Informed Search

Introduction

- Uses problem-specific knowledge beyond the definition of the problem itself

Outline

- Best-first search
- Greedy best-first search
- A^* search
- Heuristics
- Local search algorithms
- Hill-climbing search
- Simulated annealing search
- Local beam search
- Genetic algorithms

Best-first search

- Idea: use an evaluation function *f(n)* for each node
  - estimate of "desirability"

  → Expand most desirable unexpanded node

- <u>Implementation</u>:

  Order the nodes in fringe in decreasing order of desirability

- Special cases:
  - greedy best-first search
  - A^* search

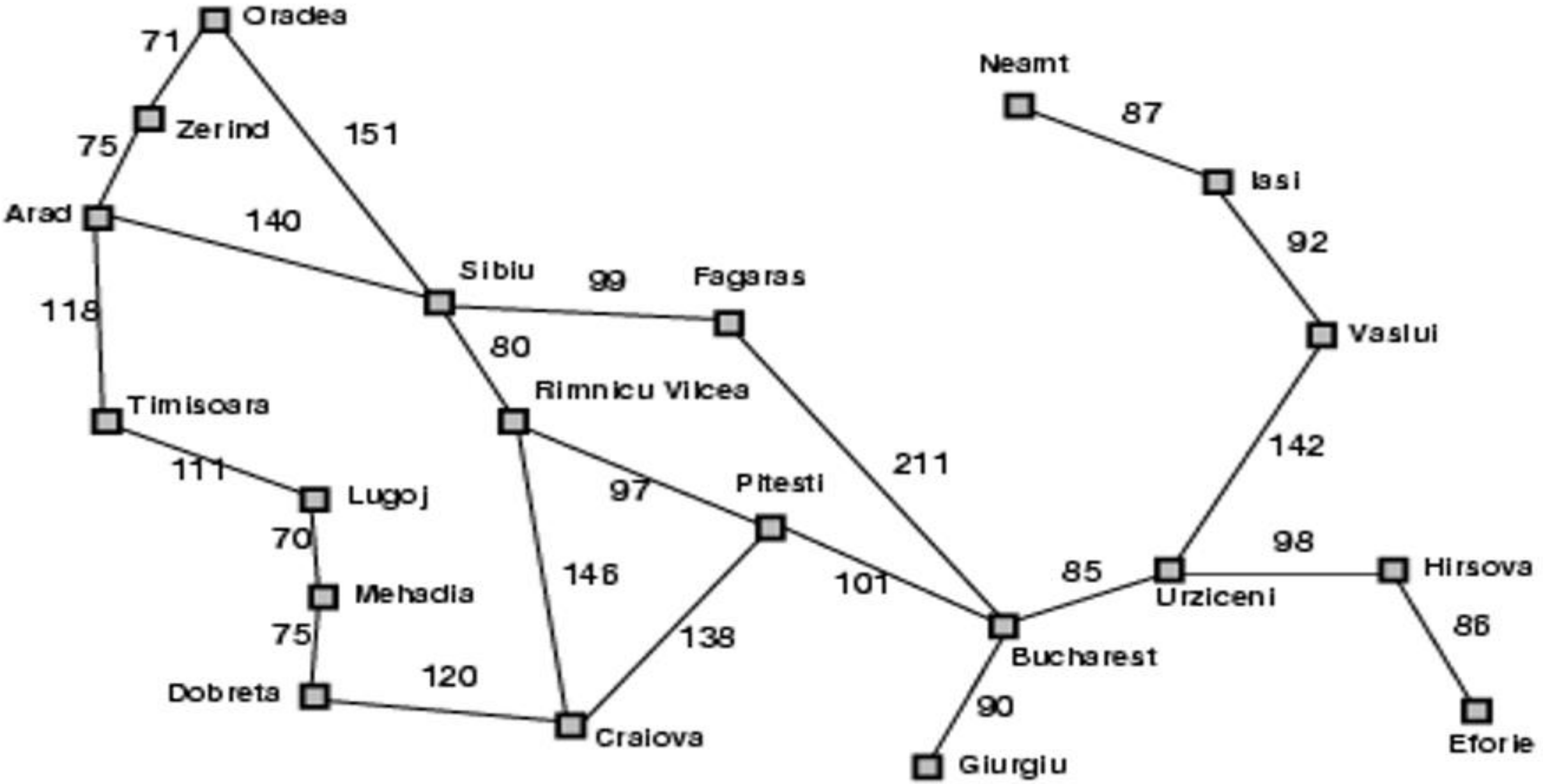
Greedy best first search

- Greedy best-first search tries to expand the node that is closest to the goal
- It evaluates nodes by using just the heuristic function; that is, $f(n) = h(n)$

Example (route-finding problems in Romania)

- Evaluation function $f(n) = h(n)$ (heuristic)= estimate of cost from n to *goal*
- $h_{SLD}(n)$ = straight-line distance from n to Bucharest

Romania with step costs in km: g(n)

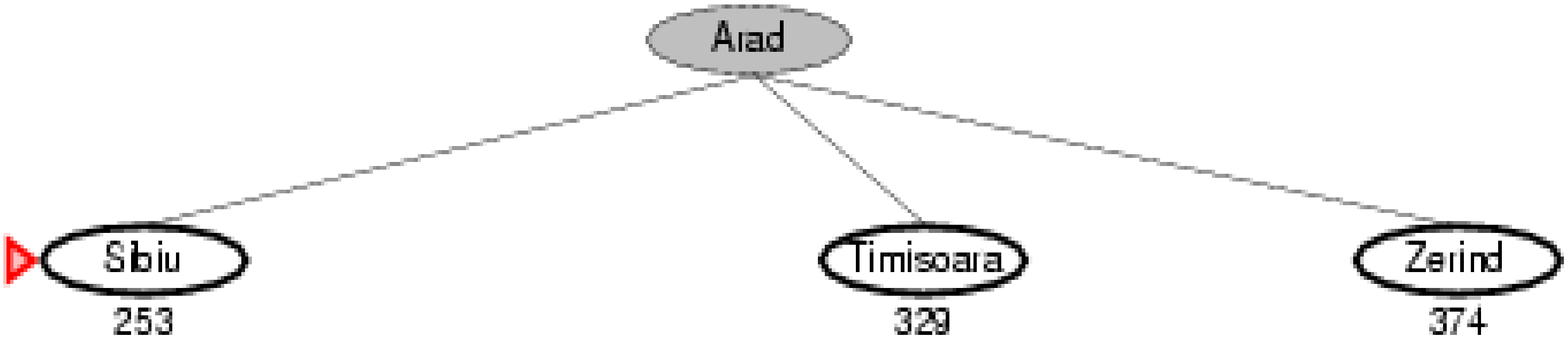


Straight-line distances to Bucharest: h(n)

