Mini Project - Different exact and approximation algorithms for Travelling-Sales-Person Problem

In [2]:
```python
# 1. Brute-Force Approach (Exact Algorithm)
# Python Implementation

import itertools
import sys
import time
def calculate_total_distance(route, distance_matrix):
    total = 0
    for i in range(len(route)):
        total += distance_matrix[route[i]][route[(i + 1) % len(route)]]
    return total


def brute_force_tsp(distance_matrix):
    n = len(distance_matrix)
    cities = list(range(n))
    min_distance = sys.maxsize
    best_route = []
# Generate all possible permutations of cities
    for perm in itertools.permutations(cities):
        current_distance = calculate_total_distance(perm, distance_matrix)
        if current_distance < min_distance:
            min_distance = current_distance
            best_route = perm
    return best_route, min_distance

# Main execution
if __name__ == "__main__":
# Example distance matrix (symmetric)
    distance_matrix = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
    ]
start_time = time.time()
route, distance = brute_force_tsp(distance_matrix)
end_time = time.time()
print("Optimal Route (Brute-Force):", route)
print("Minimum Distance:", distance)
print(f"Time taken: {end_time - start_time:.4f} seconds")
```

```
Optimal Route (Brute-Force): (0, 1, 3, 2)
Minimum Distance: 80
Time taken: 0.0000 seconds
```

In [3]:
```python
# 2. Nearest Neighbor Algorithm (Approximation Algorithm)
# Python Implementation
import time
def nearest_neighbor_tsp(distance_matrix, start=0):
    n = len(distance_matrix)
    unvisited = set(range(n))
    unvisited.remove(start)
    route = [start]
    total_distance = 0
```

```
    current = start
    while unvisited:
        next_city = min(unvisited, key=lambda city: distance_matrix[current][cit
        total_distance += distance_matrix[current][next_city]
        route.append(next_city)
        current = next_city
        unvisited.remove(next_city)
# Return to start
    total_distance += distance_matrix[current][start]
    route.append(start)
    return route, total_distance
# Main execution
if __name__ == "__main__":
# Example distance matrix (symmetric)
    distance_matrix = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
    ]

start_time = time.time()
route, distance = nearest_neighbor_tsp(distance_matrix)
end_time = time.time()
print("Route (Nearest Neighbor):", route)
print("Total Distance:", distance)
print(f"Time taken: {end_time - start_time:.6f} seconds")
```

```
Route (Nearest Neighbor): [0, 1, 3, 2, 0]
Total Distance: 80
Time taken: 0.000000 seconds
```