**Subject: Artificial Intelligence (DJ19DSC502)** 

AY: 2023-24

#### **Experiment 1**

## (Problem Solving)

Aim: Implement domain specific functions for given problems required for problem solving.

#### Theory:

There are two domain specific functions required in all problem solving methods.

1. GoalTest Function:

**goalTest(State)** Returns *true* if the input state is the goal state and *false* otherwise.

goalTest(State, Goal) Returns true if State matches Goal, and false otherwise.

#### 2. MoveGen function:

```
Initialize set of successors C to empty set.
Add M to the complement of given state N to get new state S.
If given state has Left, then add Right to S, else add Left.
If legal(S) then add S to set of successors C.
For each other-entity E in N
    make a copy S' of S,
    add E to S',
    If legal (S'), then add S' to C.
Return (C).
```

#### Lab Assignment to do:

Create MoveGen and GoalTest Functions for the given problems

#### 1. Water Jug Problem

There are two jugs available of different volumes such as a 3 litres and a 7 litres and you have to measure a different volume such as 6 litre.

#### 2. Travelling Salesman Problem

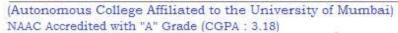
A salesman is travelling and selling his/her product to in different cities. The condition is that it has to travel each city just once.

#### 3. 8 Puzzle Problem

An initial state is given in a 8 puzzle where one place is blank out of 9 places. You can shift this blank space and get a different state to reach to a given goal state.



## DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING





## **Department of Computer Science and Engineering (Data Science)**

1.

```
▼ Water jug

[] import numpy as np import copy

Start=[0,0,0] cap=[0,0,0] print("enter the capacities of jug")] for i in range(3): cap[i]=int(input()) #arrange in increasing order cap=np.sort(cap) print(cap) #the largest will be filled with water start[2]=cap[2] print(start)

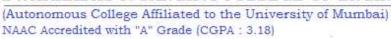
▼ Water jug

| Application | Cap |
```

```
[ ] def filljug1(start,cap):
        print("\nFilling jug 1")
        temp = copy.deepcopy(start)
        temp[0], temp[2] = cap[0],temp[2]-cap[0]
        print(temp)
      filljug1(start,cap)
```



## DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING





### **Department of Computer Science and Engineering (Data Science)**

```
[ ] def filljug12(start,cap):
    print("\nFilling jug 1 then 2")
    temp = copy.deepcopy(start)
    temp[0], temp[2] = cap[0],temp[2]-cap[0]
    if cap[1]<temp[2]:
        temp[1], temp[2] = cap[1],temp[2]-cap[1]
    if cap[1]>=temp[2]:
        temp[1], temp[2] = temp[2],0
    print(temp)
filljug12(start,cap)
```

```
def filljug21(start,cap):
    print("\nFilling jug 2 then 1")
    temp = copy.deepcopy(start)
    temp[1], temp[2] = cap[1],temp[2]-cap[1]
    if cap[0]<temp[2]:
        temp[0], temp[2] = cap[1],temp[2]-cap[1]
    if cap[0]>=temp[2]:
        temp[0], temp[2] = temp[2],0
print(temp)
filljug21(start,cap)
```

```
def movegen(start,cap):
    filljug1(start,cap)
    filljug2(start,cap)
    filljug12(start,cap)
    filljug21(start,cap)

movegen(start,cap)
```

# **Output:**

```
enter the capacities of jug
8
5
12
[ 5 8 12]
[0, 0, 12]
```

```
Filling jug 1
[5, 0, 7]

Filling jug 2
[0, 8, 4]

Filling jug 1 then 2
[5, 7, 0]

Filling jug 2 then 1
[4, 8, 0]
```

```
enter the capacities of jug

3

5

8

[3 5 8]

[0, 0, 8]
```

```
Filling jug 1
[3, 0, 5]

Filling jug 2
[0, 5, 3]

Filling jug 1 then 2
[3, 5, 0]

Filling jug 2 then 1
[3, 5, 0]
```



### DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING



(Autonomous College Affiliated to the University of Mumbai) NAAC Accredited with "A" Grade (CGPA: 3.18)

### **Department of Computer Science and Engineering (Data Science)**

2.

```
TSP
     import numpy as np
[2] def distance(point1, point2):
         return np.linalg.norm(np.array(point1) - np.array(point2))
[3] def total_distance(path, points):
         dist = 0
         for i in range(len(path) - 1):
             dist += distance(points[path[i]], points[path[i + 1]])
         dist += distance(points[path[-1]], points[path[0]])
         return dist
[4] def move_gen(path):
         neighbors = []
         for i in range(len(path)):
             for j in range(i + 1, len(path)):
                 new_path = path[:]
                 new_path[i], new_path[j] = new_path[j], new_path[i]
                 neighbors.append(new_path)
         return neighbors
[ ] def goal_test(path, points):
         return len(path) == len(points) and total distance(path, points) < float('inf')
```

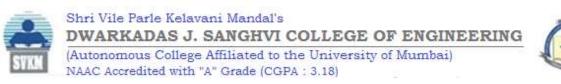
```
7] def tsp_solver(points):
    num_cities = len(points)
    initial_path = list(range(num_cities))
    current_path = initial_path
    current_distance = total_distance(current_path, points)

while True:
    neighbors = move_gen(current_path)
    found_better_path = False

for neighbor in neighbors:
    neighbor_distance = total_distance(neighbor, points)
    if neighbor_distance < current_distance:
        current_path = neighbor
        current_distance = neighbor_distance
        found_better_path = True

if not found_better_path:
        break

return current_path, current_distance</pre>
```



```
num_cities = int(input("Enter the number of cities: "))
cities = []

for i in range(num_cities):
    x, y = map(int, input(f"Enter coordinates for city {i + 1} (x, y): ").split())
    cities.append((x, y))

best_path, best_distance = tsp_solver(cities)

print("Best tour order:", best_path)
print("Total distance:", best_distance)
```

## **Output:**

```
Enter the number of cities: 4

Enter coordinates for city 1 (x, y): 0 0

Enter coordinates for city 2 (x, y): 1 1

Enter coordinates for city 3 (x, y): 0 1

Enter coordinates for city 4 (x, y): 1 0

Best tour order: [3, 1, 2, 0]

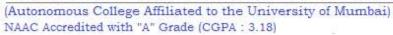
Total distance: 4.0
```



3.



## DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING





## **Department of Computer Science and Engineering (Data Science)**

```
def move_up(state):
    print("UP")
    newstate=copy.deepcopy(state)
    i,j=findblank(newstate)
    if i!=0:
        temp=newstate[i-1][j]
        newstate[i-1][j]=newstate[i][j]
        newstate[i][j]=temp
        print(newstate)
    else:
        print("Not possible")
    move_up(start)
```

```
[5] def move_down(state):
    print("\nDOWN")
    newstate=copy.deepcopy(state)
    i,j=findblank(newstate)
    if i!=2:
        temp=newstate[i+1][j]
        newstate[i+1][j]=newstate[i][j]
        newstate[i][j]=temp
        print(newstate)
    else:
        print("Not possible")

move_down(start)
```



## DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING



(Autonomous College Affiliated to the University of Mumbai) NAAC Accredited with "A" Grade (CGPA: 3.18)

## **Department of Computer Science and Engineering (Data Science)**

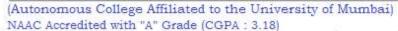
```
def move_left(state):
    print("\nLEFT")
    newstate=copy.deepcopy(state)
    i,j=findblank(newstate)
    if j!=0:
        temp=newstate[i][j-1]
        newstate[i][j-1]=newstate[i][j]
        newstate[i][j]=temp
        print(newstate)
    else:
        print("Not possible")

move_left(start)
```

```
def move_right(state):
    print("\nRIGHT")
    newstate=copy.deepcopy(state)
    i,j=findblank(newstate)
    if j!=2:
        temp=newstate[i][j+1]
        newstate[i][j+1]=newstate[i][j]
        newstate[i][j]=temp
        print(newstate)
    else:
        print("Not possible")
    new_state=move_right(start)
```



### DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING





## **Department of Computer Science and Engineering (Data Science)**

#### **OUTPUT:**

```
[- 1 2 3 4 --1 5 6 7 8 8 The entered puzzle is: [[1 2 3] [4 -1 5] [6 7 8]]

UP

[[1 -1 3] [4 2 5] [6 7 8]]

DOWN

[[1 2 3] [4 7 5] [6 -1 8]]

LEFT

[[1 2 3] [-1 4 5] [6 7 8]]

RIGHT

[[1 2 3] [4 5 -1] [6 7 8]]
```

```
C+ 1
2
3
4
5
-1
6
7
8
The entered puzzle is:
[[1 2 3]
  [4 5 -1]
  [6 7 8]]
UP
  [[1 2 -1]
  [4 5 3]
  [6 7 8]]

DOWN
  [[1 2 3]
  [4 5 8]
  [6 7 -1]]

LEFT
  [[1 2 3]
  [4 -1 5]
  [6 7 8]]

RIGHT
Not possible
```