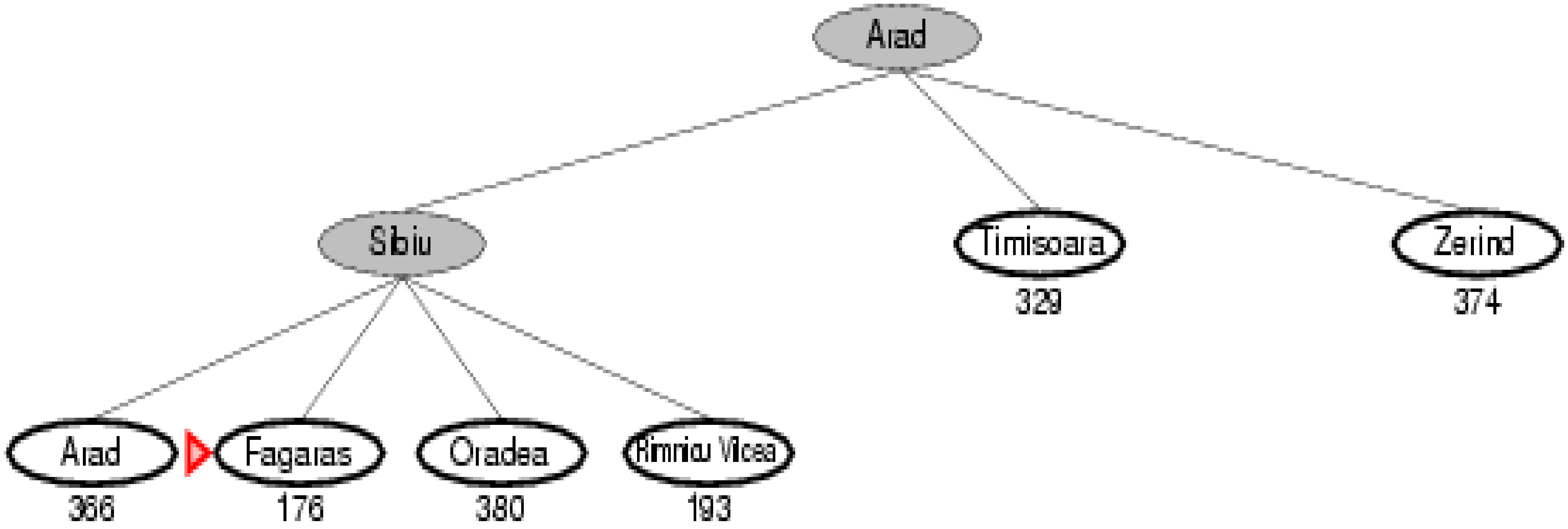
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Greedy best-first search example

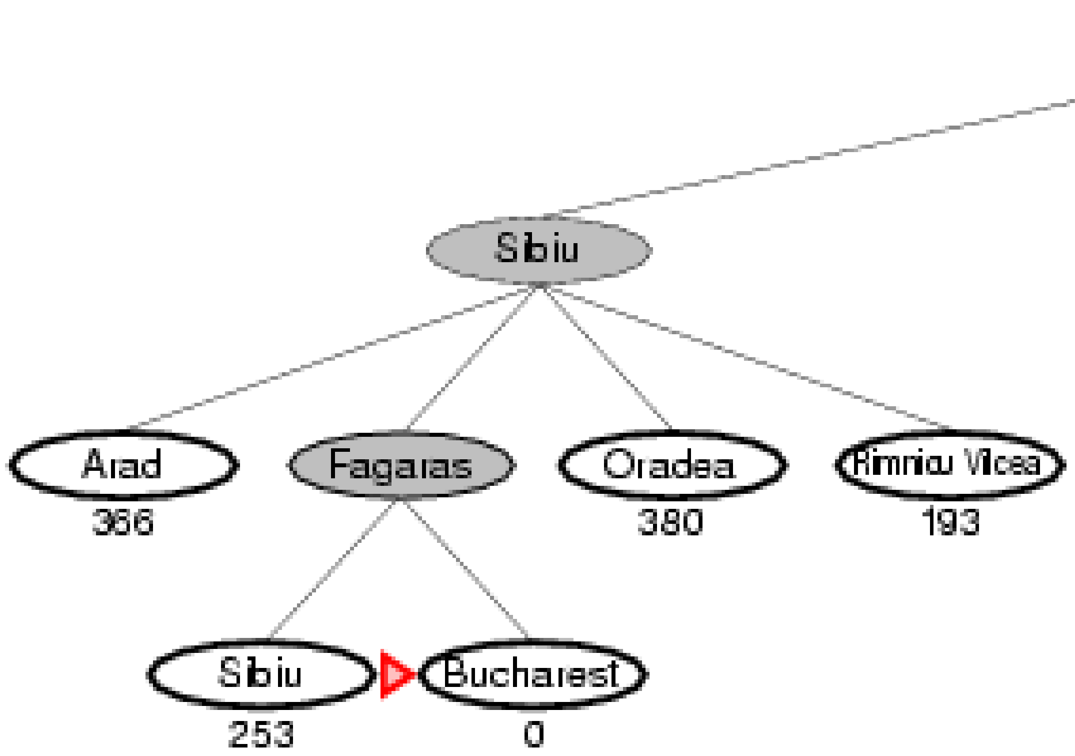
Greedy best-first search example



Greedy best-first search example



Greedy best-first search example



- It is not optimal: the path via Sibiu and Fagaras to Bucharest is 32 kilometers longer than the path through Rimnicu Vilcea and Pitesti
- This shows why the algorithm is called “greedy”—at each step it tries to get as close to the goal as it can
- Incomplete even in a finite state space, much like depth-first search
- Consider the problem of getting from Iasi to Fagaras
- The heuristic suggests that Neamt be expanded first because it is closest to Fagaras, but it is a dead end
- The solution is to go first to Vaslui—a step that is actually farther from the goal according to the heuristic—and then to continue to Urziceni, Bucharest, and Fagaras
- The algorithm will never find this solution, however, because expanding Neamt puts Iasi back into the frontier, Iasi is closer to Fagaras than Vaslui is, and so Iasi will be expanded again, leading to an infinite loop

