|  |
| --- |
| from collections import defaultdict |
|  |  |
|  | steps=-1 |
|  | found=False |
|  |  |
|  | class Graph: |
|  |  |
|  | def \_\_init\_\_(self): |
|  | self.graph=defaultdict(list) |
|  |  |
|  | def addEdge(self,vertex,neighbour): |
|  | self.graph[vertex].append(neighbour) |
|  |  |
|  | def bfs(self,node,value): |
|  | global found |
|  | visited=[node] |
|  | queue=[node] |
|  | steps=1 |
|  |  |
|  | while queue: |
|  | enode=queue.pop(0) |
|  | if value==enode: |
|  | found=True |
|  | print("Found in %d steps"%steps) |
|  | return |
|  |  |
|  | steps+=1 |
|  | for neighbour in self.graph[enode]: |
|  | if neighbour not in visited: |
|  | visited.append(neighbour) |
|  | queue.append(neighbour) |
|  |  |
|  | def dfs(self,node,visited,value,depth): |
|  | global steps,found |
|  |  |
|  | if depth<0: return |
|  |  |
|  | if node not in visited: |
|  | steps+=1 |
|  | visited.add(node) |
|  | if node==value: |
|  | found=True |
|  | print("Node found in %d steps"%steps) |
|  | return |
|  | for neighbour in self.graph[node]: |
|  | self.dfs(neighbour,visited,value,depth-1) |
|  | if node==1: |
|  | steps-=1 |
|  | return |
|  | steps+=1 |
|  |  |
|  | def iddfs(self,start,value,maxdepth): |
|  | global steps,found |
|  | visited=set() |
|  |  |
|  | for depth in range(1,maxdepth+1): |
|  | self.dfs(start,visited,value,depth) |
|  | if found: |
|  | return |
|  | visited=set() |
|  |  |
|  | def main(): |
|  | global steps,found |
|  | numOfNodes=int(input("Enter the number of nodes in graph:")) |
|  | g=Graph() |
|  |  |
|  | for i in range(1,numOfNodes+1): |
|  | neighbours=list(map(int(input("Enter the neighbours of node %d:"%i).split())) |
|  | for neighbour in neighbours: |
|  | g.addEdge(i,neighbour) |
|  |  |
|  | value=int(input("Enter the element to search:")) |
|  | print("Using BFS:") |
|  | g.bfs(1,value) |
|  | if not found: |
|  | print("Not found") |
|  |  |
|  | print("Using DFS:") |
|  | found=False |
|  | maxdepth=int(input("Enter the max depth:")) |
|  | g.dfs(1,set(),value,maxdepth) |
|  | if not found: |
|  | print("Not found") |
|  |  |
|  | print("Using DFID:") |
|  | steps=-1 |
|  | found=False |
|  | g.iddfs(1,value,maxdepth) |
|  | if not found: |
|  | print("Not found") |
|  |  |
|  | if \_\_name\_\_=="\_\_main\_\_": |
|  | main() |