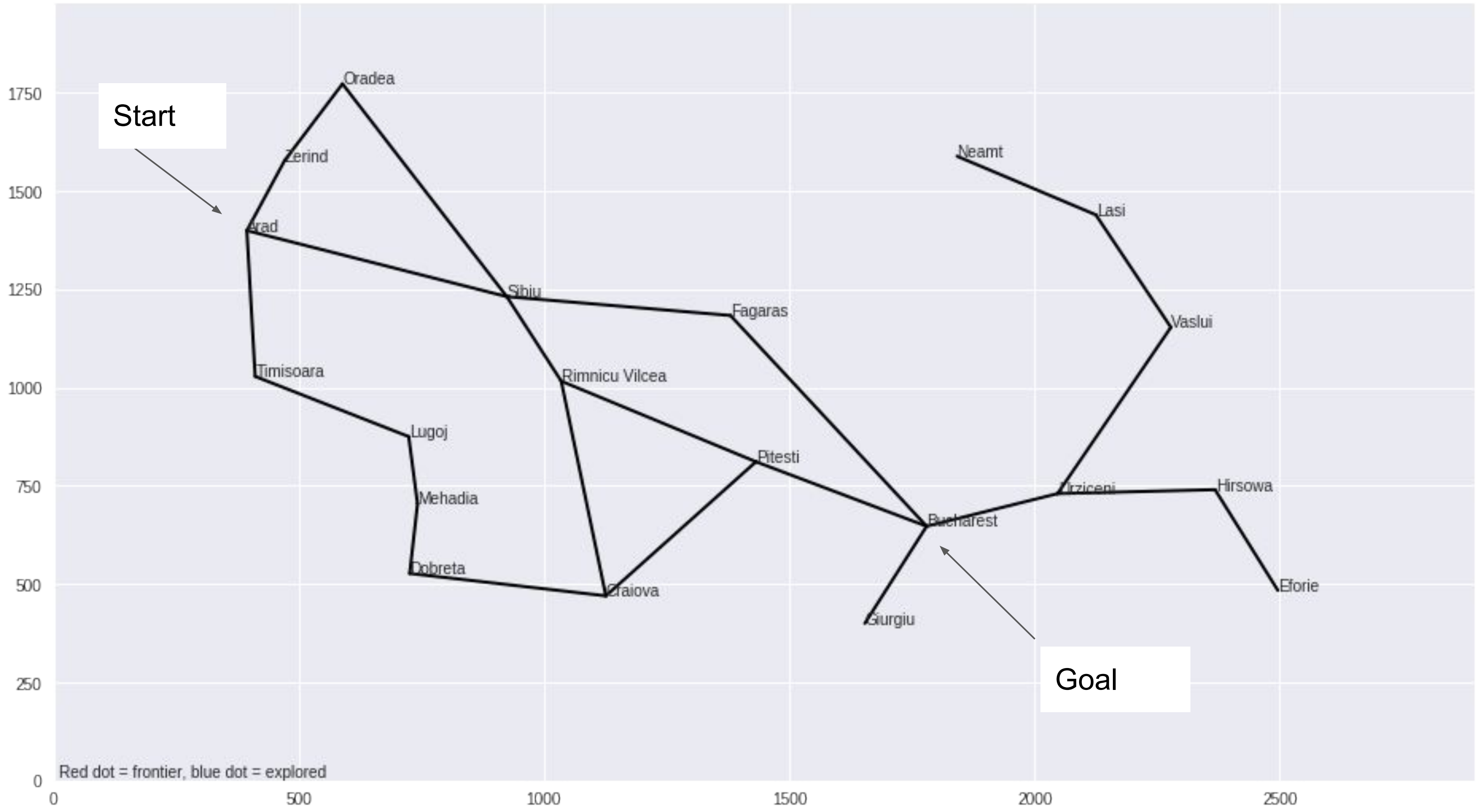
A*

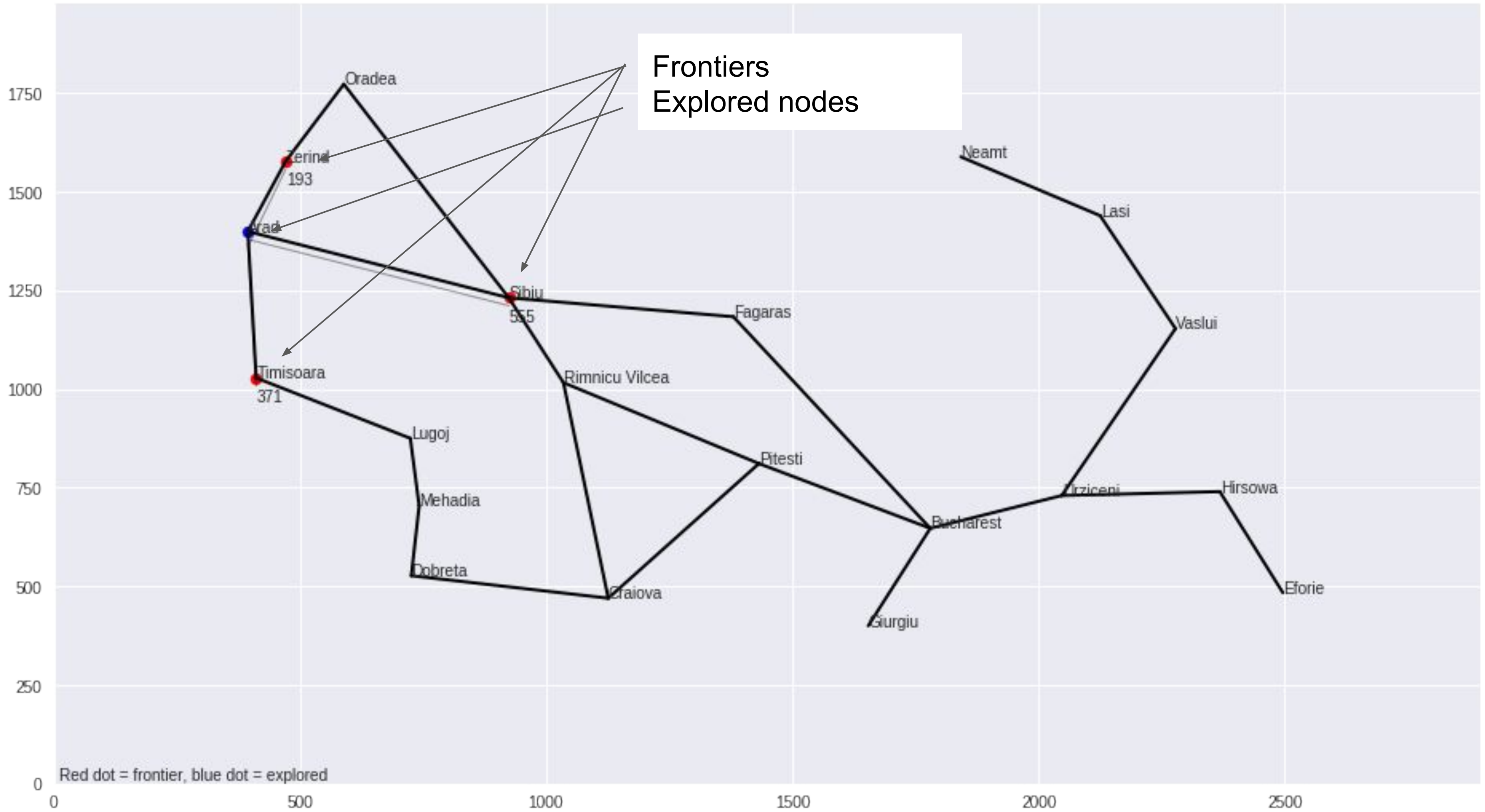
And uniform cost search

First Uniform Cost search


Oradea
Zerind
Arad
Sibiu
Fagaras
Timisoara
Rimnicu Vilcea
Lugoj
Pitesti
Mehadia
Dobreta
Craiova
Giurgiu
Bucharest
Urziceni
Hirsowa
Eforie
Neamt
Lasi
Vaslui
Red dot = frontier, blue dot = explored
0
250
500
750
1000
1250
1500
1750
0
500
1000
1500
2000
2500

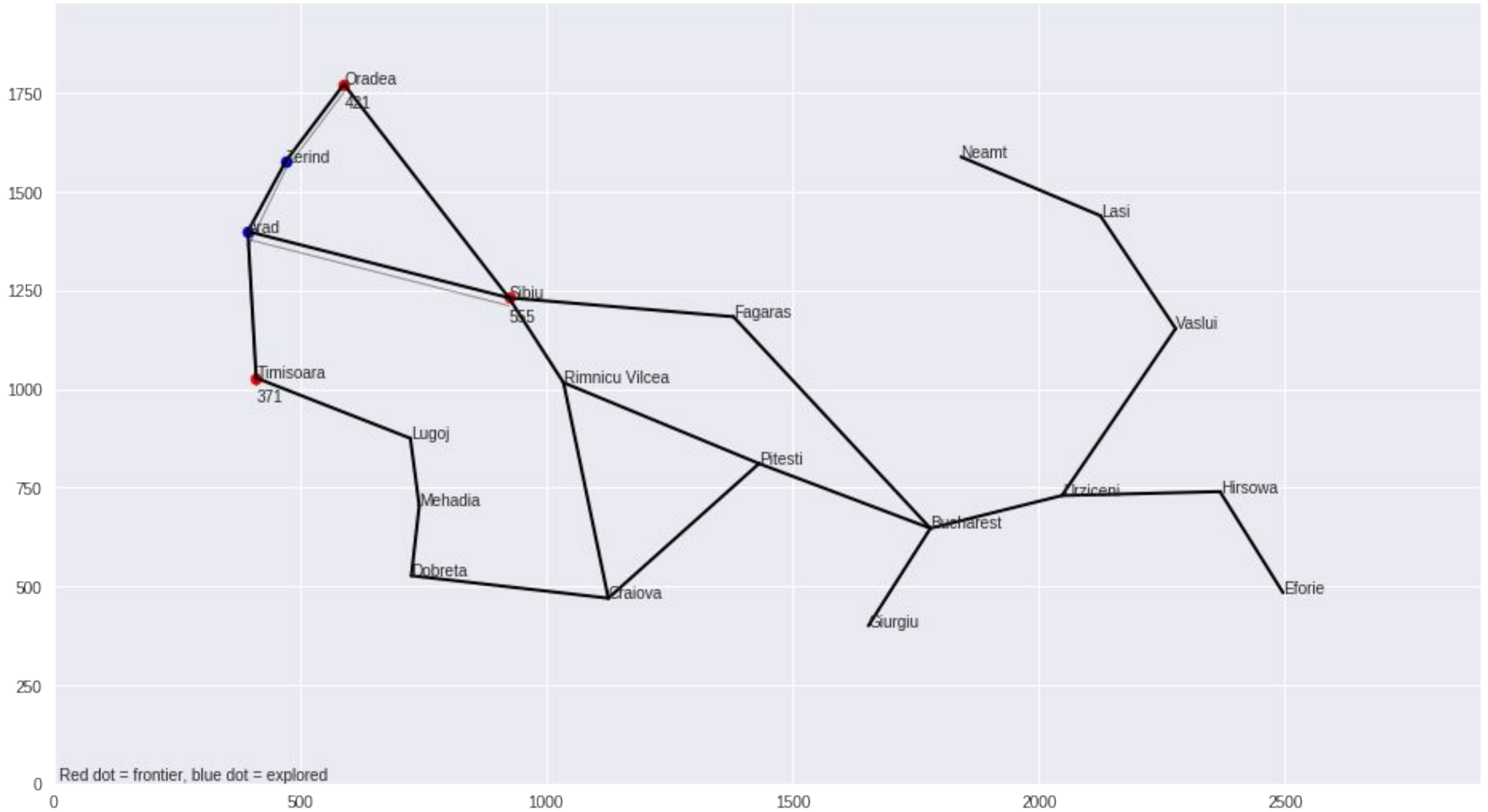
Start

Goal

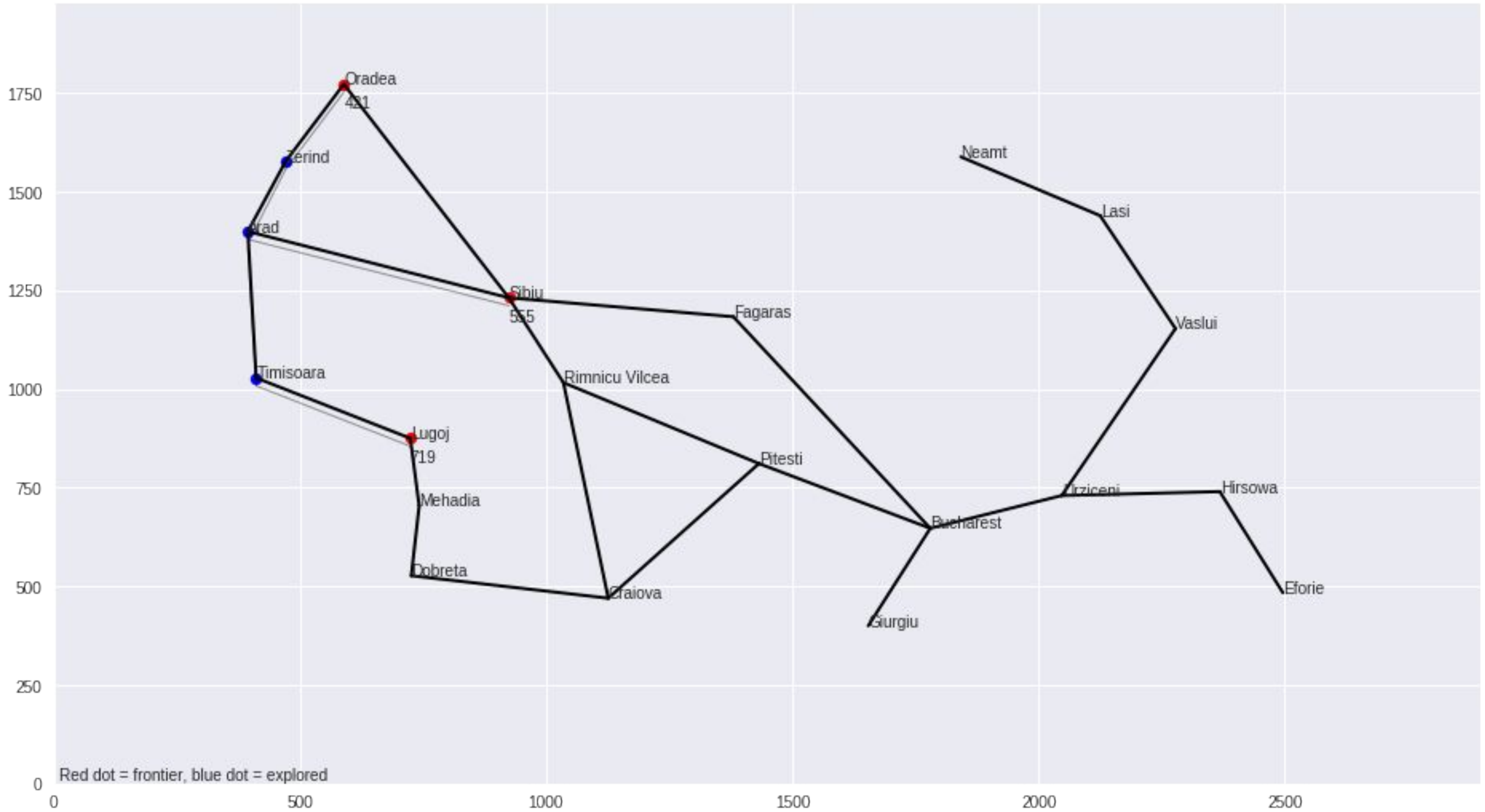


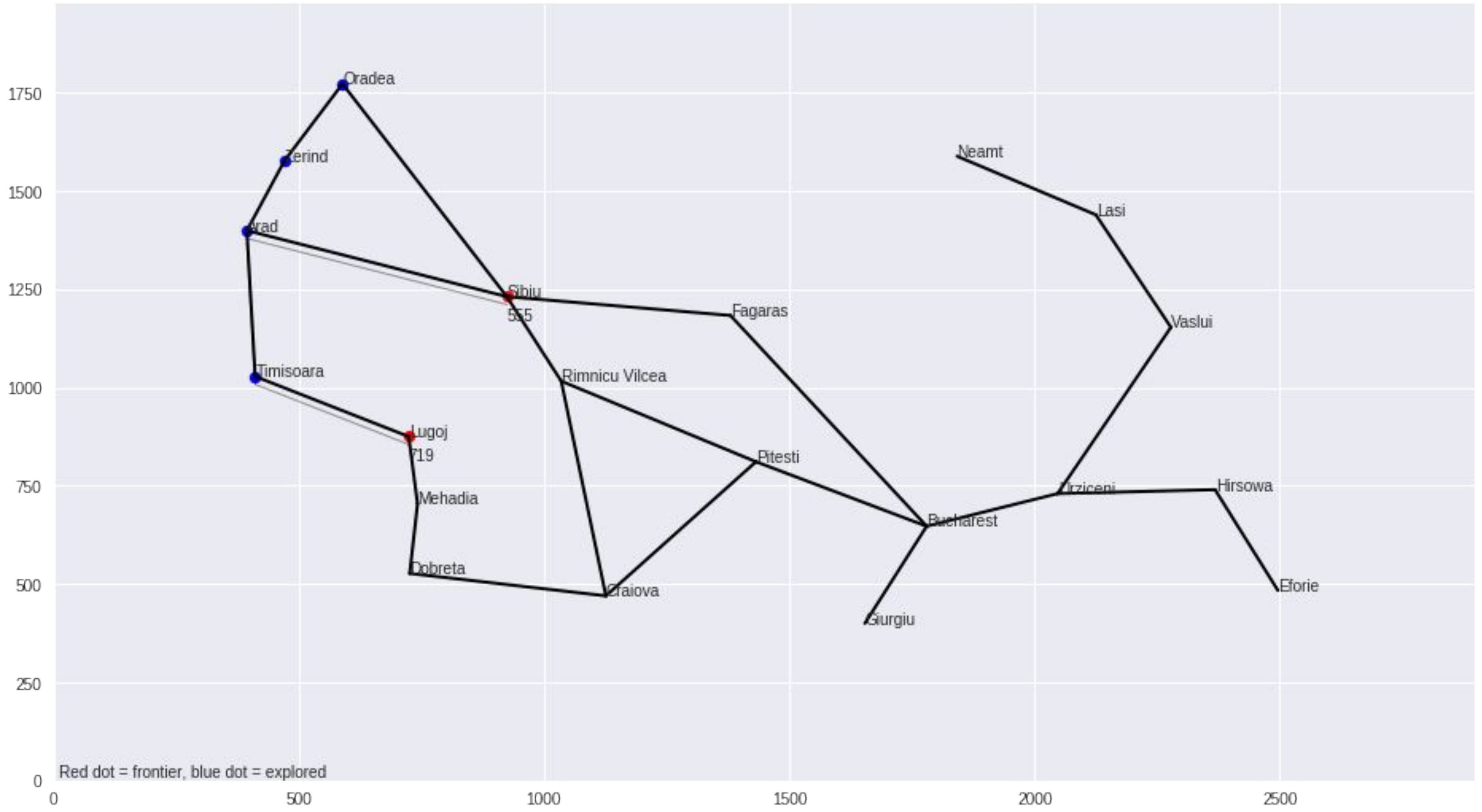
Frontiers

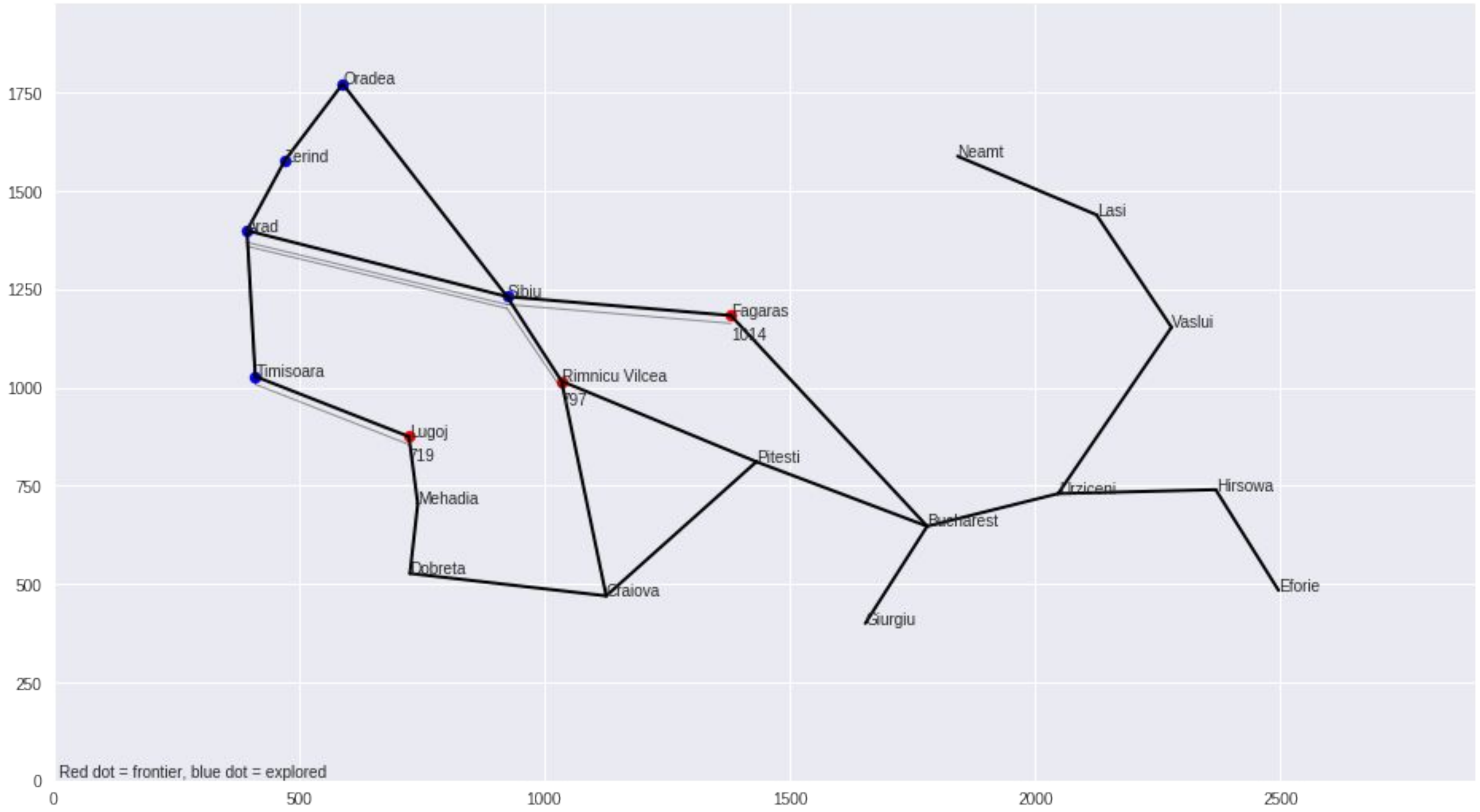
Explored nodes

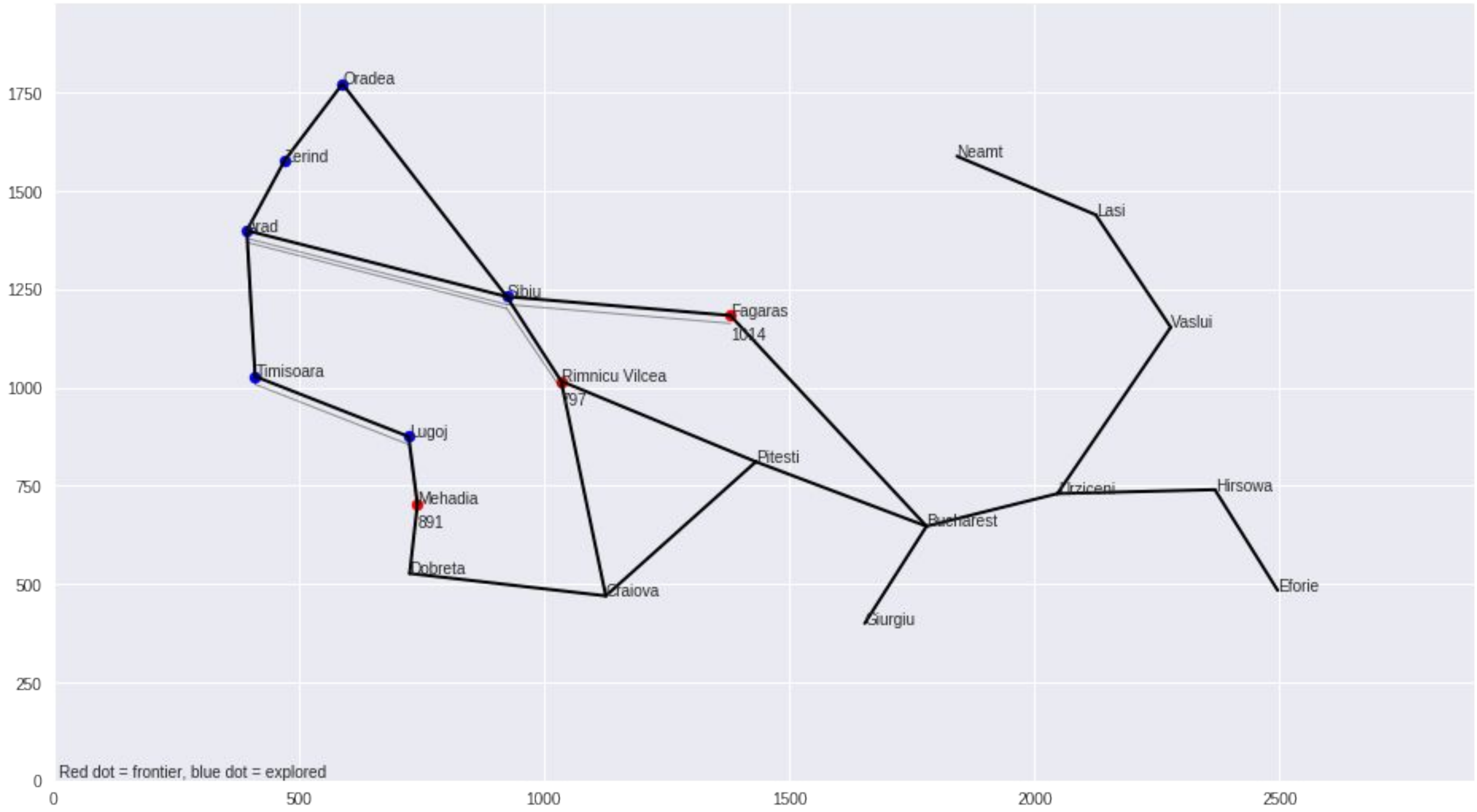


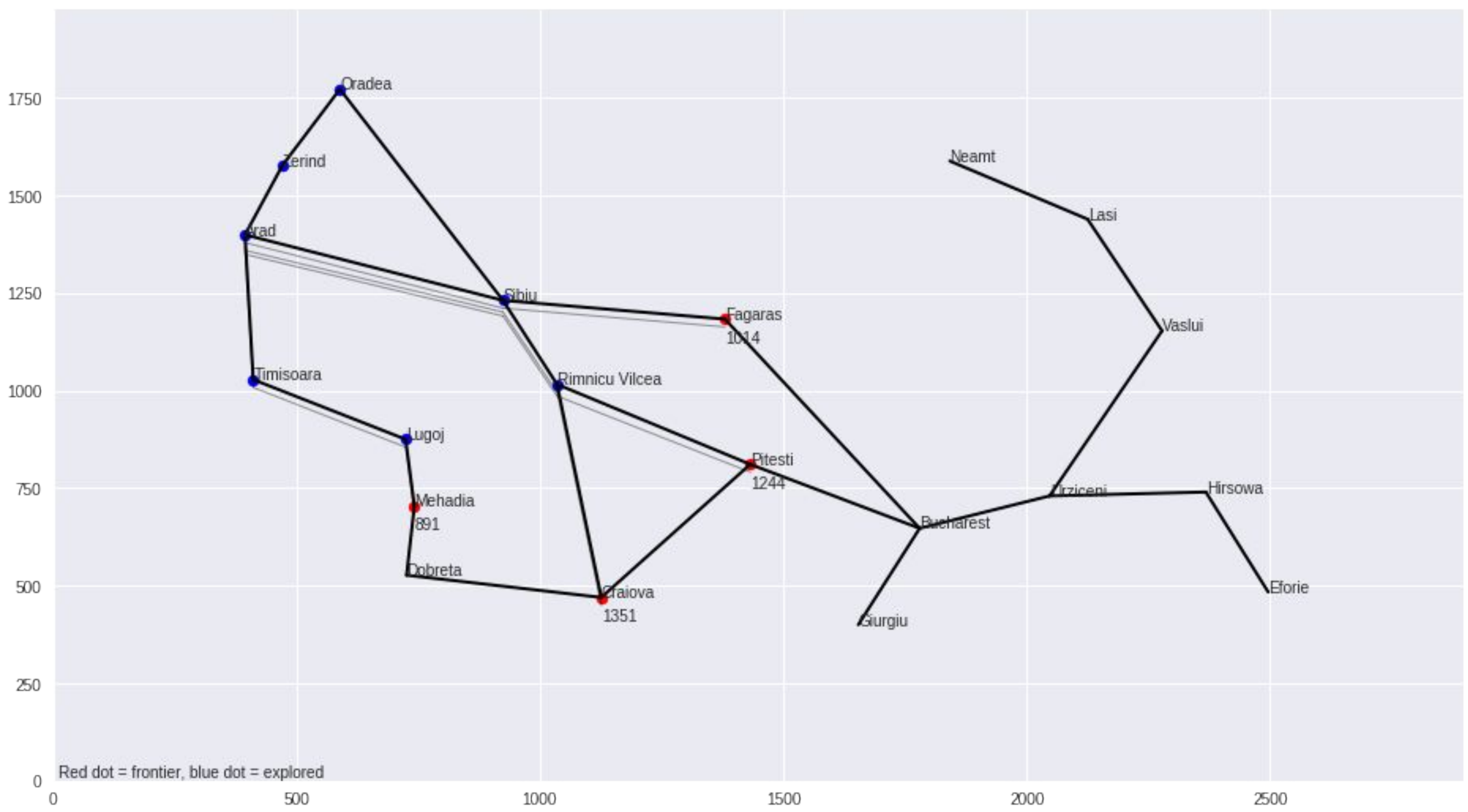
Oradea
421
Zerind
Arad
Sibiu
555
Fagaras
Timisoara
371
Rimnicu Vilcea
Lugoj
Pitesti
Mehadia
Dobreta
Craiova
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Bucharest
Eforie
Giurgiu
Red dot = frontier, blue dot = explored
1750
1500
1250
1000
750
500
250
0
0
500
1000
1500
2000
2500


Oradea
421
Zerind
Arad
Sibiu
555
Fagaras
Timisoara
Rimnicu Vilcea
Lugoj
719
Pitesti
Mehadia
Dobreta
Craiova
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Bucharest
Eforie
Giurgiu
Red dot = frontier, blue dot = explored
0
250
500
750
1000
1250
1500
1750
0
500
1000
1500
2000
2500

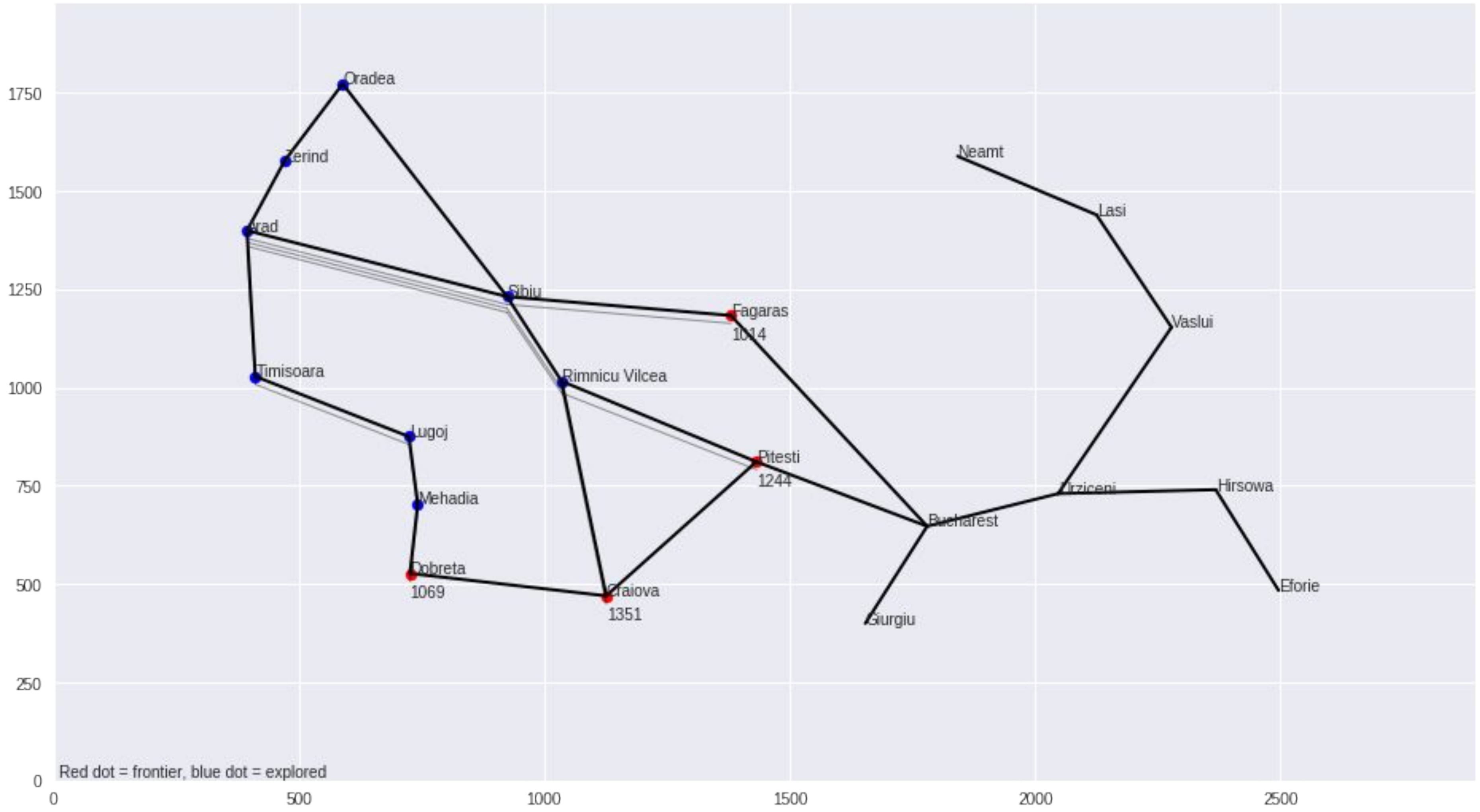

Oradea
Zerind
Arad
Sibiu
555
Fagaras
Timisoara
Rimnicu Vilcea
Lugoj
719
Pitesti
Mehadia
Dobreta
Craiova
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Bucharest
Giurgiu
Eforie
Red dot = frontier, blue dot = explored
1750
1500
1250
1000
750
500
250
0
0
500
1000
1500
2000
2500


Oradea
Zerind
Arad
Sibiu
Fagaras
1014
Timisoara
Rimnicu Vilcea
797
Lugoj
719
Mehadia
Dobreta
Craiova
Pitesti
Bucharest
Giurgiu
Urziceni
Hirsowa
Eforie
Vaslui
Lasi
Neamt
1750
1500
1250
1000
750
500
250
0
0
500
1000
1500
2000
2500
Red dot = frontier, blue dot = explored

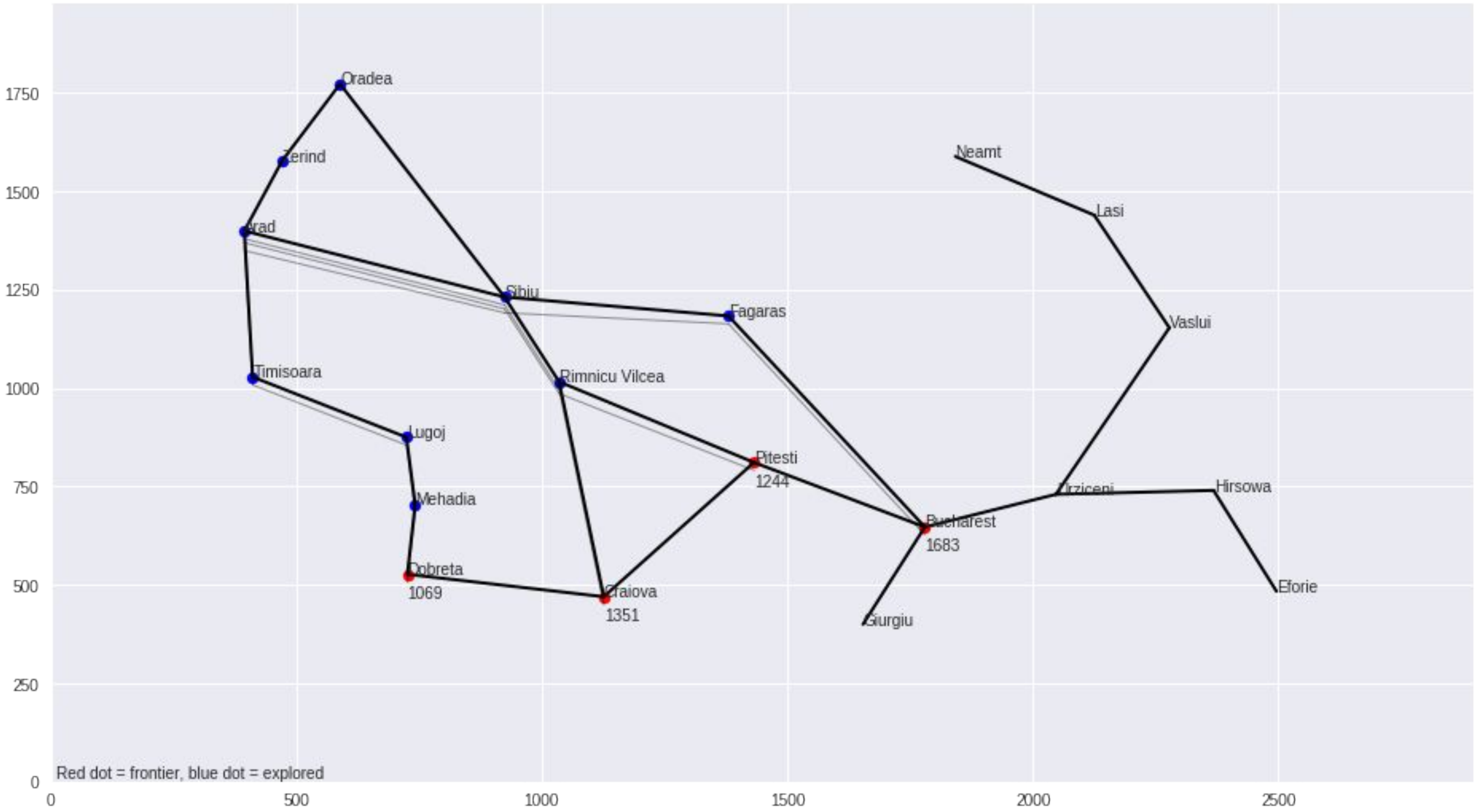

Oradea
Zerind
Arad
Sibiu
Fagaras
1014
Timisoara
Rimnicu Vilcea
797
Lugoj
Mehadia
891
Dobreta
Craiova
Pitesti
Bucharest
Giurgiu
Urziceni
Hirsowa
Eforie
Vaslui
Lasi
Neamt
Red dot = frontier, blue dot = explored
0
250
500
750
1000
1250
1500
1750
0
500
1000
1500
2000
2500



