## AI\_LAB\_1

Kola Deepti (120cs0151)

### Traveling Salesman problem

import math

maxsize = float('inf')

def copyToFinal(curr\_path):

    final\_path[:N + 1] = curr\_path[:]

    final\_path[N] = curr\_path[0]

def firstMin(adj, i, visited):

    min\_val = maxsize

    for k in range(N):

        if adj[i][k] < min\_val and i != k and not visited[k]:

            min\_val = adj[i][k]

    return min\_val

def secondMin(adj, i, visited):

    first, second = maxsize, maxsize

    for j in range(N):

        if i == j or visited[j]:

            continue

        if adj[i][j] <= first:

            second = first

            first = adj[i][j]

        elif adj[i][j] <= second:

            second = adj[i][j]

    return second

def TSPRec(adj, curr\_bound, curr\_weight, level, curr\_path, visited):

    global final\_res

    if level == N:

        if adj[curr\_path[level - 1]][curr\_path[0]] != 0:

            curr\_res = curr\_weight + adj[curr\_path[level - 1]][curr\_path[0]]

            if curr\_res < final\_res:

                copyToFinal(curr\_path)

                final\_res = curr\_res

        return

    for i in range(N):

        if adj[curr\_path[level - 1]][i] != 0 and not visited[i]:

            temp = curr\_bound

            curr\_weight += adj[curr\_path[level - 1]][i]

            if level == 1:

                curr\_bound -= (firstMin(adj, curr\_path[level - 1], visited) + firstMin(adj, i, visited)) / 2

            else:

                curr\_bound -= (secondMin(adj, curr\_path[level - 1], visited) + firstMin(adj, i, visited)) / 2

            if curr\_bound + curr\_weight < final\_res:

                curr\_path[level] = i

                visited[i] = True

                TSPRec(adj, curr\_bound, curr\_weight, level + 1, curr\_path, visited)

            curr\_weight -= adj[curr\_path[level - 1]][i]

            curr\_bound = temp

            visited = [False] \* len(visited)

            for j in range(level):

                if curr\_path[j] != -1:

                    visited[curr\_path[j]] = True

def TSP(adj):

    global N, final\_res, final\_path

    N = len(adj)

    curr\_bound = 0

    curr\_path = [-1] \* (N + 1)

    visited = [False] \* N

    for i in range(N):

        curr\_bound += (firstMin(adj, i, visited) + secondMin(adj, i, visited))

    curr\_bound = math.ceil(curr\_bound / 2)

    visited[0] = True

    curr\_path[0] = 0

    final\_res = maxsize

    TSPRec(adj, curr\_bound, 0, 1, curr\_path, visited)

adj = [[0, 600, 1000, 1900, 1100],

       [600, 0, 1900, 1900, 1500],

       [1000, 1900, 0, 1700, 1200],

       [1900, 1900, 1700, 0, 1900],

       [1100, 1500, 1200, 1900, 0]]

N = 5

final\_path = [None] \* (N + 1)

visited = [False] \* N

final\_res = maxsize

TSP(adj)

city\_names = ['a', 'b', 'c', 'd', 'e']

print("Minimum cost:", final\_res)

print("Path Taken:", end=' ')

for i in range(N + 1):

    print(city\_names[final\_path[i]], end='  ')

Output:

