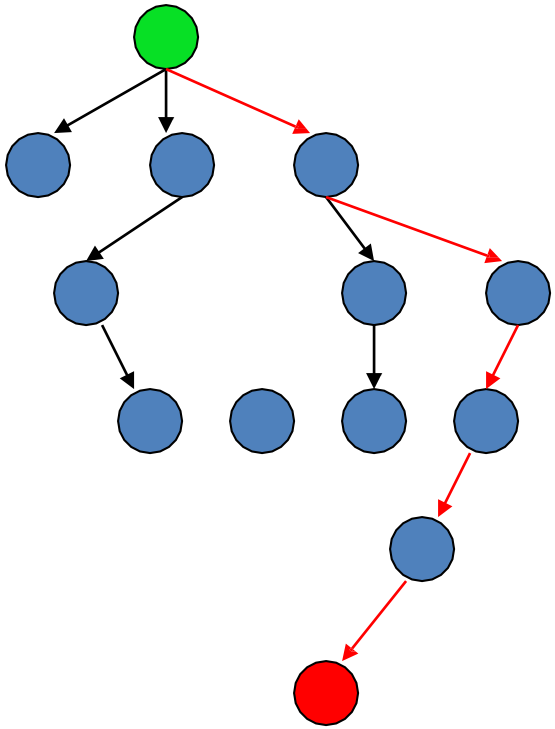
Informed search algorithms

Chapter 3

Informed (Heuristic) Search

Idea: be **smart** about what paths to try.



Blind Search vs. Informed Search

- What's the difference?

- How do we formally specify this?

  A node is selected for expansion based on an evaluation function that estimates cost to goal.

General Tree Search Paradigm

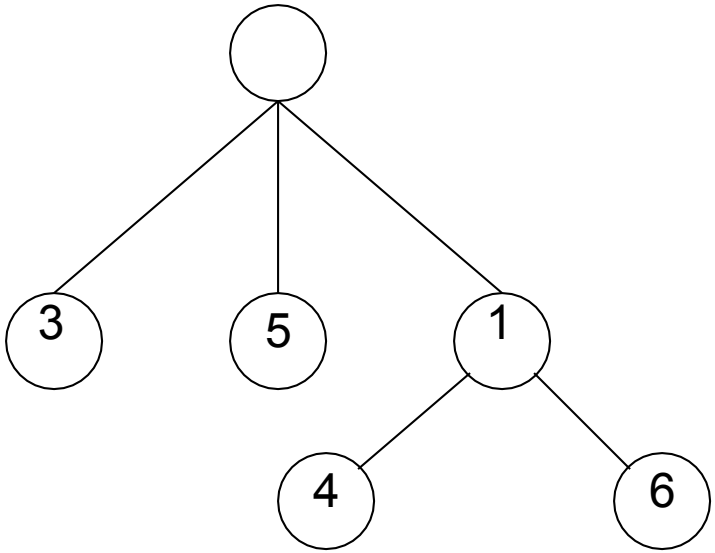
```
function tree-search(root-node)
   fringe ← successors(root-node)
   while ( notempty(fringe) )
         {node ← remove-first(fringe) //lowest f value
             state ← state(node)
             if goal-test(state) return solution(node)
             fringe ← insert-all(successors(node),fringe) }
   return failure
end tree-search
```

General Graph Search Paradigm

```
function tree-search(root-node)
   fringe ← successors(root-node)
   explored ← empty
   while ( notempty(fringe) )
         {node ← remove-first(fringe)
             state ← state(node)
             if goal-test(state) return solution(node)
             explored ← insert(node,explored)
             fringe ← insert-all(successors(node),fringe, if node not in explored)
         }
   return failure
end tree-search
```

Best-First Search

- Use an evaluation function f(n) for node n.
- Always choose the node from fringe that has the lowest f value.

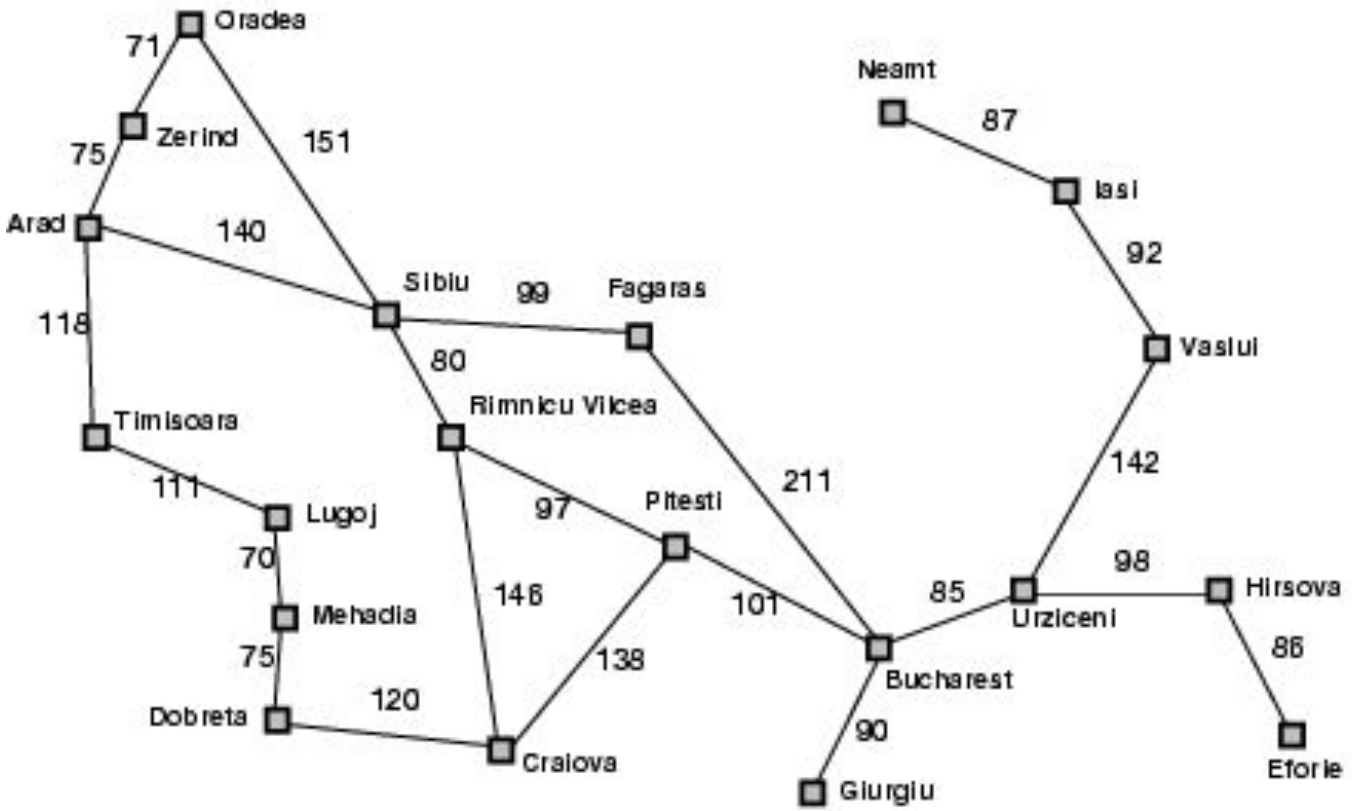


Best-first search

- A search strategy is defined by picking the order of node expansion
- Idea: use an evaluation function *f(n)* for each node
  - estimate of "desirability“

  Expand most desirable unexpanded node
- Implementation:
  Order the nodes in fringe in decreasing order of desirability
- Special cases:
  - greedy best-first search
  - A* search

Romania with step costs in km



Old (Uninformed) Friends

- Breadth First =
  - Best First
  - with f(n) = depth(n)

- Uniform cost search =
  - Best First
  - with f(n) = the sum of edge costs from start to n

g(n)

Greedy best-first search

- Evaluation function $f(n) = h(n)$ (heuristic function)
  = estimate of cost from n to *goal*

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

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

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

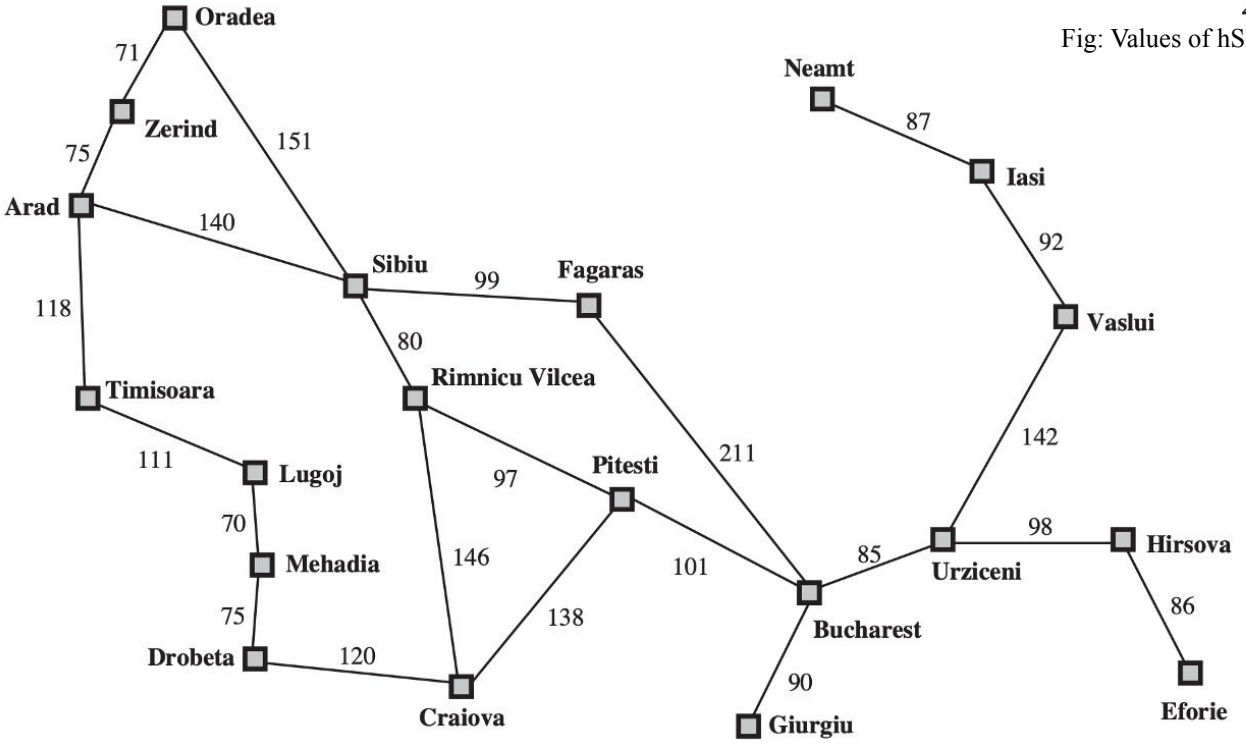
- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n) = g(n) + h(n)$

- $g(n)$ = 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

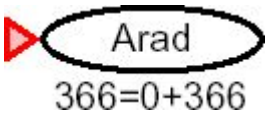
A* for Romanian Shortest Path

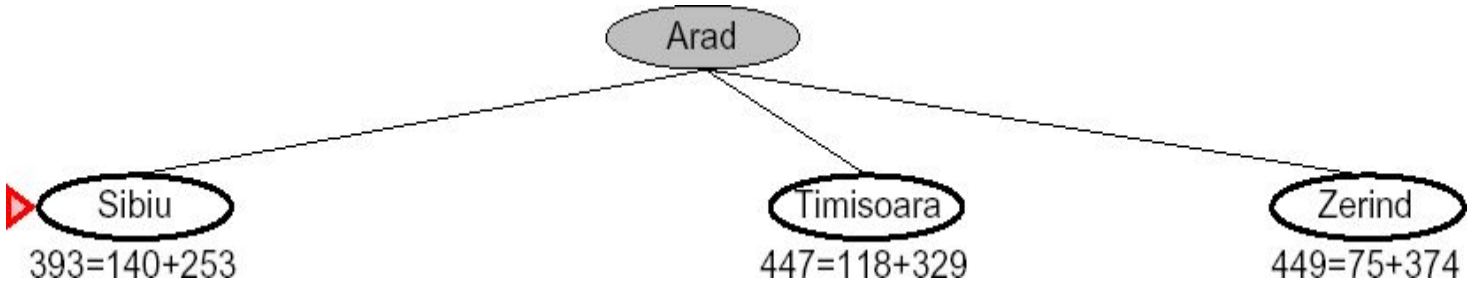
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
	244	**Zerind**	374

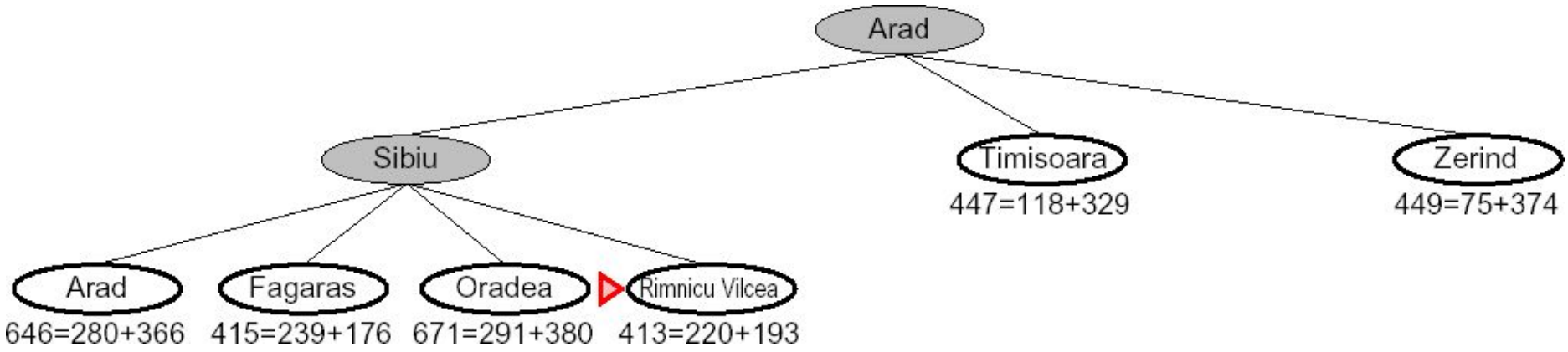
Fig: Values of hSLD —straight-line distances to Bucharest.

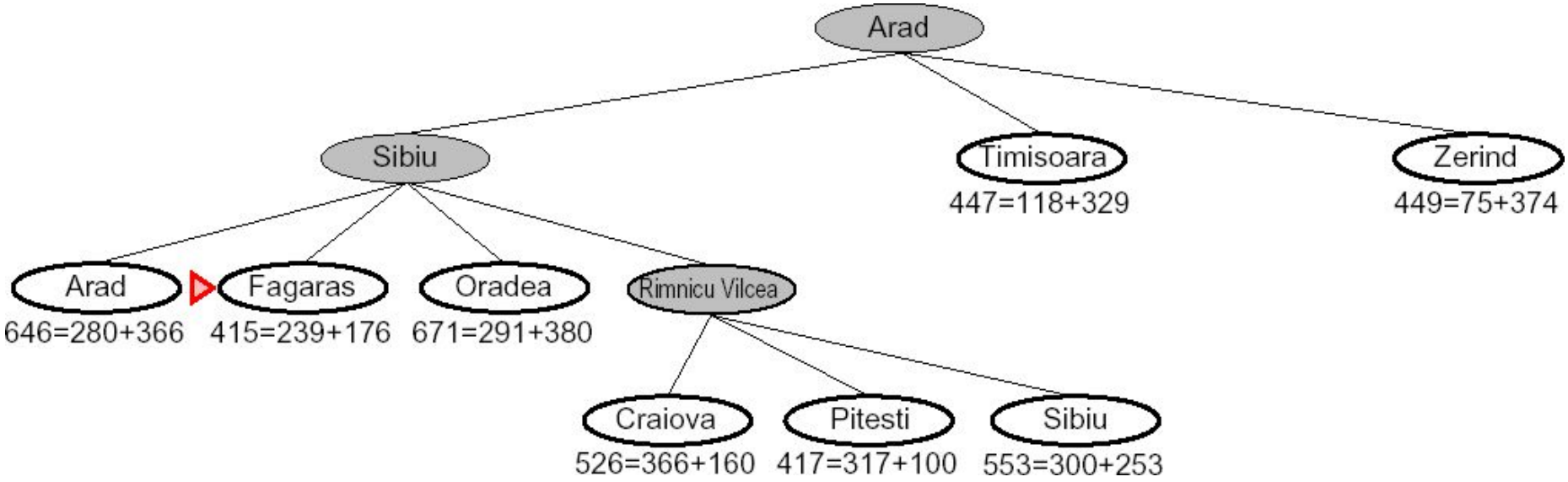


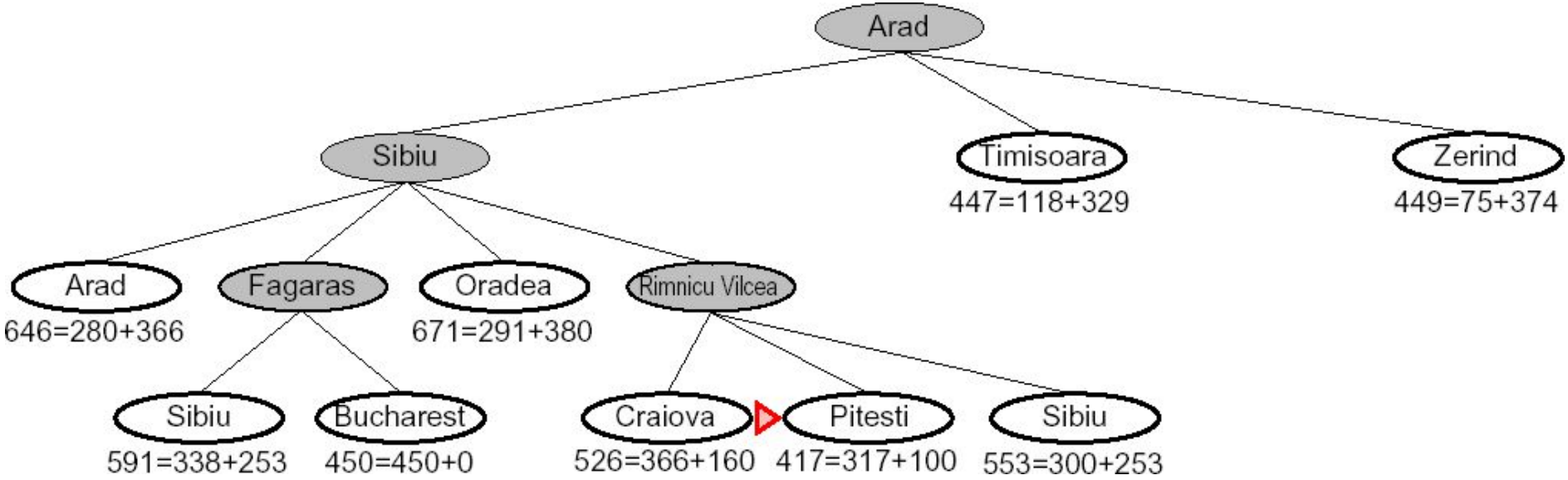
A* for Romanian Shortest Path

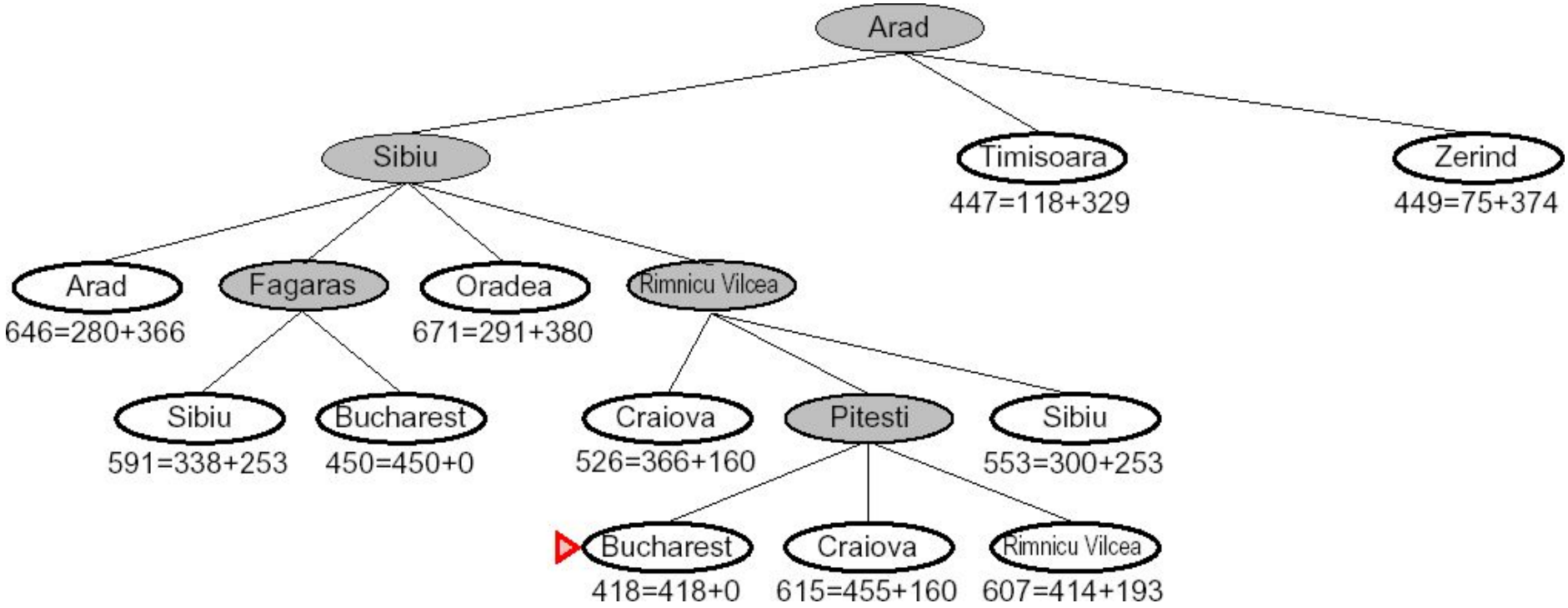



Arad
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374


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


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


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
Sibiu
553=300+253
Bucharest
418=418+0
