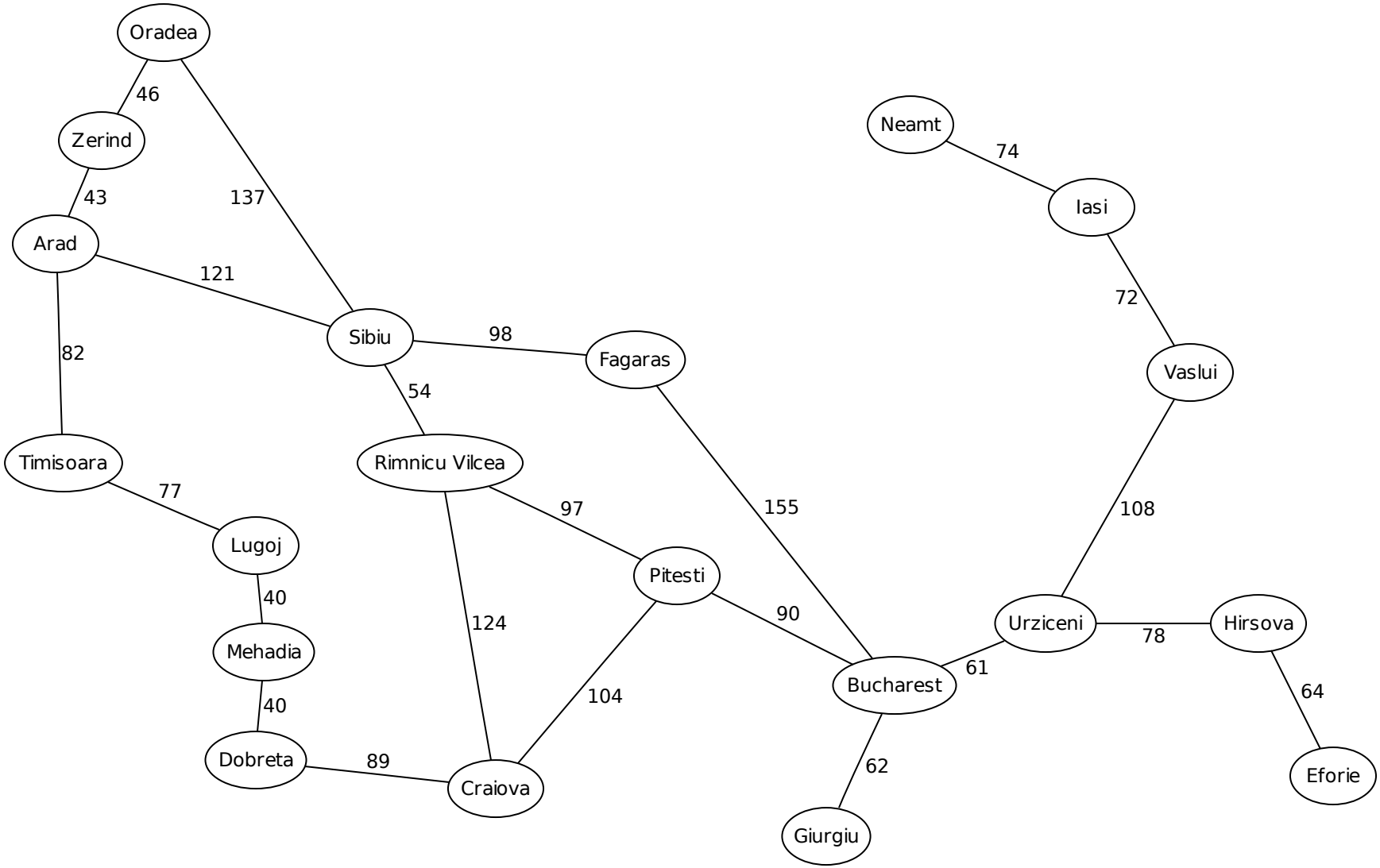
Romania

Fall Semester

Topic Outline

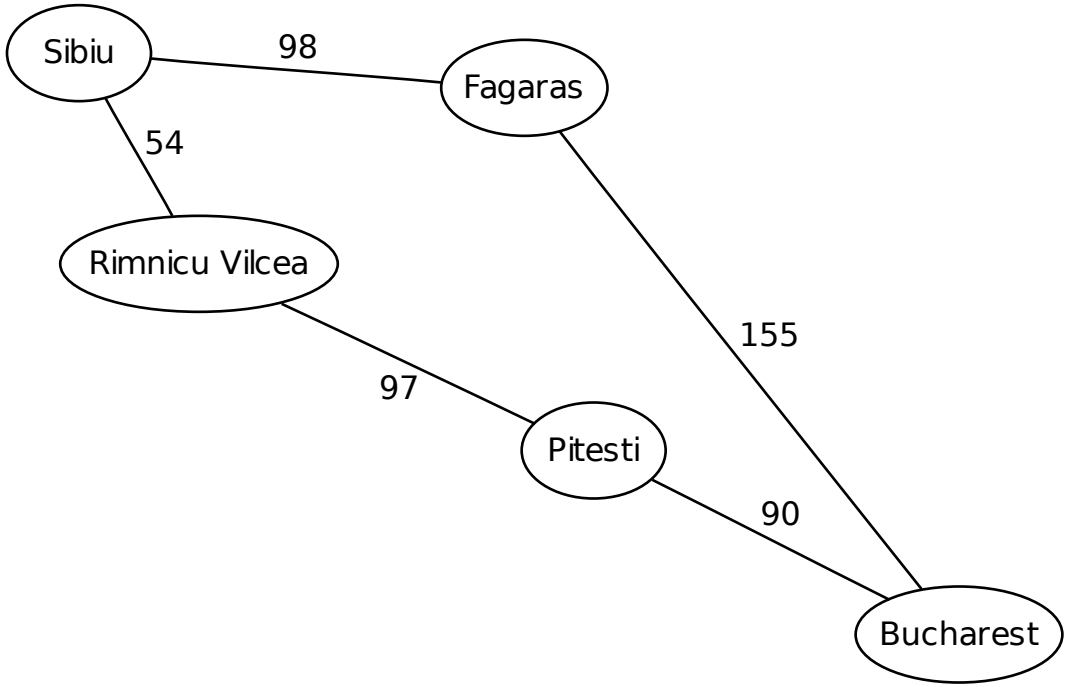
- Road Map of Romania
- Statement of the Problem
- Examples
  - Optimal path
  - Breadth-first
  - Depth-first alphabetical
- Uniform Cost Search
  - Compare to BFS
  - Priority queue
  - Memory issues
- Test cases
- A-Star Search
  - Heuristics
  - Admissibility
  - Efficiency
- Comparisons
  - Time
  - Memory
  - Dijkstra
- Up Next: Sliding Tile


Oradea
46
Zerind
43
137
Arad
121
Sibiu
98
Fagaras
82
54
Timisoara
Rimnicu Vilcea
77
97
155
Lugoj
Pitesti
40
124
90
Mehadia
40
104
61
Bucharest
Dobreta
89
Craiova
62
Giurgiu
Neamt
74
Iasi
72
Vaslui
108
Urziceni
78
Hirsova
64
Eforie

Statement of the Problem

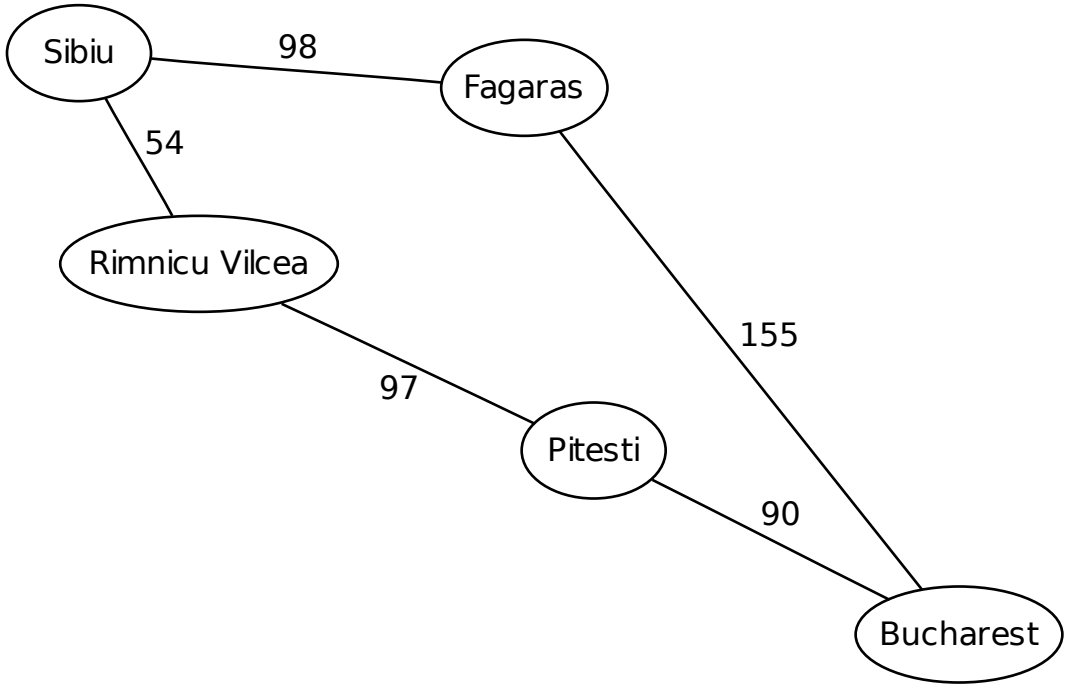
- Input. A road map, the starting city and the ending city.
- Output. A sequence of cities such that:
  - The first city is the starting city.
  - The last city is the ending city.
  - All the cities are unique.
  - Consecutive cities are connected by a road.
  - The total distance travelled is as small as possible.
- If no such sequence exists then output NO MATCH. (For the Romania problem this will not happen but Google used to have an amusing workaround for the trip from New York to London.)

