```
# Ejercicios de redes bayesianas con Python (Trabajo calificado)
# Realizado por: Correa Adrian
# Fecha: 18/05/2025


#link de GitHub:https://github.com/afca2002/Ejercicios-de-redes-bayesianas-con-Python-Trabajo-calificado-.git

 # ------------------------------------------------------------
# Pregunta 1. Implementar la red bayesiana para el problema
# de la alarma usando el código proporcionado en Python.
#
# Resolver las siguientes probabilidades:
#
# - P(j, m, ~a, b, ~e)
# - P(~a, b, e)
# - P(b | a)
#
# Además:
# - Calcular la probabilidad de b dado que llama John,
#   considerando 10,000 ocurrencias (simulaciones).
# ------------------------------------------------------------

from pomegranate import *
import random

# Nodos de la red bayesiana
# Burglary (robo) tiene una distribución discreta
burglary = Node(DiscreteDistribution({
    "b": 0.001,
    "~b": 0.999
}), name="burglary")

# Earthquake (terremoto) tiene una distribución discreta
earthquake = Node(DiscreteDistribution({
    "e": 0.002,
    "~e": 0.998
}), name="earthquake")

# Alarm (alarma) es condicional a Burglary y Earthquake
alarm = Node(ConditionalProbabilityTable([
    ["b", "e", "a", 0.95],
    ["b", "e", "~a", 0.05],
    ["b", "~e", "a", 0.94],
    ["b", "~e", "~a", 0.06],
    ["~b", "e", "a", 0.29],
    ["~b", "e", "~a", 0.71],
    ["~b", "~e", "a", 0.001],
    ["~b", "~e", "~a", 0.999]
], [burglary.distribution, earthquake.distribution]), name="alarm")

# JohnCalls (John llama) es condicional a Alarm
john_calls = Node(ConditionalProbabilityTable([
    ["a", "j", 0.9],
    ["a", "~j", 0.1],
    ["~a", "j", 0.05],
    ["~a", "~j", 0.95]
], [alarm.distribution]), name="john_calls")

# MaryCalls (Mary llama) es condicional a Alarm
mary_calls = Node(ConditionalProbabilityTable([
    ["a", "m", 0.7],
    ["a", "~m", 0.3],
    ["~a", "m", 0.01],
    ["~a", "~m", 0.99]
], [alarm.distribution]), name="mary_calls")

# Crear la red bayesiana y agregar los nodos
model = BayesianNetwork("Alarm Problem")
model.add_states(burglary, earthquake, alarm, john_calls, mary_calls)
```

```python
# Agregar las conexiones entre los nodos
model.add_edge(burglary, alarm)
model.add_edge(earthquake, alarm)
model.add_edge(alarm, john_calls)
model.add_edge(alarm, mary_calls)

# Finalizar el modelo
model.bake()

# Consultas de probabilidad
# P(j, m, ~a, b, ~e)
print("P(j, m, ~a, b, ~e):", model.probability([["b", "e", "~a", "j", "m"]]))

# P(~a, b, e)
print("P(~a, b, e):", model.probability([["b", "e", "~a", None, None]]))

# P(b | a)
observations = {"alarm": "a"}
predictions = model.predict_proba(observations)
burglary_prediction = predictions[0]  # El nodo "burglary" es el primero en el modelo

if isinstance(burglary_prediction, DiscreteDistribution):
    # Acceder a las probabilidades usando .parameters[0]
    print("P(b | a):", burglary_prediction.parameters[0]["b"])

# Probabilidad de b dado que llama John, para 10000 ocurrencias
john_calls_observations = {"john_calls": "j"}
count_b_given_j = 0
for _ in range(10000):
    predictions = model.predict_proba(john_calls_observations)
    burglary_prediction = predictions[0]
    if isinstance(burglary_prediction, DiscreteDistribution):
        if random.random() < burglary_prediction.parameters[0]["b"]:
            count_b_given_j += 1

print("P(b | John llama) (simulado en 10000 ocurrencias):", count_b_given_j / 10000)
```