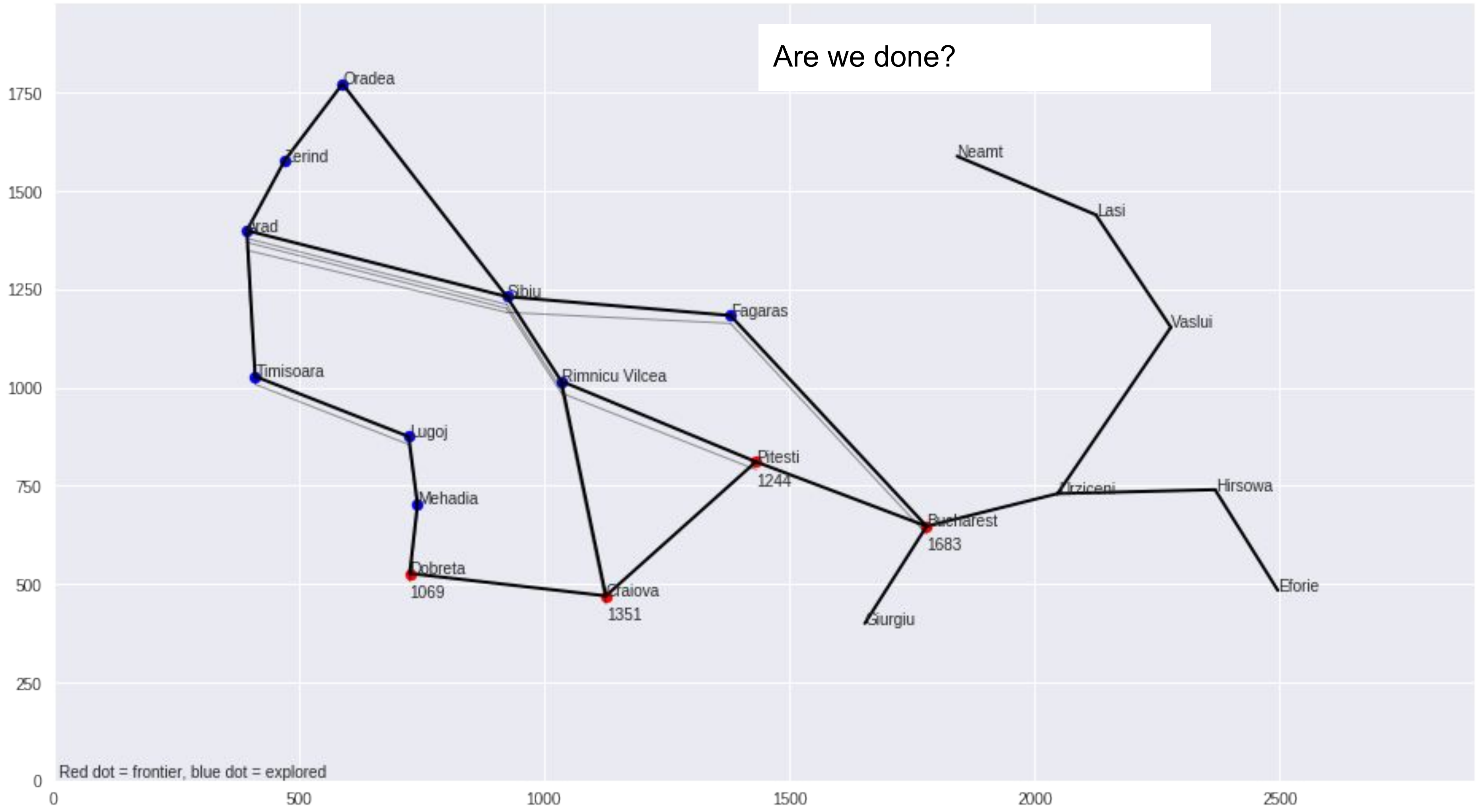
Red dot = frontier, blue dot = explored



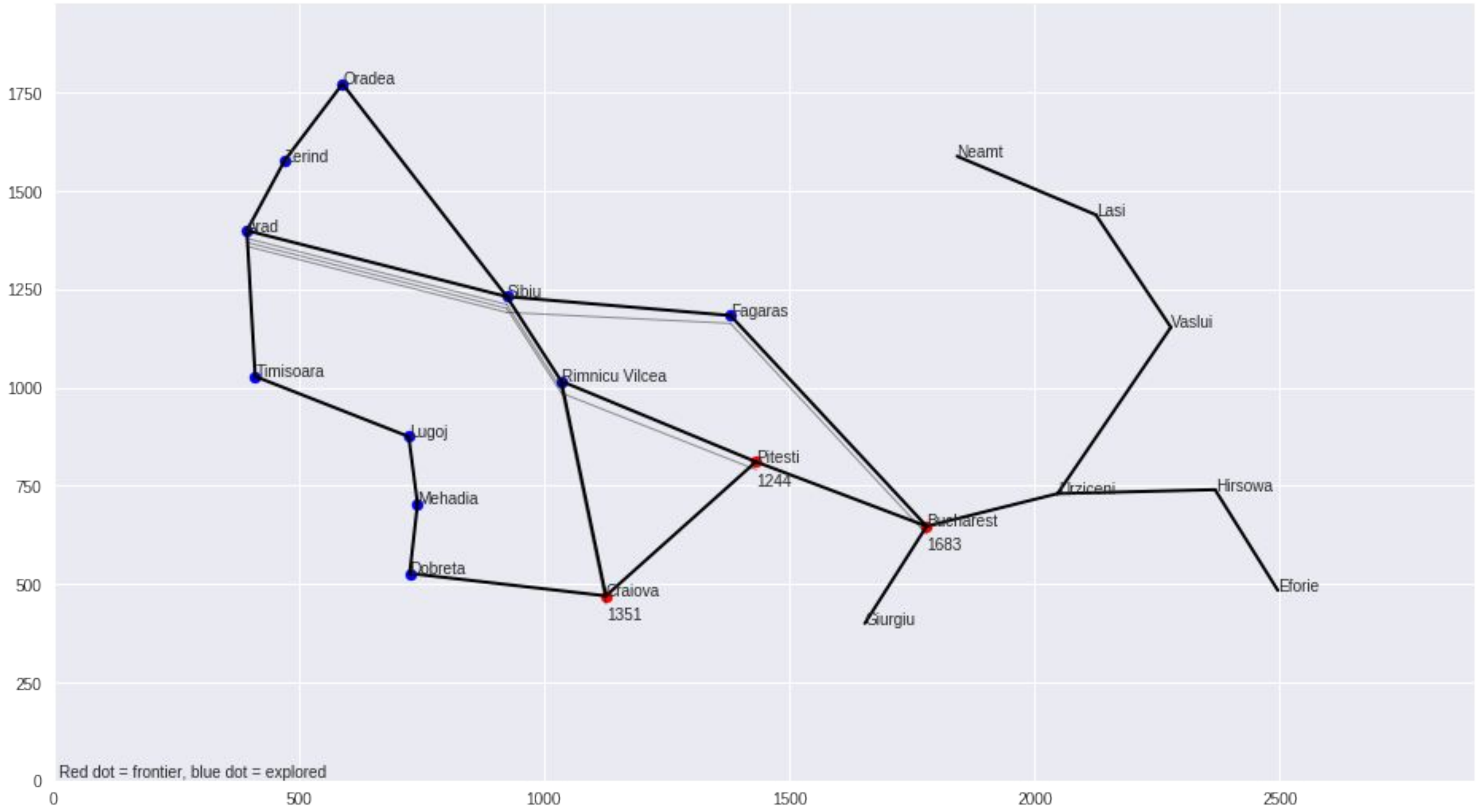
Red dot = frontier, blue dot = explored

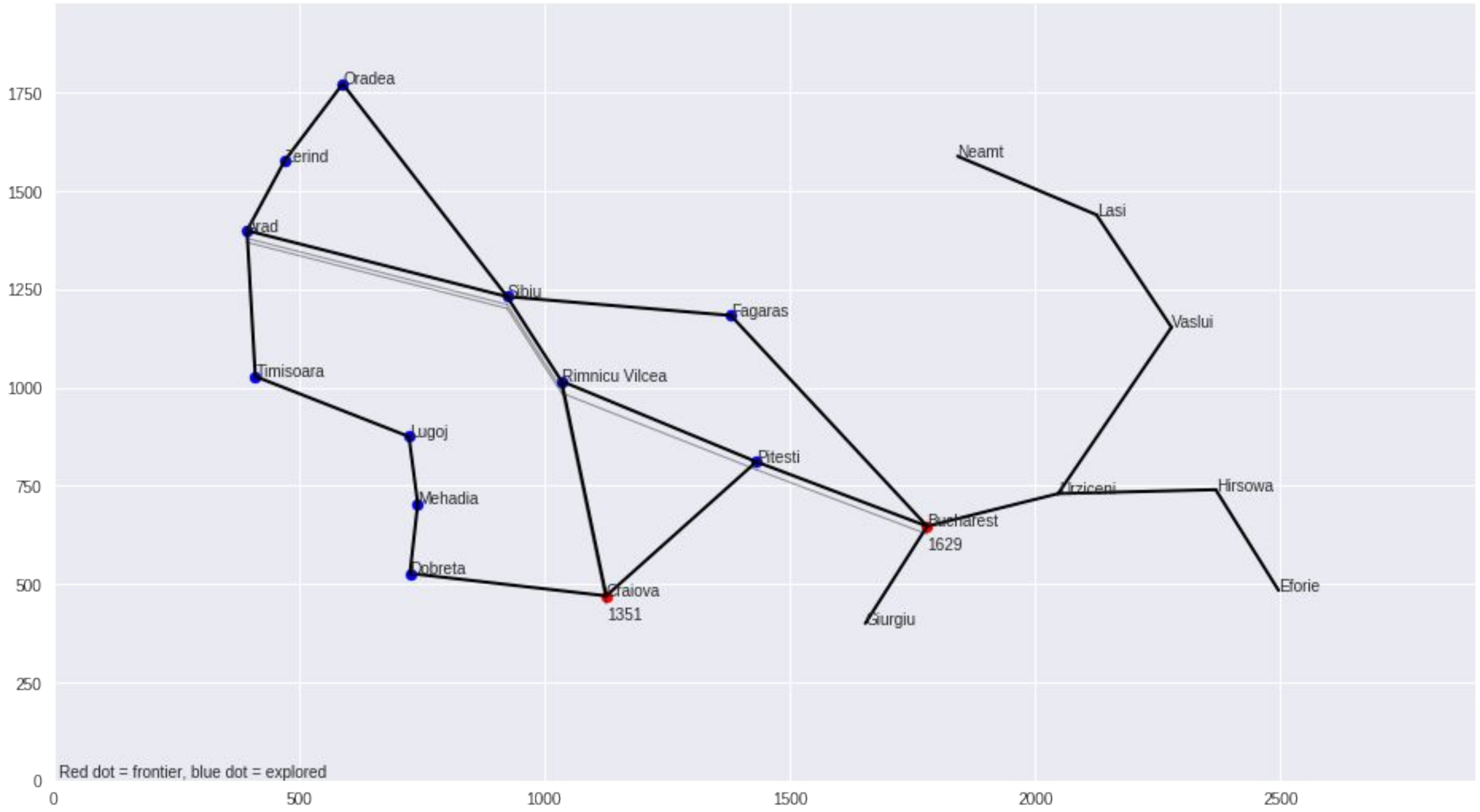

Oradea
Zerind
Arad
Sibiu
Fagaras
Timisoara
Rimnicu Vilcea
Lugoj
Pitesti
1244
Mehadia
Bucharest
1683
Dobreta
1069
Craiova
1351
Giurgiu
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Eforie
Red dot = frontier, blue dot = explored
1750
1500
1250
1000
750
500
250
0
0
500
1000
1500
2000
2500

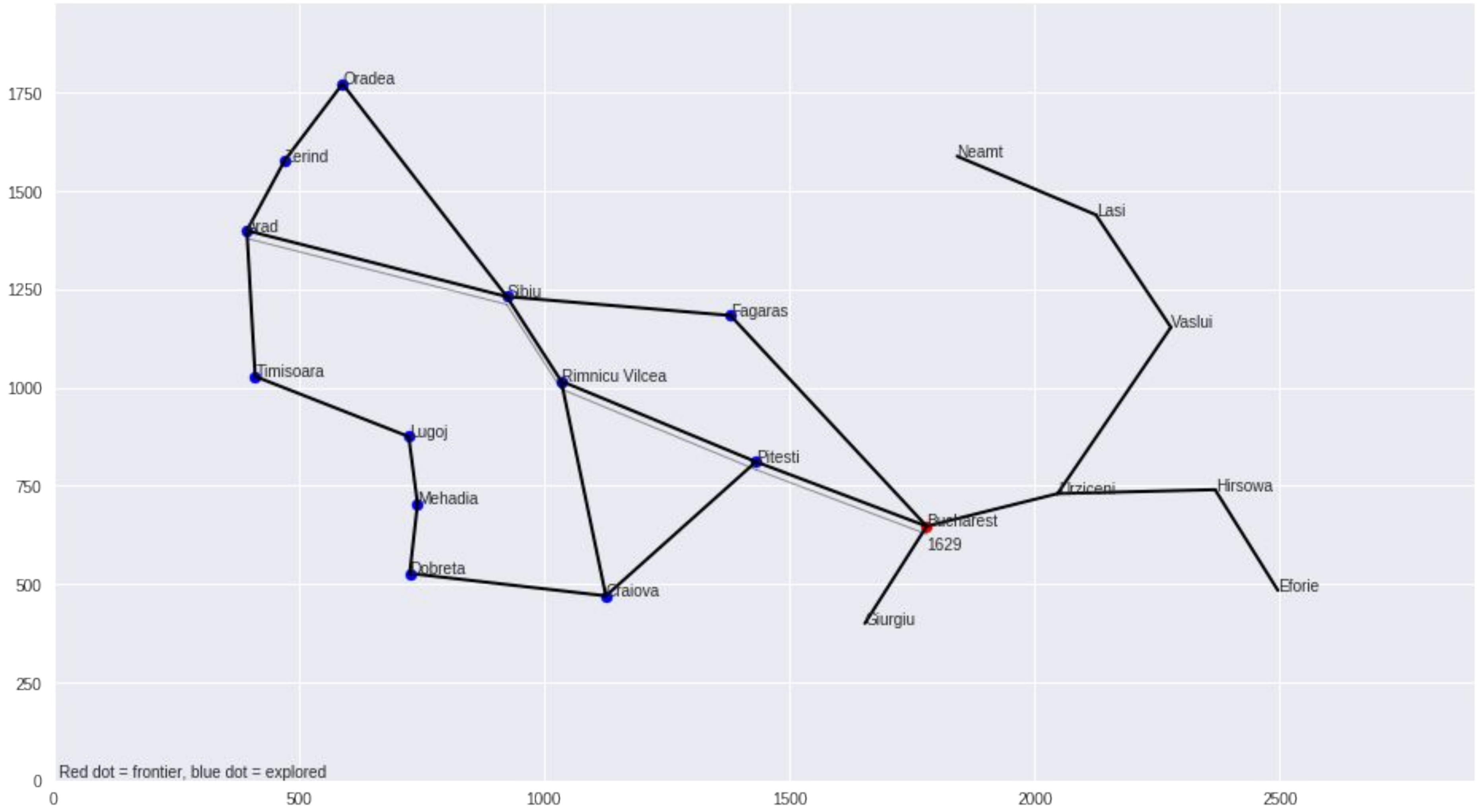
Are we done?



