```
from pgmpy.models import BayesianModel
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination
```

Programming Assignment 2

Jiaqi Yang (jxy530)

November 15th, Friday

Exercise 1. Noisy Fuel Guage (From Bishop)

Consider a model of a noisy electric fuel gauge in an old and unreliable car. The state of the guage G indicates
the fuel level F. The gauge is electric, so it depends on the fuel level F and the state of the battery B. For
simplicity, assume the states are binary with 0 and 1 representing empty and full respectively for G and F or
dead and charged for B.

a) Define this Bayes net using the pgmpy toolbox. Use the following probabilities:

p(B = 1) = 0.8 (1)

p(F = 1) = 0.9 (2)

p(G = 1|B = 0,F = 0) = 0.1 (3)

p(G = 1|B = 1,F = 0) = 0.1 (4)

p(G = 1|B = 0,F = 1) = 0.2 (5)

p(G = 1|B = 1,F = 1) = 0.9 (6)

```
# Defining the model structure for problem 1
model_p1 = BayesianModel([('F', 'G'), ('B', 'G')])

# Defining CPD of fuel level F
cpd_f = TabularCPD(variable='F', variable_card=2, values=[[0.1, 0.9]],
state_names={'F':['Empty','Full']})

# Define CPD of state of battery B
cpd_b = TabularCPD(variable='B', variable_card=2, values=[[0.2, 0.8]],
state_names={'B':['Dead','Charged']})

# Define CPD of state of guage G
#    +---------+---------+---------+---------+---------+
#    | battery |battery_0|battery_0|battery_1|battery_1|
#    +---------+---------+---------+---------+---------+
#    | fuel    | fuel_0  | fuel_1  | fuel_0  | fuel_1  |
#    +---------+---------+---------+---------+---------+
#    | guage_0 | 0.9     | 0.8     | 0.9     | 0.1     |
#    +---------+---------+---------+---------+---------+
#    | guage_1 | 0.1     | 0.2     | 0.1     | 0.9     |
#    +---------+---------+---------+---------+---------+
cpd_g = TabularCPD(variable='G', variable_card=2,
                    values=[[0.9, 0.8, 0.9, 0.1],
                            [0.1, 0.2, 0.1, 0.9]],
                   evidence=['B', 'F'],
                   evidence_card=[2, 2],
                   state_names={'G':['Empty Read', 'Full Read'],
                                'B':['Dead','Charged'],
                                'F':['Empty','Full']})


# Associating the CPDs with the network
model_p1.add_cpds(cpd_f, cpd_b, cpd_g)

# check_model checks for the network structure and CPDs and verifies that the
CPDs are correctly
# defined and sum to 1.
model_p1.check_model()
```

```
True
```

```
print('CPD of state of battery B')
print(model_p1.get_cpds('B'))
print('CPD of fuel level F')
print(model_p1.get_cpds('F'))
print('CPD of state of guage G')
print(model_p1.get_cpds('G'))
```

```
CPD of state of battery B
+------------+-----+
| B(Dead)    | 0.2 |
+------------+-----+
| B(Charged) | 0.8 |
+------------+-----+
CPD of fuel level F
+----------+-----+
| F(Empty) | 0.1 |
+----------+-----+
| F(Full)  | 0.9 |
+----------+-----+
CPD of state of guage G
+---------------+----------+---------+------------+------------+
| B             | B(Dead)  | B(Dead) | B(Charged) | B(Charged) |
+---------------+----------+---------+------------+------------+
| F             | F(Empty) | F(Full) | F(Empty)   | F(Full)    |
+---------------+----------+---------+------------+------------+
| G(Empty Read) | 0.9      | 0.8     | 0.9        | 0.1        |
+---------------+----------+---------+------------+------------+
| G(Full Read)  | 0.1      | 0.2     | 0.1        | 0.9        |
+---------------+----------+---------+------------+------------+
infer_p1 = VariableElimination(model_p1)
```

b) Derive the mathmatical expression for the probability that the fuel tank is empty given that the fuel
guage reads empty. Be sure to express this in terms of probabilities that are defined for the model.

To find the probability that the fuel tank is empty (F = 0) given that the fuel guage reads empty (G = 0), which is P(F = 0|G = 0).

P(F = 0|G = 0) = P(G = 0|F = 0) P(F = 0) / P(G = 0)

First we express P(G = 0) = P(G = 0|B,F) P(B) P(F) for all possible values for B and F (which is 0 and 1):
Therefore, we get P(¬G|¬B,¬F) P(¬B) P(¬F) + P(¬G|B,¬F) P(B) P(¬F) + P(¬G|¬B,F) P(¬B) P(F) + P(¬G|B,F) P(B) P(F)
= 0.9 x 0.2 x 0.1 + 0.8 x 0.2 x 0.9 + 0.9 x 0.8 x 0.1 + 0.1 x 0.8 x 0.9
= 0.306

Second we express P(G = 0|F = 0) = P(G = 0|F = 0, B) P(B) for all possible values for B (which is 0 and 1):
Therefore, we get P(¬G|¬B,¬F) P(¬B) + P(¬G|B,¬F) P(B)
= 0.9 x 0.2 + 0.9 x 0.8
= 0.9

