```python
from itertools import permutations

def travelling_salesman(graph, start):
    # Get all vertices except the starting point
    vertices = list(graph.keys())
    vertices.remove(start)

    min_path = float('inf')
    best_route = []

    # Generate all possible routes
    for perm in permutations(vertices):
        current_cost = 0
        k = start

        # Calculate cost of the current path
        for j in perm:
            current_cost += graph[k][j]
            k = j

        # Add cost to return to start
        current_cost += graph[k][start]

        # Update minimum path
        if current_cost < min_path:
            min_path = current_cost
            best_route = (start,) + perm + (start,)

    print("Best Route:", ' -> '.join(best_route))
    print("Minimum Cost:", min_path)


# Example Graph (Adjacency Matrix as Dictionary)
graph = {
    'A': {'A': 0, 'B': 10, 'C': 15, 'D': 20},
    'B': {'A': 10, 'B': 0, 'C': 35, 'D': 25},
    'C': {'A': 15, 'B': 35, 'C': 0, 'D': 30},
    'D': {'A': 20, 'B': 25, 'C': 30, 'D': 0}
}

# Start from node 'A'
travelling_salesman(graph, 'A')
```

```
Best Route: A -> B -> D -> C -> A
Minimum Cost: 80
```