Artificial Intelligence AI-2002 Lecture 5

Mahzaib Younas

Lecture Department of Computer Science

FAST NUCES CFD

Informed Search

- One that uses **problem-specific knowledge** beyond the definition of the problem itself.
- For Example
    - Best First Search
    - A* Search

Best-First Search

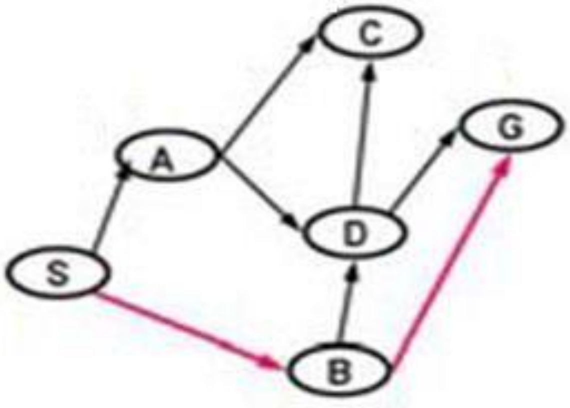
- A node is selected for expansion based on an evaluation function, $\boldsymbol{f(n)}$.
  - the *lowest* evaluation is expanded first
- Best-first search algorithms include as a component of f a **heuristic function**, denoted $\boldsymbol{h(n)}$.
- $\boldsymbol{h(n)}$ = ***estimated*** cost of the cheapest path
  - $\boldsymbol{h(n)}$ takes a *node* as **input**, but, unlike $\boldsymbol{g(n)}$, it depends only on the *state* at that node.

$$\boldsymbol{f(n) = h(n)}$$

Best-First Search

- Pick best (by **heuristic value**) element of Q
- Add path extensions to Q

	Q	Visited
1	(10 S)	S
2	**(2 A S) (3 B S)**	A, B, S
3	**(1 C A S)** (3 B S) **(4 D A S)**	C, D, B, A, S
4	(3 B S) (4 D A S)	C, D, B, A, S
5	**(0 G B S)** (4 D A S)	G, C, D, B, A, S



