Assingment 4

Backtracking

```
from typing import List
boardcount=0
def isboardok(chessboard:List,row:int,col:int):
    for c in range(col):
        if(chessboard[row][c]=='Q'):
            return False
    for r,c in zip(range(row-1,-1,-1),range(col-1,-1,-1)):
        if(chessboard[r][c]=='Q'):
            return False
    for r,c in zip(range(row+1,len(chessboard),1),range(col-1,-1,-1)):
        if(chessboard[r][c]=='Q'):
            return False
    return True


def displayboard(chessboard:List):
    for row in chessboard:
        print(row)
    print()


def placenqueens(chessboard:List,col:int):
    global boardcount
    if(col>=len(chessboard)):
        boardcount+=1
        print("Board"+str(boardcount))
        print("======================")
        displayboard(chessboard)
```

```
        print("======================\n\n")
    else:
        for row in range(len(chessboard)):
            chessboard[row][col]='Q'
            if(isboardok(chessboard,row,col)==True):
                placenqueens(chessboard,col+1)
            chessboard[row][col]='.'


chessboard=[]
N=int(input("Enter chessboard size: "))
for i in range(N):
    row=["."]*N
    chessboard.append(row)


placenqueens(chessboard,0)
```

Output

```
g:\clg\Third Year Engineering\Semester VI\Artificial Intelligence\practicals> g: && cd "g:\clg\Third Year Engineering\Semester VI\Artificial I
ntelligence\practicals" && cmd /C "C:\Users\admin\AppData\Local\Programs\Python\Python38-32\python.exe c:\Users\admin\.vscode\extensions\ms-py
thon.python-2022.2.1924087327\pythonFiles\lib\python\debugpy\launcher 52060 -- "g:\clg\Third Year Engineering\Semester VI\Artificial Intellige
nce\practicals\Backtracking Nqueen.py" "
Enter chessboard size: 4
Board1
====================
['.', '.', 'Q', '.']
['Q', '.', '.', '.']
['.', '.', '.', 'Q']
['.', 'Q', '.', '.']

====================


Board2
====================
['.', 'Q', '.', '.']
['.', '.', '.', 'Q']
['Q', '.', '.', '.']
['.', '.', 'Q', '.']

====================
```

Branch and Bound

```
def printSolution(board):
    for i in range(N):
        for j in range(N):
            print(board[i][j], end = " ")
        print()



def isSafe(row, col, nd, rd,rowLookup, ndLookup,rdLookup):
    if (ndLookup[nd[row][col]] or rdLookup[rd[row][col]] or rowLookup[row]):
        return False
    return True



def solveNQueensUtil(board, col, nd, rd,rowLookup, ndLookup,rdLookup):
    if(col >= N):
        return True
    for i in range(N):
        if(isSafe(i, col, nd, rd,rowLookup, ndLookup,rdLookup)):



            board[i][col] = 1
            rowLookup[i] = True
            ndLookup[nd[i][col]] = True
            rdLookup[rd[i][col]] = True
```

```
            if(solveNQueensUtil(board, col + 1,nd, rd,rowLookup, ndLookup,rdLookup)):
                return True
            board[i][col] = 0
            rowLookup[i] = False
            ndLookup[nd[i][col]] = False
            rdLookup[rd[i][col]] = False
    return False

def solveNQueens(N):

    board = [[0 for i in range(N)] for j in range(N)]



    nd = [[0 for i in range(N)] for j in range(N)]
    rd = [[0 for i in range(N)] for j in range(N)]
    rowLookup = [False] * N



    x = 2 * N - 1
    ndLookup = [False] * x
    rdLookup= [False] * x



    for r in range(N):
        for c in range(N):
            nd[r][c] = r + c
            rd[r][c] = r - c + N - 1

    if(solveNQueensUtil(board, 0, nd, rd,rowLookup, ndLookup,rdLookup) == False):
        print("Solution does not exist")
        return False
```

```
    printSolution(board)
    return True


N=int(input("Enter a Number: "))
solveNQueens(N)
```

