

Today

- ▶ Informed Search
- ▶ A*, Heuristics

Overview

- Informed search strategies
 - Heuristic functions
 - Local search and optimization
 - Local search in continuous spaces
 - Searching with nondeterministic actions
 - Searching with partial observations
 - Online search agents and unknown environments
-

Graph search

function GRAPH-SEARCH(*problem*, *fringe*) **returns** a solution, or failure

closed ← an empty set

fringe ← INSERT(MAKE-NODE(INITIAL-STATE[*problem*]), *fringe*)

loop do

if *fringe* is empty **then return** failure

node ← REMOVE-FRONT(*fringe*)

if GOAL-TEST(*problem*, STATE[*node*]) **then return** *node*

if STATE[*node*] is not in *closed* **then**

 add STATE[*node*] to *closed*

fringe ← INSERTALL(EXPAND(*node*, *problem*), *fringe*)

end

Best-first search

Idea: use an **evaluation function** for each node
– estimate of “desirability”

⇒ Expand most desirable unexpanded node

Implementation:

fringe is a queue sorted in decreasing order of desirability

Special cases:

greedy search

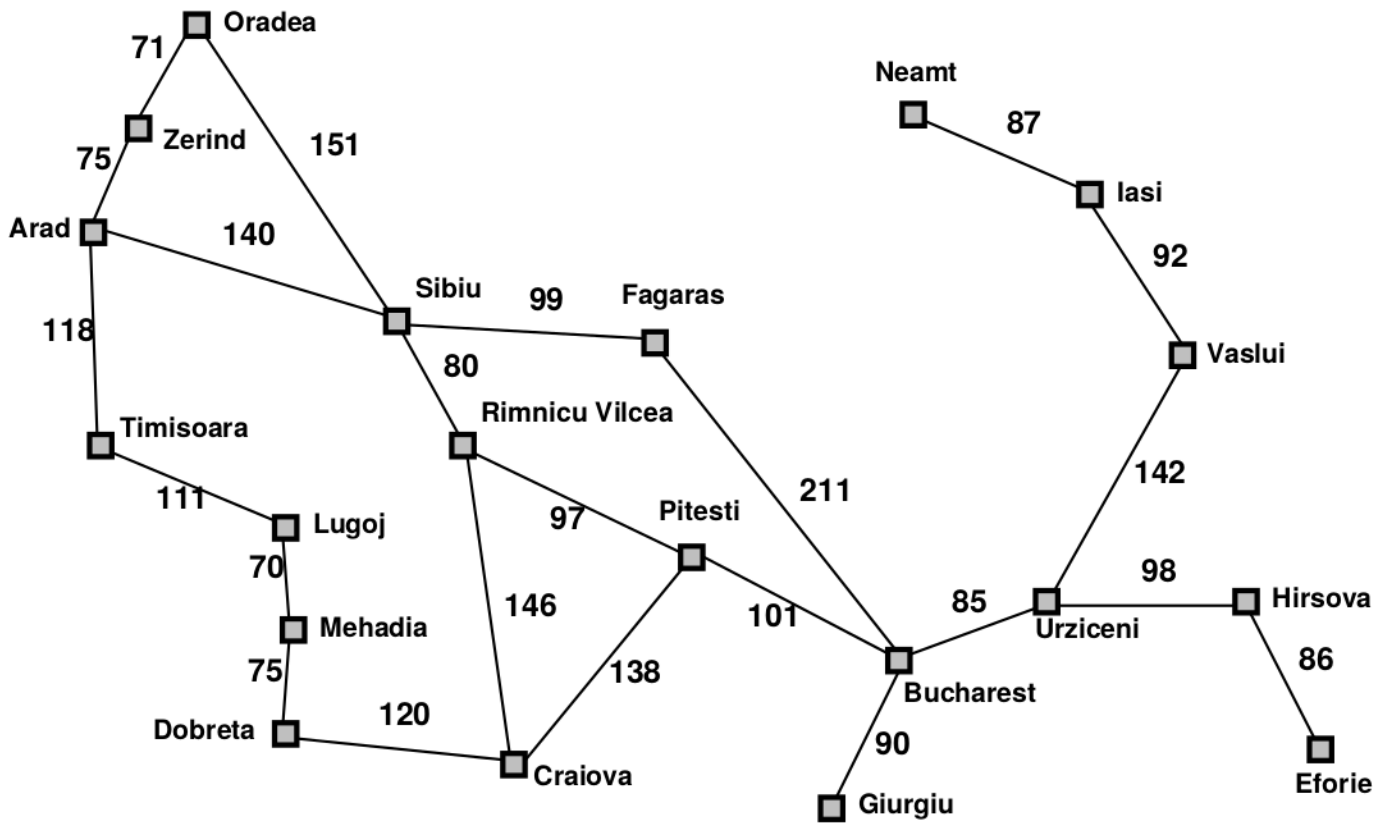
A* search

f(n) : evaluation function

lowest evaluation is selected for expansion

maintain fringe in the ascending order of f-values.

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

Greedy search

Evaluation function $h(n)$ (**h**euristic)

= estimate of cost from n to the closest goal
^ of the cheapest path

E.g., $h_{\text{SLD}}(n)$ = straight-line distance from n to Bucharest

Greedy search expands the node that **appears** to be closest to goal

if $f(n) = h(n)$ --> greedy search

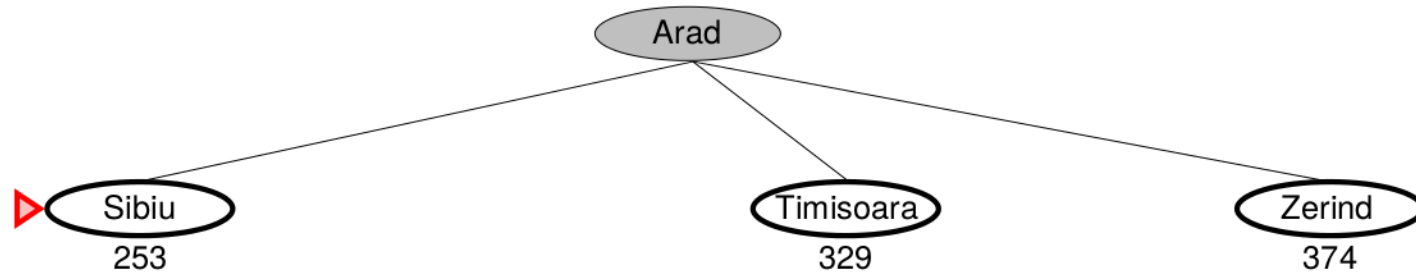
Greedy search example



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.

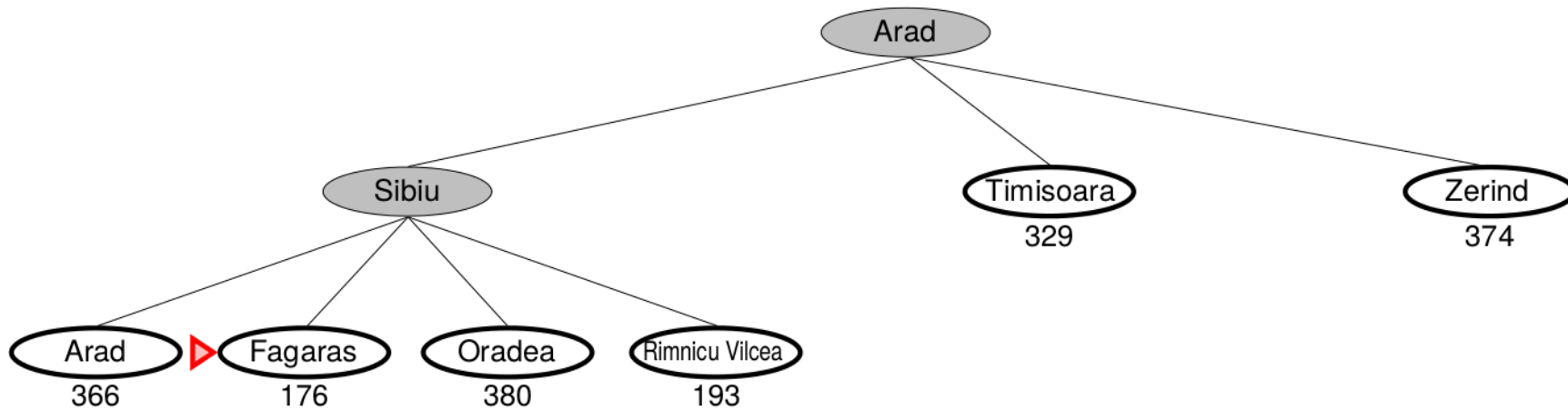
Greedy search example



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.

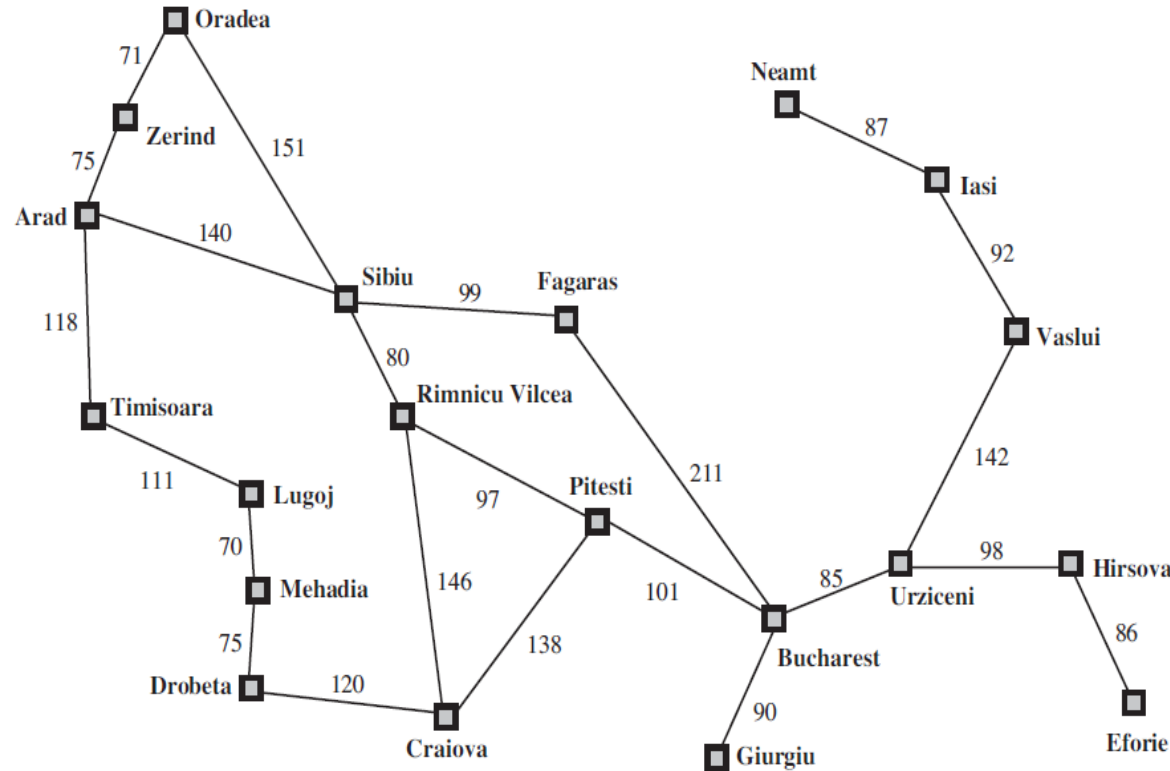
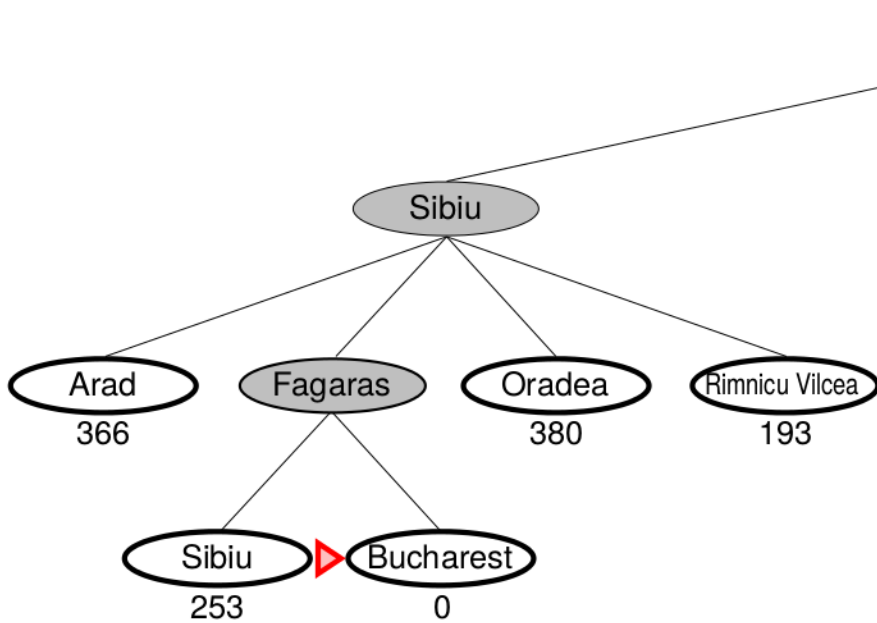
Greedy search example



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.

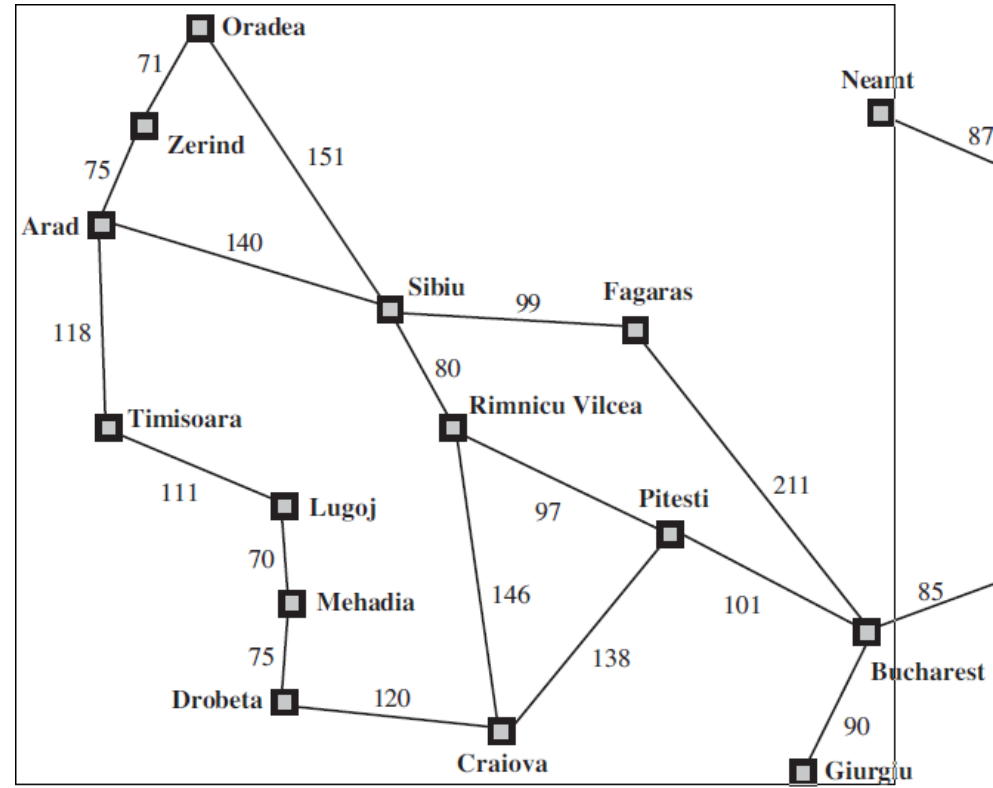
Greedy search example



Optimal?
Greedy...
false starts? from Iasi to Fagaras

Properties of greedy search

Complete??



Properties of greedy search

Complete?? No—can get stuck in loops, e.g., with Oradea as goal,

lasi → Neamt → lasi → Neamt →

Complete in finite space with repeated-state checking

Time??

Properties of greedy search

Complete?? No—can get stuck in loops, e.g.,

lasi → Neamt → lasi → Neamt →

Complete in finite space with repeated-state checking

Time?? $O(b^m)$, but a good heuristic can give dramatic improvement

Space??

Properties of greedy search

Complete?? No—can get stuck in loops, e.g.,

lasi → Neamt → lasi → Neamt →

Complete in finite space with repeated-state checking

Time?? $O(b^m)$, but a good heuristic can give dramatic improvement

Space?? $O(b^m)$ —keeps all nodes in memory

Optimal??

Properties of greedy search

Complete?? No—can get stuck in loops, e.g.,

lasi → Neamt → lasi → Neamt →

Complete in finite space with repeated-state checking

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

Idea: avoid expanding paths that are already expensive

Evaluation function $f(n) = g(n) + h(n)$

$g(n)$ = cost so far to reach n -- estimated cost from start to n

$h(n)$ = estimated cost to goal from n

$f(n)$ = estimated total cost of path through n to goal

A* search uses an **admissible** heuristic

i.e., $h(n) \leq h^*(n)$ where $h^*(n)$ is the **true** cost from n .

(Also require $h(n) \geq 0$, so $h(G) = 0$ for any goal G .)

E.g., $h_{\text{SLD}}(n)$ never overestimates the actual road distance

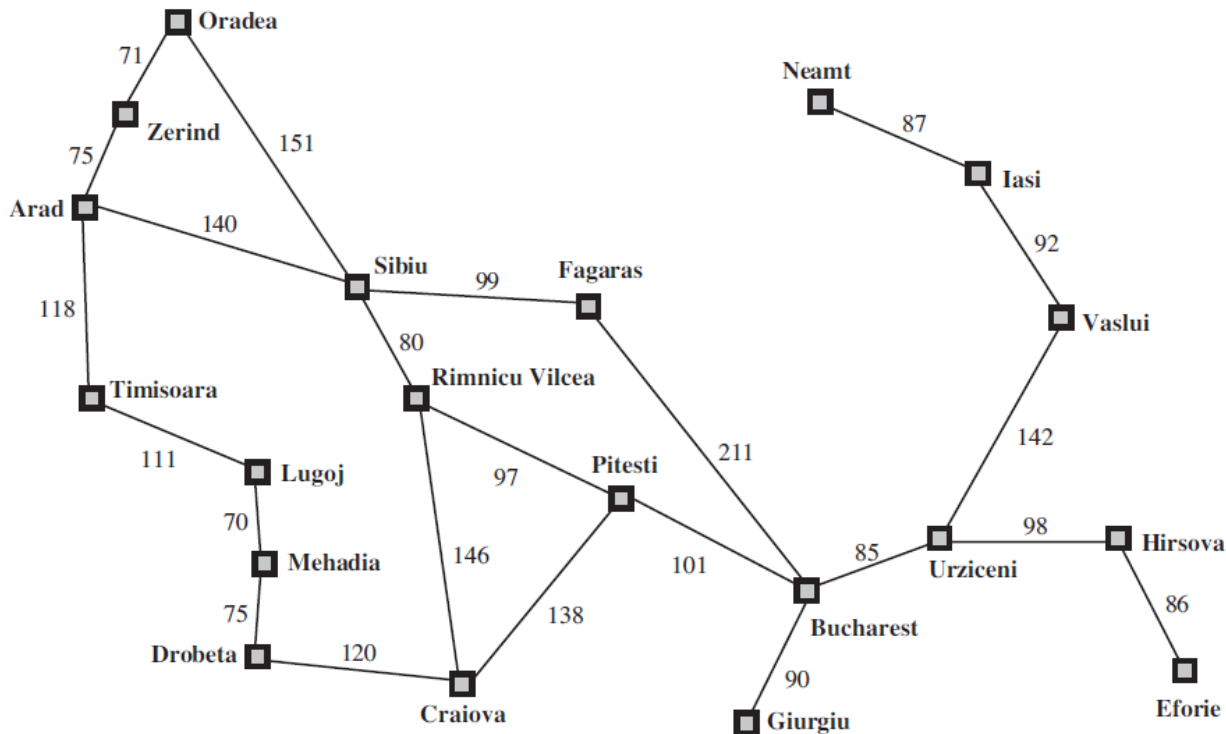
Theorem: A* search is optimal

$g(n)$ is the actual cost:

--> $f(n)$ never overestimates the cost

A* search example

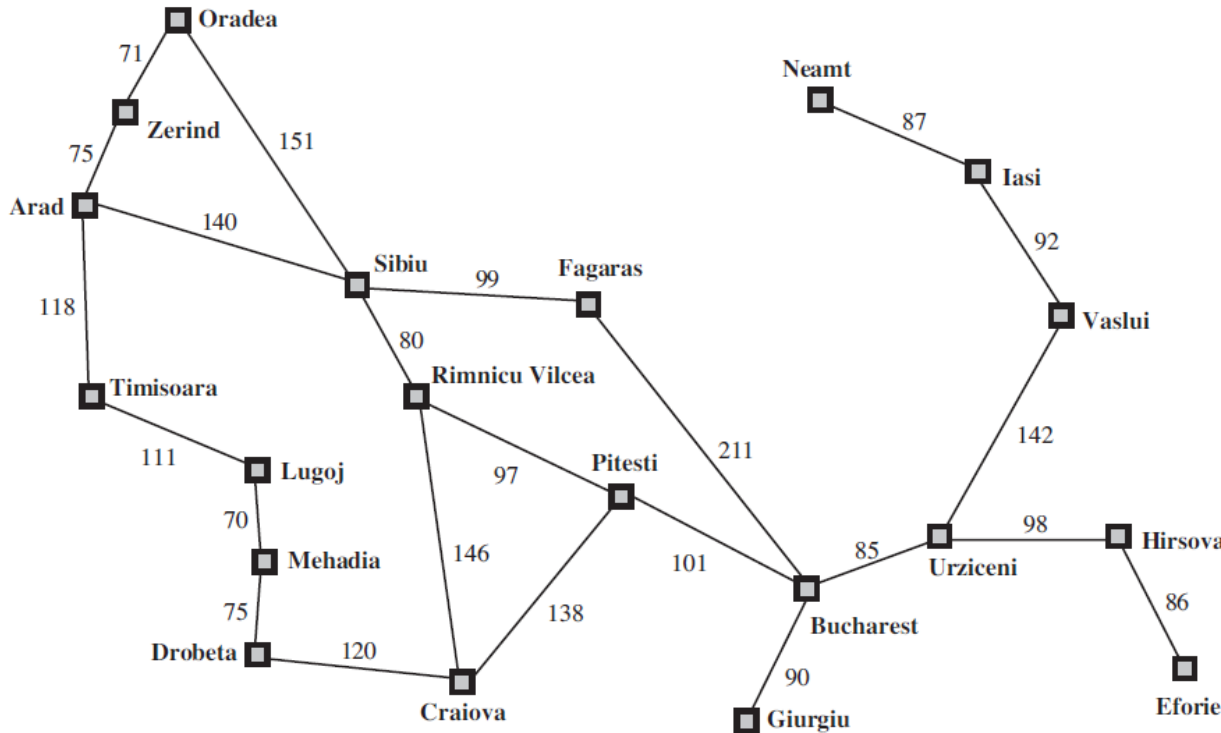
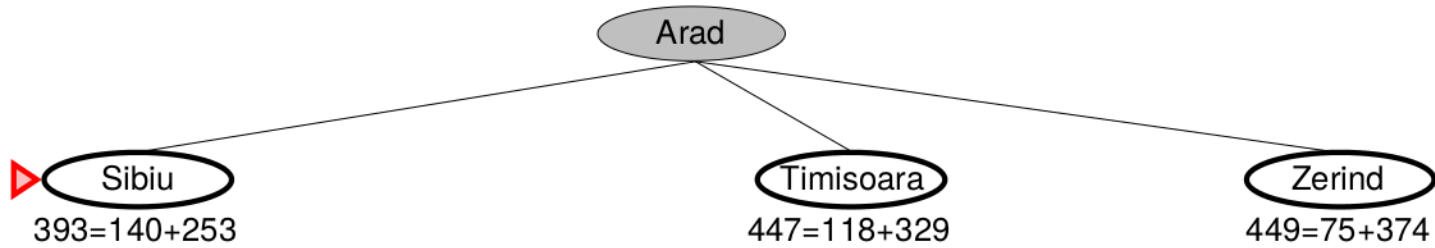
▶ Arad
 $366 = 0 + 366$



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.

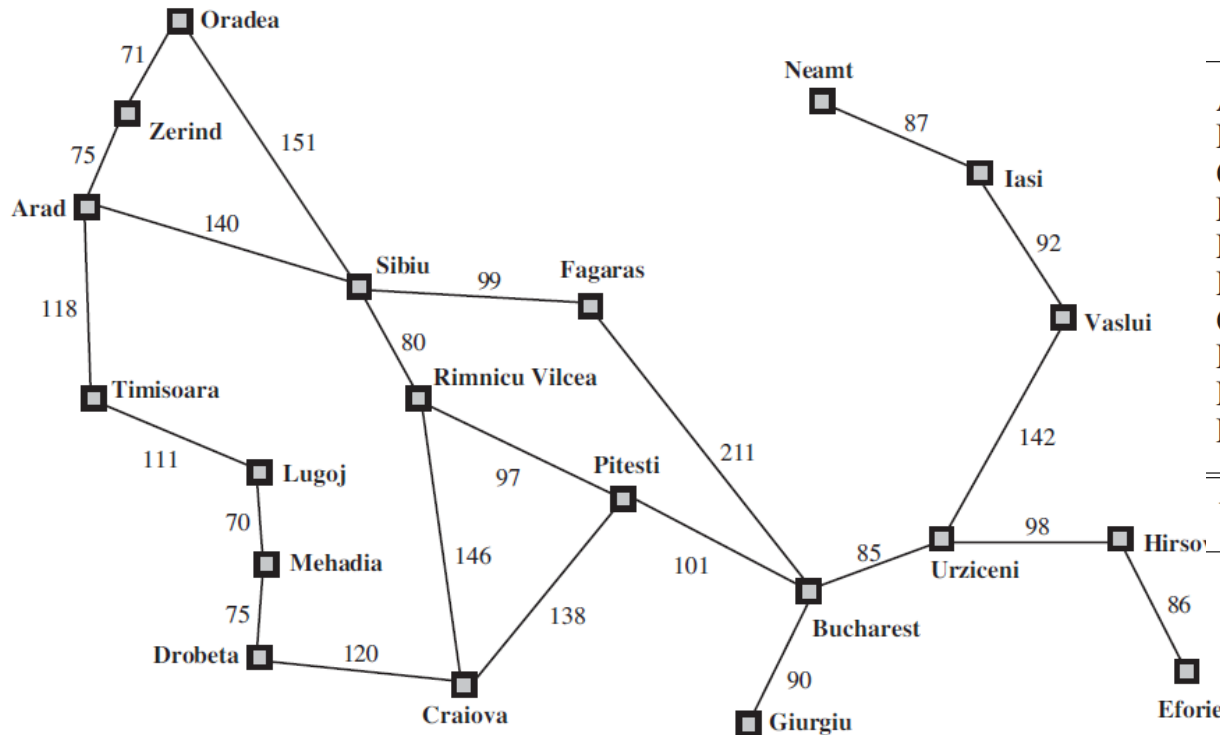
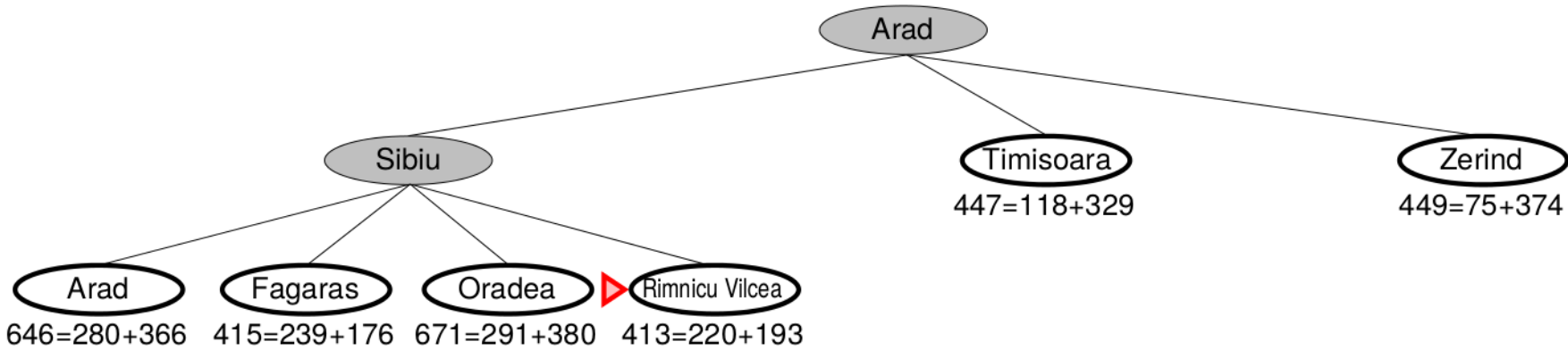
A* search example



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.

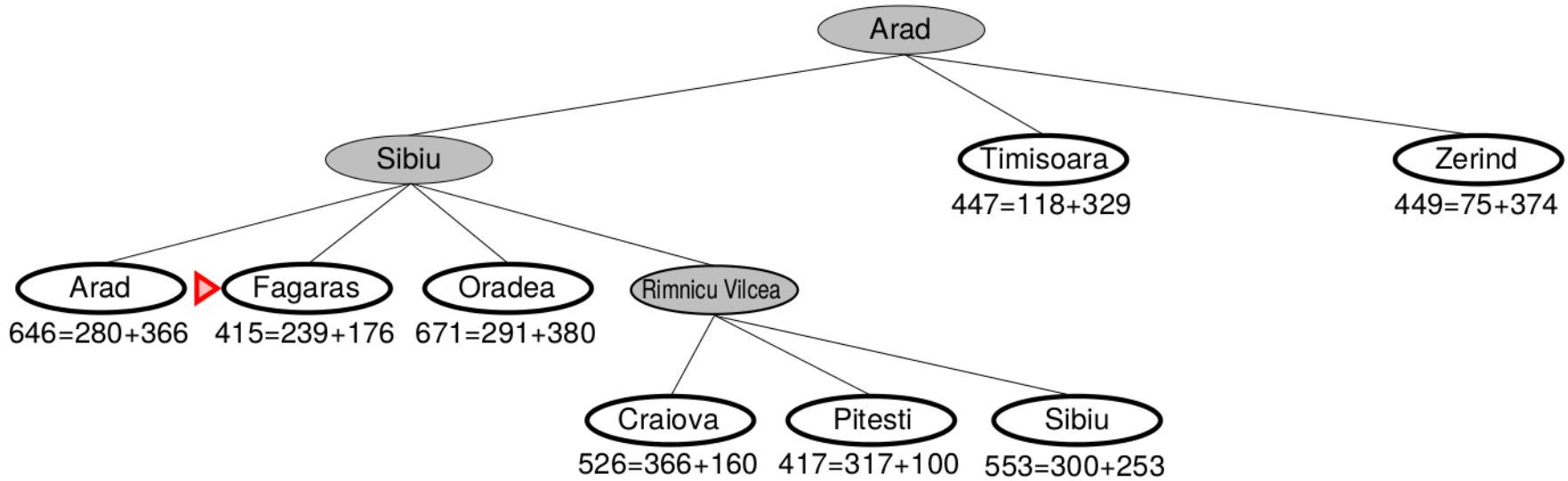
A* search example



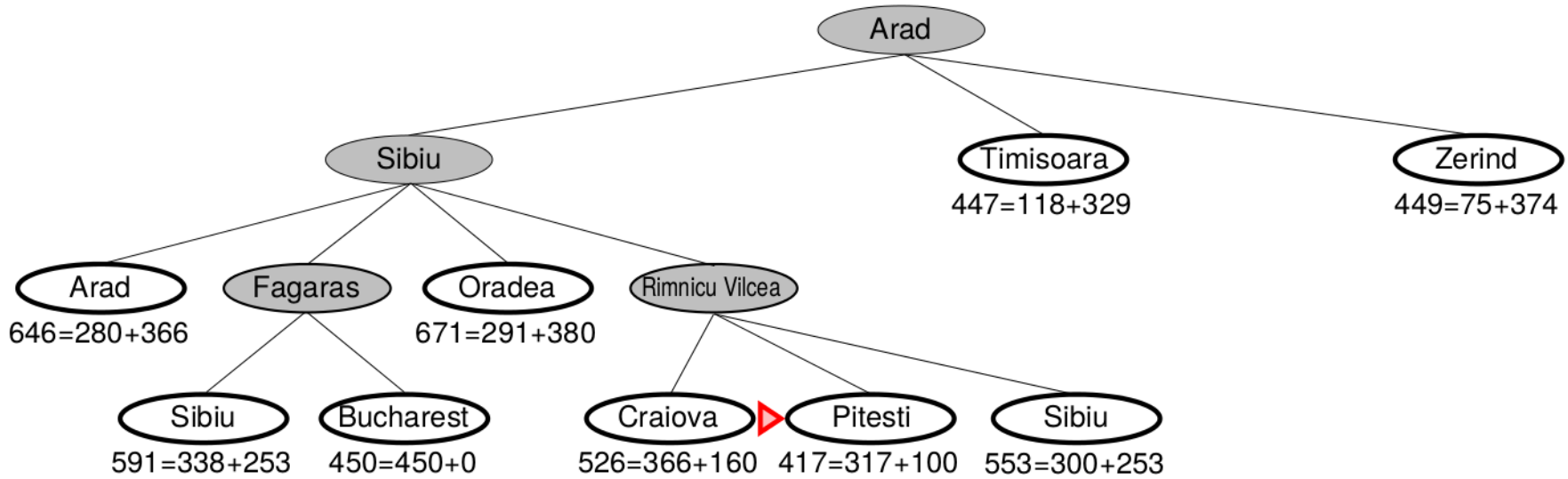
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

