B~~a~~hri~~a~~ University,



Karachi Campus

LAB EXPERIMENT NO.

\_\_\_\_\_\_\_9\_\_\_\_\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| **1** | Using python implement Hill Climbing Algorithm on any marketing domain to find an optimal solution. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Submitted On~~:~~

\_\_\_\_20/12/2022\_\_\_\_\_\_

(Date~~:~~ DD/MM/YY)

**Task# 01: - Using python implement Hill Climbing Algorithm on any marketing domain to find an optimal solution.**

# Solution: -

distances = [

[0, 800, 1000, 600],

[800, 0, 600, 1000],

[1000, 600, 0, 800],

[600, 1000, 800, 0]

] import random

#Generates a random solution def randomSol(distances):

distances = list(range(len(distances))) sol = []

for i in range(len(distances)):

city = distances[random.randint(0,len(distances) - 1)] sol.append(city) distances.remove(city)

return sol

#Calculating the length of a route def lengthOfRoute(distances,sol): routLength = 0

for i in range(len(sol)):

routLength += distances[sol[i - 1]][sol[i]]

return routLength

#getting the possible neighbors of a solution def getNeighbors(sol):

neighbors = [] for i in range(len(sol)):

for j in range(i + 1,len(sol)):

neighbor = sol.copy() neighbor[i] = sol[i] neighbor[j] = sol[j] neighbors.append(neighbor)

return neighbors

#getting the best possible neighbor def determineBestNeighbor(distances,neighbors):

bestRoutLength = lengthOfRoute(distances,neighbors[0])

bestNeighbor = neighbors[0] for neighbor in neighbors:

currRouteLength = lengthOfRoute(distances,neighbor) if currRouteLength < bestRoutLength: bestRoutLength = currRouteLength bestNeighbor = neighbor

return bestNeighbor,bestRoutLength #the hill climbing algorithm def algo(distances):

currSol = randomSol(distances) currRouteLength = lengthOfRoute(distances,currSol) neighbors = getNeighbors(currSol) bestNeighbor,bestNeighborRoutLength = determineBestNeighbor(distances,neighbors) while bestNeighborRoutLength < currRouteLength:

currSol = bestNeighbor currRouteLength = bestNeighborRoutLength neighbors = getNeighbors(currSol) bestNeighbor,bestNeighborRoutLength = determineBestNeighbor(distances,neighbors)

return currSol,currRouteLength

print(algo(distances))

# Output: -

