مقداردهی اولیه می کنیم و درغیراین صورت با استفاده ازاحتمالات جدید نمونه برداری کرده ولیست جدید particle هارا بدست می آوریم.

خروجی تست ها

```
Starting on 1-25 at 16:04:01

Question q9
===========
*** q9) Joint particle filter observeUpdate test: 0 inference errors.
*** PASS: test_cases\q9\1-JointParticleUpdate.test
*** q9) Joint particle filter observeUpdate test: 0 inference errors.
*** PASS: test_cases\q9\2-JointParticleUpdate.test
*** q9) successfully handled all weights = 0
*** PASS: test_cases\q9\3-JointParticleUpdate.test
*** q9) Joint particle filter observeUpdate test: 0 inference errors.
*** PASS: test_cases\q9\4-JointParticleUpdate.test

### Question q9: 3/3 ###


Finished at 16:04:14

Provisional grades
==================
Question q9: 3/3
------------------
Total: 3/3
```

۱۰)زمان سپری شده فیلترذرات مشترک وتست کامل

```
526     def elapseTime(self, gameState):
527         """
528         Sample each particle's next state based on its current state and the
529         gameState.
530         """
531         newParticles = []
532         for oldParticle in self.particles:
533             newParticle = list(oldParticle)  # A list of ghost positions
534
535             # now loop through and update each entry in newParticle...
536             "*** YOUR CODE HERE ***"
537             eParticles = enumerate(newParticle)
538             for particleX, particleY in eParticles:
539                 newParticle[particleX] = self.getPositionDistribution(gameState, oldParticle, particleX,
540                                                                       self.ghostAgents[particleX]).sample()
541
542             """*** END YOUR CODE HERE ***"""
543             newParticles.append(tuple(newParticle))
544         self.particles = newParticles
545
```

توضیحات : دراین تابع قصد داریم particle هارا با گذشت یک واحد زمانی update کنیم .
هربارفرض می کنیم روح ها به طورقطعی دریکی ازlegalPosition های n تایی هستند
واحتمال وجود روح ها درهریک ازlegalPosition هارا با استفاده از تابع
getPositionDistribution در زمان جدید بدست می آوریم . به ازای هریک ازاین
احتمالات بدست آمده درهربار sample می گیریم و همه این sample ها را به عنوان
particle جدید درنظرمی گیریم.

خروجی تست ها

```
Starting on 1-25 at 17:30:55

Question q10
============
*** q10) Joint particle filter elapseTime test: 0 inference errors.
*** PASS: test_cases\q10\1-JointParticlePredict.test
*** q10) Joint particle filter elapseTime test: 0 inference errors.
*** PASS: test_cases\q10\2-JointParticlePredict.test
*** q10) Joint particle filter elapseTime test: 0 inference errors.
*** PASS: test_cases\q10\3-JointParticleFull.test

### Question q10: 3/3 ###


Finished at 17:35:00

Provisional grades
==================
Question q10: 3/3
------------------
Total: 3/3
```