Output

```
 g: && cd "g:\clg\Third Year Engineering\Semester VI\Artificial Intelligence\practicals" && cmd /C "C:\Users\admin\AppData\Local\Programs\Pyth
on\Python38-32\python.exe c:\Users\admin\.vscode\extensions\ms-python.python-2022.2.1924087327\pythonFiles\lib\python\debugpy\launcher 52147 -
- "g:\clg\Third Year Engineering\Semester VI\Artificial Intelligence\practicals\N queens.py" "
Enter a Number: 4
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0

g:\clg\Third Year Engineering\Semester VI\Artificial Intelligence\practicals> g: && cd "g:\clg\Third Year Engineering\Semester VI\Artificial I
ntelligence\practicals" && cmd /C "C:\Users\admin\AppData\Local\Programs\Python\Python38-32\python.exe c:\Users\admin\.vscode\extensions\ms-py
thon.python-2022.2.1924087327\pythonFiles\lib\python\debugpy\launcher 52151 -- "g:\clg\Third Year Engineering\Semester VI\Artificial Intellige
nce\practicals\N queens.py" "
Enter a Number: 8
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```

Assingment 1

DFS

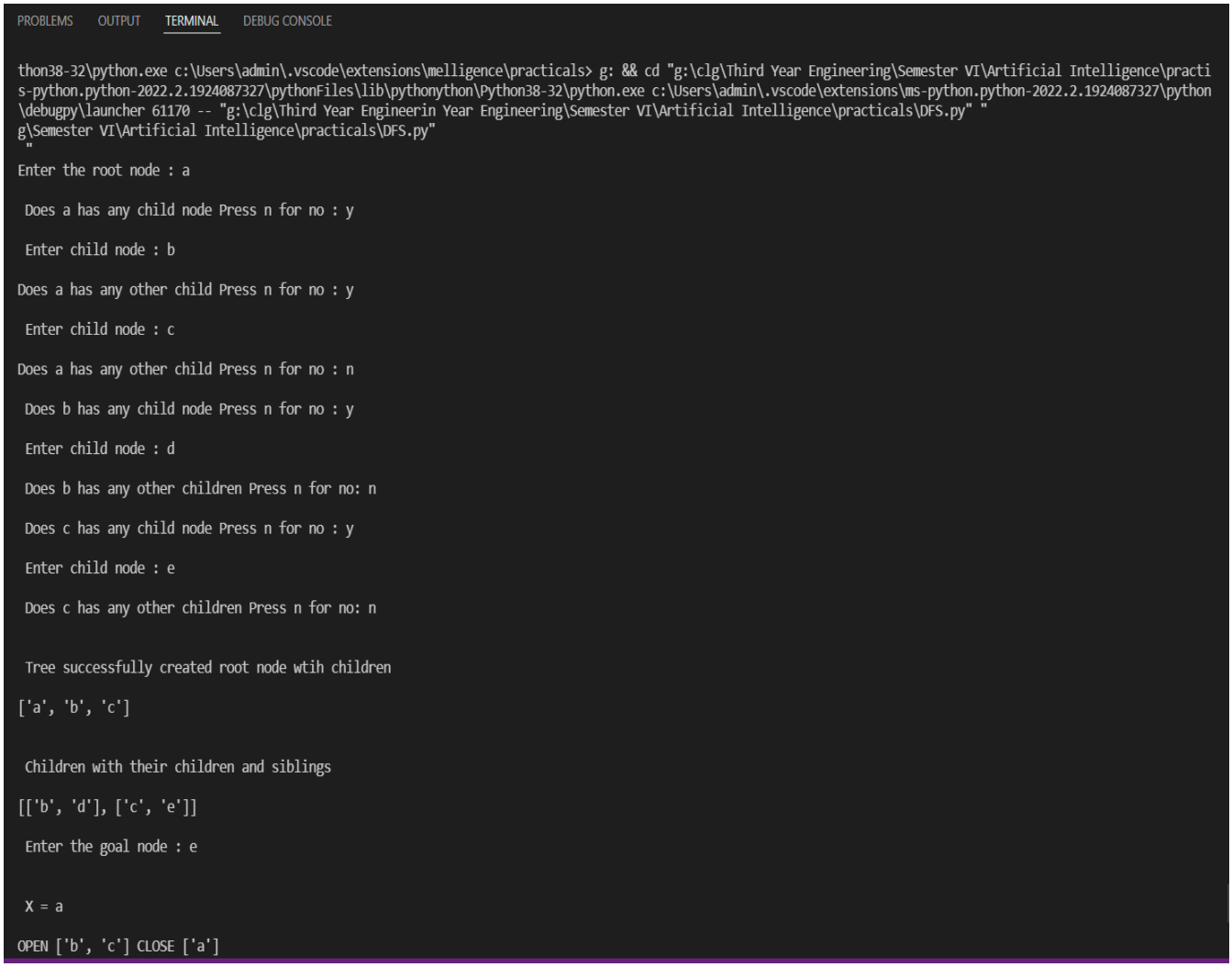
```
import itertools
def dfsalgo(childtree, openlist, closelist,goal):
    X=openlist[0]
    print("\n\n X = {}".format(X))
    closelist.append(X)
    for i in range(len(childtree)):
        if(X==childtree[i][0]):
            openlist.insert(0,childtree[i][1:])
            openlist=list(itertools.chain(*openlist))
            openlist.remove(X)
        if(X!=goal):
            print("\n OPEN {} CLOSE {}".format(openlist,closelist))
            if(X==goal):
                print('\n SUCCESS')
            elif(len(openlist)>0):
                dfsalgo(childtree, openlist, closelist,goal)
        else:
            print('\n\n FAILURE')


