COMP3411/9814: Artificial Intelligence

Solving problems by searching
Informed Search
Examples

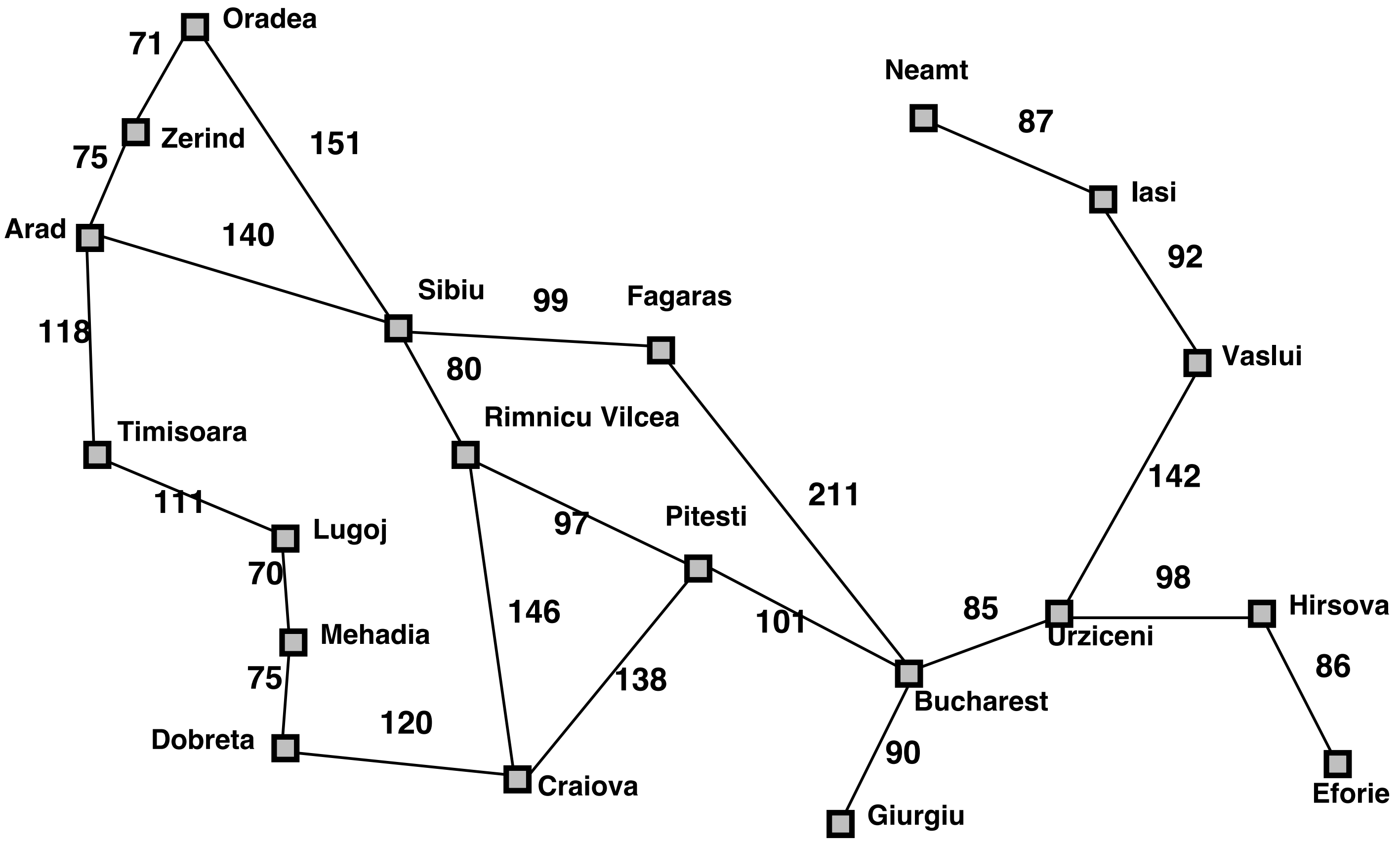
Tatjana Zrimec, 2020

Example: Romania

On holiday in Romania; currently in Arad.
Flight leaves tomorrow from Bucharest: non-refundable ticket.

- Step 1 Formulate goal:
  - be in Bucharest on time
- Step 2 Formulate problem - Specify task:
  - states: various cities
  - actions (operators) (= transitions between states): drive between cities
- Step 3 Find solution (= action sequences): sequence of cities, e.g. Arad, Sibiu, Fagaras, Bucharest
- Step 4 Execute: drive through all the cities given by the solution.

Romania with step costs in km

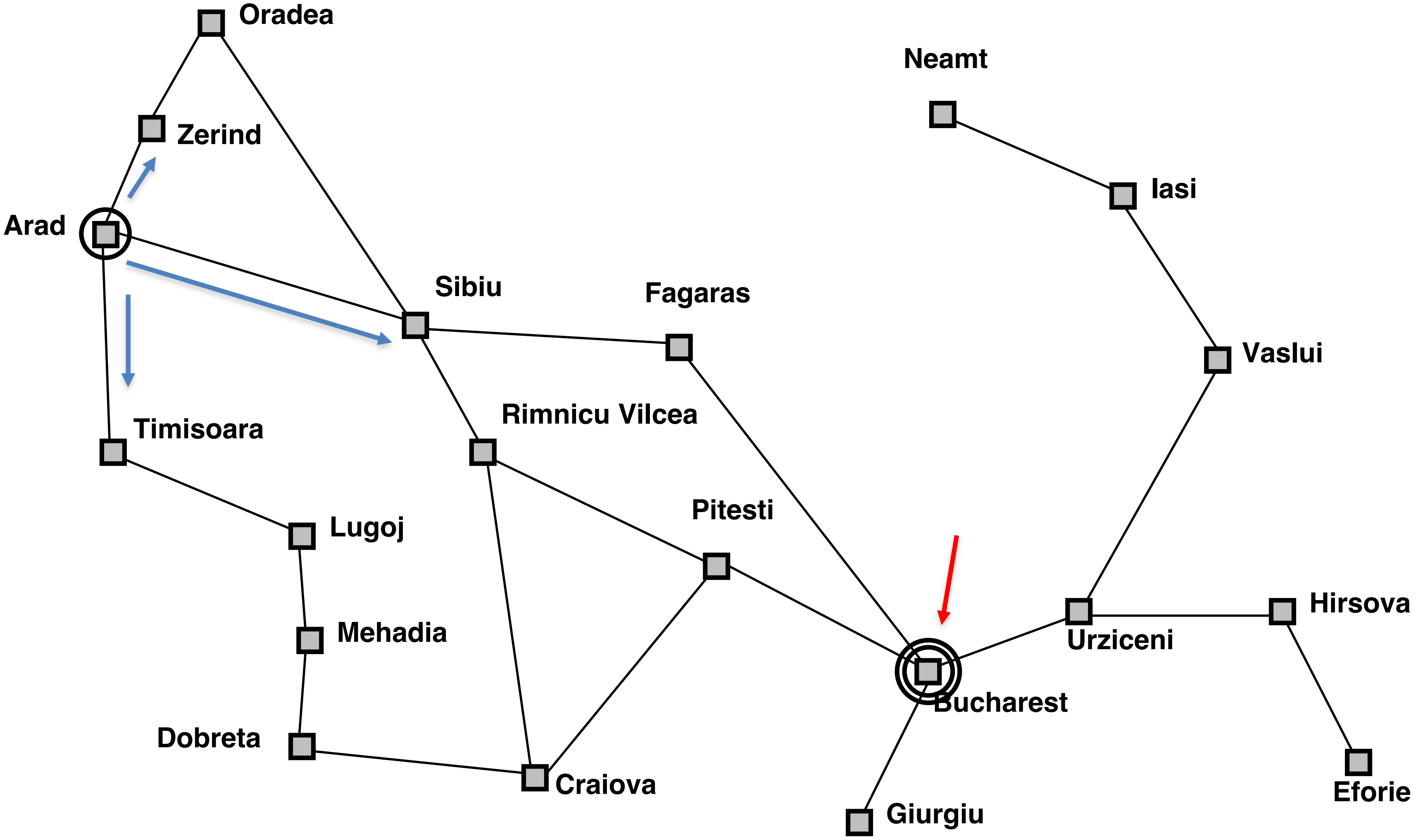


Straight–line distance to Bucharest

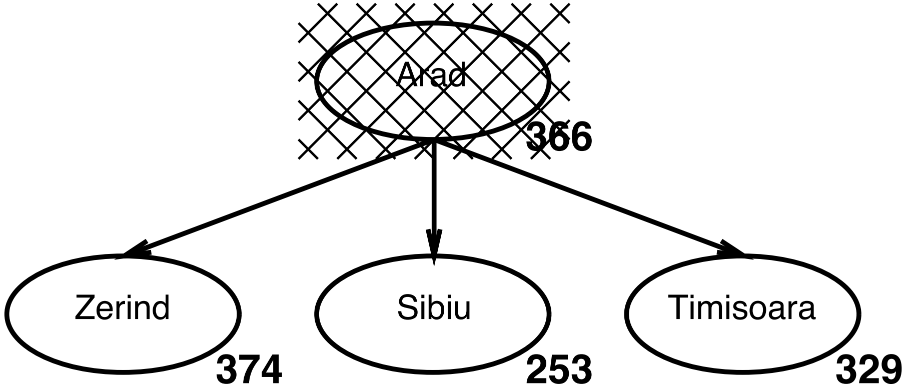
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

We are often looking for the path with the shortest total distance rather than the number of steps.

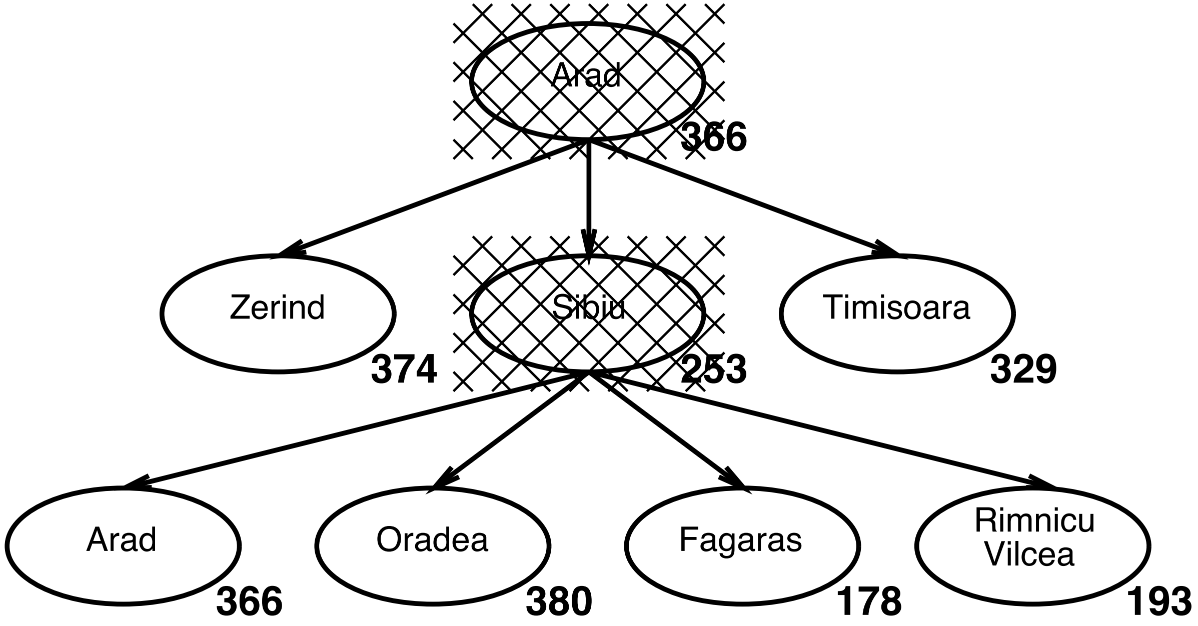
Example: Romania



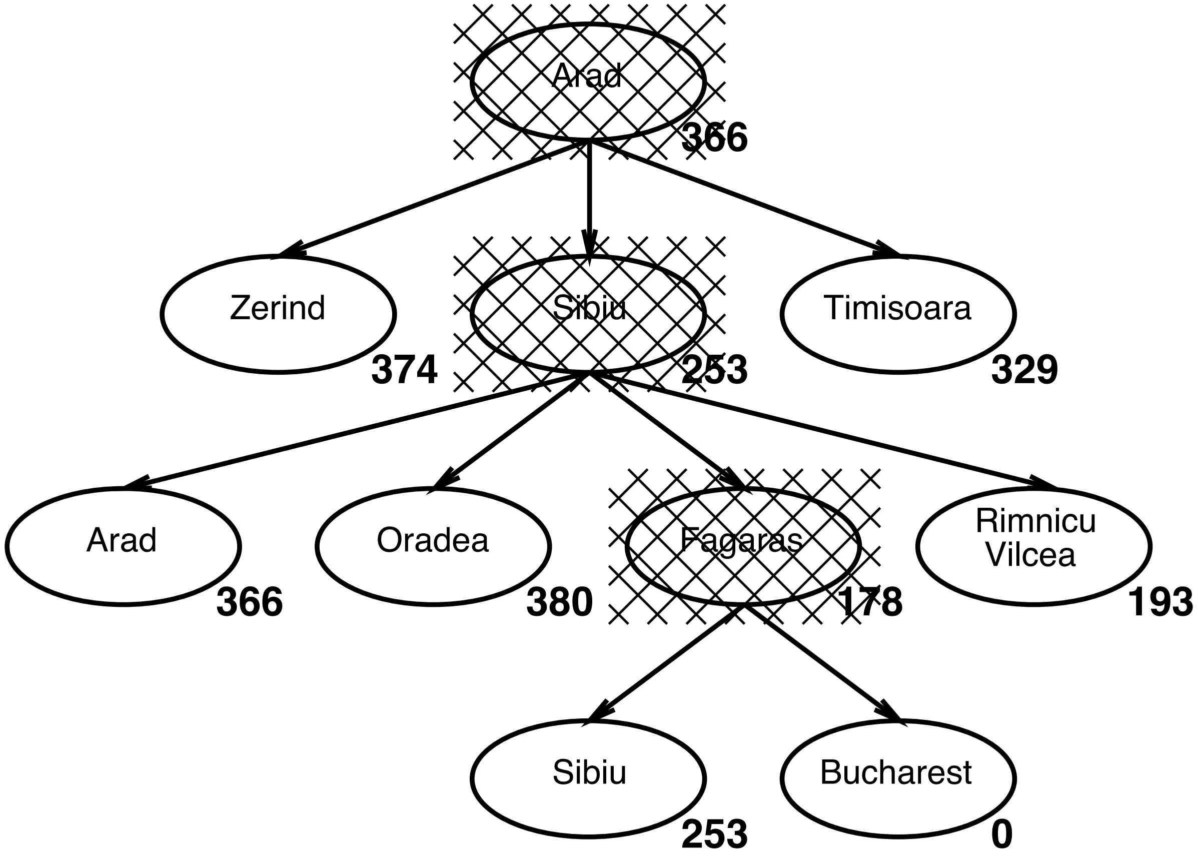
Greedy Best-First Search Example



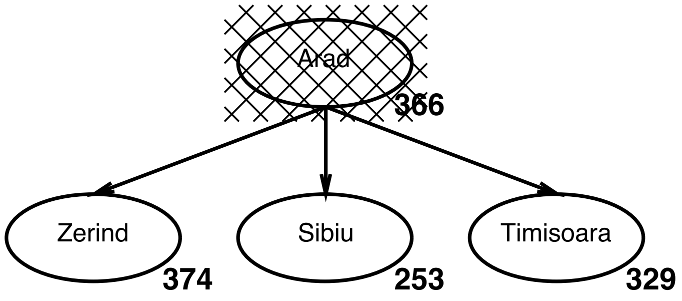
Greedy Best-First Search Example



Greedy Best-First Search Example



Greedy Best-First Search Example

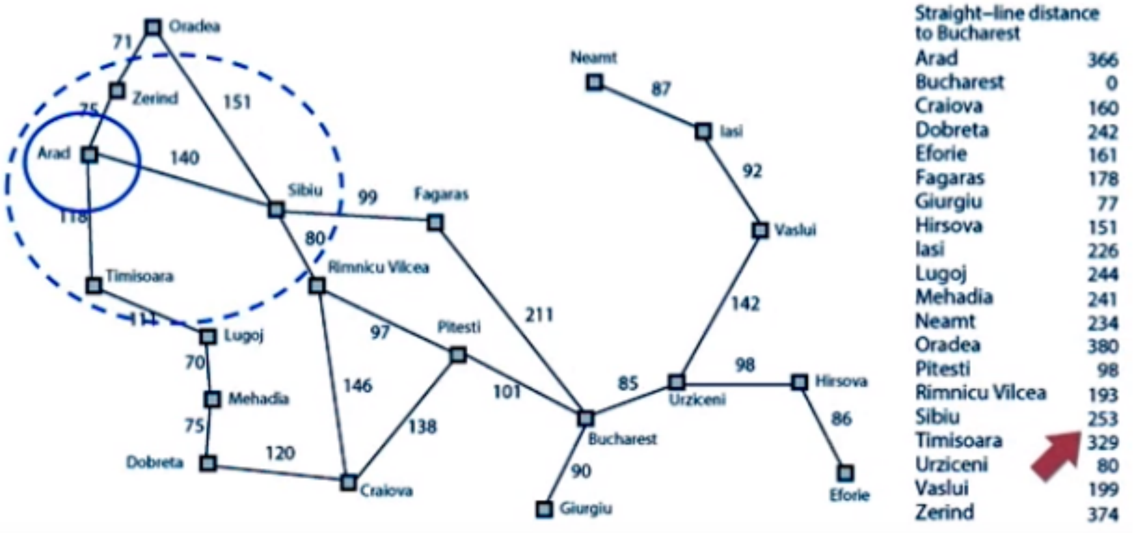


Arad
366

Zerind
374

Sibiu
253

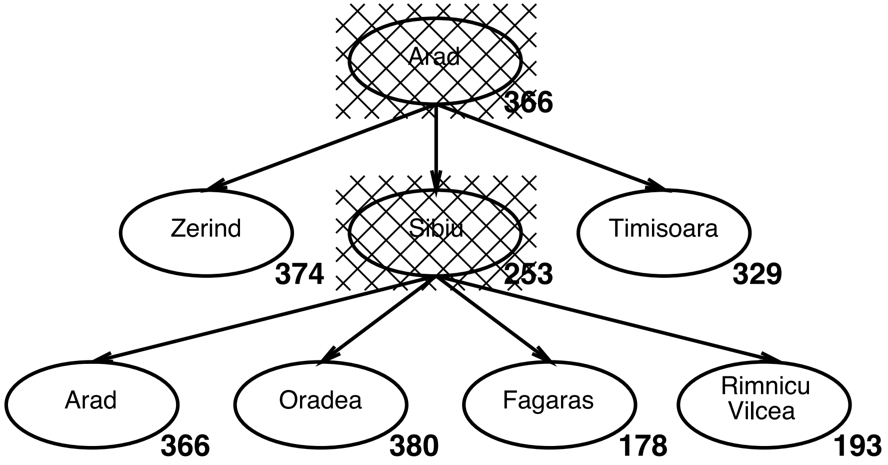
Timisoara
329

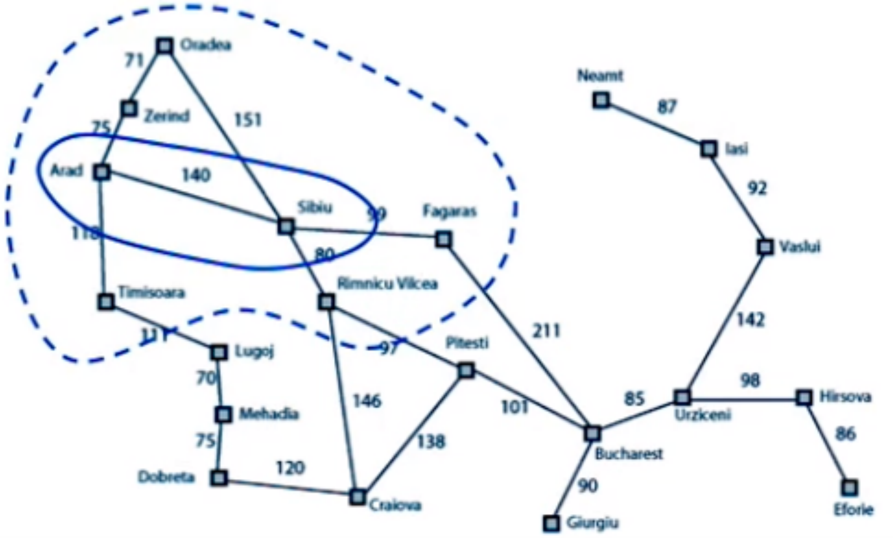


Straight–line distance to Bucharest

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

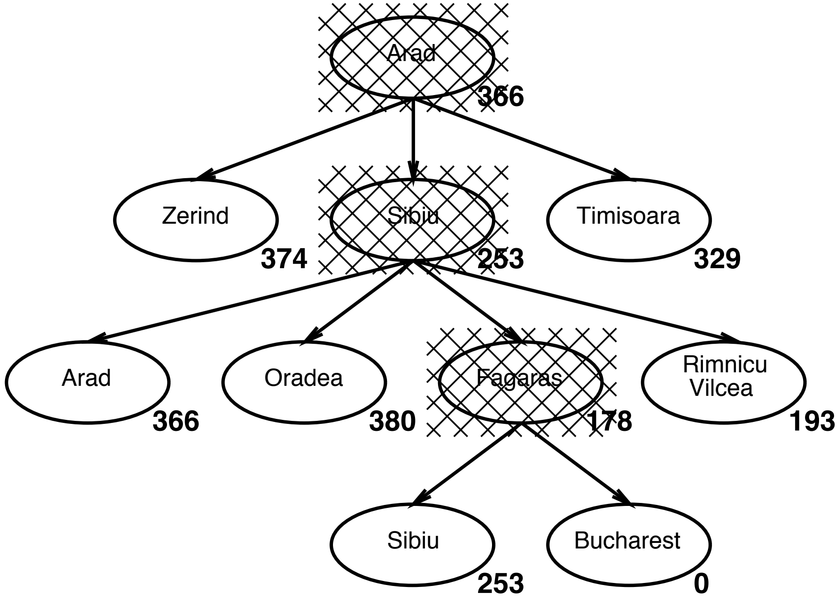
Greedy Best-First Search Example

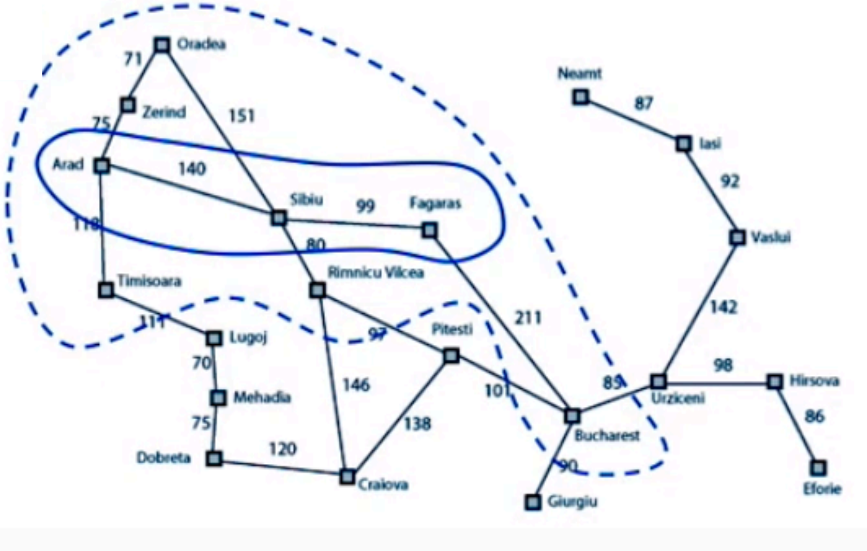




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

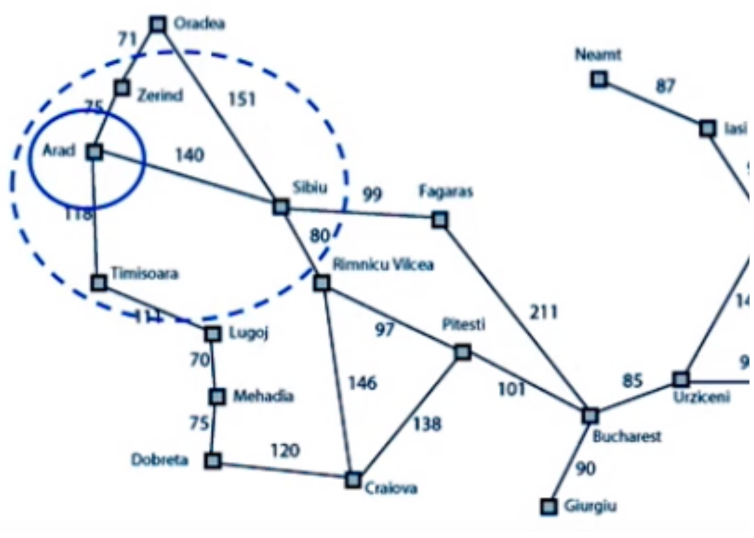
Greedy Best-First Search Example

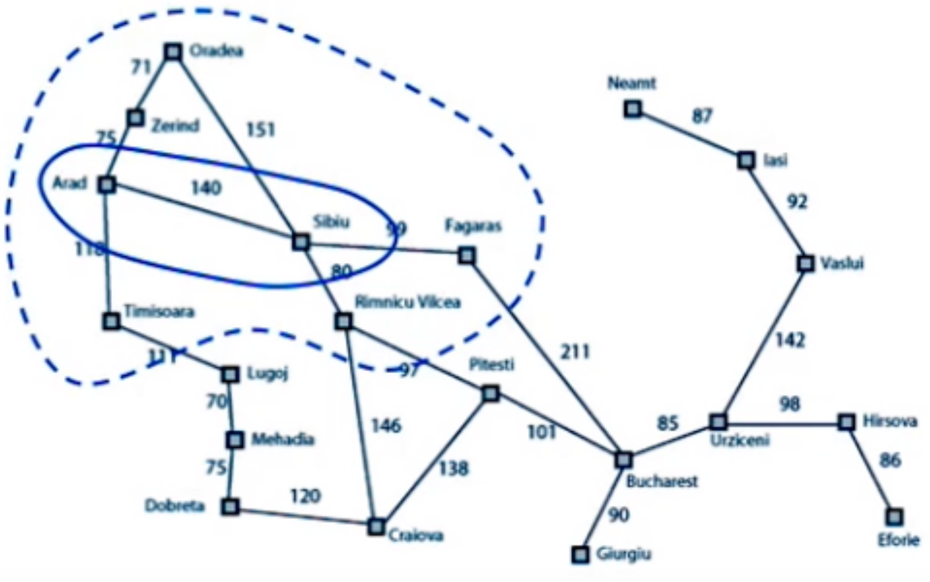




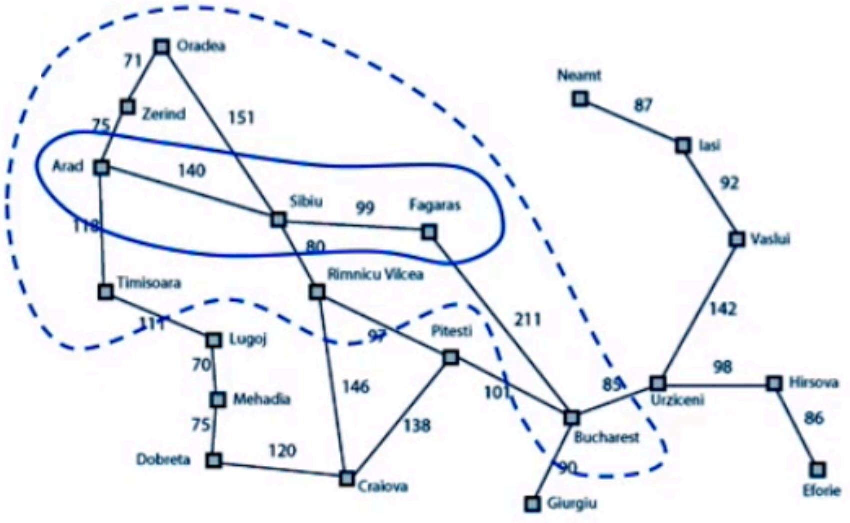
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy Best-First Search Example





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

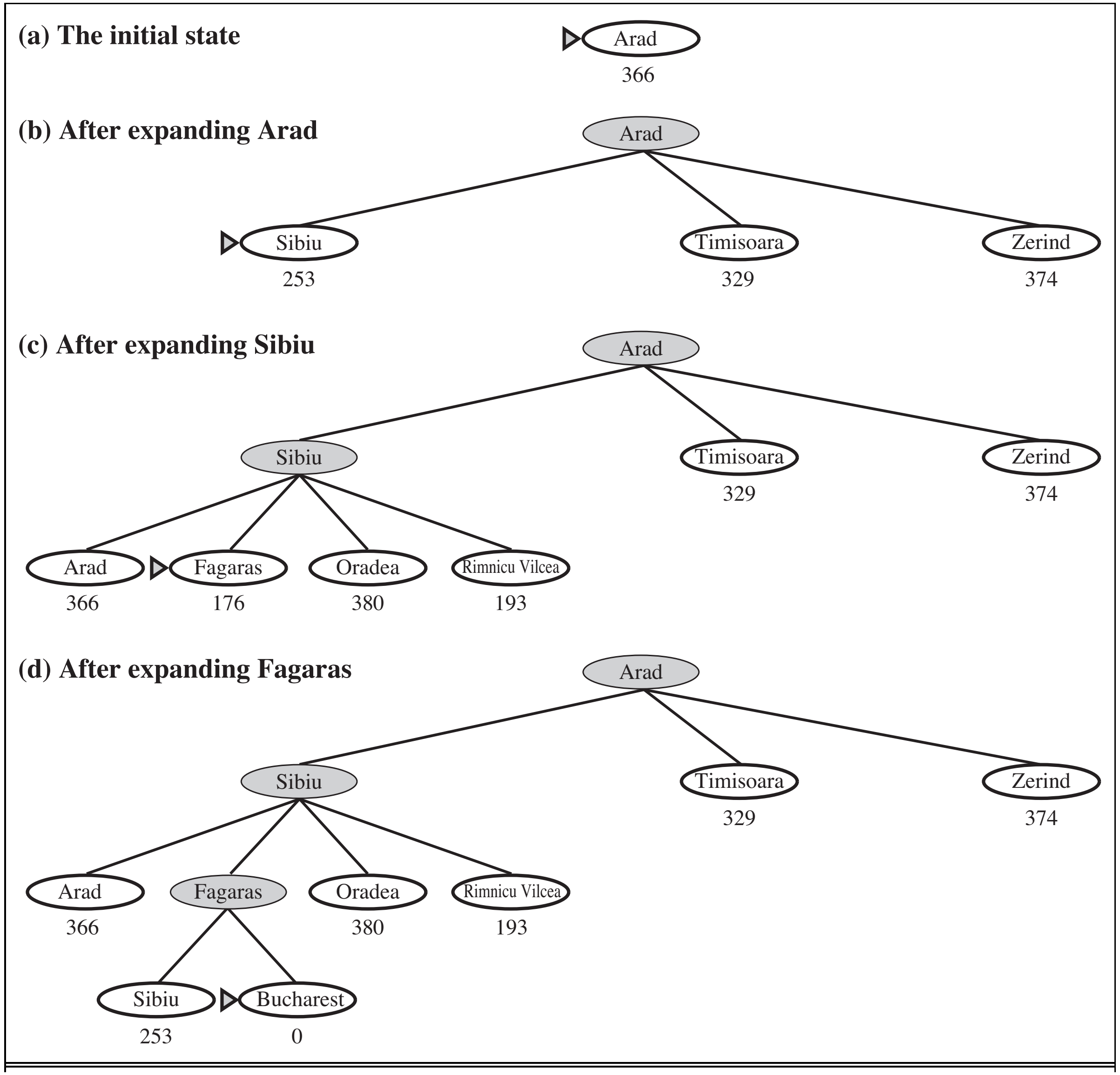


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Examples of Greedy Best-First Search

Implementation:

Order the nodes in decreasing order of desirability

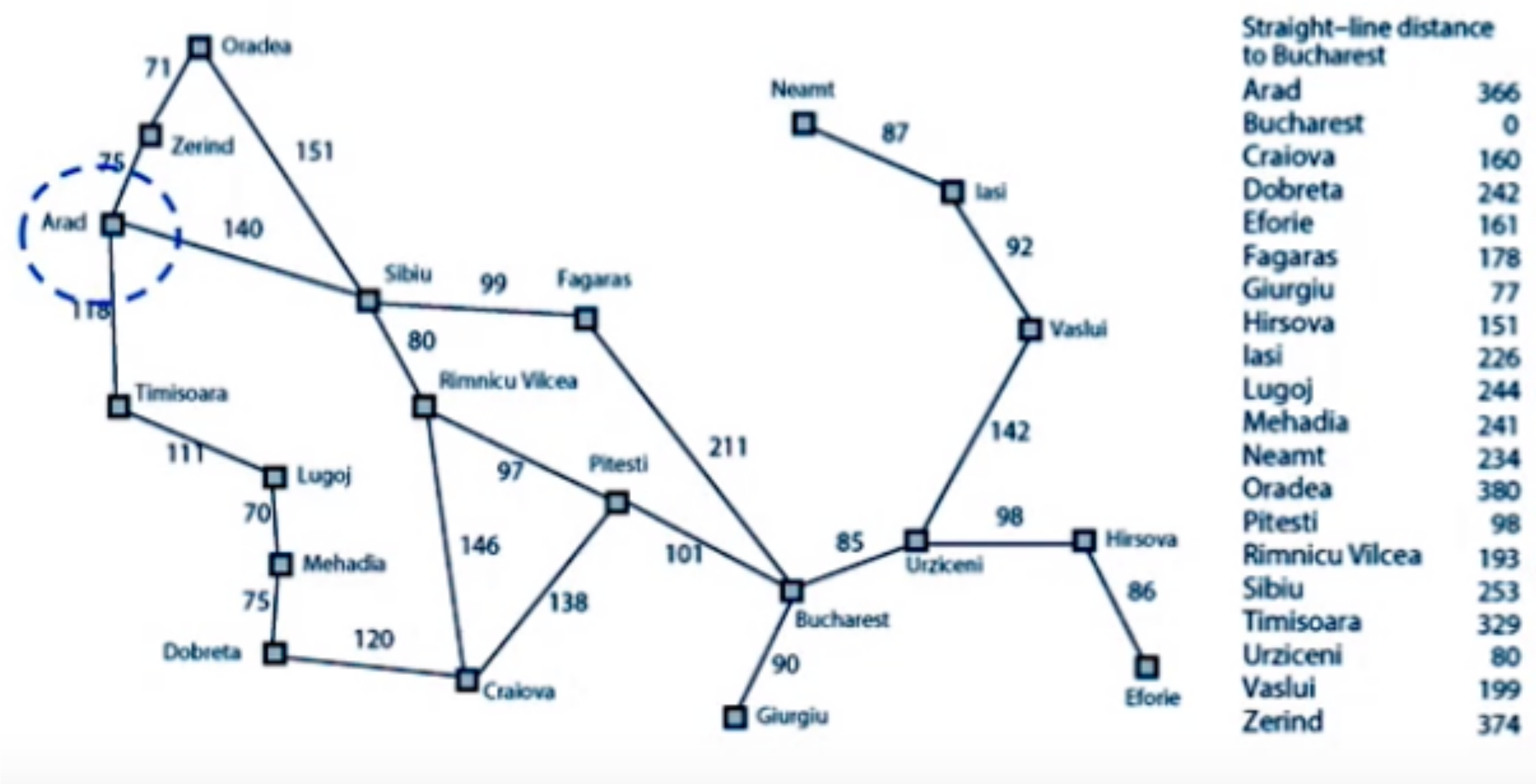


Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic h_{SLD} .

Nodes are labeled with their h-values.

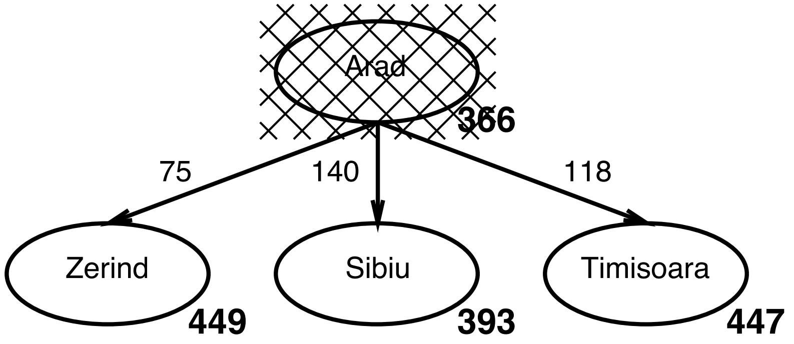
A* Search

- ❑ The SLD distances are underestimates
- ❑ Can not be a shorter path then the SLD in real life

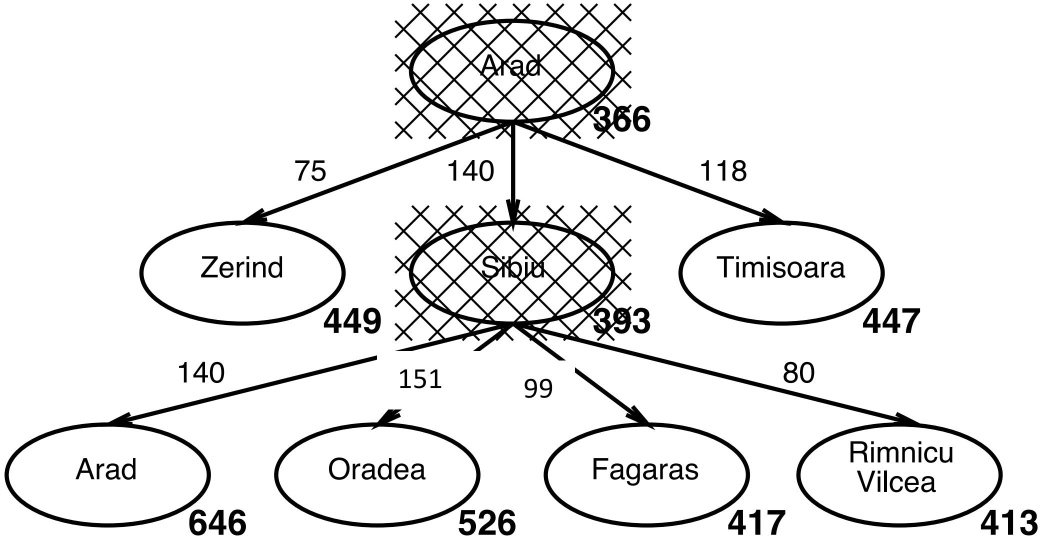


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

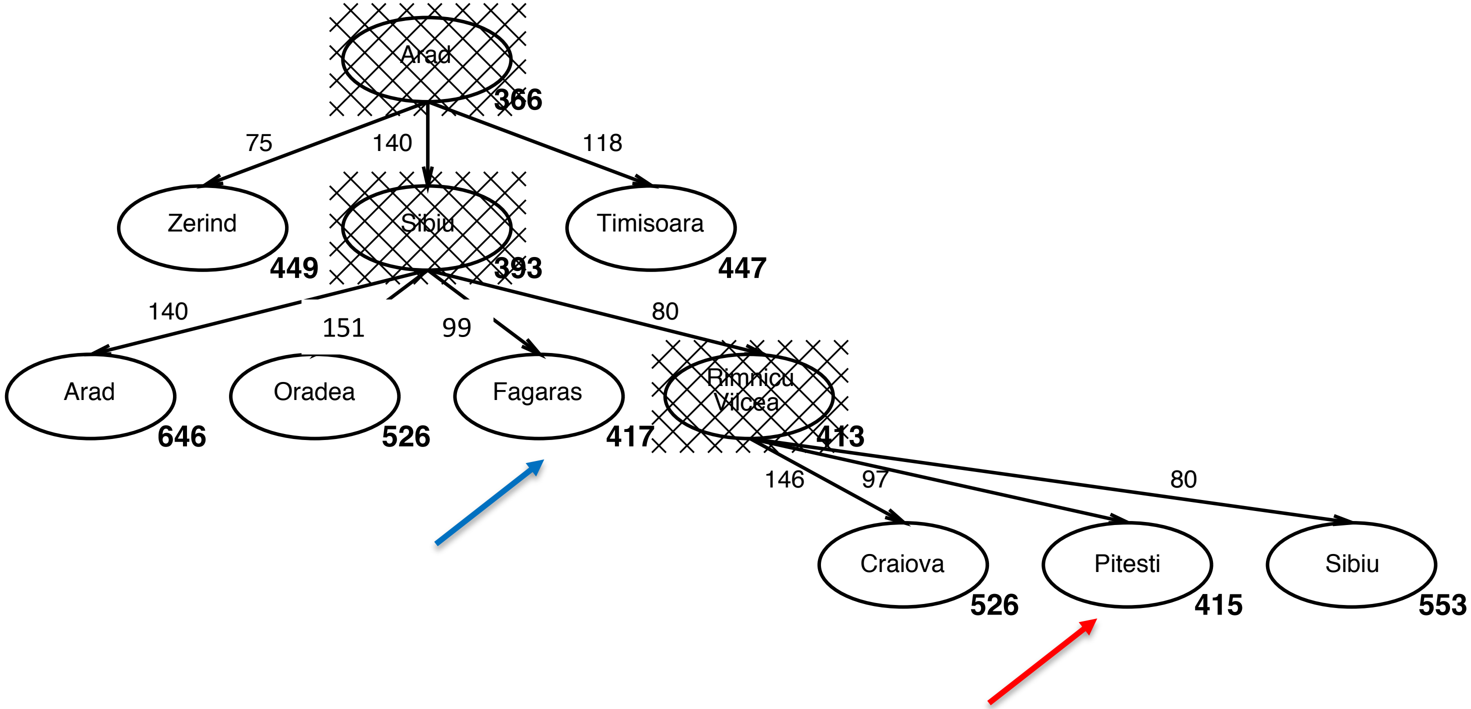
A∗ Search Example



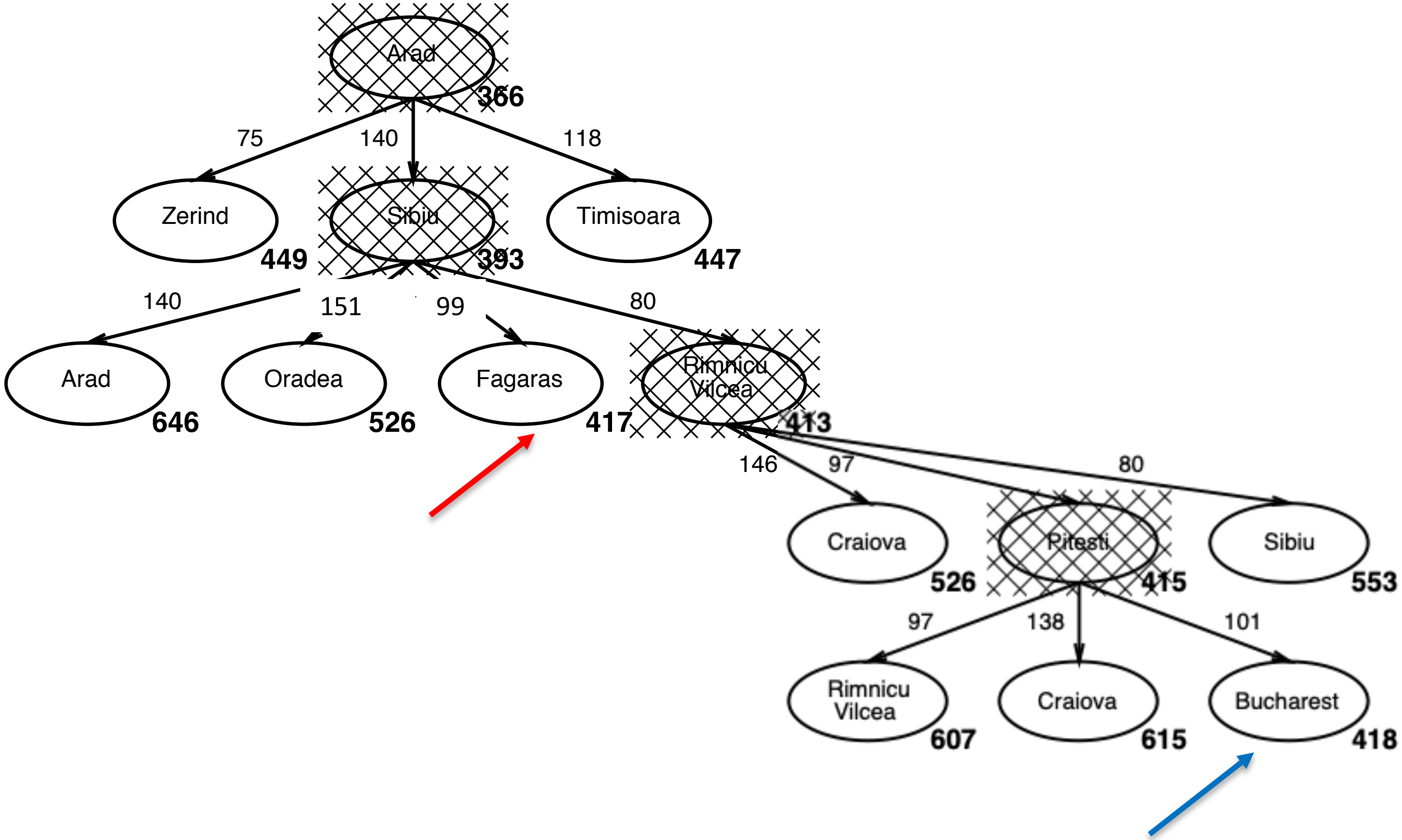
A∗ Search Example



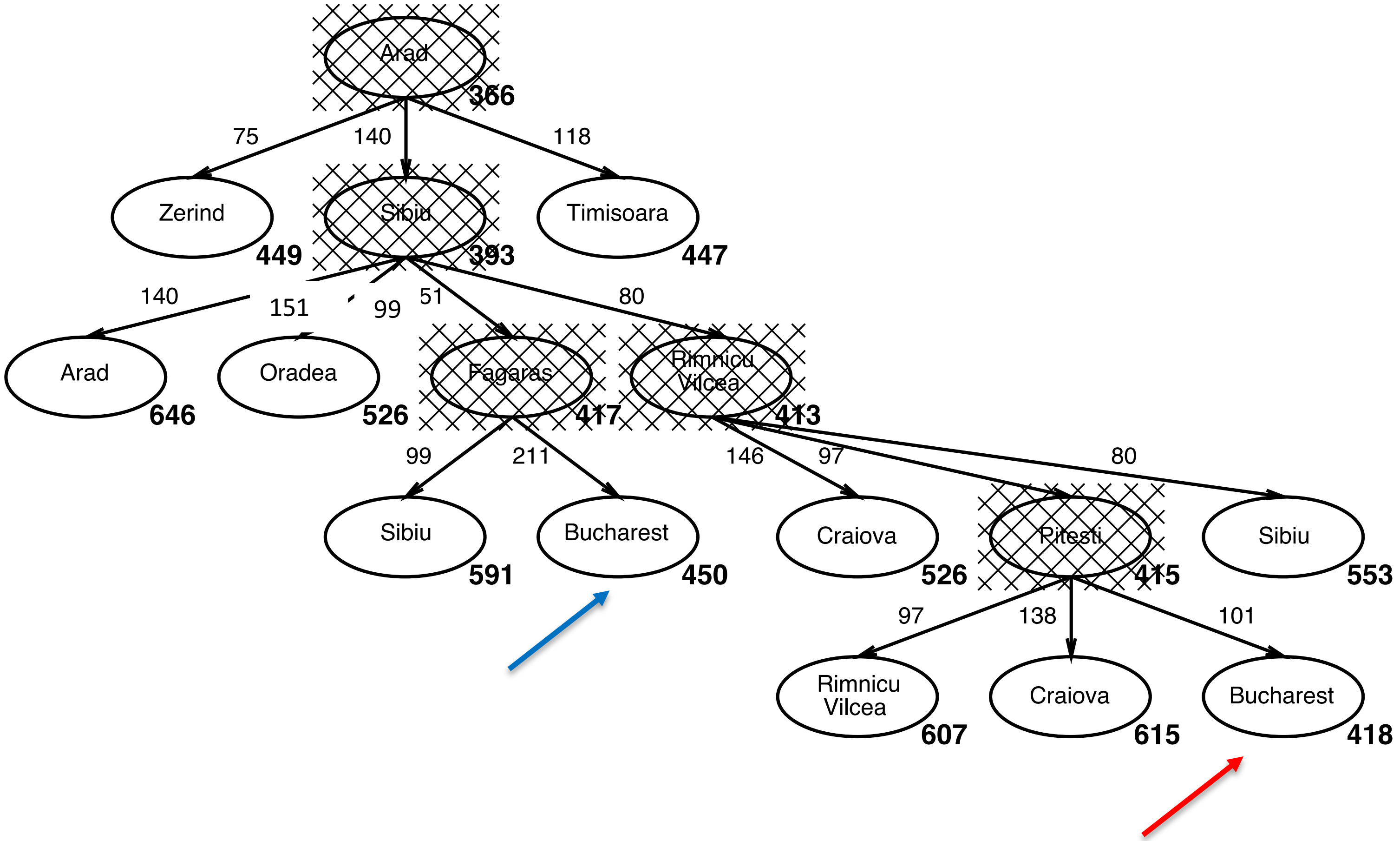
A∗ Search Example



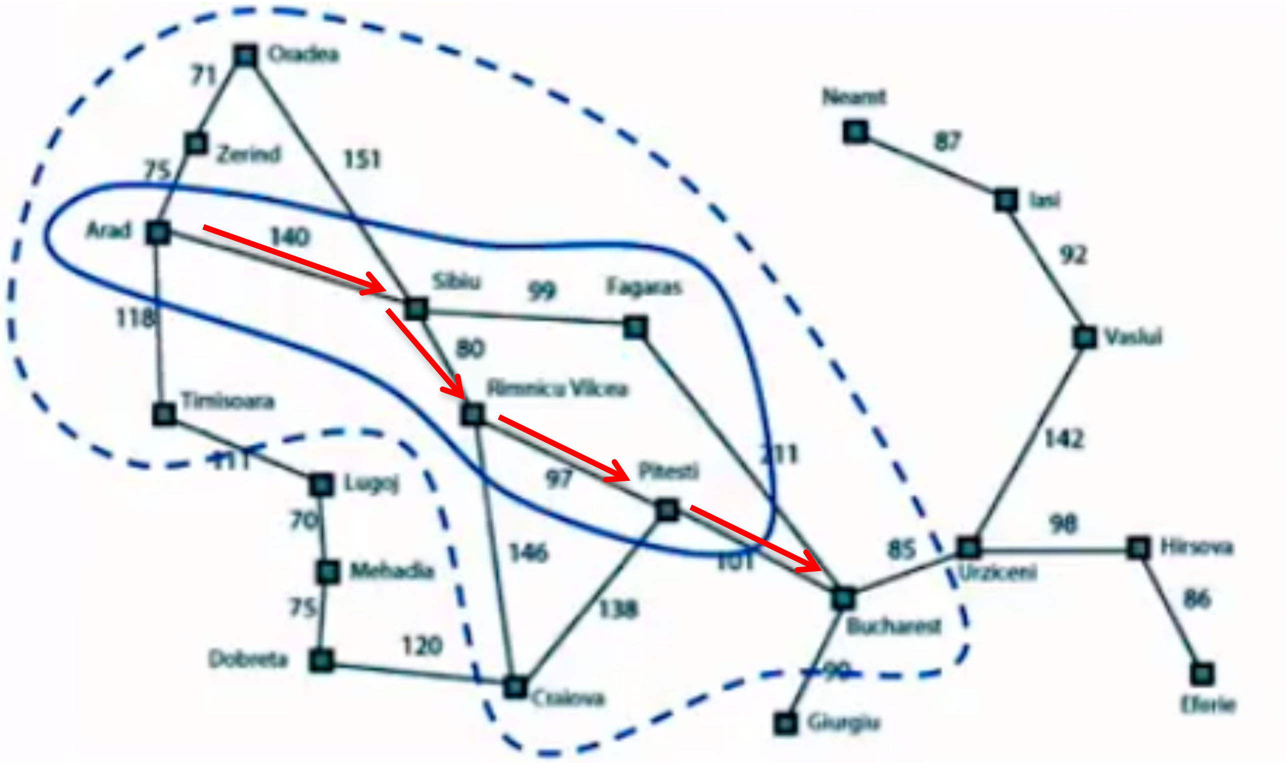
A∗ Search Example



A∗ Search Example



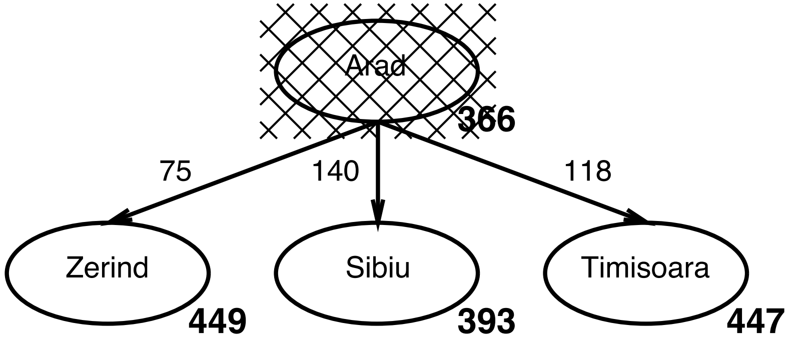
A∗ Search Example

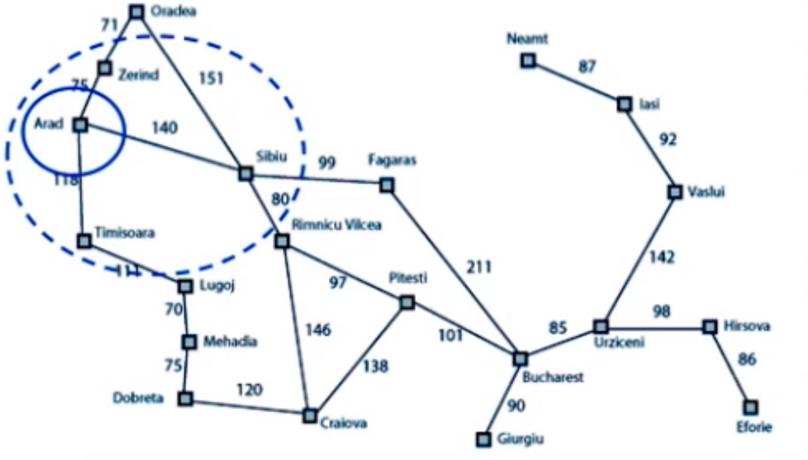


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Will stop finding the shorter path to Buchurest

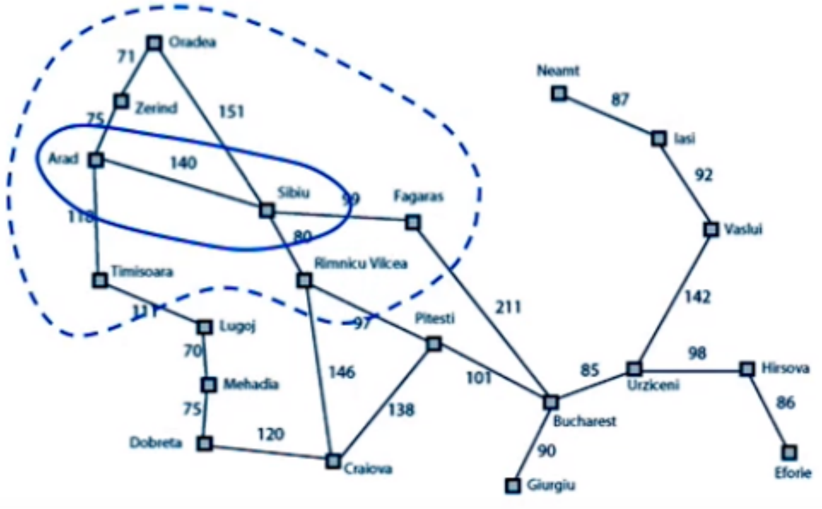
A∗ Search Example





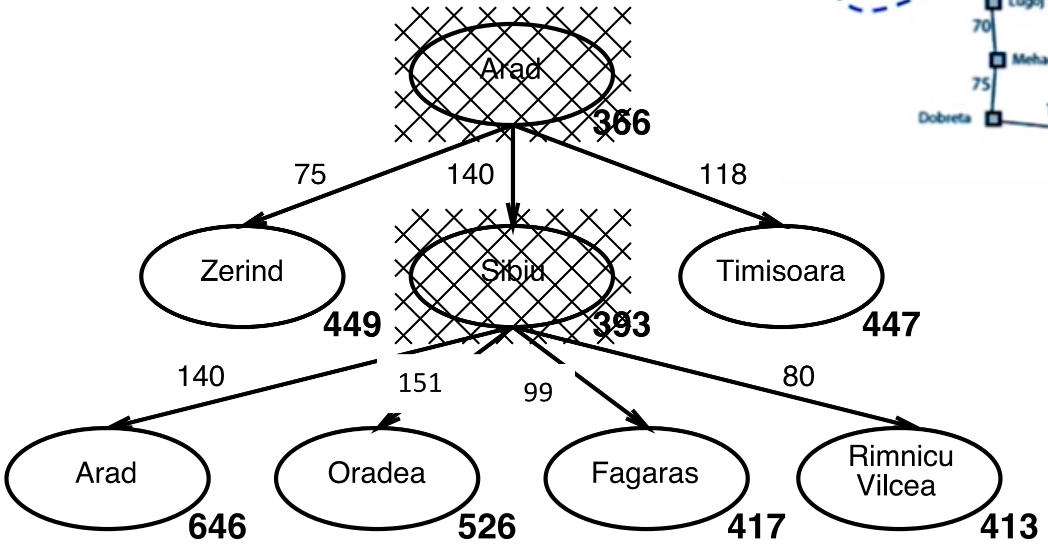
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A∗ Search Example

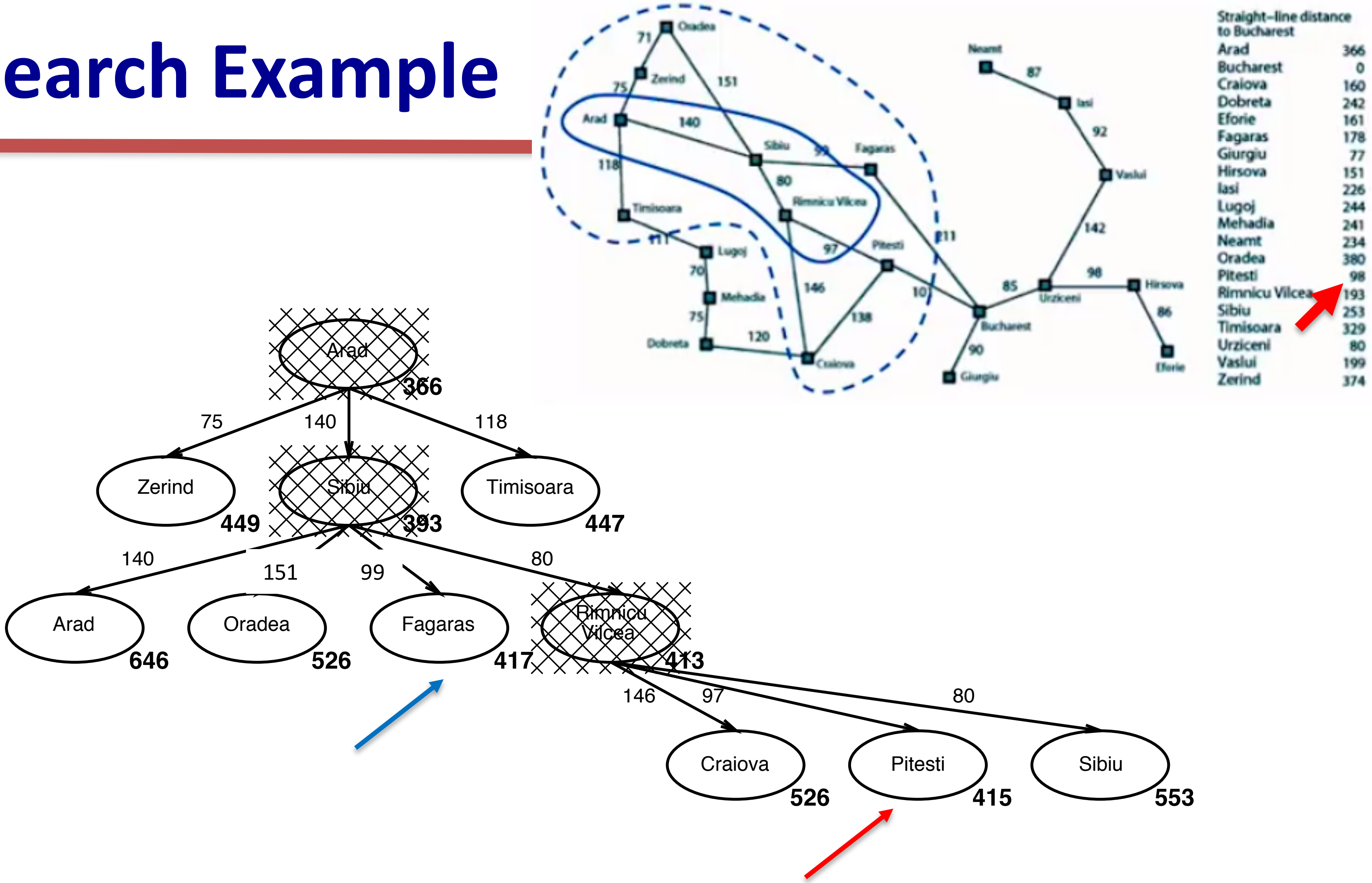


Straight–line distance to Bucharest

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



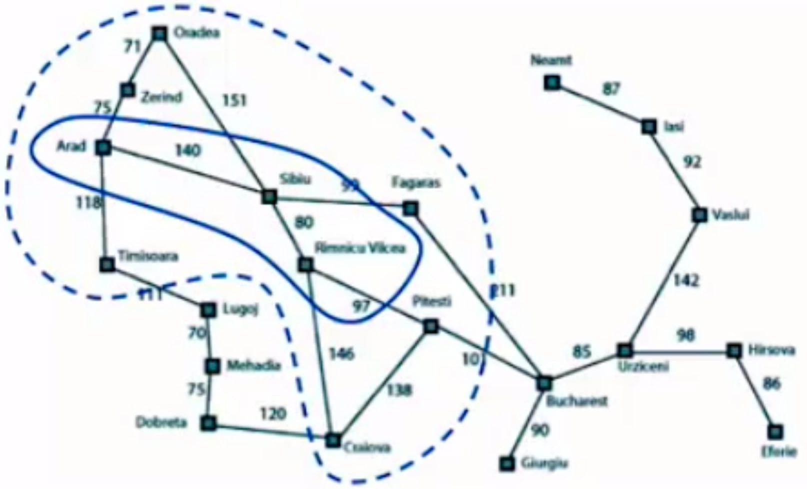
A∗ Search Example



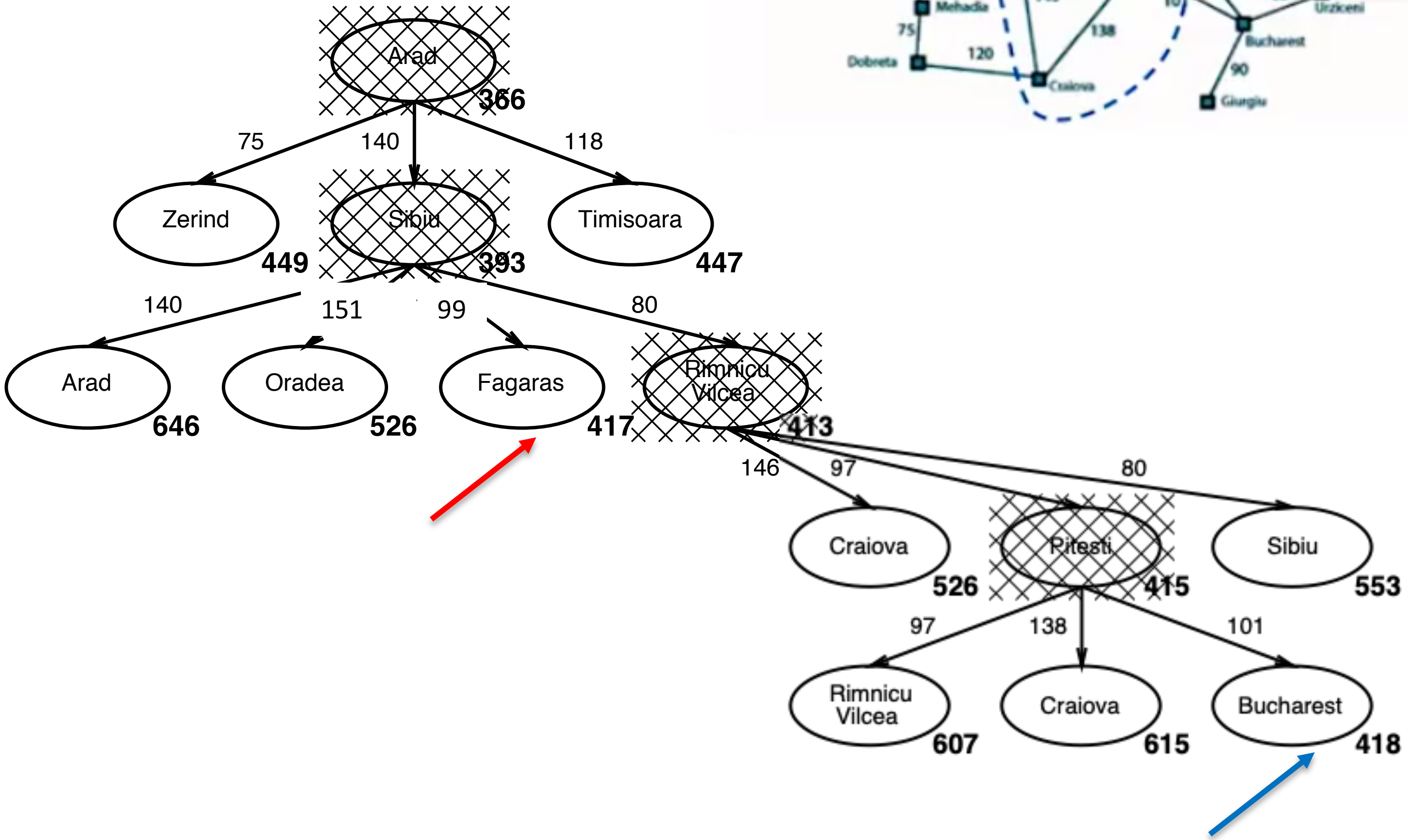
Straight–line distance to Bucharest

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

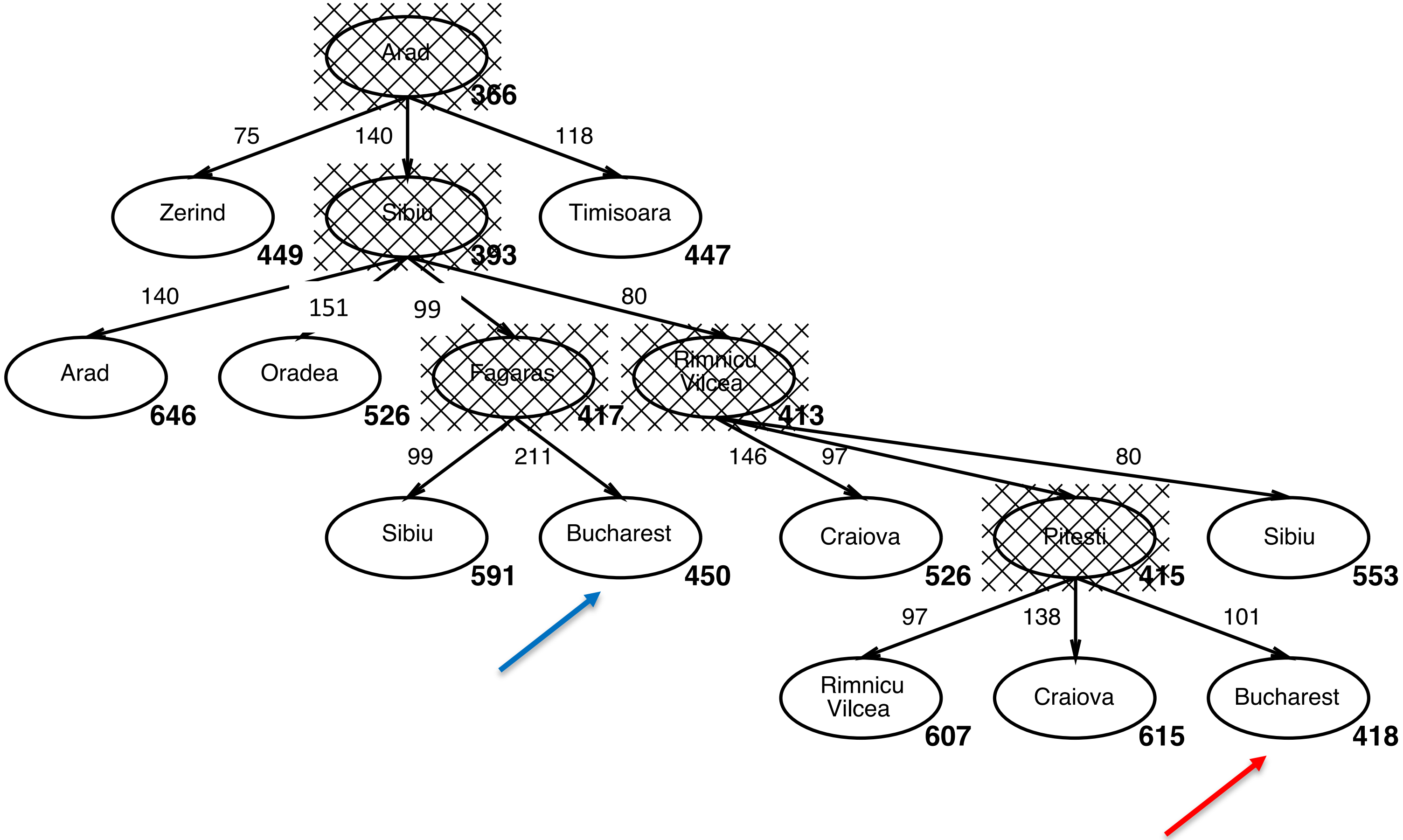
A∗ Search Example



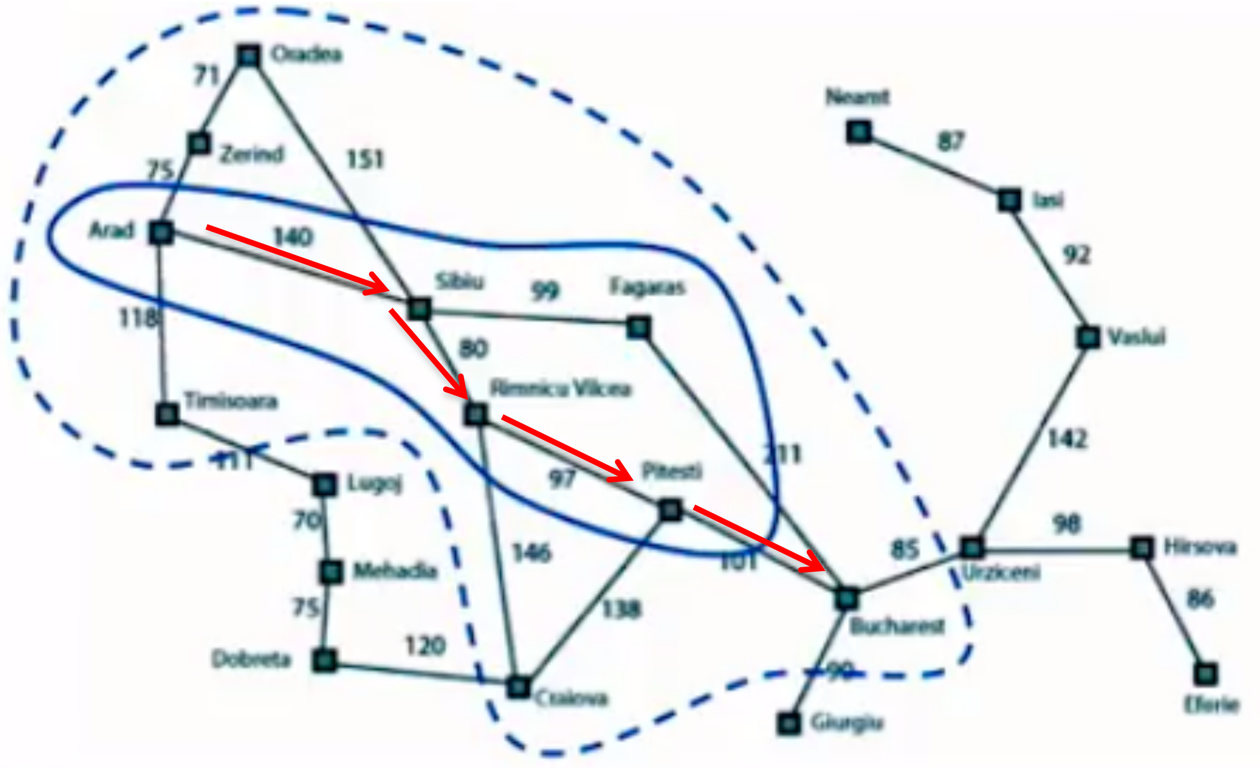
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



A∗ Search Example



A∗ Search Example



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

Will stop finding the shorter path to Buchurest

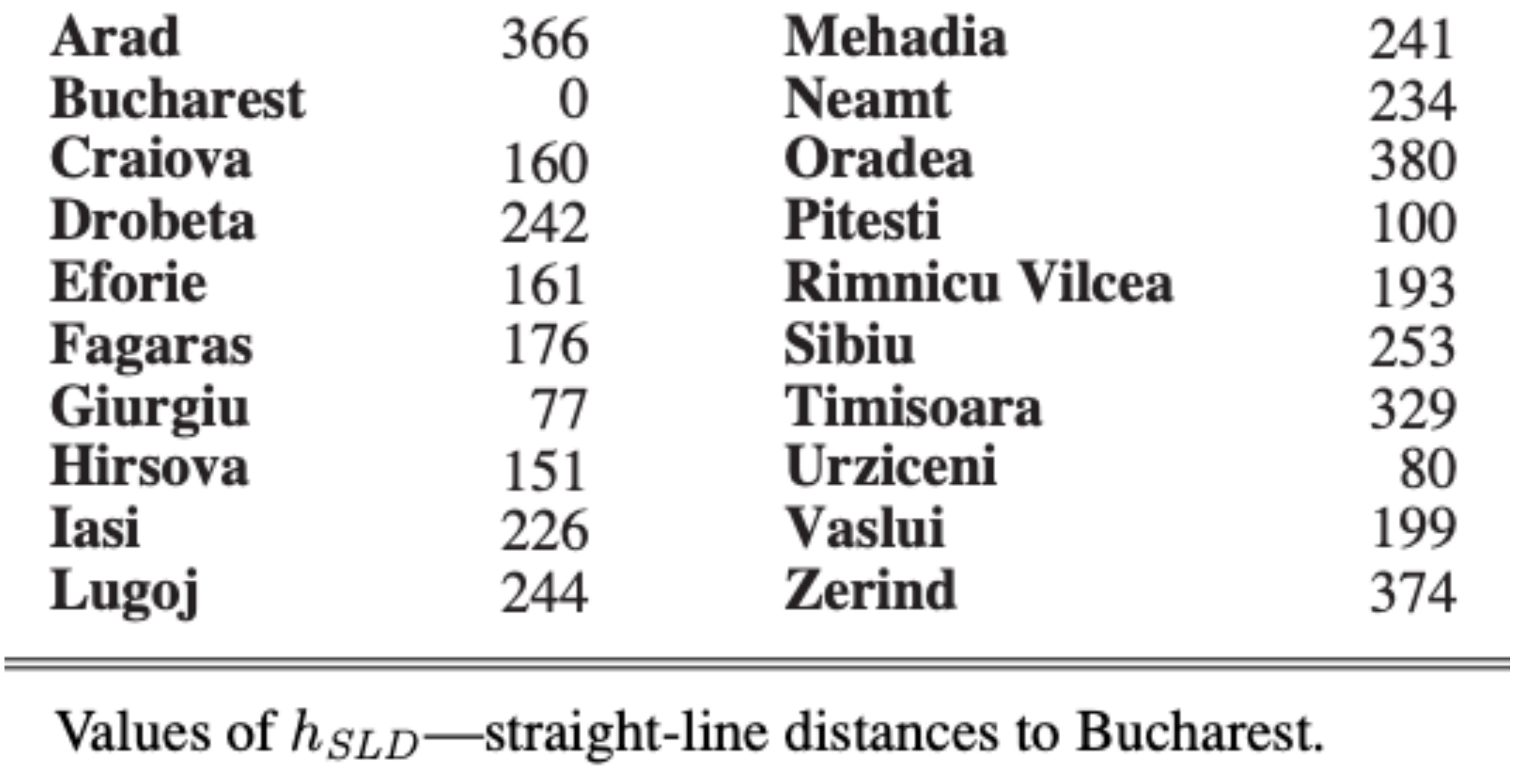
A* Search Example

A* Search Example with slightly different h values.

Russell & Norvig, *Artificial Intelligence: a Modern Approach*, Chapter 3, pp 83

A* Search Example

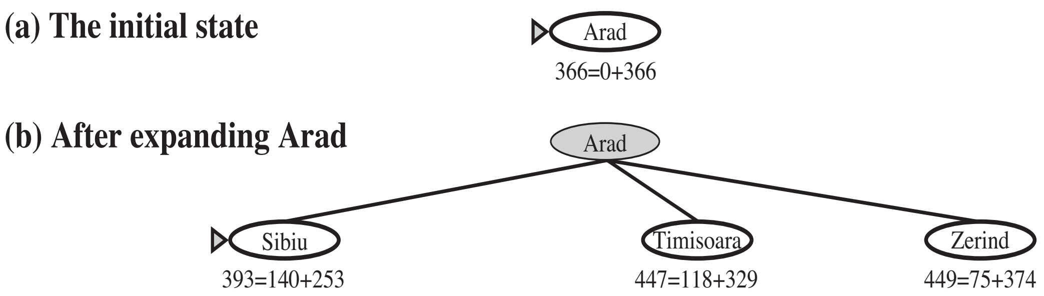
Different h_{LSD} values



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

Values of h_{SLD}—straight-line distances to Bucharest.

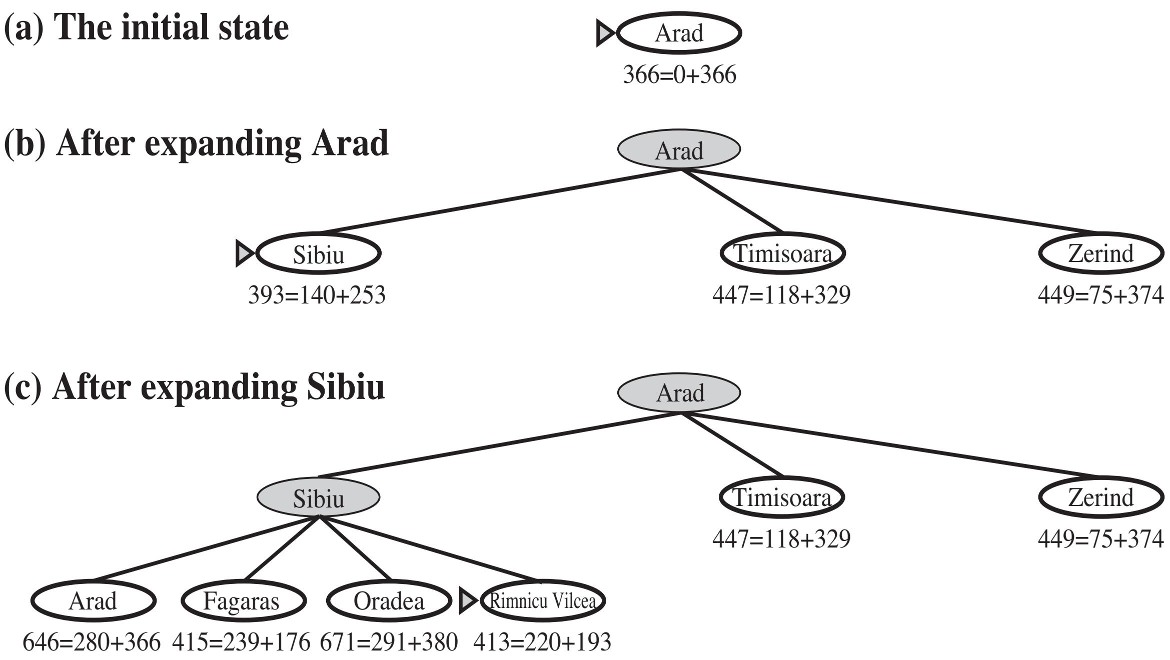
A* Search Example



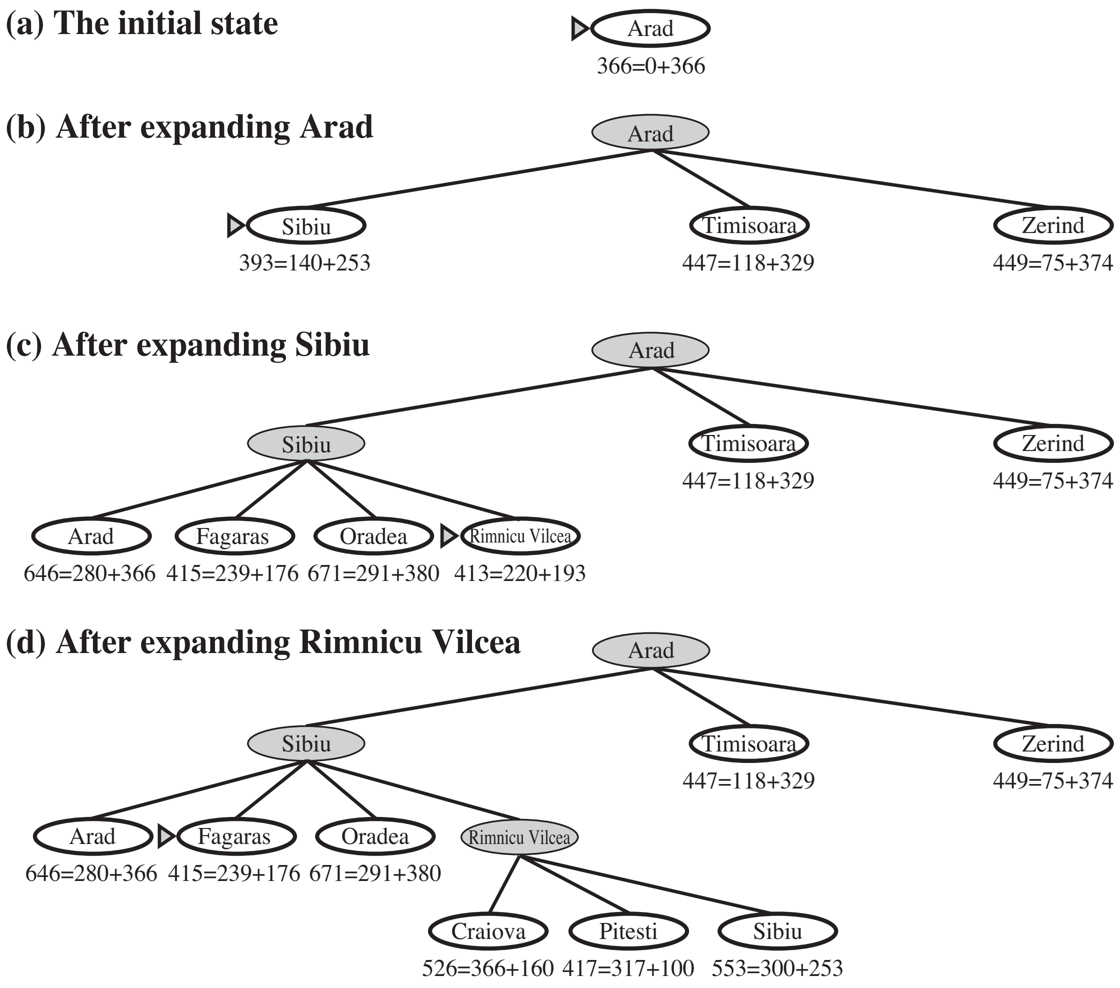
(a) The initial state

(b) After expanding Arad

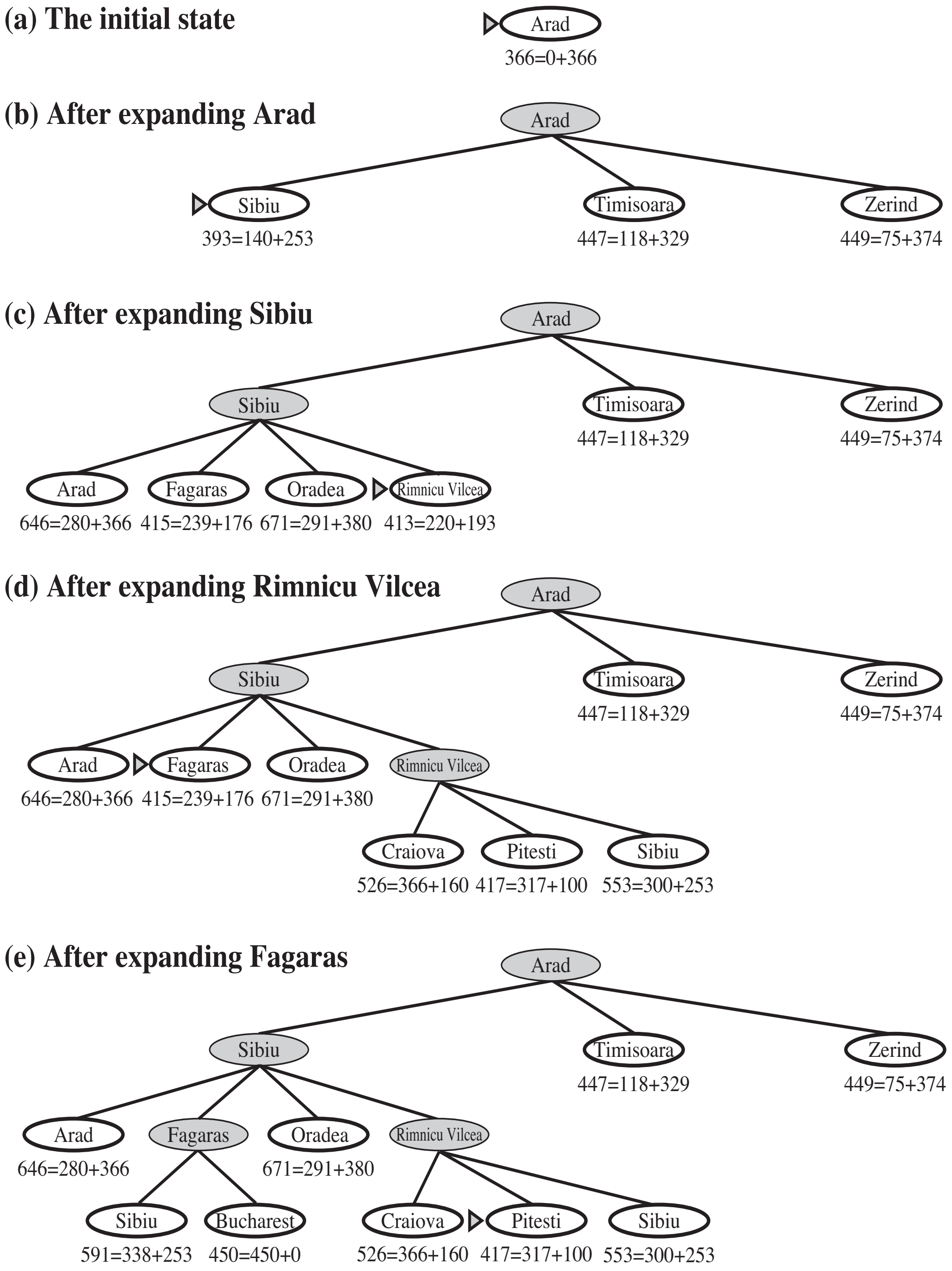
A* Search Example



A* Search Example



A* Search Example



(a) The initial state

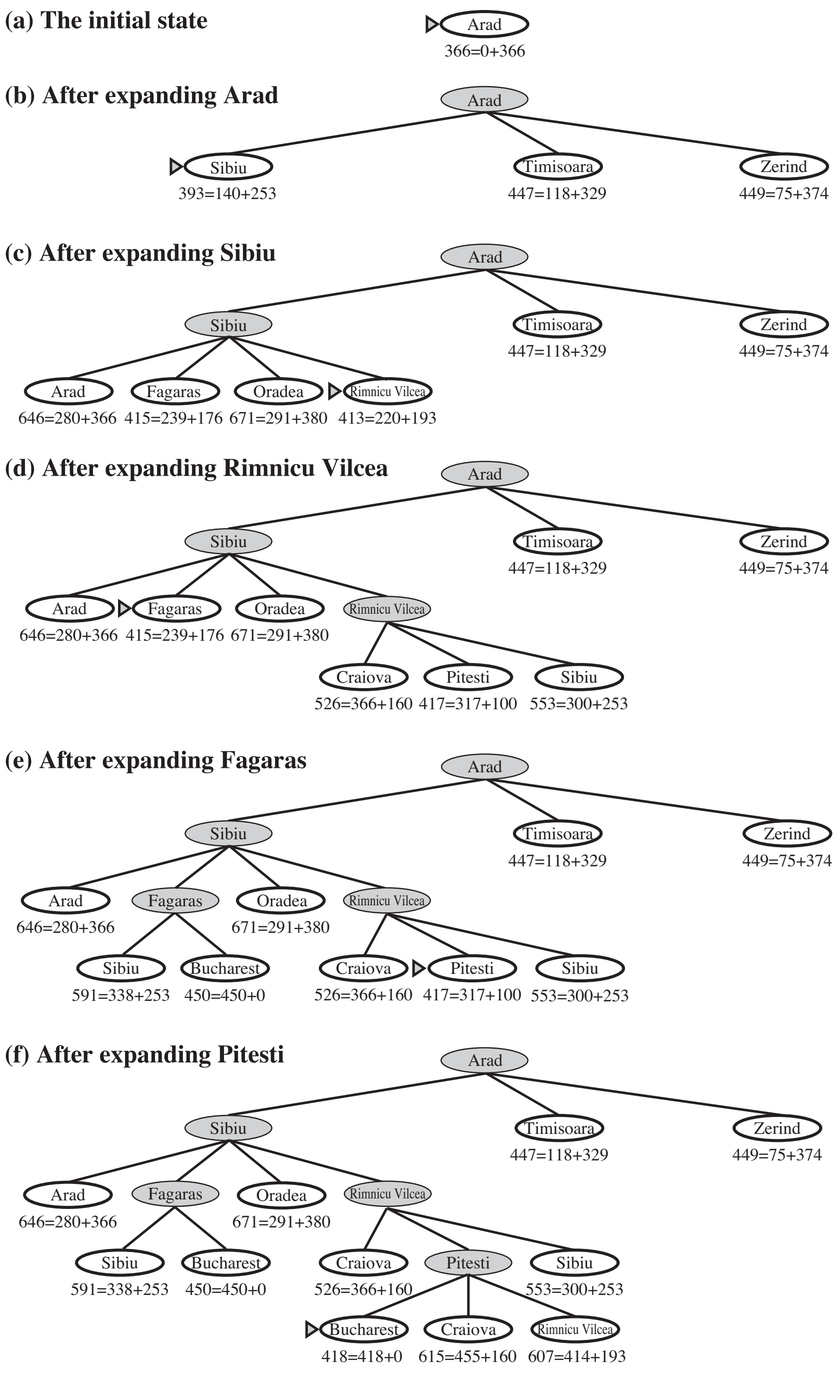
(b) After expanding Arad

(c) After expanding Sibiu

(d) After expanding Rimnicu Vilcea

(e) After expanding Fagaras

A* Search Example



(a) The initial state

(b) After expanding Arad

(c) After expanding Sibiu

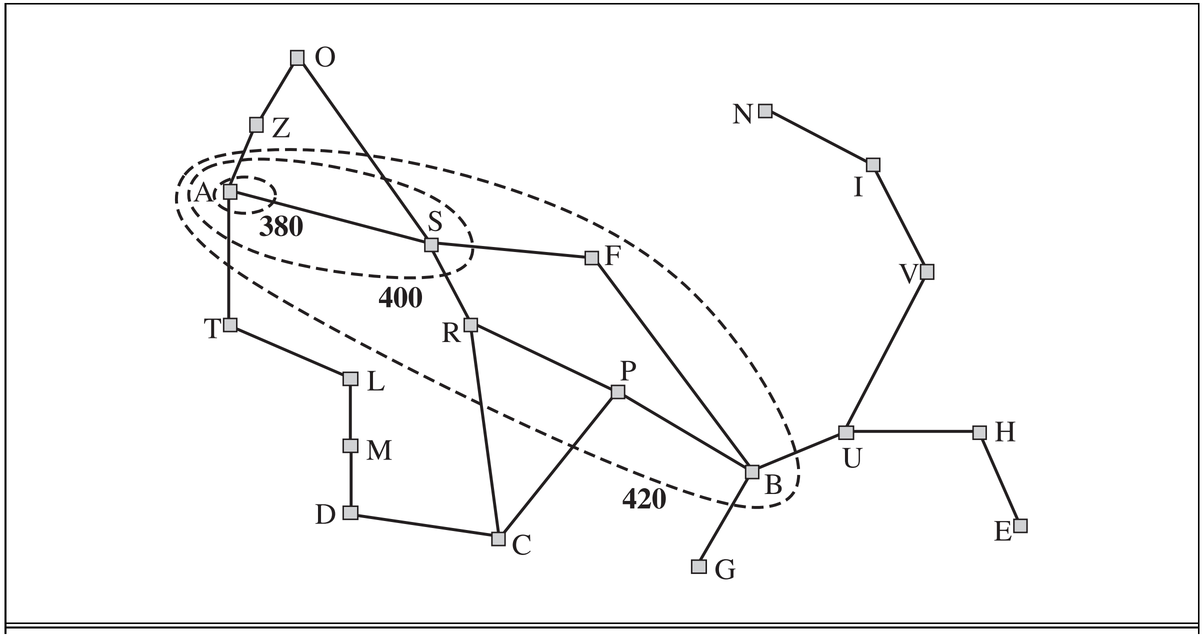
(d) After expanding Rimnicu Vilcea

(e) After expanding Fagaras

(f) After expanding Pitesti

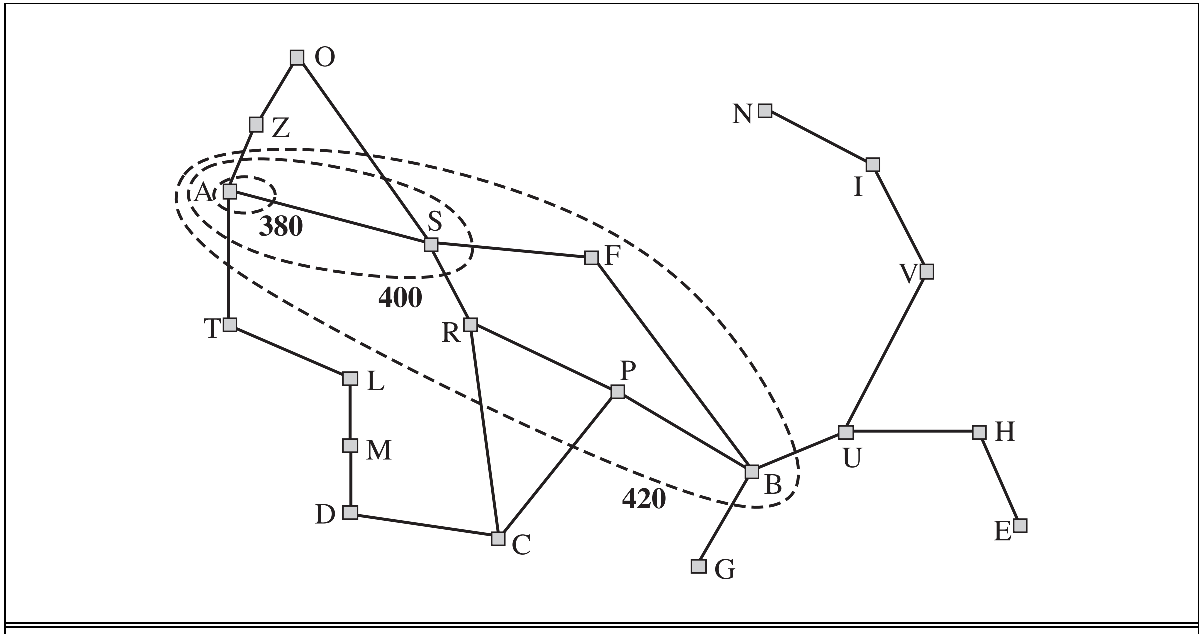
A* Search

Map of Romania showing contours at f = 380, f = 400, and f = 420, with Arad as the start state.
Nodes inside a given contour have f-costs less than or equal to the contour value.



A* Search

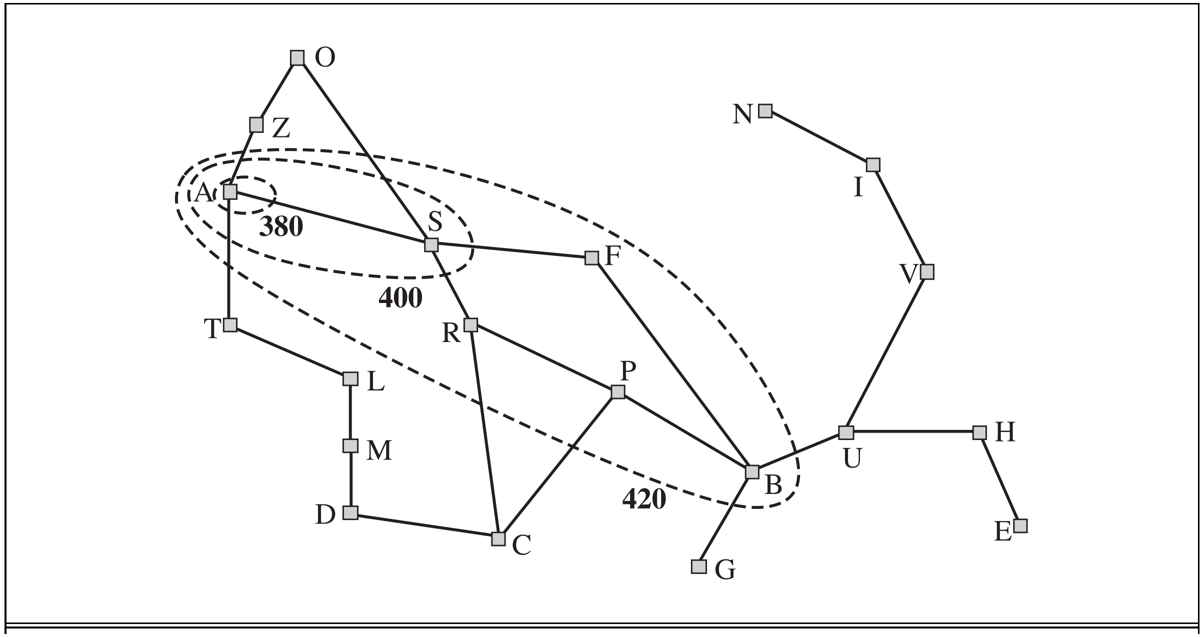
Map of Romania showing contours at f = 380, f = 400, and f = 420, with Arad as the start state. Nodes inside a given contour have f-costs less than or equal to the contour value.



Inside the contour labeled 400, all nodes have f(n) less than or equal to 400, and so on. Because A∗ expands the frontier node of lowest f-cost, we can see that an A∗ search fans out from the start node, adding nodes in concentric bands of increasing f-cost.

A* Search

Map of Romania showing contours at f = 380, f = 400, and f = 420, with Arad as the start state. Nodes inside a given contour have f-costs less than or equal to the contour value.



Inside the contour labeled 400, all nodes have f(n) less than or equal to 400, and so on. Because A∗ expands the frontier node of lowest f-cost, we can see that an A∗ search fans out from the start node, adding nodes in concentric bands of increasing f-cost.

