# HW4

R12944013 王冠程

**Show your autograder results and describe the implementation details:**

**Autograder results:**

**Q1:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q1
Starting on 5-8 at 17:06:42

Question q1
===========
*** PASS: test_cases/q1/1-small-board.test
*** PASS: test_cases/q1/2-long-bottom.test
*** PASS: test_cases/q1/3-wide-inverted.test

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


Finished at 17:06:42

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

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Q2:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q2
Starting on 5-8 at 17:08:03

Question q2
===========
*** PASS: test_cases/q2/1-product-rule.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q2/2-product-rule-extended.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q2/3-disjoint-right.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q2/4-common-right.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q2/5-grade-join.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q2/6-product-rule-nonsingleton-var.test
***     Executed FactorEqualityTest

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


Finished at 17:08:03

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

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Q3:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q3
Starting on 5-8 at 17:08:22

Question q3
===========
*** PASS: test_cases/q3/1-simple-eliminate.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q3/2-simple-eliminate-extended.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q3/3-eliminate-conditioned.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q3/4-grade-eliminate.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q3/5-simple-eliminate-nonsingleton-var.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q3/6-simple-eliminate-int.test
***     Executed FactorEqualityTest

### Question q3: 2/2 ###


Finished at 17:08:22

Provisional grades
==================
Question q3: 2/2
------------------
Total: 2/2

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Q4:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q4
Starting on 5-8 at 17:08:34

Question q4
===========
*** PASS: test_cases/q4/1-disconnected-eliminate.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q4/2-independent-eliminate.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q4/3-independent-eliminate-extended.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q4/4-common-effect-eliminate.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q4/5-grade-var-elim.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q4/6-large-bayesNet-elim.test
***     Executed FactorEqualityTest

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


Finished at 17:08:34

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

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Q5:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q5
Starting on 5-8 at 17:08:44

Question q5
===========
*** PASS: test_cases/q5/1-DiscreteDist.test
***     PASS
*** PASS: test_cases/q5/1-DiscreteDist-a1.test
***     PASS
*** PASS: test_cases/q5/1-ObsProb.test
***     PASS

### Question q5: 1/1 ###


Finished at 17:08:44

Provisional grades
==================
Question q5: 1/1
------------------
Total: 1/1

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Q6:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q6 --no-graphics
Starting on 5-8 at 17:09:11

Question q6
===========
*** q6) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases/q6/1-ExactUpdate.test
*** q6) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases/q6/2-ExactUpdate.test
*** q6) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases/q6/3-ExactUpdate.test
*** q6) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases/q6/4-ExactUpdate.test

### Question q6: 2/2 ###


Finished at 17:09:11

Provisional grades
==================
Question q6: 2/2
-----------------
Total: 2/2

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Q7:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q7 --no-graphics
Starting on 5-8 at 17:09:33

Question q7
===========
*** q7) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases/q7/1-ExactPredict.test
*** q7) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases/q7/2-ExactPredict.test
*** q7) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases/q7/3-ExactPredict.test
*** q7) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases/q7/4-ExactPredict.test

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


Finished at 17:09:34

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

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Q8:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q8 --no-graphics
Starting on 5-8 at 17:09:53

Question q8
===========
*** q8) Exact inference full test: 0 inference errors.
*** PASS: test_cases/q8/1-ExactFull.test
*** q8) Exact inference full test: 0 inference errors.
*** PASS: test_cases/q8/2-ExactFull.test
ExactInference
[Distancer]: Switching to maze distances
Average Score: 747.2
Scores:        740, 739, 765, 725, 766, 750, 721, 749, 754, 763
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: 747.200000 ***
*** smallHunt) Games won on q8 with score above 700: 10/10
*** PASS: test_cases/q8/3-gameScoreTest.test

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


Finished at 17:09:55

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

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Q9:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q9 --no-graphics
Starting on 5-8 at 17:10:18

Question q9
===========
*** q9) Particle filter initialization test: 0 inference errors.
*** PASS: test_cases/q9/1-ParticleInit.test
*** q9) numParticles initialization test: 0 inference errors.
*** PASS: test_cases/q9/2-ParticleInit.test

### Question q9: 1/1 ###


Finished at 17:10:18

Provisional grades
==================
Question q9: 1/1
------------------
Total: 1/1

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Q10:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q10 --no-graphics
Starting on 5-8 at 17:10:29