def createTree(roottree, treelength):
    try:
        for i in range(treelength):
            childtree[i].append(roottree[i+1])
            checkchild=input("\n Does "+roottree[i+1]+" has any child node Press n for no : ")
            if(checkchild=='n'):
                print()
```

```
            else:
                checkchildsibling=""
                while(checkchildsibling!='n'):
                    childname=input("\n Enter child node : ")
                    childtree[i].append(childname)
                    checkchildsibling = input("\n Does "+roottree[i+1]+" has any other children Press
n for no: ")
    except IndexError:
        pass
roottree=[]
root=input("Enter the root node : ")
roottree.append(root)
checkchild=input("\n Does "+root+" has any child node Press n for no : ")
if(checkchild=='n'):
    print(roottree)
else:
    checkchildsibling=""
    while(checkchildsibling!='n'):
        childname=input("\n Enter child node : ")
        roottree.append(childname)
        checkchildsibling = input("\nDoes "+root+" has any other child Press n for no : ")
treelength=len(roottree)
childtree=[[] for x in range(treelength-1)]
createTree(roottree,treelength)
print("\n\n Tree successfully created root node wtih children\n")
print(roottree)
print("\n\n Children with their children and siblings \n")
print(childtree)
goal=input("\n Enter the goal node : ")
openlist=[]
```

```
closelist=[]
X=roottree[0]
print("\n\n X = "+X)
openlist.append(roottree[1:])
openlist=list(itertools.chain(*openlist))
closelist.append(X)
print("\nOPEN {} CLOSE {}".format(openlist,closelist))
```

Output



BFS

```
import itertools

def bfsalgo(childtree, openlist, closelist,goal):
        '''Implementation of BFS algo'''
        X=openlist[0]
        print("\n\n X = {}".format(X))
        closelist.append(X)
        for i in range(len(childtree)):

                if(X==childtree[i][0]):
                        openlist.append(childtree[i][1:])

        openlist=list(itertools.chain(*openlist))
        del openlist[0]

        if(X!=goal):
                print("\n OPEN {}          CLOSE {}".format(openlist,closelist))

        if(X==goal):
                print('\n SUCCESS')
        elif(len(openlist)>0):
                bfsalgo(childtree, openlist, closelist,goal)
        else:
                print('\n\n FAILURE')



def createTree(treearr, treelength):
        '''Creating a child's child tree'''
        try:
```

```
                for i in range(treelength):
                        childtree[i].append(treearr[i+1])
                        checkchild=input("\n Does "+treearr[i+1]+" has any child node Press n for no
:         ")
                        if(checkchild=='n'):
                                print()
                        else:
                                checkchildsibling=""
                                while(checkchildsibling!='n'):
                                        childname=input("\n Enter child node :  ")

                                        childtree[i].append(childname)

                                        checkchildsibling = input("\n Does "+treearr[i+1]+" has any
other children Press n for no :    ")
        except IndexError:
                pass

treearr=[]
root=input("Enter the root node :         ")

treearr.append(root)

checkchild=input("\n Does "+root+" has any child node Press n for no :    ")
if(checkchild=='n'):
        print(treearr)

else:
        checkchildsibling=""
        while(checkchildsibling!='n'):

                childname=input("\n Enter child node :  ")
```

```
                treearr.append(childname)

                checkchildsibling = input("\nDoes "+root+" has any other child Press n for no :     ")

treelength=len(treearr)
childtree=[[] for x in range(treelength-1)]
createTree(treearr,treelength)

print("\n\n Tree successfully created root node wtih children\n")
print(treearr)

print("\n\n Children with their children and siblings \n")
print(childtree)

goal=input("\n Enter the goal node :       ")

openlist=[]
closelist=[]


X=treearr[0]
print("\n\n X = "+X)

openlist.append(treearr[1:])
openlist=list(itertools.chain(*openlist))
closelist.append(X)
print("\nOPEN {}           CLOSE {}".format(openlist,closelist))


bfsalgo(childtree,openlist,closelist,goal)
```

Output

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Microsoft Windows [Version 10.0.19042.1526]
(c) Microsoft Corporation. All rights reserved.

G:\clg\Third Year Engineering\Semester VI\Artificial Intelligence\practicals> cmd /C "C:\Users\admin\AppData\Local\Programs\Python\Python38-32\python.exe c:\Use
ntelligence\practicals> cmd /C "C:\Users\admin\AppData24087327\pythonFiles\lib\python\debugpy\launcher 56150 -- "g:\clg\Third Year Engineering\Semester VI\Artif
\Local\Programs\Python\Python38-32\python.exe c:\Users
\admin\.vscode\extensions\ms-python.python-2022.2.1924
087327\pythonFiles\lib\python\debugpy\launcher 56150 -
- "g:\clg\Third Year Engineering\Semester VI\Artificia
l Intelligence\practicals\BFS.py" "
Enter the root node :   s

 Does s has any child node Press n for no :     y

 Enter child node :     a

Does s has any other child Press n for no :     y

 Enter child node :     b

Does s has any other child Press n for no :     n

 Does a has any child node Press n for no :     y

 Enter child node :     c

 Does a has any other children Press n for no : n

 Does b has any child node Press n for no :     y

 Enter child node :     d

 Does b has any other children Press n for no : y

 Enter child node :     e

 Does b has any other children Press n for no : n


 Tree successfully created root node wtih children

['s', 'a', 'b']


 Children with their children and siblings
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

 Tree successfully created root node wtih children

['s', 'a', 'b']

 Children with their children and siblings

[['a', 'c'], ['b', 'd', 'e']]

 Enter the goal node :  e

 X = s

OPEN ['a', 'b'] CLOSE ['s']

 X = a

 OPEN ['c', 'b']        CLOSE ['s', 'a']

 X = c

 OPEN ['b']     CLOSE ['s', 'a', 'c']

 X = b

 OPEN ['d', 'e']        CLOSE ['s', 'a', 'c', 'b']

 X = d

 OPEN ['e']     CLOSE ['s', 'a', 'c', 'b', 'd']

 X = e

 SUCCESS

G:\clg\Third Year Engineering\Semester VI\Artificial Intelligence\practicals>
```

Assingment 2

A*

```
from  copy import deepcopy
import numpy as np
import time

def bestsolution(state):
    bestsol = np.array([], int).reshape(-1, 9)
    count = len(state) - 1
    while count != -1:
        bestsol = np.insert(bestsol, 0, state[count]['puzzle'], 0)
        count = (state[count]['parent'])
    return bestsol.reshape(-1, 3, 3)



def all(checkarray):
    set=[]
    for it in set:
        for checkarray in it:
            return 1
        else:
            return 0

def manhattan(puzzle, goal):
    a = abs(puzzle // 3 - goal // 3)
    b = abs(puzzle % 3 - goal % 3)
    mhcost = a + b
    return sum(mhcost[1:])
```

```
def misplaced_tiles(puzzle,goal):
    mscost = np.sum(puzzle != goal) - 1
    return mscost if mscost > 0 else 0


def coordinates(puzzle):
    pos = np.array(range(9))
    for p, q in enumerate(puzzle):
        pos[q] = p
    return pos


def evaluvate(puzzle, goal):
    steps = np.array([('up', [0, 1, 2], -3),('down', [6, 7, 8], 3),('left', [0, 3, 6], -1),('right', [2, 5, 8],
1)],
    dtype = [('move', str, 1),('position', list),('head', int)])
    dtstate = [('puzzle', list),('parent', int),('gn', int),('hn', int)]
    costg = coordinates(goal)
    parent = -1
    gn = 0
    hn = manhattan(coordinates(puzzle), costg)
    state = np.array([(puzzle, parent, gn, hn)], dtstate)
    dtpriority = [('position', int),('fn', int)]
    priority = np.array( [(0, hn)], dtpriority)
    while 1:
        priority = np.sort(priority, kind='mergesort', order=['fn', 'position'])
        position, fn = priority[0]
        priority = np.delete(priority, 0, 0)
        puzzle, parent, gn, hn = state[position]
        puzzle = np.array(puzzle)
```

```
        blank = int(np.where(puzzle == 0)[0])
        gn = gn + 1
        c = 1
        start_time = time.time()
        for s in steps:
            c = c + 1
            if blank not in s['position']:
                openstates = deepcopy(puzzle)
                openstates[blank], openstates[blank + s['head']] = openstates[blank +
s['head']],openstates[blank]
                if ~(np.all(list(state['puzzle']) == openstates, 1)).any():
                    end_time = time.time()
                    if (( end_time - start_time ) > 2):
                        print(" The 8 puzzle is unsolvable ! \n")
                        exit
                    hn = manhattan(coordinates(openstates), costg)
                    q = np.array([(openstates, position, gn, hn)], dtstate)
                    state = np.append(state, q, 0)
                    fn = gn + hn
                    q = np.array([(len(state) - 1, fn)], dtpriority)
                    priority = np.append(priority, q, 0)
                    if np.array_equal(openstates, goal):
                        print(' The 8 puzzle is solvable ! \n')
                        return state, len(priority)
    return state, len(priority)



def evaluvate_misplaced(puzzle, goal):
```

```
    steps = np.array([('up', [0, 1, 2], -3),('down', [6, 7, 8], 3),('left', [0, 3, 6], -1),('right', [2, 5, 8], 1)],
    dtype = [('move', str, 1),('position', list),('head', int)])
    dtstate = [('puzzle', list),('parent', int),('gn', int),('hn', int)]
    costg = coordinates(goal)
    parent = -1
    gn = 0
    hn = misplaced_tiles(coordinates(puzzle), costg)
    state = np.array([(puzzle, parent, gn, hn)], dtstate)
    dtpriority = [('position', int),('fn', int)]
    priority = np.array([(0, hn)], dtpriority)
    while 1:
        priority = np.sort(priority, kind='mergesort', order=['fn', 'position'])
        position, fn = priority[0]
        priority = np.delete(priority, 0, 0)
        puzzle, parent, gn, hn = state[position]
        puzzle = np.array(puzzle)
        blank = int(np.where(puzzle == 0)[0])
        gn = gn + 1
        c = 1
        start_time = time.time()
        for s in steps:
            c = c + 1
            if blank not in s['position']:
                openstates = deepcopy(puzzle)
                openstates[blank], openstates[blank + s['head']] = openstates[blank + s['head']],openstates[blank]
                if ~(np.all(list(state['puzzle']) == openstates, 1)).any():
                    end_time = time.time()
                    if (( end_time - start_time ) > 2):
```

```
                print(" The 8 puzzle is unsolvable \n")
                break
            hn = misplaced_tiles(coordinates(openstates), costg)
            q = np.array([(openstates, position, gn, hn)], dtstate)
            state = np.append(state, q, 0)
            fn = gn + hn
            q = np.array([(len(state) - 1, fn)], dtpriority)
            priority = np.append(priority, q, 0)
            if np.array_equal(openstates, goal):
                print(' The 8 puzzle is solvable \n')
                return state, len(priority)
    return state, len(priority)

puzzle = []
print(" Input vals from 0-8 for start state ")
for i in range(0,9):
    x = int(input("enter vals :"))
    puzzle.append(x)

goal = []
print(" Input vals from 0-8 for goal state ")
for i in range(0,9):
    x = int(input("Enter vals :"))
    goal.append(x)
n = int(input("1. Manhattan distance \n2. Misplaced tiles"))
if(n ==1 ):
    state, visited = evaluvate(puzzle, goal)
    bestpath = bestsolution(state)
    print(str(bestpath).replace('[', ' ').replace(']', ''))
```

```
    totalmoves = len(bestpath) - 1
    print('Steps to reach goal:',totalmoves)
    visit = len(state) - visited
    print('Total nodes visited: ',visit, "\n")
    print('Total generated:', len(state))
if(n == 2):
    state, visited = evaluvate_misplaced(puzzle, goal)
    bestpath = bestsolution(state)
    print(str(bestpath).replace('[', ' ').replace(']', ''))
    totalmoves = len(bestpath) - 1
    print('Steps to reach goal:',totalmoves)
    visit = len(state) - visited
    print('Total nodes visited: ',visit, "\n")
    print('Total generated:', len(state))
```

