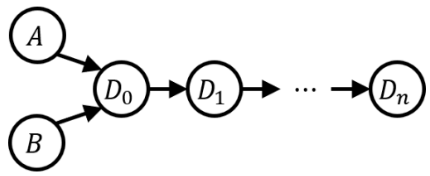
Problem 1: Descendants of Colliders (20 points)

Here we will investigate the absence of conditional independence guarantees between two random variables in a Bayes net when an arbitrary common descendant is observed. We will consider the case of a simple chain of descendants, as shown below.



Suppose that all random variables are binary. The marginal distributions of A and B are both uniform $(0.5, 0.5)$, and the CPTs of the collider D_0 and its descendants are as follows:

A	B	$\Pr(+d_0 \mid A, B)$
$+a$	$+b$	0.5
$+a$	$-b$	0.0
$-a$	$+b$	0.0
$-a$	$-b$	0.5

D_{i-1}	$\Pr(+d_i \mid D_{i-1})$
$+d_{i-1}$	0.0
$-d_{i-1}$	1.0

1. Compute the distributions Pr(A, B) and Pr(A, B| - d0). Show whether A is conditionally independent of B given D0.

Pr(A=+a) = 0.5
Pr(A=-a) = 0.5
Pr(B=+b) = 0.5
Pr(B=-b) = 0.5

Pr(A=+a, B=+b)= 0.5*0.5 = 0.25
Pr(A=+a, B=-b)= 0.5*0.5 = 0.25
Pr(A=-a, B=+b)= 0.5*0.5 = 0.25
Pr(A=-a, B=-b)= 0.5*0.5 = 0.25

Pr(+d0) = $\sum_{a,b}$Pr(+d0|A, B)Pr(A, B)= 0.5*0.25 + 0.0*0.25 + 0.0*0.25 + 0.5*0.25 = 0.25

Pr(-d0) = 1 - Pr(+d0) = 0.75

Pr(A, B|-d0) = Pr(-d0|A, B)*Pr(A, B)/Pr(-d0)
Pr(A=+a, B=+b|-d0) = 0.5*0.25/0.75 = ⅙
Pr(A=+a, B=-b|-d0) = 1.0*0.25/0.75 = ⅓
Pr(A=-a, B=+b|-d0) = 1.0*0.25/0.75 = ⅓
Pr(A=-a, B=-b|-d0) = 0.5*0.25/0.75 = ⅙

Not conditionally independent.
Pr(A=+a, B=+b|-d0) = Pr(A=+a|-d0)Pr(B=+b|-d0)

Let
$Pr(A=+a|-d0)Pr(B=+b|-d0)= xy$
$Pr(A=+a|-d0)Pr(B=-b|-d0)= x(1-y)= x-xy$
$Pr(A=-a|-d0)Pr(B=+b|-d0)= (1-x)y= y-xy$
$Pr(A=-a|-d0)Pr(B=-b|-d0)=(1-x)(1-y)=1-x-y+xy$

If $Pr(A,B|-d0) = Pr(A|-d0)Pr(B|-d0)$
Then $Pr(A=+a, B=-b|-d0) = Pr(A=-a, B=+b|-d0)$ would imply $x-xy=y-xy$, and therefore $x = y = 1/\sqrt{6}$
However, $1-x-y+xy = 1-2/\sqrt{6}+\frac{1}{6} != \frac{1}{6}$
Therefore A and B are not conditionally independent given D0.

2. Find an analytical expression for the joint probabilities Pr(A, B, di), where $0 \le i \le n$. The expression should only contain terms in the Bayes net CPTs and involve chain rule and/or marginalization operations.

$Pr(A, B, di) = \Sigma_{d0, \dots, di-1} Pr(A, B, d0, \dots, di)$
$= \Sigma_{d0, \dots, di-1} Pr(A)*Pr(B)*Pr(d0|A, B)*Pr(d1|A, B, d0)* \dots *Pr(di|A, B, d0,\dots,di-1)$
$= \Sigma_{d0, \dots, di-1} Pr(A)*Pr(B)*Pr(d0|A, B)*Pr(d1|d0)*Pr(d2|d1)* \dots *Pr(di|di-1)$

$$= \sum_{d0, \dots, di-1} Pr(A) * Pr(B) * Pr(d_0|A, B) * \prod_{k=1}^{i} Pr(d_k|d_{k-1})$$

3. Now using the values from the given CPTs, numerically compute the joint conditional distribution Pr(A, B| – di), where i is odd. You can give a brief explanation for how you simplify the analytical expression. Show whether A is conditionally independent of B given Di.

$Pr(+d0|+a,+b) = 0.5$
$Pr(+d0|+a,-b) = 0.0$
$Pr(+d0|-a,+b) = 0.0$
$Pr(+d0|-a,-b) = 0.5$

$Pr(+d0)= 0.25$

$Pr(A,B|+d0) = Pr(+d0|A,B)Pr(A,B)/Pr(+d0)$

$Pr(+a,+b|+d0) = 0.5*0.25/0.25 = 0.5$
$Pr(+a,-b|+d0) = 0.0*0.25/0.25 = 0$
$Pr(-a,+b|+d0) = 0.0*0.25/0.25 = 0$
$Pr(-a,-b|+d0) = 0.5*0.25/0.25 = 0.5$

Pr(+di|+di-1) = 0.0
Pr(-di|+di-1) = 1.0
Pr(+di|-di-1) = 1.0
Pr(-di|-di-1) = 0.0

di = -di-1 always for all i > 1.
When i is odd, di = -d0
Pr(-di|+d0) = 1.0
Pr(+di|-d0) = 1.0
And
Pr(+di|+d0) = 0.0
Pr(-di|-d0) = 0.0

Then
Pr(+a,+b|-di) = 0.5
Pr(-a,-b|-di) = 0.5
Pr(+a,-b|-di) = 0.0
Pr(-a,+b|-di) = 0.0
This indicate that A is not conditionally independent of B.
The reason is that, for example, when A=+a, it is impossible that B=-b, similarly, when A=-a, it is impossible that B=+b.

4. Finally, numerically compute the joint conditional distribution Pr(A, B| − d0, +di), where i is odd. Again, you may explain how you can do so by simplifying the relevant analytical expression. Do not refer to the local independence rule of chains from class. Show how we can conclude that A and B are conditionally independent of Di given D0.

From 1.3 we have already shown that

di = -di-1 always for all i > 1.
When i is odd, di = -d0
Pr(-di|+d0) = 1.0
Pr(+di|-d0) = 1.0
And
Pr(+di|+d0) = 0.0
Pr(-di|-d0) = 0.0

Therefore Pr(A,B|-d0,+di)=Pr(A,B|-d0) when i is odd. The transition is deterministic.

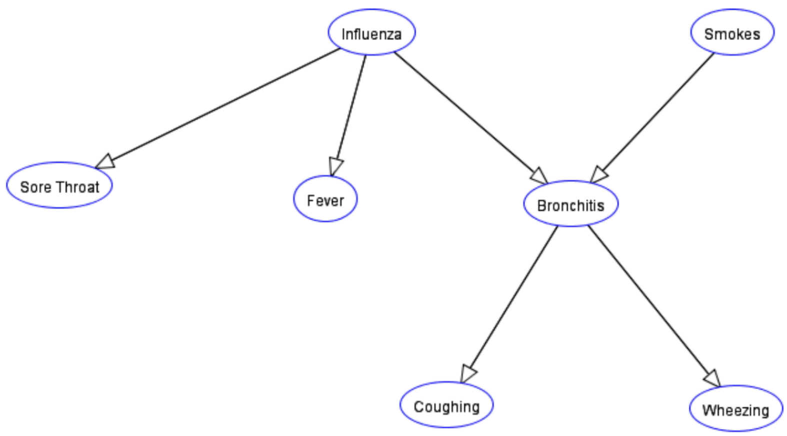
We'vd already have
Pr(A=+a, B=+b|-d0) = 0.5*0.25/0.75 = ⅙

```
Pr(A=+a, B=-b|-d0) = 1.0*0.25/0.75 = ⅓
Pr(A=-a, B=+b|-d0) = 1.0*0.25/0.75 = ⅓
Pr(A=-a, B=-b|-d0) = 0.5*0.25/0.75 = ⅙

Therefore,
Pr(A=+a, B=+b|-d0,+di) = 0.5*0.25/0.75 = ⅙
Pr(A=+a, B=-b|-d0,+di) = 1.0*0.25/0.75 = ⅓
Pr(A=-a, B=+b|-d0,+di) = 1.0*0.25/0.75 = ⅓
Pr(A=-a, B=-b|-d0,+di) = 0.5*0.25/0.75 = ⅙
```

Problem 2: Diagnostic Bayes Net (24 points)

The following Bayes net is the "Simple Diagnostic Example" from the Sample Problems of the Belief and Decision Networks tool on AIspace. All variables are binary.



1. We are interested in computing the distribution Pr(Coughing | Fever = True). Find a minimal analytical expression for this distribution (or its unnormalized version) in terms of the Bayes net CPTs, and then rewrite each term as a factor explicitly showing the variable dependencies. What is the maximum size of the intermediate factor if all marginalization is done at the end?

$$Pr(Coughing \mid Fever = True) \propto Pr(Coughing, Fever = True)$$

$$Pr(Coughing, Fever = True) = \sum_{Influenza} \sum_{Bronchitis} \sum_{Smokes} Pr(Coughing|Bronchitis) * Pr(Bronchitis|Influenza, Smokes) * Pr(Fever = True|Influenza) * Pr(Influenza) * Pr(Smokes)$$

f1(Coughing,Bronchitis)=Pr(Coughing|Bronchitis)
f2(Bronchitis,Influenza,Smokes)=Pr(Bronchitis|Influenza,Smokes)
f3(Fever,Influenza)=Pr(Fever=True|Influenza)
f4(Influenza)=Pr(Influenza)
f5(Smokes)=Pr(Smokes)

maximum size of the intermediate factor: 2^4 = 16

2. We employ variable elimination to solve the query above. Consider the following two variable orderings (you may only need to sum over a subset of each):

   (a) influenza, smokes, sore throat, fever, bronchitis, coughing, wheezing
   (b) wheezing, coughing, bronchitis, fever, sore throat, smokes, influenza

   For each of these orderings, rewrite your analytical expression by splitting up the sum over each variable, and factor out the maximal number of terms from each summation. Remember that sums are evaluated from right to left, or from inside out. What is the size of the largest intermediate factor in each?

$$Pr(Coughing,\ Fever\ = True) = \sum_{Bronchitis} Pr(Coughing|Bronchitis) \sum_{Smokes} Pr(Smokes)$$

$$\sum_{Influenza} Pr(Fever = True|Influenza)Pr(Influenza)Pr(Bronchitis|Influenza, Smokes)$$

$$= \sum_{Bronchitis} f1(Coughing, Bronchitis) \sum_{Smokes} f5(Smokes)$$

$$\sum_{Influenza} f3(Fever, Influenza)f4(Influenza)f2(Bronchitis, Influenza, Smokes)$$

Size of the largest intermediate factor: 2^3 = 8

(b)wheezing,coughing,bronchitis,fever,sore throat,smokes,influenza

$$Pr(Coughing,\ Fever\ = True) = \sum_{Influenza} Pr(Fever = True|Influenza)Pr(Influenza) \sum_{Smokes} Pr(Smokes)$$

$$\sum_{Bronchitis} Pr(Coughing|Bronchitis)Pr(Bronchitis|Influenza, Smokes)$$

$$= \sum_{Influenza} f3(Fever, Influenza)f4(Influenza) \sum_{Smokes} f5(Smokes)$$

$$\sum_{Bronchitis} f1(Coughing, Bronchitis)f2(Bronchitis, Influenza, Smokes)$$

Size of the largest intermediate factor: 2^4 = 16

3. Follow the more efficient ordering from above to numerically compute Pr(Coughing | Fever = True) using the applet parameters. You can use the applet to verify that this comes out to (0.45, 0.55), but you should carry out the computations yourself, either manually or using Python (please attach your notebook/code in your PDF if the latter). In addition to the final result, show each intermediate factor resulting from each variable summation step

Simple Diagnostic Network

Influenza	
T	0.05
F	0.95

Smokes	
T	0.2
F	0.8

	Sore Throat	
Influenza = T	T	0.3
	F	0.7
Influenza = F	T	0.001
	F	0.999

	Fever	
Influenza = T	T	0.9
	F	0.1
Influenza = F	T	0.05
	F	0.95

		Bronchitis	

Influenza = T	Smokes=T	T	0.99
		F	0.01
Influenza = T	Smokes=F	T	0.9
		F	0.1
Influenza = F	Smokes=T	T	0.7
		F	0.3
Influenza = F	Smokes=F	T	0.0001
		F	0.9999

	Coughing	
Bronchitis = T	T	0.8
	F	0.2
Bronchitis = F	T	0.07
	F	0.93

	Wheezing	
Bronchitis = T	T	0.6
	F	0.4
Bronchitis = F	T	0.001
	F	0.999

$$Pr(Coughing,\ Fever = True) = \sum_{Bronchitis} Pr(Coughing|Bronchitis) \sum_{Smokes} Pr(Smokes)$$

$$\sum_{Influenza} Pr(Fever = True|Influenza)Pr(Influenza)Pr(Bronchitis|Influenza, Smokes)$$

$$= \sum_{Bronchitis} f1(Coughing, Bronchitis) \sum_{Smokes} f5(Smokes)$$

$$\sum_{Influenza} f3(Fever, Influenza)f4(Influenza)f2(Bronchitis, Influenza, Smokes)$$

$$\sum_{Influenza} f3(Fever, Influenza) f4(Influenza) f2(Bronchitis, Influenza, Smokes)$$

Given Fever = True

	Bronchitis	
Smokes=T	T	0.99*0.05*0.9+ 0.7*0.95*0.05 = 0.0778
	F	0.01*0.05*0.9+0.3*0.95*0.05 = 0.0147
Smokes=F	T	0.9*0.05*0.9+0.0001*0.95*0.05 = 0.04050475
	F	0.1*0.05*0.9+0.9999*0.95*0.05 = 0.05199525

$$\sum_{Smokes} f5(Smokes)$$

Bronchitis	
T	0.0778*0.2 + 0.04050475*0.8 = 0.0479638
F	0.0147*0.2 + 0.05199525* 0.8 = 0.0445362

$$\sum_{Bronchitis} f1(Coughing, Bronchitis)$$

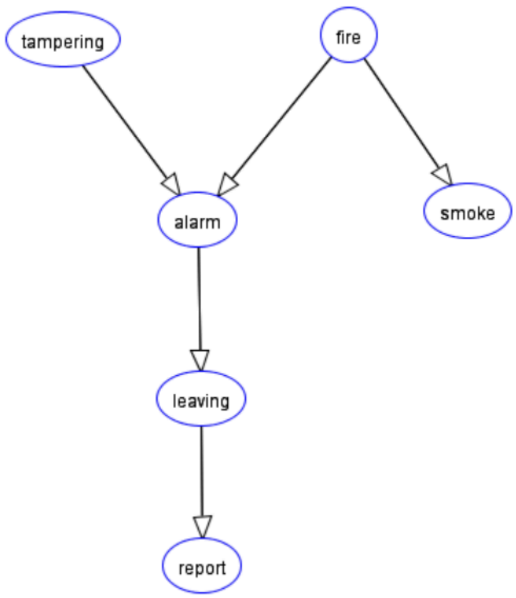
Coughing	
T	0.8*0.0479638 + 0.07*0.0445362 = 0.041488574
F	0.2*0.0479638 + 0.93*0.0445362 = 0.051011426

After Normalized:

Coughing	Fever = T	
T	T	0.45
F	T	0.55

Problem 3: Fire Alarm Bayes Net (20 points)

The following Bayes net is the "Fire Alarm Belief Network" from the Sample Problems of the Belief and Decision Networks tool on AIspace. All variables are binary.



Fire Alarm Network

- Tampering. T, 0.02; F, 0.98
- Fire. T, 0.01; F, 0.99.
- Smoke.
  Fire=T: T, 0.9; F, 0.1.
  Fire=F: T, 0.01; F, 0.99.
- Alarm.
  Tampering=T, Fire=T: T, 0.5; F, 0.5.
  Tampering=T, Fire=F: T, 0.85; F, 0.15.
  Tampering=F, Fire=T: T, 0.99; F, 0.01.
  Tampering=F, Fire=F: T, 0.0; F, 1.0.
- Leaving.
  Alarm=T: T, 0.88; F: 0.12.
  Alarm=F: T, 0.0; F, 1.0.
- Report.
  Leaving=T: T, 0.75; F: 0.25.
  Leaving=F: T, 0.01; F, 0.99.

1. Consider using likelihood weighting to solve for a posterior distribution conditioned on the following sets of evidence: i) Tampering and Fire, ii) Smoke and Report. For each case, explain whether the generated samples reflect the prior or posterior distribution. Also explain the impact of the likelihood weights in helping us to estimate the posterior in each case.

   i) Tampering and Fire

   Since 'Tampering' and 'Fire' are root nodes (causes that do not have any parents in the network), when you condition on them, the samples generated are indeed reflecting the posterior distribution directly. This is because these nodes can be set to the evidence values without needing to consider other nodes. In other words, there's no need to adjust the weights of the samples since the evidence directly sets these variables to their observed values.

   The likelihood weights do not alter the posterior because the evidence doesn't depend on the values of other variables; they only confirm the evidence we have observed.

   ii) Smoke and Report

   'Smoke' and 'Report' are not root nodes. In this case, when we perform sampling, the prior distribution is initially reflected in the samples.

   The likelihood weights serve as a corrective factor, adjusting the influence of each sample based on how well it explains the observed evidence.

2. Inspect the individual CPTs of this Bayes net in the applet. Identify all unique evidence events that are impossible to occur. (You don’t need to list any events that contain those you already listed as subsets. For example, if you identify {+t, +f} to be such an event, you do not need to write {+t, +f, +a}, {+t, +f, +s}, etc.) What would happen if we perform rejection sampling or likelihood weighting with these events as observed?

   Tampering=F, Fire=F, Alarm=T
   Alarm=F, Leaving=T

   Rejection: When performing rejection sampling with these impossible events as observed, any samples that include these combinations would be rejected.

   Likelihood Weighting: If the evidence is impossible, the weight for every sample becomes zero. This results in all samples having zero weight, meaning that no meaningful posterior estimation can be performed. The output will essentially be uninformative as all weights are zero.

3. We perform Gibbs sampling and want to resample the Fire variable conditioned on the current sample (+t, +a, +s, +l, +r). Find a minimal analytical expression for Pr(Fire | sample) (or its unnormalized version) in terms of the Bayes net CPTs, and then numerically compute it using the applet parameters. You can use the applet to verify that this comes out to (0.35, 0.65), but you should carry out the computations yourself and show your work.

   Pr(Fire | +t, +a, +s, +l, +r) ∝ Pr(F, +t, +a, +s, +l, +r)

   Pr(+f, +t, +a, +s, +l, +r) =
   Pr(+f)Pr(+t)Pr(+a|+t,+f)Pr(+s|+f)Pr(+l|+a)Pr(+r|+l)
   = 0.01×0.02×0.5×0.9×0.88×0.75 = 0.0000594

   Pr(-f, +t, +a, +s, +l, +r) =
   Pr(-f)Pr(+t)Pr(+a|+t,-f)Pr(+s|-f)Pr(+l|+a)Pr(+r|+l)
   = 0.99×0.02×0.85×0.01×0.88×0.75 = 0.000111078

   After Normalized:
   Pr(+f| +t, +a, +s, +l, +r) = 0.348
   Pr(-f| +t, +a, +s, +l, +r) = 0.652

4. Repeat the above, but now suppose that we have lost the value of Alarm in the sample. In other words, compute the distribution Pr(Fire | + t, +s, +l, +r). Why does conditioning on less information lead to potentially greater computation?

   Pr(+f, +t, +a, +s, +l, +r) + Pr(+f, +t, -a, +s, +l, +r) =
   0.01×0.02×0.5×0.9×0.88×0.75 + 0.01×0.02×0.5×0.9×0.0×0.75 =
   0.0000594

   Pr(-f| +t, +a, +s, +l, +r) + Pr(-f| +t, -a, +s, +l, +r) =
   0.99×0.02×0.85×0.01×0.88×0.75 + 0.99×0.02×0.15×0.01×0.0×0.75 =
   0.000111078

   After Normalization:
   Pr(+f| +t, +s, +l, +r) = 0.348
   Pr(-f| +t, +s, +l, +r) = 0.652

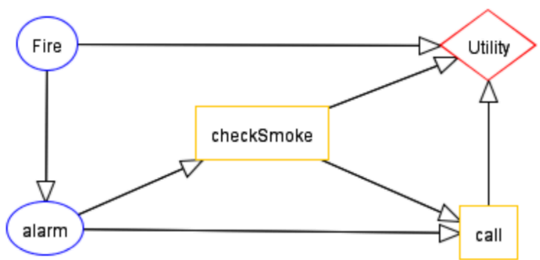
   Why?
   These probabilities match our previous calculations with Alarms known. This is because the situation where Alarm is false never contributes to cases where Leaving is true, Pr(+l|-a) = 0.

   Without direct evidence about Alarm, we must consider all possible states of this variable, which require us to compute the joint probability for each state and then sum up these probabilities.

Problem 4: Fire Alarm Decision Network (16 points)

The following Bayes net is a simplified version of the "Fire Alarm Decision Problem" from the Sample Problems of the Belief and Decision Networks tool on AIspace. All variables are binary, and the chance node CPTs as well as utility values are shown below.

While you are not required to do so, we recommend that you use **pandas** functionality to carry out the operations in this and the next problem. It is also acceptable if you prefer to perform all calculations manually. Please include your work in either case.



Fire	CheckSmoke	Call	Utility
True	True	True	−220
True	True	False	−5020
True	False	True	−200
True	False	False	−5000
False	True	True	−220
False	True	False	−20
False	False	True	−200
False	False	False	0

Fire	Pr(Fire)
T	0.01
F	0.99

Fire	Alarm	Pr(Alarm | Fire)
True	True	0.95
True	False	0.05
False	True	0.01
False	False	0.99

1. Compute the joint factor required to sum out the chance nodes that are not parents of any decision nodes. Show the resultant factor after performing the elimination.

decision nodes: checkSmoke, call
We have to eliminate the Fire node since it's the only chance node that is not a parent to any decision nodes

sum out Fire:

```
  Alarm CheckSmoke Call  utility
0     T          T    T   -4.268
1     T          T    F  -47.888
2     T          F    T   -3.880
3     T          F    F  -47.500
4     F          T    T -215.732
5     F          T    F  -22.112
6     F          F    T -196.120
7     F          F    F   -2.500
```

2. Determine the optimal decision function and expected utilities for Call. Show the table containing this information.

	CheckSmoke	Alarm	Optimal_Call	Optimal EU
0	F	F	F	-2.500
1	F	T	T	-3.880
2	T	F	F	-22.112
3	T	T	T	-4.268

3. Briefly explain whether the optimal decision function for Call actually depends on both of its parent nodes. Would the result change if we had determined the optimal decision function for CheckSmoke first, followed by Call?

The optimal decision for "Call" indeed depends on the state of its parent node "Alarm." The expected utility calculations for making the call or not are based on the current state of the alarm, which in turn depends on the presence of a fire, given by the node "Fire."

The decision function for "Call" does not explicitly depend on the "CheckSmoke" node. It is assumed that the state of "Alarm" has already integrated the information from "CheckSmoke."

If we were to determine the optimal decision function for "CheckSmoke" first, it should not affect the optimal decision function for "Call" as the utility values in the table for calling or not are already given and do not depend on whether "CheckSmoke" was true or false.

4. Determine the optimal decision function and expected utilities for CheckSmoke. Show the table containing this information. Then compute the maximum expected utility.

	Alarm	Optimal_CheckSmoke	Optimal EU for CheckSmoke
0	F	F	-2.50
1	T	F	-3.88

Max_expected_utility: -2.50 -3.88 = -6.38

```
import pandas as pd

Fire = pd.DataFrame({'Fire': ['T', 'F'], 'fF': [0.01, 0.99]})
Alarm = pd.DataFrame({'Fire': ['T', 'T', 'F', 'F'], 'Alarm': ['T', 'F', 'T',
'F'], 'fA': [0.95, 0.05, 0.01, 0.99]})
utility = pd.DataFrame({
    'Fire': ['T', 'T', 'T', 'T', 'F', 'F', 'F', 'F'],
    'CheckSmoke': ['T', 'T', 'F', 'F', 'T', 'T', 'F', 'F'],
    'Call': ['T', 'F', 'T', 'F', 'T', 'F', 'T', 'F'],
    'utility': [-220, -5020, -200, -5000, -220, -20, -200, 0]
})

joint_FA = pd.merge(Alarm, Fire, on='Fire')
joint_FA['P(F,A)'] = joint_FA['fA'] * joint_FA['fF']

joint_FAC = pd.merge(joint_FA, utility, on='Fire')
joint_FAC['EU_contribution'] = joint_FAC['P(F,A)'] * joint_FAC['utility']

resultant_factor = joint_FAC.groupby(['CheckSmoke', 'Call',
'Alarm']).sum().reset_index()[['CheckSmoke', 'Call', 'Alarm',
'EU_contribution']]

print(resultant_factor)

optimal_decision = resultant_factor.groupby(['CheckSmoke', 'Alarm']).apply(
    lambda df: df.loc[df['EU_contribution'].idxmax()]
).reset_index(drop=True)

optimal_decision = optimal_decision[['CheckSmoke', 'Alarm', 'Call',
'EU_contribution']]
optimal_decision.rename(columns={'EU_contribution': 'Optimal EU'},
inplace=True)

print(optimal_decision)

optimal_decision_checksmoke = optimal_decision.groupby('Alarm').apply(
    lambda df: df.loc[df['Optimal EU'].idxmax()]
).reset_index(drop=True)

optimal_decision_checksmoke = optimal_decision_checksmoke[['Alarm',
'CheckSmoke', 'Optimal EU']]
optimal_decision_checksmoke.rename(columns={'Optimal EU': 'Optimal EU for
CheckSmoke'}, inplace=True)

maximum_expected_utility = optimal_decision_checksmoke['Optimal EU for
CheckSmoke'].max()

print(optimal_decision_checksmoke)

print(maximum_expected_utility)
```

Problem 5: Fire Alarm Decision Network Redux (20 points)
Now suppose that the Call decision also depends on Fire; in other words, we get to observe Fire prior to making the decision.

1. Explain why we should not eliminate any chance nodes prior to computing optimal decision functions in this modified network.

In the original setup, certain conditional independencies might have allowed for the simplification of the decision process. However, once "Call" depends on both "Fire" and "Alarm," those independencies no longer hold. "Fire" and "Alarm" are no longer independent given the "Call" decision, and their interaction must be considered.

2. Determine the optimal decision function and expected utilities for Call. Show the table containing this information.

	Fire	CheckSmoke	Alarm	Optimal_Call	Optimal EU
0	F	F	F	F	0.000
1	F	F	T	F	0.000
2	F	T	F	F	-19.602
3	F	T	T	F	-0.198
4	T	F	F	T	-0.100
5	T	F	T	T	-1.900
6	T	T	F	T	-0.110
7	T	T	T	T	-2.090

3. Eliminate any chance nodes that are not parents of the remaining decision node, CheckSmoke. Show the resultant factor.

	CheckSmoke	Alarm	Optimal EU
0	F	F	-0.100
1	F	T	-1.900
2	T	F	-19.712
3	T	T	-2.288

4. Determine the optimal decision function and expected utilities for CheckSmoke. Show the table containing this information. Then compute the maximum expected utility, as well as the VPI of Fire for the Call decision.

	Alarm	CheckSmoke	Optimal EU for CheckSmoke
0	F	F	-0.1
1	T	F	-1.9

Max_expected_utility: -0.1 -1.9 = -2

VPI = -2 - (-6.38) = 4.38

5. Give an interpretation for the VPI of Fire. Why might it be beneficial to be able to observe Fire directly, in addition to (or instead of) relying on the “after-effects” of a Fire event?

The Value of Perfect Information (VPI) for Fire quantifies the benefits of knowing with certainty whether there is a fire before deciding to call for emergency services.

Indirect signs of a fire, such as smoke or an alarm, can sometimes be false indicators. Smoke might come from a source that is not an emergency, and alarms can be triggered accidentally without a fire. Directly observing or knowing a fire's presence reduces the likelihood of making a decision based on false premises, thereby avoiding unnecessary actions like false alarms that could divert emergency services from actual emergencies.

```
import pandas as pd

Fire = pd.DataFrame({'Fire': ['T', 'F'], 'fF': [0.01, 0.99]})
Alarm = pd.DataFrame({'Fire': ['T', 'T', 'F', 'F'], 'Alarm': ['T', 'F', 'T',
'F'], 'fA': [0.95, 0.05, 0.01, 0.99]})
utility = pd.DataFrame({
    'Fire': ['T', 'T', 'T', 'T', 'F', 'F', 'F', 'F'],
    'CheckSmoke': ['T', 'T', 'F', 'F', 'T', 'T', 'F', 'F'],
    'Call': ['T', 'F', 'T', 'F', 'T', 'F', 'T', 'F'],
    'utility': [-220, -5020, -200, -5000, -220, -20, -200, 0]
})

joint_FA = pd.merge(Alarm, Fire, on='Fire')
joint_FA['P(F,A)'] = joint_FA['fA'] * joint_FA['fF']

joint_FAC = pd.merge(joint_FA, utility, on='Fire')
joint_FAC['EU_contribution'] = joint_FAC['P(F,A)'] * joint_FAC['utility']

resultant_factor = joint_FAC.groupby(['Fire','CheckSmoke', 'Call',
'Alarm']).sum().reset_index()[['Fire','CheckSmoke', 'Call', 'Alarm',
'EU_contribution']]

optimal_decision = resultant_factor.groupby(['Fire','CheckSmoke',
'Alarm']).apply(
    lambda df: df.loc[df['EU_contribution'].idxmax()]
).reset_index(drop=True)

optimal_decision = optimal_decision[['Fire','CheckSmoke', 'Alarm', 'Call',
'EU_contribution']]
optimal_decision.rename(columns={'EU_contribution': 'Optimal EU'},
inplace=True)

print(optimal_decision)

optimal_decision = optimal_decision.groupby(['CheckSmoke',
'Alarm']).sum().reset_index()[['CheckSmoke', 'Alarm', 'Optimal EU']]

print(optimal_decision)

optimal_decision_checksmoke = optimal_decision.groupby('Alarm').apply(
    lambda df: df.loc[df['Optimal EU'].idxmax()]
).reset_index(drop=True)

optimal_decision_checksmoke = optimal_decision_checksmoke[['Alarm',
'CheckSmoke', 'Optimal EU']]
optimal_decision_checksmoke.rename(columns={'Optimal EU': 'Optimal EU for
CheckSmoke'}, inplace=True)

maximum_expected_utility = optimal_decision_checksmoke['Optimal EU for
CheckSmoke'].max()

print(optimal_decision_checksmoke)

print(maximum_expected_utility)
```