import itertools

def calculate\_total\_distance(permutation, distance\_matrix):

distance = 0

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

distance += distance\_matrix[permutation[i]][permutation[i + 1]]

distance += distance\_matrix[permutation[-1]][permutation[0]] # Return to start

return distance

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

n = len(distance\_matrix)

cities = list(range(n))

min\_distance = float('inf')

best\_path = None

for perm in itertools.permutations(cities[1:]): # Fix city 0 as the starting point

current\_path = [0] + list(perm)

current\_distance = calculate\_total\_distance(current\_path, distance\_matrix)

if current\_distance < min\_distance:

min\_distance = current\_distance

best\_path = current\_path

return best\_path, min\_distance

# Example distance matrix

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(f"Best path: {path}")

print(f"Minimum distance: {distance}")

