UIT2504 Artificial Intelligence

Heuristics-Based Search Strategies

C. Aravindan
<AravindanC@ssn.edu.in>

Professor of Computer Science
SSN College of Engineering

August 10, 2024

Searching for a solution

- Start with initial state in the working set
- Iterate:
  - Return failure if the working set is empty
  - Choose and remove a state x from the working set
  - If it is a goal state, return solution
  - Else, expand x and add the successor states $S(x)$ to the working set

Search Strategies

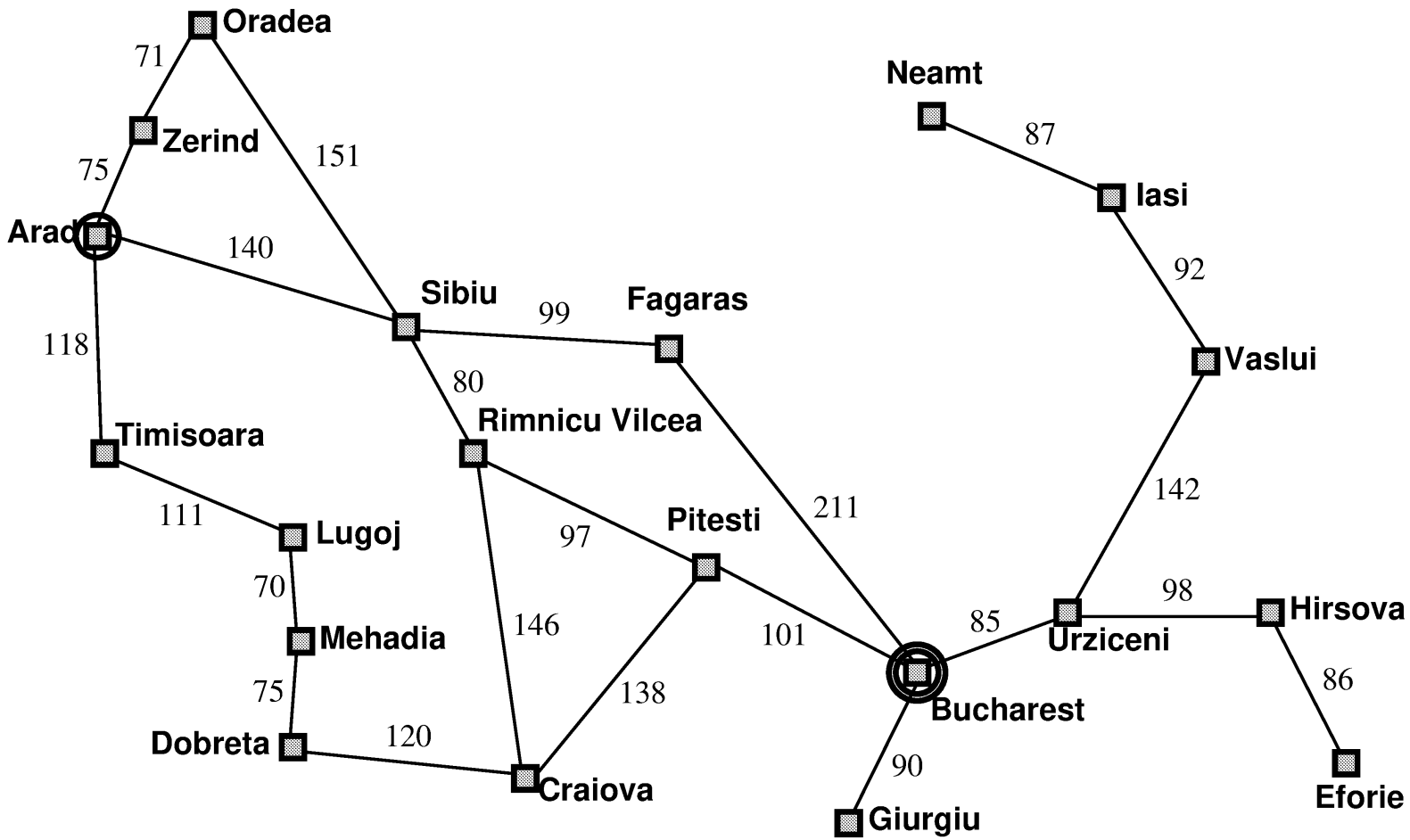
- Uninformed:
  - Breadth-First
  - Depth-First
  - Iterative Deepening
  - Bi-Directional Search
- Informed (Heuristics):
  - Best-first Greedy
  - A^*
  - Local Search Strategies
- Constraint Satisfaction

Performance Measures

- Completeness
- Time Complexity
- Space Complexity
- Optimality

Exercise

- Practice all the basic search strategies to find a route from Arad to Bucharest in the following state graph



Questions?

Evaluating a state

- Sometimes, it may be possible to design an evaluation function $f(s)$ that evaluates the "badness" (to be minimized) or "goodness" (to be maximized) of a state s

Evaluating a state

- Sometimes, it may be possible to design an evaluation function $f(s)$ that evaluates the "badness" (to be minimized) or "goodness" (to be maximized) of a state s
- In such cases, the most desirable state may be chosen from the working set

Evaluating a state

- Sometimes, it may be possible to design an evaluation function $f(s)$ that evaluates the "badness" (to be minimized) or "goodness" (to be maximized) of a state s
- In such cases, the most desirable state may be chosen from the working set
- Working set is maintained as a priority queue based on the evaluation function f

Evaluating a state

- Sometimes, it may be possible to design an evaluation function $f(s)$ that evaluates the "badness" (to be minimized) or "goodness" (to be maximized) of a state s
- In such cases, the most desirable state may be chosen from the working set
- Working set is maintained as a priority queue based on the evaluation function f
- Obviously, the quality of search depends on the evaluation function f

Heuristics

- Usually, such an evaluation function $f(s)$ is designed based on some heuristics $h(s)$ — estimation of cost of reaching a goal state from state s

Heuristics

- Usually, such an evaluation function $f(s)$ is designed based on some heuristics $h(s)$ — estimation of cost of reaching a goal state from state s
- For example, can you think of a heuristics for the route finding problem in a map?

Heuristics

- Usually, such an evaluation function $f(s)$ is designed based on some heuristics $h(s)$ — estimation of cost of reaching a goal state from state s
- For example, can you think of a heuristics for the route finding problem in a map? — Straight line distance (SLD) from the current city to the destination city

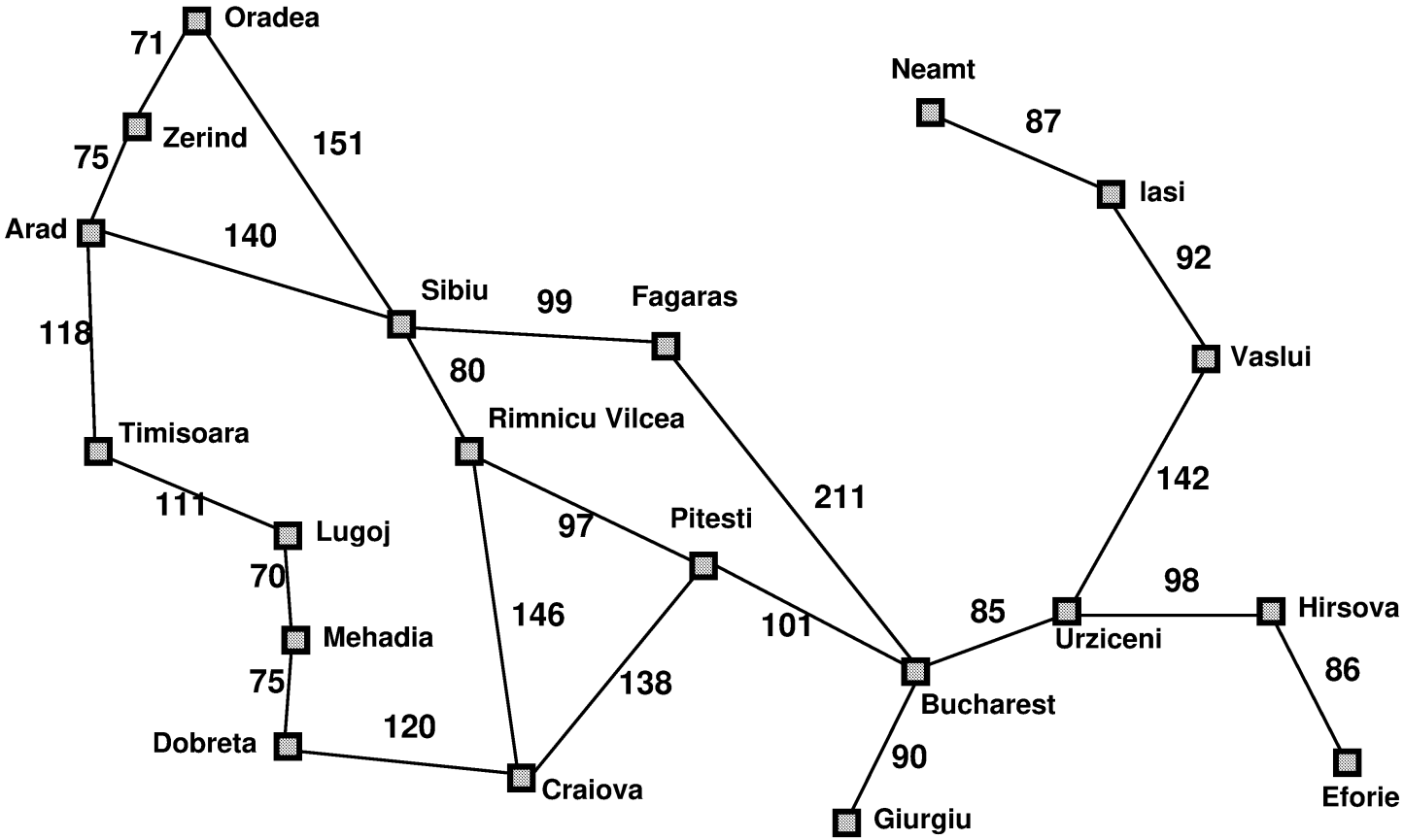
Heuristics

- Usually, such an evaluation function $f(s)$ is designed based on some heuristics $h(s)$ — estimation of cost of reaching a goal state from state s
- For example, can you think of a heuristics for the route finding problem in a map? — Straight line distance (SLD) from the current city to the destination city
- Heuristics should be an easy function to compute!

Heuristics

- Usually, such an evaluation function $f(s)$ is designed based on some heuristics $h(s)$ — estimation of cost of reaching a goal state from state s
- For example, can you think of a heuristics for the route finding problem in a map? — Straight line distance (SLD) from the current city to the destination city
- Heuristics should be an easy function to compute!
- $h(s^*)$ should be 0 for any goal state s^*

Example: Route finding problem



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Find route from Arad to Bucharest

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

- What may be a good heuristics for this state space?

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

- What may be a good heuristics for this state space?
- $h_1(s)$: Number of misplaced tiles

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

- What may be a good heuristics for this state space?
- $h_1(s)$: Number of misplaced tiles — for the above state $h_1(s) = 7$

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

- What may be a good heuristics for this state space?
- $h_1(s)$: Number of misplaced tiles — for the above state $h_1(s) = 7$
- $h_2(s)$: Sum of Manhattan distances of tiles from their goal positions

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

- What may be a good heuristics for this state space?
- $h_1(s)$: Number of misplaced tiles — for the above state $h_1(s) = 7$
- $h_2(s)$: Sum of Manhattan distances of tiles from their goal positions — for the above state $h_2(s) = 2 + 0 + 2 + 2 + 1 + 1 + 4 + 1 = 13$

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

- What may be a good heuristics for this state space?
- $h_1(s)$: Number of misplaced tiles — for the above state $h_1(s) = 7$
- $h_2(s)$: Sum of Manhattan distances of tiles from their goal positions — for the above state $h_2(s) = 2 + 0 + 2 + 2 + 1 + 1 + 4 + 1 = 13$
- Which heuristics is better?

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

- What may be a good heuristics for this state space?
- $h_1(s)$: Number of misplaced tiles — for the above state $h_1(s) = 7$
- $h_2(s)$: Sum of Manhattan distances of tiles from their goal positions — for the above state $h_2(s) = 2 + 0 + 2 + 2 + 1 + 1 + 4 + 1 = 13$
- Which heuristics is better? — an estimate which is closer to the actual is always better!

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

- What may be a good heuristics for this state space?
- $h_1(s)$: Number of misplaced tiles — for the above state $h_1(s) = 7$
- $h_2(s)$: Sum of Manhattan distances of tiles from their goal positions — for the above state $h_2(s) = 2 + 0 + 2 + 2 + 1 + 1 + 4 + 1 = 13$
- Which heuristics is better? — an estimate which is closer to the actual is always better!
- We say that h_2 dominates h_1

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

- What may be a good heuristics for this state space?
- $h_1(s)$: Number of misplaced tiles — for the above state $h_1(s) = 7$
- $h_2(s)$: Sum of Manhattan distances of tiles from their goal positions — for the above state $h_2(s) = 2 + 0 + 2 + 2 + 1 + 1 + 4 + 1 = 13$
- Which heuristics is better? — an estimate which is closer to the actual is always better!
- We say that h_2 dominates h_1
- An admissible heuristics is one which does not overestimate

Example: Sliding puzzle

- Consider the sliding puzzle, such as

3	2	7
5	8	
1	4	6

- What may be a good heuristics for this state space?
- $h_1(s)$: Number of misplaced tiles — for the above state $h_1(s) = 7$
- $h_2(s)$: Sum of Manhattan distances of tiles from their goal positions — for the above state $h_2(s) = 2 + 0 + 2 + 2 + 1 + 1 + 4 + 1 = 13$
- Which heuristics is better? — an estimate which is closer to the actual is always better!
- We say that h_2 dominates h_1
- An admissible heuristics is one which does not overestimate — in our example, both $h_1(x)$ and $h_2(x)$ are admissible

Example: n-queens problem

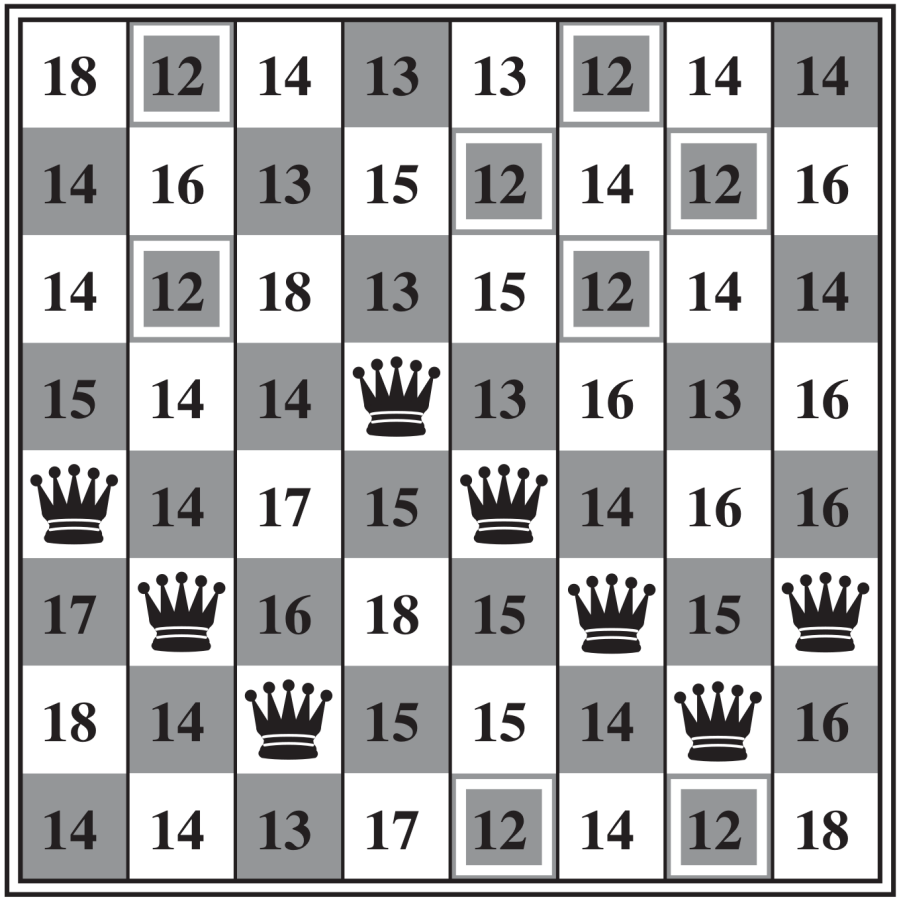
- What may a good heuristics for n-queens problem?

Example: n-queens problem

- What may a good heuristics for n-queens problem?
- Cost estimate: Number of pairs of queens that are attacking each other, either directly or indirectly

Example: *n*-queens problem

- What may a good heuristics for *n*-queens problem?
- Cost estimate: Number of pairs of queens that are attacking each other, either directly or indirectly



Questions?

Best-first greedy search

- As a simple strategy, we may let the evaluation function f to be the same as the heuristics function h

