Optimal Search Algorithm

Faculty of DS & AI
Autumn semester, 2025

Trong-Nghia Nguyen

Business AI Lab

2025-09

Content

- Optimal search
  - Definition
  - Greedy search
  - A* search
  - Properties of Heuristic Function

Content

- Optimal search
  - Definition
  - Greedy search
  - A* search
  - Properties of Heuristic Function

Optimal Search

In practice problems

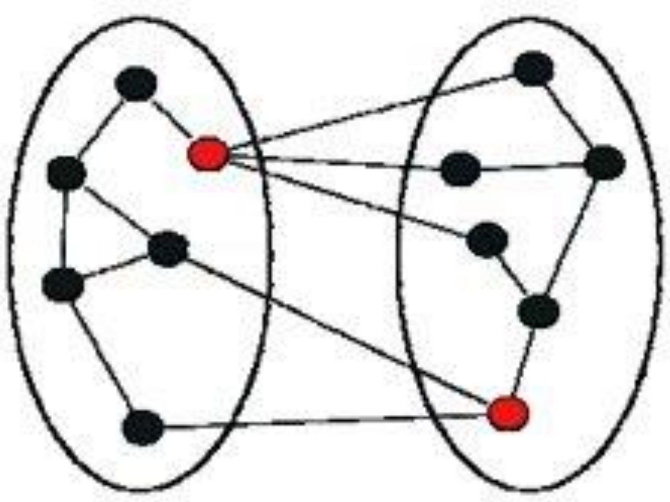
- We are often not only interested in finding a solution, but also whether the solution is optimal.
- Example:
    - Shortest path finding: consider the path cost..
    - 8-puzzle: consider the minimum number of moves to reach the goal.

👉 **In uninformed search and informed (heuristic) search, we have not yet considered path length or cost.**

Optimal Search

Graph Partition Problem:

- Given a graph, divide it into nnn equal-sized subsets such that the number of edges between subsets is minimized.



Optimal Search

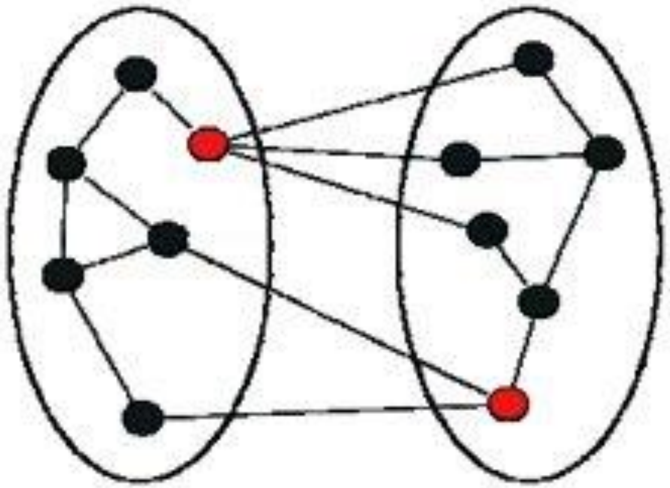
Graph Partition Problem:

- Given a graph, divide it into nnn equal-sized subsets such that the number of edges between subsets is minimized.

  - Each partition $G(V,E) \rightarrow \{G_1(V_1,E_1),G_2(V_2,E_2)\}$ is a **state**. A state can be represented by a **binary array**:
    - 0 -> vertex in group 1.
    - 1 -> vertex in group 2.
  - Example: we have state: u = $[0100011011]$
    - Group 1: $\{1,3,4,5,8\}$
    - Group 2: $\{2,6,7,9,10\}$
  - Evaluation function:
    - $F(u) = |V_1 - V_2| +$ number of cross edges (connected edge)
    - $|V_1 - V_2|$: balance term (equal partition).
    - Cross edges: edges between different groups.

❿ 👉 The goal is to find u* with **minimum $F(u)$**

❿ 👉 Optimal search = finding state u such that **f(u) is minimized**.



Optimal Search

Compare with Heuristic search

Criteria	**Heuristic Search**	**Optimal Search**
Evaluation	Based on heuristic $h(n)$	Based on total cost $g(n) + h(n)$
Goal	Find a solution quickly	Find the best (optimal) solution
Optimal guarantee	❌ No	✅ Yes (if conditions hold)
Example	Greedy Best-First Search	A*, Branch-and-bound search

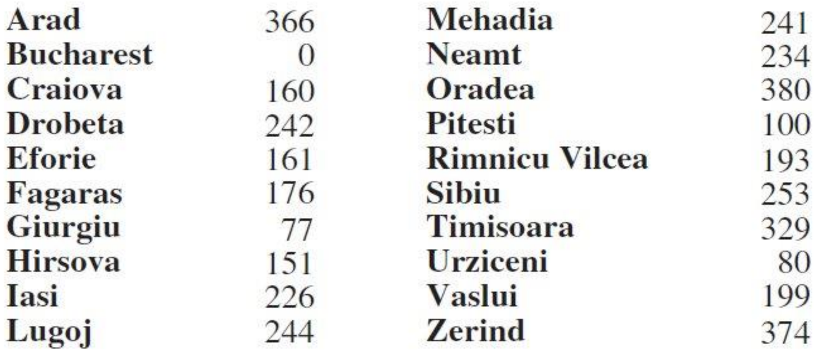
Content

- Optimal search
  - Definition
  - Greedy search
  - A* search
  - Properties of Heuristic Function

Optimal Search

Romania road map (textbook)

Arad -> 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

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.

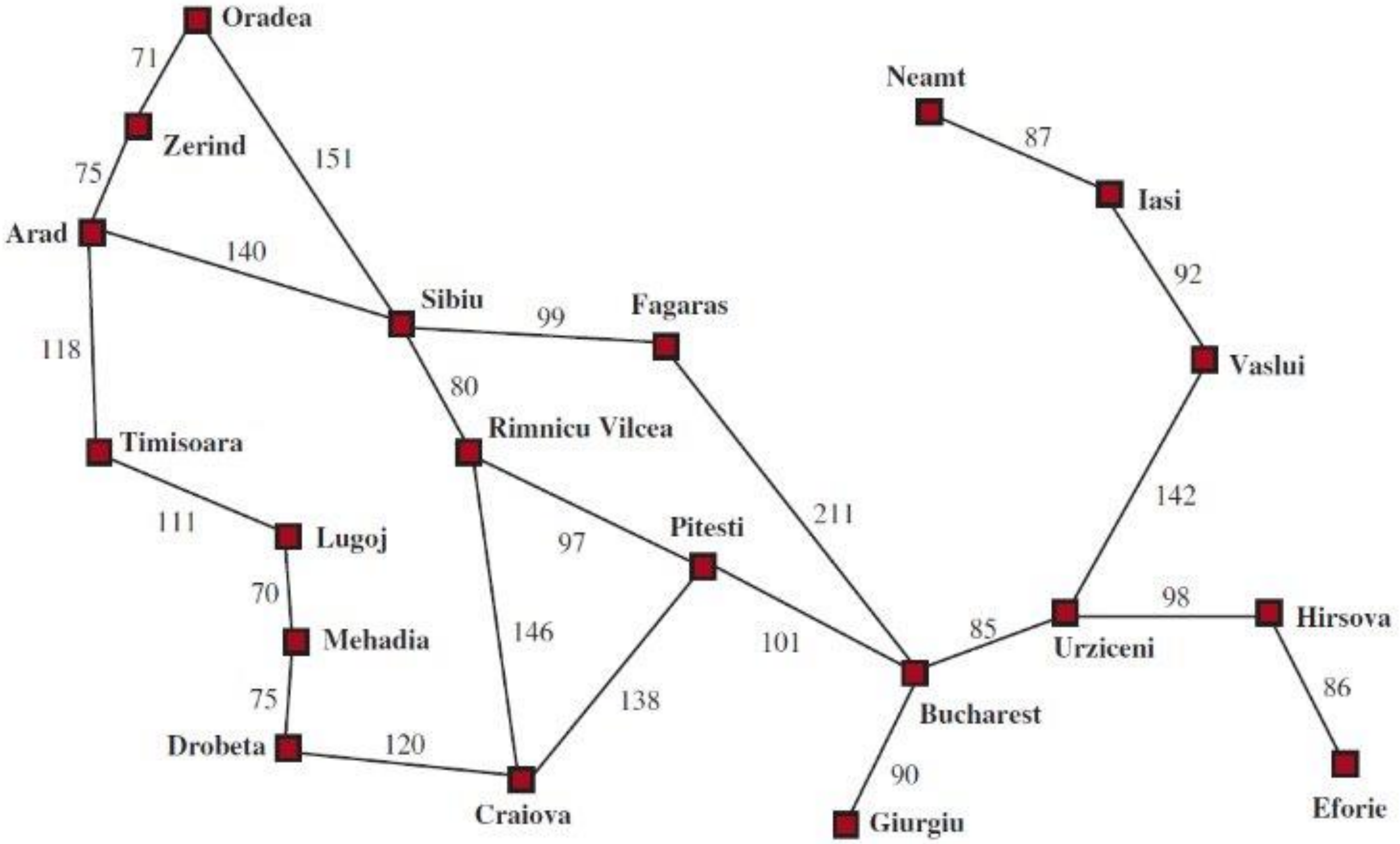


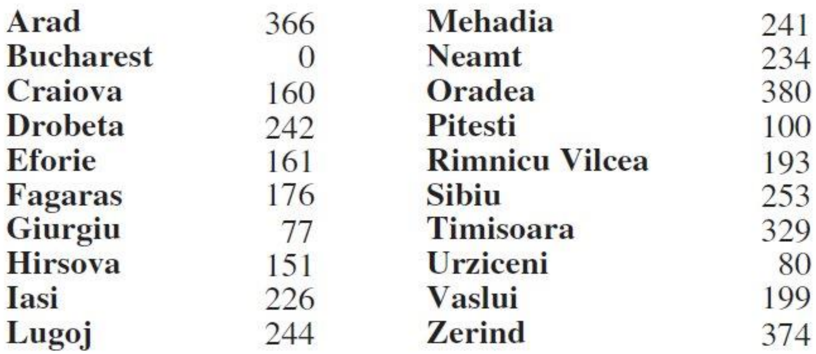
Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

Optimal Search

Romania road map (textbook)

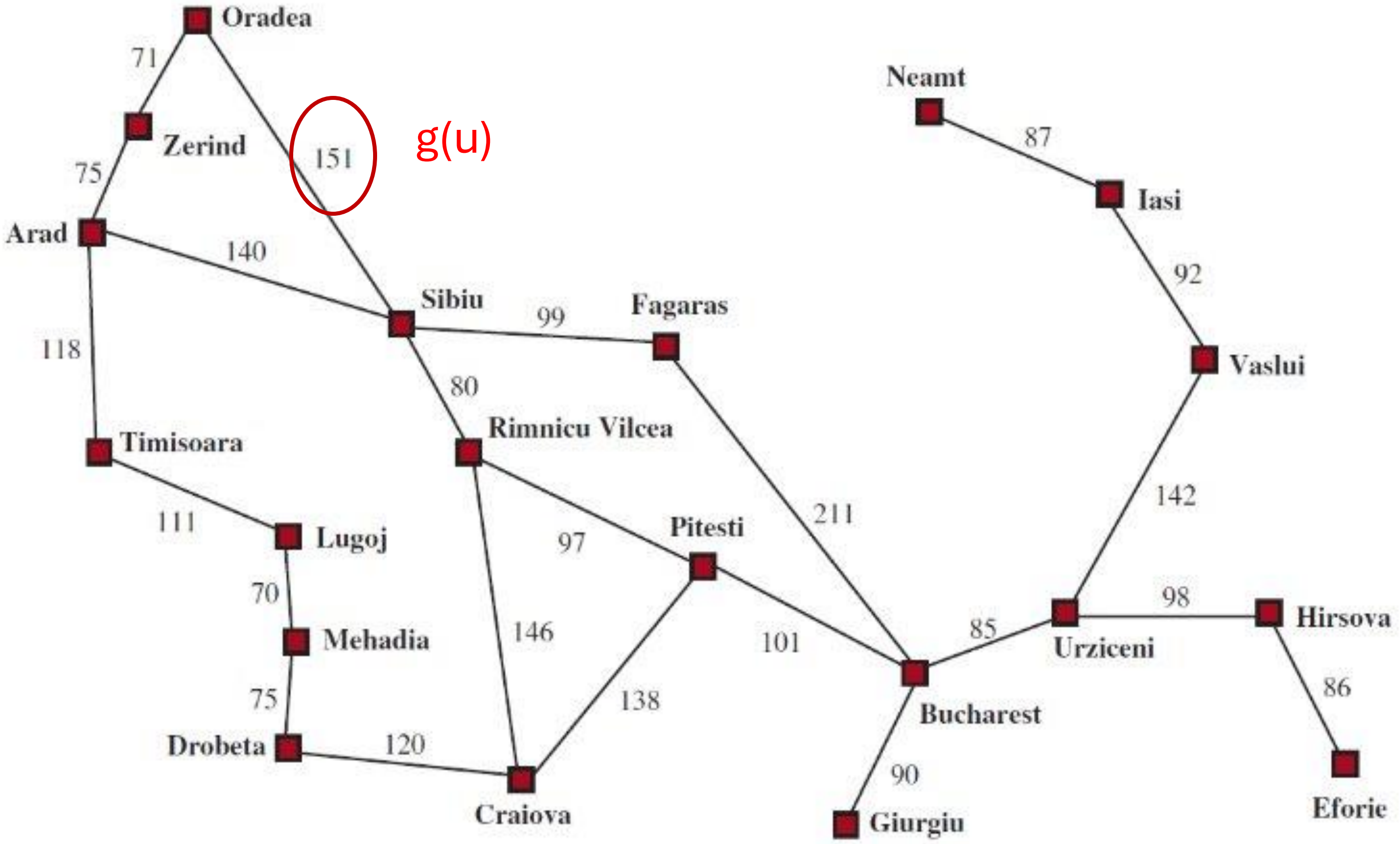
Arad -> Bucharest

h(u)



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

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.



g(u)

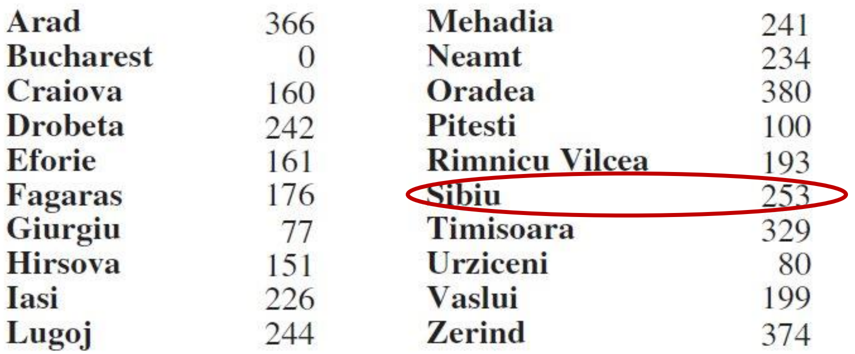
Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

Optimal Search

Romania road map (textbook)

Arad -> Bucharest

h(u)



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

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.

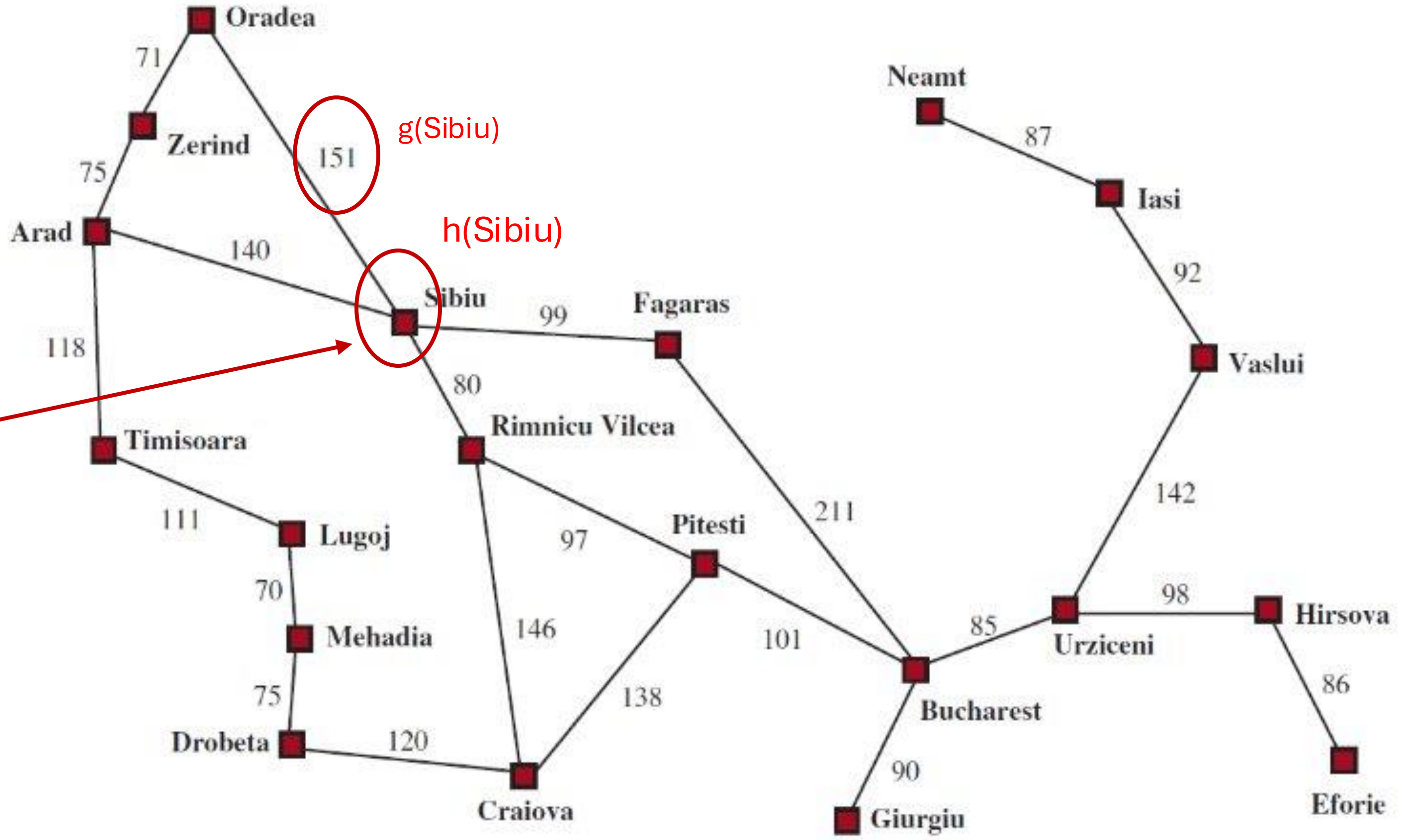
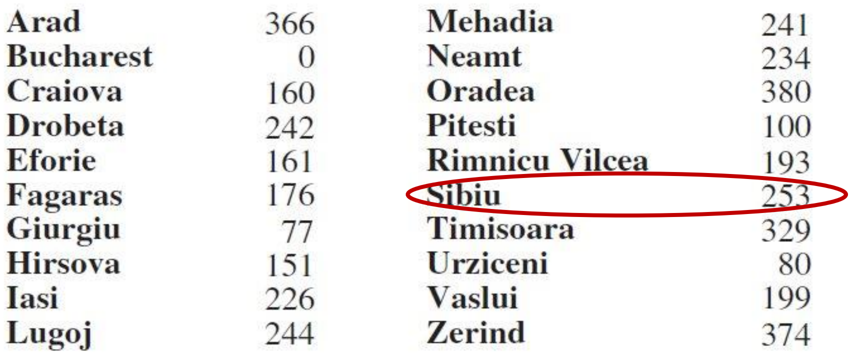


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

Optimal Search

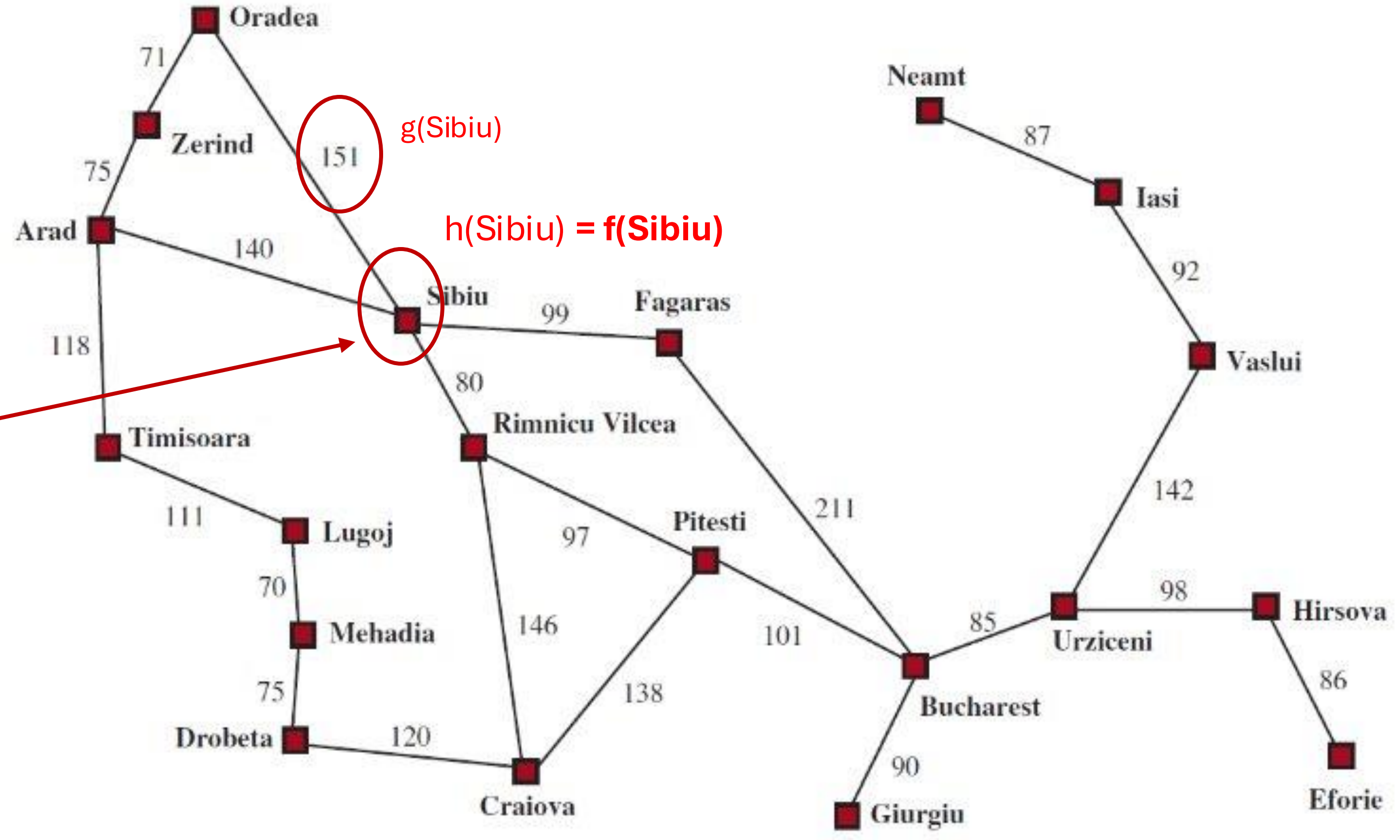
Greedy search

h(u)



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

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.



g(Sibiu)

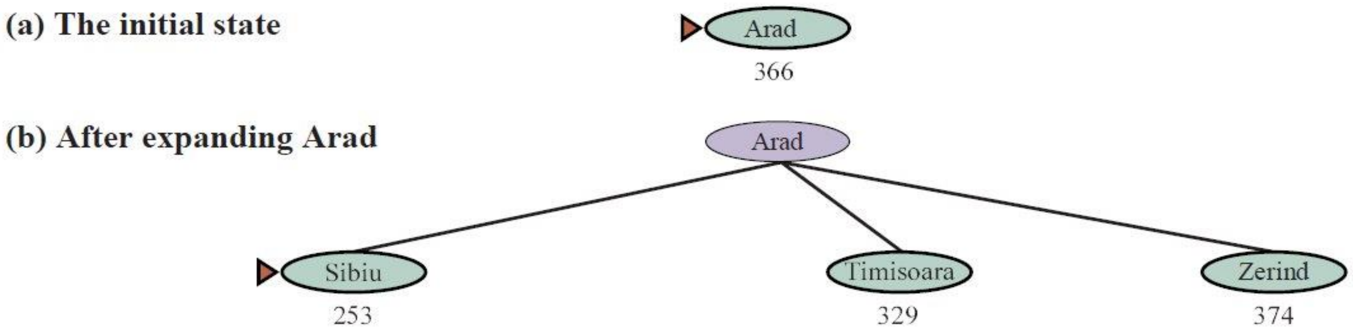
h(Sibiu) = **f(Sibiu)**

Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

- select the node with the minimum value of h(u)
- hill-climbing search

Optimal Search

Greedy search



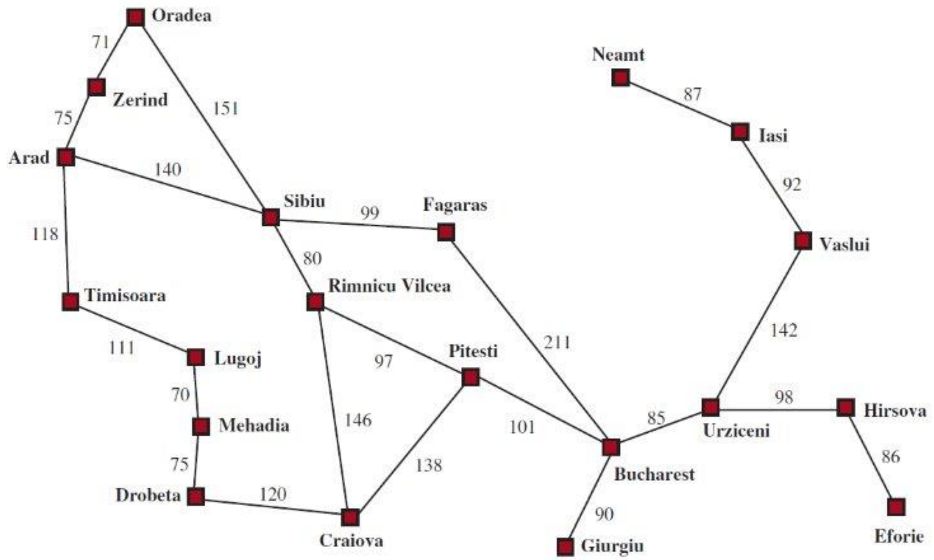


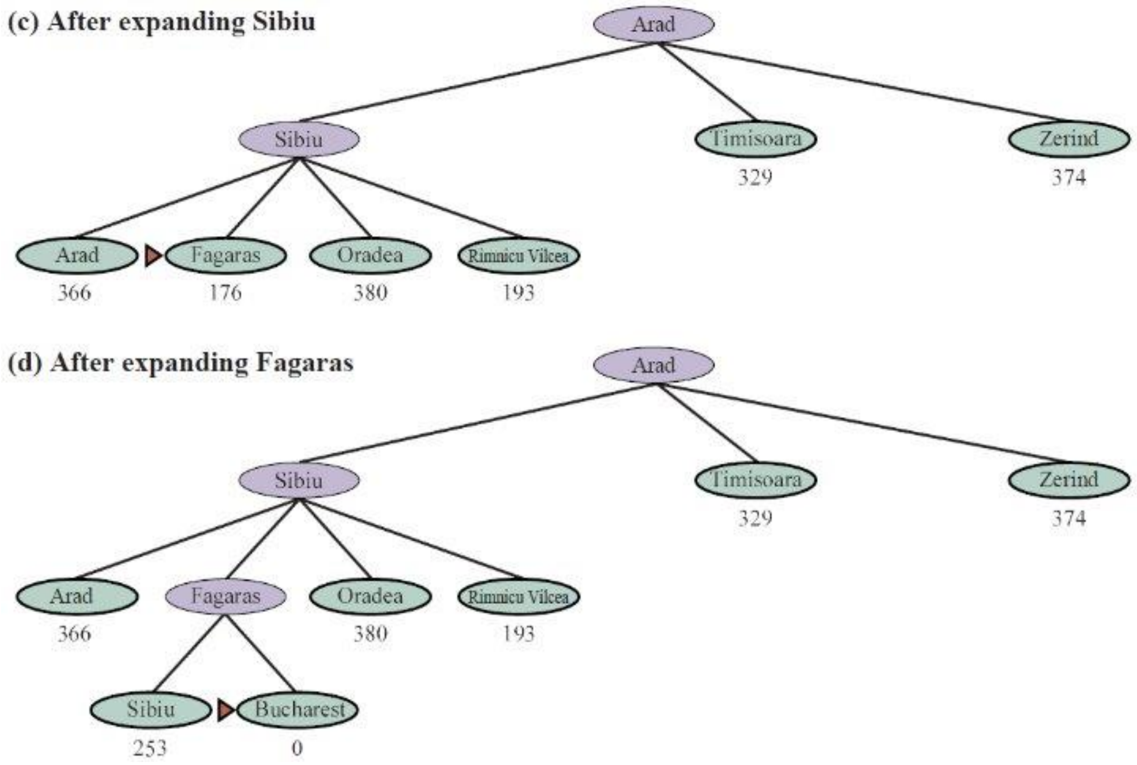
Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

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

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.

Optimal Search

Greedy search



(c) After expanding Sibiu

(d) After expanding Fagaras

Path cost for the solution =
140+99+211 = 450 miles

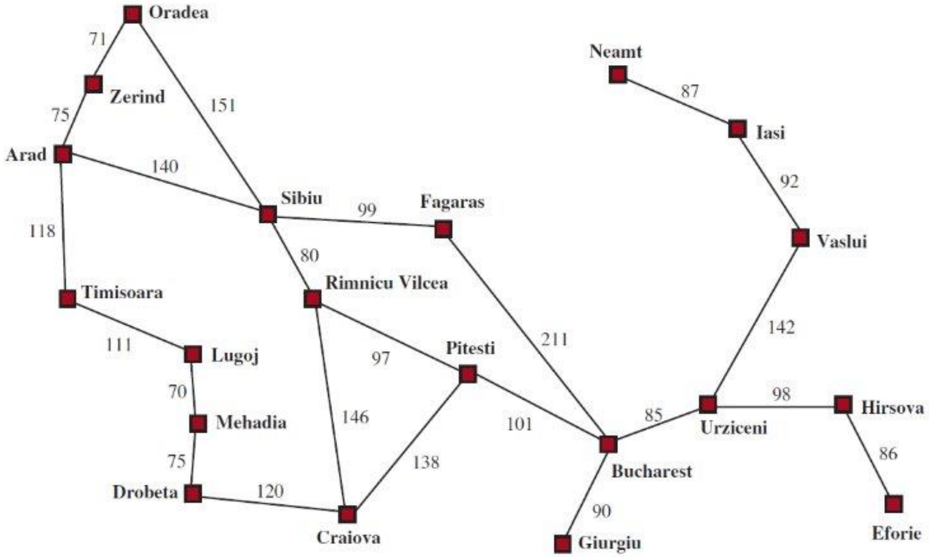
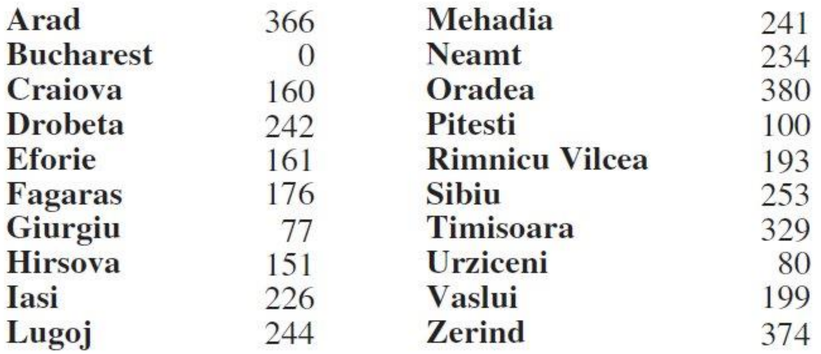


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.



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

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.

Content

- Optimal search
  - Definition
  - Greedy search
  - A* search
  - Properties of Heuristic Function

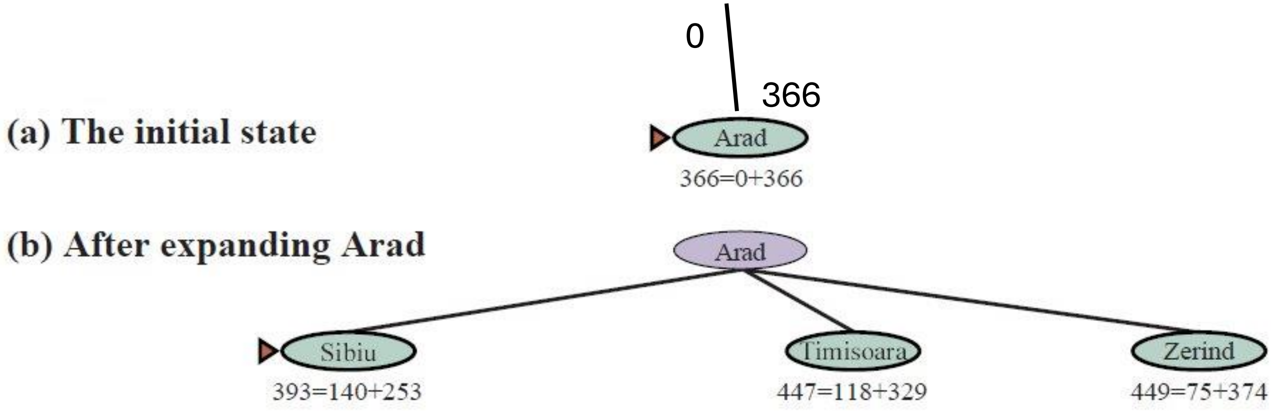
Optimal Search

A* search

- select the node with the minimum value of

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

$$f(n) = \text{ estimated cost of the cheapest solution through } n$$



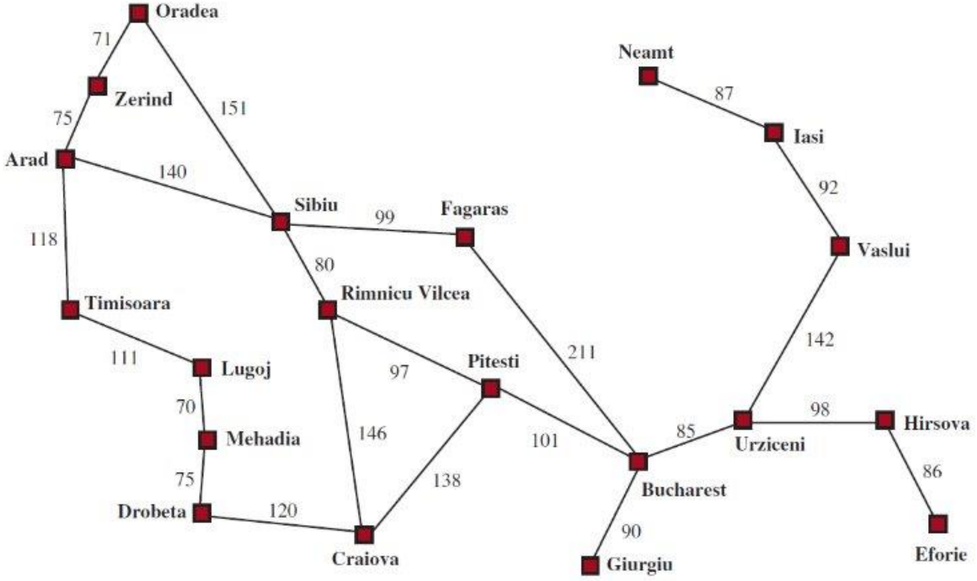
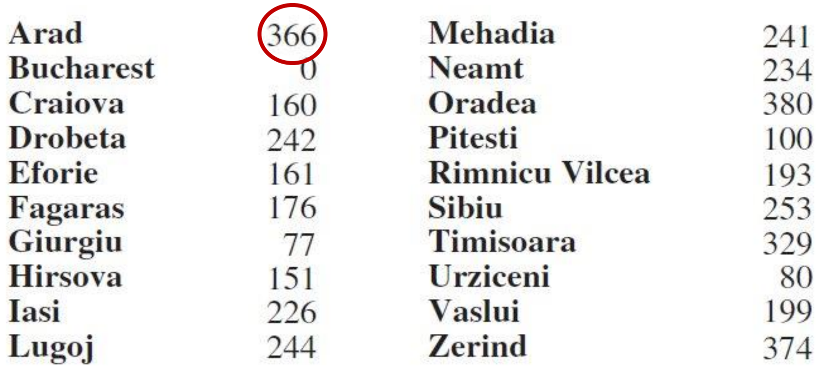


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

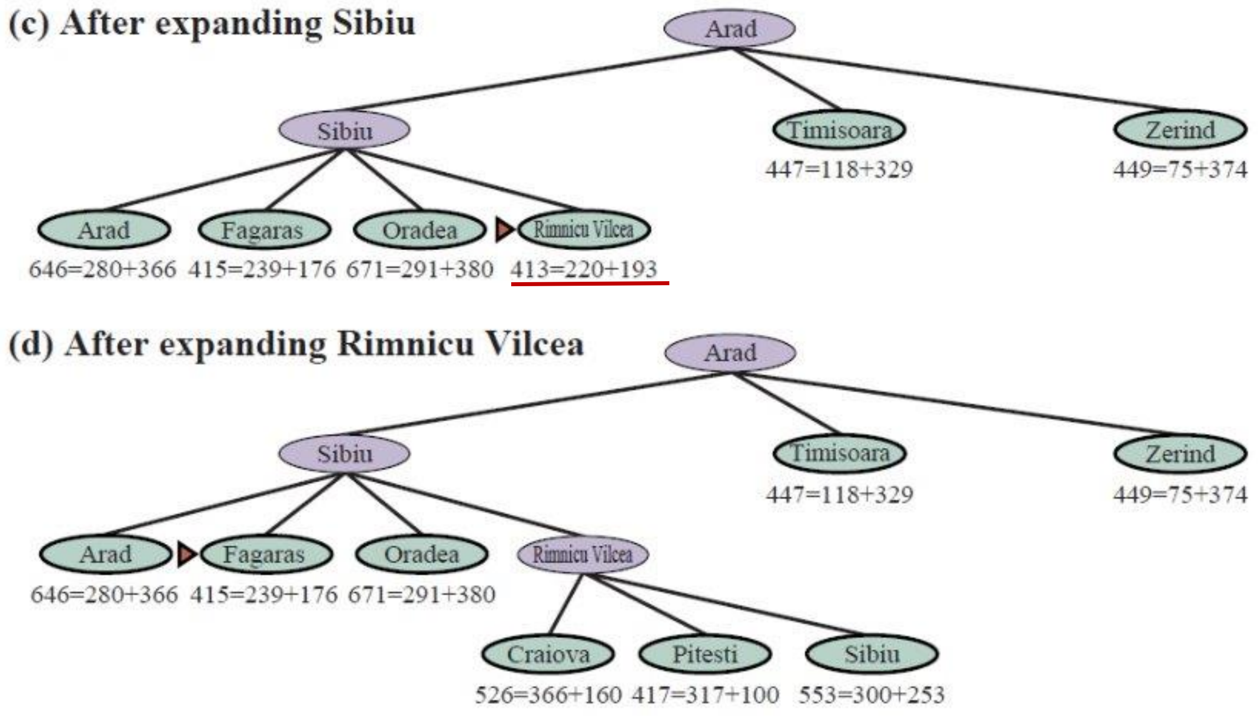


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.

Optimal Search

A* search



(c) After expanding Sibiu

(d) After expanding Rimnicu Vilcea

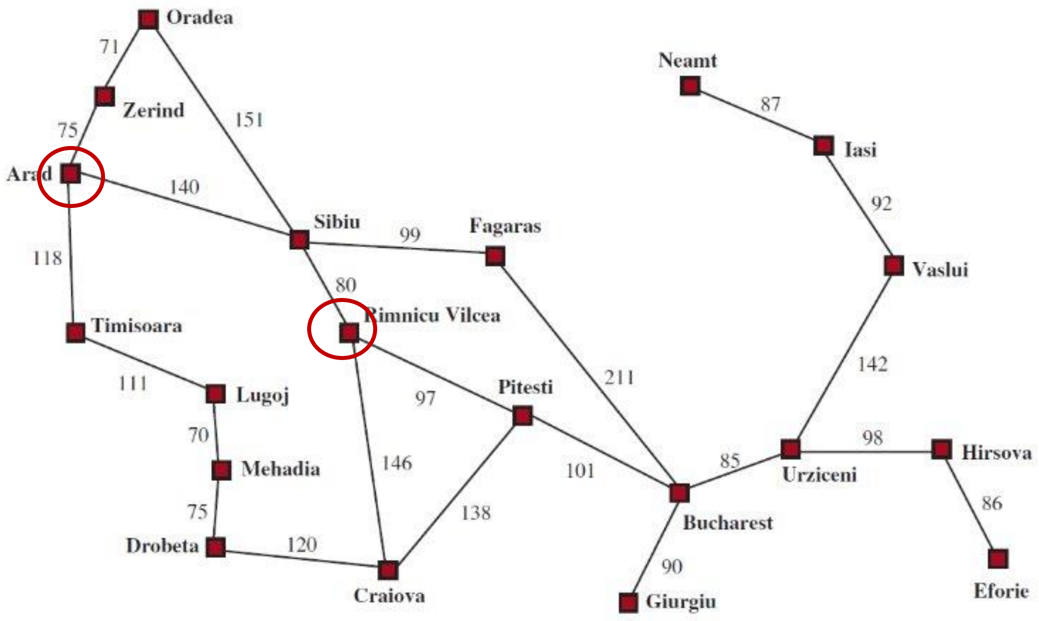
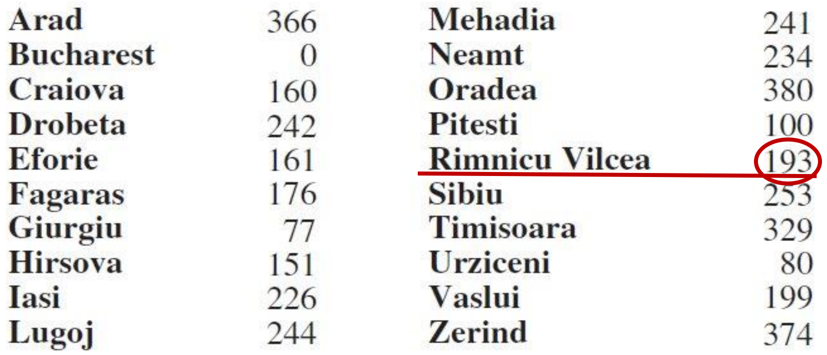


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.



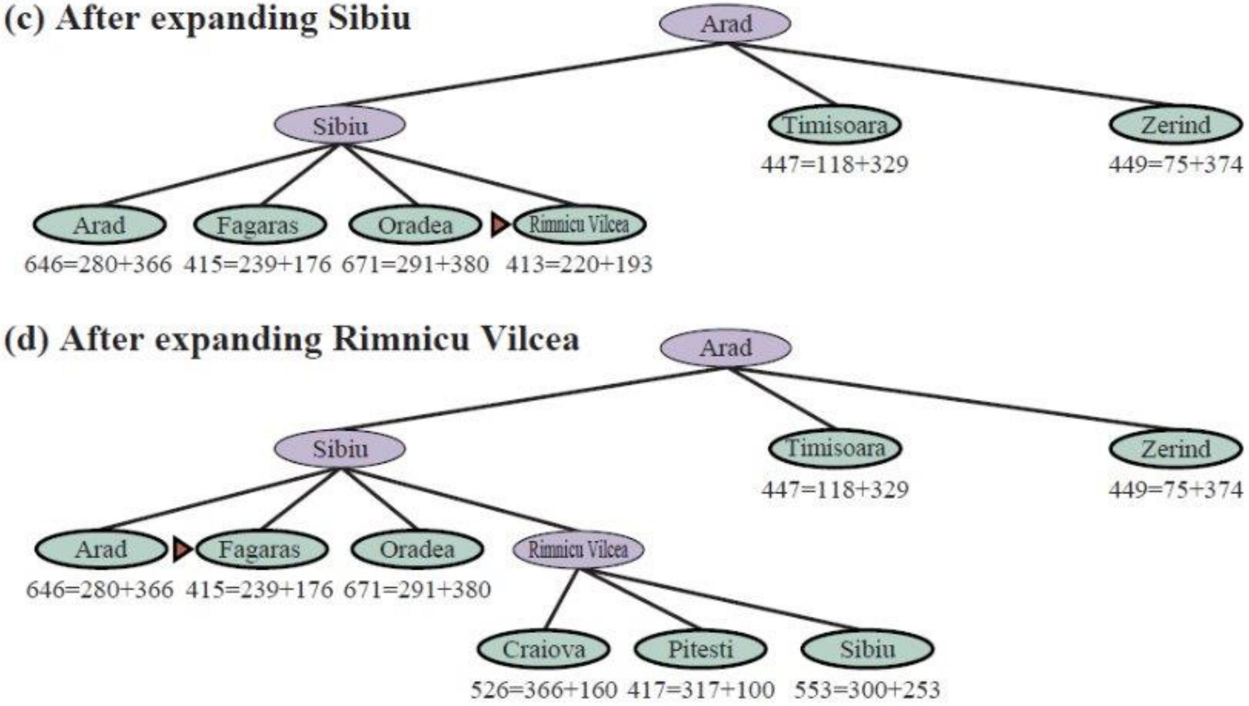
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.

Optimal Search

A* search

- Dis. from Arad to Rimicu Vilcea: g(Rimicu) = 140 + 80 = 220
- h(Rimicu) = 193
- f(Rimicu) = 220+193 = 413


(c) After expanding Sibiu

(d) After expanding Rimnicu Vilcea

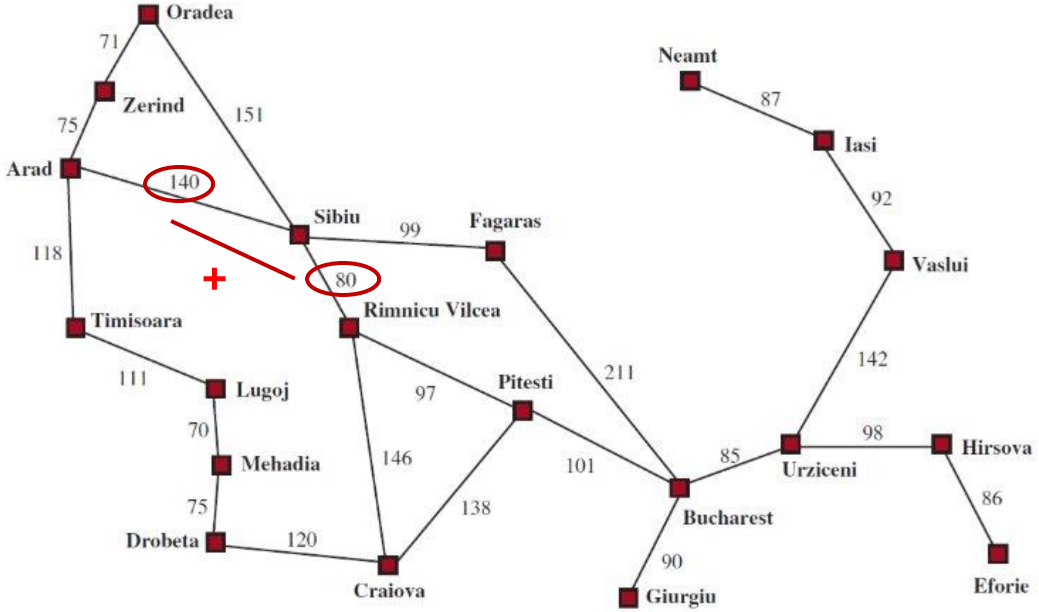
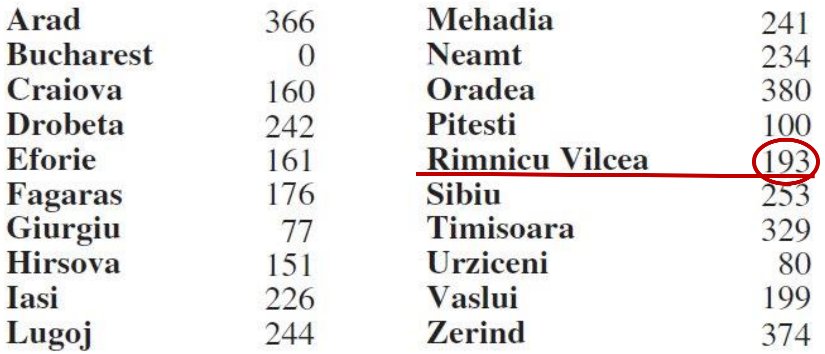


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.



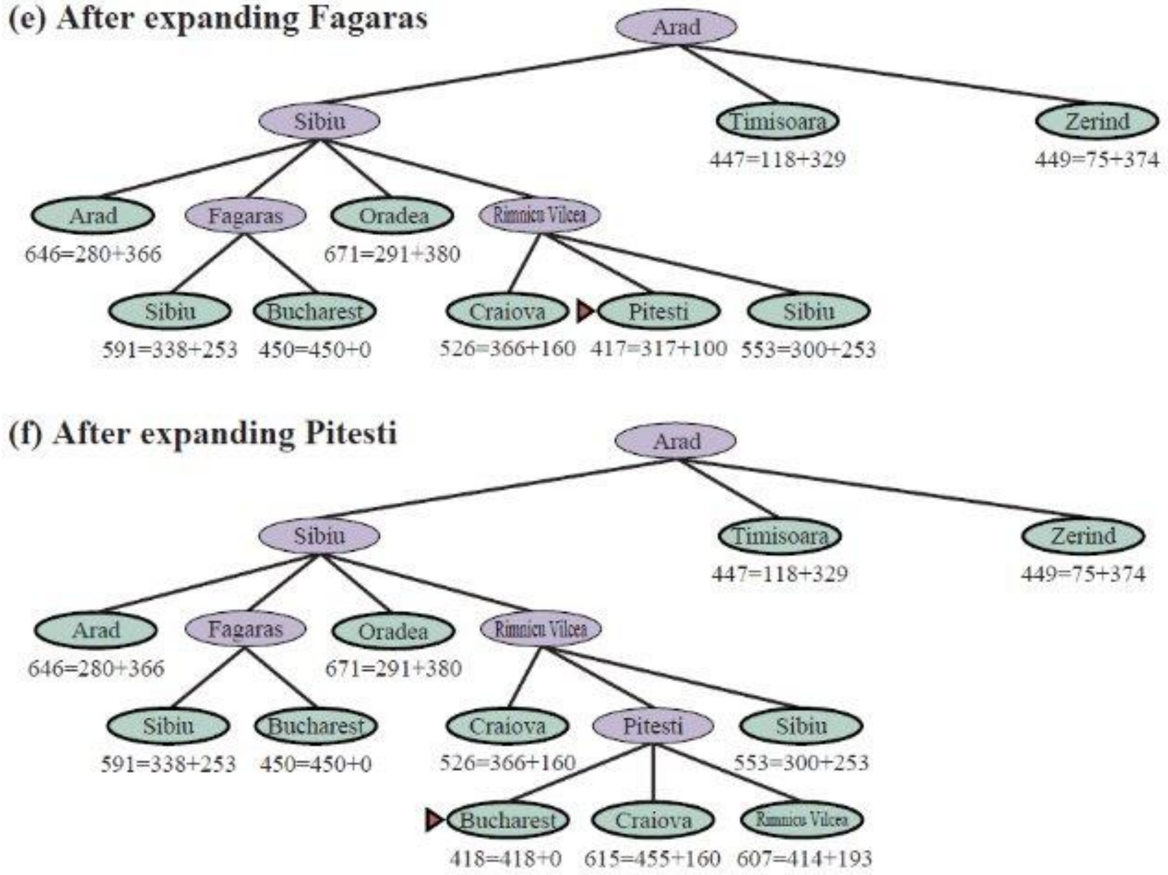
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.

- **Alway cal. cost from initial state**

Optimal Search

A* search



Path cost for the **optimal** solution =
140+80+97+101 = 418 miles

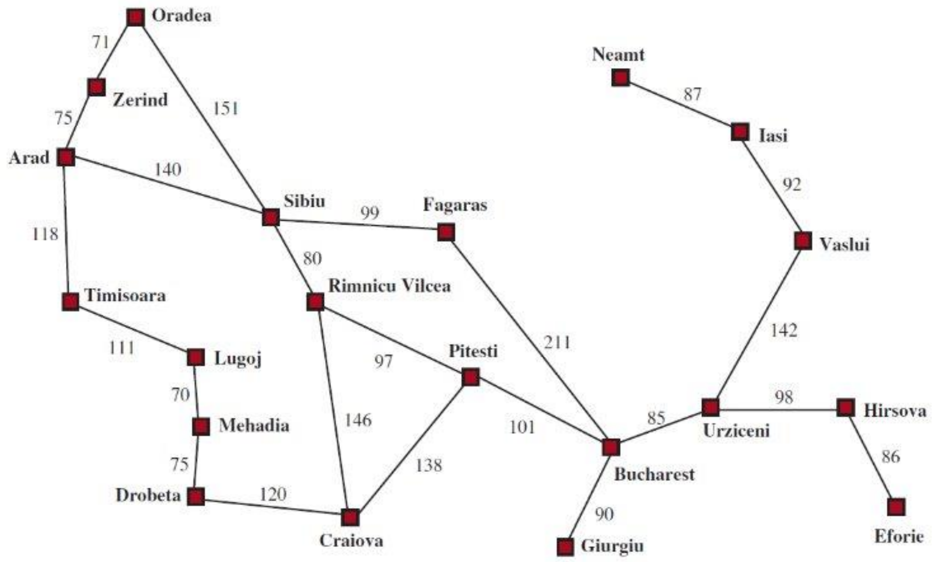
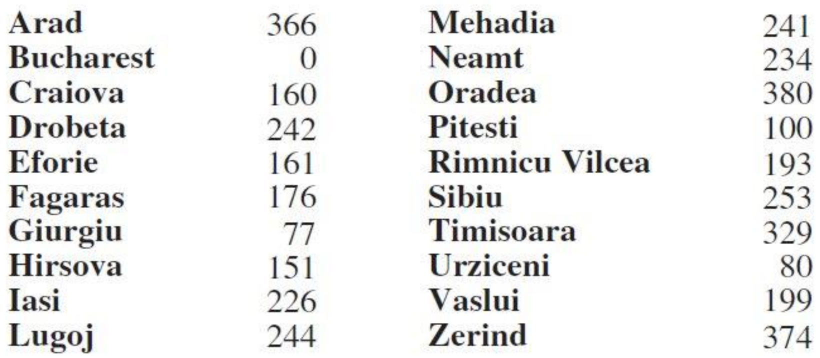


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.



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

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.

Optimal Search

A* search for 8-puzzle

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

0+4

2	8	3
1	6	4
7		5

2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Goal State

Optimal Search

A* search for 8-puzzle

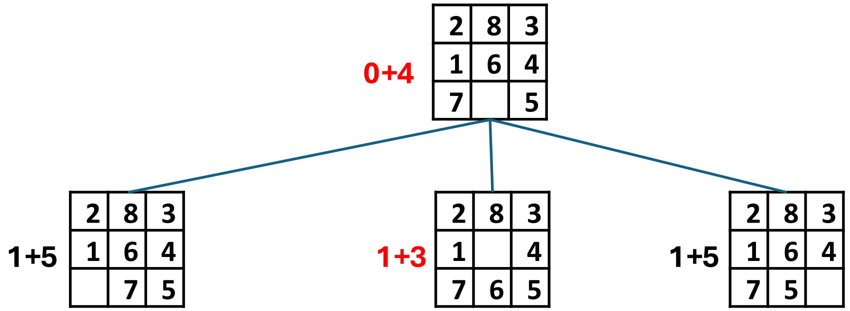
2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Goal State

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



0+4

2	8	3
1	6	4
7		5

1+5

2	8	3
1	6	4
	7	5

1+3

2	8	3
1		4
7	6	5

1+5

2	8	3
1	6	4
7	5	

Optimal Search

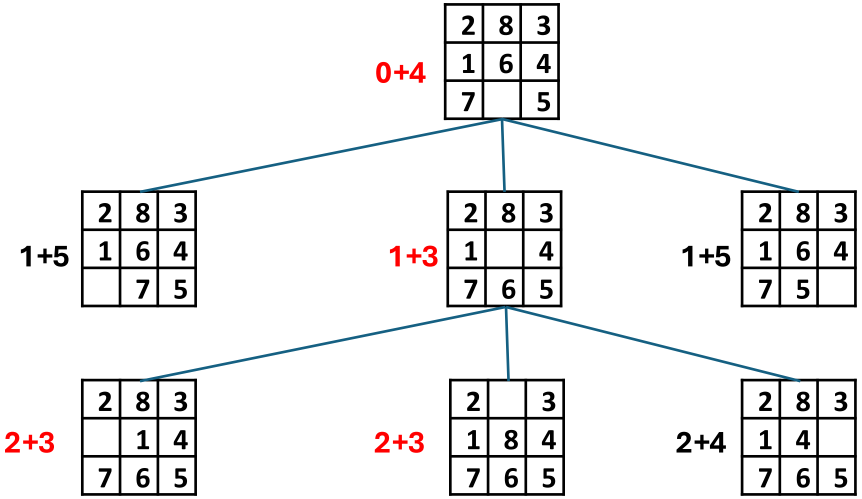
A* search for 8-puzzle

2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Goal State



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

Optimal Search

A* search for 8-puzzle

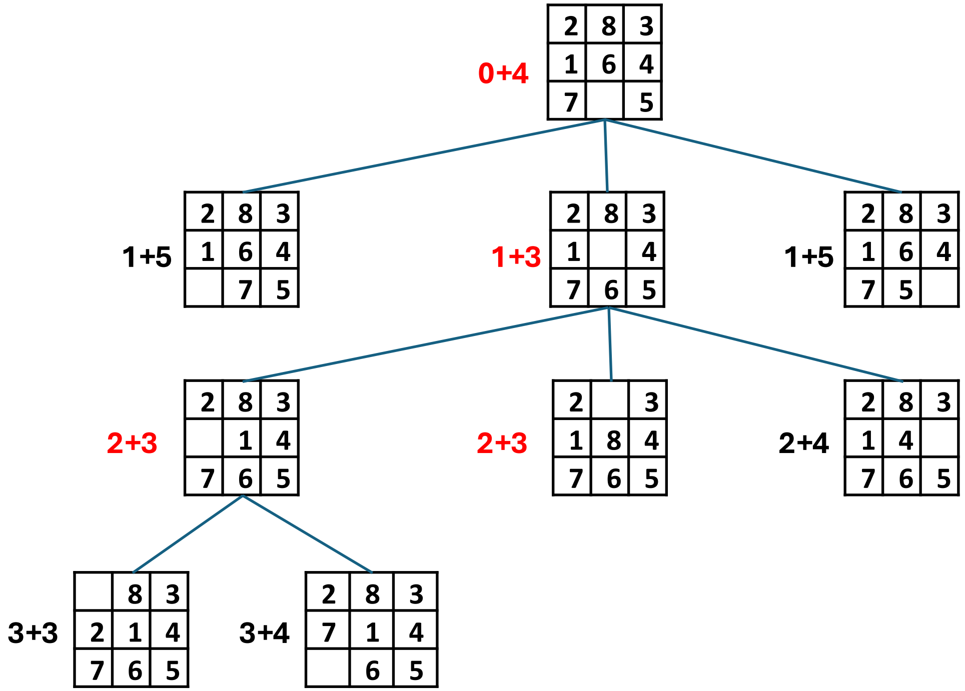
$$f(n) = g(n) + h(n)$$

2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Goal State



Optimal Search

A* search for 8-puzzle

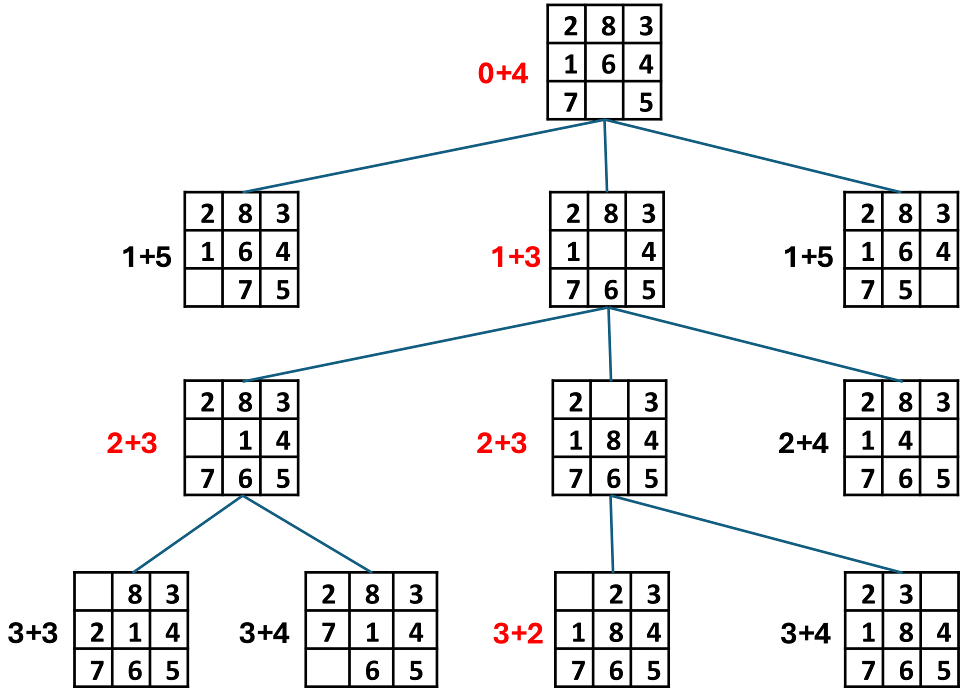
2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Goal State

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



Optimal Search

A* search for 8-puzzle

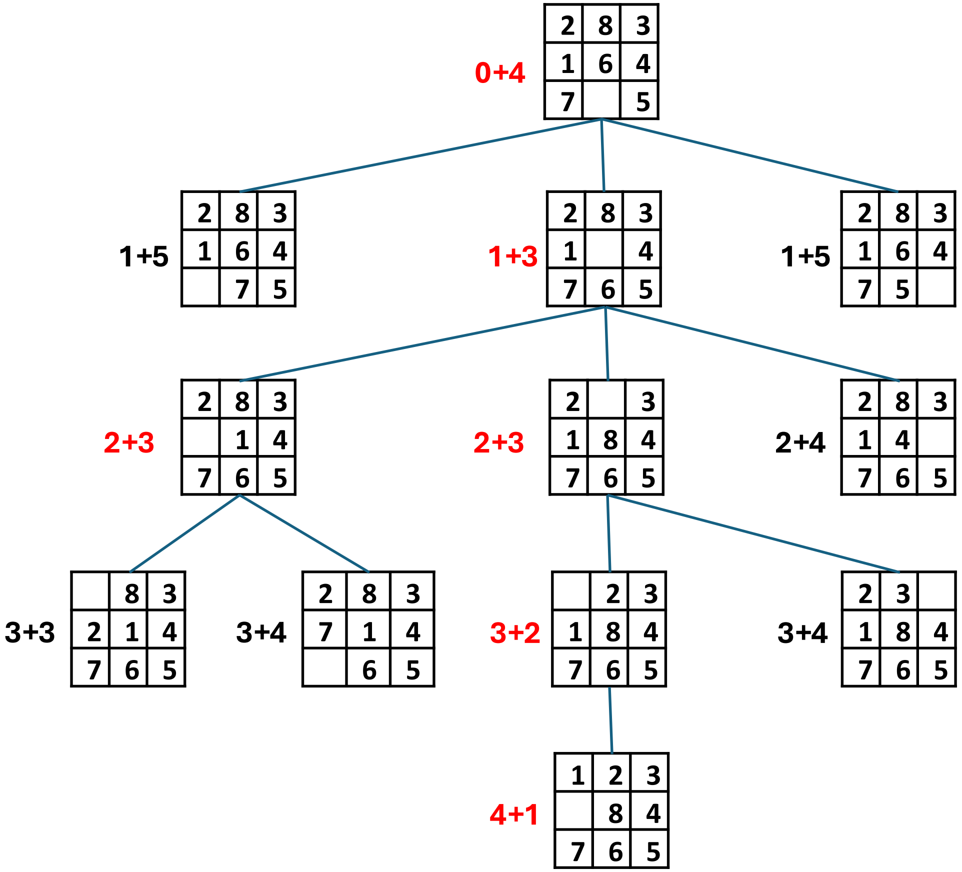
$$f(n) = g(n) + h(n)$$

2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Goal State



0+4

2	8	3
1	6	4
7		5

1+5

2	8	3
1	6	4
	7	5

1+3

2	8	3
1		4
7	6	5

1+5

2	8	3
1	6	4
7	5	

2+3

2	8	3
	1	4
7	6	5

2+3

2		3
1	8	4
7	6	5

2+4

2	8	3
1	4	
7	6	5

3+3

	8	3
2	1	4
7	6	5

3+4

2	8	3
7	1	4
	6	5

3+2

	2	3
1	8	4
7	6	5

3+4

2	3	
1	8	4
7	6	5

4+1

1	2	3
	8	4
7	6	5

Optimal Search

A* search for 8-puzzle

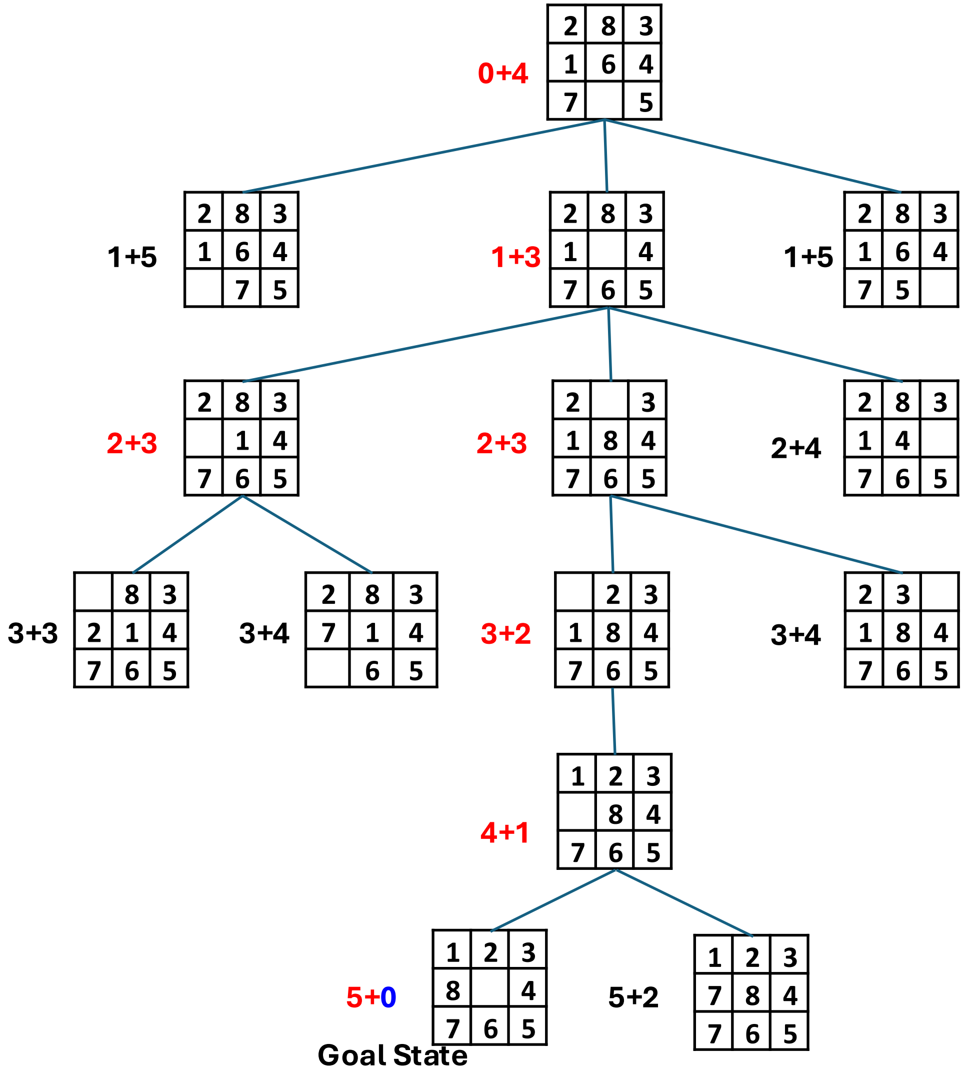
2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Goal State

