LAB 06

Implement the discussion of TSP in python.

```python
import itertools

# Given graph with 4 edges
graph = {
    1: {2: 10, 3: 15, 4: 20},
    2: {1: 10, 3: 35, 4: 25},
    3: {1: 15, 2: 35, 4: 30},
    4: {1: 20, 2: 25, 3: 30},
}

start_edge = 1


def TSP(graph, start_edge):

    # Generate all permutations of cities
    cities = list(graph.keys())
    cities.remove(start_edge)
    permutations = list(itertools.permutations(cities))

    # Add start_edge to each permutation and calculate the total distance
    min_distance = float("inf")
    optimal_route = None
    for permutation in permutations:
        distance = graph[start_edge][permutation[0]]
        for i in range(len(permutation) - 1):
            distance += graph[permutation[i]][permutation[i + 1]]
        distance += graph[permutation[-1]][start_edge]

        # Update the minimum distance and optimal route
        if distance < min_distance:
            min_distance = distance
            optimal_route = (start_edge,) + permutation + (start_edge,)
    return optimal_route, min_distance


# Find the optimal route and minimum distance
optimal_route, min_distance = TSP(graph, start_edge)

# Print the results
print("Optimal route:", optimal_route)
print("Minimum distance:", min_distance)
```

```
PROBLEMS   OUTPUT   TERMINAL   GITLENS   SQL CONSOLE   COMMENTS   DEBUG CONSOLE
PS F:\Github repo\Artificial-Intelligence> & C:/Users/amasw/AppData/Local/Programs/Python/Python311/python.exe "f:/Github repo/Artificial-Intelligence/Lab 06/travellingSalesmanProble
m.py"
Optimal route: (1, 2, 4, 3, 1)
Minimum distance: 80
PS F:\Github repo\Artificial-Intelligence>
```

Implement the discussed approach for tower of Hanoi in Python Language.

```python
def tower_of_hanoi(n, source, destination, temporary, step_count):
    """
    Parameters:
    - n: The number of disks to move.
    - source: The peg from which to move the disks.
    - destination: The peg to which to move the disks.
    - temporary: The peg to use as a temporary holding area.
    - step_count: The number of steps taken so far.
    Returns:
    - The number of steps taken to solve the problem.
    """

    if n == 1:
        step_count += 1
        print(f"{step_count}. Move disk 1 from {source} to {destination}")
        return step_count
    else:
        step_count = tower_of_hanoi(
            n - 1, source, temporary, destination, step_count)
        step_count += 1

        print(f"{step_count}. Move disk {n} from {source} to  {destination}")
        step_count = tower_of_hanoi(
            n - 1, temporary, destination, source, step_count)
        return step_count

step_count = 0
step_count = tower_of_hanoi(3, "A", "C", "B", step_count)
print(f"Total number of steps: {step_count}")
```

```
PROBLEMS   OUTPUT   TERMINAL   GITLENS   SQL CONSOLE   COMMENTS   DEBUG CONSOLE
PS F:\Github repo\Artificial-Intelligence> & C:/Users/amasw/AppData/Local/Programs/Python/Python311/python.exe "f:/Github repo/Artificial-Intelligence/Lab 06/towerOfHanoi.py"
1. Move disk 1 from A to C
2. Move disk 2 from A to  B
3. Move disk 1 from C to B
4. Move disk 3 from A to  C
5. Move disk 1 from B to A
6. Move disk 2 from B to  C
7. Move disk 1 from A to C
Total number of steps: 7
PS F:\Github repo\Artificial-Intelligence>
```