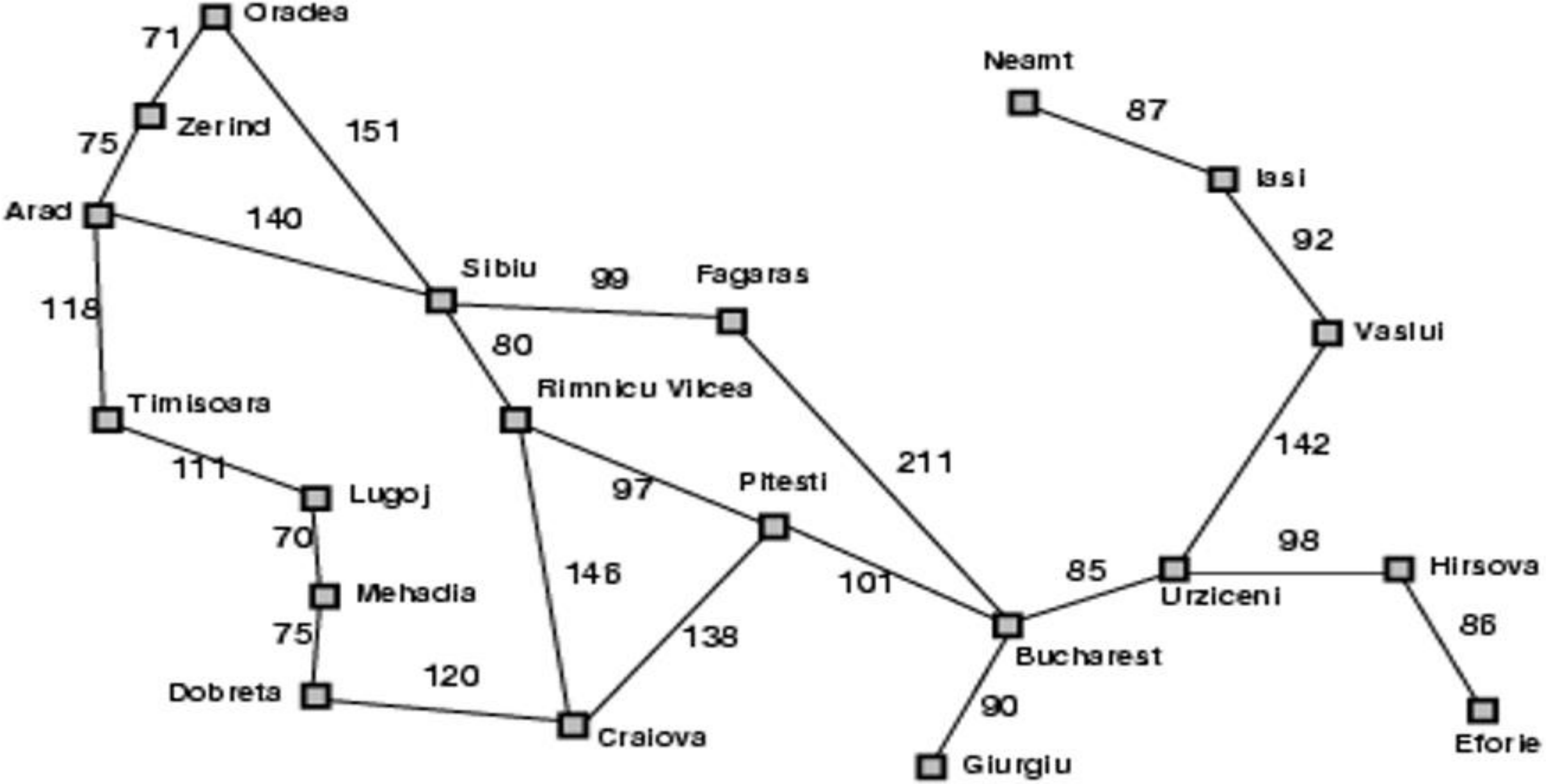
Properties of greedy best-first search

- Complete? No – can get stuck in loops,
  e.g., Iasi → Neamt → Iasi → Neamt →
- Time? $O(b^m)$, but a good heuristic can give dramatic improvement
- Space? $O(b^m)$ -- keeps all nodes in memory
- Optimal? No

A* search: Minimizing the total estimated solution cost

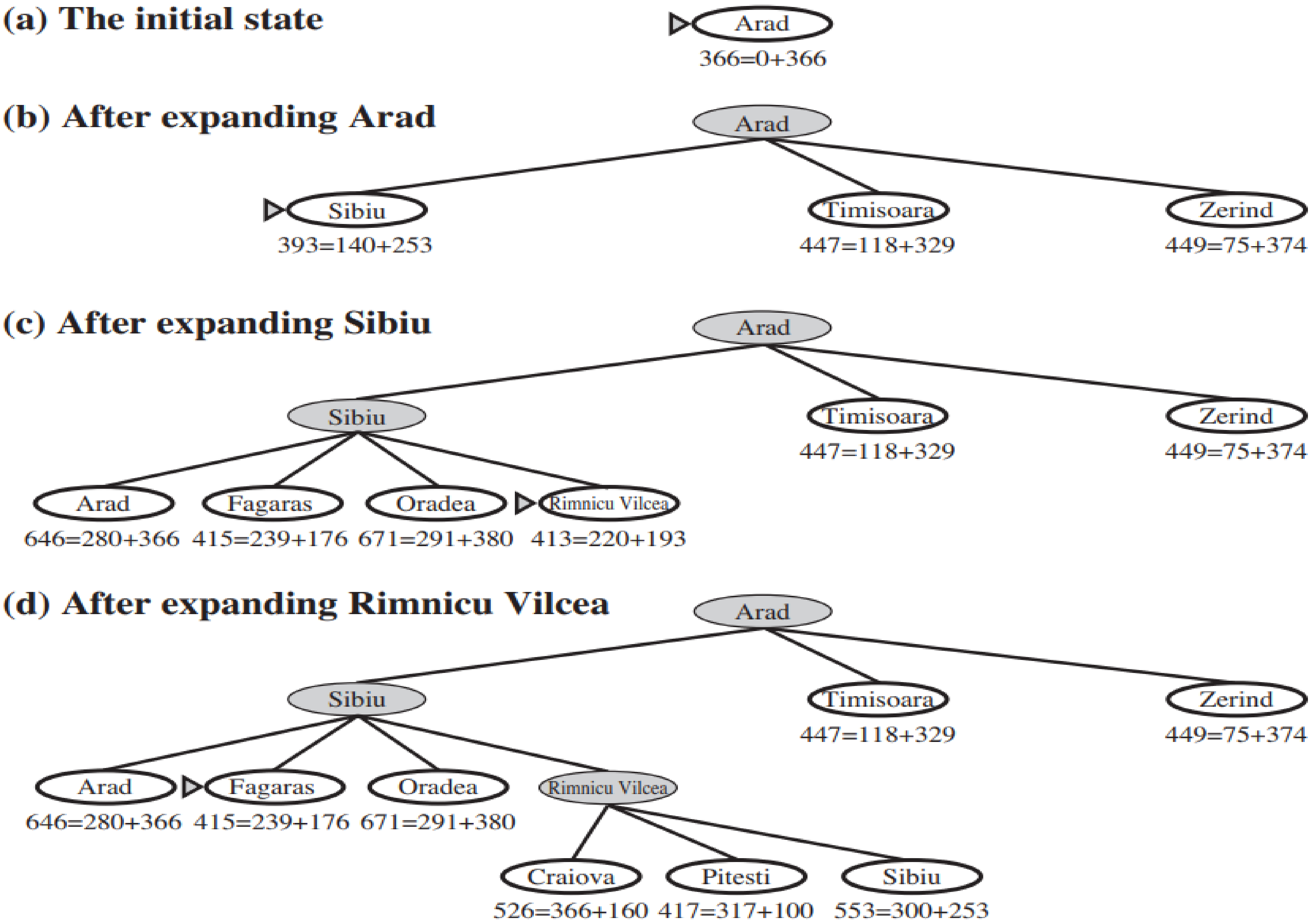
- The most widely known form of best-first search is called A^* search
- The algorithm is identical to UNIFORM-COST-SEARCH except that A^* uses g + h instead of g
- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = cost so far to reach n
- $h(n)$ = estimated cost from n to goal
- $f(n)$ = estimated total cost of path through n to goal

Romania with step costs in km: g(n)



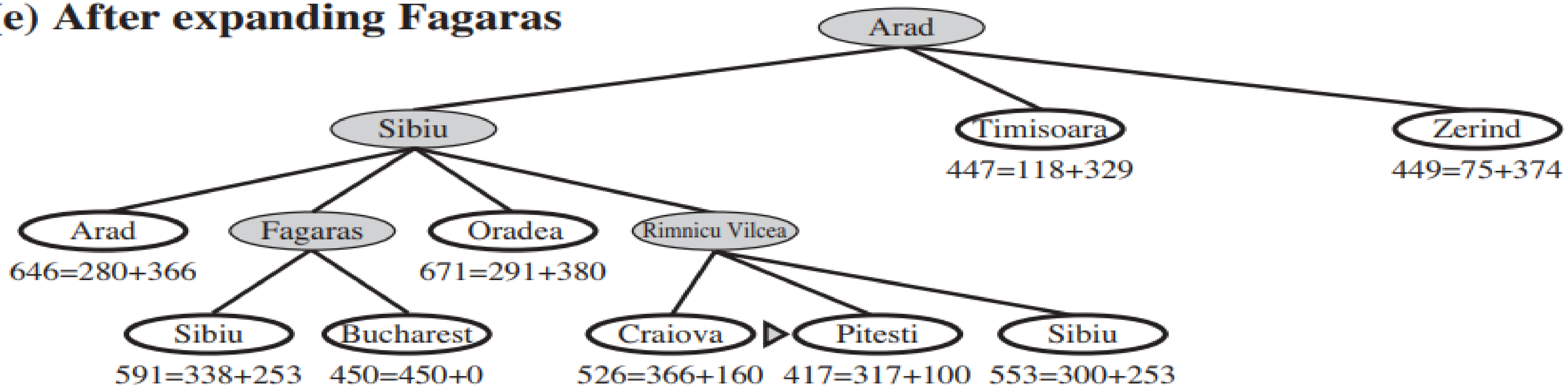
Straight-line distances to Bucharest: h(n)

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

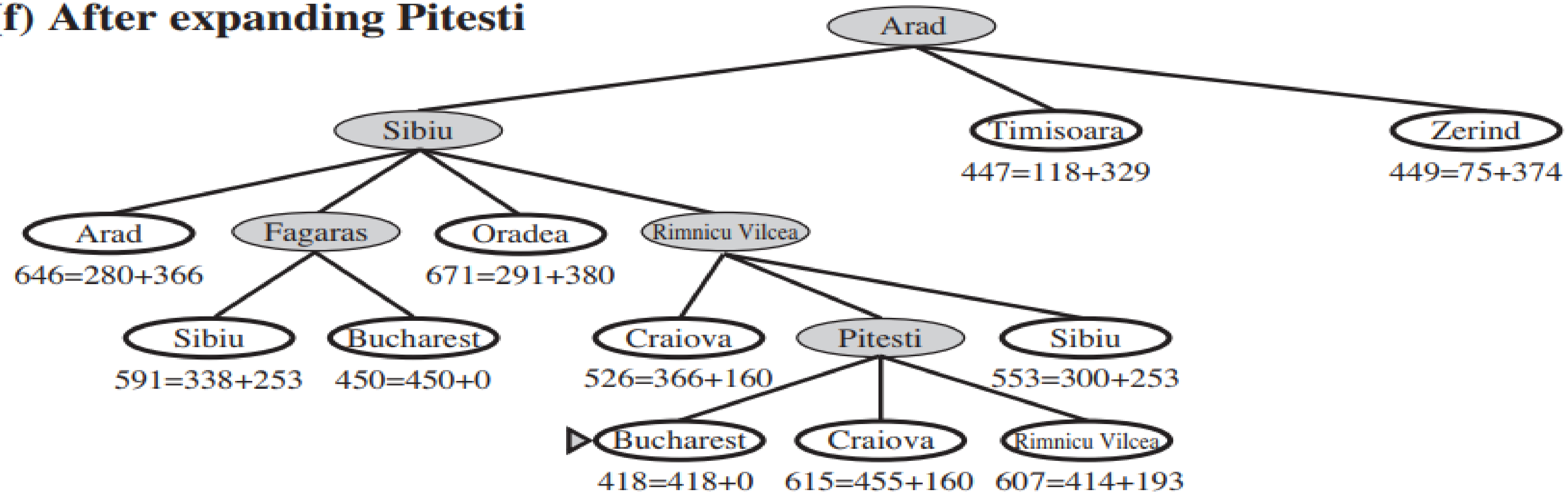


(a) The initial state
Arad
366=0+366
(b) After expanding Arad
Arad
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
413=220+193
(d) After expanding Rimnicu Vilcea
Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

(e) After expanding Fagaras



(f) After expanding Pitesti



Conditions for optimality: Admissibility and consistency

- The first condition we require for optimality is that h(n) be an admissible heuristic
- A heuristic h(n) is admissible if for every node n, $h(n) \le h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n
- An admissible heuristic is one that never overestimates the cost to reach the goal
- Admissible heuristics are by nature optimistic because they think the cost of solving the problem is less than it actually is
- An obvious example of an admissible heuristic is the straight-line distance h_{SLD} that we used in getting to Bucharest
- Straight-line distance is admissible because the shortest path between any two points is a straight line, so the straight line cannot be an overestimate

- A second, slightly stronger condition called consistency (or sometimes monotonicity) is required only for applications of A∗ to graph search
- A heuristic h(n) is consistent if, for every node n and every successor n’ of n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n’ plus the estimated cost of reaching the goal from n : $h(n) \leq c(n, a, n') + h(n')$
- Every consistent heuristic is also admissible
- Consistency is therefore a stricter requirement than admissibility