**AI Assignment 2**

**Name -** Panshul Saxena

**Batch -** 2CS10

**Roll No -** 102196006

Q1

Given two jugs- a 4 litre and 3 litre 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 litre of water into 4-litre jug.

Code:

x = 0

y = 0

state = [[0, 0]]

while(x != 2):

    if [4, y] not in state and x < 4:

        x = 4

        state.append([x, y])

        print(x, y)

    if [x, 3] not in state and y < 3:

        y = 3

        state.append([x, y])

        print(x, y)

    if [0, y] not in state and x > 0:

        x = 0

        state.append([x, y])

        print(x, y)

    if [x, 0] not in state and y > 0:

        y = 0

        state.append([x, y])

        print(x, y)

    if [4, y-(4-x)] not in state and x+y >= 4 and y > 0:

        y = y-(4-x)

        x = 4

        state.append([x, y])

        print(x, y)

    if [x-(3-y), 3] not in state and x+y >= 3 and x > 0:

        x = x-(3-y)

        y = 3

        state.append([x, y])

        print(x, y)

    if [x+y, 0] not in state and (x+y) <= 4 and y >= 0:

        x = x+y

        y = 0

        state.append([x, y])

        print(x, y)

    if [0, y+x] not in state and (x+y) <= 3 and x >= 0:

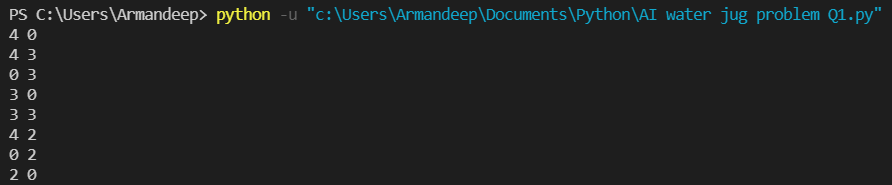
        y = x+y

        x = 0

        state.append([x, y])

        print(x, y)

Output:



Q2

Given three jugs: 12, 8 and 5 liter capacities. Largest jug is completely filled. Using these 3 jugs, split the water to obtain exactly 6 liter in largest jugs.

Code:

x = 12

y = 0

z = 0

state = [[12,0,0]]

while(x != 6):

    if [x-(8-y), 8, z] not in state and x+y >= 8 and x > 0:

        x = x-(8-y)

        y = 8

        state.append([x, y, z])

        print(x, y, z)

    if [x, y-(5-z), 5] not in state and y+z >= 5 and  y > 0:

        y = y-(5-z)

        z = 5

        state.append([x, y, z])

        print(x, y, z)

    if [x+z, y, 0] not in state and (x+z) <= 12 and z >= 0:

        x = x+z

        z = 0

        state.append([x, y, z])

        print(x, y, z)

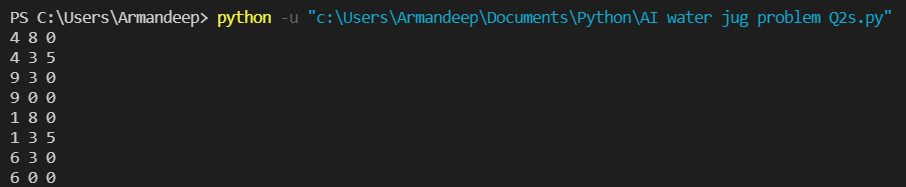
    if [x, 0, z] not in state and y > 0:

        y = 0

        state.append([x, y, z])

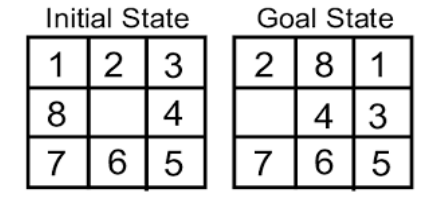
        print(x, y, z)

Output:



Q3

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



Code:

import copy

start\_state = [[1, 2, 3], [8, 0, 4], [7, 6, 5]]

goal\_state = [[2, 8, 1], [0, 4, 3], [7, 6, 5]]

q = []

cnt = 1

def find\_pos(start):

    for i in range(len(start)):

        for j in range(len(start[i])):

            if start[i][j] == 0:

                return (i, j)

def compare(curr, goal):

    if curr == goal:

        return 1

    else:

        return 0

def up(state, u, v):

    state[u][v] = state[u-1][v]  # up

    state[u-1][v] = 0

    return state

def right(state, l, m):

    state[l][m] = state[l][m+1]  # right

    state[l][m+1] = 0

    return state

def left(state, n, o):

    state[n][o] = state[n][o-1]  # left

    state[n][o-1] = 0

    return state

def down(state, r, s):

    state[r][s] = state[r+1][s]  # down

    state[r+1][s] = 0

    return state

def states(start, goal):

    pos = find\_pos(start)

    i = pos[0]

    j = pos[1]

    global cnt

    if i-1 >= 0:

        state\_1 = copy.deepcopy(start)

        state1 = up(state\_1, i, j)

        if compare(state1, goal):

            print(cnt)

            print(state1)

            return 1

        else:

            if state1 not in q:

                q.append(state1)

                cnt += 1

    if j+1 <= 2:

        state\_2 = copy.deepcopy(start)

        state2 = right(state\_2, i, j)

        if compare(state2, goal):

            print(cnt)

            print(state2)

            return 1

        else:

            if state2 not in q:

                q.append(state2)

                cnt += 1

    if j-1 >= 0:

        state\_3 = copy.deepcopy(start)

        state3 = left(state\_3, i, j)

        if compare(state3, goal):

            print(cnt)

            print(state3)

            return 1

        else:

            if state3 not in q:

                q.append(state3)

                cnt += 1

    if i+1 <= 2:

        state\_4 = copy.deepcopy(start)

        state4 = down(state\_4, i, j)

        if compare(state4, goal):

            print(cnt)

            print(state4)

            return 1

        else:

            if state4 not in q:

                q.append(state4)

                cnt += 1

def func():

    if not compare(start\_state, goal\_state):

        q.append(start\_state)

    i = 0

    while(q):

        spop = q[i]

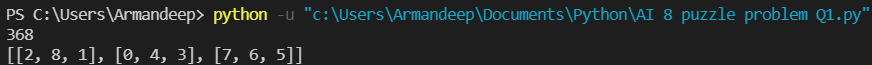
        i += 1

        if(states(spop, goal\_state)):

            break

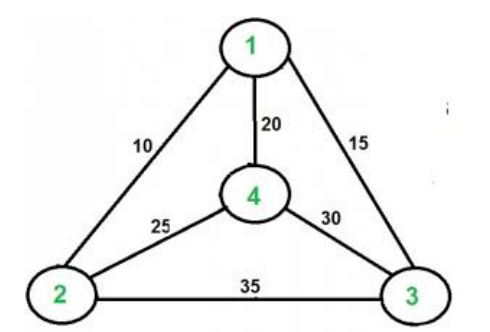
func()

Output:



Q4

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



Code:

from sys import maxsize

from itertools import permutations

V = 4

def travellingSalesmanProblem(graph, s):

    vertex = []

    for i in range(V):

        if i != s:

            vertex.append(i)

    min\_path = maxsize

    all\_perm = list(permutations(vertex))

    all\_perm.append(vertex)

    for j in range(len(all\_perm)):

        current\_pathweight = 0

        k = s

        vertex = all\_perm[j]

        for i in range(len(vertex)):

            current\_pathweight += graph[k][vertex[i]]

            k = vertex[i]

        current\_pathweight += graph[k][s]

        min\_path = min(min\_path, current\_pathweight)

    return min\_path

graph = [[0, 10, 15, 20], [10, 0, 35, 25],

         [15, 35, 0, 30], [20, 25, 30, 0]]

# s = int(input())

s = 1

print(travellingSalesmanProblem(graph, s))

Output:

