Travelling Salesman Problem (Held-Karp Algorithm)

# Python Code

def tsp\_held\_karp(graph):  
 n = len(graph)  
 memo = {}  
  
 def dp(pos, visited):  
 if (pos, visited) in memo:  
 return memo[(pos, visited)]  
  
 if visited == (1 << n) - 1:  
 return graph[pos][0] # return to starting point  
  
 min\_cost = float('inf')  
 for city in range(n):  
 if visited & (1 << city) == 0:  
 new\_cost = graph[pos][city] + dp(city, visited | (1 << city))  
 min\_cost = min(min\_cost, new\_cost)  
  
 memo[(pos, visited)] = min\_cost  
 return min\_cost  
  
 return dp(0, 1 << 0)  
  
  
# Sample distance matrix (symmetric TSP)  
graph = [  
 [0, 10, 15, 20],  
 [10, 0, 35, 25],  
 [15, 35, 0, 30],  
 [20, 25, 30, 0]  
]  
  
# Run the algorithm  
min\_cost = tsp\_held\_karp(graph)  
  
# Output  
print("Minimum cost to visit all cities and return to the starting point:", min\_cost)

# Output

Minimum cost to visit all cities and return to the starting point: 80