AI Group Assignment

Group Members:

Name	ID	SECTION
1 Temesgen Zewde	UGR/3848/12	2
2 Gizaw Dagne	UGR/6640/12	2

4) Use your A* search & Dijkstra's algorithms with your graph library to calculate Degree, Closeness, and Betweenness centralities on the graph from Question 2. Compare your A* and Dijkstra's Algorithm results. By looking at the graph drawing in the textbook, explain the results.

Dijkstra's and A* shortest path search Algorisms are efficient and selected graph search algorithms to obtain the shortest path that connects two distinct nodes in a graph.

These two graph search algorithms could optimize the shortest path to any reachable nodes from a given node than the other graph search techniques like DFS and BFS. so we will use them to test some of the centrality measures common in Social networks analysis like the centrality of Degree, Closeness and Betweenness.

These measures of centrality have their own definition of 'importance', so understanding how they work helps us to find the best one for our graph visualization applications. We will demonstrate 3 of these centrality measures on the graph given as an individual assignment in Question #2. These are:-

1. Degree
2. Closeness and
3. Betweenness

Degree Centrality:

Degree centrality is the number of links held by each node.

How many direct, 'one hop' connections each node has to other nodes in the network.

```
def degree(g,n):
    G = Nodes
    n  =  len(g.verticies[n]._nbrs)
    return (n/(len(G)-1))
```

Closeness Centrality:

Closeness centrality scores each node based on their 'closeness' or Proximity to all other nodes in the network.

It helps us to find good 'broadcasters' in social networks although it is less worthy if the nodes are highly-connected.

Betweenness Centrality:

measures the number of times a node lies on the shortest path between other nodes.

the value of both closeness Centrality and Betweenness centrality lies between 0 and 1.

To measure the centrality of the graph in the textbook, we have defined 3 functions for degree, closeness, and betweenness respectively.

- **Functions:**
  1. Degree (g, n):

     It takes a graph and one node

     Counts all its neighbors in the graph, and returns the count divided by the number of nodes.

  2. Closeness (g, n):

     It takes a graph and one node

     Sum up the path length of every combination of nodes in the graph and

     Return the node number divided by the path sum

  3. Betweenness (g, n):

Checking:

1 Degree

We can manually count the connections for every node in the graph.

Only Node ***"Sibiu" and "Bucharest"*** have 4 connected neighbor nodes which are the maximum

Let's check it by iteratively calling the Degree function for each node.

```
C=0
deg = []
```

```
for i in G:
   c = degree(g,i)
   if c >= C:
       C = c
       deg.append(i)
```

Result :

```
gizaw@Mabuntu:~/Documents/3rd_year_2/AI Repo$ python3 centrality.py
['Sibiu', 'Bucharest']  has the highest degree
gizaw@Mabuntu:~/Documents/3rd_year_2/AI Repo$
```

2 Closeness

Similar to the degree, we have checked the closest node with the same iteration through every node as follows.

```
closest = ""
B=0
for i in G:
    b = closeness(g,i)
    if b > B:
        B=b
        closest=i
print(closest ," is closest node")
```

Result:

```
gizaw@Mabuntu:~/Documents/3rd_year_2/AI Repo$ python3 centrality.py
Pitesti  is closest node
gizaw@Mabuntu:~/Documents/3rd_year_2/AI Repo$
```

2 Betweenness

Based on the definition of betweenness, anyone could estimate that the most important node in terms of connecting many nodes in the graph will be Bucharest.
It seems the node is like a transmission center (Bridge) for almost all shortest paths on the graph.

To check it, We have done the same process for betweenness centrality like this.

```
A = 0
bridge=""
for i in G:
    b = betweenness(g,i)
    if b > A:
        A=b
        bridge=i
print(bridge ," has the highest betweenness.")
```

Result:

```
gizaw@Mabuntu:~/Documents/3rd_year_2/AI Repo$ python3 centrality.py
Bucharest  has the highest betweenness.
gizaw@Mabuntu:~/Documents/3rd_year_2/AI Repo$
```

The expected result has now been investigated.

Detailed Report:

Here is the full report of the above measure of centrality to every nodes.

Table 1. Measures of centrality using the Dijksra's path solution.

no.	Nodes	Degree	Closeness	Betweenness
1	Sibiu	0.21052631578947367	0.0011900288112238508	0.30994152046783624

2	Arad	0.15789473684210525	0.0008219059566552753	0.2046783625730994
3	Rimnicu_Vilcea	0.15789473684210525	0.0014347202295552368	0.3216374269005848
4	Zerind	0.10526315789473684	0.0014347202295552368	0.3216374269005848
5	Oradea	0.10526315789473684	0.0007444849339759414	0.0007444849339759414
6	Neamt	0.05263157894736842	0.0004347229213380314	0.0
7	Iasi	0.10526315789473684	0.0005565971408483712	0.10526315789473684
8	Craiova	0.15789473684210525	0.0012032932235592148	0.19883040935672514
9	Urziceni	0.15789473684210525	0.001166216547999018	0.4444444444444444
10	Bucharest	0.0014641288433382138	0.0014641288433382138	0.5263157894736842
11	Pitesti	0.15789473684210525	0.0015963703579230382	0.4678362573099415
12	Fagaras	0.10526315789473684	0.0012360939431396785	0.0
13	Drobeta	0.10526315789473684	0.0008819979574784143	0.14619883040935672
14	Giurgiu	0.05263157894736842	0.0009765123091946343	0.0
15	Hirsova	0.10526315789473684	0.0008036545131545554	0.10526315789473684
16	Eforie	0.05263157894736842	0.000595648630008151	0.0
17	Timisoara	0.10526315789473684	0.0006155441085949396	0.06432748538011696
18	Vaslui	0.10526315789473684	0.0007322336981655619	0.19883040935672514
19	Mehadia	0.10526315789473684	0.000744893558630964	0.09941520467836257
20	Lugoj	0.10526315789473684	0.0006629680030705886	0.05263157894736842

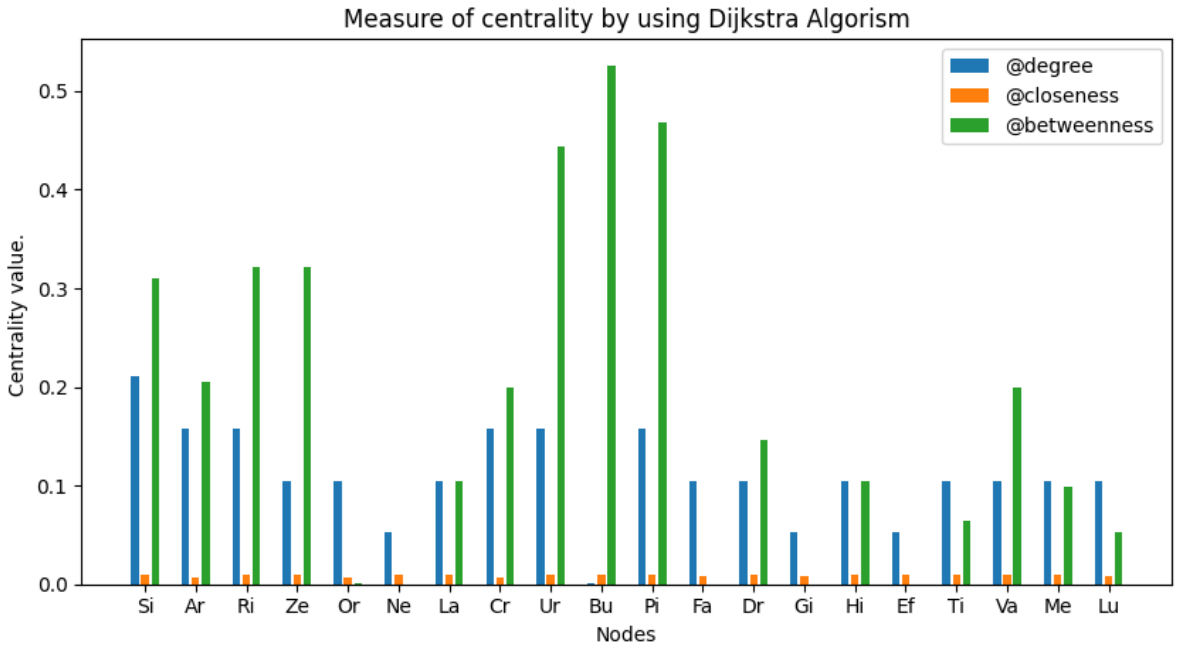

Fig bar graph of centrality measures of every node.

We have repeated the same procedure for A algorithm and the result of the two doesnt vary a lot.*