Report

Q1.

填充為貝葉斯網絡節點至 variables`

填充為貝葉斯網絡中的每條邊 directed edges(from, to)。

將每個變量的取值範圍設置為 `variableDomainsDict[var]

PAC, GHOST0, GHOST1：

地圖上所有可能的座標對 (x, y)。

OBS0、OBS1:

觀察值是非負數。觀察值與真實值的誤差在 MAX_NOISE 之內。將觀察值的取值範圍設置為真實值可能的最大範圍，即在 MAX_NOISE 加上地圖的寬度和高度的總和減去 1。

```
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 2:31:03

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

Q2.

獲取 variableDomainsDict

建立了一個包含所有輸入因子的 unconditioned Variables 的集合 unconditionedVar

建立了一個包含所有輸入因子的 conditioned Variables 的集合 conditionedVar(要去除了已經出現在 unconditionedVar 中的變數。)

對於 newFactor 的每個可能的 PossibleAssignmentDicts‘Dict’，遍歷所有輸入因子，將每個因子進行相乘，得到該 Dict 下新因子的概率值。最後，使用 newFactor.setProbability() 方法將計算得到的概率值給 newFactor 中的 Dict。

```
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 ###
```

Q3.

在因子 P(X, Y | Z) 中，X 和 Y 是未条件变量，因为它们没有被 Z 所限制或约束。而 Z 是条件变量，因为它是在给定 Z 的条件下计算 X 和 Y 的概率。

獲取 factor 的 variableDomainsDict

建立了一個包含所有輸入因子的 unconditioned Variables 的集合 unconditionedVar，要消除的變量 eliminationVariable。

建立了一個包含所有輸入因子的 conditioned Variables 的集合 conditionedVar。

對於 newFactor 的每個 PossibleAssignmentDicts ‘Dict’ ，遍歷 variableDomainsDict[eliminationVariable] 中的每個值 elim，將 Dict 中的 eliminationVariable 的值修改為 elim。

接著，使用 factor.getProbability(Dict) 獲取修改後的 Dict 中變數的概率值，並將其加總到 prob 中。

最後，我們使用 newFactor.setProbability(Dict, prob) 將計算得到的概率值設置給新因子 newFactor 中的 PossibleAssignmentDicts。。

```
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 ###
```

Q4.

初始 currentFactorsList 為 bayesNet.getAllCPTsWithEvidence(evidenceDict)
遍歷 eliminationOrder 中的 elim，執行了 joinFactorsByVariable 函數，將所有包含該變量的因子進行聯合操作。joinFactorsByVariable 函數返回了新的未聯合的因子列表 currentFactorsNotToJoin 和聯合後得到的新因子 joinedFactor。
子 joinedFactor 是否包含多個未條件化的變量，如果是，則執行 eliminate 函數，對變量 elim 進行消除操作，然後將消除後的新因子添加到未聯合的因子列表 currentFactorsNotToJoin 中。
對新的因子列表中的所有因子進行聯合操作，然後進行標準化，得到最終的條件概率結果，並返回該結果。

```
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 ###
```

Q5.

def normalize(self):
假設 total> 0
則將每一個 item 的值進行標準化，(將該 item 的 value/total)

def sample(self):
target = random.random()
設定累積機率 cumulativeP，出始為零
遍歷所有 item:
cumulativeP 加上每一個 item 機率
如果 cumulativeP 大於 target
回傳 k
def getObservationProb：
如果鬼魂在監獄中，並且 Pacman 觀察到的距離是 None，那麼返回的概率為 1，則返回 0
如果鬼魂不在監獄中： 如果 Pacman 觀察到的距離是 None，則返回的概率為 0。 如果 Pacman 觀察到了距離，計算 Pacman 觀察到這個距離的概率。

