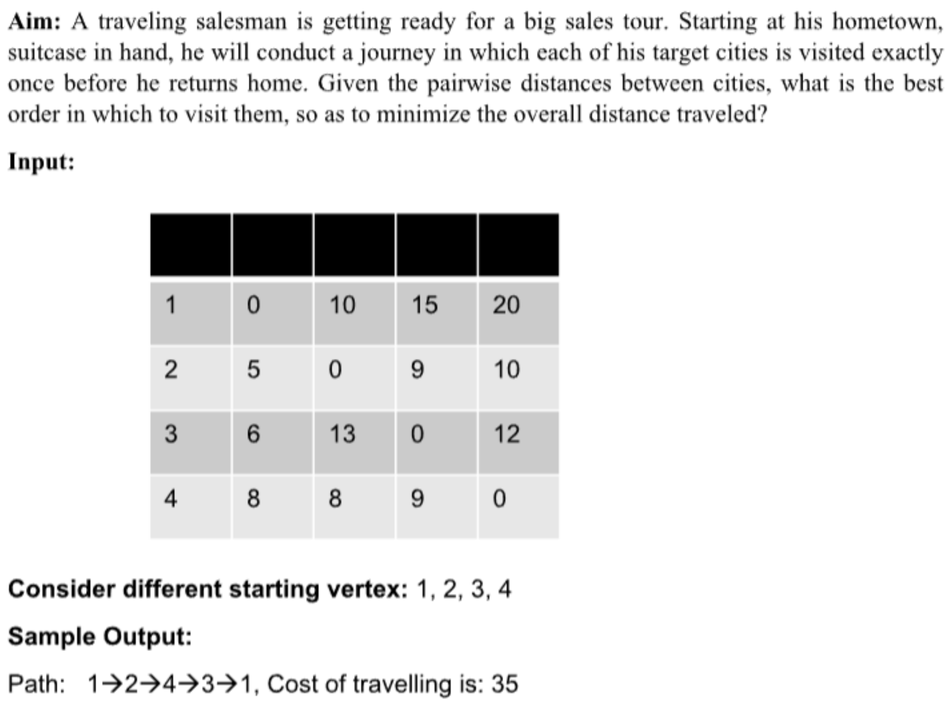
DAA Lab - Practical - 5

Name : Rishabh Jain

Roll No : A-54



Aim: A traveling salesman is getting ready for a big sales tour. Starting at his hometown, suitcase in hand, he will conduct a journey in which each of his target cities is visited exactly once before he returns home. Given the pairwise distances between cities, what is the best order in which to visit them, so as to minimize the overall distance traveled?

Input:

1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

Consider different starting vertex: 1, 2, 3, 4

Sample Output:

Path: 1→2→4→3→1, Cost of travelling is: 35

> Indented block

Solution -

```
matrix = [
[99,10,15,20],
[5,99,9,10],
[6,13,99,12],
[8,8,9,99]
]


def travel(matrix, source):
    n = len(matrix)

    def fun(i, s):
        if not s:
            return (matrix[i][source], [i, source]) if i != source else (float('inf'), [])
        else:
            min_dist = float('inf')
            min_path = []
            for j in s:
                if j != i:
                    dist, path = fun(j, s - {j})
                    dist += matrix[i][j]
                    if dist < min_dist:
                        min_dist = dist
                        min_path = [i] + path
        return min_dist, min_path

    all_cities = set(range(0, n))
    all_cities.remove(source)

    min_dist, min_path = fun(source, all_cities)

    return min_dist, min_path
```

```
for i in range(4):
    city = i;
    distance, path = travel(matrix, city)
    actual_path = " ----> ".join(str(city+1) for city in path)

    print(f"\nDistance from city {city+1} to city {city+1}: {distance}")
    print("Path ==>> ", actual_path)
```

```
Distance from city 1 to city 1: 35
Path ==>>  1 ----> 2 ----> 4 ----> 3 ----> 1

Distance from city 2 to city 2: 35
Path ==>>  2 ----> 4 ----> 3 ----> 1 ----> 2

Distance from city 3 to city 3: 35
Path ==>>  3 ----> 1 ----> 2 ----> 4 ----> 3

Distance from city 4 to city 4: 35
Path ==>>  4 ----> 3 ----> 1 ----> 2 ----> 4
```