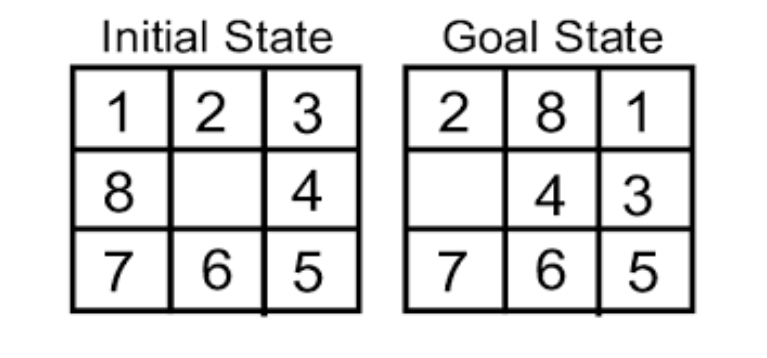
**Computer Science and Engineering Department**

**Artificial Intelligence (UCS-521)**

**Lab Assignment-2**

1. Write a code in python for the 8 puzzle problem by taking the following initial and final states



**CODE:**

#Uninformed Searching using Breadth First Search

import numpy as np  
  
initial\_arr = np.array([[1,2,3],[8,0,4],[7,6,5]])  
final\_arr = np.array([[2,8,1],[0,4,3],[7,6,5]])  
  
#All possible moves  
# up = (-1,0)  
# down = (1,0)  
# left = (0,-1)  
# right = (0,1)  
  
moves = [(-1,0),(1,0),(0,-1),(0,1)]  
movesName = ['UP', 'DOWN', 'LEFT', 'RIGHT']

#checking valid moves  
def isValidMove(initial\_arr, idx, move):  
 i = idx[0] + move[0]  
 j = idx[1] + move[1]  
 if i<len(initial\_arr) and i>=0 and j>=0 and j<len(initial\_arr):  
 return True  
 return False

def performMove(initial\_arr, idx, move):  
 i = idx[0] + move[0]  
 j = idx[1] + move[1]  
 temp\_arr = initial\_arr.copy()  
 temp = temp\_arr[i][j]  
 temp\_arr[i][j] = temp\_arr[idx[0]][idx[1]]  
 temp\_arr[idx[0]][idx[1]] = temp  
 return temp\_arr

def findZeroIndex(initial\_arr):

for i in range(0,len(initial\_arr)):  
 for j in range(0,len(initial\_arr[i])):  
 if initial\_arr[i][j] == 0:  
 return i,j

#Function to print all intermediate states and the moves involved  
def printRes(bfs, i):

if bfs[i][2] == -1:  
 print('Initial State')  
 print(bfs[i][0])  
 return 0

count = printRes(bfs,bfs[i][2])  
 ind = moves.index(bfs[i][1])  
 print(f'Move = {movesName[ind]}')  
 print(bfs[i][0])  
 return count + 1

def findSolBFS(initial\_arr, final\_arr, prevMove):

bfs = [(initial\_arr,prevMove,-1)]  
 flag = 0  
 l = len(bfs)  
 i = 0

while i<l:  
 currNode = bfs[i]  
 idx = findZeroIndex(currNode[0])

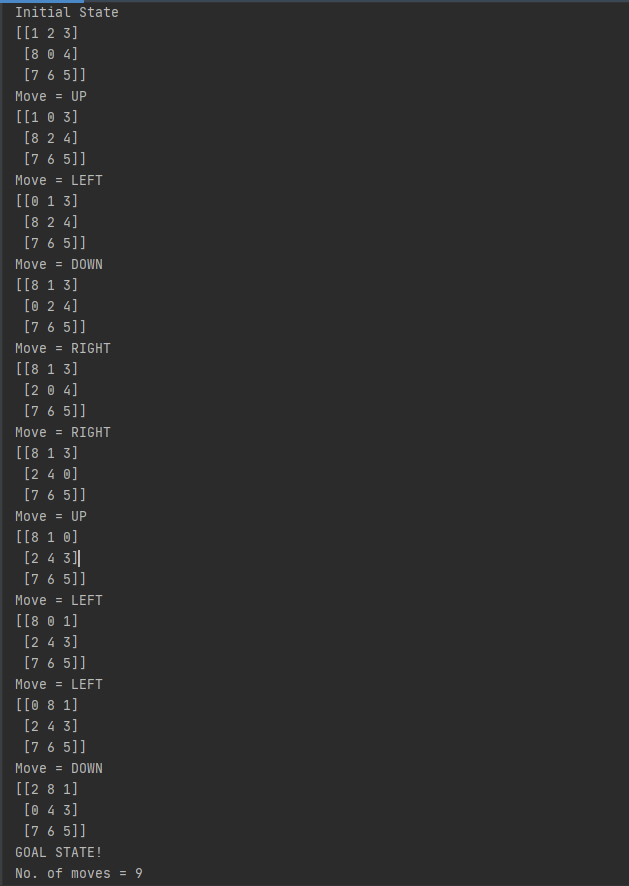
for move in moves:  
 if isValidMove(currNode[0],idx,move):  
 new\_arr = performMove(currNode[0],idx,move)  
 bfs.append((new\_arr,move, i))

if np.count\_nonzero(np.subtract(new\_arr,final\_arr)) == 0:  
 l = len(bfs)  
 count = printRes(bfs,l-1)  
 print('GOAL STATE!')  
 print(f'No. of moves = {count}')  
 flag = 1  
 break

l = len(bfs)  
 i = i+1

if flag == 1:  
 break  
  
findSolBFS(initial\_arr, final\_arr, (0,0))

**OUTPUT:**



1. Given two jugs- a 4 liter and 3 liter capacity. Neither has any measurable markers on it. There is a pump which can be used to fill the jugs with water. Simulate the procedure in Python to get exactly 2 liter of water into 4-liter jug

**CODE:**

x,y,m,n = 0,0,4,3  
  
print('Initital State = (0,0)')  
print('Capacitites = ({0},{1})'.format(m,n))  
print('Goal State = (2,0)')  
  
while x!=2 or y!=0:  
 r = int(input('Enter Rule: '))

if r==1:  
 #full x  
 if x<m:  
 x = m

if r==2:  
 #full y  
 if y<n:  
 y = n

if r==3:  
 #Empty x  
 if x>0:  
 x = 0

if r==4:  
 #empty y  
 if y>0:  
 y=0

if r==5:  
 #transfer from y to x when x+y>=m  
 if x+y>=m and y>0:  
 x,y = m,y-(m-x)

if r==6:  
 # transfer from x to y when x+y>=n  
 if x+y>=n and x>0:  
 x,y = x-(n-y),n

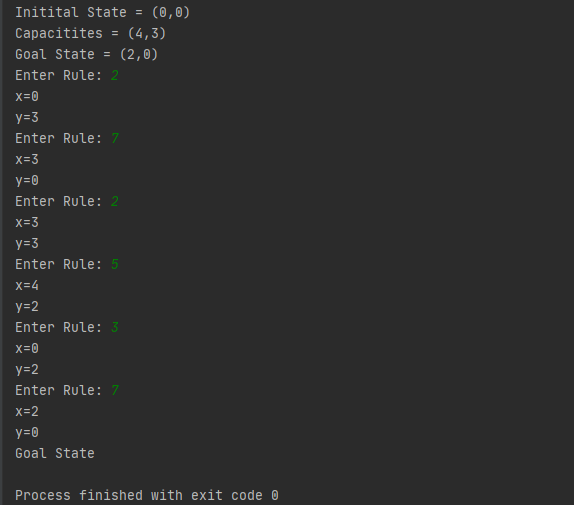
if r==7:  
 # transfer from y to x when x+y<=m  
 if x+y<=m and y>0:  
 x,y = x+y,0

if r==8:  
 # transfer from x to y when x+y<=n  
 if x+y<=n and x>0:  
 x,y = 0,x+y

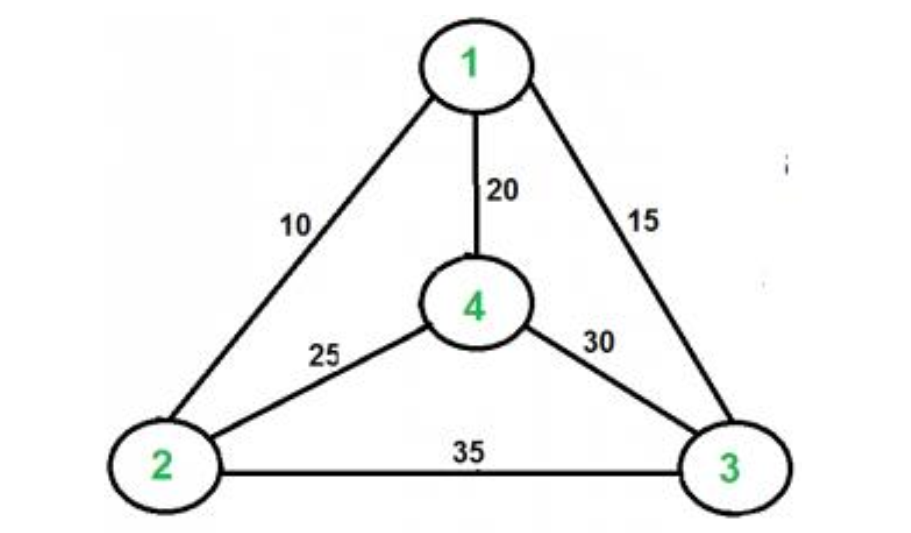
print(f'x={x}')  
 print(f'y={y}')

if x==2 and y==0:  
 print('Goal State')

**OUTPUT:**



1. Write a Python program to implement Travelling Salesman Problem (TSP). Take the starting node from the user at run time.

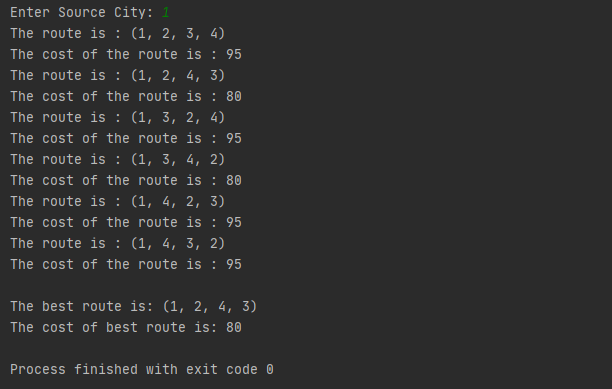


**CODE:**

graph = [[0,10,15,20],  
 [10,0,35,25],  
 [15,35,0,30],  
 [20,25,30,0]]  
  
from itertools import permutations  
l = list(permutations(range(1,4+1)))  
  
min=10000  
s = int(input('Enter Source City: '))  
  
for x in l:  
 if x[0] == s:  
 print(f'The route is : {x}')  
 sum = 0  
  
 for j in range(len(x)):  
 if(j == len(x)-1):  
 sum = sum + graph[x[j]-1][x[0]-1]  
 print(f'The cost of the route is : {sum}')  
 else:  
 sum = sum + graph[x[j]-1][x[j+1]-1]  
  
 if sum < min:  
 min = sum  
 best\_route = x  
  
print(f'\nThe best route is: {best\_route}')  
print(f'The cost of best route is: {min}')

**OUTPUT:**

(when source city is 1)



(when source city is 3)

