Assignment 2

Due: 19. November 2023, 23:59 CET

General information: This assignment should be completed in groups of **three to four people**. Submit your assignment via the corresponding Moodle activity. You can only upload one .zip file, therefore make sure to place all your files required to run/evaluate your solution into **one archive** and just upload that archive.

Group information: In this assignment, you will build upon the graph class from the first assignment and implement functions required for performing independence checks for variables. For the first two exercises please write your solution into either a .pdf or .txt file and submit your solution alongside the programming exercise in one .zip file.
The *assignment2.py* file contains the skeletons for the functions that you should implement for this assignment. You will find a reference graph implementation in the *ccbase* package in the *assignment2.zip* alongside the skeleton file. The graph implementation can be found in *ccbase/networks.py*, while the used node implementation is located in *ccbase/nodes.py*. For any function that can take either a Node object or a Node's name as string as an argument, you only need to ensure that your solution works with one or the other, although you can also implement it in such a way that works with both. You are free to use your own graph class from the first assignment instead, however, incorrect behavior of the assignment's tasks due to bugs in your graph class may still lead to point reductions.

Python exercise information: You are only allowed to use the external libraries that we provide or explicitly mention on the assignment sheet (or that are already within the provided skeleton files). Your solutions' functionality will all be tested against automated tests before being reviewed further. **You need to ensure** that your solution can directly be **imported as a Python3 module** for automatic tests and at least contains the name of the skeleton file. The easiest way to achieve this, is to directly develop your solution in the assignment2.py file and submit that. The Moodle contains information about the test environment. You will also find an example test-file alongside the skeleton file that you can use to ensure that your solution works in the evaluation environment.

Exercise 1: (5 Points)

Bayes' theorem is an important prerequisite for a lot of methods dealing with uncertain data. Given the likelihood $P(B|A)$, a prior belief $P(A)$, and the marginal likelihood $P(B)$ the posterior belief $P(A|B)$ can be calculated:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes' theorem one has not to rely on mere truth/false statements anymore, as is the case in most traditional AI approaches. Instead, probability values that represent ones degrees of belief, are assigned to statements. Bayes' theorem can then be used to calculate how this degree of belief changes under incoming new information that itself has a specific uncertainty.

Consider the following classical example:

`It is raining:`	$p(A = true)$	$= 0.4$
`It is not raining:`	$p(A = false)$	$= 0.6$
`The lawn is wet, if it rains:`	$p(B = true \| A = true)$	$= 0.98$
`The lawn is not wet, if it rains:`	$p(B = false \| A = true)$	$= 0.02$
`The lawn is wet, if it does not rain:`	$p(B = true \| A = false)$	$= 0.15$
`The lawn is not wet, it it does not rain:`	$p(B = false \| A = false)$	$= 0.85$

Task 1: Calculate by hand: $p(B = false)$

Task 2: Calculate by hand: $p(A = true|B = true)$

Exercise 2: (5 Points)

Add the following statements to the previous example:

`The sun is shinning, if it rains:`	$p(C = true \| A = true)$	$= 0.02$
`The sun is not shining, if it rains:`	$p(C = false \| A = true)$	$= 0.98$
`The sun is shining, if it does not rain:`	$p(C = true \| A = false)$	$= 0.7$
`The sun is not shining, if it does not rain:`	$p(C = false \| A = false)$	$= 0.3$

You can assume that $B \perp\!\!\!\perp C|A$, i.e. B and C are independent of each other in the presence of A.

Task 1: Calculate by hand: $p(A = true|B = false, C = true)$

Task 2: Calculate by hand: $p(B = true|C = false)$

Task 3: Calculate by hand: $p(C = false|B = true)$

Exercise 3: (3 Points)

Causal structures like *forks, chains* and *colliders* can be used in a Bayesian network to test for conditional independence given evidence. Implement the following methods:

Task 1: (1 Point) *find_forks(dg)*: Returns a list of nodes that are forks in the directed graph *dg*.

Task 2: (1 Point) *find_chains(dg)*: Returns a list of nodes that are chains in the directed graph *dg*.

Task 3: (1 Point) *find_collider(dg)*: Returns a list of nodes that are collider in the directed graph *dg*.

Exercise 4: (4 Points)

As mentioned in the lecture, two different directed graphs can make the same (conditional) independence statements. Implement the following functions in order to check if two graphs are *Markov equivalent*:

Task 1: (2 Points) *find_immoralities(graph)*: Returns all immoralities (i.e. two nodes with the same child but no direct connection) of the given graph.

Task 2: (1 Point) *same_skeleton(graph1, graph2)*: Returns true if the two graphs have the same skeleton as one another.

Task 3: (1 Point) *markov_equivalent(graph1, graph2)*: Returns true if the two graphs are Markov equivalent and false otherwise.

Exercise 5: (3 Points)

Implement the function *get_paths(graph, node_x, node_y)* which computes all *undirected* paths (where each node on the path is only visited once) between node X and node Y in the given (directed or un-directed) graph and returns these as a list of lists.