```
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 ###
```

Q6.
保存舊的 belief distribution
獲取 pacman 的位置及監獄位置
創建新的離散分佈來存儲新的 belief distribution
根據觀察、pacman 的位置、鬼魂的位置和監獄的位置計算新的 belief distribution
P(X|e)是給定觀察 e 的情況下幽靈位於位置 X 的後驗 Belief。
P(e|X)是在幽靈位於位置 X 的情況下觀察 e 的概率。我們可以使用 self.getObservationProb()來計算這個值。
P(X)是幽靈位於位置 X 的先驗 Belief。
P(e)是觀察 e 的概率，可以通過對所有可能的幽靈位置進行求和來計算。
P(X|e)= P(e|X)P(X)/P(e)(P(e) = 1 可忽略)
將新的 belief distribution 更新到 self.beliefs 中
Normalize self.beliefs

```
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 ###
```

Q7.

建立一個空的離散分佈，用於存儲更新後對於物體位置的 Belief distribution

對於每個可能的 oldPos，從 oldPos 移動到每個可能新位置的概率分佈 newPosDist。

對於 newPosDist 中的每個新位置 newPos 和相應的概率 prob（P(x|y)），將每個新位置的 belief 值加上舊位置的 belief 值乘以 pacmn 移動到該新位置的概率 (P(x)+=P(x|y)*P(y))

$P(x)=y\sum P(x \mid y)\cdot P(y)$

$P(x)$是時間 t+1, 是幽靈的位置在 x 的 Belief distribution（更新的 belief distribution）

$P(x \mid y)$從 y 移動到 x 的過渡機率。

$P(y)$ 是時間 t時，幽靈在 y 的位置的 Belief distribution（先前的 belief distribution）

將更新後的信念分佈 newBeliefs 賦值給 self.beliefs。

Normalize self.beliefs

```
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 ###
```

Q8.

找到每個幽靈最可能的位置

計算 Pacman 與每個幽靈之間的距離

使用負數保存距離，以便後面使用 argMax 找到最小距離

找到最近的幽靈，並選擇行動以靠近它

嘗試所有可能的行動，並保存移動後的距離

使用負數保存距離，以便後面使用 argMax 找到最小距離

```
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: 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 q8 with score above 700: 10/10
*** PASS: test_cases/q8/3-gameScoreTest.test

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

Q9.

def initializeUniformly：

PositionPart 每個位置應該填充的粒子數

remainingPart 不能平均分配的剩餘粒子數。

將每個合法位置都填充 PositionPart 個粒子（[position] * PositionPart）

將剩餘的粒子添加到列表中，以保證粒子的總數是 numParticles

（legalPositions[:remainingPart]）

def getBeliefDistribution(self):

建立新的離散分佈來存儲新的 belief distribution

對以樣本為鍵的字典物件進行+1 計數，即可得到每一個樣本出現的次數

Normalize beliefs

```
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 ###
```

Q10.

創建一個空的離散分佈物件 Weight

對於每個粒子，計算觀測值給定下，粒子位置的概率，並將該概率加入到相應粒子的權重中。

對權重進行歸一化操作，使得所有粒子的權重之和為 1

如果更新後的 Weigt.total()等於 0，則重新初始化粒子，將其均勻地分佈在地圖上。

否則將粒子設置為從更新後的分佈中抽樣得到的值。

```
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: 180.2
Scores:        188, 192, 198, 186, 167, 180, 184, 187, 164, 156
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: 180.200000 ***
*** oneHunt) Games won on q10 with score above 100: 10/10
*** PASS: test_cases/q10/6-ParticleUpdate.test

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

Q11.

創建一個空列表 newParts，用於存儲新的粒子狀態。

對於 self.particles 中的每個粒子：

使用 getPositionDistribution 方法獲取給定 gameState 時粒子的下一個位置的分佈。

從這個分佈中抽取一個樣本，即下一個時間步的新位置。

將抽取的新位置加入到 newParts 中。

將新的粒子列表 newParts 賦值給 self.particles

```
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: 382.8
Scores:        386, 389, 363, 388, 388
Win Rate:      5/5 (1.00)
Record:        Win, Win, Win, Win, Win
*** Won 5 out of 5 games. Average score: 382.800000 ***
*** smallHunt) Games won on q11 with score above 300: 5/5
*** PASS: test_cases/q11/6-ParticlePredict.test

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