AI LAB

19F-0151

Ahmad Raza

[Company name]  [Company address]

**Task:**

from collections import defaultdict  
from itertools import permutations  
  
class Graph:  
  
 def \_\_init\_\_(self):  
 self.graph = defaultdict(list)  
  
 def addEdge(self, u, v):  
 self.graph[u].append(v)  
  
 def printGraph(self):  
 print(self.graph)  
 def generateallsol(self):  
 li = list(self.graph.keys())[:]  
 li.extend(['A'])  
 sol = [x for x in set(permutations(li)) if x[0] == 'A' and x[-1] == 'A']  
 print("All Possible Solutions Are :")  
 print(sol)  
 return sol  
 def findactualpathvalue(self):  
 pass  
 def hillclimbing(self):  
 pass  
*# Driver code*g = Graph()  
  
g.addEdge('A', 'B')  
g.addEdge('A', 'C')  
g.addEdge('A', 'D')  
g.addEdge('B', 'A')  
g.addEdge('B', 'C')  
g.addEdge('B', 'D')  
g.addEdge('C', 'A')  
g.addEdge('C', 'B')  
g.addEdge('C', 'D')  
g.addEdge('D', 'A')  
g.addEdge('D', 'B')  
g.addEdge('D', 'C')  
  
actualvalues = \  
 {'AB': 25,  
 'AD': 15,  
 'BD': 45,  
 'BC': 10,  
 'CD': 5,  
 'AC': 10,  
 'BA': 25,  
 'DA': 15,  
 'DB': 45,  
 'CB': 10,  
 'DC': 5,  
 'CA': 10,  
 }  
  
g.printGraph()  
*#g.hillclimbing()  
#g.findactualpathvalue()*path = g.generateallsol()

**Output:**

Application

Description automatically generated with low confidence