Last class

- Breadth-first search
- Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative deepening depth-first search
- Bidirectional search

Uninformed search

No additional information beyond the problem definition

Heuristic Search and Evolutionary Algorithms

Lecture 3: Informed Search

Chao Qian （钱超）

Associate Professor, Nanjing University, China

Email: qianc@nju.edu.cn
Homepage: http://www.lamda.nju.edu.cn/qianc/

Informed Search

Informed search uses problem-specific knowledge beyond the definition of the problem

Heuristic function:

$h(n)$ = estimated cost of the optimal path from the state at node n to a goal state

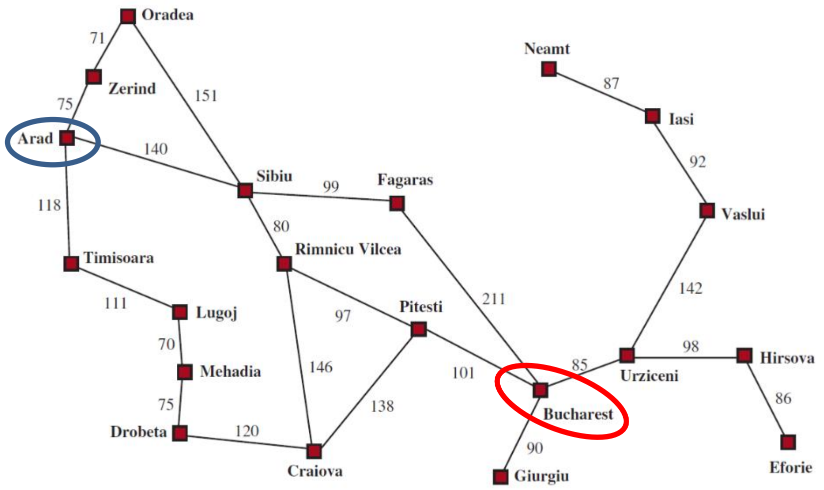
- $h(n)$ depends on the state instead of the node
- $h(n)$ measures the goodness of a state
- $h(n)$ is non-negative, and is equal to 0 if n is a goal node

How to utilize the heuristic function $h(n)$?

Greedy best-first search

Main idea: expand the node with the lowest $h(n)$ value

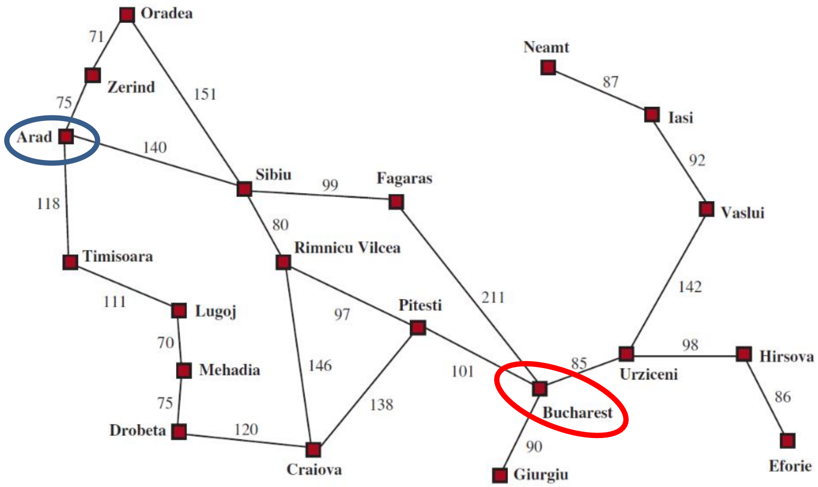
Problem: find the shortest path from Arad to Bucharest



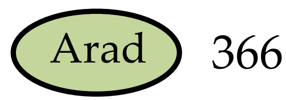
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

$h(n)$: straight-line distances to Bucharest

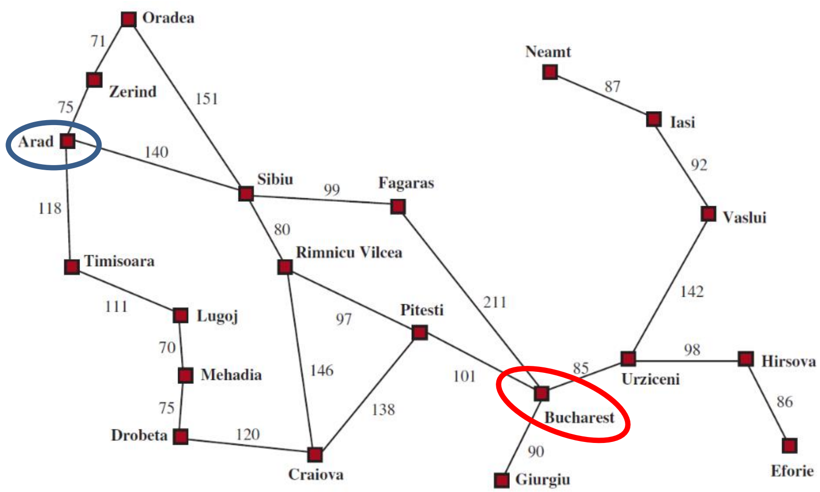
Greedy best-first search - example



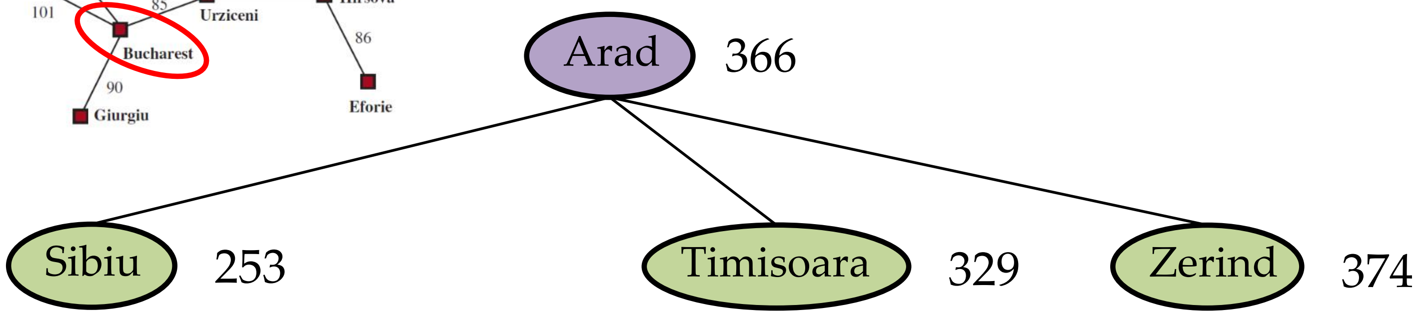
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



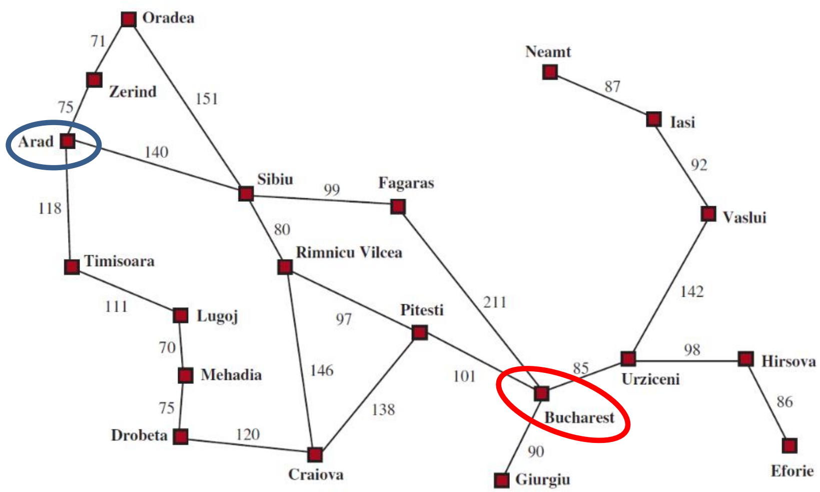
Greedy best-first search - example



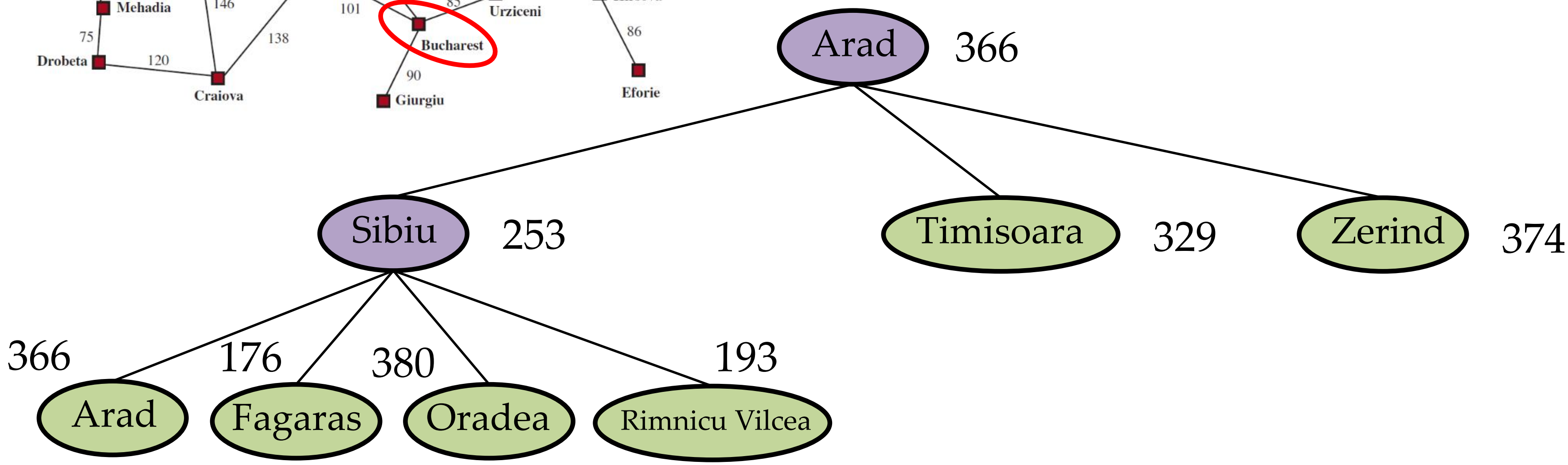
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



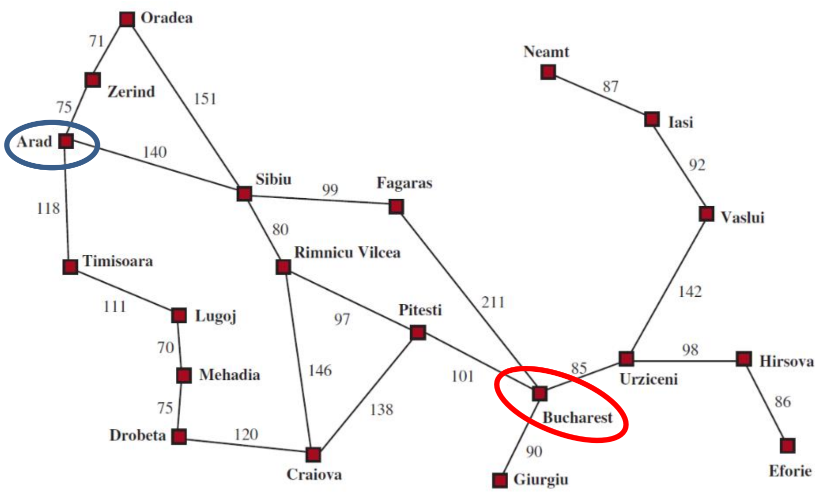
Greedy best-first search - example



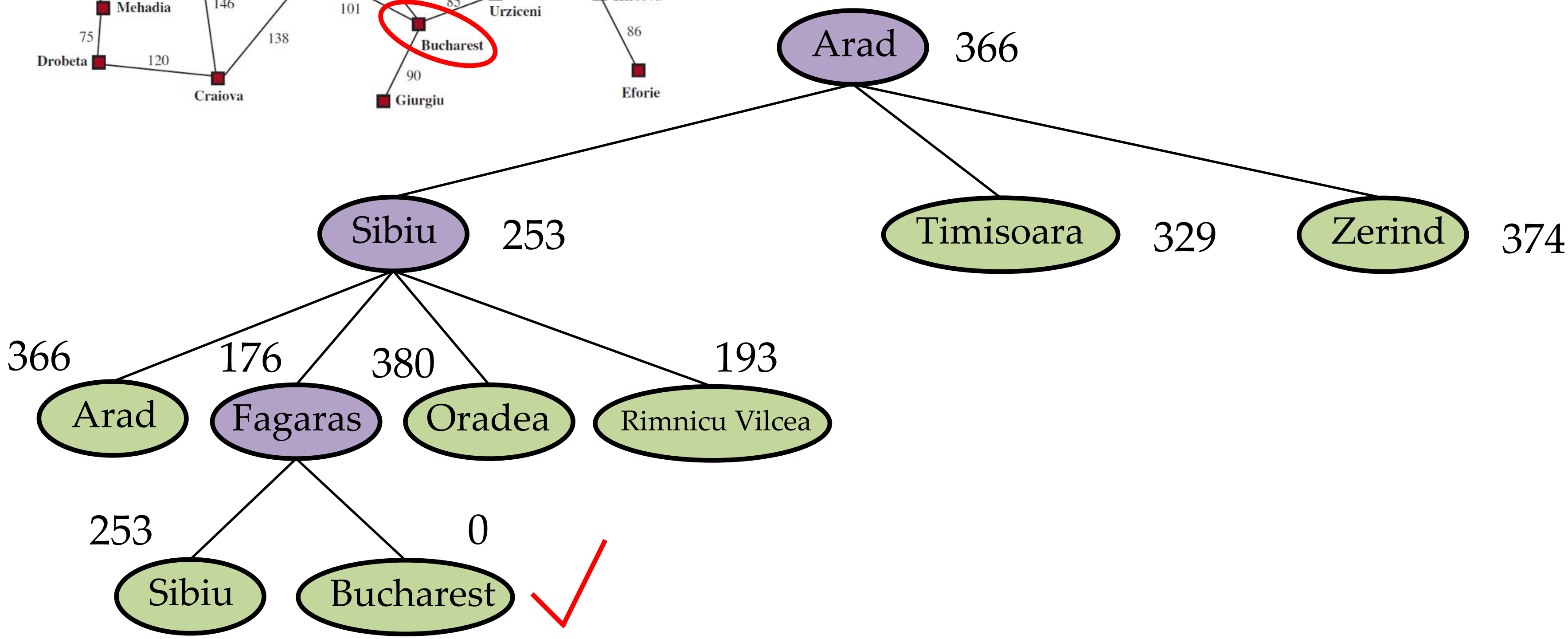
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



Greedy best-first 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

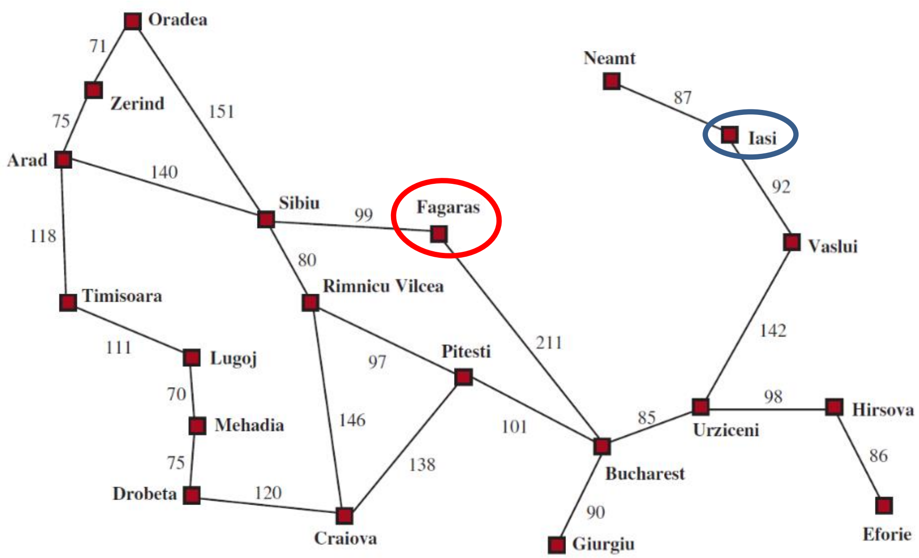


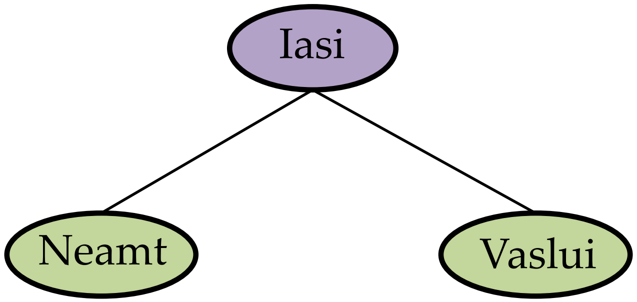
Greedy best-first search - performance

Main idea: expand the node with the lowest $h(n)$ value

- Complete

    if using tree-search, no



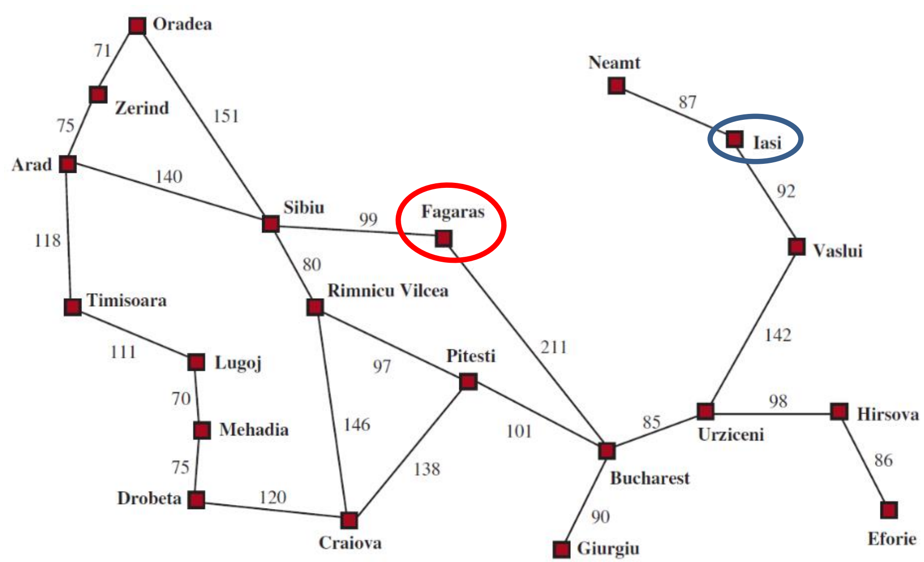


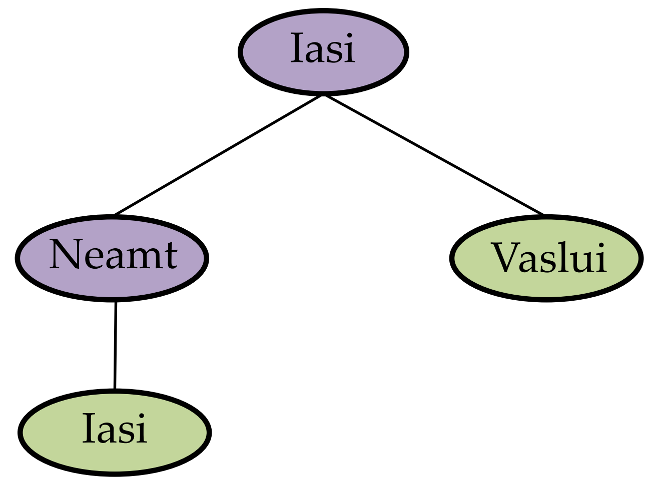
Greedy best-first search - performance

Main idea: expand the node with the lowest $h(n)$ value

- Complete

    if using tree-search, no





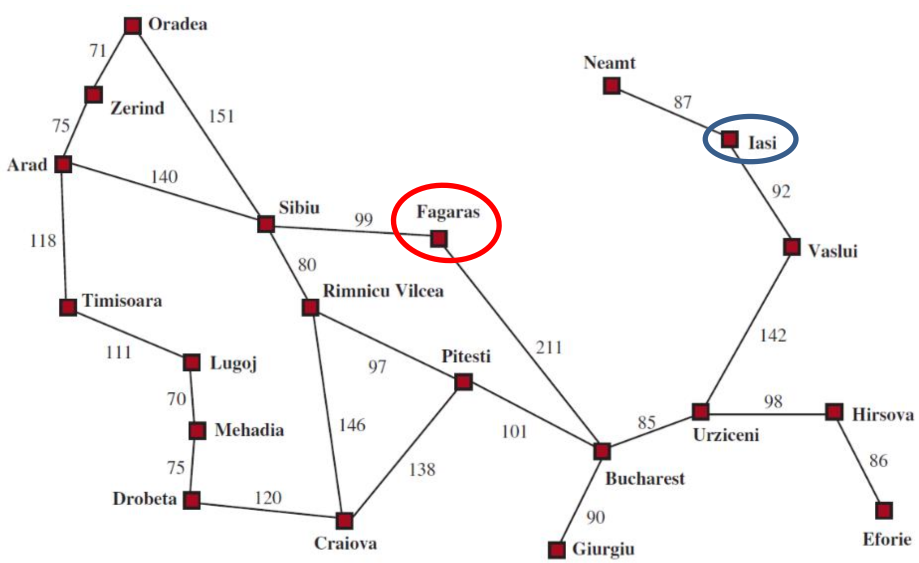
Greedy best-first search - performance

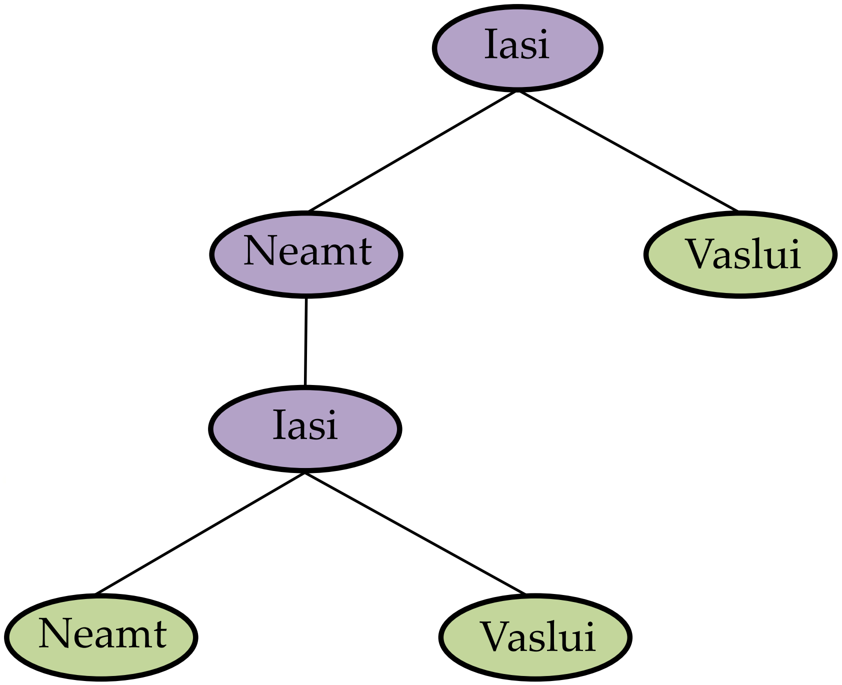
Main idea: expand the node with the lowest $h(n)$ value

- Complete

  if using tree-search, no

  if using graph-search and the state space is finite, yes





Greedy best-first search - performance

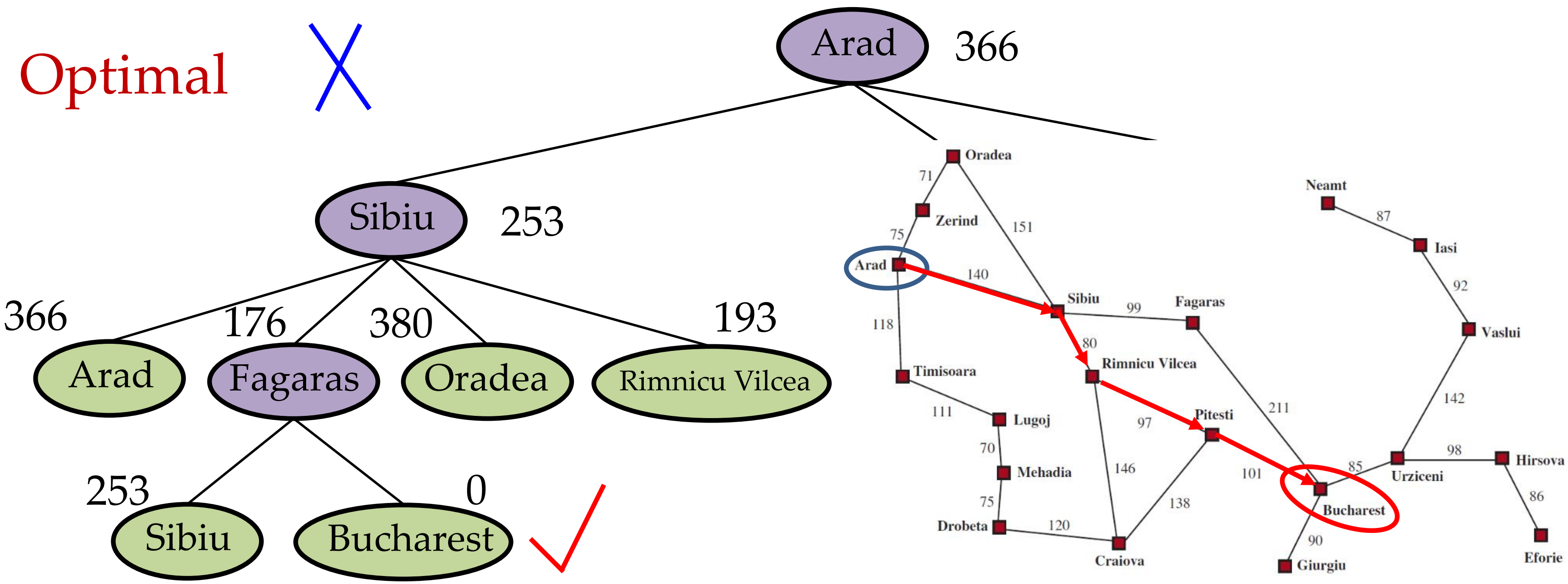
Main idea: expand the node with the lowest $h(n)$ value

- Complete

  if using tree-search, no

  if using graph-search and the state space is finite, yes

- Optimal



Greedy best-first search - performance

Main idea: expand the node with the lowest $h(n)$ value

- Complete

  if using tree-search, no

  if using graph-search and the state space is finite, yes

- Optimal ✗

- Time complexity

  $O(b^m)$, *where m is the maximum depth of any node, if using tree-search*

- Space complexity

  $O(b^m)$, *where m is the maximum depth of any node, if using tree-search*

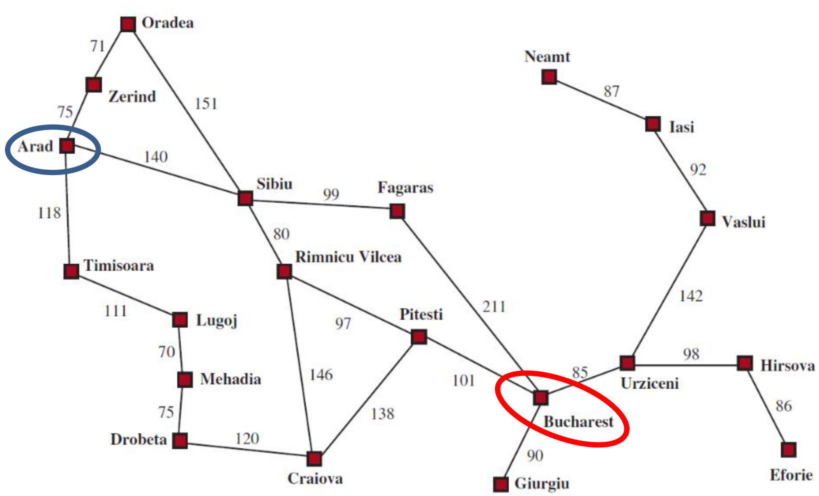
A* search

Main idea: expand the node n with the lowest value of

$$g(n) + h(n)$$

Path cost from the initial node to the node n

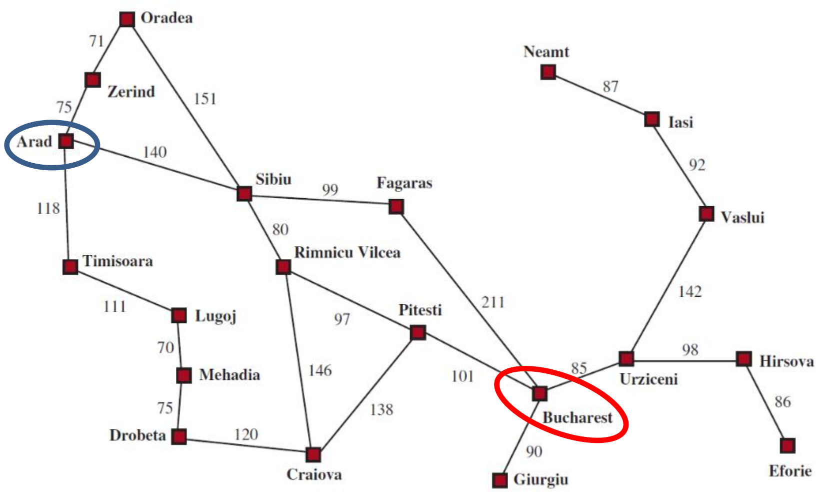
Estimated cost of the optimal path from the node n to a goal



$h(n)$: straight-line distances to Bucharest

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

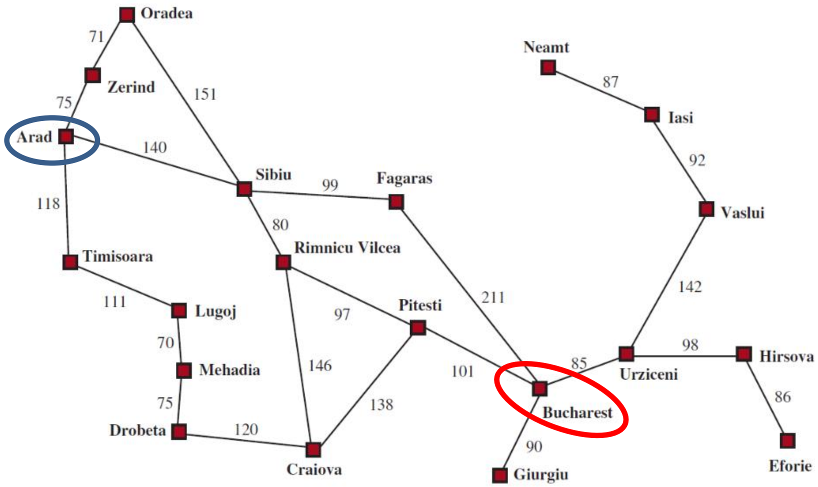
A* search - example



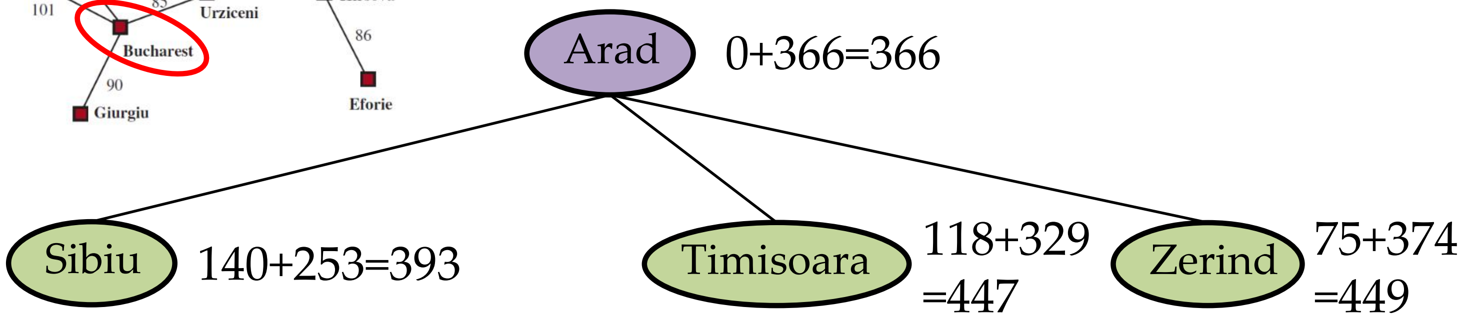
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Arad 0+366=366

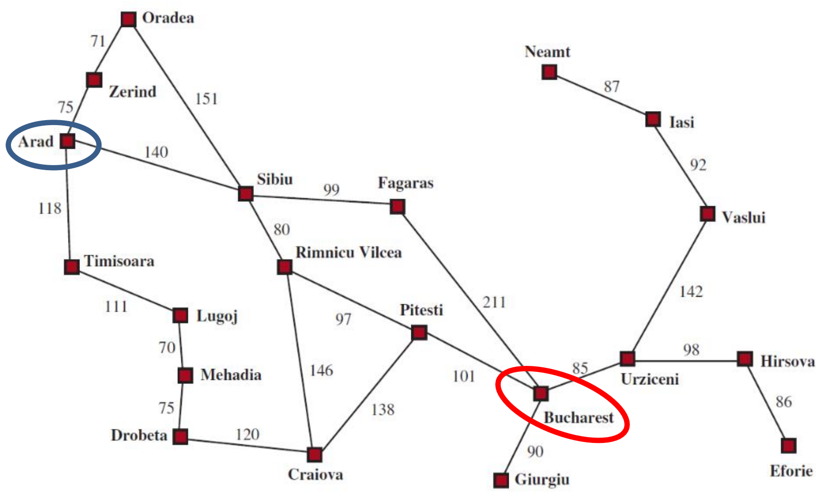
A* search - example



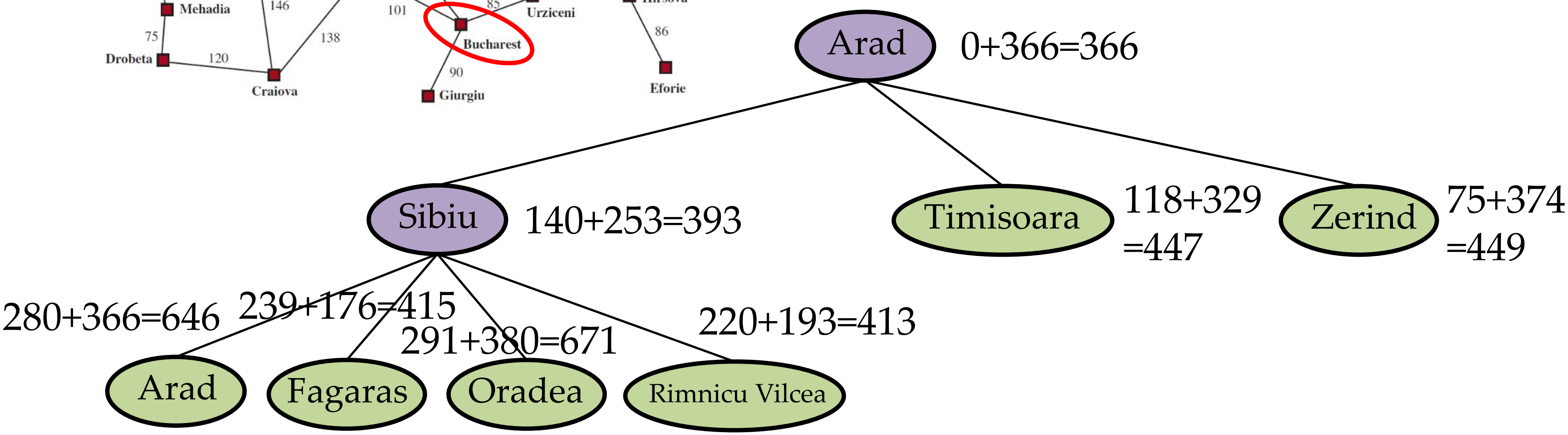
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



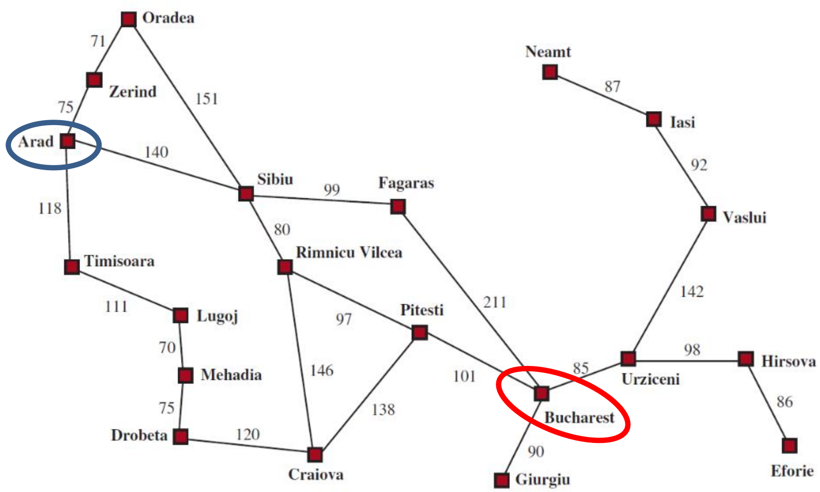
A* search - example



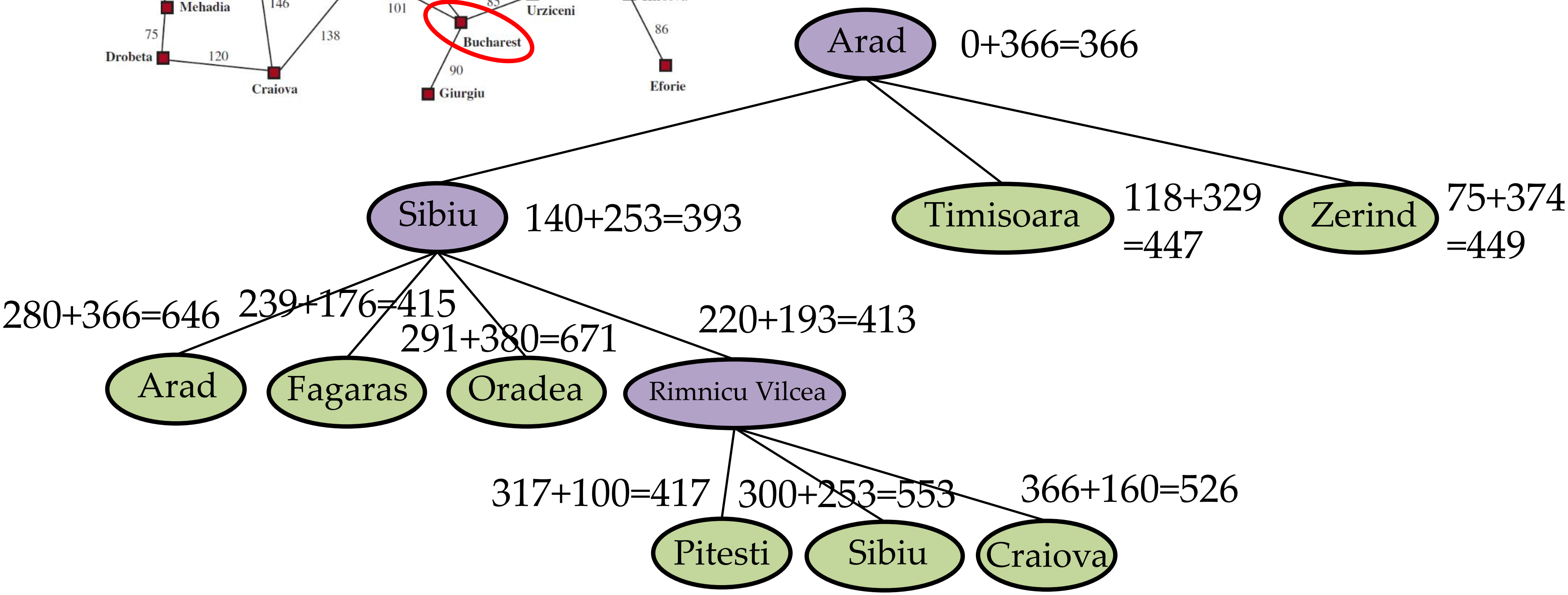
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



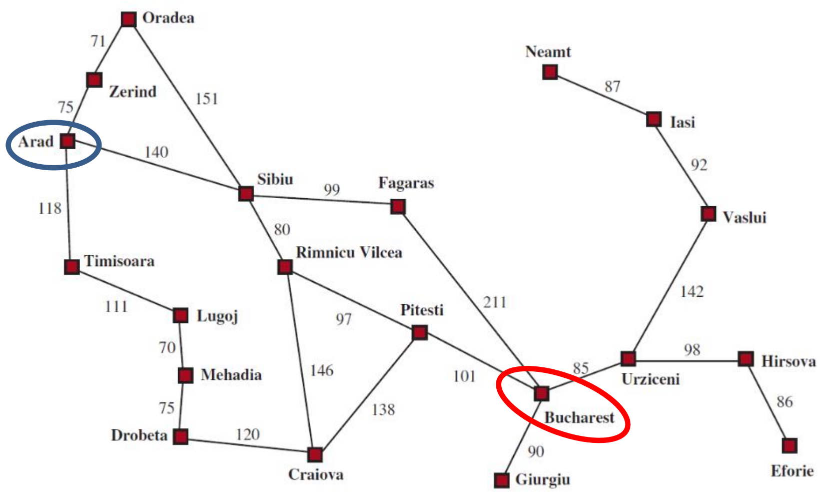
A* search - example



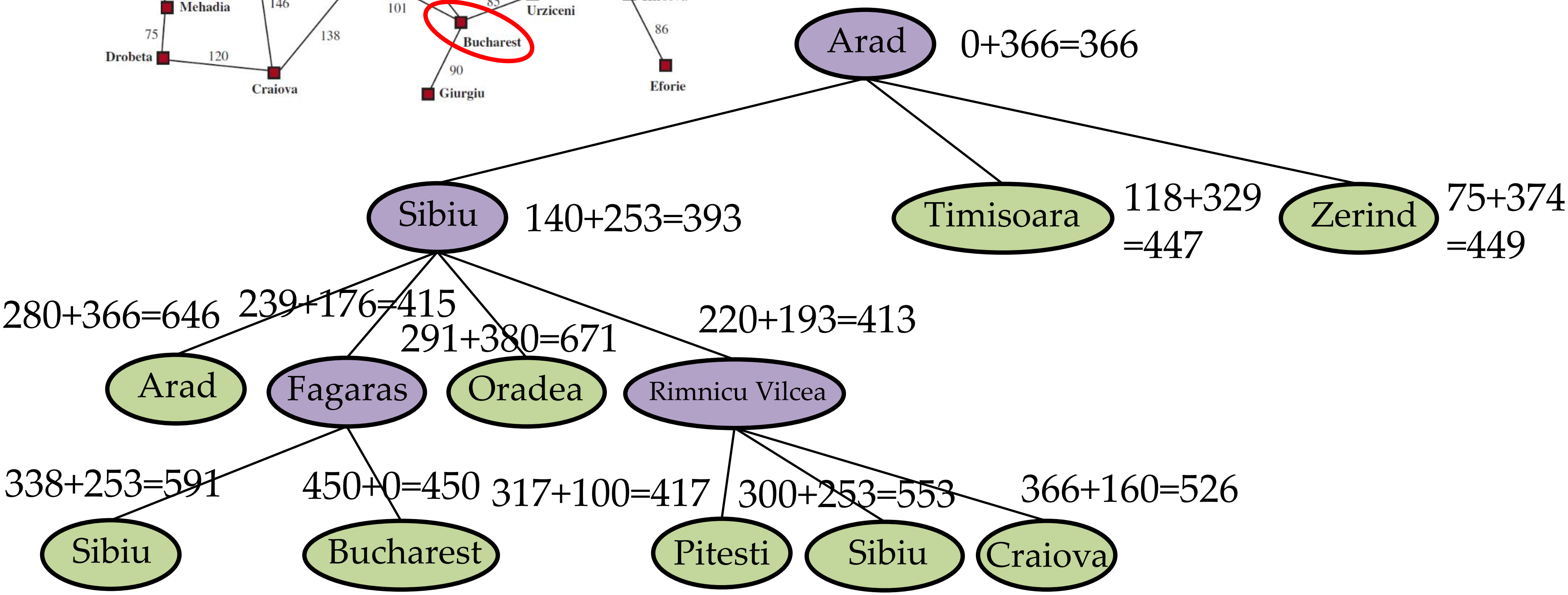
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



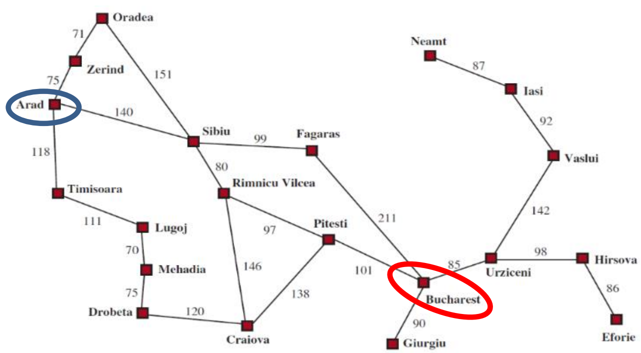
A* search - example



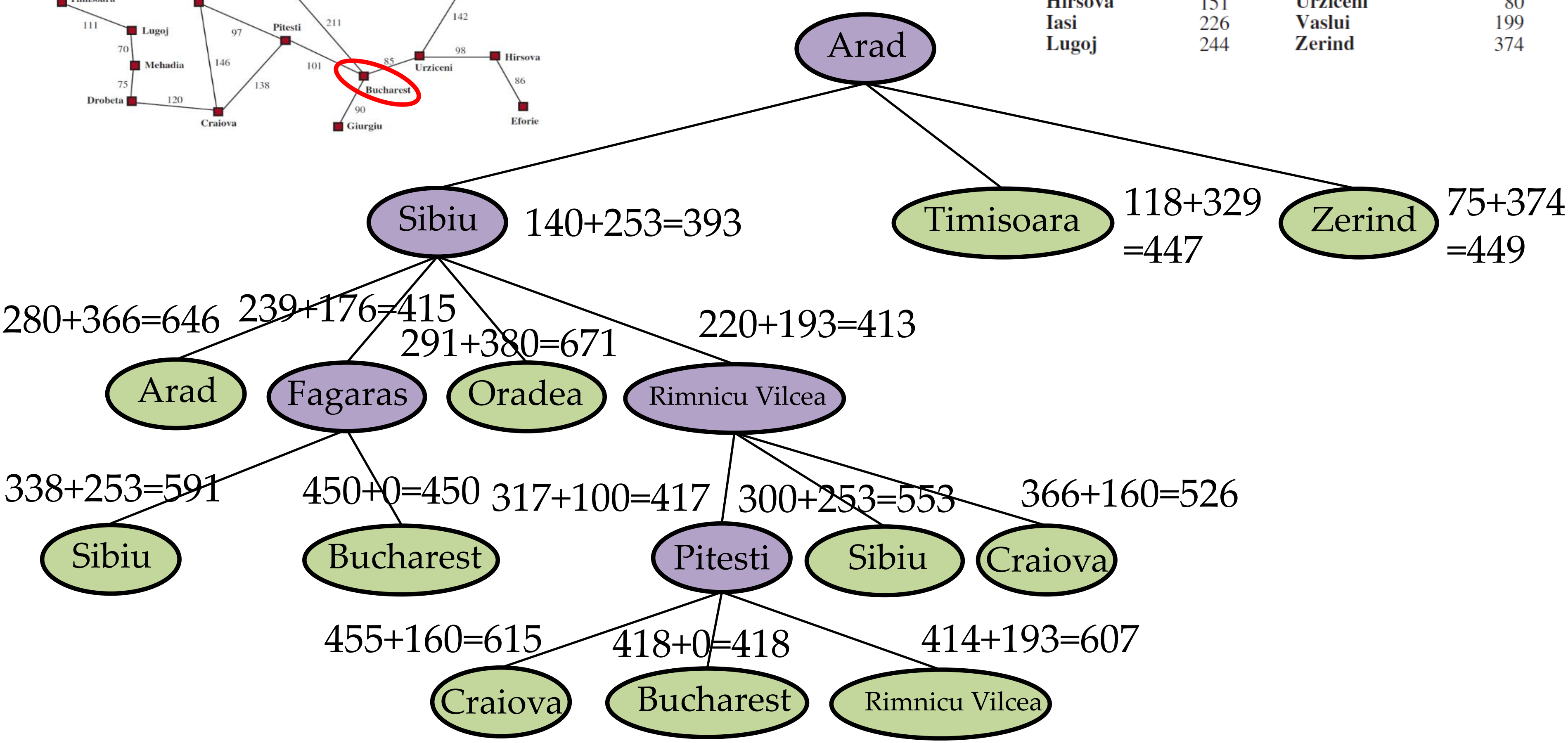
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



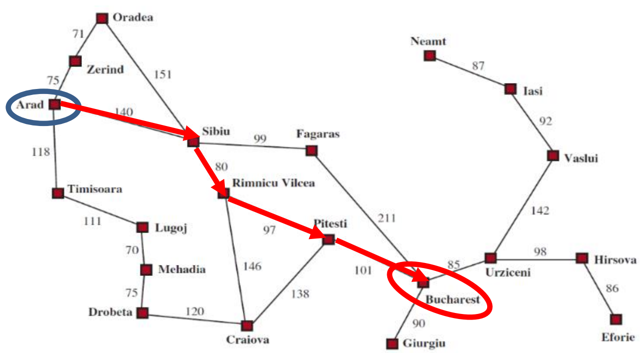
A* search - example



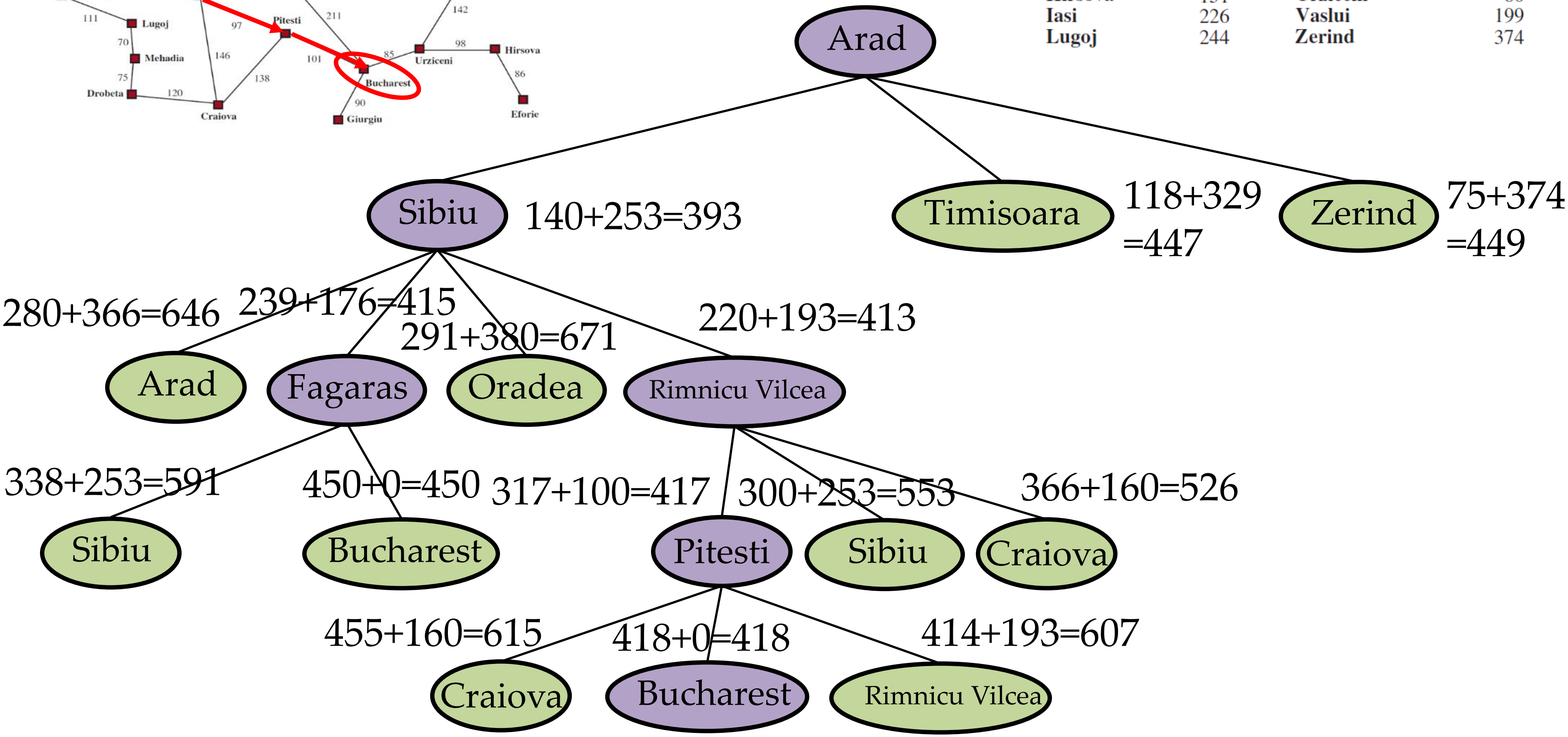
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



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



A* search - optimality

Main idea: expand the node n with the lowest value of

$$g(n) + h(n)$$

Path cost from the initial node to the node n

Estimated cost of the optimal path from the node n to a goal

- Optimal

  the tree-search version is optimal if $h(n)$ is admissible

A heuristic h is **admissible** if

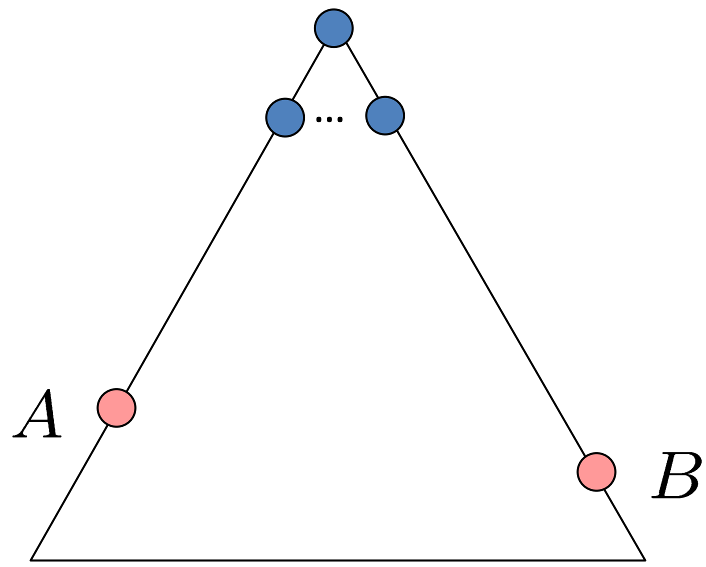
$$0 \leq h(n) \leq h^*(n)$$

where $h^*(n)$ is the true cost of the optimal path from n to a goal

Proof of optimality

Assume:

- A is an optimal goal node
- B is a suboptimal goal node
- h is admissible

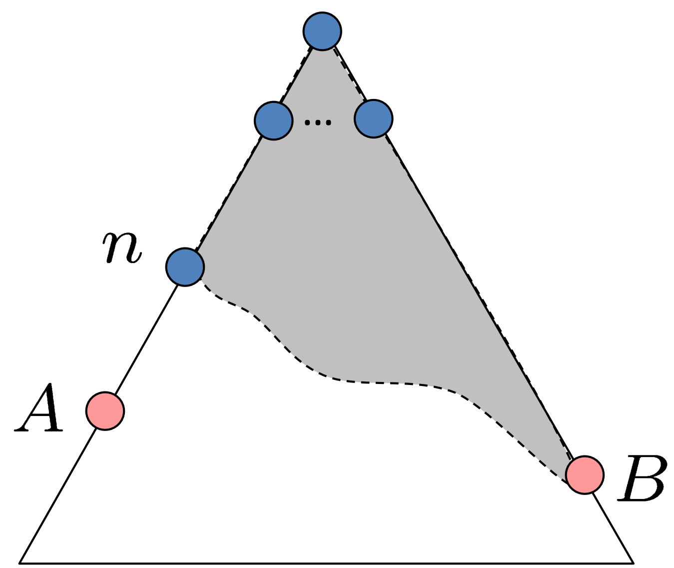


Claim:

- A will be expanded before B

Proof of optimality

- Currently, B is in the frontier
- Some ancestor n of A is in the frontier
- Claim: n will be expanded before B



To compare $g(n)+h(n)$ with $g(B)+h(B)$
and show $\boxed{g(n)+h(n) < g(B)+h(B)}$

Definition of A and B

$$g(A) < g(B)$$
$$h(A) = h(B) = 0$$

$$g(n)+h(n) \leq g(A)$$
$$= g(A)+h(A)$$

$$g(A)+h(A) < g(B)+h(B)$$

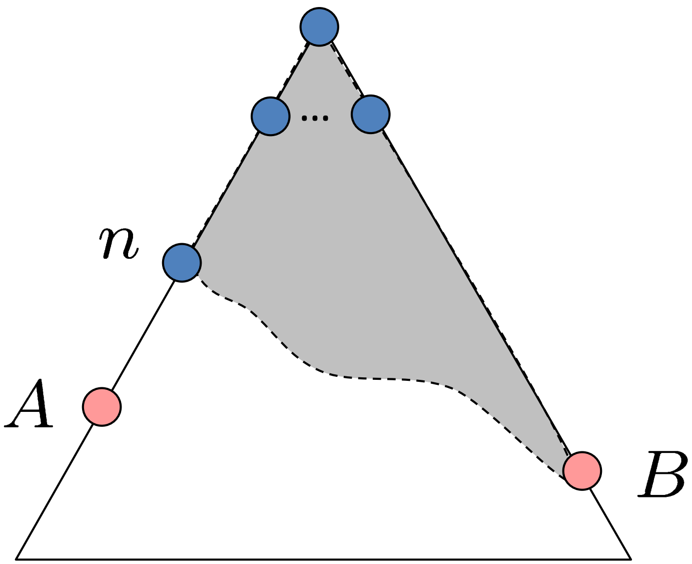
Admissibility of h $\quad h(A)=0$

Proof of optimality

- Currently, B is in the frontier
- Some ancestor n of A is in the frontier
- Claim: n will be expanded before B

$$\uparrow$$

$$g(n) + h(n) < g(B) + h(B)$$



All ancestors of A will be expanded before B

$$\Downarrow$$

A will be expanded before B

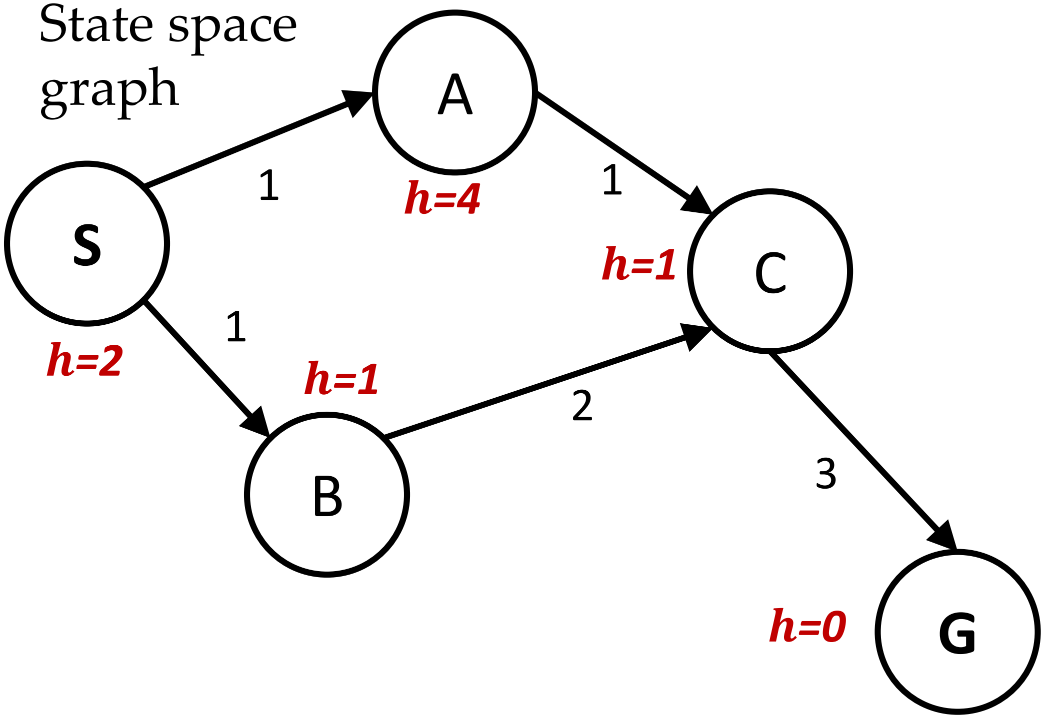
Optimal

A* search - optimality

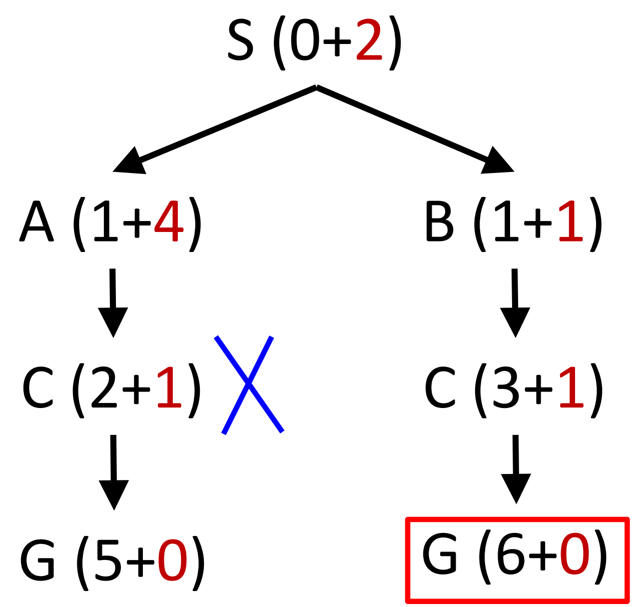
Admissible heuristic *h* $\Rightarrow$ The tree-search version of A* search is optimal

How about the graph-search version of A* search? Optimal?

State space graph



Search tree



A* search - optimality

Main idea: expand the node n with the lowest value of

$$g(n) + h(n)$$

Path cost from the initial node to the node n

Estimated cost of the optimal path from the node n to a goal

- Optimal

  the tree-search version is optimal if $h(n)$ is admissible

  the graph-search version is optimal if $h(n)$ is consistent

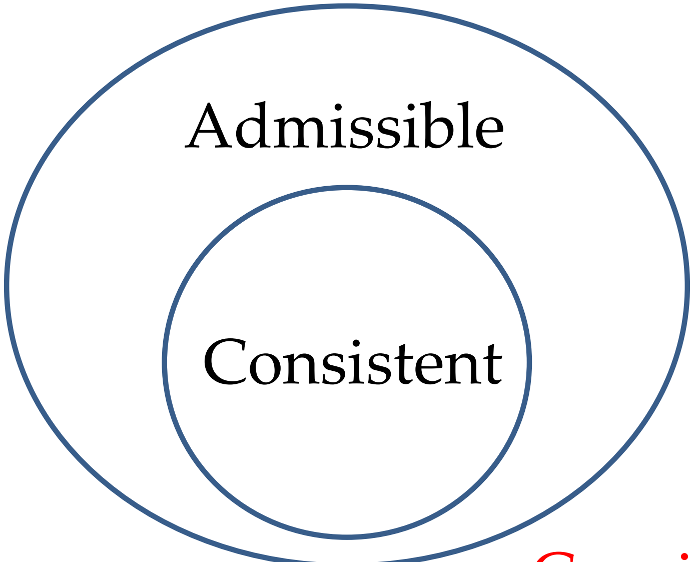
A heuristic h is **consistent** if

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

where n' is the successor of n generated by action a'

Admissible and consistent heuristic

Consistent $\Rightarrow$ Admissible



- $h(n)$: estimated cost of the optimal path from the node n to a goal
- $h^*(n)$: true cost of the optimal path from the node n to a goal
- Assume that the nodes on the optimal path are $n, n', n'', \dots, n^*$

Consistent

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

$$\le c(n, a', n') + c(n', a'', n'') + \cdots + h(n^*) = h^*(n)$$ Admissible

Consistent heuristic

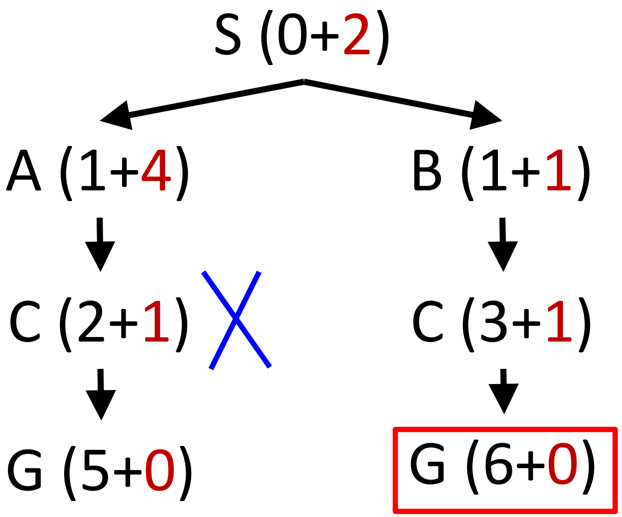
Claim: the values of $g(n) + h(n)$ along any path are non-decreasing

- Assume that n' is the successor of n generated by action a'

$$\begin{aligned} g(n') + h(n') &= g(n) + c(n, a', n') + h(n') \\ &\geq g(n) + h(n) \end{aligned}$$

Consistent

Search tree



Impossible for consistent heuristic

Proof of optimality

- The values of $g(n) + h(n)$ along any path are non-decreasing

Claim: when a goal node n is expanded, the optimal path to a goal has been found

Otherwise, there is another frontier node n' on the optimal path, whose corresponding goal node is denoted as n^*

$$g(n') + h(n') \leq g(n^*) + \overbrace{h(n^*)}^{0} < g(n) + \overbrace{h(n)}^{0}$$

$$\Downarrow$$

n' will be expanded before n, making a contradiction

A* search – performance

Main idea: expand the node n with the lowest value of

$$g(n) + h(n)$$

Path cost from the initial node to the node n

Estimated cost of the optimal path from the node n to a goal

- Optimal

  the tree-search version is optimal if $h(n)$ is admissible

  the graph-search version is optimal if $h(n)$ is consistent

Uniform-cost search: expand the node n with the lowest cost $g(n)$

Optimal

A special case of A* search with the heuristic $h(n) = 0 \longrightarrow$ Consistent

A* search – performance

Main idea: expand the node n with the lowest value of

$$g(n) + h(n)$$

- Optimal

  the tree-search version is optimal if $h(n)$ is admissible

  the graph-search version is optimal if $h(n)$ is consistent

- Complete (for consistent heuristic)

  if the branching factor b is finite and the cost (i.e., <u>the increment on $g + h$</u>) of every step exceeds some small positive constant ϵ

A* search expands nodes with non-decreasing values of $g(n) + h(n)$

A* search – performance

- Optimal

  the tree-search version is optimal if $h(n)$ is admissible

  the graph-search version is optimal if $h(n)$ is consistent

- Complete (for consistent heuristic)

  if the branching factor b is finite and "the cost" (i.e., <u>the increment on $g + h$</u>) of every step exceeds some small positive constant ϵ

- Complexity (for consistent heuristic)

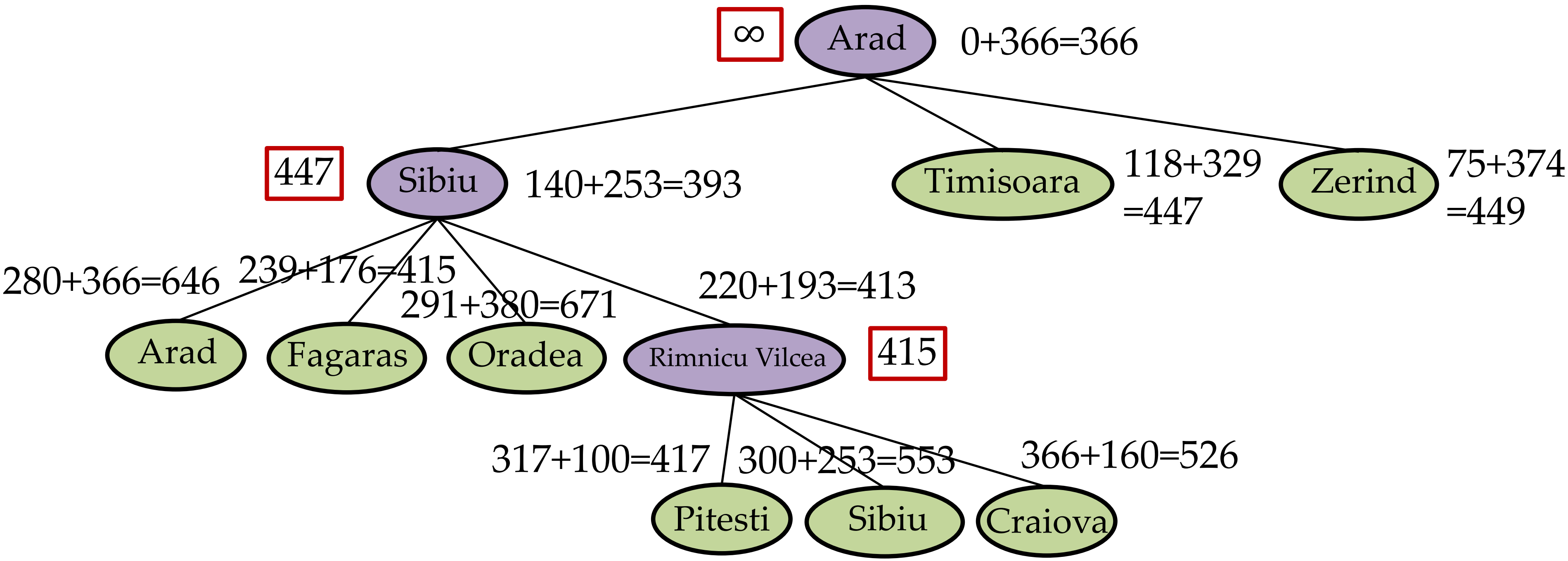
  expand all nodes with $g(n) + h(n) < C^*$ and some nodes with $g(n) + h(n) = C^*$, where C^* is the cost of the optimal path

  Time: optimally efficient for any given consistent heuristic

  Space: can still be exponential $\longrightarrow$ How to reduce?

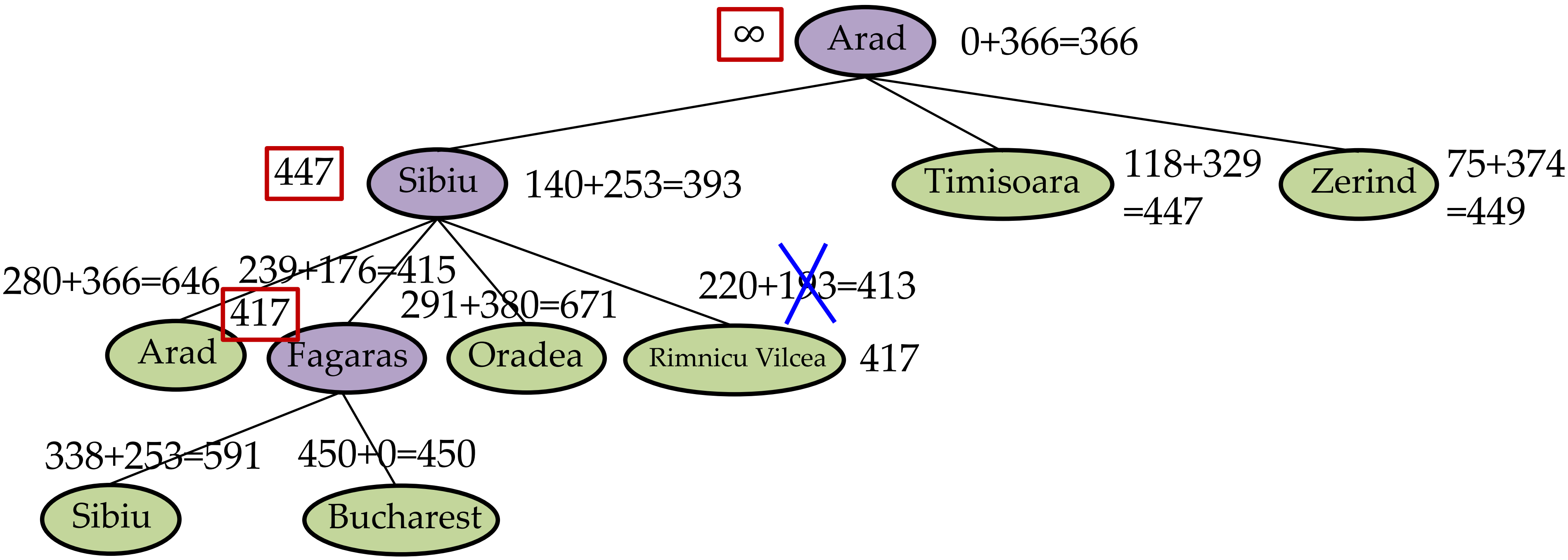
Recursive best-first search

Main idea: for each node on the path, record the $g + h$ value of the best alternative path available from any ancestor of the node



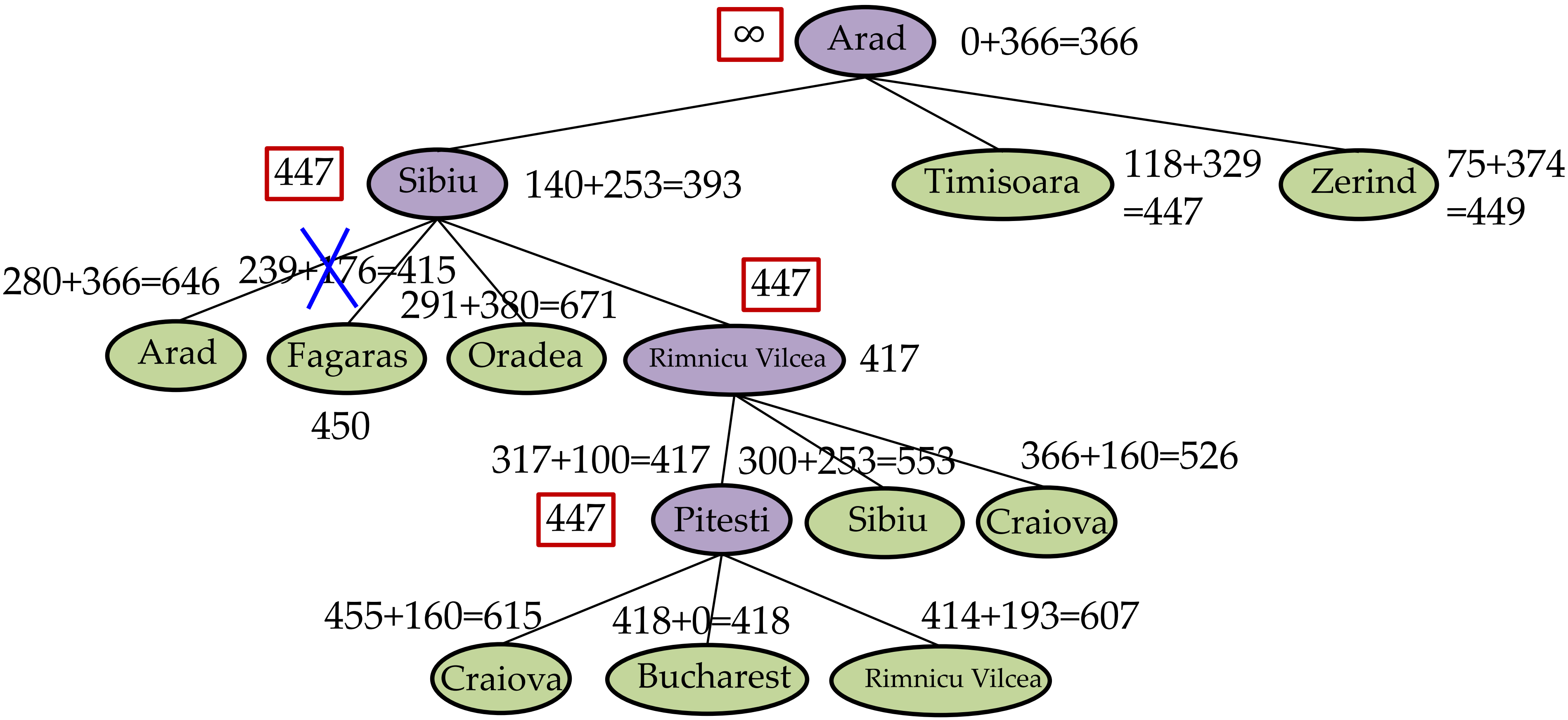
Recursive best-first search

Main idea: for each node on the path, record the $g + h$ value of the best alternative path available from any ancestor of the node



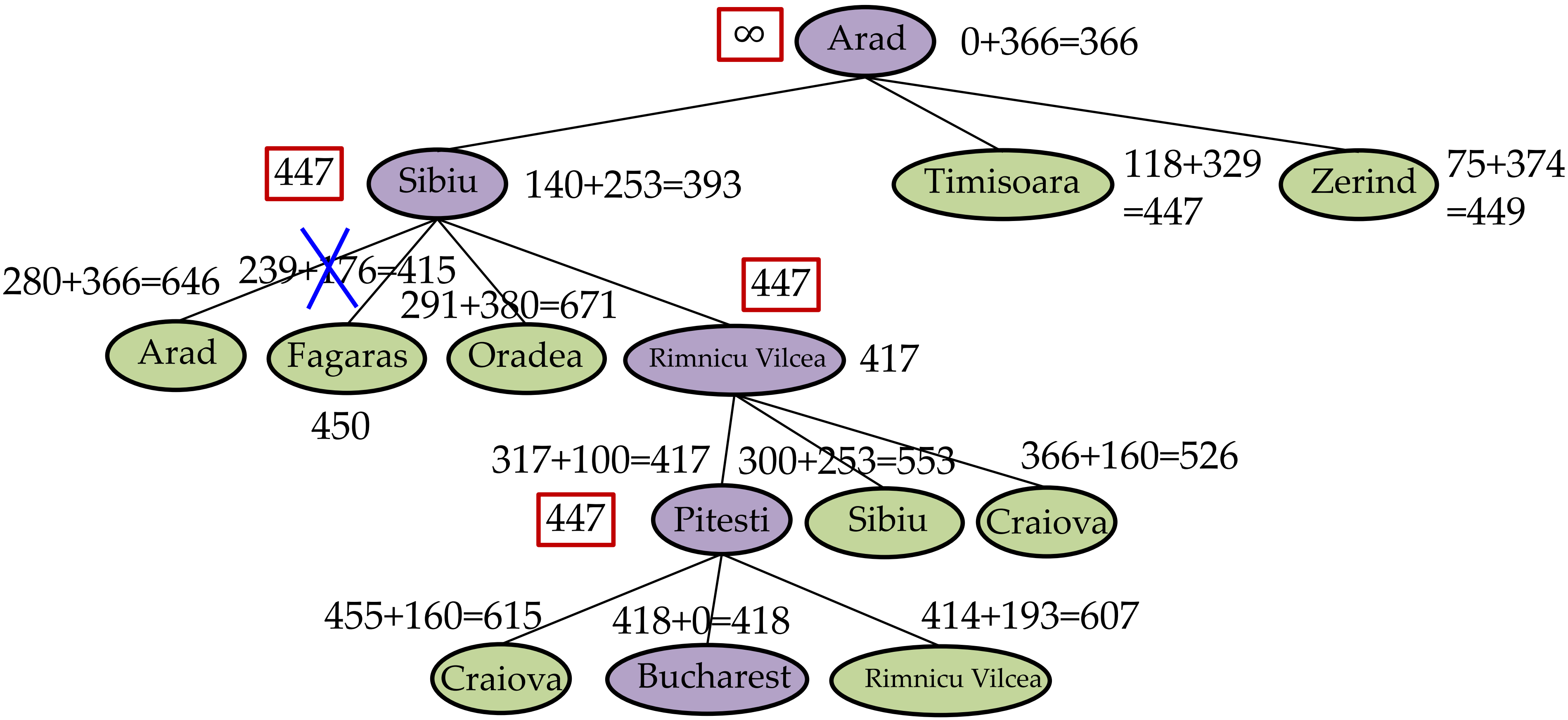
Recursive best-first search

Main idea: for each node on the path, record the $g + h$ value of the best alternative path available from any ancestor of the node



Recursive best-first search

Main idea: for each node on the path, record the $g + h$ value of the best alternative path available from any ancestor of the node



Recursive best-first search

Main idea: for each node on the path, record the $g + h$ value of the best alternative path available from any ancestor of the node;

if the best $g + h$ value of the generated nodes exceeds the recorded value of the current expanded node, unwind back to the alternative path

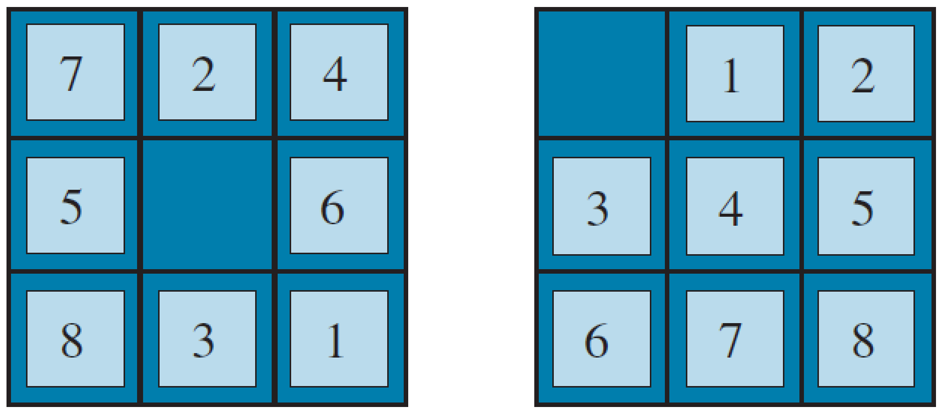
- The search process is the same as A* search
- The space complexity is linear in the depth of the deepest optimal path
- The time complexity increases because of node reexpansion

Heuristic generation from relaxed problems

Main idea: $h(n)$ = the cost of the optimal solution of a relaxed problem

Compared with the original problem, there are fewer restrictions on the actions

Example: 8-puzzle



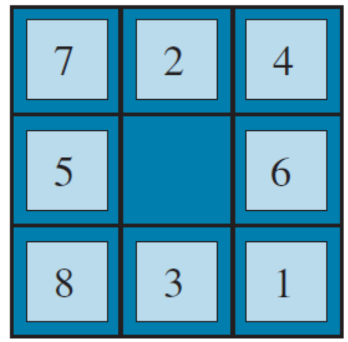
Start State Goal State

Heuristic generation from relaxed problems

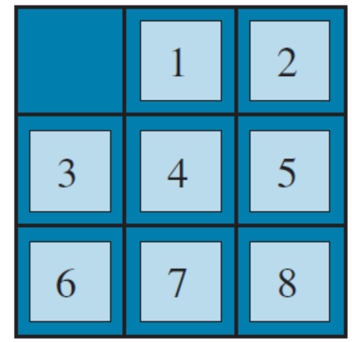
Example: 8-puzzle

$h_1 = 8$

$h_2 = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$



Start State



Goal State

A tile can move from square A to square B

Original: if A is adjacent to B and B is blank

Relaxed:

- if A is adjacent to B → h_2 = the sum of the $\boxed{\text{distances}}$ of the tiles from their goal positions
- if B is blank
- No restriction → h_1 = the number of misplaced tiles

Manhattan distance

Heuristic generation from relaxed problems

Main idea: $h(n)$ = the cost of the optimal solution of

$\boxed{\text{a relaxed problem}}$

Compared with the original problem, there are fewer restrictions on the actions

A supergraph of the original state space

$$h(n) \begin{cases} \text{Admissible: } 0 \leq h(n) \leq h^*(n) \\ \text{Consistent: } h(n) \leq c(n, a', n') + h(n') \end{cases}$$

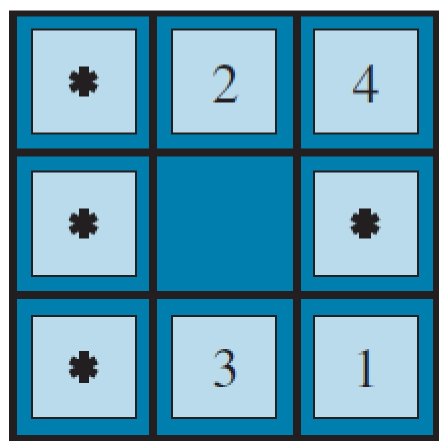
$h(n)$ should be easy to be computed

Heuristic generation from subproblems

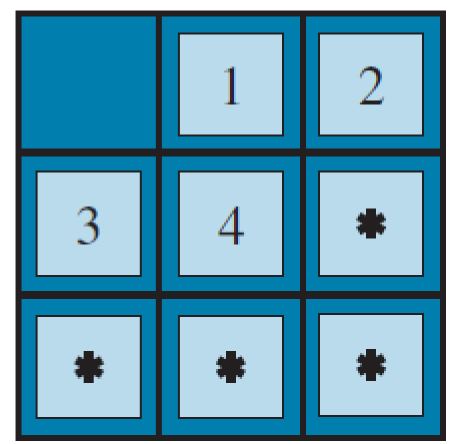
Main idea: $h(n)$ = the cost of the optimal solution of $\boxed{\text{a subproblem}}$

Compared with the original problem, there are more goal states, containing the original ones

Example: 8-puzzle



Start State



Goal State

Heuristic generation from subproblems

Main idea: $h(n)$ = the cost of the optimal solution of

a subproblem

Compared with the original problem, there are more goal states, containing the original ones

$$h(n) \begin{cases} \text{Admissible: } 0 \le h(n) \le h^*(n) \\ \text{Consistent: } h(n) \le c(n, a', n') + h(n') \end{cases}$$

Heuristic generation from new problems

Main idea: $h(n)$ = the cost of the optimal solution of a new problem

Consistent: $h(n) \le c(n, a', n') + h(n')$?

$\Uparrow$

The actions of the original problem can be performed for the new problem

Heuristic generation - example

Traveling salesman problem: to find the shortest route to visit each city exactly once, starting and ending in the same city

relax

Minimum spanning tree: to find a connected subgraph (connecting all the cites) with the minimum sum of edge weights

- Actions: *can go to non-visited cities, and return to the origin city at last*
- Goal states: *In(the start city), Visited({all the cities})*

relax

- Actions: *can go to any city, where the cost of an already performed action is 0*
- Goal states: *In(any city), Visited({all the cities})*

Heuristic generation by learning

The start state

The cost of the optimal solution

$\boldsymbol{x_1}$

7	2	4
	5	6
8	3	1

$\longrightarrow$ y_1

⋮

$\boldsymbol{x_n}$

5	2	4
7		3
8	1	6

$\longrightarrow$ y_n

Learning algorithms $\Rightarrow$ $h(n)$: not necessarily admissible

Goodness of heuristic

A* search $\begin{cases} \text{heuristic } h_1 \\ \\ \text{heuristic } h_2 \end{cases}$

Which one is good?

Performance measure: the number N of generated nodes for the same problem instance

⇕

Effective branching factor b^*

the depth of the optimal solution

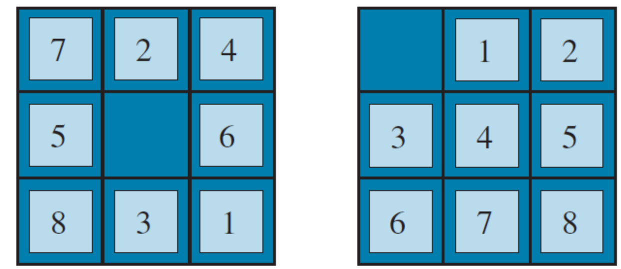
$$N = b^* + (b^*)^2 + \cdots + (b^*)^d$$

Goodness of heuristic

Example: 8-puzzle

h_1 = the number of misplaced tiles

h_2 = the sum of the distances of the tiles from their goal positions



Start State　　Goal State

	Effective branching factor b^*		
d	Iterative deepening DFS	$A^*(h_1)$	$A^*(h_2)$
2	2.45	1.79	1.79
4	2.87	1.48	1.45
6	2.73	1.34	1.30
8	2.80	1.33	1.24
10	2.79	1.38	1.22
12	2.78	1.42	1.24
14	-	1.44	1.23
16	-	1.45	1.25
18	-	1.46	1.26
20	-	-	-

For each d, the average of 100 random instances

h_2 is better than h_1

Informed search is better

Goodness of heuristic

A heuristic h_2 **dominates** another heuristic h_1 if

$$\forall n:\ h_2(n) \geq h_1(n)$$

- A* search with consistent heuristic

  expand all nodes with $g(n) + h(n) < C^*$ *and some nodes with* $g(n) + h(n) = C^*$*, where* C^* *is the cost of the optimal path*

$$g(n) + h_2(n) < C^* \Leftrightarrow h_2(n) < C^* - g(n)$$

$$\Downarrow$$

A* search with h_1 will generally expand more nodes $\Leftarrow$ $h_1(n) < C^* - g(n)$

Goodness of heuristic

A heuristic h_2 **dominates** another heuristic h_1 if

$$\forall n\colon\ h_2(n) \geq h_1(n)$$

$$\Downarrow$$

h_2 is generally better than h_1, i.e., A* search with h_2 will expand less nodes in general

Goodness of heuristic

$$\left.\begin{array}{l} h_1(n), h_2(n), \dots, h_m(n) \\ \text{Any two are non-dominated} \end{array}\right\} \begin{array}{l} \text{Composite heuristic:} \\ \max\{h_1(n), h_2(n), \dots, h_m(n)\} \end{array}$$

- The composite heuristic dominates any component heuristic $h_i(n)$

$$\forall n:\ \max\{h_1(n), h_2(n), \dots, h_m(n)\} \geq h_i(n)$$

- If $\forall i: h_i(n)$ is admissible, the composite heuristic is admissible

$$\forall n:\ \max\{h_1(n), h_2(n), \dots, h_m(n)\} \leq h^*(n)$$

- If $\forall i: h_i(n)$ is consistent, the composite heuristic is consistent

$$\max\{h_1(n), \dots, h_m(n)\} \leq c(n, a', n') + \max\{h_1(n'), \dots, h_m(n')\}$$

Summary

- Greedy best-first search
- A* search
- Recursive best-first search
- Heuristic generation
- Heuristic goodness

Informed (heuristic) search

Uses problem-specific knowledge beyond the problem definition

References

- S. J. Russell and P. Norvig. Artificial Intelligence: A Modern Approach. Chapter 3.5-3.6, Third edition.

Assignment - 1

Task: apply uninformed/informed search algorithms to play the game of Pacoban-2players

Deadline: Oct. 23