Output:

```
elligence\practicals\Astar.py" "
 Input vals from 0-8 for start state
enter vals :1
enter vals :0
enter vals :3
enter vals :4
enter vals :2
enter vals :5
enter vals :7
enter vals :8
enter vals :6
 Input vals from 0-8 for goal state
Enter vals :1
Enter vals :2
Enter vals :3
Enter vals :4
Enter vals :5
Enter vals :6
Enter vals :7
Enter vals :8
Enter vals :0
1. Manhattan distance
2. Misplaced tiles1
 The 8 puzzle is solvable !

   1 0 3
   4 2 5
   7 8 6

   1 2 3
   4 0 5
   7 8 6

   1 2 3
   4 5 0
   7 8 6

   1 2 3
   4 5 6
   7 8 0
Steps to reach goal: 3
Total nodes visited:  3

Total generated: 9
```

```
\python\debugpy\launcher 51564 -- "g:\clg\Third Year Engineering\Semester VI\Artificial Intelligence\practicals\Astar.py" "
 Input vals from 0-8 for start state
enter vals :1
enter vals :0
enter vals :3
enter vals :4
enter vals :2
enter vals :6
enter vals :7
enter vals :5
enter vals :8
 Input vals from 0-8 for goal state
Enter vals :1
Enter vals :2
Enter vals :3
Enter vals :4
Enter vals :5
Enter vals :6
Enter vals :7
Enter vals :8
Enter vals :0
1. Manhattan distance
2. Misplaced tiles2
 The 8 puzzle is solvable

   1 0 3
   4 2 6
   7 5 8

   1 2 3
   4 0 6
   7 5 8

   1 2 3
   4 5 6
   7 0 8

   1 2 3
   4 5 6
   7 8 0
Steps to reach goal: 3
Total nodes visited:  3

Total generated: 9
```

Assingment 3

Prims

```
import sys

class Graph():

    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]
    def printMST(self, parent):
        print ("Edge \tWeight")
        for i in range(1, self.V):
            print (parent[i], "-", i, "\t", self.graph[i][parent[i]])

    def minKey(self, key, mstSet):

        min = sys.maxsize

        for v in range(self.V):
            if key[v] < min and mstSet[v] == False:
                min = key[v]
                min_index = v
        return min_index

    def primMST(self):
        key = [sys.maxsize] * self.V
        parent = [None] * self.V
```

```
        key[0] = 0
        mstSet = [False] * self.V

        parent[0] = -1

        for cout in range(self.V):
            u = self.minKey(key, mstSet)
            mstSet[u] = True
            for v in range(self.V):

                if self.graph[u][v] > 0 and mstSet[v] == False and key[v] > self.graph[u][v]:
                        key[v] = self.graph[u][v]
                        parent[v] = u

        self.printMST(parent)

g = Graph(5)
g.graph= []
n = int(input("Enter number of elements : "))
for i in range(0, n):
    ele = [int(input()), int(input()),int(input()),int(input()),int(input())]
    g.graph.append(ele)
print(g.graph)
g.primMST()
```

Output

```
G:\clg\Third Year Engineering\Semester VI\Artificial Intelligence\practicals> g: && cd "g:\clg\Third Year Engineering\Semester VI\Artificial Intelligence\practicals" && cmd /C "C:\Users\admin\AppData\Local\Programs\Python\Python38-32\python.exe c:\Users\admin\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher 61201 -- "g:\clg\Third Year Engineering\Semester VI\Artificial Intelligence\practicals\Prims.py" "
Enter number of elements : 5
0
2
0
6
0
2
0
3
8
5
0
3
0
0
7
6
8
0
0
9
0
5
7
9
0
[[0, 2, 0, 6, 0], [2, 0, 3, 8, 5], [0, 3, 0, 0, 7], [6, 8, 0, 0, 9], [0, 5, 7, 9, 0]]
Edge    Weight
0 - 1    2
1 - 2    3
0 - 3    6
1 - 4    5
```

