Name: Devanshi Jain
En-No: 22162101006
Batch: 51

Institute of Computer Technology
B. Tech Computer Science and Engineering

Sub: Algorithm Analysis and Design

Practical 12

“Rocket Singh: Salesman of the Year” is a travelling salesman, who sales good in various cities. One day in the morning, he decided to visit all the cities to sales good and come back to the starting city (from where he has started). Travelling Salesman Problem (TSP) is a touring problem in which n cities and distance between each pair is given. We have to help him to find a shortest route to visit each city exactly once and come back to the starting point.

Sample Input:

```
[[∞, 20, 30, 10, 11],
[15, ∞, 16, 4, 2],
[3, 5, ∞, 2, 4],
[19, 6, 18, ∞, 3],
[16, 4, 7, 16, ∞]]
```

Sample Output:

```
Minimum Path
1 – 4 = 10
4 – 2 = 6
2 – 5 = 2
```

5 – 3 = 7

3 – 1 = 3

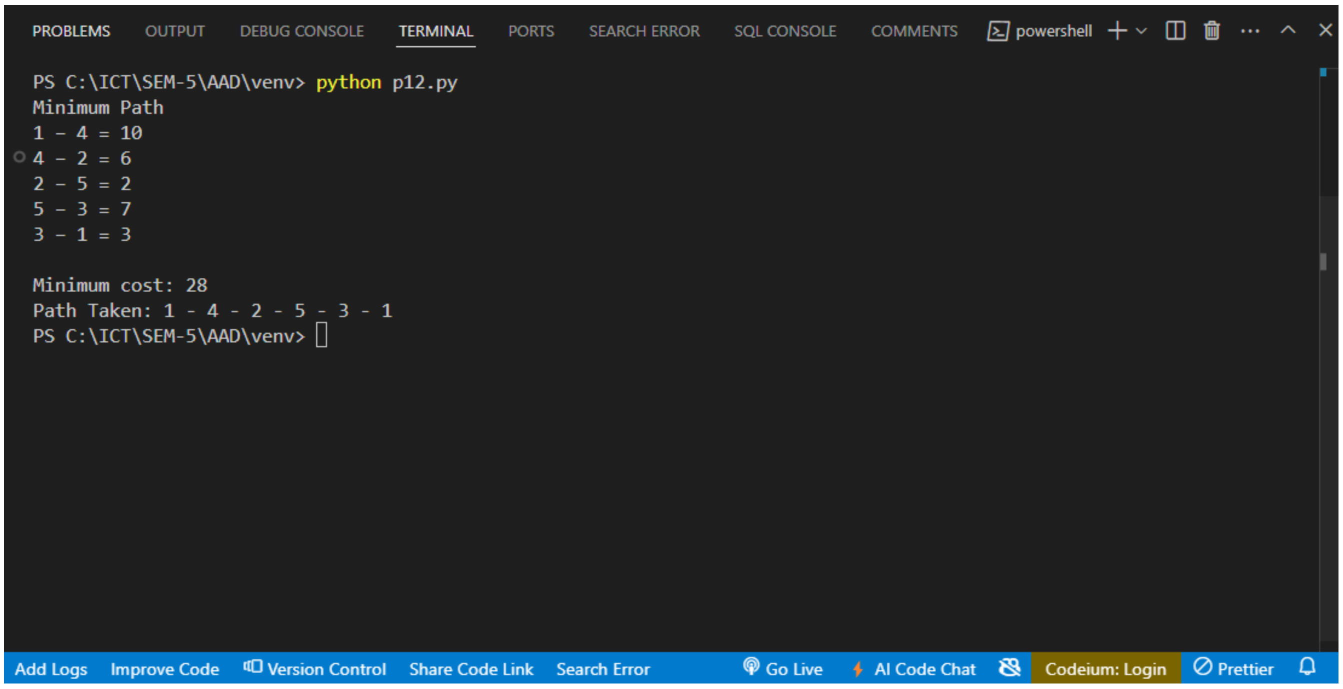
Minimum cost: 28

Path Taken: 1 - 4 - 2 - 5 - 3 – 1

CODE:

```python
import itertools

def calculate_cost(path, distance_matrix):
    total_cost = 0
    for i in range(len(path)):
        total_cost += distance_matrix[path[i]][path[(i + 1) % len(path)]]
    return total_cost

def tsp_brute_force(distance_matrix):
    n = len(distance_matrix)
    cities = list(range(n))

    min_cost = float('inf')
    best_path = None

    # Generate all possible paths (permutations)
    for perm in itertools.permutations(cities):
        cost = calculate_cost(perm, distance_matrix)
        if cost < min_cost:
            min_cost = cost
            best_path = perm

    return min_cost, best_path

def main():
    distance_matrix = [
        [float('inf'), 20, 30, 10, 11],
        [15, float('inf'), 16, 4, 2],
        [3, 5, float('inf'), 2, 4],
        [19, 6, 18, float('inf'), 3],
        [16, 4, 7, 16, float('inf')]
    ]

    min_cost, best_path = tsp_brute_force(distance_matrix)
    print("Minimum Path")
    for i in range(len(best_path)):
        city_from = best_path[i] + 1  # to make it 1-indexed
        city_to = best_path[(i + 1) % len(best_path)] + 1  # to make it 1-indexed
        cost = distance_matrix[best_path[i]][best_path[(i + 1) % len(best_path)]]
        if i == len(best_path) - 1:
            print(f"{city_from} - {city_to} = {cost}")  # Last to first
        else:
            print(f"{city_from} - {city_to} = {cost}")

    print(f"\nMinimum cost: {min_cost}")
    path_taken = ' - '.join(str(city + 1) for city in best_path) + ' - ' +
str(best_path[0] + 1)
    print(f"Path Taken: {path_taken}")

if __name__ == "__main__":
    main()
```

OUTPUT:



```
PS C:\ICT\SEM-5\AAD\venv> python p12.py
Minimum Path
1 - 4 = 10
4 - 2 = 6
2 - 5 = 2
5 - 3 = 7
3 - 1 = 3

Minimum cost: 28
Path Taken: 1 - 4 - 2 - 5 - 3 - 1
PS C:\ICT\SEM-5\AAD\venv>
```