Lastly, we could calculate based on P(F = 0|G = 0) = P(G = 0|F = 0) P(F = 0) / P(G = 0)
= 0.9 x 0.1 / 0.306
≈ 0.29412

c) Use the toolbox to calculate the numerical value of this expression from the model you defined above.

```
f_dist_0 = infer_p1.query(['F'], evidence={'G':'Empty Read'})
print('the probability that the fuel tank is empty (F = 0) given that the fuel
guage reads empty (G = 0), which is P(F = 0|G = 0):')
print(f_dist_0)
```

```
Finding Elimination Order: : 100%|
██████████████████████████████████████████████████| 1/1 [00:00<00:00,
1004.86it/s]
Eliminating: B: 100%|
█████████████████████████████████████████████████████████████████| 1/1
[00:00<00:00, 501.65it/s]
```

```
the probability that the fuel tank is empty (F = 0) given that the fuel guage
reads empty (G = 0), which is P(F = 0|G = 0):
+----------+----------+
| F        |   phi(F) |
+==========+==========+
| F(Empty) |   0.2941 |
+----------+----------+
| F(Full)  |   0.7059 |
+----------+----------+
```

d) Show how our beliefs about the state of the fuel tank change when
1) we haven't made any observations yet;
2) we see that the fuel guage reads empty
3) we test the battery and find that it is dead.
For each case, write the probabilistic expression and derive the numerical value using the toolbox.

```python
# The first situation that we haven't made any observations yet
f_dist_1 = infer_p1.query(['F'])
print('1) we haven’t made any observations yet')
print(f_dist_1)

# The second situation that we see that fuel guage reads empty
f_dist_2 = infer_p1.query(['F'], evidence={'G':'Empty Read'})
print('2) we see that the fuel guage reads empty')
print(f_dist_2)

# The third situation that we see that battery is found dead
f_dist_3 = infer_p1.query(['F'], evidence={'B':'Dead', 'G':'Empty Read'})
print('3) we see that the fuel guage reads empty, and we test the battery and
find that it is dead')
print(f_dist_3)
```

```
Finding Elimination Order: : 100%|
██████████████████████████████████████████████████████| 2/2 [00:00<00:00,
2003.01it/s]
Eliminating: B: 100%|
████████████████████████████████████████████████████████████████████| 2/2
[00:00<00:00, 501.47it/s]
```

```
1) we haven’t made any observations yet
+----------+----------+
| F        |   phi(F) |
+==========+==========+
| F(Empty) |   0.1000 |
+----------+----------+
| F(Full)  |   0.9000 |
+----------+----------+
Finding Elimination Order: : 100%|
██████████████████████████████████████████████████████| 1/1 [00:00<00:00,
332.54it/s]
Eliminating: B: 100%|
████████████████████████████████████████████████████████████████████| 1/1
[00:00<00:00, 501.05it/s]
```

```
2) we see that the fuel guage reads empty
+----------+----------+
| F        |   phi(F) |
+==========+==========+
| F(Empty) |   0.2941 |
+----------+----------+
| F(Full)  |   0.7059 |
+----------+----------+
```

```
Finding Elimination Order: : : 0it [00:00, ?it/s]
0it [00:00, ?it/s]
```

```
3) we see that the fuel guage reads empty, and we test the battery and find that
it is dead
+----------+----------+
| F        |   phi(F) |
+==========+==========+
| F(Empty) |   0.1111 |
+----------+----------+
| F(Full)  |   0.8889 |
+----------+----------+
```

e) Explain why our belief (i.e. the probability) that the fuel tank is empty goes down when learn that the
battery is dead.

Mathematically,

P(F = 0|G = 0) = P(G = 0|F = 0) P(F = 0) / P(G = 0)

P(F = 0|G = 0, B = 0) = P(G = 0|B = 0, F =0) P(F = 0) / (P(G = 0|B = 0, F) P(F) for all possible F value, which is 0 and 1)

Where P(B = 0) is cancelled between numberator and denominator. Comparing these two expression, it is obvious that the second expression will be smaller than the first expression, given a set of fixed probabilities.

Logically,

G is common evidence of F and B. In this V structure, (F|B) and (F/B|G). Therefore, in this case with G given, F and B are not independent

When G is observed, in this case the state of guage is empty read. If we observe the battery is dead, it will decrease the chance of the empty tank: because that the reading of guage is empty due to dead battery increases.

f) Explain what our belief about the fuel tank be if we had only observed the dead battery.

```
f_dist_4 = infer_p1.query(['F'], evidence={'B':'Dead'})
print('Our belief about the fuel tank be if we had only observed the dead
battery:')
print(f_dist_4)
```

```
Finding Elimination Order: : 100%|
██████████████████████████████████████████████████████| 1/1 [00:00<00:00,
1002.22it/s]
Eliminating: G: 100%|
████████████████████████████████████████████████████████████████████| 1/1
[00:00<00:00, 501.47it/s]
```

```
Our belief about the fuel tank be if we had only observed the dead battery:
+----------+----------+
| F        |   phi(F) |
+==========+==========+
| F(Empty) |   0.1000 |
+----------+----------+
| F(Full)  |   0.9000 |
+----------+----------+
```

As discussed above, G is common evidence of F and B. In this V structure, (F|B) and (F/B|G). Therefore, in the previous case with G given, F and B are not independent. However, without G given, F and B are independent of each other: since G is not observed, any change in F will reflect in G instead of B in this case. Therefore, when we observed the battery is dead, the probability that the guage reading is empty increases; in the meanwhile, the state of fuel tank is not affected by the change.

Verified by the calculated result of the toolbox, our belief about the fuel tank given the dead battery is exactly same as the chance of full/empty fuel tank, which is (0.9, 0.1). Therefore, in this scenerio, these two variables are independent of each other.

Exercise 2. Wet Grass (from Pearl/Barber)

Now we will do an only slightly more complicated model that also illustrates the concept of “explaining away.”
Tracey leaves her house one morning and sees that the grass is wet. She wonders: Was it due to an overnight
rain? Or that last night she forgot to turn off her sprinkler?

a) Show graphical model of the Bayes net that models this scenario. Use the following variables: R = it
rained last night; S = the sprinkler was left on; and G = the grass is wet.

```
     Spinkler(S)             Rain(R)

          \                   /

            \               /

              \           /

                \       /

                  ↘   ↙

                Wet Grass(G)
```

b) Write out the probability tables for this model assuming that: there is a 20% chance of overnight rain;
a 10% chance that she left the sprinkler on; rain always makes the grass wet and if it doesn't rain, the
grass isn't wet, but the sprinkler fails 5% of the time.

```
CPD of State of Spinkler
+------------+-----+
| S(Turn-off)| 0.9 |
+------------+-----+
| S(Turn-on) | 0.1 |
+------------+-----+

CPD of State of Rain
+------------+-----+
| R(Not Rain)| 0.8 |
+------------+-----+
| R(Rain)    | 0.2 |
+------------+-----+

CPD of State of Grass
+---------------+--------------+-------------+------------+------------+
| S             | S(Turn-off)  | S(Turn-off) | S(Turn-on) | S(Turn-on) |
+---------------+--------------+-------------+------------+------------+
| R             | R(Not Rain)  |   R(Rain)   | R(Not Rain)|  R(Rain)   |
+---------------+--------------+-------------+------------+------------+
| G(Dry)        | 1            | 0           | 0.05       | 0          |
+---------------+--------------+-------------+------------+------------+
| G(Wet)        | 0            | 1           | 0.95       | 1          |
+---------------+--------------+-------------+------------+------------+
```

c) Define this model using the toolbox.

```
# Defining the model structure for problem 2
model_p2 = BayesianModel([('S', 'G'), ('R', 'G')])

# Defining CPD of state of spinkler S
cpd_s = TabularCPD(variable='S', variable_card=2, values=[[0.9, 0.1]],
state_names={'S':['Turn-off','Turn-on']})

# Define CPD of state of rain R
```

```
cpd_r = TabularCPD(variable='R', variable_card=2, values=[[0.8, 0.2]],
state_names={'R':['Not Rain','Rain']})

# Define CPD of state of grass G
# +---------------+--------------+-------------+------------+------------+
# | S             | S(Turn-off)  | S(Turn-off) | S(Turn-on) | S(Turn-on) |
# +---------------+--------------+-------------+------------+------------+
# | R             | R(Not Rain)  |   R(Rain)   | R(Not Rain)|  R(Rain)   |
# +---------------+--------------+-------------+------------+------------+
# | G(Dry)        | 1            | 0           | 0.05       | 0          |
# +---------------+--------------+-------------+------------+------------+
# | G(Wet)        | 0            | 1           | 0.95       | 1          |
# +---------------+--------------+-------------+------------+------------+
cpd_g = TabularCPD(variable='G', variable_card=2,
                   values=[[1, 0, 0.05, 0],
                           [0, 1, 0.95, 1]],
                  evidence=['S', 'R'],
                  evidence_card=[2, 2],
                  state_names={'G':['Dry', 'Wet'],
                               'S':['Turn-off','Turn-on'],
                               'R':['Not Rain','Rain']})


# Associating the CPDs with the network
model_p2.add_cpds(cpd_s, cpd_r, cpd_g)

# check_model checks for the network structure and CPDs and verifies that the
CPDs are correctly
# defined and sum to 1.
model_p2.check_model()
```

```
True
```

```
print('CPD of state of spinkler S')
print(model_p2.get_cpds('S'))
print('CPD of state of rain R')
print(model_p2.get_cpds('R'))
print('CPD of state of grass G')
print(model_p2.get_cpds('G'))
```

```
CPD of state of spinkler S
+-------------+-----+
| S(Turn-off) | 0.9 |
+-------------+-----+
| S(Turn-on)  | 0.1 |
+-------------+-----+
CPD of state of rain R
+-------------+-----+
| R(Not Rain) | 0.8 |
+-------------+-----+
| R(Rain)     | 0.2 |
+-------------+-----+
CPD of state of grass G
+--------+-------------+-------------+-------------+------------+
| S      | S(Turn-off) | S(Turn-off) | S(Turn-on)  | S(Turn-on) |
+--------+-------------+-------------+-------------+------------+
| R      | R(Not Rain) | R(Rain)     | R(Not Rain) | R(Rain)    |
+--------+-------------+-------------+-------------+------------+
| G(Dry) | 1.0         | 0.0         | 0.05        | 0.0        |
+--------+-------------+-------------+-------------+------------+
| G(Wet) | 0.0         | 1.0         | 0.95        | 1.0        |
+--------+-------------+-------------+-------------+------------+
```

d) Calculate the probabilities for Tracey's original queries using the toolbox (you do not have to provide the
mathematical derivation).

```
infer_p2 = VariableElimination(model_p2)

# Was it due to an overnight rain?
r_dist_1 = infer_p2.query(['R'], evidence={'G':'Wet'})
print('Was it due to an overnight rain?')
print(r_dist_1)

# Or that last night she forgot to turn off her sprinkler?
s_dist_1 = infer_p2.query(['S'], evidence={'G':'Wet'})
print('Or that last night she forgot to turn off her sprinkler?')
print(s_dist_1)
```

```
Finding Elimination Order: : 100%|
██████████████████████████████████████████████████████| 1/1 [00:00<00:00,
501.35it/s]
Eliminating: S: 100%|
██████████████████████████████████████████████████████████████████| 1/1
[00:00<00:00, 1002.22it/s]
```

```
Was it due to an overnight rain?
+-------------+----------+
| R           |   phi(R) |
+=============+==========+
| R(Not Rain) |   0.2754 |
+-------------+----------+
| R(Rain)     |   0.7246 |
+-------------+----------+
```

```
Finding Elimination Order: : 100%|
█████████████████████████████████████████████████████| 1/1 [00:00<00:00,
1003.18it/s]
Eliminating: R: 100%|
██████████████████████████████████████████████████████████████████| 1/1
[00:00<00:00, 1002.46it/s]
```

```
Or that last night she forgot to turn off her sprinkler?
+-------------+----------+
| S           |   phi(S) |
+=============+==========+
| S(Turn-off) |   0.6522 |
+-------------+----------+
| S(Turn-on)  |   0.3478 |
+-------------+----------+
```

e) Now suppose that Tracey observes her neighbor, Jack, also has wet grass. Augment your model to
incorporate this new information. Use the variable J to represent the state of Jack's grass. Assume that
the rain always makes Jack's grass wet, but if it didn't rain, his grass has a 15% chance of being wet
for some other reason. Write the graph for the Bayes net model, the probability tables, and the updated
code using the toolbox.

```
 Tracey's Spinkler(S)        Rain(R)

        \                   /     \

          \               /         \

            \           /             \

              \       /                 \

                ↘   ↙                     ↘

           Tracey's Wet Grass(G)       Jack's Wet Grass (J)

CPD of State of Tracey's Spinkler(S)
+------------+-----+
| S(Turn-off)| 0.9 |
+------------+-----+
| S(Turn-on) | 0.1 |
+------------+-----+

CPD of State of Rain(R)
+------------+-----+
| R(Not Rain)| 0.8 |
+------------+-----+
| R(Rain)    | 0.2 |
+------------+-----+

CPD of State of Tracey's Gras(G)
+---------------+--------------+-------------+------------+------------+
| S             | S(Turn-off)  | S(Turn-off) | S(Turn-on) | S(Turn-on) |
+---------------+--------------+-------------+------------+------------+
| R             | R(Not Rain)  |   R(Rain)   | R(Not Rain)|  R(Rain)   |
+---------------+--------------+-------------+------------+------------+
| G(Dry)        | 1            | 0           | 0.05       | 0          |
+---------------+--------------+-------------+------------+------------+
| G(Wet)        | 0            | 1           | 0.95       | 1          |
+---------------+--------------+-------------+------------+------------+

CPD of State of Jack's Grass (J)
+---------------+--------------+-------------+
| R             | R(Not Rain)  |   R(Rain)   |
+---------------+--------------+-------------+
| J(Dry)        | 0.85         | 0           |
+---------------+--------------+-------------+
| J(Wet)        | 0.15         | 1           |
+---------------+--------------+-------------+
```

```
# Defining the model structure for problem 2e
model_p2_updated = BayesianModel([('S', 'G'), ('R', 'G'), ('R', 'J')])

# Defining CPD of state of Tracey's spinkler S
cpd_s = TabularCPD(variable='S', variable_card=2, values=[[0.9, 0.1]],
state_names={'S':['Turn-off','Turn-on']})

# Define CPD of state of rain R
cpd_r = TabularCPD(variable='R', variable_card=2, values=[[0.8, 0.2]],
state_names={'R':['Not Rain','Rain']})

# Define CPD of state of Tracey's grass G
# +---------------+--------------+-------------+------------+------------+
# | S             | S(Turn-off)  | S(Turn-off) | S(Turn-on) | S(Turn-on) |
# +---------------+--------------+-------------+------------+------------+
# | R             | R(Not Rain)  |   R(Rain)   | R(Not Rain)|  R(Rain)   |
# +---------------+--------------+-------------+------------+------------+
# | G(Dry)        | 1            | 0           | 0.05       | 0          |
# +---------------+--------------+-------------+------------+------------+
# | G(Wet)        | 0            | 1           | 0.95       | 1          |
# +---------------+--------------+-------------+------------+------------+
cpd_g = TabularCPD(variable='G', variable_card=2,
                   values=[[1, 0, 0.05, 0],
                           [0, 1, 0.95, 1]],
                  evidence=['S', 'R'],
                  evidence_card=[2, 2],
                  state_names={'G':['Dry', 'Wet'],
                               'S':['Turn-off','Turn-on'],
                               'R':['Not Rain','Rain']})

# Define CPD of state of Jack's grass
# +---------------+--------------+-------------+
# | R             | R(Not Rain)  |   R(Rain)   |
# +---------------+--------------+-------------+
# | J(Dry)        | 0.85         | 0           |
# +---------------+--------------+-------------+
# | J(Wet)        | 0.15         | 1           |
# +---------------+--------------+-------------+
cpd_j = TabularCPD(variable='J', variable_card=2,
                   values=[[0.85, 0],
                           [0.15, 1]],
                  evidence=['R'],
                  evidence_card=[2],
                  state_names={'J':['Dry', 'Wet'],
                               'R':['Not Rain','Rain']})
```

```
# Associating the CPDs with the network
model_p2_updated.add_cpds(cpd_s, cpd_r, cpd_g, cpd_j)

# check_model checks for the network structure and CPDs and verifies that the
CPDs are correctly
# defined and sum to 1.
model_p2_updated.check_model()
```

```
True
```

f) Derive the probabilistic expression and numeric value for the probability that the sprinkler was left on
given that Tracey sees that both her grass and Jack’s are wet.

To find the probability that the sprinkler was left (S = 1) given that Tracey sees that both her grass and Jack’s are wet (G = 1, J = 1).

P(S = 1 | G = 1, J = 1) = P(G = 1, J = 1 | S = 1) P(S = 1) / P(G = 1, J = 1)

R is common cause of J and G, and therefore when R is observed an change in G or J, it doesn't affect J or G since it's only dependent on R: hence, A | C | B. However, the influence flows from A to C when B is not observed, therefore in this case, A and C is not independent of each other.

To calculate P(G = 1, J = 1 | S = 1)
= P(G = 1, J = 1 | S = 1, R) for all possible R value (which is 0 and 1)
= P(G = 1, J = 1 | S = 1, R = 1) P(R = 1) + P(G = 1, J = 1 | S = 1, R = 0) P(R = 0)
= 1 x 0.2 + 0.95 x 0.15 x 0.8
= 0.314

To calculate P(G = 1, J = 1)
= P(G = 1, J = 1 | S, R) for all possible S, R value (which is 0 and 1)
= P(G = 1, J = 1 | S = 0, R = 0) P(R = 0) P(S = 0) + P(G = 1, J = 1 | S = 1, R = 0) P(R = 0) P(S = 1) +
P(G = 1, J = 1 | S = 0, R = 1) P(R = 1) P(S = 0) + P(G = 1, J = 1 | S = 1, R = 1) P(R = 1) P(S = 1)
= 0 x 0.8 x 0.9 + 0.15 x 0.95 x 0.8 x 0.1 + 1 x 0.2 x 0.9 + 1 x 0.2 x 0.1
= 0.2114

Therefore, we can get P(S = 1 | G = 1, J = 1) = P(G = 1, J = 1 | S = 1) P(S = 1) / P(G = 1, J = 1)
= 0.314 x 0.1 / 0.2114
≈ 0.14853

The probabilistic expression and numeric value is verified by the program below.

```
# The probability that the sprinkler was left on given that Tracey sees that
both her grass and Jack’s are wet
infer_p2_updated = VariableElimination(model_p2_updated)

# Was it due to an overnight rain?
s_dist_2 = infer_p2_updated.query(['S'], evidence={'G':'Wet', 'J':'Wet'})
print('The probability that the sprinkler was left on given that Tracey sees
that both her grass and Jack’s are wet:')
print(s_dist_2)
```

```
Finding Elimination Order: : 100%|
██████████████████████████████████████████████████| 1/1 [00:00<00:00,
1002.94it/s]
Eliminating: R: 100%|
███████████████████████████████████████████████████████████████| 1/1
[00:00<00:00, 501.23it/s]
```

```
The probability that the sprinkler was left on given that Tracey sees that both
her grass and Jack’s are wet:
+-------------+----------+
| S           |   phi(S) |
+=============+==========+
| S(Turn-off) |   0.8515 |
+-------------+----------+
| S(Turn-on)  |   0.1485 |
+-------------+----------+
```

g) Explain why this new observation lowers the Tracey’s belief she left the sprinkler on.

Mathematically,

P(S = 1|G = 1) = P(G = 1|S = 1) P(S = 1) / P(G = 1)

P(S = 1|G = 1, J = 1) = P(G = 1, J = 1|S = 1) P(S = 1) / P(G = 1, J = 1)

Comparing these two expression, it is obvious that the second expression will be smaller than the first expression, given a set of fixed probabilities.

Logically,

R is common cause of G and J. The influence flows from J to G when R is not observed, while J/G is only dependent on R when R observed. We get G/J and G|J|R. In this case, we don't know if it rained last night, and we only know both G and J are 1. Both of these observations affect Tracey's judgment on R. Since both Tracey' and Jack's grasses are wet, it is less likely caused by two different reasons, if not raining. Therefore, our belief of rain increases.

G is the common evidence of S and R. If G is not observed, the increase of raining probability increases the probability of wet grass, but leaves the probability of turned-on sprinkle. However, since in this case we already know that G = 1, the state of R will affect the belief of S. Since we prove above that the belief of rain increases, it is less likely to believe that Tracey left her sprinkle on overnight. Therefore, our belief of spinkle on decreases.

Exercise 3. (Extra credit) Design your own Bayesian network.

Implement a belief network of your own choosing or design. It should use discrete variables (which could
be binary) and be more complex that the examples above. Use the model to illustrate deductive inference
problems.

The grading is necessarily subjective, but here is a rubric:

• Was the scenario you are trying to model clearly described?

• Were the variables and states well-chosen and clearly explained?

• How well did the examples to illustrate the model?

• Did the model go beyond or is distinct from what was already convered in the questions above? For example: larger numbers of nodes with more complex interactions; more variable states; or using simplifying assumptions for conditional probabilities like noisy-OR.

The scenerio is about fire alarm in my apartment.

The state of Drill D (Fire Drill) represents whether there is a fire drill holden in my apartment.

The state of Fire F represents whether there is an actual fire happened in my apartment.

The state of Fire Alarm A represent whether the fire alarm goes off in my apartment. Its cause could be fire drill D and Actual fire F.

The state of Notice N represent whether a notice is posted about the upcoming fire drill in my apartment. Its cause could only be Fire Drill D.

The state of Smoke M represent whether you can smell smoke in my apartment. Its cause could only be Actual Fire.

The state of Evacuation E represent whether I decide to leave the building. Its cause could be Notice N, Smoke M, and Fire Alarm A.

Graph for Bayesian Network

```
     Actual Fire (F)        Fire Drill (D)

       /    \                   /     \

     /       \                /         \

    /          \            /             \

  /              \        /                 \

↙                  ↘   ↙                     ↘

Smoke (S)        Fire Alarm (A)         Notice (N)
\                     |                       /

     \                |                   /

         \            |               /

            \         |           /

               ↘      ↓      ↙
                 Evacuation (E)
```

CPDs of each variable

```
CPD of State of Fire Drill (D)
+-----------------+-------+
| D(Happened)     | 0.001 |
+-----------------+-------+
| D(Not Happened) | 0.999 |
+-----------------+-------+
There is a small chance that my apartment holds a fire drill.
```

```
CPD of State of Actual (F)
+------------------+--------+
| F(Happened)      | 0.0002 |
+------------------+--------+
| F(Not Happened)  | 0.9998 |
+------------------+--------+
There is an even smaller chance that my apartment has an actual fire.

CPD of State of Notice(N)
+---------------+--------------+---------------------+
| D             | D(Happened)  |   D(Not Happened)   |
+---------------+--------------+---------------------+
| N(Posted)     | 0.98         | 0                   |
+---------------+--------------+---------------------+
| N(Not Posted) | 0.02         | 1                   |
+---------------+--------------+---------------------+
The concierges are good, so if there is no fire drill, there will not be notice
of fire drill posted.
However, sometimes people destroyes the notice for fun, so if there is going to
be a fire drill, there is 2% of chance that     the notice will be destroyed by
these villains.

CPD of State of Smoke (S)
+---------------+--------------+---------------------+
| F             | F(Happened)  |   F(Not Happened)   |
+---------------+--------------+---------------------+
| S(Smelled)    | 0.8          | 0.05                |
+---------------+--------------+---------------------+
| S(Not Smelled)| 0.2          | 0.95                |
+---------------+--------------+---------------------+
Since I live in the south side of my apartment, if a fire actually goes on in
north side, I might not smell the smoke.
However, since my neighbor is a terrible cook, if there is no actual fire
happened, I still sometimes smell the smoke!

CPD of State of Fire Alarm (A)
+---------------+------------------+-----------------+----------------+---------
----+
| D             | D(Not Happened)  | D(Not Happened) | D(Happened)    |
D(Happened) |
+---------------+------------------+-----------------+----------------+---------
----+
| F             | F(Not Happened)  | F(Happened)     | F(Not Happened)|
F(Happened) |
+---------------+------------------+-----------------+----------------+---------
----+
| A(Not Almared)| 1                | 0.05            | 0.1            | 0.01
   |
+---------------+------------------+-----------------+----------------+---------
----+
| G(Alarmed)    | 0                | 0.95            | 0.9            | 0.99
   |
+---------------+------------------+-----------------+----------------+---------
----+
If there is no drill or fire, the alarm will be not alarmed since they are well-
protected.
However, for the same reason, they might not alarm during an actual fire or a
fire drill or even both of them for a small       chance.
```

```
CPD of State of Evacuation (E)
+---------------+------------------+-----------------+----------------+---------
--------+
| N             | N(Not Posted)    | N(Not Posted)   | N(Not Posted)  | N(Not
Posted)   |
+---------------+------------------+-----------------+----------------+---------
--------+
| A             | A(Not Alarmed)   | A(Not Alarmed)  | A(Alarmed)     |
A(Alarmed)      |
+---------------+------------------+-----------------+----------------+---------
--------+
| S             | S(Not Smelled)   | S(Smelled)      | S(Not Smelled) |
S(Smelled)      |
+---------------+------------------+-----------------+----------------+---------
--------+
| E(Not Leave)  | 1                | 0.02            | 0.1            | 0
        |
+---------------+------------------+-----------------+----------------+---------
--------+
| E(Leave)      | 0                | 0.98            | 0.9            | 1
        |
+---------------+------------------+-----------------+----------------+---------
--------+


+---------------+------------------+-----------------+----------------+---------
--------+
| N             | N(Posted)        | N(Posted)       | N(Posted)      |
N(Posted)       |
+---------------+------------------+-----------------+----------------+---------
--------+
| A             | A(Not Alarmed)   | A(Not Alarmed)  | A(Alarmed)     |
A(Alarmed)      |
+---------------+------------------+-----------------+----------------+---------
--------+
| S             | S(Not Smelled)   | S(Smelled)      | S(Not Smelled) |
S(Smelled)      |
+---------------+------------------+-----------------+----------------+---------
--------+
| E(Not Leave)  | 1                | 0.01            | 0.4            | 0.5
        |
+---------------+------------------+-----------------+----------------+---------
--------+
| E(Leave)      | 0                | 0.99            | 0.6            | 0.5
        |
+---------------+------------------+-----------------+----------------+---------
--------+
Generally, I will be more reluctant if I saw a notice about the fire drill since
I know it is most likely a fire drill. If     I smell smoke I will be more
cautious and carefully thinking about whether to leave. If I hear the fire
alarm, I will much     more likely to leave.
```

```python
# Defining the model structure for problem 2
model_p3 = BayesianModel([('D', 'N'), ('D', 'A'), ('F', 'A'),  ('F', 'S'),
('N', 'E'),  ('A', 'E'),  ('S', 'E')])

# Defining CPD of state of fire drill D
```

```python
cpd_d = TabularCPD(variable='D', variable_card=2, values=[[0.999, 0.001]],
state_names={'D':['Not Happened','Happened']})

# Define CPD of state of actual fire F
cpd_f = TabularCPD(variable='F', variable_card=2, values=[[0.9998, 0.0002]],
state_names={'F':['Not Happened','Happened']})

# Define CPD of state of notice N
cpd_n = TabularCPD(variable='N', variable_card=2,
                   values=[[1, 0.02],
                           [0, 0.98]],
                  evidence=['D'],
                  evidence_card=[2],
                  state_names={'N':['Not Posted', 'Posted'],
                               'D':['Not Happened','Happened']})

# Define CPD of state of fire alarm A
cpd_a = TabularCPD(variable='A', variable_card=2,
                   values=[[1, 0.05, 0.1, 0.01],
                           [0, 0.95, 0.9, 0.99]],
                  evidence=['D', 'F'],
                  evidence_card=[2, 2],
                  state_names={'A':['Not Alarmed', 'Alarmed'],
                               'D':['Not Happened','Happened'],
                               'F':['Not Happened','Happened']})

# Define CPD of state of smoke S
cpd_s = TabularCPD(variable='S', variable_card=2,
                   values=[[0.95, 0.2],
                           [0.05, 0.8]],
                  evidence=['F'],
                  evidence_card=[2],
                  state_names={'S':['Not Smelled', 'Smelled'],
                               'F':['Not Happened','Happened']})

# Define CPD of state of evacuation E
cpd_e = TabularCPD(variable='E', variable_card=2,
                   values=[[1, 0.98, 0.1, 0, 1, 0.01, 0.4, 0.5],
                           [0, 0.02, 0.9, 1, 0, 0.99, 0.6, 0.5]],
                  evidence=['N', 'A', 'S'],
                  evidence_card=[2, 2, 2],
                  state_names={'E':['Not Leave', 'Leave'],
                               'N':['Not Posted', 'Posted'],
                               'A':['Not Alarmed', 'Alarmed'],
                               'S':['Not Smelled', 'Smelled']})

# Associating the CPDs with the network
model_p3.add_cpds(cpd_d, cpd_f, cpd_n, cpd_a, cpd_s, cpd_e)

# check_model checks for the network structure and CPDs and verifies that the
CPDs are correctly
# defined and sum to 1.
model_p3.check_model()
```

```
True
```

```
print('CPD of state of fire drill D')
print(model_p3.get_cpds('D'))
print('CPD of state of actual fire F')
print(model_p3.get_cpds('F'))
print('CPD of state of notice N')
print(model_p3.get_cpds('N'))
print('CPD of state of fire alarm A')
print(model_p3.get_cpds('A'))
print('CPD of state of smoke S')
print(model_p3.get_cpds('S'))
print('CPD of state of evacuation E')
print(model_p3.get_cpds('E'))
```

```
CPD of state of fire drill D
+-----------------+-------+
| D(Not Happened) | 0.999 |
+-----------------+-------+
| D(Happened)     | 0.001 |
+-----------------+-------+
CPD of state of actual fire F
+-----------------+--------+
| F(Not Happened) | 0.9998 |
+-----------------+--------+
| F(Happened)     | 0.0002 |
+-----------------+--------+
CPD of state of notice N
+---------------+-----------------+-------------+
| D             | D(Not Happened) | D(Happened) |
+---------------+-----------------+-------------+
| N(Not Posted) | 1.0             | 0.02        |
+---------------+-----------------+-------------+
| N(Posted)     | 0.0             | 0.98        |
+---------------+-----------------+-------------+
CPD of state of fire alarm A
+----------------+-----------------+-----------------+-----------------+--------
-----+
| D              | D(Not Happened) | D(Not Happened) | D(Happened)     |
D(Happened) |
+----------------+-----------------+-----------------+-----------------+--------
-----+
| F              | F(Not Happened) | F(Happened)     | F(Not Happened) |
F(Happened) |
+----------------+-----------------+-----------------+-----------------+--------
-----+
| A(Not Alarmed) | 1.0             | 0.05            | 0.1             | 0.01
     |
+----------------+-----------------+-----------------+-----------------+--------
-----+
| A(Alarmed)     | 0.0             | 0.95            | 0.9             | 0.99
     |
+----------------+-----------------+-----------------+-----------------+--------
-----+
CPD of state of smoke S
+----------------+-----------------+-------------+
| F               | F(Not Happened) | F(Happened) |
+-----------------+-----------------+-------------+
| S(Not Smelled)  | 0.95            | 0.2         |
+-----------------+-----------------+-------------+
| S(Smelled)      | 0.05            | 0.8         |
+-----------------+-----------------+-------------+
CPD of state of evacuation E
+---------------+----------------+----------------+----------------+-------------
--+-----------------+----------------+----------------+-------------+
| N             | N(Not Posted)  | N(Not Posted)  | N(Not Posted)  | N(Not
Posted) | N(Posted)      | N(Posted)      | N(Posted)      | N(Posted)   |
+---------------+----------------+----------------+----------------+-------------
--+-----------------+----------------+----------------+-------------+
| A             | A(Not Alarmed) | A(Not Alarmed) | A(Alarmed)     | A(Alarmed)
| A(Not Alarmed) | A(Not Alarmed) | A(Alarmed)     | A(Alarmed) |
+---------------+----------------+----------------+----------------+-------------
--+-----------------+----------------+----------------+-------------+
| S             | S(Not Smelled) | S(Smelled)     | S(Not Smelled) | S(Smelled)
| S(Not Smelled) | S(Smelled)     | S(Not Smelled) | S(Smelled) |
+---------------+----------------+----------------+----------------+-------------
--+-----------------+----------------+----------------+-------------+
| E(Not Leave)  | 1.0            | 0.98           | 0.1            | 0.0
  | 1.0             | 0.01           | 0.4            | 0.5         |
+---------------+----------------+----------------+----------------+-------------
--+-----------------+----------------+----------------+-------------+
| E(Leave)      | 0.0            | 0.02           | 0.9            | 1.0
  | 0.0             | 0.99           | 0.6            | 0.5         |
+---------------+----------------+----------------+----------------+-------------
--+-----------------+----------------+----------------+-------------+
```

I am really interested in the following questions:
1) If no observation is made, what's the chance that I will leave my apartment for evacuation?
2) If I hear an alarm, what's the chance that I will leave my apartment for evacuation?
3) If I smell smoke, what's the chance that I will leave my apartment for evacuation?
4) If I head an alarm and saw a notice about it before, what's the chance that I will leave my apartment for evacuation?

```
infer_p3 = VariableElimination(model_p3)
```

```
# 1) If no observation is made, what's the chance that I will leave my apartment
for evacuation?
print("1) If no observation is made, what's the chance that I will leave my
apartment for evacuation?")
print(infer_p3.query(['E']))
```

```
1) If no observation is made, what's the chance that I will leave my apartment
for evacuation?
```

```
Finding Elimination Order: : 100%|
██████████████████████████████████████████████████| 5/5 [00:00<00:00,
5015.91it/s]
Eliminating: D: 100%|
███████████████████████████████████████████████████████████████| 5/5
[00:00<00:00, 501.34it/s]
```

```
+--------------+----------+
| E            |   phi(E) |
+==============+==========+
| E(Not Leave) |   0.9983 |
+--------------+----------+
| E(Leave)     |   0.0017 |
+--------------+----------+
```

```python
# 2) If I hear an alarm, what's the chance that I will leave my apartment for evacuation?
print("2) If I hear an alarm, what's the chance that I will leave my apartment for evacuation?")
print(infer_p3.query(['E'], evidence={'A':'Alarmed'}))
```

```
2) If I hear an alarm, what's the chance that I will leave my apartment for evacuation?
```

```
Finding Elimination Order: : 100%|
█████████████████████████████████████████████████████| 4/4 [00:00<00:00, 4011.77it/s]
Eliminating: D: 100%|
███████████████████████████████████████████████████████████████████| 4/4 [00:00<00:00, 668.47it/s]
```

```
+--------------+----------+
| E            |   phi(E) |
+==============+==========+
| E(Not Leave) |   0.3328 |
+--------------+----------+
| E(Leave)     |   0.6672 |
+--------------+----------+
```

```python
# 3) If I smell smoke, what's the chance that I will leave my apartment for evacuation?
print("3) If I smell smoke, what's the chance that I will leave my apartment for evacuation?")
print(infer_p3.query(['E'], evidence={'S':'Smelled'}))
```

```
3) If I smell smoke, what's the chance that I will leave my apartment for evacuation?
```

```
Finding Elimination Order: : 100%|
█████████████████████████████████████████████████████| 4/4 [00:00<00:00, 4010.81it/s]
Eliminating: D: 100%|
███████████████████████████████████████████████████████████████████| 4/4 [00:00<00:00, 501.35it/s]
```

```
+--------------+----------+
| E            |   phi(E) |
+==============+==========+
| E(Not Leave) |   0.9765 |
+--------------+----------+
| E(Leave)     |   0.0235 |
+--------------+----------+
```

```python
# 4) If I head an alarm and saw a notice about it before, what's the chance that
I will leave my apartment for evacuation?
print("2) If I hear an alarm, what's the chance that I will leave my apartment
for evacuation?")
print(infer_p3.query(['E'], evidence={'N':'Posted', 'A':'Alarmed'}))
```

```
2) If I hear an alarm, what's the chance that I will leave my apartment for
evacuation?
```

```
Finding Elimination Order: : 100%|
██████████████████████████████████████████████████████| 3/3 [00:00<00:00,
3009.55it/s]
Eliminating: D: 100%|
████████████████████████████████████████████████████████████████████| 3/3
[00:00<00:00, 752.12it/s]
```

```
+--------------+----------+
| E            |   phi(E) |
+==============+==========+
| E(Not Leave) |   0.4050 |
+--------------+----------+
| E(Leave)     |   0.5950 |
+--------------+----------+
```