```python
from itertools import permutations

def travelling_salesman(graph, start=0):
    n = len(graph)
    vertices = [i for i in range(n) if i != start]

    min_path_cost = float('inf')
    best_path = None

    for perm in permutations(vertices):
        current_cost = 0
        k = start

        for j in perm:
            current_cost += graph[k][j]
            k = j

        current_cost += graph[k][start]  # return to starting point

        if current_cost < min_path_cost:
            min_path_cost = current_cost
            best_path = (start,) + perm + (start,)

    print(f"\nStarting city: {start+1}")
    print(f"Minimum cost: {min_path_cost}")
    print(f"Path: {' -> '.join(str(city+1) for city in best_path)}")

# Example graph
graph = [
    [0, 10, 15, 20],  # city 1
    [10, 0, 35, 25],  # city 2
    [15, 35, 0, 30],  # city 3
    [20, 25, 30, 0]   # city 4
]

# Testing with different starting cities
travelling_salesman(graph, start=0)  # Start from city 1
travelling_salesman(graph, start=1)  # Start from city 2
travelling_salesman(graph, start=2)  # Start from city 3
travelling_salesman(graph, start=3)  # Start from city 4
```

```python
def tower_of_hanoi(n, source, helper, destination):
    if n == 1:
        print(f"Move disk 1 from {source} to {destination}")
        return
    tower_of_hanoi(n-1, source, destination, helper)
    print(f"Move disk {n} from {source} to {destination}")
    tower_of_hanoi(n-1, helper, source, destination)

# Get user input
n = int(input("Enter the number of disks: "))

# Call the function
tower_of_hanoi(n, 'A', 'B', 'C')
print(f"\nTotal moves: {2**n - 1} (2^{n} - 1)")
```