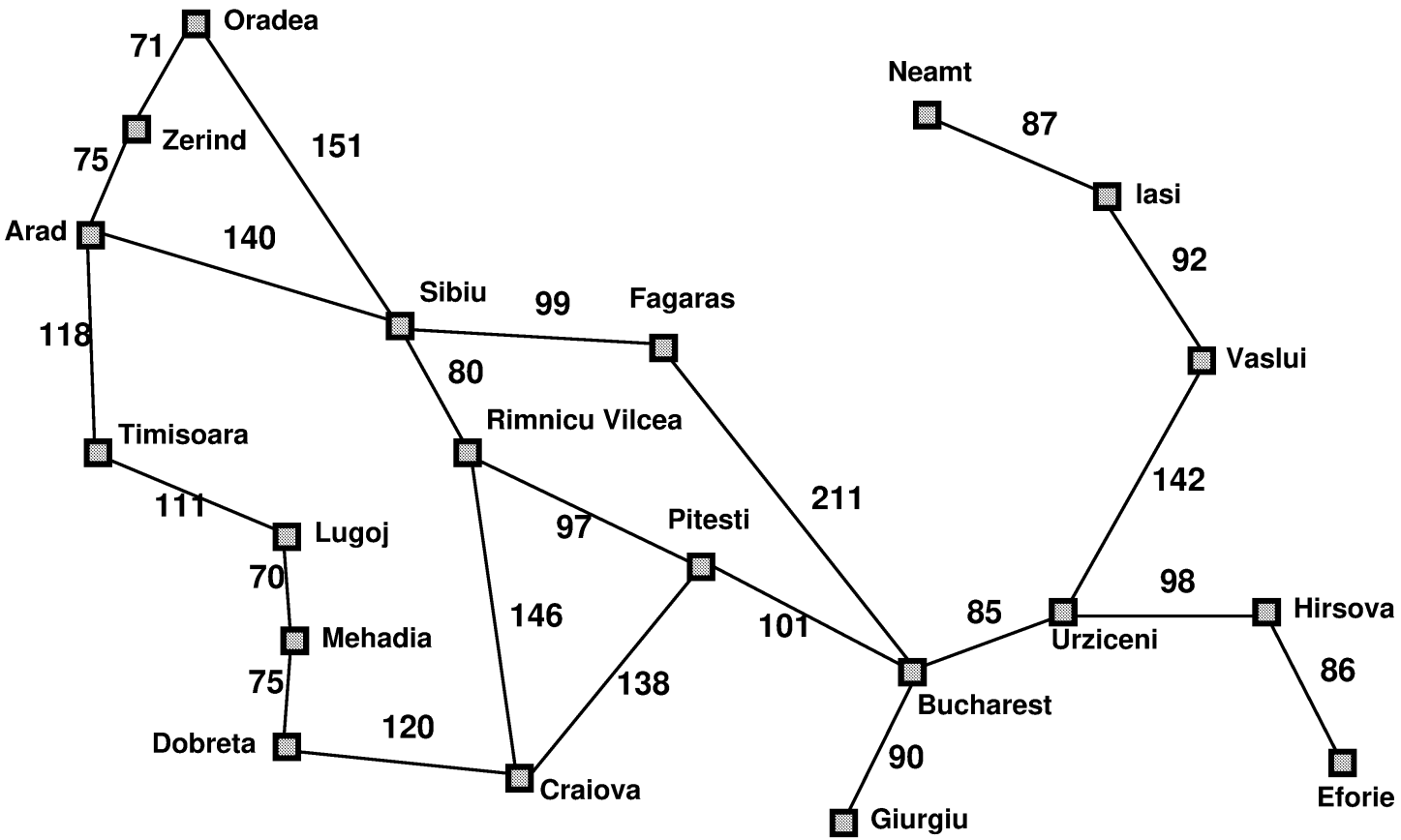
Best-first greedy search

- As a simple strategy, we may let the evaluation function f to be the same as the heuristics function h
- Nodes in the working set (priority queue) are organized based on estimated cost and the one with the least cost is given preference

Best-first greedy search

- As a simple strategy, we may let the evaluation function f to be the same as the heuristics function h
- Nodes in the working set (priority queue) are organized based on estimated cost and the one with the least cost is given preference
- This is a generalization of greedy design strategy, that you have learnt in the previous semester

Best-first greedy search

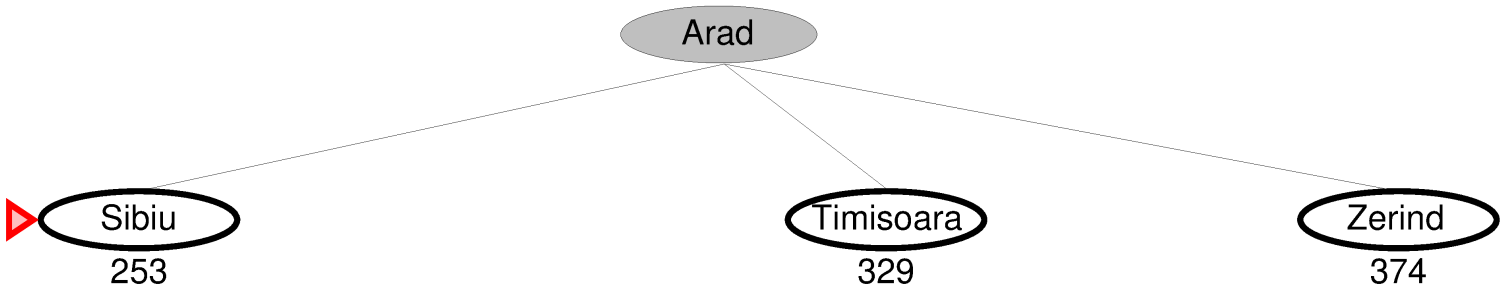


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

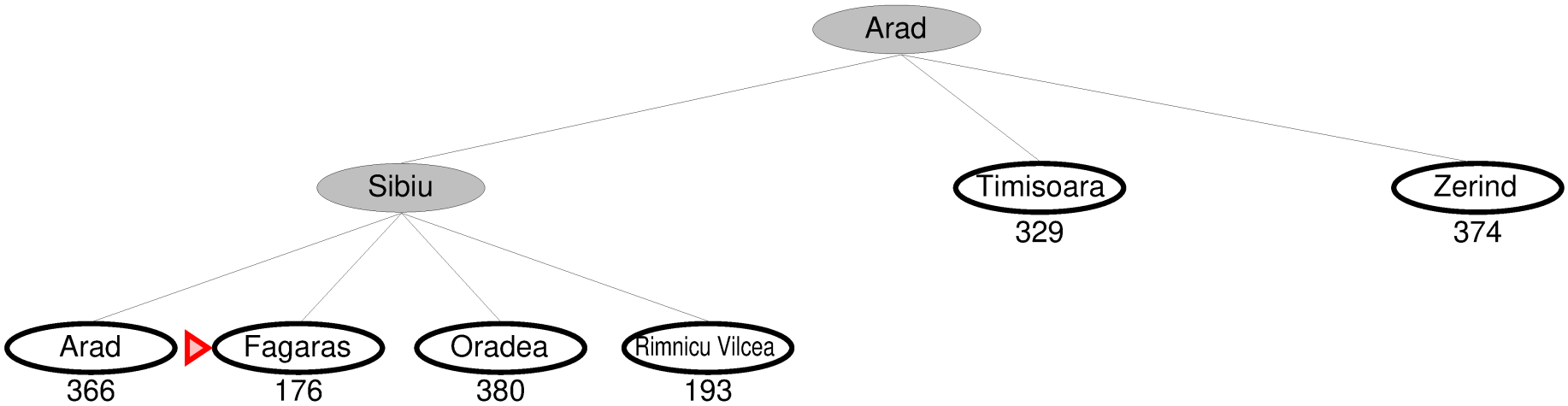
Find the best route from Arad to Bucharest

Best-first greedy search

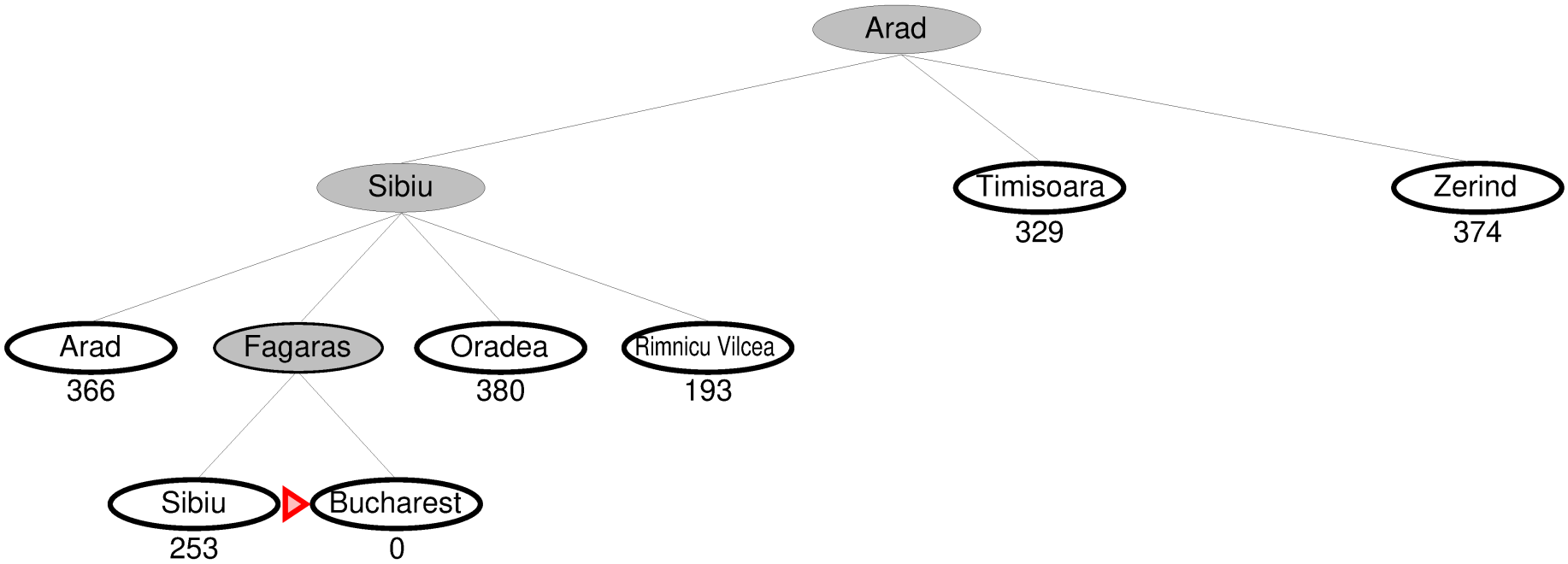
Best-first greedy search



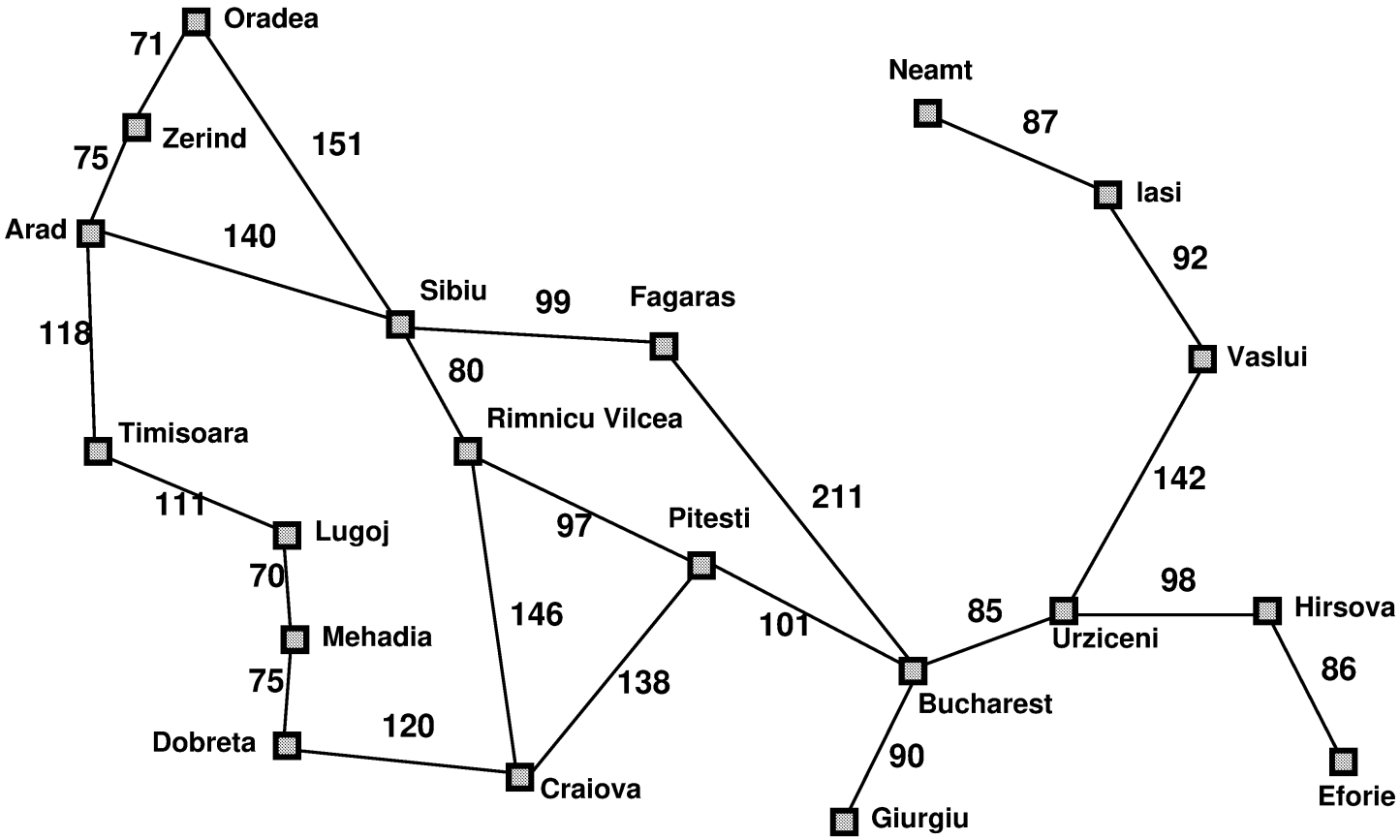
Best-first greedy search



Best-first greedy search

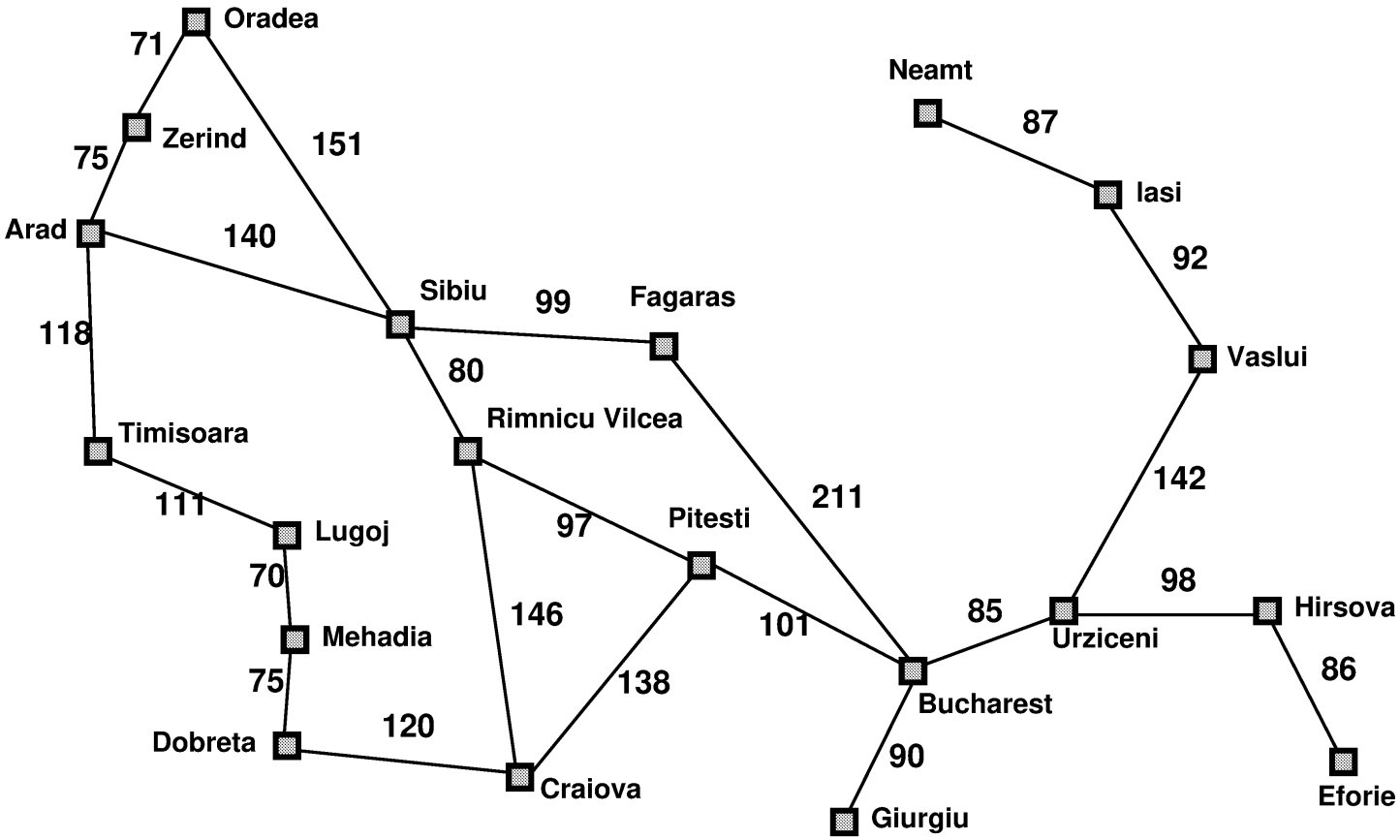


Best-first greedy search



