In [100…

```
import csv
import pulp
import pandas as pd
from pulp import *

NODESNUMBER = 13

class Node:
    def __init__(self,index, name, latitude, longitude, neighbors, arc_weights):
        self.index = index
        self.name = name
        self.latitude = latitude
        self.longitude = longitude
        self.neighbors = neighbors
        self.arc_weights = arc_weights

    index = None
    name = None
    latitude = None
    longitude = None
    neighbors = None
    arc_weights = None


class Calculations:
    def __init__(self):
        self.source = -1
        self.destination = -1
        self.nodes = []
        self.routes = []
        csv_file = pd.read_csv(r'Dataset.csv')

    # creating nodes in the corresponding graph

        for index, row in csv_file.iterrows():
            node = Node(int(row["Place_index"]),
                                row["Place_name"],
                                float(row["Latitude"]),
                                float(row["Longitude"]),
                                list(map(int, row["Neighbors_indice"].split(","))),
                                list(map(float, row["Neighbors_weight"].split(",")))
                                )

            self.nodes.append(node)

    # creating a 2D list to store arcs and their corresponding weights. This list is not z
    # arcs[i][j] returns the weight of the arc between nodes with index i and j. it returr
    # them

        self.arcs = [[10000 for _ in range(NODESNUMBER+1)] for _ in range(NODESNUMBER+1)]
        for node in self.nodes:
            for i in range(len(node.neighbors)):
                self.arcs[node.index][node.neighbors[i]] = node.arc_weights[i]

        self.lpProblem = LpProblem("FindShortestRoute", LpMinimize)
        current_cost = None

    # define a list to store variables
        self.lp_vars_list = [[0 for _ in range(NODESNUMBER)] for _ in range(NODESNUMBER)]

        for i in range(NODESNUMBER):
            for j in range(NODESNUMBER):
                self.lp_vars_list[i][j] = LpVariable("x"+'_{0:02d}'.format(i)+'_{0:02d}'.f
```

```
    # updating variables with arcs weights
        for i in range(NODESNUMBER):
            for j in range(NODESNUMBER):
                current_cost += self.lp_vars_list[i][j] * arcs[i][j]

    # updating Problem with new variables list
        self.lpProblem += current_cost, "minimizing cuurent_cost"


    # a function to set start and finish nodes as well as initial constraints
    def initialize_constraints(self, src, dst):
        self.source = src
        self.destination = dst
        for i in range(NODESNUMBER):
            left_hand_side = None
            if i != self.source and i != self.destination:
                for j in range(NODESNUMBER):
                    left_hand_side += self.lp_vars_list[i][j] - self.lp_vars_list[j][i]

                self.lpProblem += left_hand_side == 0, "constraint " + '_{0:02d}'.format(i

        left_hand_side = None
        for j in range(NODESNUMBER):
            left_hand_side += self.lp_vars_list[self.source][j]

        self.lpProblem += left_hand_side == 1, "constraint " + '_{0:02d}'.format(self.sour

        # here we initialize constraints for source input
        left_hand_side = None
        for j in range(NODESNUMBER):
            left_hand_side += self.lp_vars_list[j][self.source]

        self.lpProblem += left_hand_side == 0, "constraint " + '_{0:02d}'.format(self.sour

        left_hand_side = None
        for j in range(NODESNUMBER):
            left_hand_side += self.lp_vars_list[j][self.destination]

        self.lpProblem += left_hand_side == 1, "constraint " + '_{0:02d}'.format(self.dest

        # here we initialize constraints for destination output
        left_hand_side = None
        for j in range(NODESNUMBER):
            left_hand_side += self.lp_vars_list[self.destination][j]

        self.lpProblem += left_hand_side == 0, "constraint " + '_{0:02d}'.format(self.dest



    def retrieve_solution(self):
            current_route = {}

            for variable in self.lpProblem.variables():
                if variable.varValue:
                    destination_name = int(variable.name[5:7])
                    source_name = int(variable.name[2:4])
