CS486 - Homework Assignment 2 Chen Lang 20528863

1. Code

factor.py

```
import numpy as np


class Factor(object):
    # class variables
    # self.vars : list of variables in order
    # self.idInfo : dict of vars and vals
    # self.pTable : probability table


    # __init__()
    # vars: list of variables
    # vals: 2d list of values of variables
    # p: 2d list of probability of values
    def __init__(self, vars, vals, p):
        if len(vars) == 0:
            return


        # Set variables
        self.vars = vars


        # Get count of variables
        cntVars = len(vars)
        # Generate identity information
        self.idInfo = dict()
        for i in xrange(0, cntVars):
            self.idInfo[vars[i]] = vals[i]
```

```
        # Generate np array
        self.pTable = np.array(p)
        rankTable = ()
        for i in xrange(0, cntVars):
            curTuple = (len(vals[i]),)
            rankTable = rankTable + curTuple
        # Reshape
        self.pTable = self.pTable.reshape(rankTable)


# print_table_recurse(): print_table() helper function
def print_table_recurse(self, curTuple):
        # Print probability recursively
        lenCurTuple = len(curTuple)
        if (len(self.vars) == lenCurTuple):
            # Base: Print
            for i in xrange(0, len(self.vars)):
                vals = self.idInfo[self.vars[i]]
                print vals[curTuple[i]],
                print(','),
            print(': '),
            print self.pTable.item(curTuple)
        else:
            # Next variable
            vals = self.idInfo[self.vars[lenCurTuple]]
            for i in xrange(0, len(vals)):
                nextTuple = (i, )
                self.print_table_recurse(curTuple + nextTuple)


# print_table(): Print current factor in a neat form
```

```
def print_table(self):
        # Use for printing probability
        curTuple = ()
        print(self.vars)
        self.print_table_recurse(curTuple)


# copy(): Duplicate current factor
def copy(self):
        fN = Factor([],[],[])
        fN.vars = list(self.vars)
        fN.idInfo = dict(self.idInfo)
        fN.pTable = self.pTable.copy()
        return fN


# sort_factor(): Sort variables in this factor
def sort_factor(self):
      for i in xrange(0, len(self.vars) - 1):
            for j in xrange(0, len(self.vars) - 1 - i):
                  if (self.vars[i] > self.vars[i + 1]):
                        # swap variables
                        tmp = self.vars[i]
                        self.vars[i] = self.vars[i + 1]
                        self.vars[i + 1] = tmp
                        # swap axes
                        self.pTable = np.swapaxes(self.pTable, i, i + 1)


# restrict()
# f: Factor object
# variable: Restricted variable
# value: Variable's value
```

```
@staticmethod
def restrict(f, variable, value):
    # Create new factor object & copy
    fN = f.copy()
    # Find index of variable
    idxVar = fN.vars.index(variable)
    # Construct sliceObjTuple
    sliceObjTuple = ()
    for i in xrange(0, idxVar):
        vals = fN.idInfo[fN.vars[i]]
        sliceObjTuple += slice(None),
    # Process target variable's value
    vals = fN.idInfo[fN.vars[idxVar]]
    idxVal = vals.index(value)
    sliceObjTuple += slice(idxVal, idxVal + 1),
    # 1. Slice vars
    var = fN.vars[idxVar]
    del fN.vars[idxVar]
    # 2. Slice vals
    del fN.idInfo[var]
    # 3. Slice pTable
    fN.pTable = fN.pTable[sliceObjTuple]
    rankTable = ()
    for i in xrange(0, len(fN.vars)):
        vals = fN.idInfo[fN.vars[i]]
        curTuple = (len(vals),)
        rankTable = rankTable + curTuple
    fN.pTable = fN.pTable.reshape(rankTable)
    return fN
```

```
# multiply()
# fl: Left Factor object
# fr: Right Factor object
@staticmethod
def multiply(fl, fr):
    # Sort each factor
    fl.sort_factor()