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

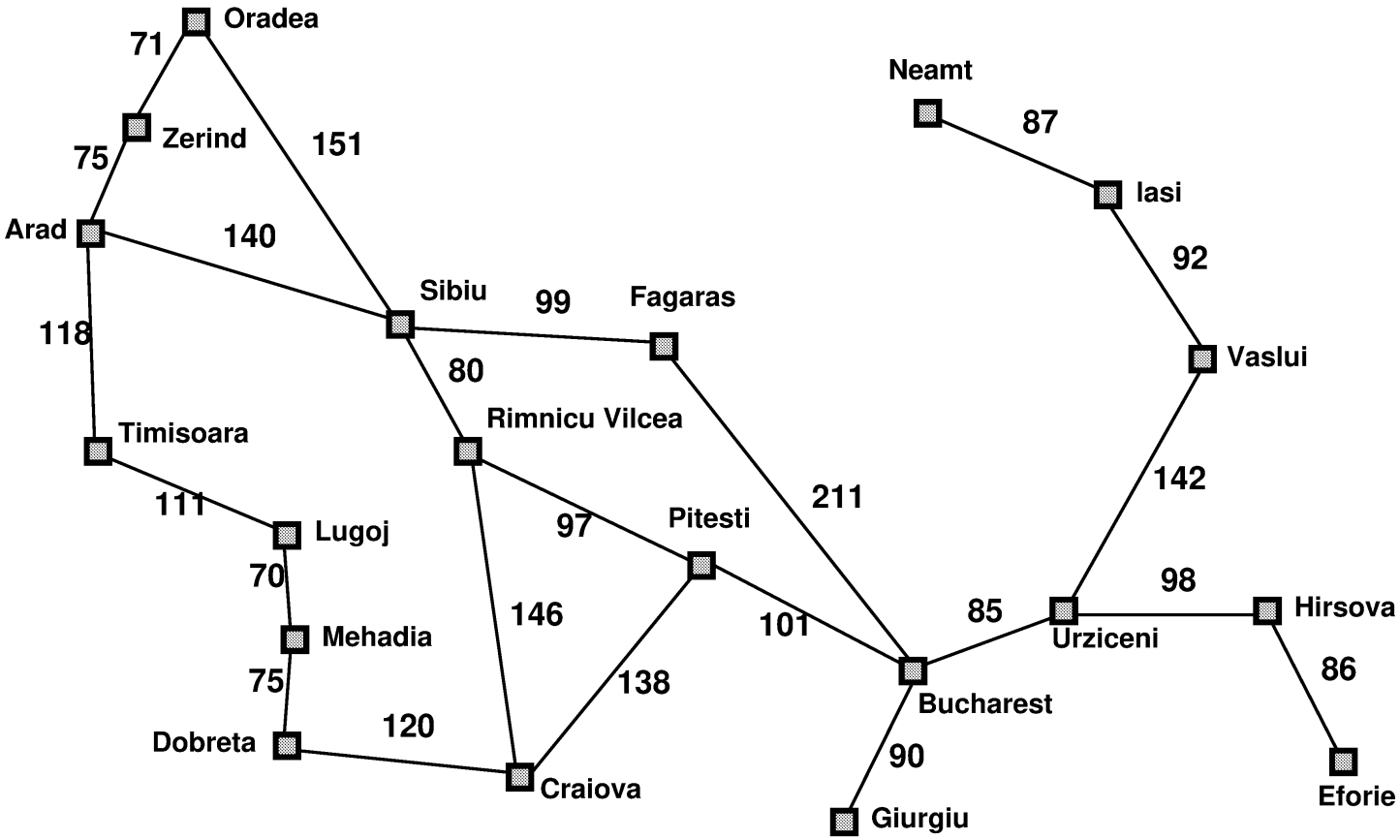
Best-first greedy search



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Solution found: Arad → Sibiu → Fagaras → Bucharest with total cost

