ایلیا نورانی 9931092

گزارش پروژه چهارم مبانی هوش مصنوعی

Q1 :

توضیح کد:

این متود احتمال p(noisy distance | manhatan distance) را به عنوان خروجی باز می گرداند. در ابتدا چک می کند اگر مقدار noisy distance ما none باشد (یعنی سنسور فاصله را none بر می گرداند) و همجنین روح در زندان باشد 1 و در غیر این صورت 0 بر می گرداند. از طرف دیگر اگر سنسور فاصله را none بر نگرداند و روح در زندان باشد باید 0 را خروجی دهد. در حالت های دیگر باید p(noisy distance | manhatan distance) محاسبه شود و به عنوان خروجی باز گردانده شود.

کد و خروجی آن در ادامه آمده است:

```
195     def getObservationProb(self, noisyDistance, pacmanPosition, ghostPosition, jailPosition):
196         """
197         Return the probability P(noisyDistance | pacmanPosition, ghostPosition).
198         """
199
200         if noisyDistance is None:
201
202             if ghostPosition == jailPosition:
203                 return 1
204             return 0
205
206         if noisyDistance is not None and ghostPosition == jailPosition:
207             return 0
208
209         return busters.getObservationProbability(noisyDistance, manhattanDistance(pacmanPosition, ghostPosition))
210
211         "*** YOUR CODE HERE  ---------done-------- ***"
212         raiseNotDefined()
213
```

```
D:\AUT\AI\p4\ghostbusters>python autograder.py -q q1
Starting on 1-24 at 18:29:42

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

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


Finished at 18:29:42

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

Q2 :

توضیح کد :

در این متود باید باور ها را در همه ی موقعیت ها(توسط self.allposition در دسترس می باشند و حلقه ای که در خط 323 است) بروزرسانی شوند. بروز رسانی توسط تابع self.getObservationProb انجام می شود(خط 324).در نهایت نیز مقادیر بروزرسانی شده را نرمالایز و به self.beliefs اساین می کنیم.

کد و خروجی این قسمت به صورت زیر است:

```
304     def observeUpdate(self, observation, gameState):
305         """
306         Update beliefs based on the distance observation and Pacman's position.
307
308         The observation is the noisy Manhattan distance to the ghost you are
309         tracking.
310
311         self.allPositions is a list of the possible ghost positions, including
312         the jail position. You should only consider positions that are in
313         self.allPositions.
314
315         The update model is not entirely stationary: it may depend on Pacman's
316         current position. However, this is not a problem, as Pacman's current
317         position is known.
318         """
319
320
321         all_pos = self.allPositions
322         newBeliefs = self.beliefs.copy()
323         pacmanPosition = gameState.getPacmanPosition()
324         jailPosition = self.getJailPosition()
325
326         for positions in all_pos:
327             newBeliefs[positions] = self.getObservationProb(observation, pacmanPosition, positions, jailPosition) * self.beliefs[positions]
328
329         newBeliefs.normalize()
330         self.beliefs = newBeliefs
331
332         "*** YOUR CODE HERE -----done----- ***"
333         ##raiseNotDefined()
334
```

```
D:\AUT\AI\p4\ghostbusters>python autograder.py -q q2 --no-graphics
Starting on 1-24 at 18:32:31

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 18:32:31

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

```
55      def normalize(self):
56          """
57          Normalize the distribution such that the total value of all keys sums
58          to 1. The ratio of values for all keys will remain the same. In the case
59          where the total value of the distribution is 0, do nothing.
60
61          >>> dist = DiscreteDistribution()
62          >>> dist['a'] = 1
63          >>> dist['b'] = 2
64          >>> dist['c'] = 2
65          >>> dist['d'] = 0
66          >>> dist.normalize()
67          >>> list(sorted(dist.items()))
68          [('a', 0.2), ('b', 0.4), ('c', 0.4), ('d', 0.0)]
69          >>> dist['e'] = 4
70          >>> list(sorted(dist.items()))
71          [('a', 0.2), ('b', 0.4), ('c', 0.4), ('d', 0.0), ('e', 4)]
72          >>> empty = DiscreteDistribution()
73          >>> empty.normalize()
74          >>> empty
75          {}
76          """
77
78          Total = self.total()
79          Keys = self.keys()
80
81          if Total == float(0):
82              return
83
84          for key in Keys:
85              self[key] = float(self[key]) / Total
86
87
88          "*** YOUR CODE HERE ----done---- ***"
89          ##raiseNotDefined()
```

Q3 :

توضیح کد :

ابتدا یک شی از نوع DiscreteDistribution می سازیم (discreteDist). سپس بر روی تمام موقعیت های مکانی روح حلقه می زنیم و به ازای موقعیت های جدیدی که به وجود می آیند , توزیع موقعیت جدید را بروزرسانی می کنیم:

discreteDist[newpos] += self.beliefs[oldPos] * probability

پس از بروزرسانی و اتمام حلقه , discreteDist را نرمالایز می کنیم و آن را به self.beliefs (باور ها) اساین می کنیم.

کد به همراه خروجی در ادامه آمده است:

```
336     def elapseTime(self, gameState):
337         """
338         Predict beliefs in response to a time step passing from the current
339         state.
340
341         The transition model is not entirely stationary: it may depend on
342         Pacman's current position. However, this is not a problem, as Pacman's
343         current position is known.
344         """
345
346         discreteDist = DiscreteDistribution()
347
348         for oldPos in self.allPositions:
349
350             for newPos, prob in self.getPositionDistribution(gameState, oldPos).items():
351                 discreteDist[newPos] += self.beliefs[oldPos] * prob
352
353         discreteDist.normalize()
354
355         self.beliefs = discreteDist
356
357         "*** YOUR CODE HERE  -----done----- ***"
358         ##raiseNotDefined()
359
360     def getBeliefDistribution(self):
361         return self.beliefs
362
363
```

```
D:\AUT\AI\p4\ghostbusters>python autograder.py -q q3 --no-graphics
Starting on 1-24 at 18:36:23

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 18:36:25

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

Q4 :

توضیح کد :

در خط 140 تا 145 ابتدا موقعیت پکمن و حرکات مجاز آن و سپس روح های زنده و توزیع باورهای موقعیت (خط 144) را برای آن ها به صورت لیست ذخیره می کند.

از خط 147 تا 153 با استفاده از توزیع باور ها , داخل یک حلقه برای هر موقعیت فاصله روح تا پکمن را حساب می کند و در صورتی که صفر (در پیمایش اول حلقه) یا کوچکتر از distance بود , فاصله را بروز رسانی می کند و نزدیک ترین موقعیت (closestPos) را موقعیت آن روح قرار می دهد.

در حلقه for دوم بر روی حرکت های مجاز پکمن یک پیمایش انجام می شود و حرکت متناظر برای رسیدن به نزدیک ترین روح(کوتاهترین فاصله) به عنوان خروجی تعیین می شود و در نهایت آن حرکت توسط تابع باز گردانده می شود.

کد و خروجی به صورت زیر است :

```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]]

        distance = 0
        closestPos = None
        for position in livingGhostPositionDistributions:
            if distance == 0 or self.distancer.getDistance(pacmanPosition, position.argMax()) < distance:
                closestPos = position.argMax()
                distance = self.distancer.getDistance(pacmanPosition, position.argMax())

        ret = None
        for action in legal:
            if self.distancer.getDistance(Actions.getSuccessor(pacmanPosition, action), closestPos) <= distance:
                ret = action
                distance = self.distancer.getDistance(Actions.getSuccessor(pacmanPosition, action), closestPos)
        return ret

        "*** YOUR CODE HERE  ----done---- ***"

```

```
D:\AUT\AI\p4\ghostbusters>python autograder.py -q q4 --no-graphics
Starting on 1-24 at 18:39:08

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: 763.1
Scores:        778, 765, 758, 760, 746, 763, 769, 780, 752, 760
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: 763.100000 ***
*** smallHunt) Games won on q4 with score above 700: 10/10
*** PASS: test_cases\q4\3-gameScoreTest.test

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


Finished at 18:39:12

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

:Q5

توضیح کد:

در ابتدا یک لیست از ذرات درست می کنیم و تعداد ذرات به همراه موقعیت های مجاز را ذخیره می کنیم.

بعد از آن توسط یک حلقه تو در تو (در مجموع به اندازه particle number) نمونه هایی که از legalPos گرفته شده است را به self.particles اضافه می کنیم.گ

- همانطور که در صورت سوال گفته شد برای پیاده سازی از لیست استفاده شده است.

کد به همراه خروجی در ادامه آمده است :

```
375     def initializeUniformly(self, gameState):
376         """
377         Initialize a list of particles. Use self.numParticles for the number of
378         particles. Use self.legalPositions for the legal board positions where
379         a particle could be located. Particles should be evenly (not randomly)
380         distributed across positions in order to ensure a uniform prior. Use
381         self.particles for the list of particles.
382         """
383         self.particles = []
384 
385         particleNumber = self.numParticles
386         legalPos = self.legalPositions
387 
388         for num in range(int(particleNumber / len(legalPos))):
389             for sample in legalPos:
390                 self.particles.append(sample)
391 
392         "*** YOUR CODE HERE  -----done----- ***"
393         #raiseNotDefined()
394 
```

```
458     def getBeliefDistribution(self):
459         """
460         Return the agent's current belief state, a distribution over ghost
461         locations conditioned on all evidence and time passage. This method
462         essentially converts a list of particles into a belief distribution.
463 
464         This function should return a normalized distribution.
465         """
466 
467         dist = DiscreteDistribution()
468         particles = self.particles
469         for particle in particles:
470             dist[particle] += 1
471         dist.normalize()
472 
473         return dist
474 
475         "*** YOUR CODE HERE  ----done---- ***"
476         ##raiseNotDefined()
477 
```

```
D:\AUT\AI\p4\ghostbusters>python autograder.py -q q5 --no-graphics
Starting on 1-24 at 18:42:22

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

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


Finished at 18:42:22

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

Q6 :

توضیح کد :

ابتدا یک شی از نوع DiscreteDistriution می سازیم. همچنین یک متغیر تعریف می کنیم (zero) برای اینکه چک کنیم تمام مقادیر صفر هستند یا نه(مقدار اولیه = true). همچنین بقیه متغیر های مورد نیاز مانند particles که ذرات را ذخیره می کند و ... را مقدار دهی اولیه می کنیم.

در خط 413 تا 419 وزن های ذرات که برابر با احتمال آنها می باشد را به Dist[particle] اضافه می کنیم و value آن ها در متغیر distVal ذخیره می شود. اگر یکی از value ها مخالف صفر باشد یعنی حالتی که همه ارزش ها برابر صفر هستند رخ نداده و نیاز به استفاده از initializeUniformly نداریم و خط 429 به جای 426 اجرا می شود (zero = false). در غیر این صورت خط 426 اجرا می شود.

در خط 429 از از توزیع وزنی نمونه برداری می شود تا لیست جدید ساخته شود.

کد و خروجی به شکل زیر است:

```
395     def observeUpdate(self, observation, gameState):
396         """
397         Update beliefs based on the distance observation and Pacman's position.
398
399         The observation is the noisy Manhattan distance to the ghost you are
400         tracking.
401
402         There is one special case that a correct implementation must handle.
403         When all particles receive zero weight, the list of particles should
404         be reinitialized by calling initializeUniformly. The total method of
405         the DiscreteDistribution may be useful.
406         """
407
408         dist = DiscreteDistribution()
409         zero = True
410         particles = self.particles
411         particleNumber = self.numParticles
412         pacmanPosition = gameState.getPacmanPosition()
413         jailPosition = self.getJailPosition()
414
415         for particle in particles:
416
417             dist[particle] += self.getObservationProb(observation, pacmanPosition, particle, jailPosition)
418
419
420         distVal = dist.values()
421
422         for val in distVal:
423             if val != 0:
424                 zero = False
425
426         if zero:
427             self.initializeUniformly(gameState)
428
429         else:
430             self.particles = [dist.sample() for none in range(0, particleNumber)]
431
432
433
434         "*** YOUR CODE HERE -----done----- ***"
435         #raiseNotDefined()
436
```

```python
    def sample(self):
        """
        Draw a random sample from the distribution and return the key, weighted
        by the values associated with each key.

        >>> dist = DiscreteDistribution()
        >>> dist['a'] = 1
        >>> dist['b'] = 2
        >>> dist['c'] = 2
        >>> dist['d'] = 0
        >>> N = 100000.0
        >>> samples = [dist.sample() for _ in range(int(N))]
        >>> round(samples.count('a') * 1.0/N, 1)  # proportion of 'a'
        0.2
        >>> round(samples.count('b') * 1.0/N, 1)
        0.4
        >>> round(samples.count('c') * 1.0/N, 1)
        0.4
        >>> round(samples.count('d') * 1.0/N, 1)
        0.0
        """


        Total = self.total()
        Items = self.items()

        example = 0

        randomVariable = random.uniform(example, Total)

        for key, value in Items:

            example += value
            if example >= randomVariable:
                return key


        "*** YOUR CODE HERE  ----done-----***"
```

```
D:\AUT\AI\p4\ghostbusters>python autograder.py -q q6 --no-graphics
Starting on 1-24 at 18:45:03

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: 188.1
Scores:        185, 186, 197, 187, 164, 191, 196, 185, 194, 196
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: 188.100000 ***
*** oneHunt) Games won on q6 with score above 100: 10/10
*** PASS: test_cases\q6\6-ParticleUpdate.test

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


Finished at 18:45:07

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

Q7:

توضیح کد :

یک شی از نوع DiscreteDist تعریف می کنیم و بر روی self.particles (ذرات) پیمایش انجام می دهیم.

سپس در خط 447 توزیع موقعیت های جدید روح را در newPosDist ذخیره می کنیم و به discreteDist اضافه می کنیم .

در نهایت این لیست جدید ذرات را در self.particles قرار می دهیم.

کد به همراه خروجی در ادامه آمده است:

```
437     def elapseTime(self, gameState):
438         """
439         Sample each particle's next state based on its current state and the
440         gameState.
441         """
442
443
444         discreteDist = []
445
446         for i, oldPos in enumerate(self.particles):
447
448             newPosDist = self.getPositionDistribution(gameState, oldPos).sample()
449             discreteDist.append(newPosDist)
450
451         self.particles = discreteDist
452
453
454         "*** YOUR CODE HERE ----done---- ***"
455         ##raiseNotDefined()
456
```

```
D:\AUT\AI\p4\ghostbusters>python autograder.py -q q7 --no-graphics
Starting on 1-24 at 18:48:29

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: 377.0
Scores:        389, 380, 380, 369, 367
Win Rate:      5/5 (1.00)
Record:        Win, Win, Win, Win, Win
*** Won 5 out of 5 games. Average score: 377.000000 ***
*** smallHunt) Games won on q7 with score above 300: 5/5
*** PASS: test_cases\q7\6-ParticlePredict.test

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


Finished at 18:49:09

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

Q8:

توضیح کد:

در خطوط 501 تا 504 همانند بخش های قبل متغیر های مورد نیاز مقدار دهی اولیه می شوند.

در خطوط 506 تا 509 ابتدا یک جایگشت (permutation) از موقعیت روح ها درست می شود و بعد از آن لیست آن جایگشت به particles اساین می شود. در نهایت آن لیست را shuffle می کنیم تا ذرات در صفحه به صورت یکنواخت قرار نگیرند و لیست جدید را در self.particles ذخیره می کنیم.

کد به همراه خروجی در ادامه آمده است:

```
496     def initializeUniformly(self, gameState):
497         """
498         Initialize particles to be consistent with a uniform prior. Particles
499         should be evenly distributed across positions in order to ensure a
500         uniform prior.
501         """
502 
503         legal = self.legalPositions
504         ghostsNum = self.numGhosts
505         self.particles = []
506 
507         perm = itertools.product(legal, repeat=ghostsNum)
508         particles = list(perm)
509         random.shuffle(particles)
510         self.particles = particles
511 
512         "*** YOUR CODE HERE -----done----- ***"
513         ##raiseNotDefined()
514 
```

```
D:\AUT\AI\p4\ghostbusters>python autograder.py -q q8 --no-graphics
Starting on 1-24 at 18:49:56

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

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


Finished at 18:49:56

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

Q9 :

توضیح کد :

مشابه متود observeUpdate در بخش های قبل است با این تفاوت که بروزرسانی در این بخش بر اساس فواصل روح ها است. یعنی لیست ذرات بر اساس مشاهده تمام فواصل روح ها وزن می شوند و نمونه گیری انجام می شود.

خط 544 تا 550 مانند بخش قبل است. در حلقه اول یک متغیر prob(احتمال) تعریف می شود و مقدار یک به آن اساین می شود. سپس در خط 555 تا 557 وزن ذرات بر اساس فاصله آن ها تغییر می کند و بروز رسانی می شود.

بقیه کد نیز مانند بخش قبل است (حالت صفر بودن مقادیر پیاده سازی شده و در نهایت در صورت false بودن zero نمونه گیری انجام می شود).

کد و خروجی به شکل زیر است:

```
    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.
        """

        dist = DiscreteDistribution()
        zero = True
        particles = self.particles
        pacmanPosition = gameState.getPacmanPosition()
        jailPosition = self.getJailPosition(self.numGhosts)

        for particle in particles:

            prob = 1.

            for i in range(self.numGhosts):
                prob = prob * self.getObservationProb(observation[i], pacmanPosition, particle[i], jailPosition)
            dist[particle] = dist[particle] + prob

        beliefVal = dist.values()

        for value in beliefVal:
            if value != 0:
                zero = False

        if zero:
            self.initializeUniformly(gameState)

        else:
            self.particles = [dist.sample() for none in range(self.numParticles)]

        "*** YOUR CODE HERE -----done----- ***"
        ##raiseNotDefined()
```

```
D:\AUT\AI\p4\ghostbusters>python autograder.py -q q9 --no-graphics
Starting on 1-24 at 18:52:26

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 18:52:31

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

:Q10

توضیح کد :

در ابتدا یک لیست برای ذخیره موقعیت روح ها تعریف می کنیم. سپس موقعیت های فعلی روح را را داخل لیست newPart ذخیره می کنیم(این لیست موقتی است و بروز رسانی می شود).در خطوط 589 و 590 توزیع را بر روی موقعیت های جدید روح محاسبه می کند (با استفاده از موقعیت های قبلی روح ها) و بعد از اتمام حلقه لیست newParts را به newParts اضافه می کند. در نهایت newParts را در self.particles ذخیره می کند.

کد و خروجی در ادامه آمده است:

```
575     def elapseTime(self, gameState):
576         """
577         Sample each particle's next state based on its current state and the
578         gameState.
579         """
580         newParts = []
581         for prevGhostPositions in self.particles:
582             newPart = list(prevGhostPositions)  # A list of ghost positions
583 
584             # now loop through and update each entry in newParticle...
585 
586 
587             "*** YOUR CODE HERE ***"
588 
589             particles = enumerate(newPart)
590             for i, particle2 in particles:
591                 newPart[i] = self.getPositionDistribution(gameState, prevGhostPositions, i, self.ghostAgents[i]).sample()
592 
593             #raiseNotDefined()
594 
595             """*** END YOUR CODE HERE  ----done----- ***"""
596             newParts.append(tuple(newPart))
597         self.particles = newParts
598 
599 
600 # One JointInference module is shared globally across instances of MarginalInference
601 jointInference = JointParticleFilter()
602 
```

```
D:\AUT\AI\p4\ghostbusters>python autograder.py -q q10 --no-graphics
Starting on 1-24 at 18:54:10

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 18:54:55

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