Assingment 6

Expert System

```
go:-
hypothesis(Disease),
write('It is suggested that the patient has '),
write(Disease),
nl,
undo;
write('Sorry, the system is unable to identify the disease'),nl,undo.


hypothesis(cold) :-
symptom(headache),
symptom(runny_nose),
symptom(sneezing),
symptom(sore_throat),
nl,
write('Advices and Sugestions:'),
nl,
write('1: Tylenol'),
nl,
write('2: Panadol'),
nl,
write('3: Nasal spray'),
nl,
write('Please weare warm cloths because'),
nl,!.
```

```
hypothesis(influenza) :-
symptom(sore_throat),
symptom(fever),
symptom(headache),
symptom(chills),
symptom(body_ache),
nl,
write('Advices and Sugestions:'),
nl,
write('1: Tamiflu'),
nl,
write('2: Panadol'),
nl,
write('3: Zanamivir'),
nl,
write('Please take a warm bath and do salt gargling because'),
nl,!.


hypothesis(typhoid) :-
symptom(headache),
symptom(abdominal_pain),
symptom(poor_appetite),
symptom(fever),
nl,
write('Advices and Sugestions:'),
nl,
write('1: Chloramphenicol'),
```

```
nl,
write('2: Amoxicillin'),
nl,
write('3: Ciprofloxacin'),
nl,
write('4: Azithromycin'),
nl,
write('Please do complete bed rest and take soft diet because'),
nl,!.


hypothesis(chicken_pox) :-
symptom(rash),
symptom(body_ache),
symptom(fever),
nl,
write('Advices and Sugestions:'),
nl,
write('1: Varicella vaccine'),
nl,
write('2: Immunoglobulin'),
nl,
write('3: Acetomenaphin'),
nl,
write('4: Acyclovir'),
nl,
write('Please do have oatmeal bath and stay at home because'),
nl.
```

```
hypothesis(measles) :-
symptom(fever),
symptom(runny_nose),
symptom(rash),
symptom(conjunctivitis),
nl,
write('Advices and Sugestions:'),
nl,
write('1: Tylenol'),
nl,
write('2: Aleve'),
nl,
write('3: Advil'),
nl,
write('4: Vitamin A'),
nl,
write('Please get rest and use more liquid because'),
nl,!.


hypothesis(malaria) :-
symptom(fever),
symptom(sweating),
symptom(headache),
symptom(nausea),
symptom(vomiting),
symptom(diarrhea),
nl,
```

```
write('Advices and Sugestions:'),
nl,
write('1: Aralen'),
nl,
write('2: Qualaquin'),
nl,
write('3: Plaquenil'),
nl,
write('4: Mefloquine'),
nl,
write('Please do not sleep in open air and cover your full skin because'),
nl,!.


ask(Question) :-
write('Does the patient has the symptom '),
write(Question),
write('? : '),
read(Response),
nl,
( (Response == yes ; Response == y)
->
assert(yes(Question)) ;
assert(no(Question)), fail).
:- dynamic yes/1,no/1.

symptom(S) :-
(yes(S)
->
```

true ;

(no(S)

->

fail ;

ask(S))).

undo :- retract(yes(_)),fail.

undo :- retract(no(_)),fail.

undo.

Output

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/admin/Documents/Prolog/medicalExpert.pl compiled 0.00 sec, 12 clauses
?- go.
Does the patient has the symptom headache? : y.

Does the patient has the symptom runny_nose? : |: n.

Does the patient has the symptom sore_throat? : |: n.

Does the patient has the symptom abdominal_pain? : |: n.

Does the patient has the symptom rash? : |: n.

Does the patient has the symptom fever? : |: y.

Does the patient has the symptom sweating? : |: y.

Does the patient has the symptom nausea? : |: y.

Does the patient has the symptom vomiting? : |: y.

Does the patient has the symptom diarrhea? : |: y.


Advices and Sugestions:
1: Aralen
2: Qualaquin
3: Plaquenil
4: Mefloquine
Please do not sleep in open air and cover your full skin because
It is suggested that the patient has malaria
true .

?-
```