With uniform-cost search (A∗ search using h(n) = 0), the bands will be “circular” around the start state. With more accurate heuristics, the bands will stretch toward the goal state and become more narrowly focused around the optimal path.

Summary

- **Uninformed search** methods have access only to the problem definition. The basic algorithms are as follows:
  - **Breadth-first search** expands the shallowest nodes first; it is complete, optimal for unit step costs, but has exponential space complexity.
  - **Uniform-cost search** expands the node with lowest path cost, g(n), and is optimal for general step costs.
  - **Depth-first search** expands the deepest unexpanded node first. It is neither complete nor optimal, but has linear space complexity.
  - **Depth-limited search** adds a depth bound.
  - **Iterative deepening search** calls depth-first search with increasing depth limits until a goal is found. It is complete, optimal for unit step costs, has time complexity comparable to breadth-first search, and has linear space complexity.
  - **Bidirectional search** can enormously reduce time complexity, but it is not always applicable and may require too much space.

Summary

- **Informed search** methods may have access to a **heuristic** function h(n) that estimates the cost of a solution from n.
  - The generic **best-first search** algorithm selects a node for expansion according to an **evaluation function**.
  - **Greedy best-first search** expands nodes with minimal h(n). It is not optimal but is often efficient.
  - **A∗ search** expands nodes with minimal f (n) = g(n) + h(n). A∗ is complete and optimal, provided that h(n) is admissible (for TREE-SEARCH) or consistent (for GRAPH-SEARCH). The space complexity of A∗ is still prohibitive. (**RBFS** (recursive best-first search))
- The performance of heuristic search algorithms depends on the quality of the heuristic function. One can sometimes construct good heuristics by relaxing the problem definition, by storing precomputed solution costs for subproblems in a pattern database, or by learning from experience with the problem class.

References

- ❑ Russell & Norvig, *Artificial Intelligence: a Modern Approach*, Chapter 3.