Question q10
============
*** q10) Particle filter observe test: 0 inference errors.
*** PASS: test_cases/q10/1-ParticleUpdate.test
*** q10) Particle filter observe test: 0 inference errors.
*** PASS: test_cases/q10/2-ParticleUpdate.test
*** q10) Particle filter observe test: 0 inference errors.
*** PASS: test_cases/q10/3-ParticleUpdate.test
*** q10) Particle filter observe test: 0 inference errors.
*** PASS: test_cases/q10/4-ParticleUpdate.test
*** q10) successfully handled all weights = 0
*** PASS: test_cases/q10/5-ParticleUpdate.test
ParticleFilter
[Distancer]: Switching to maze distances
Average Score: 165.2
Scores:        178, 198, 161, 195, 189, 149, 165, 159, 158, 100
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: 165.200000 ***
*** oneHunt) Games won on q10 with score above 100: 10/10
*** PASS: test_cases/q10/6-ParticleUpdate.test

### Question q10: 2/2 ###


Finished at 17:10:31

Provisional grades
==================
Question q10: 2/2
------------------
Total: 2/2

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Q11:**

```
kcwang@506-B04-R12944013:/data/master1_2/AI/AI2024-hw4$ python autograder.py -q q11 --no-graphics
Starting on 5-8 at 17:10:47

Question q11
============
*** q11) Particle filter full test: 0 inference errors.
*** PASS: test_cases/q11/1-ParticlePredict.test
*** q11) Particle filter full test: 0 inference errors.
*** PASS: test_cases/q11/2-ParticlePredict.test
*** q11) Particle filter full test: 0 inference errors.
*** PASS: test_cases/q11/3-ParticlePredict.test
*** q11) Particle filter full test: 0 inference errors.
*** PASS: test_cases/q11/4-ParticlePredict.test
*** q11) Particle filter full test: 0 inference errors.
*** PASS: test_cases/q11/5-ParticlePredict.test
ParticleFilter
[Distancer]: Switching to maze distances
Average Score: 366.6
Scores:        369, 380, 373, 364, 347
Win Rate:      5/5 (1.00)
Record:        Win, Win, Win, Win, Win
*** Won 5 out of 5 games. Average score: 366.600000 ***
*** smallHunt) Games won on q11 with score above 300: 5/5
*** PASS: test_cases/q11/6-ParticlePredict.test

### Question q11: 2/2 ###


Finished at 17:10:57

Provisional grades
==================
Question q11: 2/2
------------------
Total: 2/2

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.
```

**Total:**

```
Provisional grades
==================
Question q1: 2/2
Question q2: 3/3
Question q3: 2/2
Question q4: 2/2
Question q5: 1/1
Question q6: 2/2
Question q7: 2/2
Question q8: 1/1
Question q9: 1/1
Question q10: 2/2
Question q11: 2/2
------------------
Total: 20/20
```

**Implementation details:**

**Q1. Bayes Net Structure**

We just need to constructs an empty Bayes Net with the structure described in Figure 1, which includes the components listed below:

**Variables**:[Pacman, Ghost0, Ghost1, Observation0, Observation1]

**Edges**:[(Ghost0, Observation0), (Pacman, Observation0),(Pacman, Observation1), (Ghost1, Observation1) ]

**Variable Domains**: It's a dictionary that contains 5 elements (Pacman, Ghost0, Ghost1, Observation0, Observation1)

For the key Pacman, Ghost0, Ghost1, it contains all the possible positions they can be.

For the key Observation0, Observation1, it contains Manhattan distances of Pacman to ghosts ± noise, and are non-negative.

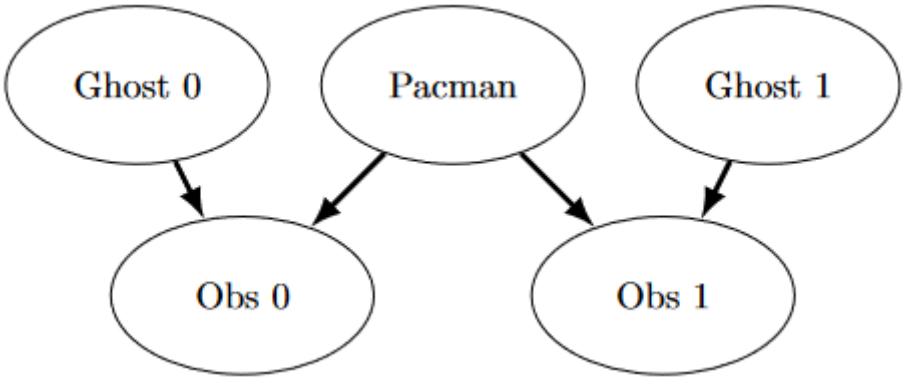


Figure 1

**Q2. Join Factors**

This function combines multiple probabilistic factors into a single factor, representing the joint probability distribution by multiplying corresponding probabilities. This function is crucial for expanding conditional probability distributions into joint distributions, allowing the incorporation of multiple conditional relationships into a unified probability model. It also ensures that unconditioned variables appear in only one input factor to maintain well-defined joint distributions.

This function initializes sets for unconditioned and conditioned variables, and assumes a shared variableDomainsDict from the first factor(Since all the input Factors have come from the same BayesNet, their variableDomainsDicts are all the same.). It iteratively updates the unconditioned and conditioned sets based on the properties of each factor. And, removing all the unconditioned variables from conditioned variables.Then, it creates a new factor with these variables and calculates the joint probability for every possible assignment by multiplying the probabilities from all factors involved. This computed factor is then returned.

**Q3. Eliminate (not ghosts yet)**

This function is to remove a given variable from a Factor. It takes a Factor and a variable to be eliminated as inputs, and outputs a new Factor that excludes the specified variable. The process involves summing up all entries in the Factor that only differ in the value of the variable being eliminated.

This function gets the value of unconditioned, conditioned variables and variableDomainsDict of input factor. And, remove the eliminationVariable from the unconditioned variable.Then, use these variables to create the new Factor. The function iterates over all possible assignments of variables, computing the sum of probabilities for each assignment where the elimination variable takes on all possible values within its domain. These probabilities are then set in the new Factor, which represents the probability distribution with the specified variable removed, maintaining the context of the other variables.

**Q4. Variable Elimination**

The inferenceByVariableElimination function is to calculate the conditional probability distribution of specified query variables given some evidence, using a sequence of interleaved join and elimination operations based on a specified order of variables (elimination order). The process begins by joining all factors that contain a specific variable and then eliminating that variable. This process is repeated for each variable in the elimination order until only the query and evidence variables remain. The final result should represent the normalized conditional probability P(queryVariables | evidenceDict).

The function first retrieves all relevant conditional probability tables (CPTs) with the provided evidence from the Bayesian network. It then iterates through the specified elimination order, joining factors that involve each variable and subsequently eliminating that variable. The function checks to ensure that any factor we are about to eliminate contains more than one unconditioned variable and discards the factor that has only one unconditioned variable. After processing all variables in the elimination order, it joins any remaining factors into a single factor and normalizes this factor to ensure that the probabilities sum to one. This ensures the result is a valid conditional probability distribution.

**Q5a. DiscreteDistribution Class**

This class is designed to effectively handle and adjust probability distributions across discrete elements, which are used to represent possible states or observations within probabilistic models. The normalize method computes the overall sum of the values in the distribution, and if this sum is non-zero, it normalizes each value by dividing it by the total sum to ensure the distribution totals to one; if the sum is zero or the distribution is empty, it leaves the distribution unchanged to prevent division by zero. The sample method selects an element randomly from the distribution by generating a random number up to the total sum of the values, then iterating over the elements and accumulating their weights until this

accumulated sum surpasses the generated random number, at which point the current element is chosen as the sample.

**Q5b. Observation Probability**

The primary objective of this function is to compute the probability P(noisyDistance | pacmanPosition,ghostPosition), which assesses the probability of observing a certain noisy distance in Pacman's sensor, based on the actual positions of Pacman and the ghost. This calculation is crucial for evaluating how probable a specific distance measurement is under the actual game conditions. Initially, the function checks if the ghost is imprisoned; if it is, the function returns 1.0 if the noisyDistance is None, otherwise it returns 0.0. Similarly, if the noisyDistance is None, the function returns 0.0. The function then determines the true distance between Pacman and the ghost using the manhattanDistance function. Lastly, it employs the busters.getObservationProbability method to calculate and return the probability.

**Q6. Exact Inference Observation**

The observeUpdate method is designed to update the agent's beliefs about ghost positions based on new observations from Pacman's sensors. This update is crucial for maintaining an accurate belief distribution across all potential ghost positions on the game map, which includes both legal positions and a special jail position. The beliefs are stored as a DiscreteDistribution object within self.beliefs. The method leverages the self.getObservationProb function to calculate the probability of an observation given the current positions of Pacman and a potential ghost position, as well as the jail position, and updates the belief for each position accordingly.

This function first retrieves the jail position and Pacman's current position. It then iterates over all possible positions for the ghost as listed in self.allPositions. For each position, it calculates the observation probability using the provided noisy Manhattan distance to the ghost, Pacman's position, and the jail position. This probability is then used to update the belief at each position by multiplying the current belief value by the computed probability. After iterating through all positions, the method normalizes the belief distribution to ensure that the total probability across all positions sums to one, thus maintaining the distribution as a proper probability distribution.