    fr.sort_factor()
    # Calculate common variables
    commonVars = list()
    for var in fl.vars:
        if var in fr.vars:
            commonVars.append(var)
    for var in fr.vars:
        if var in fl.vars:
            if not(var in commonVars):
                commonVars.append(var)
    # Calculate union variables
    unionVars = list(fl.vars)
    for var in fr.vars:
        if not (var in commonVars):
            unionVars.append(var)
    # Sort both lists
    commonVars.sort()
    unionVars.sort()

    # Check each variable
    flTuple = ()
    frTuple = ()
    for var in unionVars:
```

```
            if var in fl.vars:
                flTuple += (len(fl.idInfo[var]), )
            else:
                flTuple += (1, )
            if var in fr.vars:
                frTuple += (len(fr.idInfo[var]), )
            else:
                frTuple += (1, )
        # Reshape
        pTableL = fl.pTable.reshape(flTuple)
        pTableR = fr.pTable.reshape(frTuple)

        # Create new factor object
        fN = Factor([],[],[])
        fN.pTable = pTableL * pTableR
        fN.vars = unionVars
        fN.idInfo = dict()
        for var in unionVars:
            if var in fl.idInfo:
                fN.idInfo[var] = list(fl.idInfo[var])
            if var in fr.idInfo:
                fN.idInfo[var] = list(fr.idInfo[var])
        return fN

# sumout()
# f: Factor object
# variable: Summout variable
@staticmethod
def sumout(f, variable):
        # Create new factor object & copy
```

```
        fN = f.copy()
        # Get var index & update variables list
        varIdx = fN.vars.index(variable)
        del fN.vars[varIdx]
        # Update idInfo
        del fN.idInfo[variable]
        # Update pTable
        fN.pTable = fN.pTable.sum(axis = varIdx)
        return fN

# normalize()
# f: Factor object
@staticmethod
def normalize(f):
        # Create new factor object & copy
        fN = f.copy()
        # Normalize
        sum = fN.pTable.sum()
        fN.pTable = fN.pTable / sum
        return fN

# inference()
# fList: List of Factor objects
# queryVars: query Variables
# orderedHiddenVarsList: List of strings of Variable
# evidenceList: Dict of Variable : Value
@staticmethod
def inference(fList, queryVars, orderedHiddenVarsList, evidenceList):
        # Restrict by evidence
        for e in evidenceList:
```

```
        fListN = list()
        for factor in fList:
            if (e in factor.vars):
                fRestrict = Factor.restrict(factor, e, evidenceList[e])
                print 'Restrict:',
                print e
                fRestrict.print_table()
                fListN.append(fRestrict)
            else:
                fListN.append(factor)
        # Update factor list
        fList = fListN

# Elimination
for hV in orderedHiddenVarsList:
        fListM = list() # list of factors needed to be multiplied
        fListNM = list() # list of factors not needed to be multiplied
        # Split
        for factor in fList:
            if (hV in factor.vars):
                fListM.append(factor)
            else:
                fListNM.append(factor)

        # Multiply all
        fProduct = reduce(Factor.multiply, fListM)
        print 'Multiply:',
        print hV
        fProduct.print_table()
        # Sumout
```

```
            fSumout = Factor.sumout(fProduct, hV)
            print 'Sumout:',
            print hV
            fSumout.print_table()
            # Update factor list
            fList = fListNM
            fList.append(fSumout)

        # The remaining factors only refer to query variable
        # Take product & normalize
        fProduct = reduce(Factor.multiply, fList)
        print 'Last Multiply:'
        fProduct.print_table()
        print 'Normalize'
        fResult = Factor.normalize(fProduct)
        return fResult
```

q2b1.py

```
from factor import Factor

def q2b1():
    f1 = Factor(['Trav'], \
    [['t', 'f']], \
    [0.05, 0.95])

    f2 = Factor(['Fraud', 'Trav'], \
    [['t', 'f'], ['t', 'f']], \
    [0.01, 0.004, 0.99, 0.996])
```

```
    fL = [f1, f2]
    qL = ['Fraud']
    hL = ['Trav']
    eL = dict()
    fRes = Factor.inference(fL, qL, hL, eL)
    fRes.print_table()

q2b1()
```

q2b2.py

```
from factor import Factor

def q2b2():
    f1 = Factor(['Trav'], \
    [['t', 'f']], \
    [0.05, 0.95])

    f2 = Factor(['Fraud', 'Trav'], \
    [['t', 'f'], ['t', 'f']], \
    [0.01, 0.004, 0.99, 0.996])

    f3 = Factor(['FP', 'Fraud', 'Trav'], \
    [['t', 'f'], ['t', 'f'], ['t', 'f']], \
    [0.9, 0.1, 0.9, 0.01, 0.1, 0.9, 0.1, 0.99])

    f4 = Factor(['IP', 'Fraud', 'OC'], \
    [['t', 'f'], ['t', 'f'], ['t', 'f']], \
    [0.15, 0.051, 0.1, 0.001, 0.85, 0.949, 0.9, 0.999])
```

```
    f5 = Factor(['CRP', 'OC'], \
    [['t', 'f'], ['t', 'f']], \
    [0.1, 0.01, 0.9, 0.99])

    f6 = Factor(['OC'], \
    [['t', 'f']], \
    [0.8, 0.2])

    fL = [f1, f2, f3, f4, f5, f6]
    qL = ['Fraud']
    hL = ['Trav', 'OC']
    eL = dict(FP = 't', IP = 'f', CRP = 't')
    fRes = Factor.inference(fL, qL, hL, eL)
    fRes.print_table()

q2b2()
```

q2c.py

```
from factor import Factor

def q2c():
    f1 = Factor(['Trav'], \
    [['t', 'f']], \
    [0.05, 0.95])

    f2 = Factor(['Fraud', 'Trav'], \
    [['t', 'f'], ['t', 'f']], \
    [0.01, 0.004, 0.99, 0.996])
```

```
    f3 = Factor(['FP', 'Fraud', 'Trav'], \
    [['t', 'f'], ['t', 'f'], ['t', 'f']], \
    [0.9, 0.1, 0.9, 0.01, 0.1, 0.9, 0.1, 0.99])

    f4 = Factor(['IP', 'Fraud', 'OC'], \
    [['t', 'f'], ['t', 'f'], ['t', 'f']], \
    [0.15, 0.051, 0.1, 0.001, 0.85, 0.949, 0.9, 0.999])

    f5 = Factor(['CRP', 'OC'], \
    [['t', 'f'], ['t', 'f']], \
    [0.1, 0.01, 0.9, 0.99])

    f6 = Factor(['OC'], \
    [['t', 'f']], \
    [0.8, 0.2])

    fL = [f1, f2, f3, f4, f5, f6]
    qL = ['Fraud']
    hL = ['OC']
    eL = dict(FP = 't', IP = 'f', CRP = 't', Trav = 't')
    fRes = Factor.inference(fL, qL, hL, eL)
    fRes.print_table()

q2c()
```

q2d.py

```
from factor import Factor

def q2d():
```

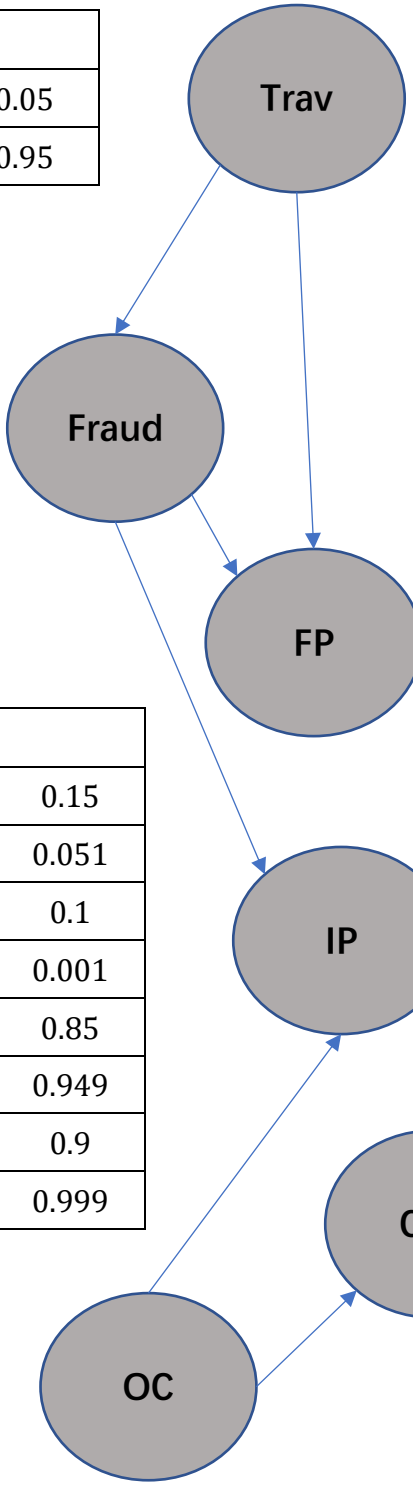
```
f1 = Factor(['Trav'], \

[['t', 'f']], \

[0.05, 0.95])


f2 = Factor(['Fraud', 'Trav'], \

[['t', 'f'], ['t', 'f']], \

[0.01, 0.004, 0.99, 0.996])


f3 = Factor(['FP', 'Fraud', 'Trav'], \

[['t', 'f'], ['t', 'f'], ['t', 'f']], \

[0.9, 0.1, 0.9, 0.01, 0.1, 0.9, 0.1, 0.99])


f4 = Factor(['IP', 'Fraud', 'OC'], \

[['t', 'f'], ['t', 'f'], ['t', 'f']], \

[0.15, 0.051, 0.1, 0.001, 0.85, 0.949, 0.9, 0.999])


f5 = Factor(['CRP', 'OC'], \

[['t', 'f'], ['t', 'f']], \

[0.1, 0.01, 0.9, 0.99])


f6 = Factor(['OC'], \

[['t', 'f']], \

[0.8, 0.2])


fL = [f1, f2, f3, f4, f5, f6]

qL = ['Fraud']

hL = ['Trav', 'FP', 'OC', 'CRP']

eL = dict(IP = 't')

fRes = Factor.inference(fL, qL, hL, eL)

fRes.print_table()
```

```
    print('==========================================')


    fL = [f1, f2, f3, f4, f5, f6]

    qL = ['Fraud']

    hL = ['Trav', 'FP', 'OC']

    eL = dict(IP = 't', CRP = 't')

    fRes = Factor.inference(fL, qL, hL, eL)

    fRes.print_table()


q2d()
```

2 (a).

f_1 = **Pr(Trav)**

Trav	
True	0.05
False	0.95



f_2 = **Pr(Fraud | Trav)**

Fraud	Trav	
True	True	0.01
True	False	0.004
False	True	0.99
False	False	0.996

f_3 = **Pr(FP | Fraud, Trav)**

FP	Fraud	Trav	
True	True	True	0.9
True	True	False	0.1
True	False	True	0.9
True	False	False	0.01
False	True	True	0.1
False	True	False	0.9
False	False	True	0.1
False	False	False	0.99

f_4 = **Pr(IP | Fraud, OC)**

IP	Fraud	OC	
True	True	True	0.15
True	True	False	0.051
True	False	True	0.1
True	False	False	0.001
False	True	True	0.85
False	True	False	0.949
False	False	True	0.9
False	False	False	0.999

f_5 = **Pr(CRP | OC)**

CRP	OC	
True	True	0.1
True	False	0.01
False	True	0.9
False	False	0.99

f_6 = **Pr(OC)**

OC	
True	0.8
False	0.2

2 (b) (1).

Factors: $f_1(\text{Trav}), f_2(\text{Fraud}, \text{Trav})$

Query: Pr(Fraud)

Evidence: $\emptyset$

Elim. Order: Trav

Restriction: None

Step 1: Add $f_7(\text{Fraud}) = \sum_{Trav} f_2(\text{Fraud}, \text{Trav}) f_1(\text{Trav})$

Remove $f_2(\text{Fraud}, \text{Trav})$, $f_1(\text{Trav})$

Last Factor: $f_7(\text{Fraud})$. This is normalized factor, so Pr(Fraud)= f_7 (Fraud).

Elimination procedure:

Multiply f_1 and f_2 with **Trav** and sumout to f_7

Fraud	
True	0.0043
False	0.9957

$\Pr(\text{Fraud}) = 0.0043 = 0.43\%$

2 (b) (2).

Factors: $f_1(\text{Trav})$, $f_2(\text{Fraud}, \text{Trav})$, $f_3(\text{FP}, \text{Fraud}, \text{Trav})$, $f_4(\text{IP}, \text{Fraud}, \text{OC})$, $f_5(\text{CRP}, \text{OC})$, $f_6(\text{OC})$

Query: $\Pr(\text{Fraud}|\text{FP} = \text{true}, \text{IP} = \text{false}, \text{CRP} = \text{true})$

Evidence: FP = true, IP = false, CRP = true

Elim. Order: Trav, OC

Restriction: Replace $f_3(\text{FP}, \text{Fraud}, \text{Trav})$ with $f_7(\text{Fraud}, \text{Trav}) = f_3(\text{FP} = \text{true}, \text{Fraud}, \text{Trav})$; Replace $f_4(\text{IP}, \text{Fraud}, \text{OC})$ with $f_8(\text{Fraud}, \text{OC}) = f_4(\text{IP} = \text{false}, \text{Fraud}, \text{OC})$; Replace $f_5(\text{CRP}, \text{OC})$ with $f_9(\text{OC}) = f_5(\text{CRP} = \text{true}, \text{OC})$.

Step 1: Add $f_{10}(\text{Fraud}) = \sum_{Trav} f_7(\text{Fraud}, \text{Trav})\, f_2(\text{Fraud}, \text{Trav}) f_1(\text{Trav})$

Remove $f_7(\text{Fraud}, \text{Trav})$, $f_2(\text{Fraud}, \text{Trav})$, $f_1(\text{Trav})$

Step 2: Add $f_{11}(\text{Fraud}) = \sum_{OC} f_8(\text{Fraud}, \text{OC})\, f_9(\text{OC}) f_6(\text{OC})$

Remove $f_8(\text{Fraud}, \text{OC})$, $f_9(\text{OC})$, $f_6(\text{OC})$

Last Factor: $f_{10}(\text{Fraud})$, $f_{11}(\text{Fraud})$. This product $f_{10}(\text{Fraud}) * f_{11}(\text{Fraud})$ is unnormalized posterior. So, $\Pr(\text{Fraud}|\text{FP} = \text{true}, \text{IP} = \text{false}, \text{CRP} = \text{true}) = \alpha f_{10}(\text{Fraud}) * f_{11}(\text{Fraud})$.

Elimination procedure:

Restrict **FP** in f_3 to f_7

Fraud	Trav	
True	True	0.9
True	False	0.1
False	True	0.9
False	False	0.01

Restrict **IP** in f_4 to f_8

Fraud	OC	
True	True	0.85
True	False	0.949
False	True	0.9
False	False	0.999

Restrict **CRP** in f_5 to f_9

OC	
True	0.1
False	0.01

Multiply all factors with **Trav** and Sumout to f_{10}

Fraud	
True	0.00083
False	0.054012

Multiply all factors with **OC** and sumout to f_{11}

Fraud	
True	0.069898
False	0.073998

Multiply the rest factors:

Fraud	
True	$5.8 * 10^{-5}$
False	0.004

After normalizing:

Fraud	
True	0.0143
False	0.9857

$\Pr(\text{Fraud}|\text{FP} = \text{true}, \text{IP} = \text{false}, \text{CRP} = \text{true}) = 0.0143 = 1.43\%$

2 (c).

Factors: $f_1(\text{Trav})$, $f_2(\text{Fraud}, \text{Trav})$, $f_3(\text{FP}, \text{Fraud}, \text{Trav})$, $f_4(\text{IP}, \text{Fraud}, \text{OC})$, $f_5(\text{CRP}, \text{OC})$, $f_6(\text{OC})$

Query: $\Pr(\text{Fraud}|\text{FP} = \text{true}, \text{IP} = \text{false}, \text{CRP} = \text{true}, \text{Trav} = \text{true})$

Evidence: FP = true, IP = false, CRP = true, Trav = true

Elim. Order: OC

Restriction: Replace $f_3(\text{FP}, \text{Fraud}, \text{Trav})$ with $f_7(\text{Fraud}, \text{Trav}) = f_3(\text{FP} = \text{true}, \text{Fraud}, \text{Trav})$; Replace $f_4(\text{IP}, \text{Fraud}, \text{OC})$ with $f_8(\text{Fraud}, \text{OC}) = f_4(\text{IP} = \text{false}, \text{Fraud}, \text{OC})$; Replace $f_5(\text{CRP}, \text{OC})$ with $f_9(\text{OC}) = f_5(\text{CRP} = \text{true}, \text{OC})$; Replace $f_2(\text{Fraud}, \text{Trav})$ with $f_{10}(\text{Fraud}) = f_2(\text{Fraud}, \text{Trav} = \text{true})$; Replace $f_7(\text{Fraud}, \text{Trav})$ with $f_{11}(\text{Fraud}) =$

f_7(Fraud, Trav = true).

Step 1: Add $f_{12}(\text{Fraud}) = \sum_{OC} f_8(\text{Fraud}, \text{OC})\, f_9(\text{OC})$

Remove $f_8(\text{Fraud}, \text{OC})$, $f_9(\text{OC})$

Last Factor: $f_{10}(\text{Fraud})$, $f_{11}(\text{Fraud})$, $f_{12}(\text{Fraud})$. This product $f_{10}(\text{Fraud}) * f_{11}(\text{Fraud}) * f_{12}(\text{Fraud})$ is unnormalized posterior. So, Pr(Fraud|FP = true, IP = false, CRP = true, Trav = true)= $\alpha f_{10}(\text{Fraud}) * f_{11}(\text{Fraud}) * f_{12}(\text{Fraud})$.

Elimination procedure:

Restrict **FP** in f_3 to f_7

Fraud	Trav	
True	True	0.9
True	False	0.1
False	True	0.9
False	False	0.01

Restrict **IP** in f_4 to f_8

Fraud	OC	
True	True	0.85
True	False	0.949
False	True	0.9
False	False	0.999

Restrict **CRP** in f_5 to f_9

OC	
True	0.1
False	0.01

Restrict **Trav** in f_2 to f_{10}

Fraud	
True	0.01
False	0.99

Restrict **Trav** in f_7 to f_{11}

Fraud	
True	0.9
False	0.9

Multiply all factors with **OC** and sumout to f_{12}

Fraud	
True	0.069898
False	0.073998

Multiply the rest factors:

Fraud	
True	$3.1 * 10^{-5}$
False	0.0033

After normalizing:

Fraud	
True	0.00945
False	0.99055

$\Pr(\text{Fraud}|\text{FP} = \text{true}, \text{IP} = \text{false}, \text{CRP} = \text{true}, \text{Trav} = \text{true}) = 0.00945 = 0.945\%$

2 (d).

The action taken is **making an unimportant purchase over internet** first, which can make CRP = true.

When not doing so, the CRP is unknown, this will generate the following query:

Factors: $f_1(\text{Trav})$, $f_2(\text{Fraud}, \text{Trav})$, $f_3(\text{FP}, \text{Fraud}, \text{Trav})$, $f_4(\text{IP}, \text{Fraud}, \text{OC})$, $f_5(\text{CRP}, \text{OC})$, $f_6(\text{OC})$

Query: $\Pr(\text{Fraud}|\text{IP} = \text{true})$

Evidence: IP = true

Elim. Order: Trav, FP, OC, CRP

Restriction: Replace $f_4(\text{IP}, \text{Fraud}, \text{OC})$ with $f_7(\text{Fraud}, \text{OC}) = f_4(\text{IP} = \text{true}, \text{Fraud}, \text{OC})$.

