**Name:** Jaydeep Solanki

**Roll:** 22BEC059

**Practical 4:** Difference between brute force and nearest neighbor method.

Code: **Brute Force Method**

import time

from itertools import permutations

import numpy as np

import pandas as pd

from tabulate import tabulate

def runM1():

    mat = pd.read\_csv("6citytsp.csv", header=None).values

    table\_data = {}

    for row in range(mat.shape[0]):

        startCity = row

        mat = pd.read\_csv("6citytsp.csv", header=None).values

        cityNames = list(range(mat.shape[0]))

        cityNames.remove(startCity)

        per = list(permutations(cityNames))

        st = time.process\_time()

        bestTourLength = np.inf

        bestTour = []

        for tour in per:

            tourLength = 0

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

                tourLength += mat[tour[i], tour[i + 1]]

                tourLength += mat[tour[i + 1], startCity]

                tourLength += mat[startCity, tour[0]]

            if tourLength < bestTourLength:

                bestTourLength = tourLength

                bestTour = list(tour)

        et = time.process\_time()

        time\_taken\_ms = (et - st) \* 1000

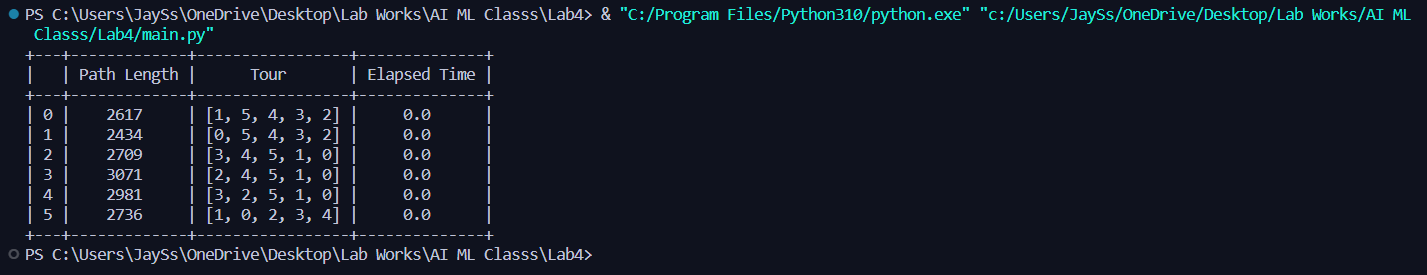
        table\_data[row] = [bestTourLength, bestTour, time\_taken\_ms]

    print(tabulate(pd.DataFrame(table\_data).T, tablefmt="pretty", headers=["Path Length", "Tour", "Elapsed Time"]))

if \_\_name\_\_ == '\_\_main\_\_':

    runM1()

Output:



Code: **Nearest Neighbor Method**

import time

import numpy as np

import pandas as pd

from tabulate import tabulate

def runM2():

    df = pd.read\_csv('6citytsp.csv', header=None).values.astype(float)

    table\_data = {}

    for row in range(df.shape[0]):

        df = pd.read\_csv('6citytsp.csv', header=None).values.astype(float)

        stCity = row

        nextBestCity = 0

        tourLength = 0

        tour = [stCity]

        df[df == 0] = np.inf

        df1 = df.copy()

        st = time.process\_time()

        for i in range(df.shape[0] - 1):

            if i == 0:

                tourLength += min(df[stCity, :])

                nextBestCity = np.argmin(df[stCity, :])

                tour.append(nextBestCity)

                df[:, stCity] = np.inf

                df[:, nextBestCity] = np.inf

            else:

                tourLength += min(df[nextBestCity, :])

                nextBestCity = np.argmin(df[nextBestCity, :])

                tour.append(nextBestCity)

*# df[:, nextBestCity] = np.inf*

        tourLength += df1[nextBestCity, stCity]

        et = time.process\_time()

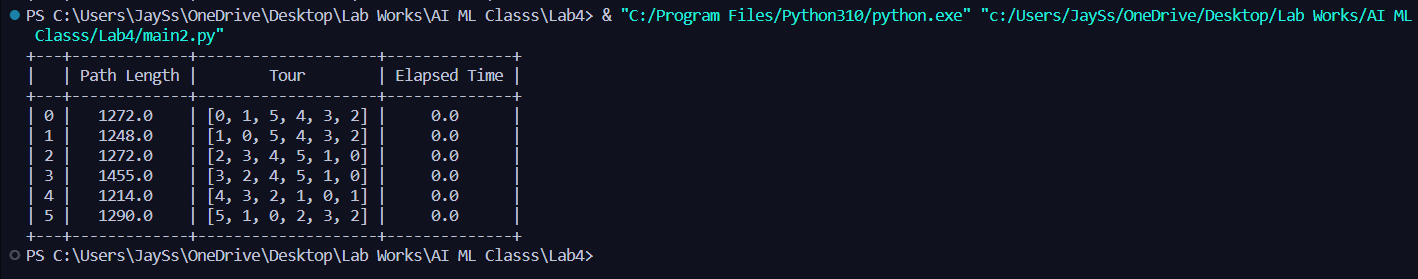
        table\_data[row] = [tourLength, tour, et - st]

    print(tabulate(pd.DataFrame(table\_data).T, tablefmt="pretty", headers=["Path Length", "Tour", "Elapsed Time"]))

if \_\_name\_\_ == '\_\_main\_\_':

    runM2()

Output:



Both Tables:

