SIMULATED ANALYSIS

INPUT:

import random

import math

def simulated\_annealing(cities, initial\_temp, cooling\_rate, max\_iterations):

current\_path = random.sample(cities, len(cities))

current\_cost = calculate\_cost(current\_path)

best\_path = current\_path

best\_cost = current\_cost

temp = initial\_temp

for i in range(max\_iterations):

next\_path = get\_neighbor(current\_path)

next\_cost = calculate\_cost(next\_path)

delta\_cost = next\_cost - current\_cost

if delta\_cost < 0:

current\_path = next\_path

current\_cost = next\_cost

if current\_cost < best\_cost:

best\_path = current\_path

best\_cost = current\_cost

else:

acceptance\_probability = math.exp(-delta\_cost / temp)

if random.random() < acceptance\_probability:

current\_path = next\_path

current\_cost = next\_cost

temp = temp \* cooling\_rate

return best\_path, best\_cost

def calculate\_cost(path):

cost = 0

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

cost += dist(path[i], path[i + 1])

cost += dist(path[-1], path[0])

return cost

def get\_neighbor(path):

new\_path = path.copy()

x = random.randint(0, len(path) - 1)

y = random.randint(0, len(path) - 1)

new\_path[x], new\_path[y] = new\_path[y], new\_path[x]

return new\_path

def dist(city1, city2):

return math.sqrt((city1[0] - city2[0]) \*\* 2 + (city1[1] - city2[1]) \*\* 2)

cities = [(0, 0), (1, 2), (2, 2), (3, 3), (1, 4), (2, 5)]

initial\_temp = 100

cooling\_rate = 0.99

max\_iterations = 1000

best\_path, best\_cost = simulated\_annealing(cities, initial\_temp, cooling\_rate, max\_iterations)

print("Best path: ", best\_path)

print("Best cost: ", best\_cost) 