Step 1: Add $f_8(\text{FP}, \text{Fraud}) = \sum_{Trav} f_1(\text{Trav})\, f_2(\text{Fraud}, \text{Trav}) f_3(\text{FP}, \text{Fraud}, \text{Trav})$

Remove $f_1(\text{Trav})$, $f_2(\text{Fraud}, \text{Trav})$, $f_3(\text{FP}, \text{Fraud}, \text{Trav})$

Step 2: Add $f_9(\text{Fraud}) = \sum_{FP} f_8(\text{FP}, \text{Fraud})$

Remove $f_8(\text{FP}, \text{Fraud})$

Step 3: Add $f_{10}(\text{CRP}, \text{Fraud}) = \sum_{OC} f_7(\text{Fraud}, \text{OC}) f_5(\text{CRP}, \text{OC}) f_6(\text{OC})$

Remove $f_7(\text{Fraud}, \text{OC})$, $f_5(\text{CRP}, \text{OC})$, $f_6(\text{OC})$

Step 4: Add $f_{11}(\text{Fraud}) = \sum_{CRP} f_{10}(\text{CRP}, \text{Fraud})$

Remove $f_{10}(\text{CRP}, \text{Fraud})$

Last Factor: $f_9(\text{Fraud})$, $f_{11}(\text{Fraud})$. This product $f_9(\text{Fraud}) * f_{11}(\text{Fraud})$ is unnormalized posterior. So, $\Pr(\text{Fraud}|\text{IP} = \text{true}) = \alpha f_9(\text{Fraud}) * f_{11}(\text{Fraud})$.

Elimination procedure:

Restrict **IP** in f_4 to f_7

Fraud	OC	
True	True	0.15
True	False	0.051
False	True	0.1
False	False	0.001

Multiply all factors with **Trav** and sumout to f_8

FP	Fraud	
True	True	0.00083
True	False	0.054012
False	True	0.00347
False	False	0.941688

Multiply all factors with **FP** and sumout to f_9

Fraud	
True	0.0043
False	0.9957

Multiply all factors with **OC** and sumout to f_{10}

CRP	Fraud	
True	True	0.012102
True	False	0.008002
False	True	0.118098
False	False	0.072198

Multiply all factors with **CRP** and sumout to f_{11}

Fraud	
True	0.1302
False	0.0802

Multiply the rest factors:

Fraud	
True	0.00056
False	0.07986

After normalizing:

Fraud	
True	0.00696213393
False	0.99303786607

$\Pr(\text{Fraud}|\text{IP} = \text{true}) = 0.00696213393 = 0.696213393\%$

When doing so, the CRP is true, this will generate the following query:

Factors: $f_1(\text{Trav})$, $f_2(\text{Fraud}, \text{Trav})$, $f_3(\text{FP}, \text{Fraud}, \text{Trav})$, $f_4(\text{IP}, \text{Fraud}, \text{OC})$, $f_5(\text{CRP}, \text{OC})$, $f_6(\text{OC})$

Query: $\Pr(\text{Fraud}|\text{IP} = \text{true}, \text{CRP} = \text{true})$

Evidence: $\text{IP} = \text{true}, \text{CRP} = \text{true}$

Elim. Order: Trav, FP, OC

Restriction: Replace $f_4(\text{IP}, \text{Fraud}, \text{OC})$ with $f_7(\text{Fraud}, \text{OC}) = f_4(\text{IP} = \text{true}, \text{Fraud}, \text{OC})$; Replace $f_5(\text{CRP}, \text{OC})$ with $f_8(\text{OC}) = f_5(\text{CRP} = \text{true}, \text{OC})$

Step 1: Add $f_9(\text{FP}, \text{Fraud}) = \sum_{Trav} f_1(\text{Trav})\, f_2(\text{Fraud}, \text{Trav}) f_3(\text{FP}, \text{Fraud}, \text{Trav})$

Remove $f_1(\text{Trav})$, $f_2(\text{Fraud}, \text{Trav})$, $f_3(\text{FP}, \text{Fraud}, \text{Trav})$