Best-first greedy search

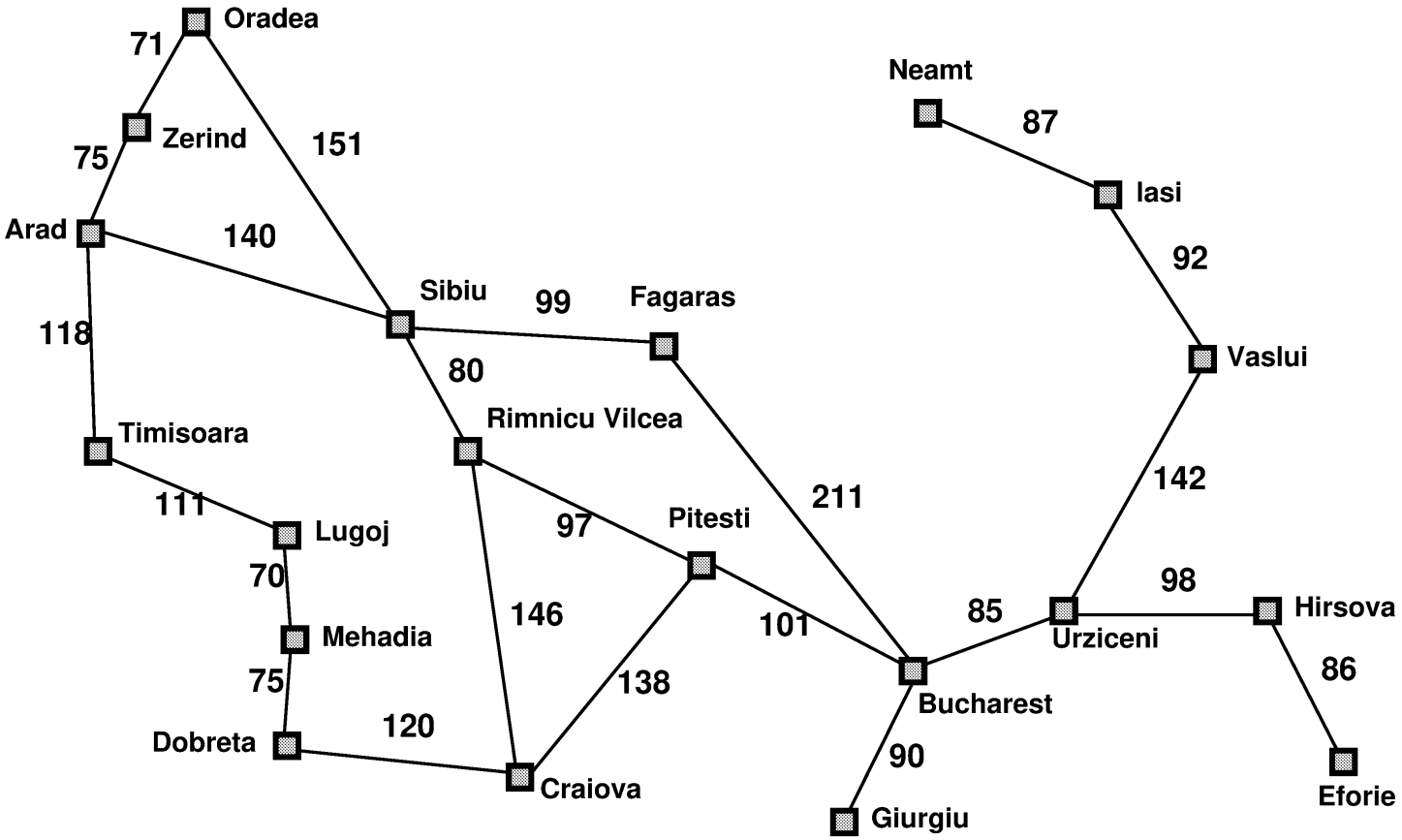


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Solution found: Arad → Sibiu → Fagaras → Bucharest with total cost 450

Is it optimal?