                    current_route[source_name] = destination_name

            current_position = self.source
            while current_position != self.destination:
                self.routes.append(current_position)
                current_position = current_route[current_position]

            self.routes.append(self.destination)
            print("This is the shortest path: ",self.routes)
```

```
            print("Shortest path cost is : ", value(self.lpProblem.objective))


print("Enter start point index \n")
source_index = int(input())

print("Enter destination point index \n")
destination_index = int(input())

problem_solver = Calculations()

problem_solver.initialize_constraints(source_index, destination_index)

problem_solver.lpProblem.solve()
print("Origin point is {}. With {} as index.\n".format(problem_solver.nodes[source_index-1
      "Summarized coordinates of this point are: ## {}, {} ##\n".format(problem_solver.noc
                                                                        problem_solver.nodes[

print("Destination point is {}. With {} as index.\n".format(problem_solver.nodes[destinati
      "Summarized coordinates of this point are: ## {}, {} ## \n".format(problem_solver.no
                                                                        problem_solver.nodes[
problem_solver.retrieve_solution()
```

```
Enter start point index

2
Enter destination point index

12
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /home/aamer/.local/lib/python3.8/site-packages/pulp/apis/../solverdir/cbc/l
inux/64/cbc /tmp/b66a67b02fe74d04a8ef78c4be058ede-pulp.mps timeMode elapsed branch printin
gOptions all solution /tmp/b66a67b02fe74d04a8ef78c4be058ede-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 20 COLUMNS
At line 855 RHS
At line 871 BOUNDS
At line 1041 ENDATA
Problem MODEL has 15 rows, 169 columns and 316 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Continuous objective value is 30 - 0.00 seconds
Cgl0002I 25 variables fixed
Cgl0004I processed model has 13 rows, 133 columns (133 integer (133 of which binary)) and
266 elements
Cutoff increment increased from 1e-05 to 0.9999
Cbc0038I Initial state - 0 integers unsatisfied sum - 0
Cbc0038I Solution found of 30
Cbc0038I Before mini branch and bound, 133 integers at bound fixed and 0 continuous
Cbc0038I Mini branch and bound did not improve solution (0.00 seconds)
Cbc0038I After 0.00 seconds - Feasibility pump exiting with objective of 30 - took 0.00 se
conds
Cbc0012I Integer solution of 30 found by feasibility pump after 0 iterations and 0 nodes
(0.00 seconds)
Cbc0001I Search completed - best objective 30, took 0 iterations and 0 nodes (0.00 second
s)
Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost
Cuts at root node changed objective from 30 to 30
Probing was tried 0 times and created 0 cuts of which 0 were active after adding rounds of
cuts (0.000 seconds)
Gomory was tried 0 times and created 0 cuts of which 0 were active after adding rounds of
```

```
cuts (0.000 seconds)
Knapsack was tried 0 times and created 0 cuts of which 0 were active after adding rounds o
f cuts (0.000 seconds)
Clique was tried 0 times and created 0 cuts of which 0 were active after adding rounds of
cuts (0.000 seconds)
MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were active after ad
ding rounds of cuts (0.000 seconds)
FlowCover was tried 0 times and created 0 cuts of which 0 were active after adding rounds
of cuts (0.000 seconds)
TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after adding rounds
of cuts (0.000 seconds)
ZeroHalf was tried 0 times and created 0 cuts of which 0 were active after adding rounds o
f cuts (0.000 seconds)

Result - Optimal solution found

Objective value:                30.00000000
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             0.00
Time (Wallclock seconds):       0.00

Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.00

Origin point is Lamiz Coffee. With 2 as index.
 Summarized coordinates of this point are: ## 35.70201975, 51.40538127 ##

Destination point is Dey General Hospital. With 12 as index.
 Summarized coordinates of this point are: ## 35.74824752, 51.41152013 ##

This is the shortest path:  [2, 3, 5, 8, 9, 12]
Shortest path cost is :  30.0
```

In [ ]: