INFORMED SEARCH ALGORITHMS

Chapter 3

Best-first search

- Informed search algorithms

  use problem-specific knowledge to speed up the search process

Outline

- Best-first search
  - **Greedy** best-first search
  - **A*** search
- Heuristics

Review: Tree search algorithms

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

initialize the frontier using the initial state of problem

loop do

if the frontier is empty **then return** failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the corresponding solution

expand the chosen node, adding the resulting nodes to the frontier

A search strategy is defined by picking the order of node expansion

Best-first search

- Informed search algorithms

  use problem-specific knowledge to speed up the search process

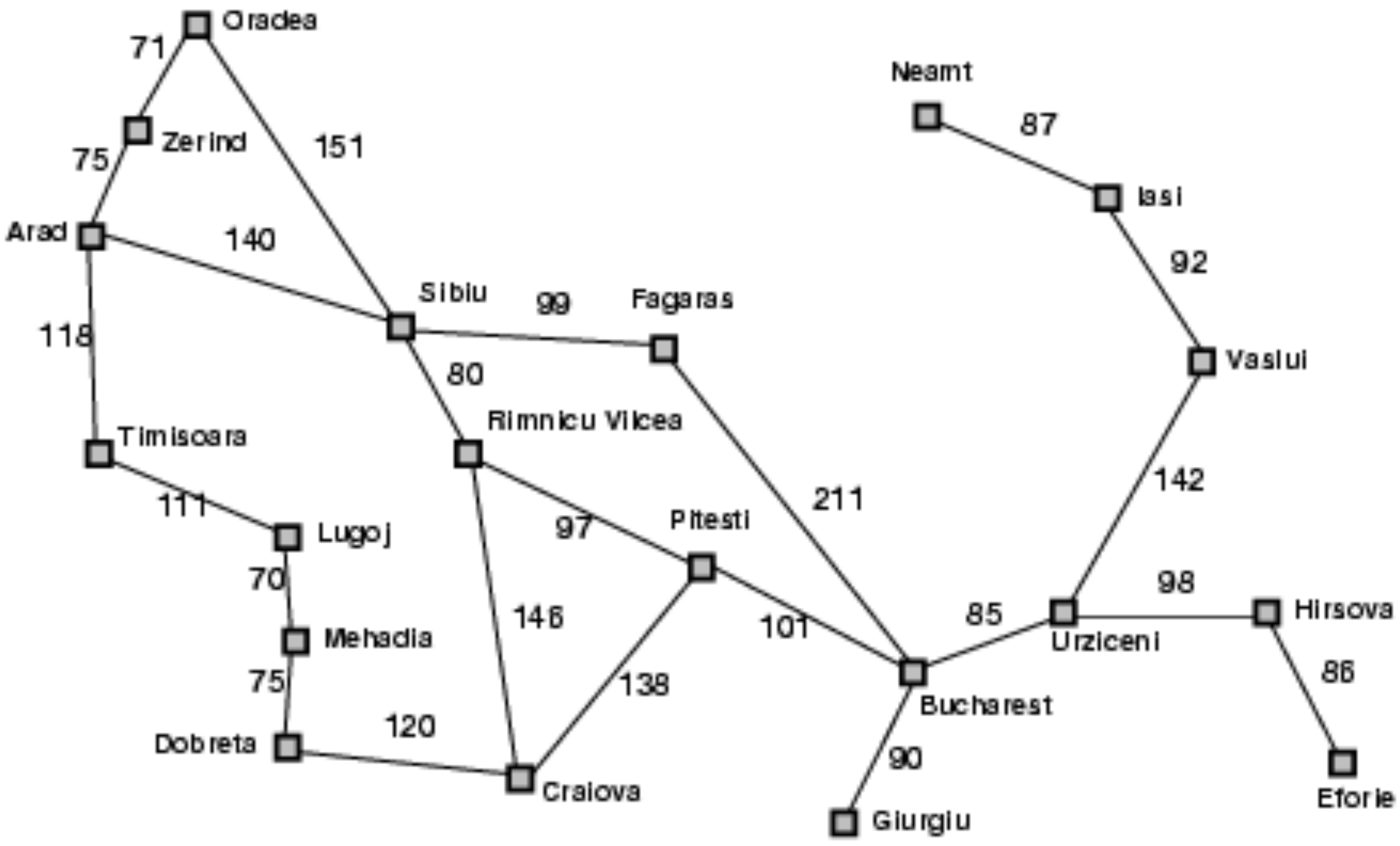
- Idea: use an **evaluation function f(n)** for each node
  - Estimate of "desiderability"
  - to choose the node selected for expansion
  - is construed as a cost estimate
  - Expand node with the lowest f(n) first

- Special cases
  - **Greedy** best-first search
  - $\mathbf{A}^*$ search

Best-first search

- Most algorithms use a **heuristic function** h(n) as a component of f
  - **h(n) =** __**estimated cost**__ of the __**cheapest path**__ from the state of the node n to a goal state
  - depends only on the state associated with n
  - Assumptions:
    - h(n) is non negative
    - h(n)=0 at every goal state
  - **Example:** in Romania, we may __**estimate**__
    - the **cost of the cheapest path** from Arad to Bucharest
    - **via** the __**straight-line distance**__ from Arad to Bucharest
    - $h_{SLD}(n)$ = straight-line distance from n to Bucarest

Romania with step costs in km



Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search

- Evaluation function $\boxed{f(n) = h(n)}$ (heuristic)

  ***h(n)*:** estimated cost of the cheapest path from **n** to **goal**

- Example: $h_{SLD}(n)$ = *straight-line distance* **from *n* to Bucharest**

- Greedy best-first search expands the node that appears to be closest to goal

We expand **first** the node with the lowest value of h(n)

Greedy best-first search example

h(n)

Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

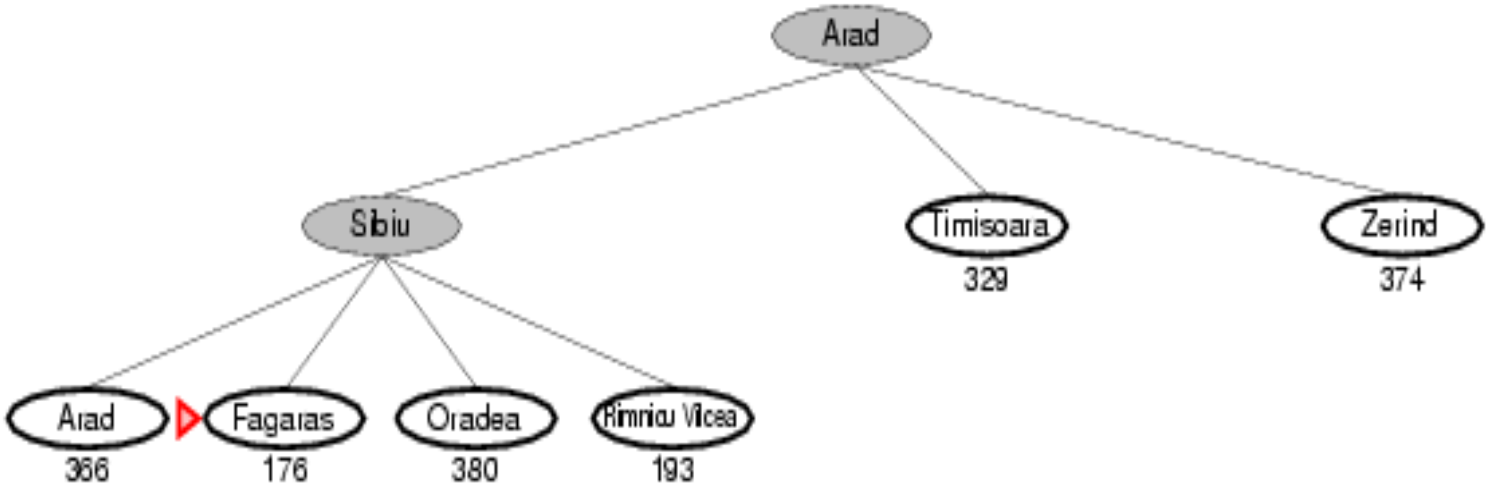
Greedy best-first search example

h(n)

Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search example

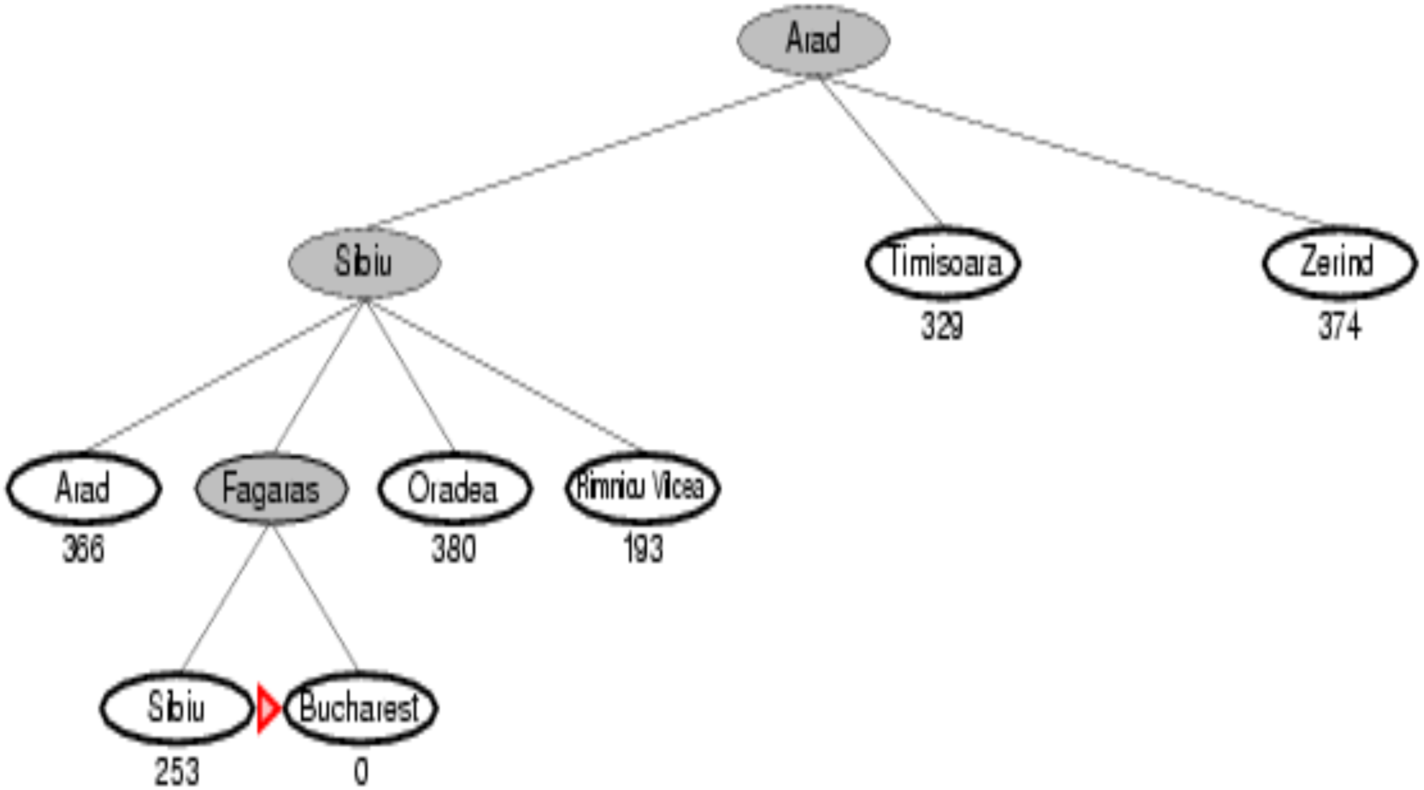


h(n)

Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search example

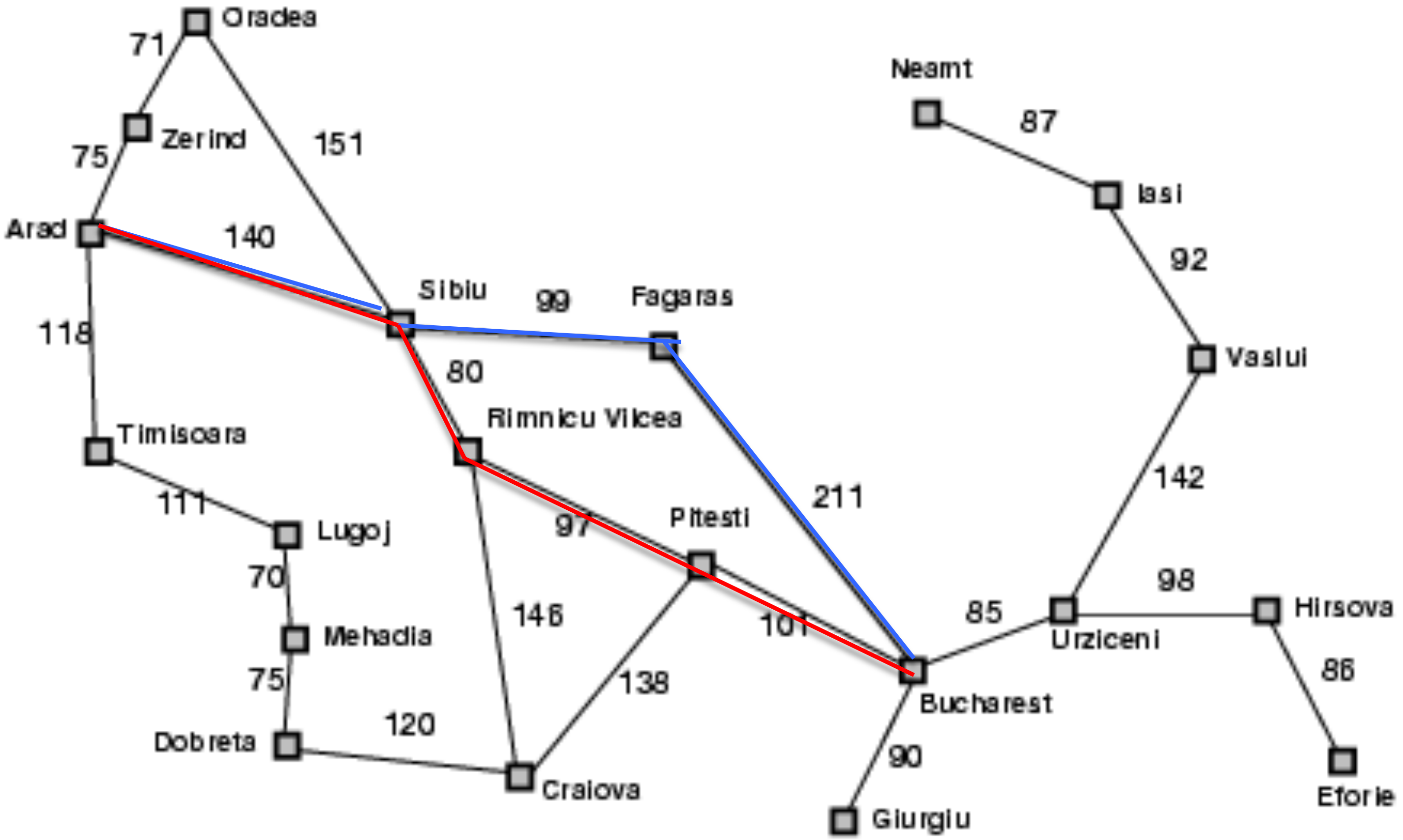


h(n)

Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

returned by **greedy BFS**

optimal



h(n)

Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

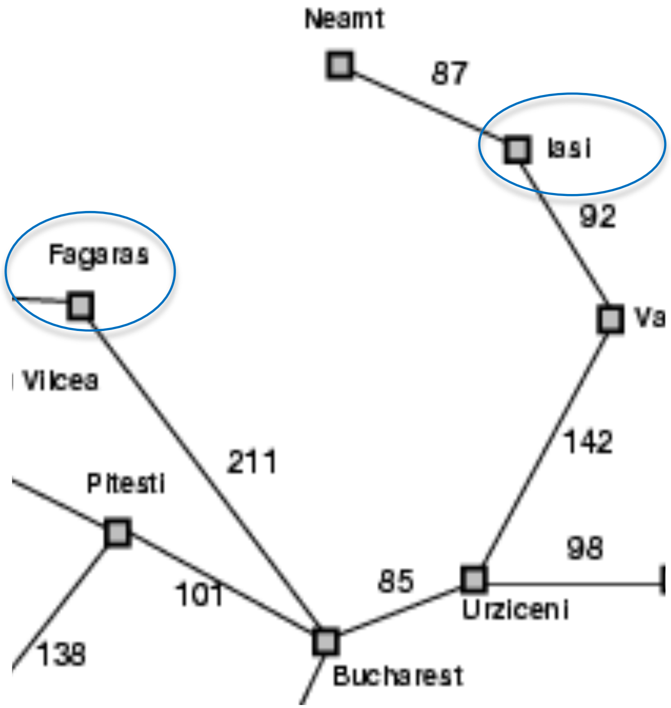
Properties of greedy best-first search

b: branching factor
m: the maximum depth of the search space

- Complete? No (tree-search)
  - NO, consider the problem of going from Iasi to Fagars: Iasi→ Neamt →Iasi→ Neamt

    The **heuristic** suggests **Neamt** be expanded **first** because it is closest to **Fagaras**,

    but **it is a dead end**
  - Complete in finite state 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

Outline

- Best-first search
  - **Greedy** best-first search
  - **A*** search
- Heuristics

A^* search

- **Idea:** avoid expanding **paths** that are **already expensive**
- Evaluation function $\boxed{f(n) = g(n) + h(n)}$
  - **g(n)** = **path cost** from the **start node** to **node n**
  - **h(n)** = estimated cost of the cheapest path from **n** to **goal**
  - **f(n)** = estimated cost of the cheapest **solution through n**
- Like **uniform-cost search** with **f=g+h** instead of **g**

A* search example

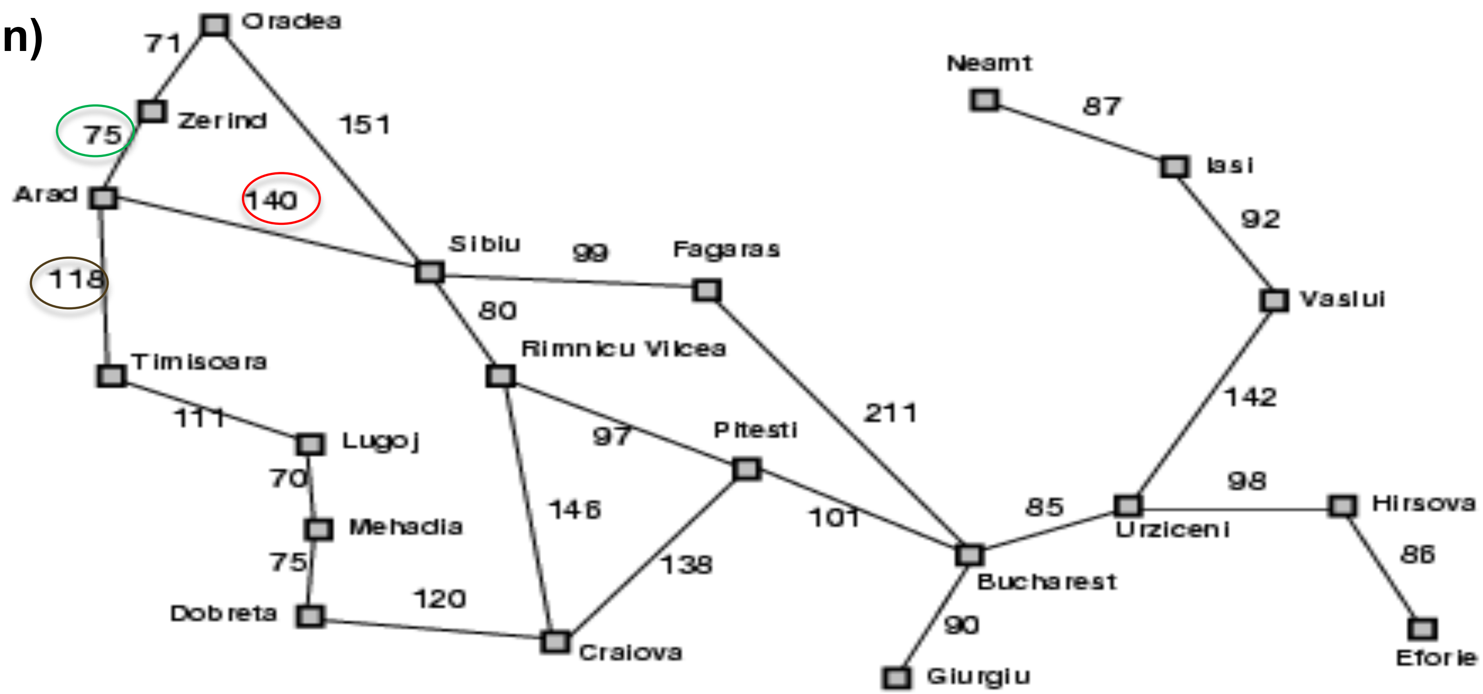
We expand **first** the node with the **lowest value** of **g(n) + h(n)**

Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

We expand first the node with the **lowest value** of **g(n)** + **h(n)**

g(n)



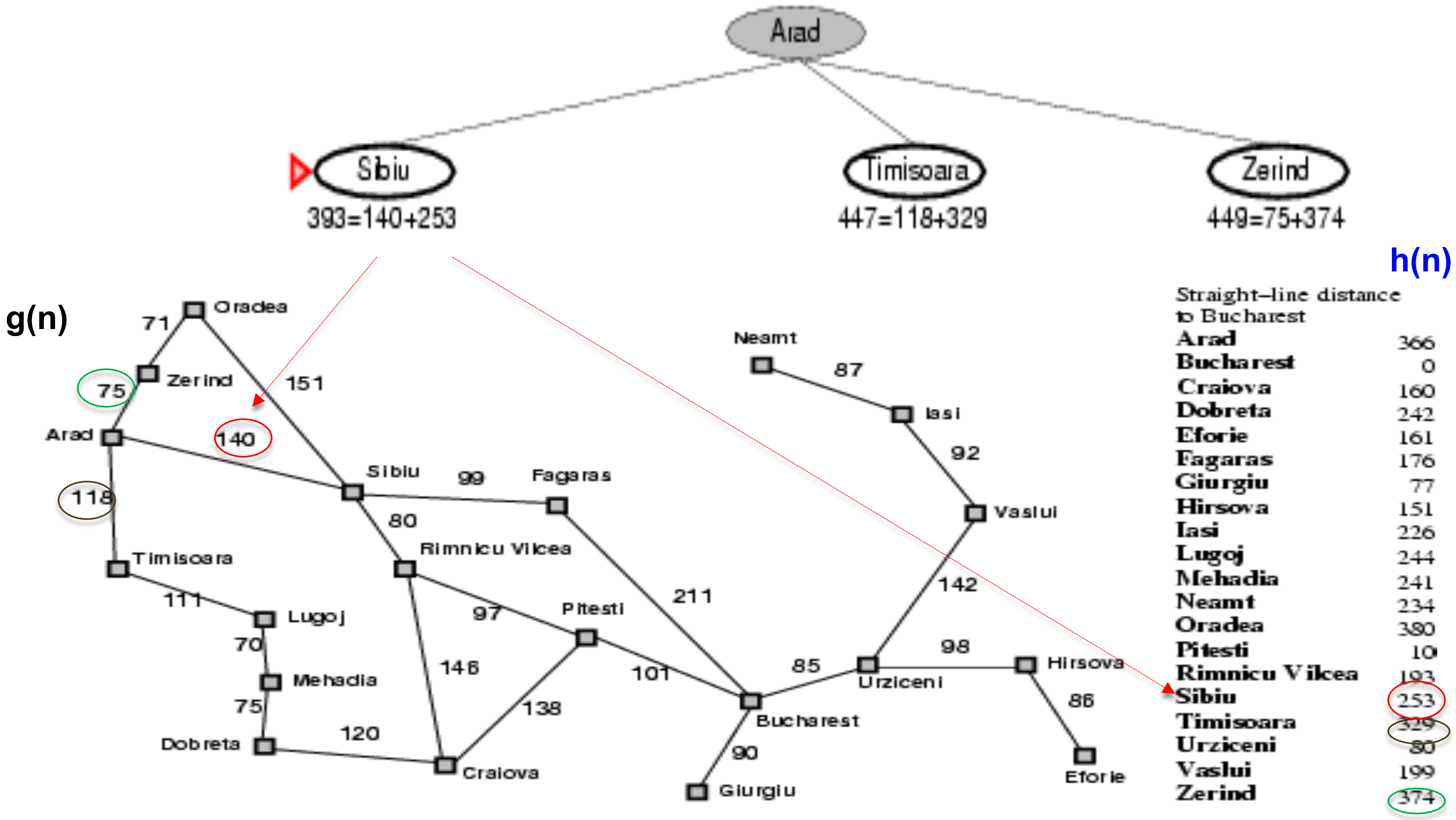
h(n)

Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10[illegible]
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

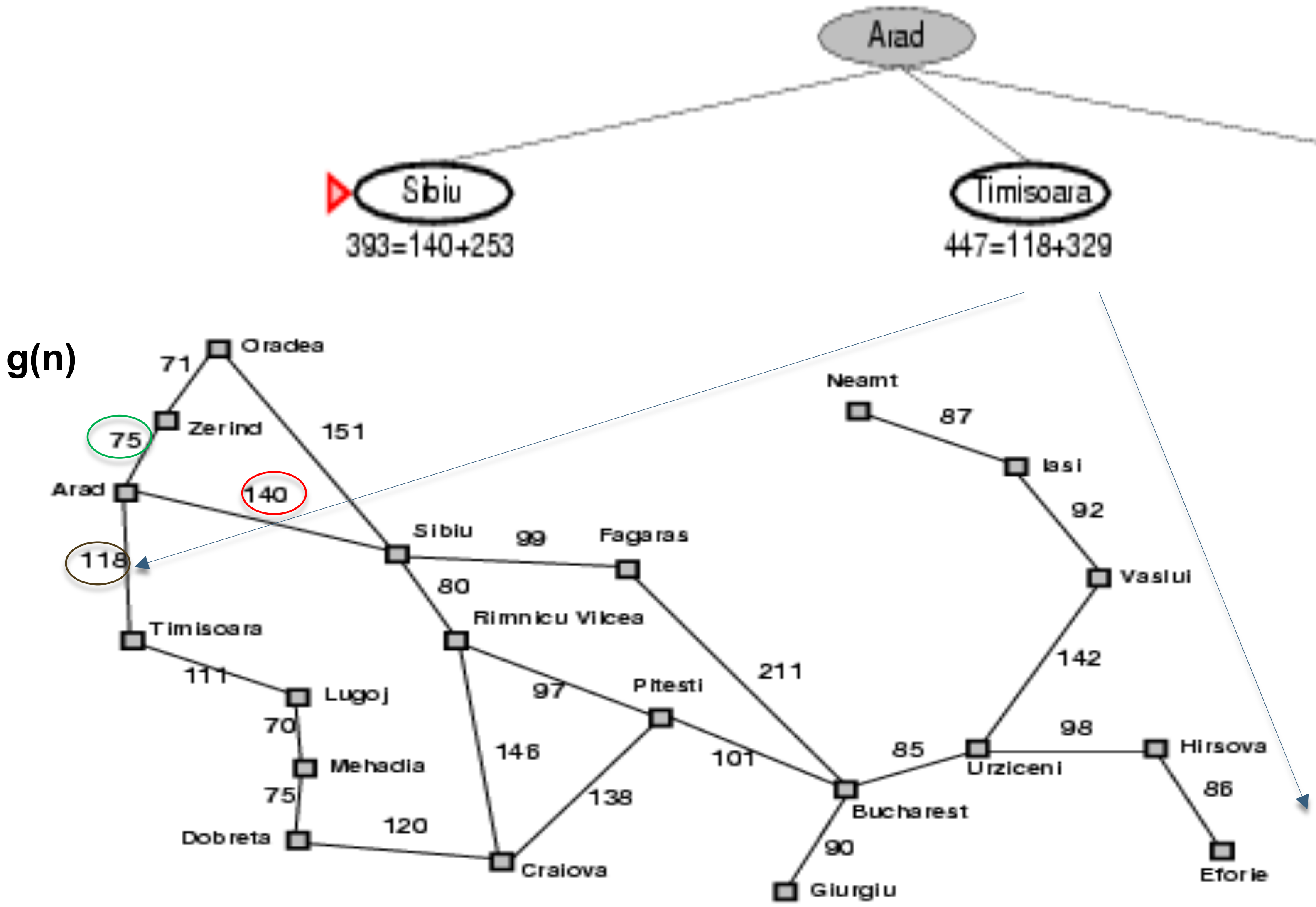
A* search example

We expand first the node with the lowest value of g(n) + h(n)



A* search example

We expand **first** the node with the **lowest value** of **g(n) + h(n)**

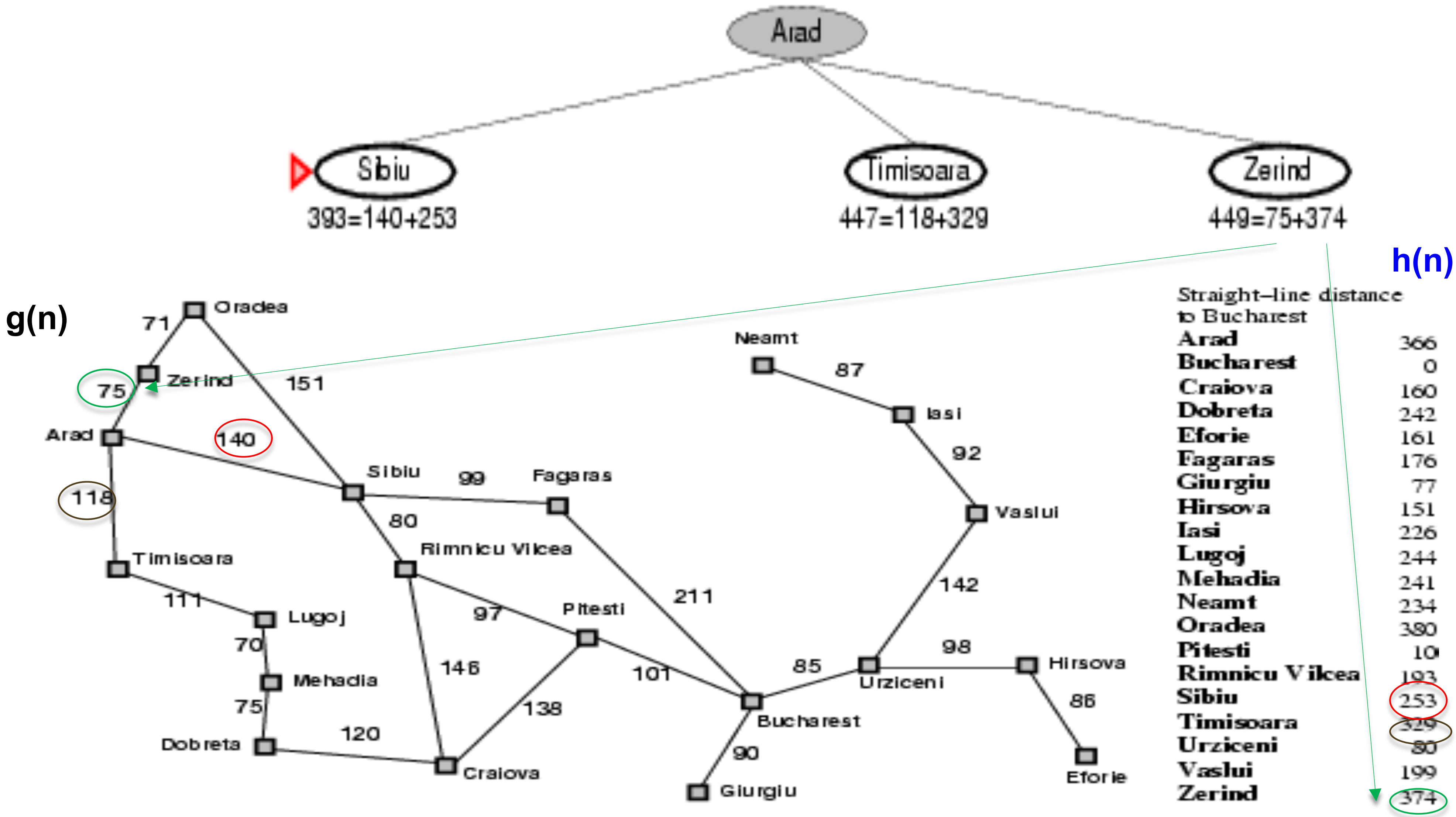


h(n)

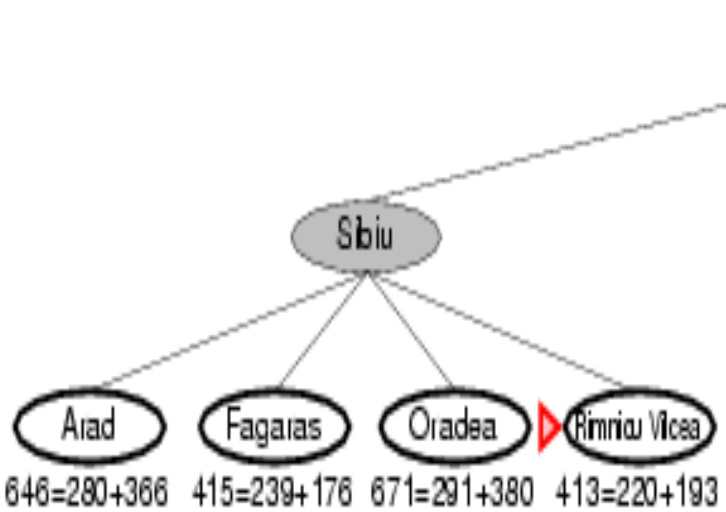
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10[illegible]
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

We expand **first** the node with the **lowest value** of **g(n) + h(n)**



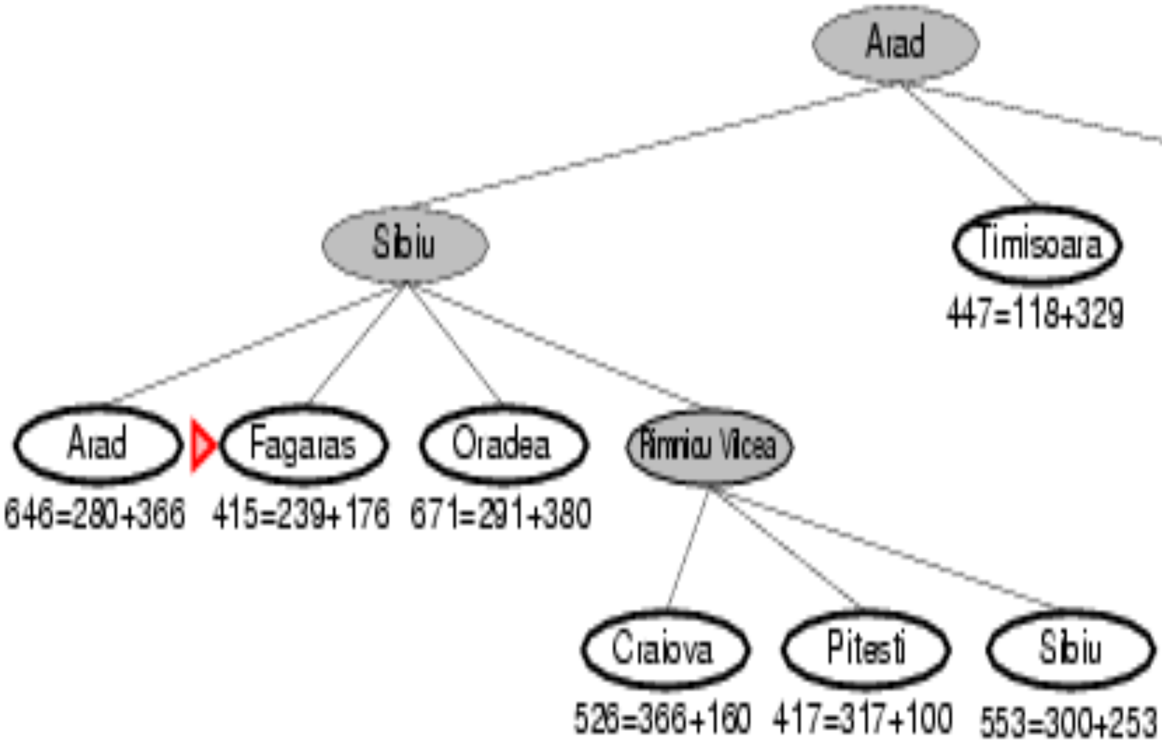
A* search example



Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

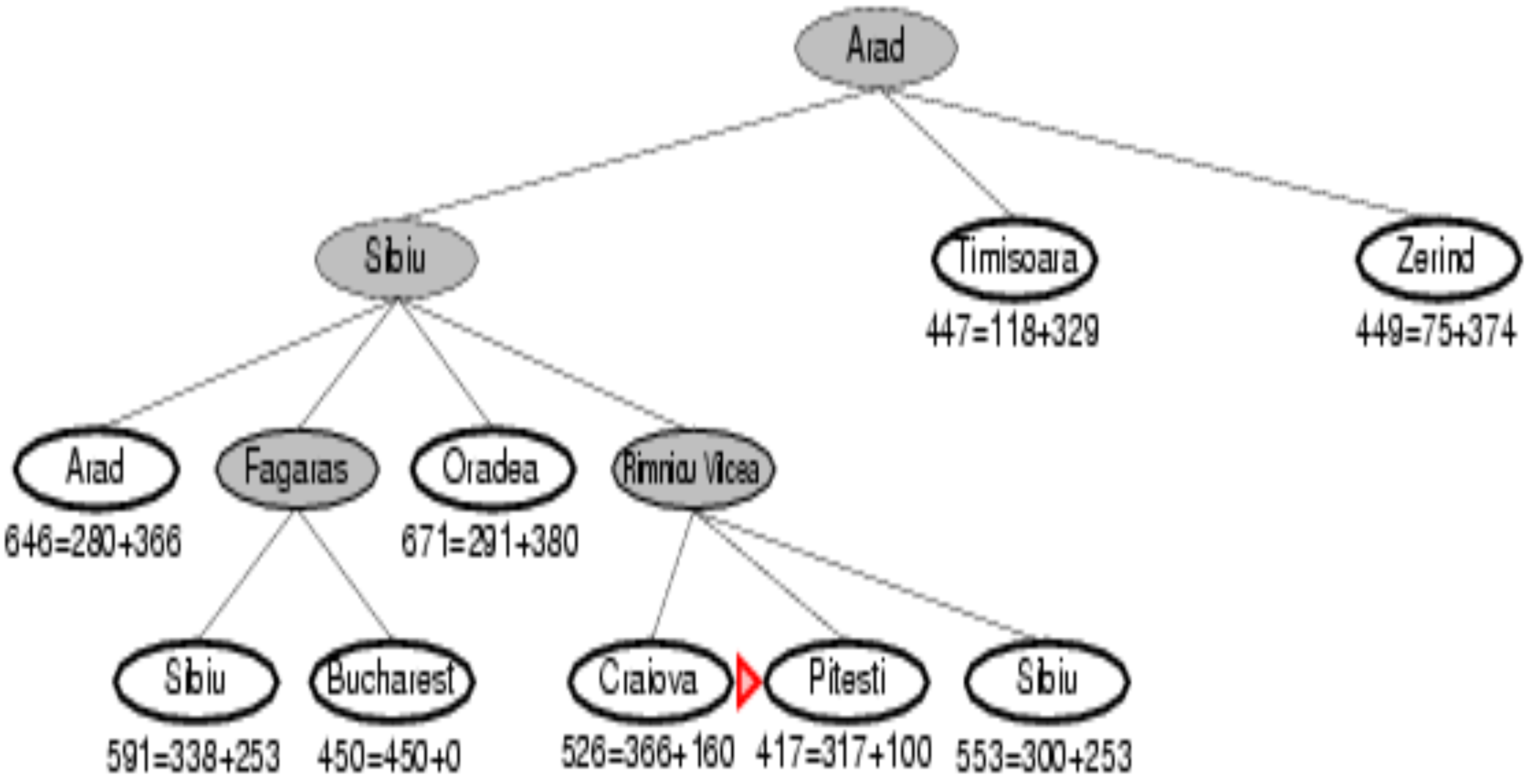
A* search example



Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

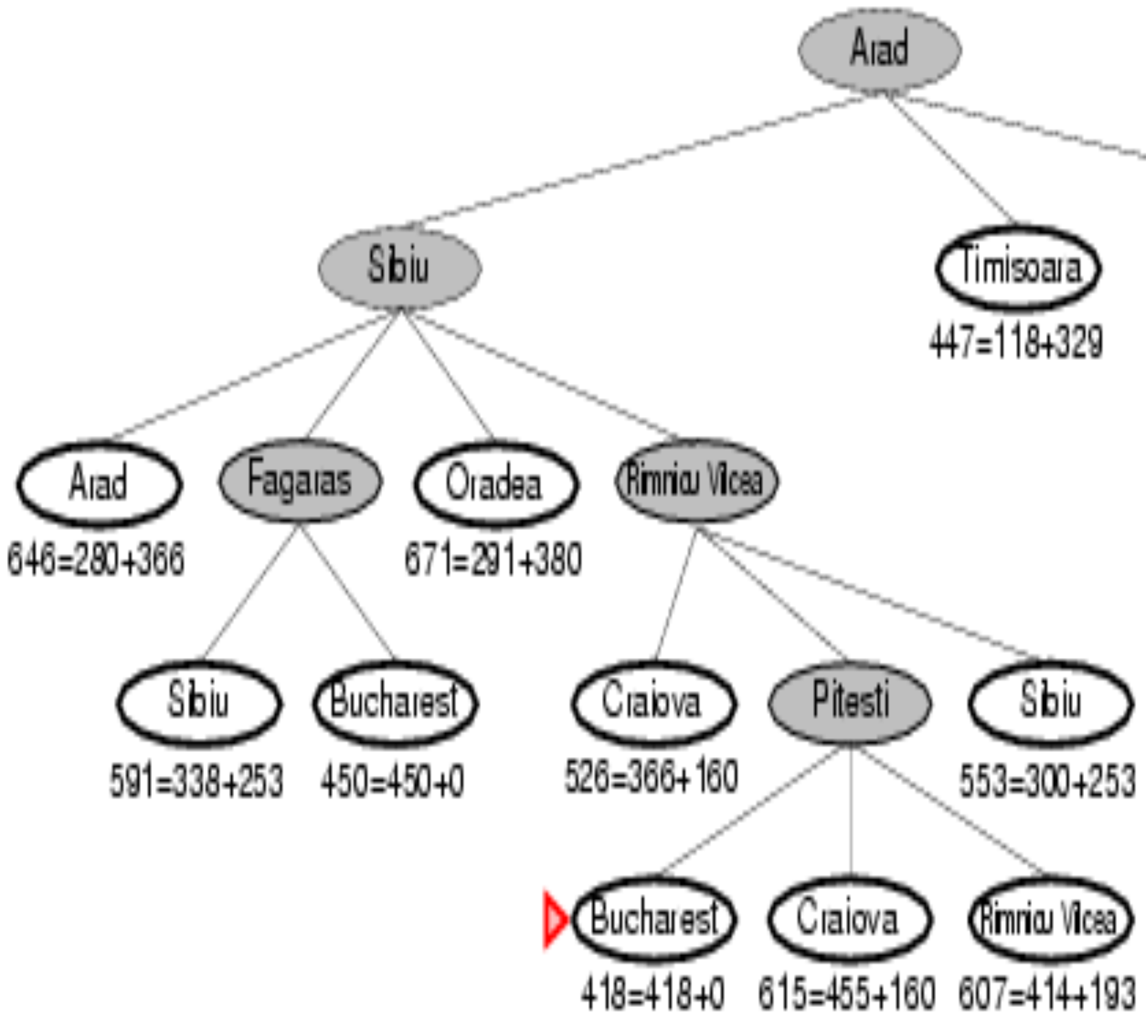
A* search example



Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
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	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Admissible heuristics

- A heuristic h is **admissible** if

  for every node *n*, $h(n) \leq h^*(n)$

  where $h^*(n)$ is the **true cost** from *n* to ***goal***

- An admissible heuristic never overestimates the cost to reach the goal

- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

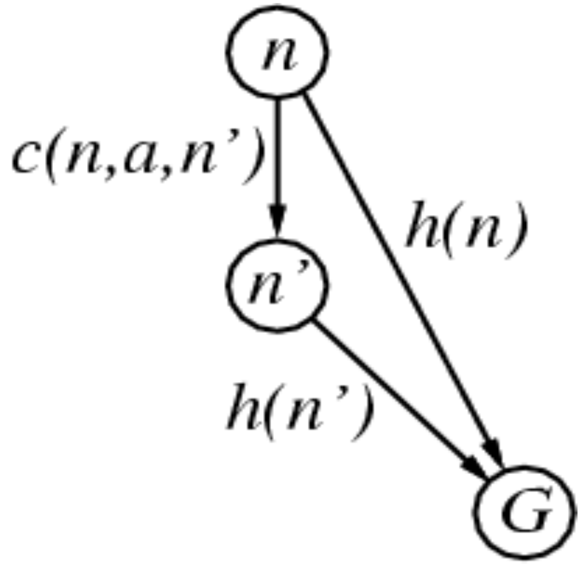
- **Theorem**: If *h* is admissible,

  A^* using TREE-SEARCH is optimal

Consistent heuristics

- A heuristic h is **consistent**
  if **for every** node ***n***,
  for every successor *n'* of *n* generated **by an action *a***,

$$h(n) \leq c(n, a, n') + h(n')$$



- Instance of triangular inequality

- Consistency → admissibility

h consistent → A* graph-search optimal

- **Theorem**: If *h* is consistent
  A* using `GRAPH-SEARCH` is optimal

Properties of A*

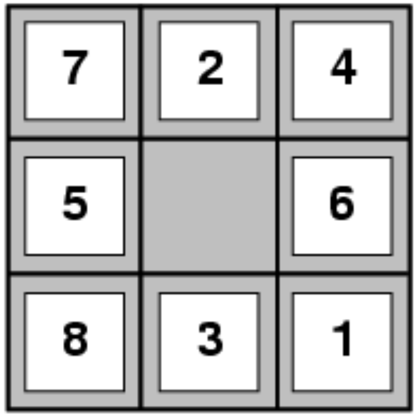
- Complete? Yes (unless there are infinitely many nodes with f $\leq f(G)$)
- Time? Exponential
- Space? Keeps all nodes in memory
- Optimal? Yes (with `TREE-SEARCH`, admissible heuristic;
  `GRAPH-SEARCH`, consistent heuristic)

Admissible heuristics

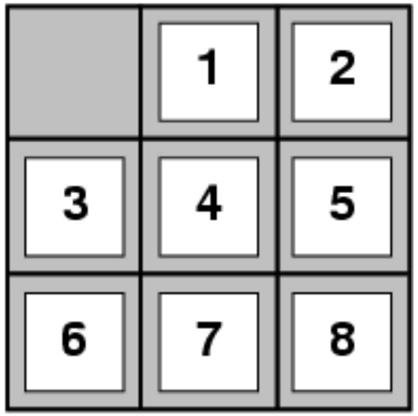
Review: **8-puzzle**
goal: to slide the tiles **horizontally or vertically** into the empty space **until** the configuration matches the goal configuration

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance
  (i.e, the sum of the horizontal and vertical distances of the tiles from their goal positions)



Start State



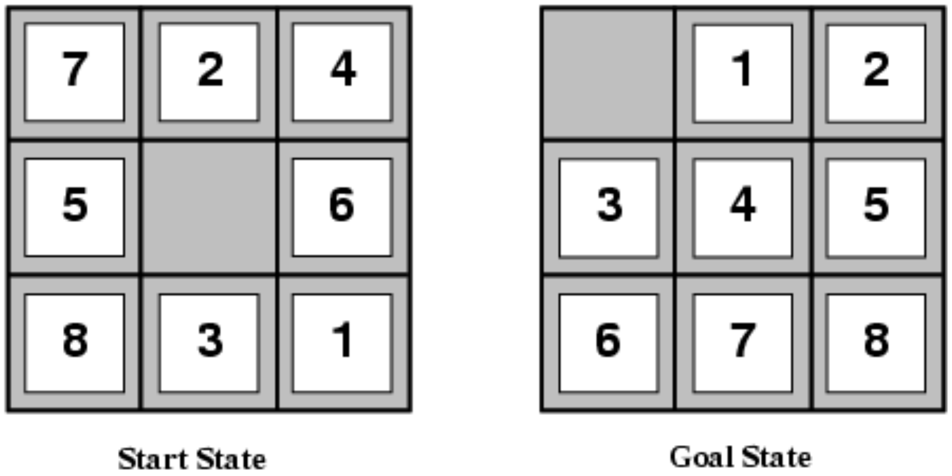
Goal State

- $h_1(S)$ = ?
- $h_2(S)$ = ?

Admissible heuristics

E.g., for the 8-puzzle:

- ***$h_1(n)$*** = number of misplaced tiles



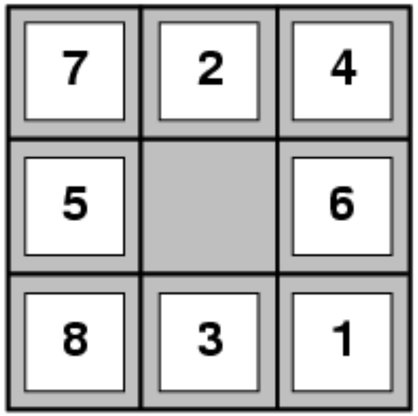
- $h_1(S)$?
- $h_1(S)$ = **8**
- Admissible: any tile out of place must be moved at least once

Admissible heuristics

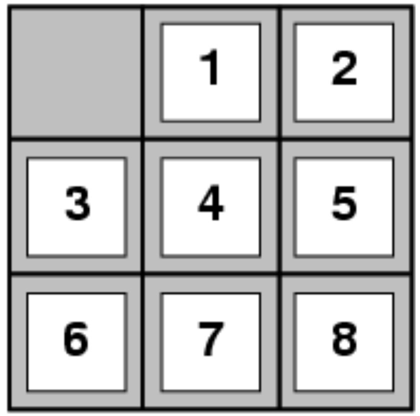
E.g., for the 8-puzzle:

- $\boldsymbol{h_2(n)}$ = Manhattan distance

(i.e., the sum of the horizontal and vertical distances of the tiles from their goal positions)



Start State



Goal State

- $h_2(S)$ =?
- $h_2(S)$ = 3+1+2+2+2+3+3+2 = **18**
- Admissible: any move can move at most one tile one step closer to the goal

Dominance

8-puzzle:
h_1 number of misplaced tiles
h_2 Manhattan distance

- If $h_2(n) \geq h_1(n)$ **for all node *n*** (both admissible)

  then h_2 **dominates** h_1

- h_2 is **better than** h_1 **for search**

  h_1 will **<u>expand</u> all the nodes** h_2 expands and **<u>possibly more</u>**

- **Comparison** of the **search costs** (average number of generated nodes) for **A* with** h_1, **A* with** h_2 and **IDS** (Iterative Deepening Search)
  - ***d* =12** **IDS** = 3,644,035 nodes
    $A^*(h_1)$ = 227 nodes
    $A^*(h_2)$ = **73 nodes**
  - ***d*=24** **IDS** = too many nodes
    $A^*(h_1)$ = 39,135 nodes
    $A^*(h_2)$ = **1,641 nodes**

d = solution length

Generating admissible heuristics from relaxed problems

- A **problem** with **fewer restrictions on the actions** is called a **relaxed problem**
- A **superset of actions** is available from **each state**
- The **graph** of the relaxed problem is a **supergraph** of the original
- Any optimal solution of the original problem is also
  a solution of the relaxed problem
- The cost of an optimal solution of the relaxed problem
  may be the same **or** lower than
  the cost of an optimal solution of the original problem

→

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

Generating admissible heuristics from relaxed problems

Admissible **heuristics** for the 8-puzzle:

h_1 number of misplaced tiles

h_2 Manhattan distance

- **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 (**instead** of just to the adjacent **empty square**) then $h_1(n)$ gives the **exact** number of steps of the optimal solution

- If the rules of the **8-puzzle** are relaxed so that **a tile can move** one square in any direction, even onto an occupied square, then $h_2(n)$ gives the **exact** number of steps of the optimal solution