Name: Atish Kumar(120CS0173)

Bayesian Network Lab 10

```
!pip install pomegranate==v0.14.9
```

```
Collecting pomegranate==v0.14.9
  Downloading pomegranate-0.14.9.tar.gz (4.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.7/4.7 MB 16.8 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting cython<3.0.0,>=0.22.1 (from pomegranate==v0.14.9)
  Using cached Cython-0.29.36-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_24_x86_64.whl (1.9 MB)
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.10/dist-packages (from pomegranate==v0.14.9) (1.23.5)
Requirement already satisfied: joblib>=0.9.0b4 in /usr/local/lib/python3.10/dist-packages (from pomegranate==v0.14.9) (1.3.2)
Requirement already satisfied: networkx>=2.4 in /usr/local/lib/python3.10/dist-packages (from pomegranate==v0.14.9) (3.2)
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.10/dist-packages (from pomegranate==v0.14.9) (1.11.3)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from pomegranate==v0.14.9) (6.0.1)
Building wheels for collected packages: pomegranate
  Building wheel for pomegranate (pyproject.toml) ... done
  Created wheel for pomegranate: filename=pomegranate-0.14.9-cp310-cp310-linux_x86_64.whl size=18331243 sha256=bdf54e8c17bab9e61d49
  Stored in directory: /root/.cache/pip/wheels/14/e7/b2/189a2d351ac4ae073cfa17ce9d56936d59af5712a18028fc31
Successfully built pomegranate
Installing collected packages: cython, pomegranate
  Attempting uninstall: cython
    Found existing installation: Cython 3.0.4
    Uninstalling Cython-3.0.4:
      Successfully uninstalled Cython-3.0.4
Successfully installed cython-0.29.36 pomegranate-0.14.9
```

```
import math
from pomegranate import *
```

Notations :Alarm : A Burgulary : B Earthquake : E JohnCalls : J MaryCalls : M

```
B = DiscreteDistribution({'1': 0.001, '0': 1-0.001})
E = DiscreteDistribution({'1': 0.002, '0': 1-0.002})
```

Creating Conditional Probability Table for Node B,D,E,F,G,H,I

```
A = ConditionalProbabilityTable(
 [[ '0', '0', '1', 0.001 ],
 [ '0', '1', '1', 0.29 ],
 [ '1', '0', '1', 0.94 ],
 [ '1', '1', '1', 0.95 ],
 [ '0', '0', '0', 1-0.001 ],
 [ '0', '1', '0', 1-0.29],
 [ '1', '0', '0', 1-0.94 ],
 [ '1', '1', '0', 1-0.95]],[B,E])
J = ConditionalProbabilityTable(
 [[ '0', '0', 1-0.05 ],
 [ '0', '1', 0.05 ],
 [ '1', '0', 1-0.90 ],
 [ '1', '1', 0.90 ]],[A])
M = ConditionalProbabilityTable(
 [[ '0', '0', 1-0.01 ],
 [ '0', '1', 0.01 ],
 [ '1', '0', 1-0.7 ],
 [ '1', '1', 0.7 ]],[A])
```

Creating States

```
b = State(B, name="b")
e = State(E, name="e")
a = State(A, name="a")
j = State(J,name="j")
m= State(M,name="m")
```

```
# Create the Bayesian network object
model = BayesianNetwork("Alarm-network")
# Add the states to the network
model.add_states(b,e,a,j,m)
```

Add edges to the model. The edges represent which states are parents of which other states.

```
model.add_edge(b,a)
model.add_edge(e,a)
model.add_edge(a,j)
model.add_edge(a,m)
```

Model must be baked to finalize the internals

```
model.bake()
```

Calculating the Probabilities

```
# P(b/m)
model.predict_proba([{'m': '1'}])[0][0].parameters[0]['1']
```

```
0.05611745403893674
```

```
# p(b/a,~m,j)
model.predict_proba([{'a': '1','m':'0','j':'1'}])[0][0].parameters[0]
```

```
{'1': 0.37355122828189946, '0': 0.6264487717181005}
```

```
#p(e/a,~m,j)
model.predict_proba([{'a': '1','m':'0','j':'1'}])[0][1].parameters[0]
```

```
{'1': 0.23100870196890552, '0': 0.7689912980310945}
```

Joint Probablility

```
#P(J ^ M ^ A ^ ¬B ^ ¬ E) = P(J | A) P(M | A) P(A| ¬ B ^ ¬ E) P(¬ B) P(¬E)
# (b,e,a,j,m)
#p(j/a)
p1=model.predict_proba([{'a': '1'}])[0][3].parameters[0]['1']
#p(m/a)
p2=model.predict_proba([{'a': '1'}])[0][4].parameters[0]['1']
#p(a/~b,~e)
p3=model.predict_proba([{'b': '0','e':'0'}])[0][2].parameters[0]['1']
#p(~b)
p4=model.predict_proba([{}])[0][0].parameters[0]['0']
#p(~e)
p5=model.predict_proba([{}])[0][1].parameters[0]['0']
print(p1,p2,p3,p4,p5)
prob=p1*p2*p3*p4*p5
print(prob)
```

```
0.8999999999999998 0.6999999999999998 0.0010000000000006649 0.9989999999999996 0.9979999999999996
0.0006281112600004167
```