from itertools import permutations

def calculate\_distance(path, distance\_matrix):

total\_distance = 0

for i in range(len(path) - 1):

total\_distance += distance\_matrix[path[i]][path[i + 1]]

total\_distance += distance\_matrix[path[-1]][path[0]] # Return to start

return total\_distance

def tsp\_brute\_force(distance\_matrix):

num\_cities = len(distance\_matrix)

cities = list(range(num\_cities))

min\_path = None

min\_distance = float('inf')

for perm in permutations(cities):

current\_distance = calculate\_distance(perm, distance\_matrix)

if current\_distance < min\_distance:

min\_distance = current\_distance

min\_path = perm

return min\_path, min\_distance

# Example distance matrix (symmetric)

distance\_matrix = [

[0, 10, 15, 20],

[10, 0, 35, 25],

[15, 35, 0, 30],

[20, 25, 30, 0]

]

path, distance = tsp\_brute\_force(distance\_matrix)

print("Optimal path:", path)

print("Minimum distance:", distance)

Output:

