Name: Ankit Kumar Sahu

Reg No.: 20BAI1005

Foundations of AI (CSE1014)

Lab 9

Bayesian Belief network

CODE

BN.py

```
import copy


class Node(object):
    def __init__(self, name=""):
        self.name = name
        self.parents = dict()
        self.children = dict()

    def add_parent(self, parent):
        if not isinstance(parent, Node):
            raise ValueError("Parent must be an instance of Node class.")
        pname = parent.name
        self.parents[pname] = parent

    def add_child(self, child):

        if not isinstance(child, Node):
            raise ValueError("Parent must be an instance of Node class.")
        cname = child.name
        self.children[cname] = child




class BN(object):
    def __init__(self):
        self.nodes = dict()

    def add_edge(self, edge):
        (pname, cname) = edge
        if pname not in self.nodes:
            self.nodes[pname] = Node(name=pname)
        if cname not in self.nodes:
            self.nodes[cname] = Node(name=cname)
        parent = self.nodes.get(pname)
        child = self.nodes.get(cname)
        parent.add_child(child)
        child.add_parent(parent)
```

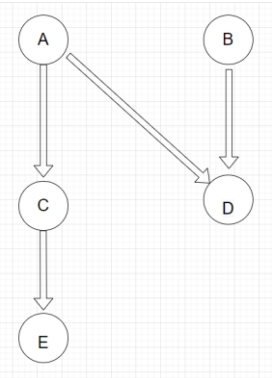
```
def find_obs_anc(self, observed):
    visit_nodes = copy.copy(observed)
    obs_anc = set()

    while len(visit_nodes) > 0:
        next_node = self.nodes[visit_nodes.pop()]
        for parent in next_node.parents:
            obs_anc.add(parent)

    return obs_anc

def is_dsep(self, start, end, observed):
    obs_anc = self.find_obs_anc(observed)
    via_nodes = [(start, "up")]
    visited = set()

    while len(via_nodes) > 0:
        (nname, direction) = via_nodes.pop()
        node = self.nodes[nname]

        if (nname, direction) not in visited:
            visited.add((nname, direction))

            if nname not in observed and nname == end:
                return False
            if direction == "up" and nname not in observed:
                for parent in node.parents:
                    via_nodes.append((parent, "up"))
                for child in node.children:
                    via_nodes.append((child, "down"))
            elif direction == "down":
                if nname not in observed:
                    for child in node.children:
                        via_nodes.append((child, "down"))
                if nname in observed or nname in obs_anc:
                    for parent in node.parents:
                        via_nodes.append((parent, "up"))
    return True
```

```
def get_skeleton(self):
    skeleton = set()
    for (nname, node) in self.nodes.iteritems():
        for parent in node.parents:
            if (nname, parent) not in skeleton and (parent, nname) not in skeleton:
                skeleton.add((nname, parent))
        for child in node.parents:
            if (nname, child) not in skeleton and (child, nname) not in skeleton:
                skeleton.add((nname, child))

    return skeleton

def get_skeleton_immor(self):
    skeleton = self.get_skeleton()
    immoral = set()
    for (nname, node) in self.nodes.iteritems():
        if len(node.parents) > 1:
            parents = node.parents.keys()
            for i1 in range(len(parents)-1):
                for i2 in range(i1+1, len(parents)):
                    p1, p2 = parents[i1], parents[i2]
                    if ((p1, p2) not in skeleton and (p2, p1) not in skeleton
                        and (p1, p2) not in immoral and (p2, p1) not in immoral):
                        immoral.add((p1, p2))

    return (skeleton, immoral)
```

dseparation.py

```
from BN import *

nnode = int(input("No. of nodes : "))
nedge = int(input("No. of edges : "))
nquery = int(input("No. of queries : "))

edges = []
for line in range(nedge):
    edge = input().rstrip().split(" ")
    edges += [edge]

queries = []
for line in range(nquery):
    query = input().rstrip().split(" ")
    (start, end, observed) = (query[0], query[1], query[3:])
    queries += [(start, end, observed)]

myBN = BN()
for edge in edges:
    myBN.add_edge(edge)
for (start, end, observed) in queries:
    print(myBN.is_dsep(start, end, observed))
```

OUTPUT




```
PS C:\Files\Class\AI\lab\bayes-net\src> python .\dseparation.py
No. of nodes : 5
No. of edges : 4
No. of queries : 4
A C
A D
B D
C E
Queries
A | C
D C
A | D
A D | B
True
False
True
False
```