Lab 4

Ahmad Raza

**Task:**

import networkx as nx  
import matplotlib.pyplot as plt  
  
class Graph:  
 def \_\_init\_\_(self):  
 self.visual = []  
 self.graph = {}  
  
 def addEdges(self, a, b):  
 temp = [a, b]  
 self.visual.append(temp)  
  
 def visualize(self):  
 G = nx.Graph()  
 G.add\_edges\_from(self.visual)  
 nx.draw\_networkx(G, node\_size=2500)  
 plt.show()  
  
 def addNode(self, n):  
 if n in self.graph:  
 print('Node is already present in graph')  
 else:  
 self.graph[n] = []  
  
 def addEdge(self, n1, n2):  
 if n1 not in self.graph:  
 print(n1, ' not present in graph')  
 elif n2 not in self.graph:  
 print(n2, ' not present in graph')  
 else:  
 self.graph[n1].append(n2)  
 self.graph[n2].append(n1)  
  
 def DFS(self, startNode, visited, graph, s):  
 if startNode not in visited:  
 print(startNode)  
 visited.add(startNode)  
 for i in graph[startNode]:  
 if i == s:  
 print(s, " - Node Searched!!")  
 exit(0)  
 self.DFS(i, visited, graph, s)  
  
 def DLS(self, startPoint, destination, limit):  
 if startPoint == destination: return True  
 if limit <= 0: return False  
 for i in self.graph[startPoint]:  
 if (self.DLS(i, destination, limit - 1)):  
 return True  
 return False  
  
 def IDDFS(self, startPoint, destination, limit):  
 for i in range(limit):  
 if (self.DLS(startPoint, destination, i)):  
 return True  
 return False  
  
  
  
def main():  
 visited = set()  
 G=Graph()  
 G.addNode("Maldon")  
 G.addNode("Calcton")  
 G.addNode("Tiptree")  
 G.addNode("Fee Ring")  
 G.addNode("Blaxhall")  
 G.addNode("Dunwich")  
 G.addNode("Harwish")  
  
 G.addEdge("Maldon", "Calcton")  
 G.addEdge("Maldon", "Fee Ring")  
 G.addEdge("Tiptree", "Maldon")  
 G.addEdge("Tiptree", "Calcton")  
 G.addEdge("Tiptree", "Fee Ring")  
 G.addEdge("Blaxhall", "Harwish")  
 G.addEdge("Blaxhall", "Fee Ring")  
 G.addEdge("Dunwich", "Harwish")  
 G.addEdge("Dunwich", "Blaxhall")  
 G.addEdge("Harwish", "Tiptree")  
 G.addEdge("Harwish", "Calcton")  
  
 G.addEdges('Maldon', 'Calcton')  
 G.addEdges('Maldon', 'Tiptree')  
 G.addEdges('Maldon', 'Feering')  
 G.addEdges('Calcton', 'Tiptree')  
 G.addEdges('Calcton', 'Harwich')  
 G.addEdges('Tiptree', 'Harwich')  
 G.addEdges('Tiptree', 'Feering')  
 G.addEdges('Feering', 'Blaxhall')  
 G.addEdges('Blaxhall', 'Dunwich')  
 G.addEdges('Blaxhall', 'Harwich')  
 G.addEdges('Dunwich', 'Harwich')  
 G.visualize()  
  
 choice = int(input("1. DFS\n2. DLS\n3. Iterative deepening search\n0. Exit\nChoice: "))  
 if choice == 0:  
 exit(0)  
 elif choice == 1:  
 search = input("Enter City you want to reach: ")  
 if search not in G.graph:  
 print(" Kindly enter valid city name !!!!! ")  
 exit(0)  
 print("Path to reach from Maldon to ", search, " is given below:")  
 G.DFS("Maldon", visited, G.graph, search)  
 elif choice == 2:  
 destination = input("Enter The city you want to go: ")  
 limit = int(input("Enter Depth Limit: "))  
 if limit < 0 or limit > 4:  
 print("Invalid Depth Limit Entered!!")  
 startPoint = "Maldon"  
  
 if G.DLS(startPoint, destination, limit) == True:  
 print(destination, " can be reached!!")  
 else:  
 print(destination, " cannot be reached within provided depth limit!!")  
 elif choice == 3:  
 destination = input("Enter The city you want to go: ")  
  
 limit = int(input("Enter Depth Limit: "))  
 if limit < 0 or limit > 4:  
 print("Invalid Depth Limit Entered!!")  
 startPoint = "Maldon"  
 if G.IDDFS(startPoint, destination, limit) == True:  
 print(destination, " can be reached!!")  
 else:  
 print(destination, " cannot be reached within provided depth limit!!")  
main()

**Output:**

A group of blue balloons

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated