Introduction to AI - Assignment I

Benchmarking the Algorithms

Ten random cities were chosen and 9 pairs are formed out of them. This table is for labeling the pairs.

Pair	Label
Zerind-Craiova	A
Zerind-Vaslui	B
Zerind-Pitesti	C
Zerind-Rimnicu	D
Zerind-Giurgiu	E
Zerind-Drobeta	F
Zerind-Sibiu	G
Zerind-Hirsova	H
Zerind-Timisoara	I

Experimental results

The labels indicate the city pairs and they are used to minimize the table size/complexity.

N.B: For the heuristic function we used Haversine's geodesic distance formula to calculate the straight line distance between two cities using their geographical latitude and longitude, and we have seen in the experiments that it underestimates the actual costs i.e. the actual travel distances and hence an admissible heuristic.

Summary of the Experiments

This table contains the **average time** taken for each alorithm in seconds for each path search.

The labels A, B, C … are the different city pairs.

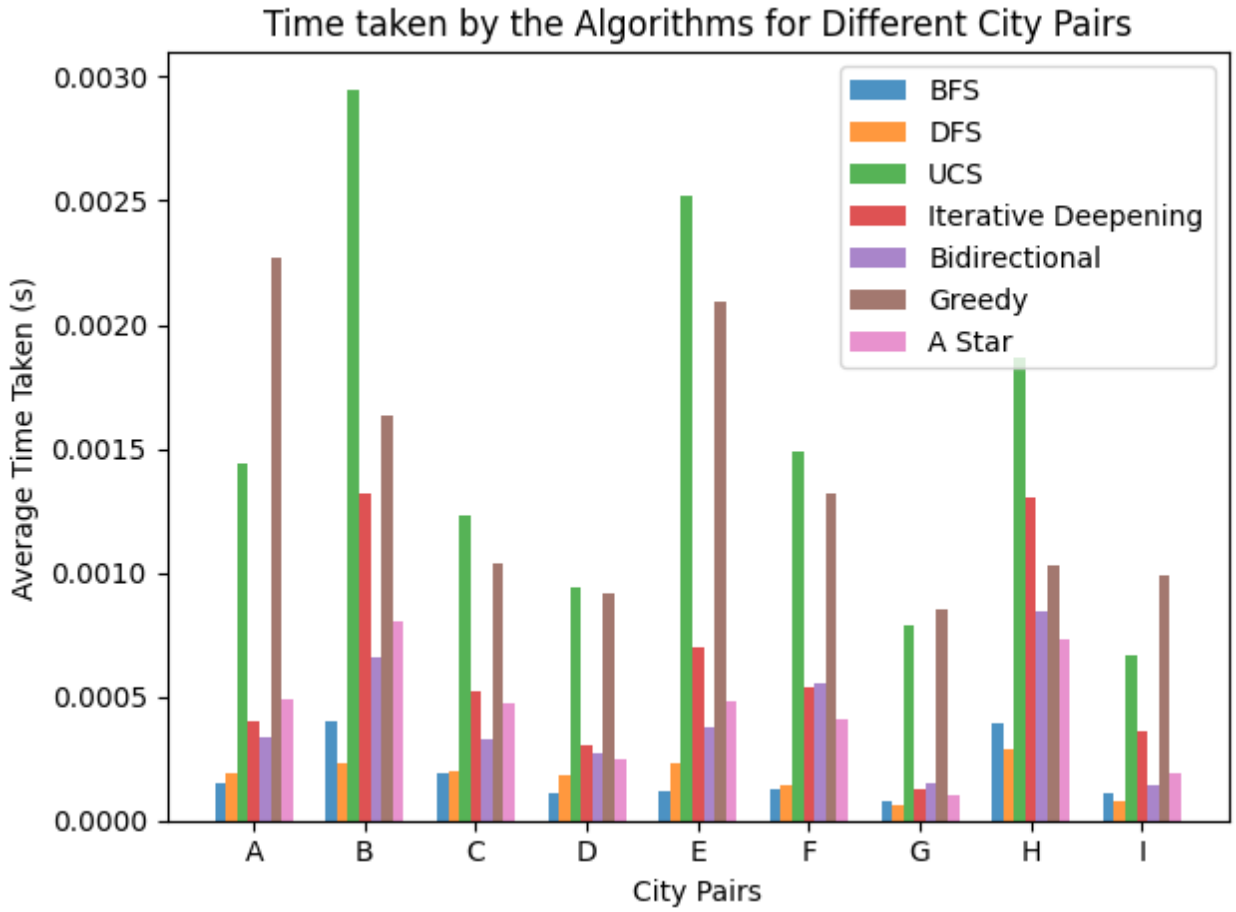
BFS – Breadth First Search

DFS – Depth First Search

UCS – Uniform cost Search

Algorithm	A	B	C	D	E	F	G	H	I
BFS	0.00015	0.0004	0.00019	0.00011	0.00012	0.00013	8e-05	0.00039	0.00011
DFS	0.00019	0.00023	0.0002	0.00018	0.00023	0.00014	6e-05	0.00029	8e-05
UCS	0.00144	0.00295	0.00123	0.00094	0.00252	0.00149	0.00079	0.00187	0.00067
Iterative Deepening	0.0004	0.00132	0.00052	0.0003	0.0007	0.00054	0.00013	0.0013	0.00036
Bidirectional	0.00034	0.00066	0.00033	0.00027	0.00038	0.00055	0.00015	0.00084	0.00014
Greedy	0.00227	0.00163	0.00104	0.00092	0.00209	0.00132	0.00085	0.00103	0.00099
A Star	0.00049	0.0008	0.00047	0.00025	0.00048	0.00041	0.0001	0.00073	0.00019

Figure 1: Time comparison of the Algorithms



This table contains the **costs (in miles)** of the paths found by the algorithms.

The labels A, B, C … are the different city pairs.

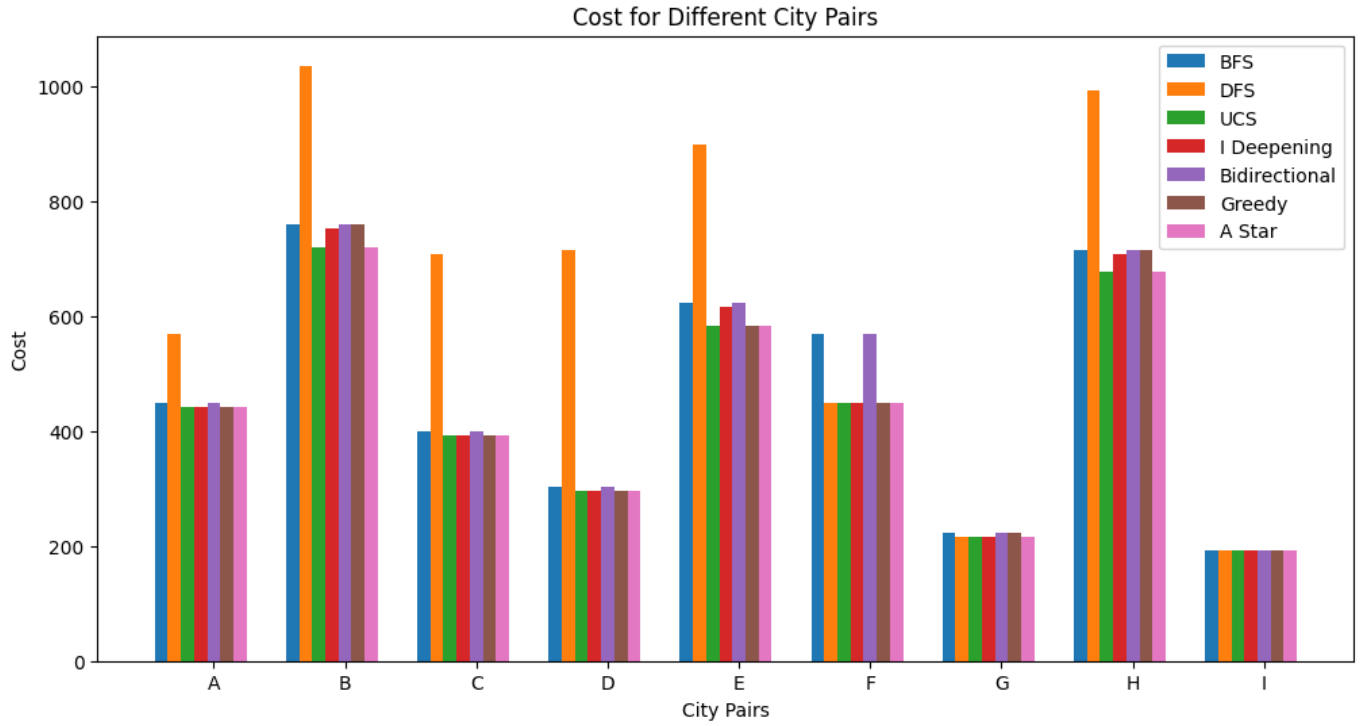
BFS – Breadth First Search

DFS – Depth First Search

UCS – Uniform cost Search

Algorithm	A	B	C	D	E	F	G	H	I
BFS	448	759	399	302	622	568	222	715	193
DFS	569	1035	707	715	898	449	215	991	193
UCS	441	720	392	295	583	449	215	676	193
Iterative Deepening	441	752	392	295	615	449	215	708	193
Bidirectional	448	759	399	302	622	568	222	715	193
Greedy	441	759	392	295	583	449	222	715	193
A Star	441	720	392	295	583	449	215	676	193

Figure 1: Cost comparison of the Algorithms



Review of the data

- From the time data provided, we can see that the performance of each algorithm varies depending on the city pair being tested.
- We can also see that BFS and A* seem to outperform the others on the different pairs of cities being tested.
- It can be difficult to judge the performance of the algorithms based solely on data because there are many factors that can affect the results e.g. implementation of the algorithms, the hardware and software environment in which they are run and others.
- Based on the above analysis, we can see that the BFS and Bidirectional algorithms have the same average cost.
- We can also see that UCS and A* have always exactly the same cost and guaranteed to find the shortest path.
- The DFS has by far high cost compared to the rest of it.

Creating random graphs (Erdős–Rényi model graphs with fixed probabilities)

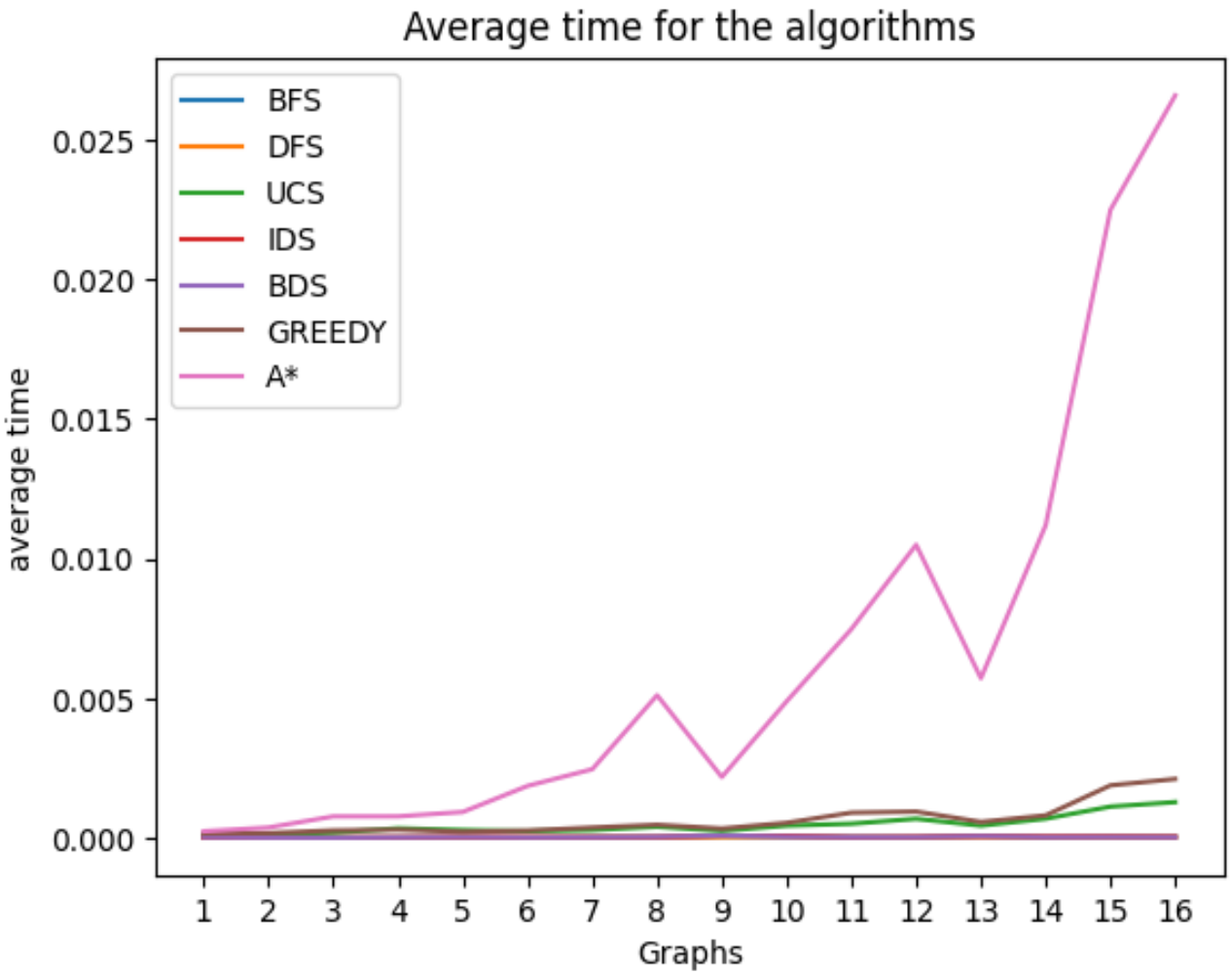
Random graphs with a number of nodes n = 10, 20, 30, 40 were created. The nodes were randomly connected with the probability of edges p = 0.2, 0.4, 0.6, 0.8 and with randomly generated edge weights between 1 and 10. In total, will be 16 different graphs. Five nodes are selected randomly and the algorithms are applied. Here is the average time of the time taken to find a solution for each algorithm and graph.

This table contains the average time in seconds taken by the algorithms.

Graphs	BFS	DFS	UCS	IDS	BDS	GREEDY	A*
1	1.67e-05	1.17e-05	1.03e-04	1.83e-05	2.33e-05	1.50e-04	2.35e-04
2	8.00e-06	1.60e-05	9.60e-05	1.30e-05	9.00e-06	1.70e-04	3.69e-04
3	1.00e-05	1.50e-05	1.84e-04	9.00e-06	6.00e-06	2.66e-04	7.68e-04
4	1.30e-05	1.20e-05	3.40e-04	1.50e-05	1.80e-05	2.90e-04	7.70e-04
5	2.40e-05	2.40e-05	2.79e-04	2.70e-05	2.80e-05	2.19e-04	9.27e-04
6	1.10e-05	2.10e-05	2.39e-04	1.60e-05	1.60e-05	2.62e-04	1.86e-03
7	2.10e-05	1.70e-05	2.83e-04	2.40e-05	2.20e-05	3.69e-04	2.46e-03
8	2.30e-05	1.40e-05	3.97e-04	3.00e-05	3.30e-05	4.67e-04	5.11e-03
9	5.40e-05	3.50e-05	2.49e-04	7.80e-05	8.80e-05	3.34e-04	2.18e-03
10	3.70e-05	4.60e-05	4.37e-04	7.10e-05	4.00e-05	5.37e-04	4.89e-03
11	2.80e-05	1.60e-05	5.06e-04	2.20e-05	2.40e-05	9.02e-04	7.48e-03
12	1.60e-05	1.90e-05	6.83e-04	3.30e-05	2.60e-05	9.41e-04	1.05e-02
13	6.30e-05	2.80e-05	4.38e-04	6.40e-05	5.60e-05	5.70e-04	5.72e-03
14	1.90e-05	3.30e-05	6.86e-04	4.30e-05	2.80e-05	8.05e-04	1.12e-02
15	2.90e-05	2.80e-05	1.12e-03	4.60e-05	2.90e-05	1.88e-03	2.25e-02
16	3.40e-05	3.00e-05	1.28e-03	3.50e-05	2.40e-05	2.11e-03	2.66e-02

	Graphs															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of nodes	10	10	10	10	20	20	20	20	30	30	30	30	40	40	40	40
Probability (p)	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8

- **The heuristic function we designed for these random graphs is a modified version of dijkstra's algorithm that returns the shortest path cost between the start and end nodes and this will never overestimate the actual cost which makes it admissible. But as we can see from the results of the experiment, the heuristic function tends to take some time to calculate the heuristic value and this in turn has affected the overall performance of the A* and the greedy algorithms.**
- **A new efficient heuristic function might be needed to make the A* algorithm perform better in terms of the average time taken by it.**
- **The average time taken by the A* is high due to the heuristic function used.**
- **BFS and DFS are relatively performed faster.**



N.B: The numbers 1, 2, 3, 4… on the x axis indicate the 16 different graphs with different probability and number of nodes. For example, for graph 16, number of nodes = 40 and probability = 0.8

Node centrality rankings

Degree centrality

Here are ten top ranked cities using degree centrality.

Rank	City	Degree centrality
1	Sibiu	4
2	Bucharest	4
3	Arad	3
4	Craiova	3
5	Rimnicu	3
6	Pitesti	3
7	Urziceni	3
8	Oradea	2
9	Zerind	2
10	Timisoara	2

Closeness centrality

Here are ten top ranked cities using closeness centrality.

Rank	City	closeness centrality
1	Pitesti	0.00340
2	Rimnicu	0.00327
3	Bucharest	0.00310
4	Craiova	0.00294
5	Sibiu	0.00289
6	Fagaras	0.0027941
7	Urziceni	0.0027937
8	Drobeta	0.002473
9	Giurgiu	0.002454
10	Arad	0.002432

Eigenvector centrality

Here are ten top ranked cities using eigenvector centrality.

Rank	City	eigenvector centrality
1	Bucharest	0.41921
2	Pitesti	0.38052
3	Fagaras	0.37225
4	Craiova	0.36318
5	Sibiu	0.36142
6	Rimnicu	0.3561
7	Arad	0.19946
8	Oradea	0.18126

9	Urziceni	0.15026
10	Drobeta	0.13929

PageRank centrality

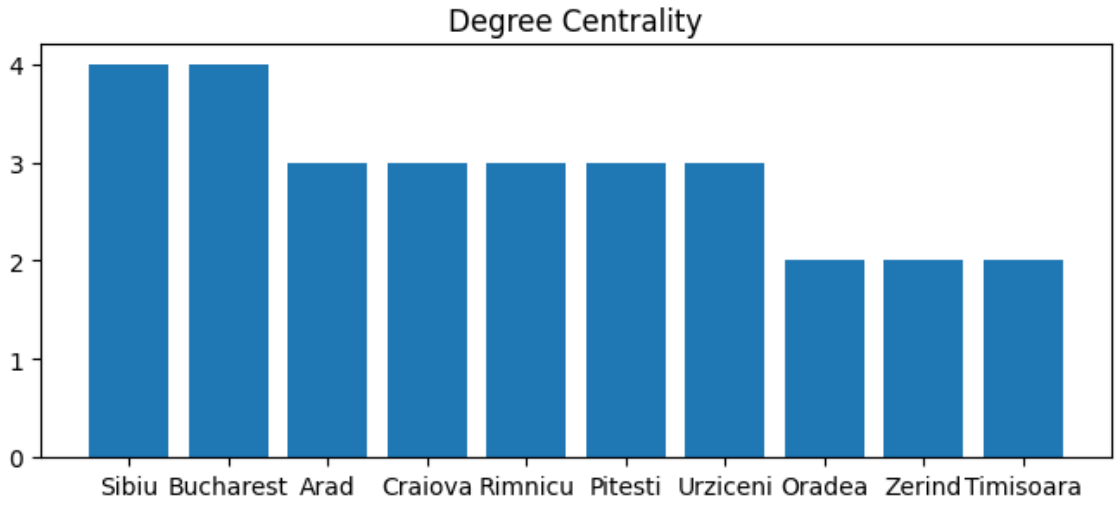
Here are ten top ranked cities using pagerank centrality.

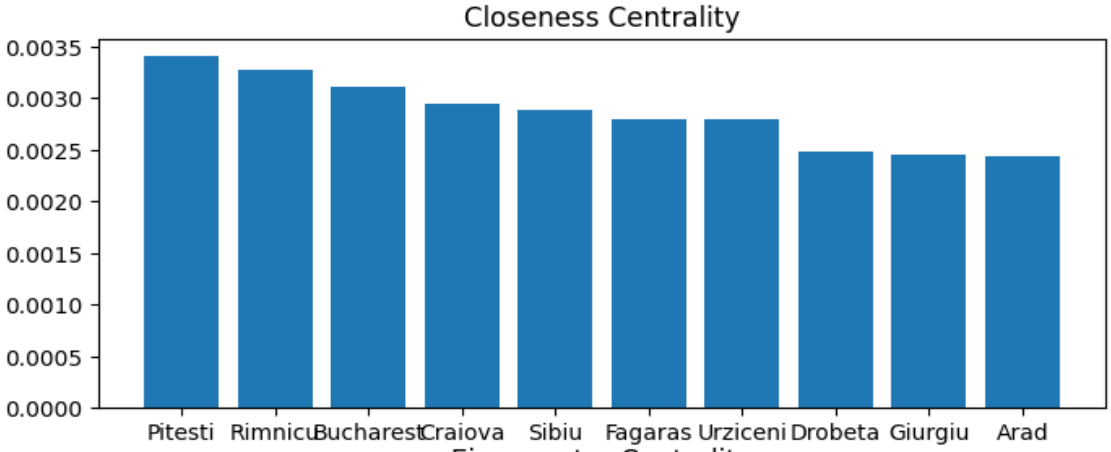
Rank	City	PageRank centrality
1	Bucharest	0.08146
2	Sibiu	0.0753
3	Urziceni	0.0683
4	Arad	0.05995
5	Craiova	0.05783
6	Pitesti	0.05708
7	Rimnicu	0.05606
8	Iasi	0.0552
9	Hirsova	0.05202
10	Vaslui	0.05031

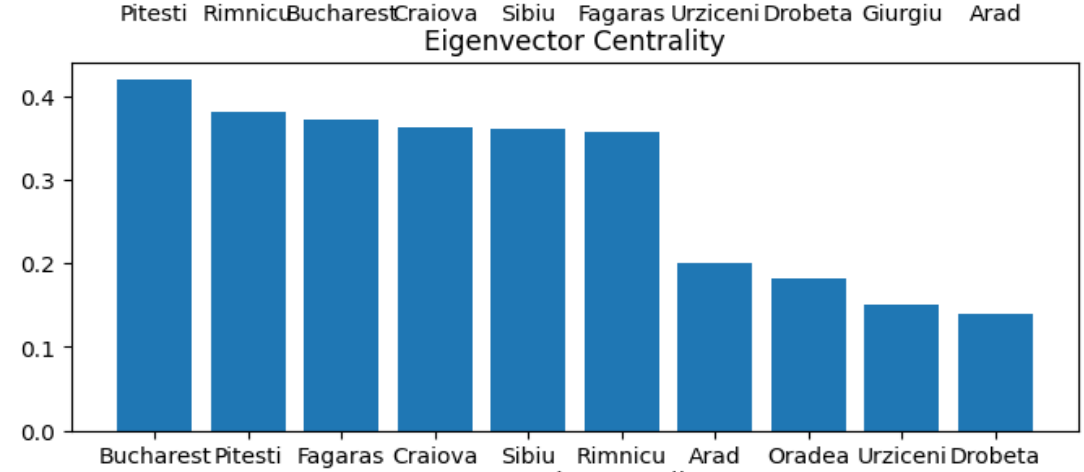
Betweenness centrality

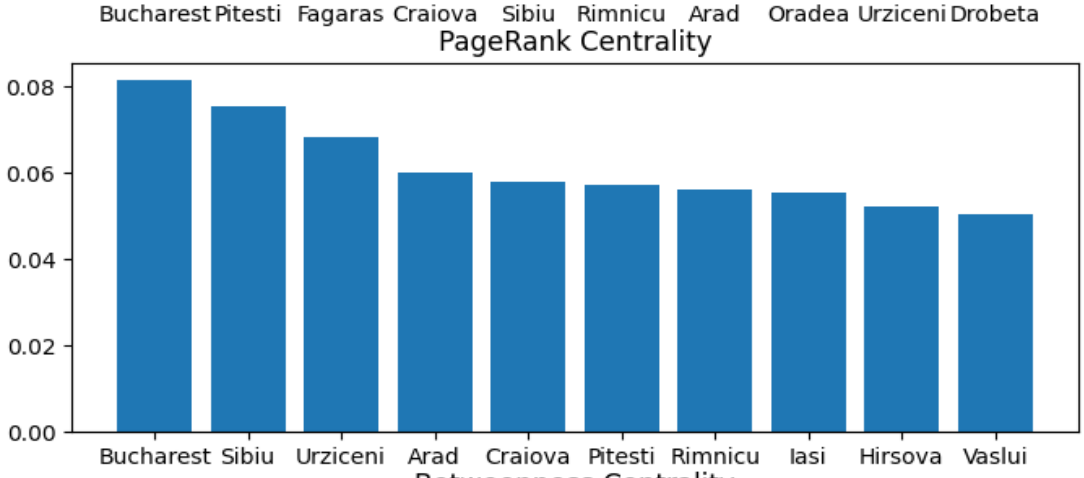
Top ten cities ranked using betweenness centrality.

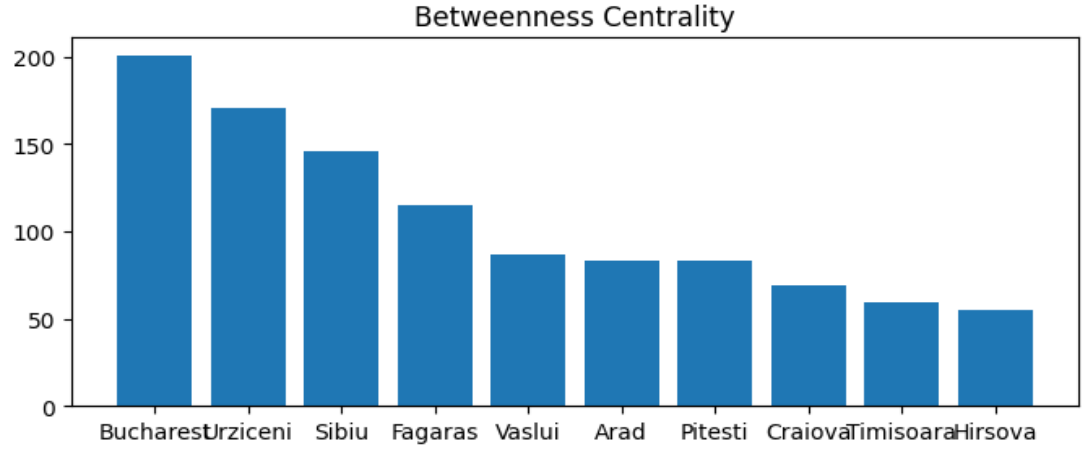
Rank	City	betweenness centrality
1	Bucharest	201.0
2	Urziceni	171.0
3	Sibiu	146.0
4	Fagaras	115.0
5	Vaslui	87.0
6	Arad	83.0
7	Pitesti	83.0
8	Craiova	69.0
9	Timisoara	59.0
10	Hirsova	55.0


Degree Centrality
4
3
2
1
0
Sibiu
Bucharest
Arad
Craiova
Rimnicu
Pitesti
Urziceni
Oradea
Zerind
Timisoara


Closeness Centrality
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
Pitesti
Rimnicu
Bucharest
Craiova
Sibiu
Fagaras
Urziceni
Drobeta
Giurgiu
Arad


Eigenvector Centrality
0.4
0.3
0.2
0.1
0.0
Bucharest
Pitesti
Fagaras
Craiova
Sibiu
Rimnicu
Arad
Oradea
Urziceni
Drobeta


PageRank Centrality
0.08
0.06
0.04
0.02
0.00
Bucharest
Sibiu
Urziceni
Arad
Craiova
Pitesti
Rimnicu
Iasi
Hirsova
Vaslui


Betweenness Centrality
200
150
100
50
0
Bucharest
Urziceni
Sibiu
Fagaras
Vaslui
Arad
Pitesti
Craiova
Timisoara
Hirsova

Conclusion

Based on the centralities calculated for the graph of cities and distances, we can observe the following:

- ✓ The cities with the highest degree centrality are Sibiu and Bucharest, which have four edges each. This suggests that these cities are well-connected to other cities in the network.
- ✓ The cities with the highest closeness centrality are Pitesti, Rimnicu, and Bucharest. This suggests that these cities are the most central and accessible in the network, as they have the shortest distances to all other cities in the graph.
- ✓ The cities with the highest eigenvector centrality are Bucharest, Pitesti, and Fagaras. This suggests that these cities are not only well-connected, but are also connected to other well-connected cities in the network.
- ✓ The cities with the highest PageRank centrality are Bucharest, Sibiu, and Urziceni. This suggests that these cities are the most influential in the network.
- ✓ The cities with the highest betweenness centrality are Bucharest, Urziceni, and Sibiu. This suggests that these cities play a critical role in connecting other cities in the network, as they lie on the shortest paths between many pairs of cities.