Craiova
615=455+160
Rimnicu Vilcea
607=414+193

Admissible heuristics

- A heuristic function $h(n)$ is admissible if for every node n, $h(n) \leq h^*(n)$,

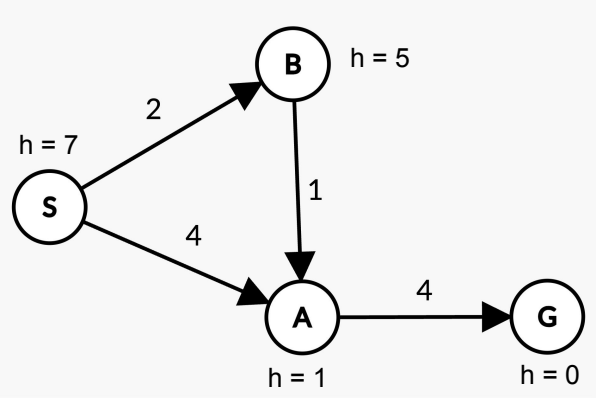
  where $h^*(n)$ is the true cost to reach the goal state from n.

- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic

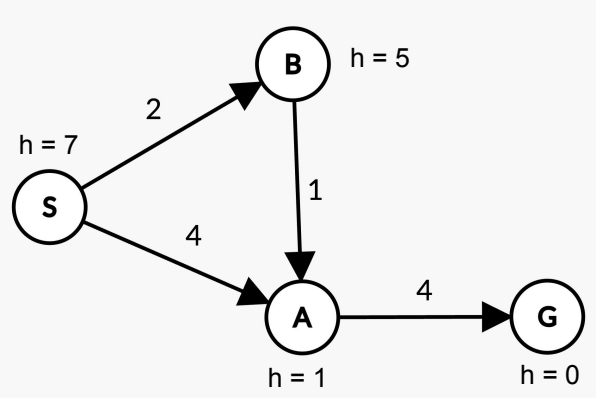
- Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

- Theorem: If $h(n)$ is admissible, A* using `TREE-SEARCH` is optimal

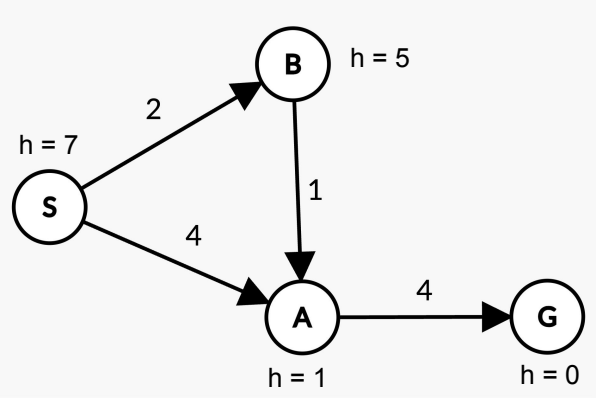
Admissible heuristics – Tree Search



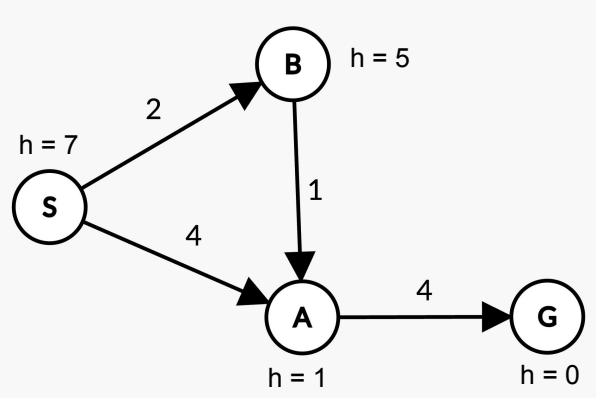
Admissible heuristics – Tree Search



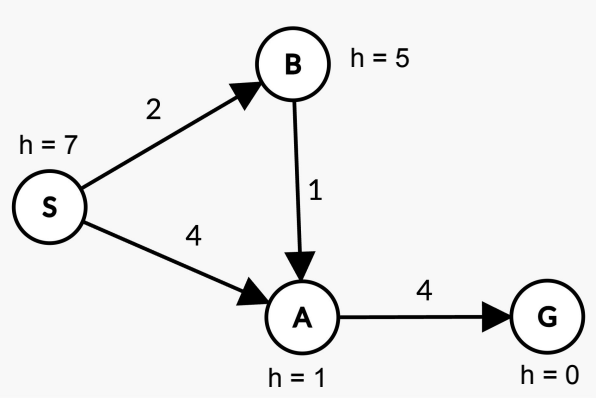
Admissible heuristics – Tree Search



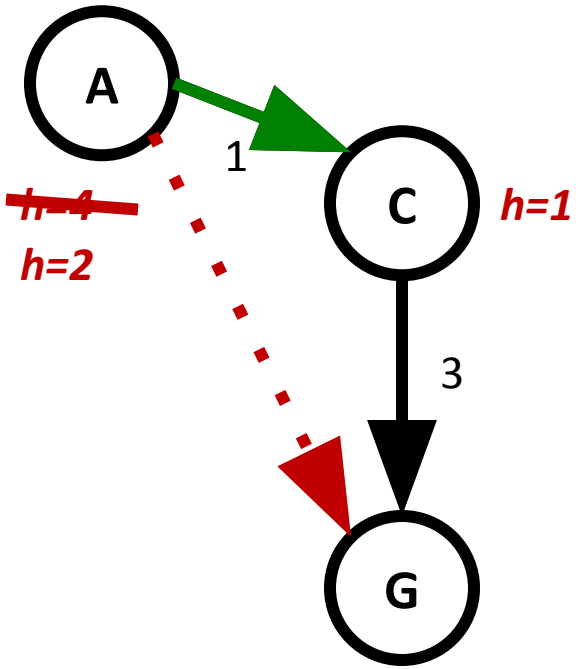
Admissible heuristics – Tree Search



Admissible heuristics – Tree Search



Consistency of Heuristics



- h(n) is consistent if
  - for every node **n**
  - and for every successor **s** of **n**
  - h(n) <= c(n, s) + h(s)

- Consequences of consistency:

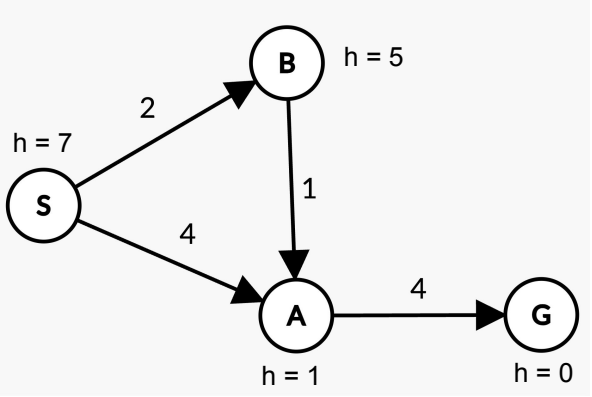
  The f value along a path never decreases

  h(A) <= c(A, C) + h(C)

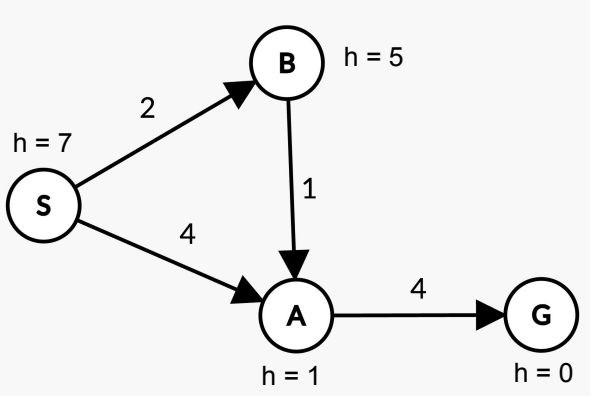
  A* graph search is optimal

- Theorem: If *h(n)* is consistent, A* using `GRAPH-SEARCH` is optimal

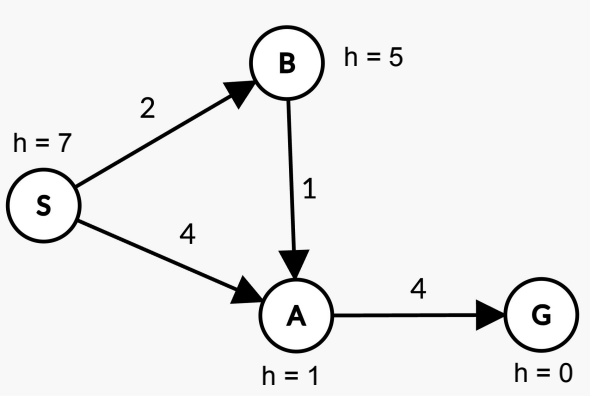
Example – Graph Search



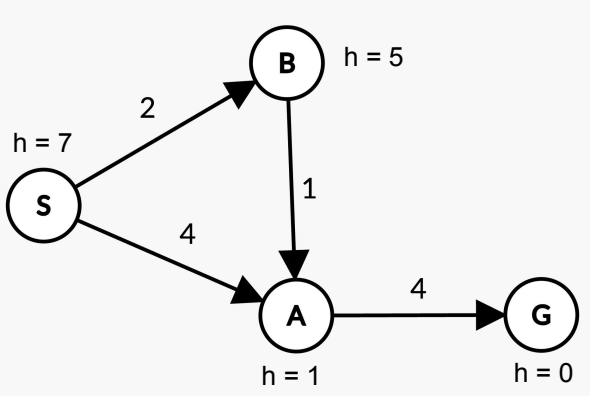
Example – Graph Search



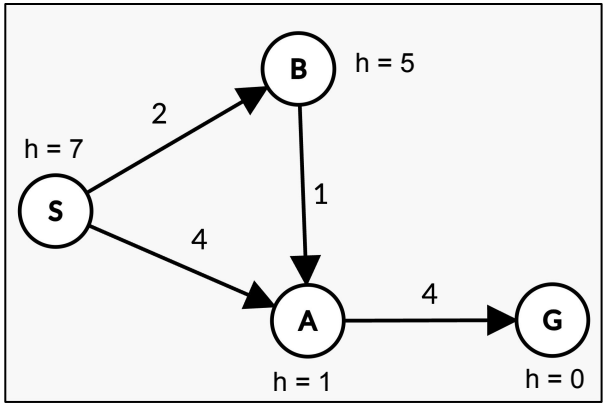
Example – Graph Search



Example – Graph Search



Checking Consistency of a heuristic



A heuristic is consistent if:

$$h(n) \leq c(n, s) + h(s)$$

$* \boldsymbol{n = current\ node}$
$\boldsymbol{s = successor\ of\ n}$

For $n = S$ and $s = A$

$h(S) \leq c(S, A) + h(A)$
$7 \leq 4 + 1$
$7 \leq 5$ X

For $n = S$ and $s = B$

$h(S) \leq c(S, B) + h(B)$
$7 \leq 2 + 5$
$7 \leq 7$

For $n = A$ and $s = G$

$h(A) \leq c(A, G) + h(G)$
$1 \leq 4 + 0$
$1 \leq 4$

For $n = B$ and $s = A$

$h(B) \leq c(B, A) + h(A)$
$5 \leq 1 + 1$
$5 \leq 2$ X

- h(S) and h(B) are not consistent
- Therefore, the heuristics is not consistent
- Graph search will not guarantee optimal solution
- If we set h(S) = 5 and h(B) = 2, the heuristic will be consistent.
- Change these values in the graph and perform graph search. You will get the optimal path to the goal.