Best-first greedy search

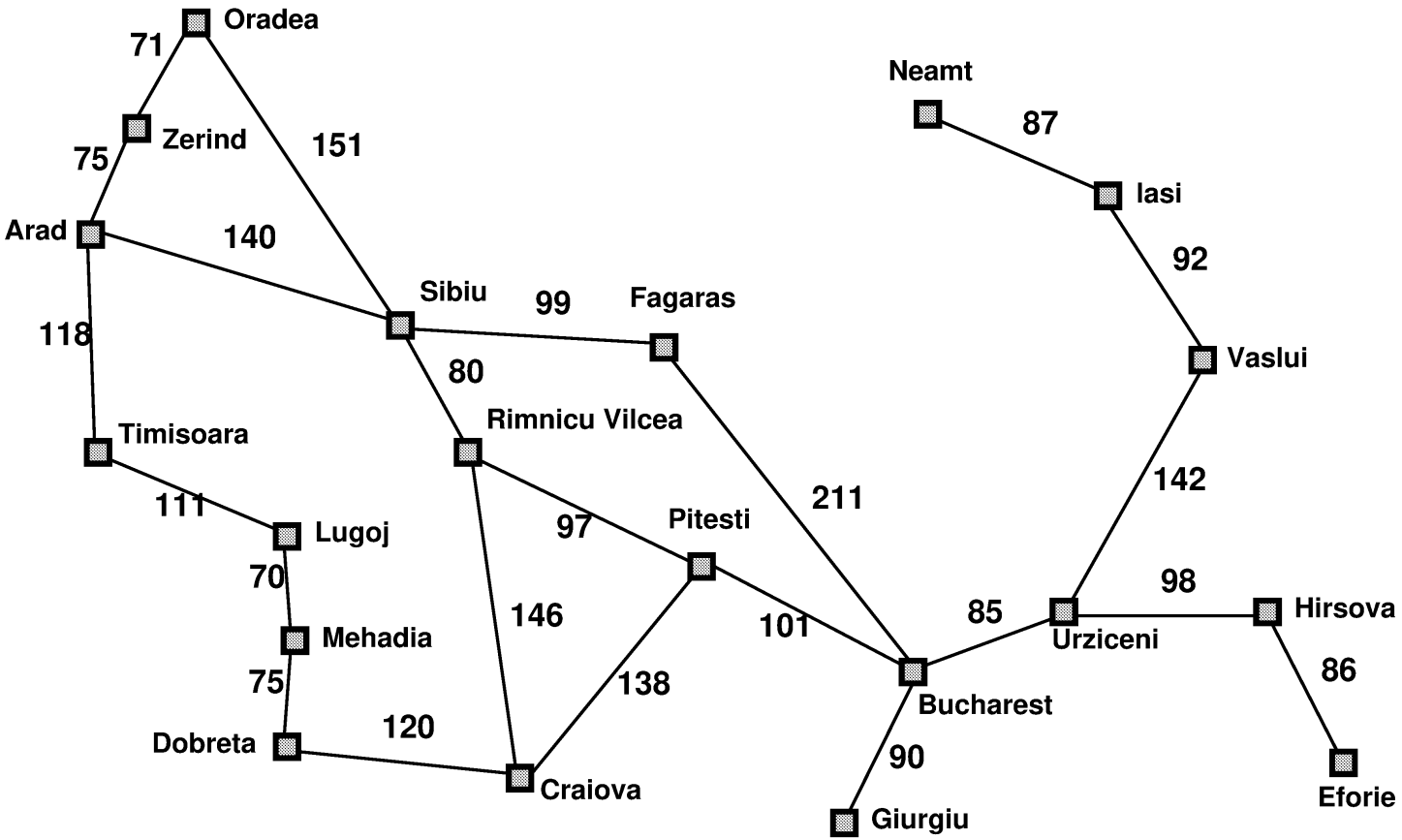


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Solution found: Arad → Sibiu → Fagaras → Bucharest with total cost 450

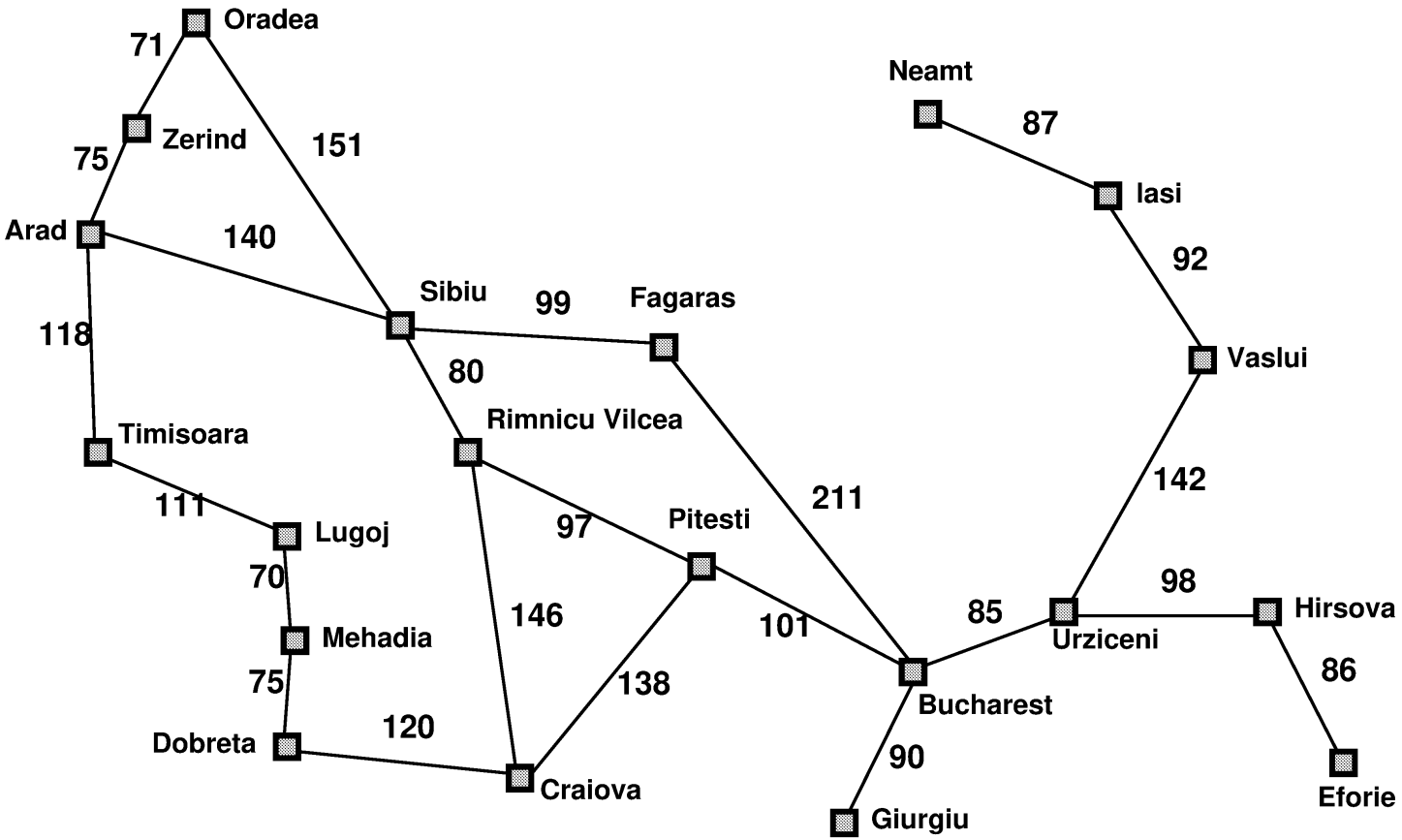
Is it optimal? — No!

Best-first greedy search



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Best-first greedy search



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Find a path from Iasi to Fagaras

Best-First Greedy: Complexities

- Is Greedy strategy complete?

Best-First Greedy: Complexities

- Is Greedy strategy complete? — No! — not in general
- Is it optimal?

Best-First Greedy: Complexities

- Is Greedy strategy complete? — No! — not in general
- Is it optimal? — No!
- Time complexity?

Best-First Greedy: Complexities

- Is Greedy strategy complete? — No! — not in general
- Is it optimal? — No!
- Time complexity? — $O(b^m)$
- Space complexity?

Best-First Greedy: Complexities

- Is Greedy strategy complete? — No! — not in general
- Is it optimal? — No!
- Time complexity? — $O(b^m)$
- Space complexity? — $O(b^m)$

Questions?