Oradea
Zerind
Arad
Sibiu
Fagaras
Timisoara
Rimnicu Vilcea
Lugoj
Pitesti
1244
Mehadia
Dobreta
1069
Craiova
1351
Bucharest
1683
Giurgiu
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Eforie
Red dot = frontier, blue dot = explored
1750
1500
1250
1000
750
500
250
0
0
500
1000
1500
2000
2500


Oradea
Zerind
Arad
Sibiu
Fagaras
Timisoara
Rimnicu Vilcea
Lugoj
Pitesti
1244
Mehadia
Dobreta
Craiova
1351
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Bucharest
1683
Eforie
Giurgiu
Red dot = frontier, blue dot = explored
1750
1500
1250
1000
750
500
250
0
0
500
1000
1500
2000
2500

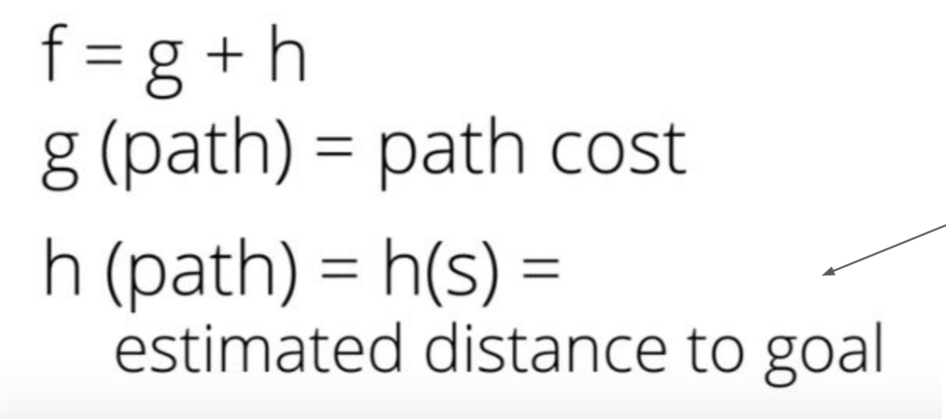

Oradea
Zerind
Arad
Sibiu
Fagaras
Timisoara
Rimnicu Vilcea
Lugoj
Pitesti
Mehadia
Dobreta
Craiova
1351
Bucharest
1629
Giurgiu
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Eforie
Red dot = frontier, blue dot = explored
0
250
500
750
1000
1250
1500
1750
0
500
1000
1500
2000
2500


Oradea
Zerind
Arad
Sibiu
Fagaras
Timisoara
Rimnicu Vilcea
Lugoj
Pitesti
Mehadia
Dobreta
Craiova
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Bucharest
1629
Giurgiu
Eforie
Red dot = frontier, blue dot = explored
0
250
500
750
1000
1250
1500
1750
0
500
1000
1500
2000
2500

Steps

```
frontiers = [start]
explored  = []
if start == goal: return path
while frontiers not empty
    frontier = frontiers with lowest cost
    if frontier == goal: return path to frontier
    remove frontier from frontiers and add location to explored path
    for node connected to frontier there isn't explored
        new cost = frontier cost + cost for going to node
        add node with new cost and previous nodes to frontier if not in list,
                                                            replace it if it is in list and new path i cheaper
```

A*



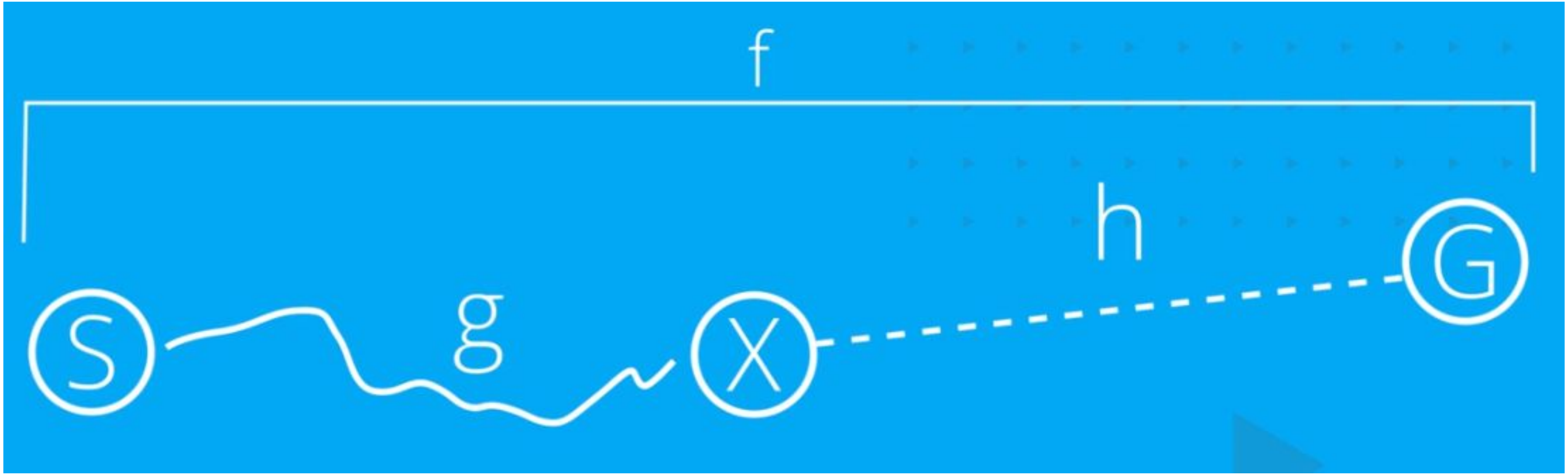
$f = g + h$

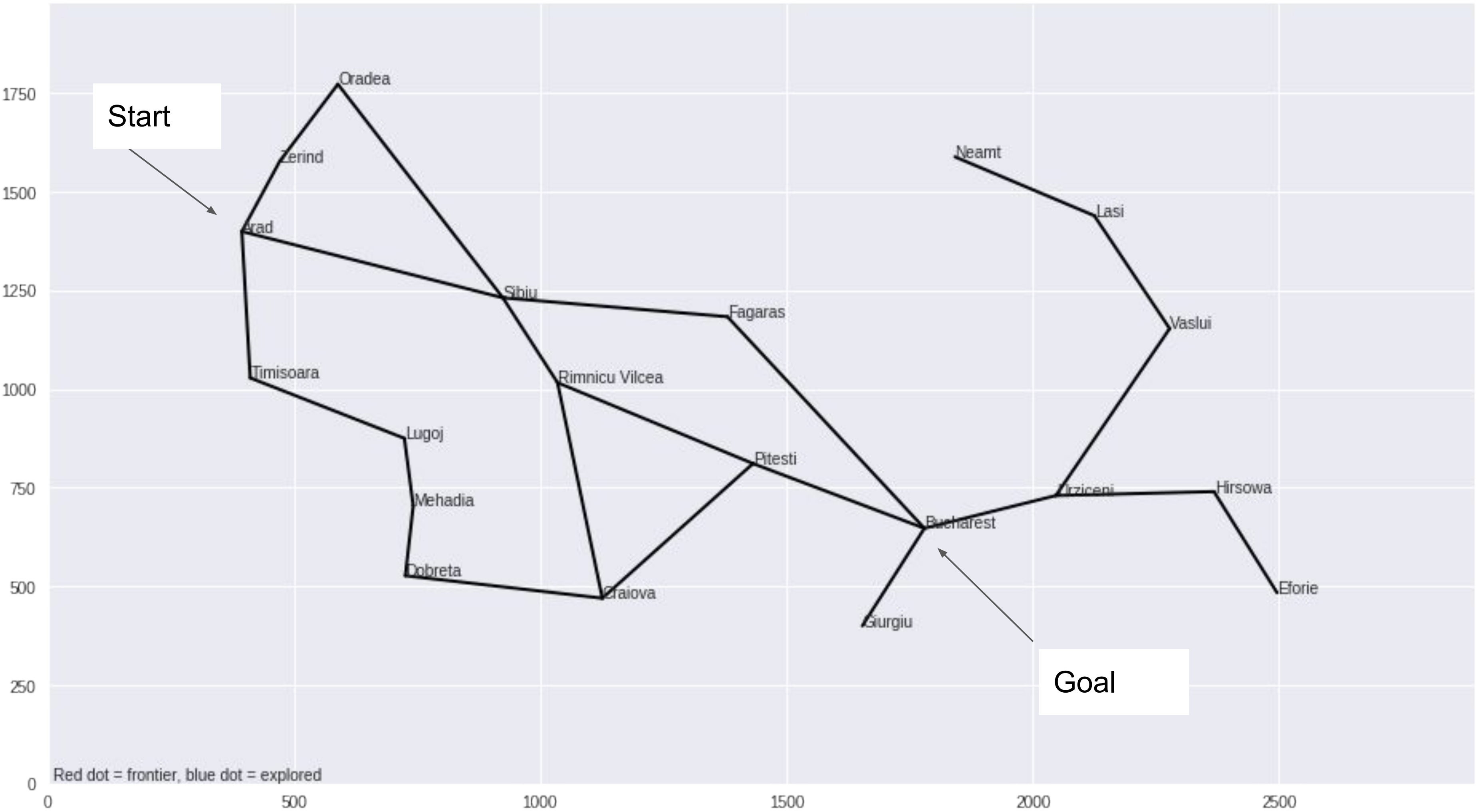
g (path) = path cost

h (path) = $h(s)$ =

estimated distance to goal

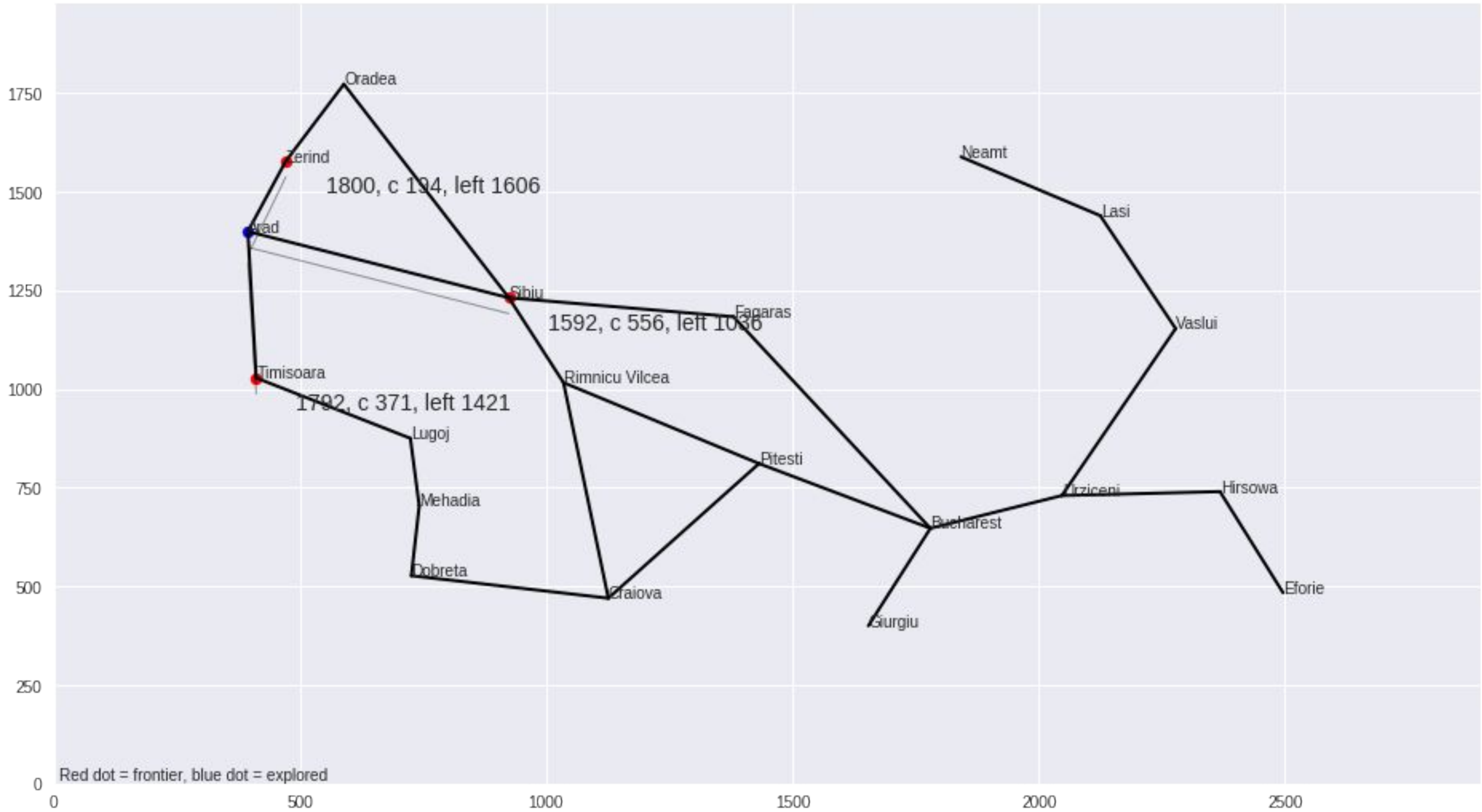
h(s) has to be underestimated to ensure the shortest path is found

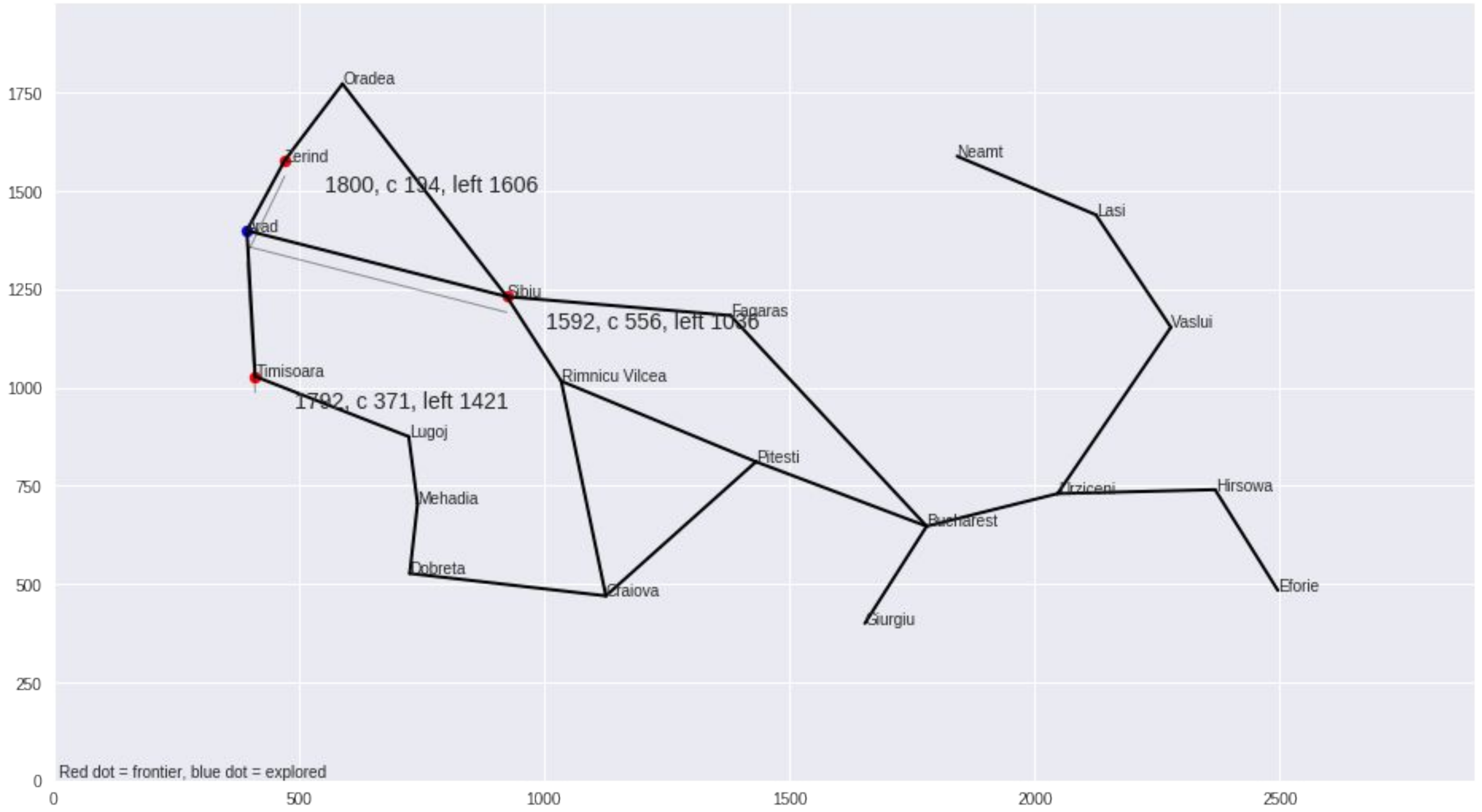


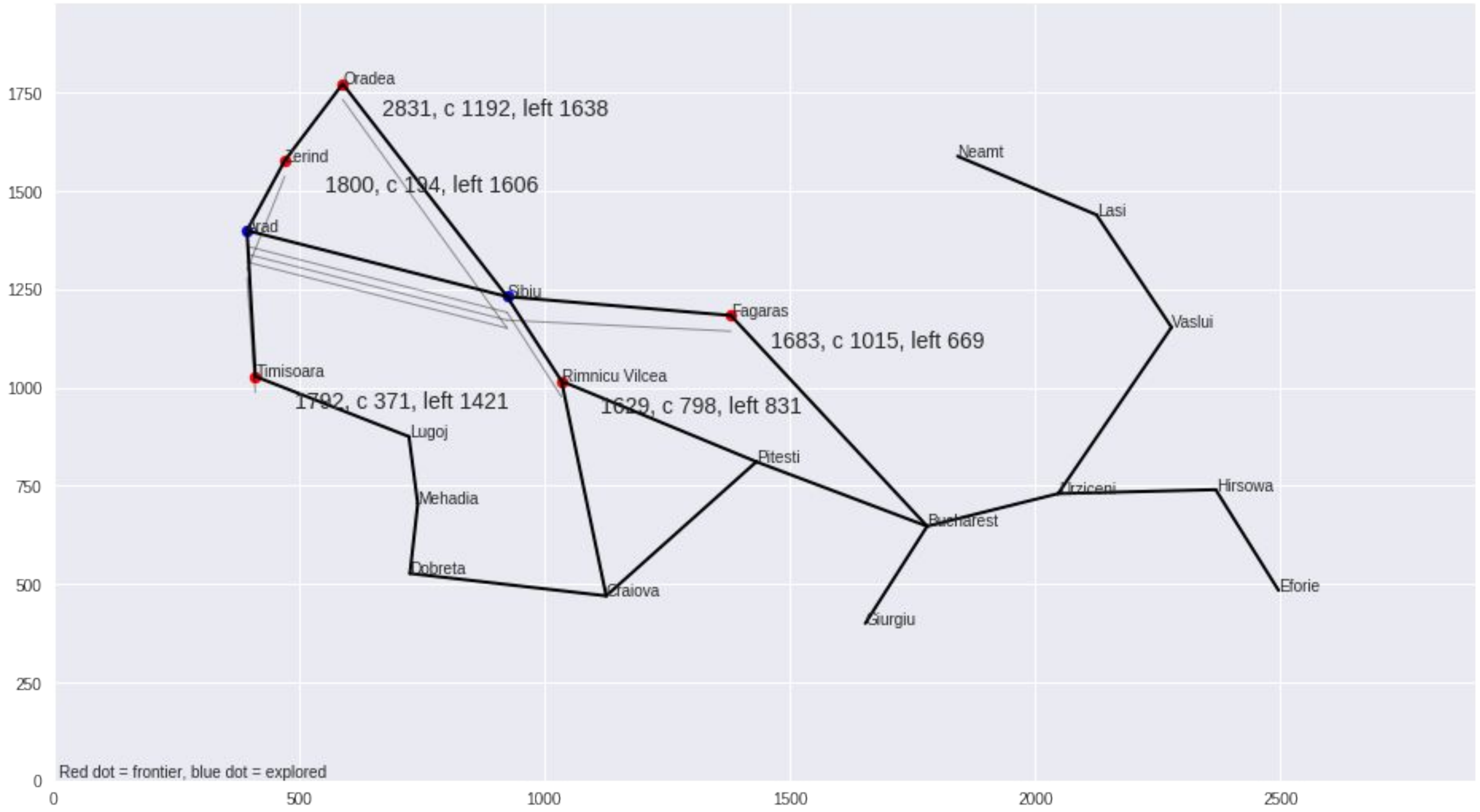

Oradea
Zerind
Arad
Sibiu
Fagaras
Timisoara
Rimnicu Vilcea
Lugoj
Pitesti
Mehadia
Dobreta
Craiova
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Bucharest
Giurgiu
Eforie
Red dot = frontier, blue dot = explored
0
250
500
750
1000
1250
1500
1750
0
500
1000
1500
2000
2500