Step 2: Add $f_{10}(\text{Fraud}) = \sum_{FP} f_9(\text{FP}, \text{Fraud})$

Remove $f_9(\text{FP}, \text{Fraud})$

Step 3: Add $f_{11}(\text{Fraud}) = \sum_{OC} f_7(\text{Fraud}, \text{OC}) f_8(\text{OC}) f_6(\text{OC})$

Remove $f_7(\text{Fraud}, \text{OC})$, $f_8(\text{OC})$, $f_6(\text{OC})$

Last Factor: $f_{10}(\text{Fraud})$, $f_{11}(\text{Fraud})$. This product $f_{10}(\text{Fraud}) * f_{11}(\text{Fraud})$ is unnormalized posterior. So, $\Pr(\text{Fraud}|\text{IP} = \text{true}, \text{CRP} = \text{true}) = \alpha f_{10}(\text{Fraud}) * f_{11}(\text{Fraud})$.

Elimination procedure:

Restrict **IP** in f_4 to f_7

Fraud	OC	
True	True	0.15
True	False	0.051
False	True	0.1
False	False	0.001

Restrict **CRP** in f_5 to f_8

OC	
True	0.1
False	0.01

Multiply all factors with **Trav** and sumout to f_9

FP	Fraud	
True	True	0.00083
True	False	0.054012
False	True	0.00347
False	False	0.941688

Multiply all factors with **FP** and sumout to f_{10}

Fraud	
True	0.0043
False	0.9957

Multiply all factors with **OC** and sumout to f_{11}

Fraud	
True	0.012102
False	0.008002

Multiply the rest factors:

Fraud	
True	$5.2 * 10^{-6}$
False	0.079676

After normalizing:

Fraud	
True	0.00648890285
False	0.99351109715

$$\Pr(\text{Fraud}|\text{IP} = \text{true}, \text{CRP} = \text{true}) = 0.00648890285 = 0.648890285\%$$

Above all,

$$\Pr(\text{Fraud}|\text{IP} = \text{true}) - \Pr(\text{Fraud}|\text{IP} = \text{true}, \text{CRP} = \text{true})$$

$$= 0.696213393\% - 0.648890285\%$$

$$= 0.047323108\%$$

The probability of a fraud gets reduced is 0.047323108%

3 (a)

(i) No, D and G are dependent. There is a path from D to G which is not blocked since there is no evidence in that path. So, D and G are dependent.

(ii) No, D and G are dependent. There is an evidence F in one path from D to G, but there is another path from D to G that is not blocked. So, D and G are dependent.

(iii) Yes, A and G are independent. The indirect path from A to A and other indirect paths from G to C enter B, but there is no evidence for B, and B does not have descendent. So, B blocks all paths between A and G, and they are d-separated. So, A and G are independent.

(iv) No, A and G are dependent. The indirect path from G to D goes into C, and the indirect path from A to B goes out C. C is not in the evidence set, so this path between A and G is not blocked. So, A and G are dependent.

(v) Yes, A and G are independent. Firstly, the indirect path from A to B leaves C, and indirect path from G to D enters C. Also, there is an evidence for C. So, this path is blocked. Secondly, the indirect path from A to B and the indirect path from G to E both leave C, and there is an evidence for C. So, this path is also blocked. Thus, paths between A and G are blocked, and they are d-separated. So, A and G are independent.

(vi) Yes, A and G are independent. Firstly, the indirect path from A to C and the indirect path from G to F both leave D, and there is an evidence for D. So, this path is blocked. Secondly, the indirect path from A to C and the indirect path from G to F both enter E, and there is no evidence for E. So, this path is also blocked. Thus, paths between A and G are blocked, and they are d-separated. So, A and G are independent.

(vii) No, A and G are dependent. The indirect path from G to F and the indirect path from A to C both enter E. E is in the evidence set, so this path between A and G is not blocked. So, A and G are dependent.

3 (b)

(1) C is relevant since C is the query variable.

(2) D is relevant since D is the parent of C, and C is relevant.

(3) E is relevant since E is in evidence set, and E is the descendent of a relevant node C.

(4) F is relevant since F is the parent of E, and F is relevant.

Above all, the subset of relevant variables that is sufficient to answer this query is {C, D, E, F}.