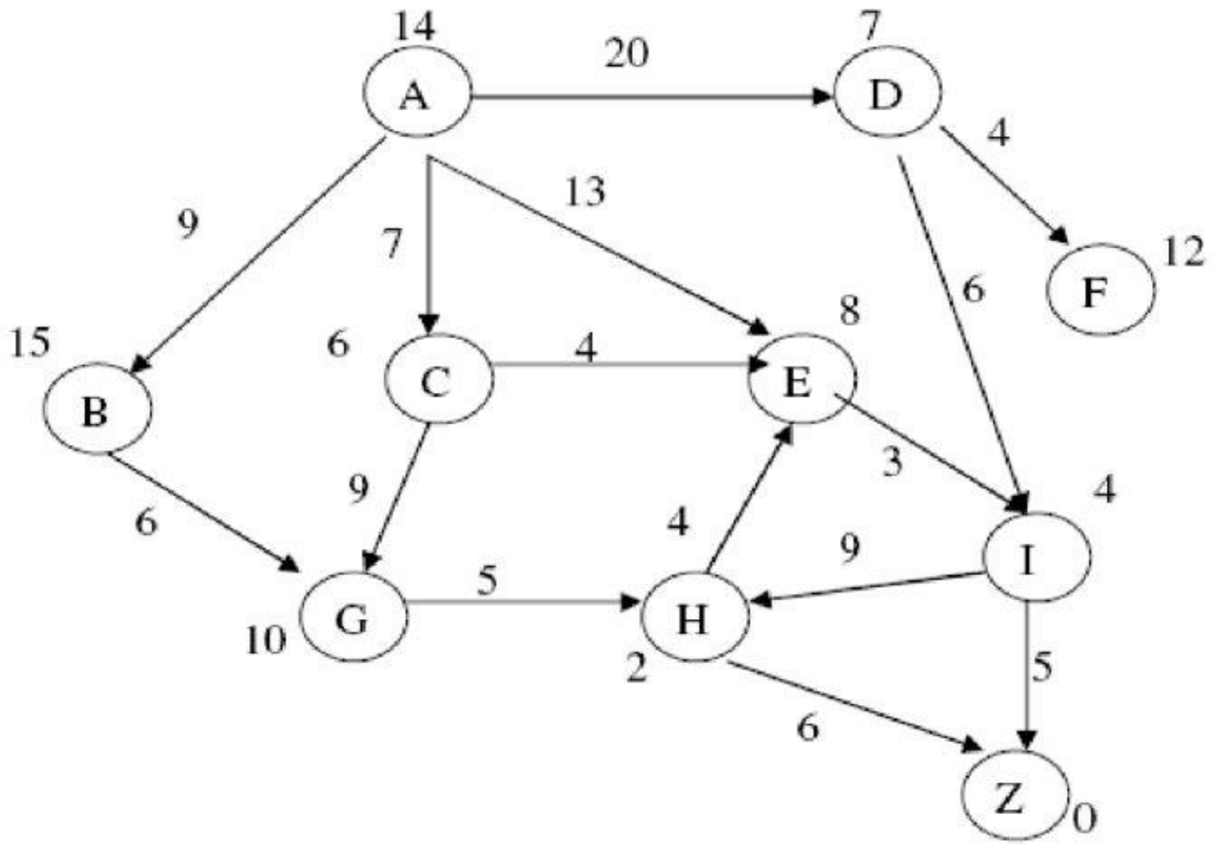
$$f(n) = g(n) + h(n)$$



Optimal Search

A* search: Find the shortest path from A to Z using A*

- The value attached to each vertex is h(u).
- The value attached to each edge is the cost to change state k(u,v).
- Note: when g+h are the same, the **smaller g** is preferred.



Content

- Optimal search
  - Definition
  - Greedy search
  - A* search
  - Properties of Heuristic Function

Optimal Search

Properties of Heuristic Function

- admissibility

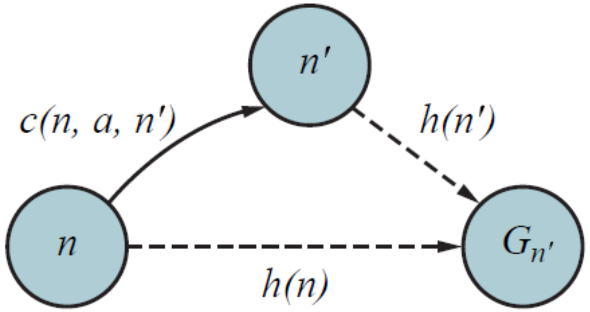
$$h(n) \leq h^{*}(n)$$

- consistency (monotonicity)

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

- dominancy: h_2 **dominates** h_1

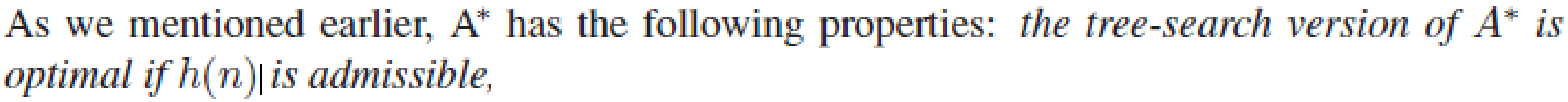
$$h_2(n) \geq h_1(n), \text{ for any node } n$$



Optimal Search

Optimality of A* algorithm

- A* is optimal if it uses an admissible(consistent) heuristic



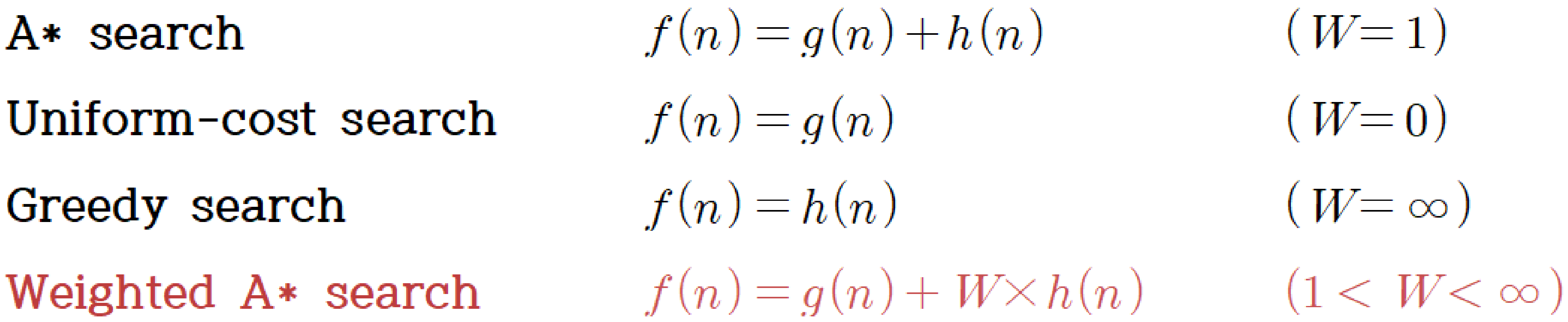
As we mentioned earlier, A* has the following properties: *the tree-search version of A* is optimal if $h(n)$| is admissible,*

Efficiency of A* algorithm

- A* with $h_2(n)$ is more efficient than A* with $h_1(n)$, if h_2 dominates h_1

Optimal Search

Weighted A* search



A* search	$f(n) = g(n) + h(n)$	$(W = 1)$
Uniform-cost search	$f(n) = g(n)$	$(W = 0)$
Greedy search	$f(n) = h(n)$	$(W = \infty)$
Weighted A* search	$f(n) = g(n) + W \times h(n)$	$(1 < W < \infty)$

- inadmissible heuristic → risk of missing optimal solution

Optimal Search

Weighted A* search

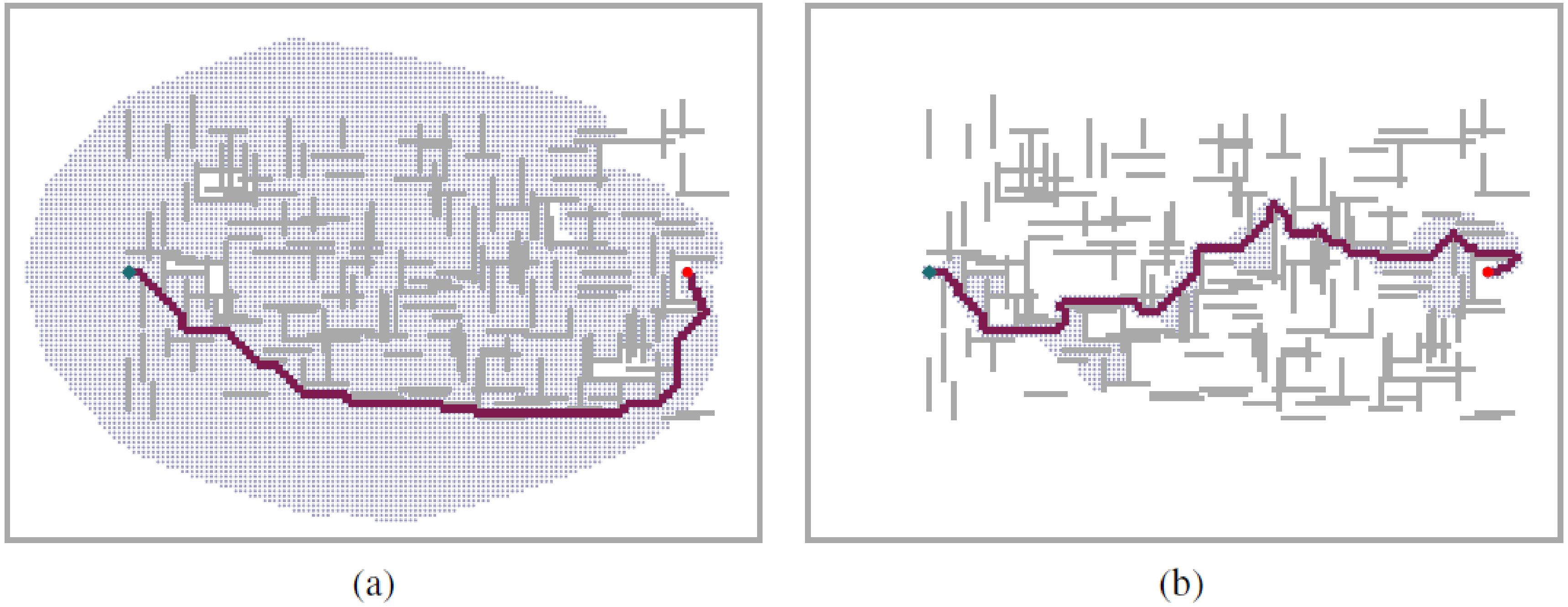


Figure 3.21 Two searches on the same grid: (a) an A* search and (b) a weighted A* search with weight $W = 2$. The gray bars are obstacles, the purple line is the path from the green start to red goal, and the small dots are states that were reached by each search. On this particular problem, weighted A* explores 7 times fewer states and finds a path that is 5% more costly.

Thank you!

You're now ready to explore the exciting world of AI!