CSE422 Lab01

Name: Subhi Bhuiyan

ID: 17201116

**Level 1**

file = open('/content/level1.txt',mode='r')

file = file.readlines()

nodeNumber = file[0]

connections = file[1]

enemy = file[15]

sourceNode = 0

connections = int(connections.strip())

enemy = int(enemy.strip())

adj\_list = {}

for i in range(2,connections+2):

node = file[i].strip().split(" ")

node1,node2 = node

node3 = int(node1)

node4 = int(node2)

if node3 not in adj\_list.keys():

adj\_list[node3] = list()

adj\_list[node3].append(node4)

if node4 not in adj\_list.keys():

adj\_list[node4] = list()

adj\_list[node4].append(node3)

print(adj\_list)

#BFS algorithm

from queue import Queue

visited = {}

level = {}

parent = {}

bfs\_traversal\_output = []

queue = Queue()

for node in adj\_list.keys():

visited[node] = False

parent[node] = None

level[node] = -1

visited[sourceNode] = True

level[sourceNode] = 0

queue.put(sourceNode)

while not queue.empty():

u = queue.get()

bfs\_traversal\_output.append(u)

for v in adj\_list[u]:

if not visited[v]:

visited[v] = True

parent[v] = u

level[v] = level[u] + 1

queue.put(v)

print(level[enemy])

**Level 2**

file = open('/content/level2.txt',mode='r')

file = file.readlines()

nodeNumber = file[0]

connections = file[1]

enemy = file[14]

connections = int(connections.strip())

enemy = int(enemy.strip())

adj\_list = {}

for i in range(2,connections+2):

node = file[i].strip().split(" ")

node1,node2 = node

node3 = int(node1)

node4 = int(node2)

if node3 not in adj\_list.keys():

adj\_list[node3] = list()

adj\_list[node3].append(node4)

if node4 not in adj\_list.keys():

adj\_list[node4] = list()

adj\_list[node4].append(node3)

print(adj\_list)

#BFS algorithm run for Nora

from queue import Queue

visited = {}

level = {}

parent = {}

bfs\_traversal\_output = []

queue = Queue()

for node in adj\_list.keys():

visited[node] = False

parent[node] = None

level[node] = -1

sourceNode = 5

visited[sourceNode] = True

level[sourceNode] = 0

queue.put(sourceNode)

while not queue.empty():

u = queue.get()

bfs\_traversal\_output.append(u)

for v in adj\_list[u]:

if not visited[v]:

visited[v] = True

parent[v] = u

level[v] = level[u] + 1

queue.put(v)

print(level[enemy])

noraMove = level[enemy]

#BFS algorithm for Lara

from queue import Queue

visited = {}

level = {}

parent = {}

bfs\_traversal\_output = []

queue = Queue()

for node in adj\_list.keys():

visited[node] = False

parent[node] = None

level[node] = -1

sourceNode = 3

visited[sourceNode] = True

level[sourceNode] = 0

queue.put(sourceNode)

while not queue.empty():

u = queue.get()

bfs\_traversal\_output.append(u)

for v in adj\_list[u]:

if not visited[v]:

visited[v] = True

parent[v] = u

level[v] = level[u] + 1

queue.put(v)

print(level[enemy])

laraMove = level[enemy]

#Comparison for Winner

if noraMove < laraMove:

print("Nora")

elif laraMove < noraMove:

print("Lara")

else:

print("Both")

**Level 3**

file = open('/content/level3.txt',mode='r')

file = file.readlines()

nodeNumber = file[0]

connections = file[1]

enemy = file[16]

connections = int(connections.strip())

enemy = int(enemy.strip())

adj\_list = {}

for i in range(2,connections+2):

node = file[i].strip().split(" ")

node1,node2 = node

node3 = int(node1)

node4 = int(node2)

#construct reverse directed graph

if node4 not in adj\_list.keys():

adj\_list[node4] = list()

adj\_list[node4].append(node3)

print(adj\_list)

k = int(input("Number of participants "))

print("Enter position node for participant ")

p = []

for i in range(k):

p = input()

#BFS algorithm

from queue import Queue

visited = {}

level = {}

parent = {}

bfs\_traversal\_output = []

queue = Queue()

for node in adj\_list.keys():

visited[node] = False

parent[node] = None

level[node] = -1

sourceNode = 9

visited[sourceNode] = True

level[sourceNode] = 0

queue.put(sourceNode)

while not queue.empty():

u = queue.get()

bfs\_traversal\_output.append(u)

for v in adj\_list[u]:

if not visited[v]:

visited[v] = True

parent[v] = u

level[v] = level[u] + 1

queue.put(v)

move\_list = []

for i in range(k):

move\_list = level[p(i)]

print(min(move\_list))