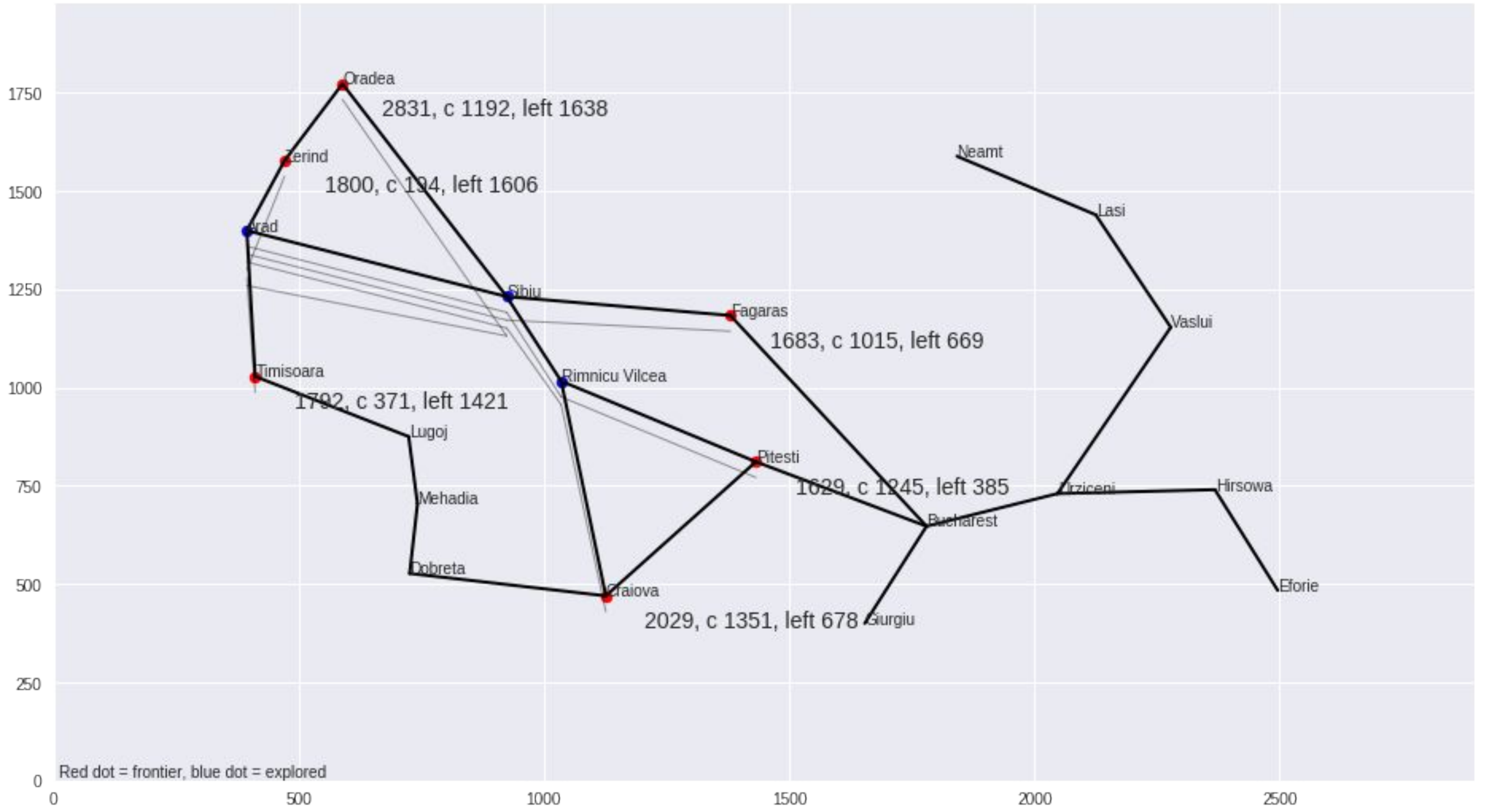
Start

Goal

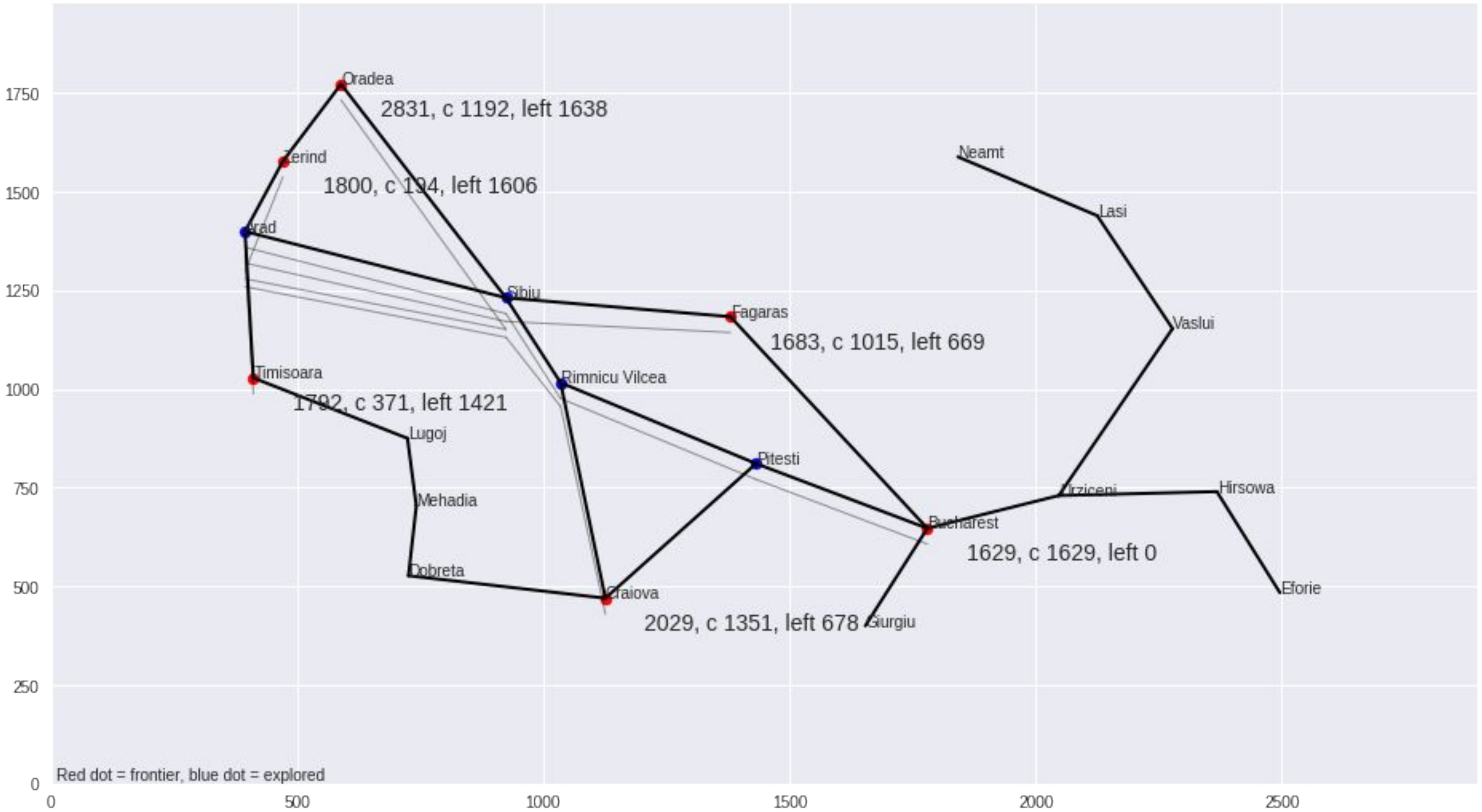

Oradea
Zerind
1800, c 194, left 1606
Neamt
Lasi
Arad
Sibiu
Fagaras
1592, c 556, left 1036
Vaslui
Timisoara
Rimnicu Vilcea
1792, c 371, left 1421
Lugoj
Pitesti
Mehadia
Urziceni
Hirsowa
Bucharest
Dobreta
Craiova
Eforie
Giurgiu
Red dot = frontier, blue dot = explored
0
250
500
750
1000
1250
1500
1750
0
500
1000
1500
2000
2500


Oradea
Zerind
1800, c 194, left 1606
Arad
Sibiu
1592, c 556, left 1036
Fagaras
Timisoara
1792, c 371, left 1421
Rimnicu Vilcea
Lugoj
Mehadia
Pitesti
Dobreta
Craiova
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Bucharest
Eforie
Giurgiu
Red dot = frontier, blue dot = explored
1750
1500
1250
1000
750
500
250
0
0
500
1000
1500
2000
2500

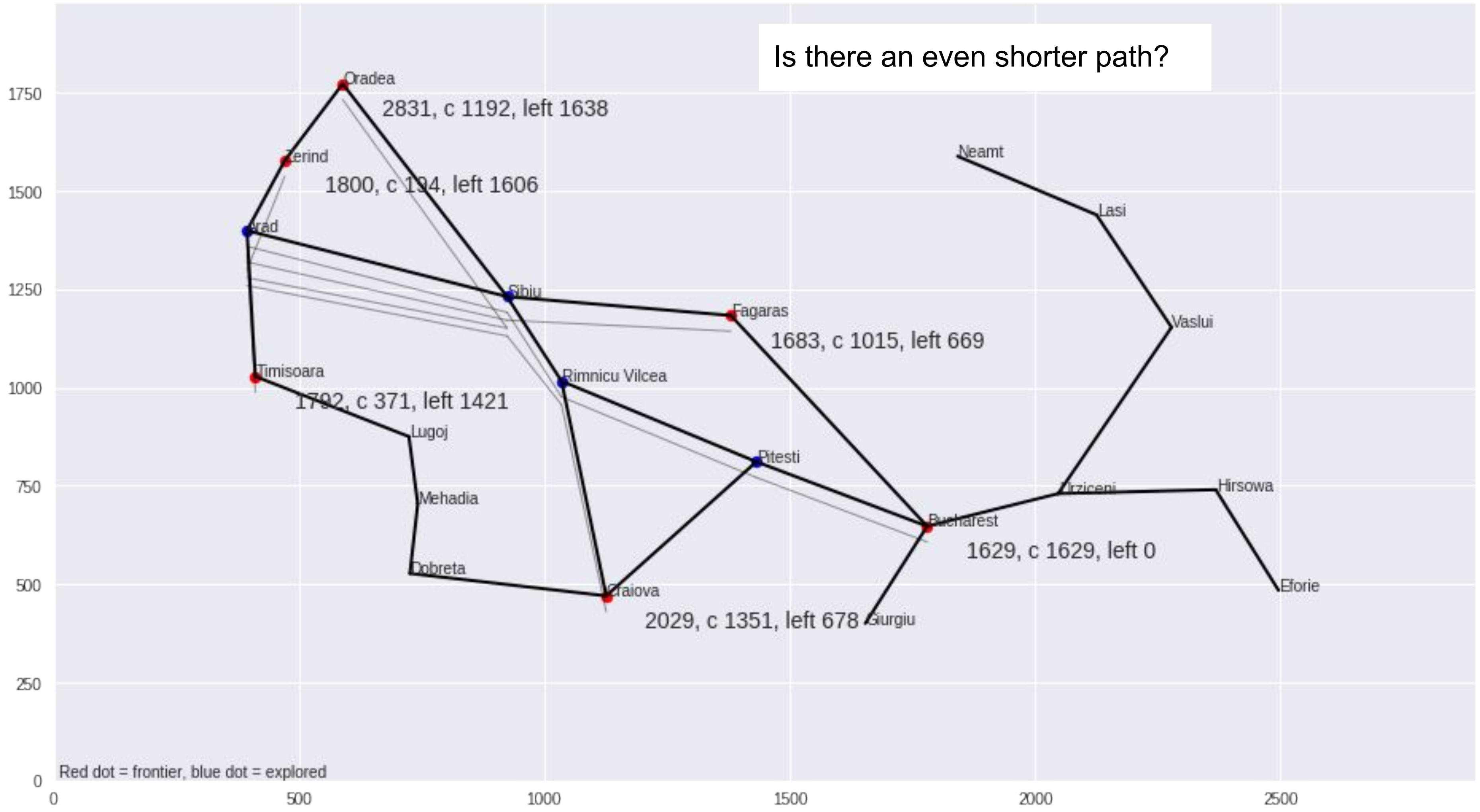

Oradea
2831, c 1192, left 1638
Zerind
1800, c 194, left 1606
Arad
Sibiu
Fagaras
1683, c 1015, left 669
Timisoara
1792, c 371, left 1421
Rimnicu Vilcea
1629, c 798, left 831
Lugoj
Mehadia
Dobreta
Craiova
Pitesti
Bucharest
Giurgiu
Neamt
Lasi
Vaslui
Urziceni
Hirsowa
Eforie
Red dot = frontier, blue dot = explored
1750
1500
1250
1000
750
500
250
0
0
500
1000
1500
2000
2500



Red dot = frontier, blue dot = explored


Oradea
2831, c 1192, left 1638
Zerind
1800, c 194, left 1606
Arad
Sibiu
Fagaras
1683, c 1015, left 669
Neamt
Lasi
Vaslui
Timisoara
1792, c 371, left 1421
Rimnicu Vilcea
Lugoj
Pitesti
Mehadia
Urziceni
Hirsowa
Bucharest
1629, c 1629, left 0
Dobreta
Craiova
2029, c 1351, left 678
Giurgiu
Eforie
Red dot = frontier, blue dot = explored
1750
1500
1250
1000
750
500
250
0
0
500
1000
1500
2000
2500

Is there an even shorter path?



Steps

```
frontiers = [start]
explored  = []
if start == goal: return path
while frontiers not empty
    frontier = frontiers with lowest F cost
    if frontier == goal: return path to frontier
    remove frontier from frontiers and add location to explored path
    for node connected to frontier there isn't explored
        new G cost = frontier G cost + cost for going to node
        new H cost = h(node)
        new F cost = G + H
        add node with G and F and previous nodes to frontier if not in list,
                                        replace it if it is in list and new paths F cost is cheaper
```

Links

A* with array of ancestors and PQ: https://bit.ly/2YDbL2Q

A* (unoptimized), uniform cost & breadth first: https://github.com/benjaco/search-algorithms