CMPSC 448: Machine Learning. Homework 7.
Due: April 27

1. Instructions

- You cannot look at anyone else's code.
- Fill in and upload hw7.py to gradescope.
- All code (except import statements) in hw7.py should be inside functions (importing homework1.py should not cause code to execute).
- Code must have comments and any constants should be stored in a variable defined near the top of your file.
- Read the instructions below carefully
- For most of the questions, we will not provide testing code. It is your responsibility to test your own functions and make sure they run without throwing exceptions.
- Another reason for you to develop testing code is a very similar assignment will be given as the take-home final.

2. Bayesian Networks

2.1. **Numeric Stability.**

In this assignment, to avoid rounding issues, you will need to use Fraction datatypes in python. Here is an example of their usage:

```
from fractions import Fraction as frac
half = frac(1,2)
```

If you choose not to use fraction datatypes, you might get a correct answer marked as incorrect due to rounding issues.

2.2. **Bayesian Network Parameters.** The parameters of a Bayesian network will be passed to your code through a parameter **bn**. It will be a class and its usage is described as follows: suppose the Bayesian network has directed edges $(a, b), (a, c), (b, d), (c, d)$. The parameters of this network are the conditional probabilities $P(a), P(b|a), P(c|a), P(d|b, c)$. The corresponding **bn** variable will be defined something like this:

```
import numpy as np
from fractions import Fraction as frac

class BayesNet1:
   def __init__(self, seed, k=10):
       prng = np.random.RandomState(seed)
       prob_a = frac(prng.randint(1, 2**k), 2**k) # P(a=1)
       prob_b = {(1,): frac(prng.randint(1, 2**k), 2**k), # P(b=1 | a=1)
                 (0,): frac(prng.randint(1, 2**k), 2**k) # P(b=1 | a=0)
                }
       prob_c = {(1,): frac(prng.randint(1, 2**k), 2**k), # P(c=1 | a=1)
                 (0,): frac(prng.randint(1, 2**k), 2**k) # P(c=1 | a=0)
                }
       prob_d = {
                 (0, 0): : frac(prng.randint(1, 2**k), 2**k), # P(d=1 | b=0, c=0)
                 (0, 1): : frac(prng.randint(1, 2**k), 2**k), # P(d=1 | b=0, c=1)
                 (1, 0): : frac(prng.randint(1, 2**k), 2**k), # P(d=1 | b=1, c=0)
                 (1, 1): : frac(prng.randint(1, 2**k), 2**k), # P(d=1 | b=1, c=1)
                }

    def a(value): #returns P(a=value)
        if value == 1:
            return prob_a
        else:
            return 1-prob_a

    def b(value, a): #returns P(b=value | a)
        tmp = prob_b[(a,)]
        if value == 1:
            return tmp
        else:
            return 1-tmp

    def c(value, a): #returns P(c=value | a)
        tmp = prob_c[(a,)]
        if value == 1:
            return tmp
        else:
            return 1-tmp

    def d(value, b,c): #returns P(d=value | b, c)
        tmp = prob_d[(b,c)]
        if value == 1:
            return tmp
        else:
            return 1-tmp


# example usage
bn = BayesNet1()
# get parameter p(a=1)
bn.a(value=1) # must call with arg names, bn.a(1) is incorrect
bn.a(value=0) # get parameter p(a=0)
bn.d(value=0, b=1, c=0) # get parameter P(d=0 | b=1, c=0)

bd.a(value=1, d=2) #throws error because P(a|d) is not a parameter
```

If a parameter is not needed for a particular problem, using it may throw an exception. For example, for the Bayesian network in the code above, $P(a, b)$ can be computed without using the parameter $P(d \mid b, c)$. That is, after you write $P(a, b)$ in terms of the network parameters and simplify, you will notice that $P(d|b, c)$ is not used at all. Thus our implementation of **bn** might not define the function **bn.d(value, b, c)**. This is used to test that you simplified the expression correctly.

2.3. **Types of Questions.** There will be two types of questions:

(1) Probability calculations: given network parameters, compute probabilities such as $P(A = 1 \mid B = 0, C = 1)$.

(2) D-separation. The questions might ask you if A is conditionally independent of D given E. You will write a function that returns **result**, **pathverdict** where **result** is the answer (are they conditionally independent?) and **pathverdict** looks like:

```
  [
    ((’a’, ’b’, ’d’), False)
    ((’a’, ’e’, ’d’), False)
  ]
```

each element in the list is a tuple. The first part of the tuple describes the path (e.g., a,e,d is an undirected path from a to d) and the second part of the tuple tells us if that path is blocked or not. Make sure **pathverdict** contains all of the appropriate undirected paths. A path cannot repeat nodes (so a e a e d is not a path). You have to hard-code the appropriate paths, and results. In other words, for d-separations questions, your functions should look like:

```
def question0():
   parthverdict = [
      ((’a’, ’b’, ’d’), False)
      ((’a’, ’e’, ’d’), False)
    ]
   result = False
   return result, pathverdict
```

In this case, this answer indicates that the first path is not blocked and the second path is not blocked.

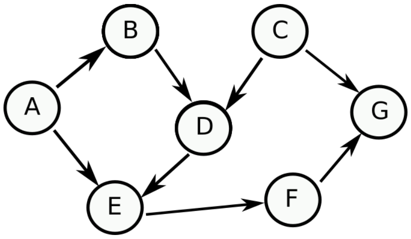

FIGURE 1. Bayesian Network 1

Question 1 (5 pts). *Fill in the function* `def question1Part1()` *that tells us if, in Figure 1,* $A \perp\!\!\!\perp D \mid G, B$*? Your function should return* ***result***, ***pathverdict*** *(see instructions on the previous page), where* ***result*** *is the True/False answer to the question, and* ***pathverdict*** *is the set of* ***all*** *paths between A and D and their status (blocked or not). This question will have a test that you can see.*

Question 2 (5 pts). *Fill in the function* `def question1Part2()` *that tells us if, in Figure 1,* $A \perp\!\!\!\perp G \mid D, F$*? Your function should return* ***result***, ***pathverdict*** *(see instructions on the previous page), where* ***result*** *is the True/False answer to the question, and* ***pathverdict*** *is the set of* ***all*** *paths between A and G and their status (blocked or not).*

Question 3 (5 pts). *Fill in the function* `def question1Part3()` *that tells us if, in Figure 1,* $A \perp\!\!\!\perp G \mid F$*? Your function should return* ***result***, ***pathverdict*** *(see instructions on the previous page), where* ***result*** *is the True/False answer to the question, and* ***pathverdict*** *is the set of* ***all*** *paths between A and G and their status (blocked or not).*

Question 4 (5 pts). *Fill in the function* `def question1Part4()` *that tells us if, in Figure 1,* $A \perp\!\!\!\perp G \mid B$*? Your function should return* ***result***, ***pathverdict*** *(see instructions on the previous page), where* ***result*** *is the True/False answer to the question, and* ***pathverdict*** *is the set of* ***all*** *paths between A and G and their status (blocked or not).*

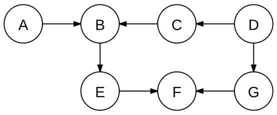

Figure 2. Bayesian Network 2

Question 5 (5 pts). *Fill in the function* `def question2Part1()` *that tells us if, in Figure 2,* $A \perp\!\!\!\perp D \mid G$*? Your function should return* ***result***, ***pathverdict*** *(see instructions), where* ***result*** *is the True/False answer to the question, and* ***pathverdict*** *is the set of* ***all*** *paths between* A *and* D *and their status (blocked or not).*

Question 6 (5 pts). *Fill in the function* `def question2Part2()` *that tells us if, in Figure 2,* $A \perp\!\!\!\perp D \mid F$*? Your function should return* ***result***, ***pathverdict*** *(see instructions), where* ***result*** *is the True/False answer to the question, and* ***pathverdict*** *is the set of* ***all*** *paths between* A *and* D *and their status (blocked or not).*

Question 7 (5 pts). *Fill in the function* `def question2Part3()` *that tells us if, in Figure 2,* $B \perp\!\!\!\perp G \mid E, D$*? Your function should return* ***result***, ***pathverdict*** *(see instructions), where* ***result*** *is the True/False answer to the question, and* ***pathverdict*** *is the set of* ***all*** *paths between* A *and* D *and their status (blocked or not).*

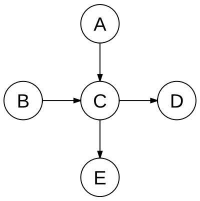

Figure 3. Bayesian Network 3

Question 8 (4 pts). *This question uses the Bayesian Network in Figure 3.*
Fill in the function `def question3Part1(a,b,c,d, bn)` *that returns* $P(a, b, c, d)$ *in this network. The parameter bn gives you the network parameters (see instructions). For this question, we will give 2 test cases. In 1 test case, bn will provide all of the network parameters and in 1 test case, bn will only provide the parameters that are absolutely needed to compute this answer.*

Question 9 (4 pts). *This question uses the Bayesian Network in Figure 3.*
Fill in the function `def question3Part2(a,b,c, bn)` *that returns* $P(a, b, c)$ *in this network. The parameter bn gives you the network parameters (see instructions).*

Question 10 (4 pts). *This question uses the Bayesian Network in Figure 3.*
Fill in the function `def question3Part3(d, bn)` *that returns* $P(d)$ *in this network. The parameter bn gives you the network parameters (see instructions).*

Question 11 (4 pts). *This question uses the Bayesian Network in Figure 3.*
Fill in the function `def question3Part4(a,b,c, bn)` *that returns* $P(a, b|c)$ *in this network. The parameter bn gives you the network parameters (see instructions). For this question, we will give 2 test cases. In 1 test case, bn will provide all of the network parameters and in 1 test case, bn will only provide the parameters that are absolutely needed to compute this answer.*

Question 12 (4 pts). *This question uses the Bayesian Network in Figure 3.*
Fill in the function `def question3Part4(c, d, bn)` *that returns* $P(c|d)$ *in this network. The parameter bn gives you the network parameters (see instructions).*

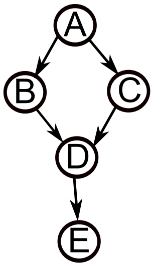

Figure 4. Bayesian Network 4

Question 13 (4 pts). *This question uses the Bayesian Network in Figure 4.*
Fill in the function `def question4Part1(b,c,d, bn)` *that returns* $P(b, c, d)$ *in this network. The parameter bn gives you the network parameters (see instructions).*

Question 14 (4 pts). *This question uses the Bayesian Network in Figure 4.*
Fill in the function `def question4Part2(c, d, bn)` *that returns* $P(d|c)$ *in this network. The parameter bn gives you the network parameters (see instructions). note that the variables given to your function are in alphabetical order (c first then d) but the answer should return* $P(d|c)$

Question 15 (4 pts). *This question uses the Bayesian Network in Figure 4.*
Fill in the function `def question4Part3(d, e, bn)` *that returns* $P(d \mid e)$ *in this network. The parameter bn gives you the network parameters (see instructions).*