Example: FROM Sibiu TO Bucharest (Optimal path)



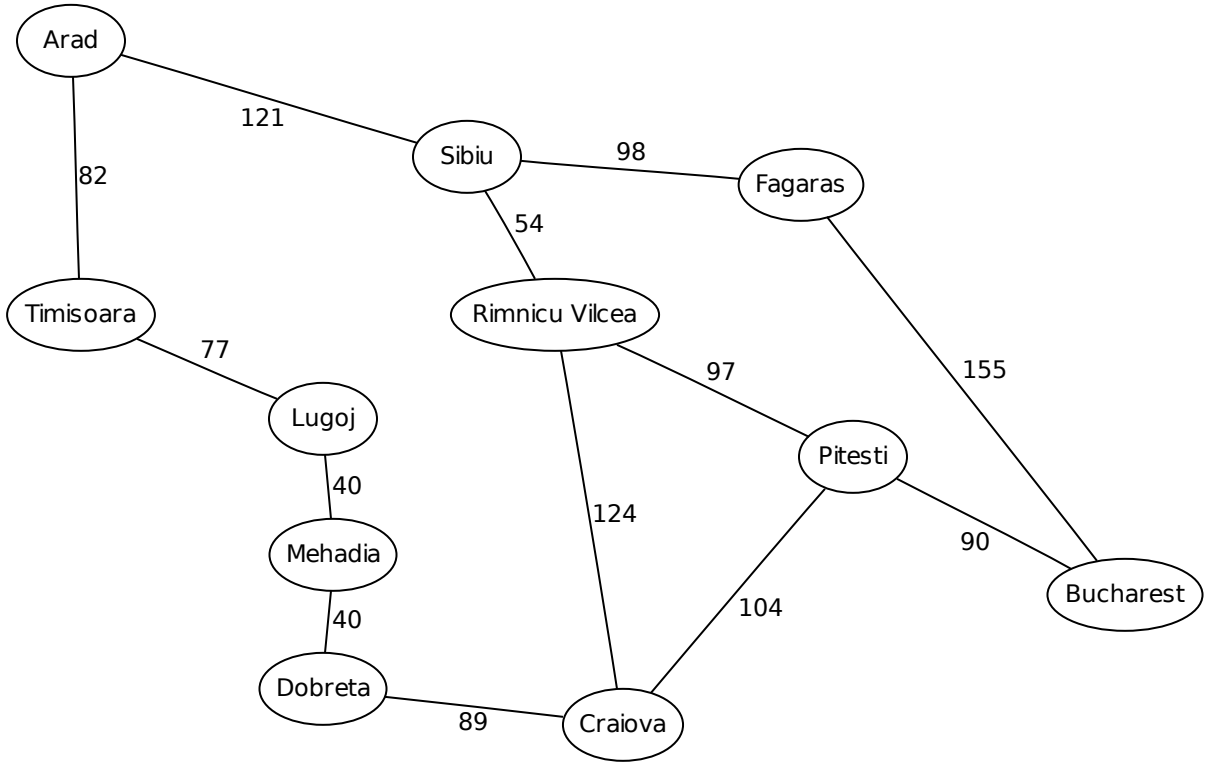
- PATH: 241, Sibiu, Rimnicu Vilcea, Pitesti, Bucharest
- Note that the distance has been rounded at each step.

Example: FROM Sibiu TO Bucharest (Breadth-first)



- PATH: 253, Sibiu, Fagaras, Bucharest
- Note that this is not the shortest path.

Example: FROM Sibiu TO Bucharest (Depth-first alphabetical)



- PATH: 643, Sibiu, Arad, Timsoara ...Craiova, Pitesti, Bucharest

Uniform Cost Search (1)

- Replace the FIFO queue from BFS with a priority queue.
- Order the paths in this queue by total distance travelled.
- Python trick.
  - Store distance as first element of heterogeneous list.
  - For example, `queue=[[0,'Sibiu']]`.
- Maintain order using:
  - The `sort` command.
  - A linear or binary search, then `insert` or `append`.
  - The `heapq` module.

Uniform Cost Search (2)

- Still no recursion.
- Finds the optimal path on a graph with positive weights.
- Proof. We consider less expensive paths before more expensive paths. Thus, when first we find a solution, if there had been a less expensive solution then we would have already found it!
- Note carefully, this is only true if we check for the target when paths come out of the queue and not when they go in.
- Queue of paths may still require an excessive amount of memory.
- See also, Russell and Norvig page 75.

Test Cases

- FROM Arad TO Bucharest
  - The classic.
- FROM Fagaras TO Neamt
  - So close, yet so far away.
- FROM Giurgiu TO Zerind
  - Looks interesting.
- FROM Mehadia TO Sibiu
  - A real horse race.

Lab Assignment: Uniform Cost

- First modify DFS and BFS to work for Romania.
  - For comparison, calculate the total cost of the path found.
- Then implement the Uniform Cost Search.
- For testing purposes print a trace showing:
  - The sequence of paths considered.
  - The entire priority queue (unless it's a heap) at each step.
- Let the user choose the cities.
- Output the solution found by each search algorithm and its total path cost. Consider various neighbor orderings for DFS.

A-Star Search (1)

- Heuristic. An estimate on the remaining cost to reach the goal.
- Admissibile heuristic. An underestimate on the remaining cost.
  - For example, the straight-line distance between the ending city and any other city in Romania.
- Order the paths in the priority queue by the sum of the total cost so far *plus* this estimate on the remaining cost.
- Efficiency. For a given heuristic the A-star search is the most efficient algorithm for finding an optimal path.
- See also, Russell and Norvig page 97.

A-Star Search (2)

- Good path. A path that doesn't yet lead to the target but will eventually and with optimal path cost.
- Bad path. A path that already leads to the target but does so with suboptimal path cost.
- We sort by $f = g + h$ and call the optimal path cost C, so:
  - If h is an underestimate then $f_{good} < C$.
  - By definition $C < f_{bad} = g + 0$, because it's bad.
  - Since $f_{good} < C < f_{bad}$, the good stays in front of the bad.
  - If h were possibly an overestimate instead then this might not be true and we could get stuck with the bad path.

Example: FROM Sibiu TO Bucharest (Uniform Cost)

```
  0, Sibiu
 54, Sibiu, Rimnicu Vilcea
 98, Sibiu, Fagaras
121, Sibiu, Arad
137, Sibiu, Oradea
151, Sibiu, Rimnicu Vilcea, Pitesti
164, Sibiu, Arad, Zerind
178, Sibiu, Rimnicu Vilcea, Craiova
203, Sibiu, Arad, Timisoara
241, Sibiu, Rimnicu Vilcea, Pitesti, Bucharest
```

Example: FROM Sibiu TO Bucharest (A-Star)

```
233 =   0 + 233, Sibiu
240 =  54 + 186, Sibiu, RV
241 = 151 +  90, Sibiu, RV, Pitesti
241 = 241 +   0, Sibiu, RV, Pitesti, Bucharest
```

- Note. This is an extreme example where no alternative paths need to be considered. This is not typical. In general, for other problems and heuristics, do not expect the algorithm to work so well. This is the best-case efficiency possible.

Comparisons (1)

- Statistics for BFS assuming branching factor is 10, node size is 1000 bytes, able to process 10,000 nodes per second[1]:

Depth	**Nodes**	**Time**	**Memory**
2	1100	0.11 seconds	1 megabyte
4	111,100	11 seconds	106 megabytes
6	10^7	19 minutes	10 gigabytes
8	10^9	31 hours	1 terabyte
10	10^{11}	129 days	101 terabytes
12	10^{13}	35 years	10 petabytes
14	10^{15}	3,523 years	1 exabyte

- Prefix, mega is million (6), giga is billion (9), tera is trillion (12), peta is quadrillion (15), exa is quintillion(18).
- Uniform Cost and A-Star will do better, but they still blow up.

[1]Russell, Stuart, and Peter Norvig. Artificial Intelligence: A Modern Approach. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2003. 74.

Comparisons (2)

- Dijkstra. Initialize distances to zero or infinity. Find best distance not yet seen. Dynamic programming, update neighbors if this distance *plus* edge cost is better. Stop when target seen.
- Some other famous algorithms:
  - Bellman-Ford. Shortest path with negative weights, as in reduction of the job-scheduling-with-deadlines problem.
  - Ford-Fulkerson. Max flow in a network from source to sink.
  - Prim. Minimal spanning tree (MST), one tree grows bigger.
  - Kruskall. MST, a forest of many trees connect together.

Lab Assignment: A-Star Search

- Still let the user choose the cities.
- Output the solution found by each search algorithm and its total path cost.
  - A-Star with straight-line distance heuristic.
  - Uniform Cost.
  - DFS with greedy best first ordering.
  - BFS.
  - Additional algorithms are also encouraged!
- Note that the A-Star should find the same solution as Uniform Cost Search only faster (i.e., there's only one optimal path).