**Q7. Exact Inference with Time Elapse**

The elapseTime method is designed to update the belief distribution for ghost positions over time, taking into account the ghosts' possible movements. This method helps to predict where ghosts might be based on their known movement patterns and the constraints of the game map, such as walls and the limitation that ghosts cannot move more than one space per time step.

To implement this, first create a new instance of a DiscreteDistribution object named newDist. Loop through each of the old positions, then fetch the probability distribution for possible new positions based on each old position. Use this distribution to update the new beliefs, incorporating the probabilities from the previous position's beliefs. After processing all positions, set self.beliefs to this updated newPosDist to reflect the adjusted understanding of ghost positions.

**Q8. Exact Inference Full Test**

The chooseAction method is to help Pacman make strategic moves towards ghosts based on his current belief about their positions. After updating his beliefs through observing and accounting for the passage of time, Pacman uses this method to determine the most likely positions of each ghost that remains uncaptured. From these positions, he chooses an action that will minimize the distance to the nearest ghost.

The method first identifies the most likely position of each ghost by utilizing the maximum value from the belief distributions of the uncaptured ghosts. It then evaluates all legal actions Pacman can take from his current position. For each action, the method calculates the successor position (where Pacman would end up if he took that action) and determines the maze distance from this new position to the closest ghost's most likely position. The action that results in the shortest distance to any ghost is selected as the best move for Pacman, optimizing his approach to closing in on and capturing the ghosts. Finally, we return this action.

**Q9. Approximate Inference Initialization and Beliefs**

This function is designed to track a ghost's location within a game environment using a collection of particles, where each particle signifies a potential ghost location. The class includes methods to uniformly distribute these particles across all legal positions for initial setup and to transform this particle distribution into a belief distribution, indicating the likelihood of the ghost occupying various locations.

The **initializeUniformly** function distributes a specified number of particles uniformly across all legal positions on the game board, ensuring a uniform prior distribution of beliefs. It does so by repeatedly adding legal positions to the particle list until all particles are assigned. If the number of particles exceeds the number of legal positions, it cycles through the legal positions again until all particles are placed. The **getBeliefDistribution** function then converts this list of particles into a DiscreteDistribution object by counting the occurrences of each position among the particles. This count represents the belief strength that the ghost is at each position. The distribution is then normalized to ensure that the sum of probabilities across all positions equals one, reflecting a valid probability distribution.

**Q10. Approximate Inference Observation**

The observeUpdate method is to update the belief distribution of ghost positions based on new sensory observations from Pacman, specifically the noisy Manhattan distance from Pacman to the ghost. This method adjusts the weight of each particle based on the likelihood of the observed distance given the particle's position relative to Pacman and the jail position. After recalculating these weight, the method resamples the particles from this updated distribution to reflect the new information. A special condition handled by this method is when all particle probability sum to zero after the update. In such cases, the particle list is reinitialized uniformly to reset the belief distribution.

In the provided implementation, each particle's weight is updated by calculating the observation probability that connects Pacman's position, the particle's position, and the jail position. These probabilities are aggregated into a DiscreteDistribution, which is then normalized to form a probability distribution. If the sum of all weights in the distribution is zero (indicating that no existing particle configuration is plausible under the new observation), the particles are uniformly reinitialized across all legal positions to restore a balanced distribution. Finally, the method resamples from this updated distribution to

reconstruct the particle list, ensuring each new particle's inclusion probability corresponds to its weight in the distribution, thereby preparing the belief state for further updates based on subsequent observations.

**Q11. Approximate Inference with Time Elapse**

The elapseTime function is designed to simulate the passage of time by updating the position of each particle in the belief distribution, effectively predicting ghost movements based on their previous positions. This function iterates through the current list of particles, using a model to determine their new positions based on possible movements allowed by the game dynamics. The aim is to maintain a dynamic and realistic representation of where the ghost could potentially be located after a time step, mirroring the capabilities of exact inference by using an approximate method that involves sampling from a probability distribution of potential new positions.

This function first calculates the current belief distribution from the existing particles and then initializes a new distribution for updating these beliefs based on possible movements. For each legal position (oldPos), it uses the getPositionDistribution method to fetch the probability distribution over new positions (newPos) that the ghost can move to from oldPos. It then updates the newDist distribution by accumulating probabilities weighted by the current beliefs about each oldPos. After forming this comprehensive new distribution, the function resamples from this updated distribution to create a new list of particles, then, store this list to self.particles.