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

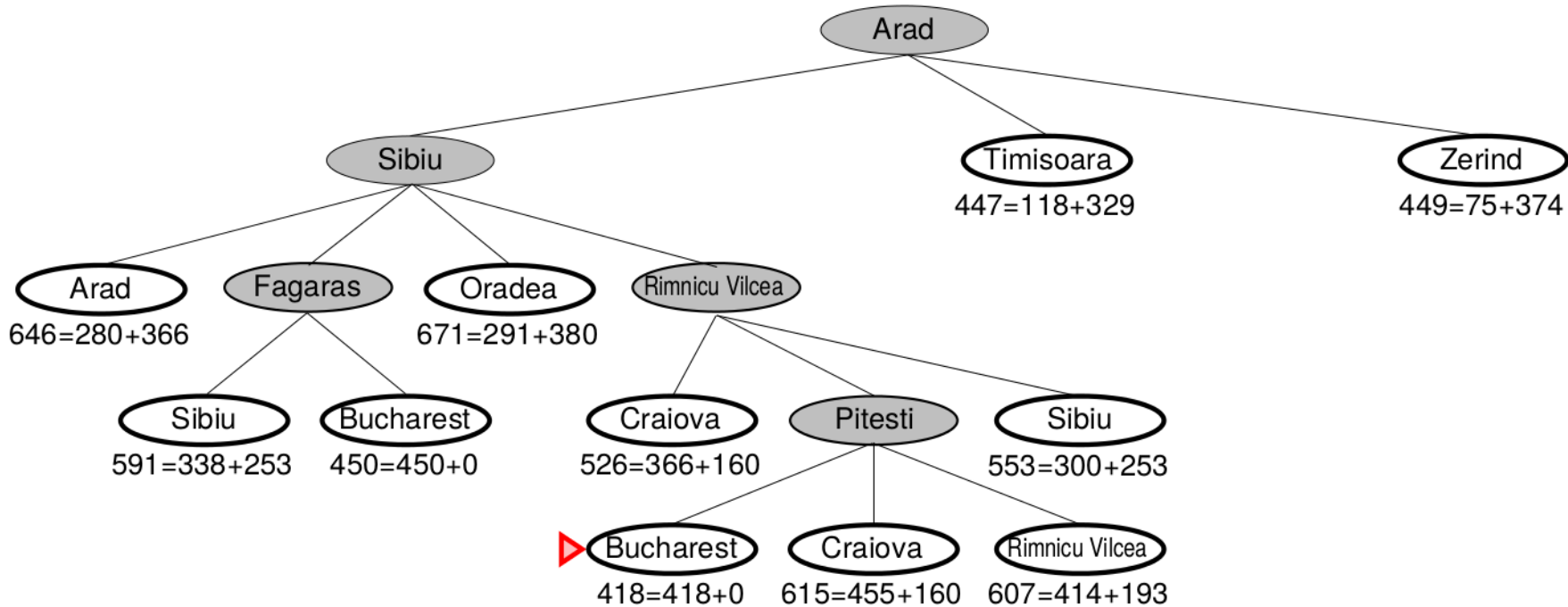
A* search example



A* search example



A* search example



Optimal? Yes if heuristic is

1- admissible ... never overestimates: $h(n) \leq h^*(n)$

2- consistency... $h(n) < h(n') + c(n,a,n')$

Every consistent heuristic is admissible.

* Admissible but inconsistent heuristics require bookkeeping to ensure optimality.

Properties of A^*

Complete??

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time??

Properties of A^*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time?? Exponential in [relative error in $h \times$ length of soln.]

Space??

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time?? Exponential in [relative error in $h \times$ length of soln.]

Space?? Keeps all nodes in memory

Optimal??

Properties of A*

Complete?? Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time?? Exponential in [relative error in $h \times$ length of soln.]

Space?? Keeps all nodes in memory

Optimal?? Yes—cannot expand f_{i+1} until f_i is finished

A* expands all nodes with $f(n) < C^*$

A* expands some nodes with $f(n) = C^*$

A* expands no nodes with $f(n) > C^*$

Admissible heuristics

E.g., for the 8-puzzle:

$h_1(n)$ = number of misplaced tiles

$h_2(n)$ = total **Manhattan** distance

(i.e., no. of squares from desired location of each tile)

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

$h_1(S) = ??$

$h_2(S) = ??$

Admissible heuristics

E.g., for the 8-puzzle:

$h_1(n)$ = number of misplaced tiles

$h_2(n)$ = total **Manhattan** distance

(i.e., no. of squares from desired location of each tile)

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

$h_1(S) = ??$ 6

$h_2(S) = ??$ $4+0+3+3+1+0+2+1 = 14$

Dominance

If $h_2(n) \geq h_1(n)$ for all n (both admissible)
then h_2 dominates h_1 and is better for search

Typical search costs:

$d = 14$ IDS = 3,473,941 nodes

$A^*(h_1) = 539$ nodes

$A^*(h_2) = 113$ nodes

$d = 24$ IDS \approx 54,000,000,000 nodes

$A^*(h_1) = 39,135$ nodes

$A^*(h_2) = 1,641$ nodes

every node expanded by h_2 will also
be expanded by h_1 .

Given any admissible heuristics h_a, h_b ,

$$h(n) = \max(h_a(n), h_b(n))$$

is also admissible and dominates h_a, h_b

QUESTION: HOW TO INVENT ADMISSIBLE HEURISTICS?

Relaxed problems

Admissible heuristics can be derived from the **exact** solution cost of a **relaxed** version of the problem

If the rules of the 8-puzzle are relaxed so that a tile can move **anywhere**, then $h_1(n)$ gives the shortest solution

If the rules are relaxed so that a tile can move to **any adjacent square**, then $h_2(n)$ gives the shortest solution

Key point: the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem.

eg. traveling salesman problem