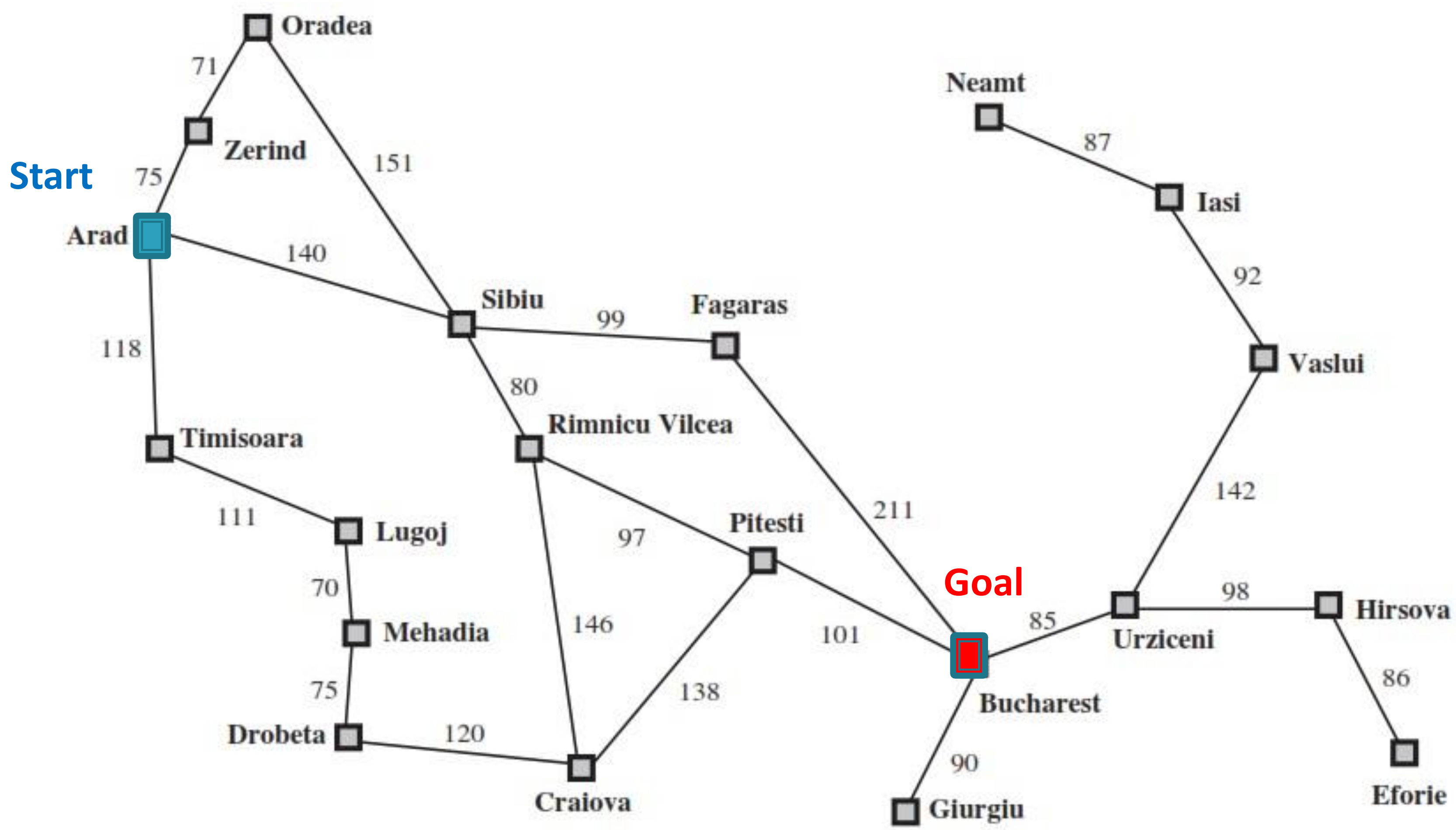
Heuristic Values

A=2	C=1	S=10
B=3	D=4	G=0

- **Blue Color represents added paths**
- Heuristic value in node state is in front.

(4DBS) is not included in the list because 'D' is already available in "**Visited List**"

Example



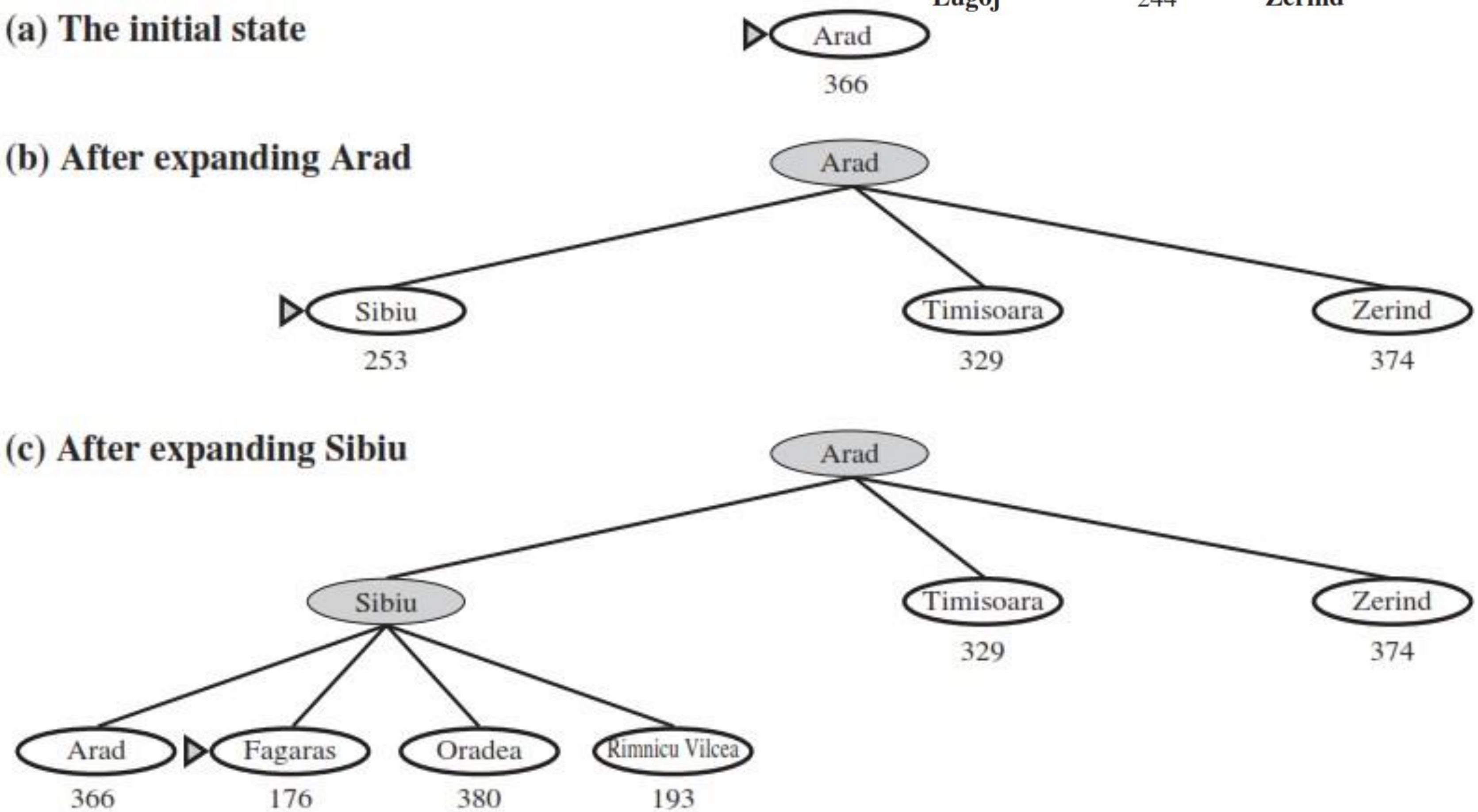
A simplified road map of part of Romania.

Values of h_{SLD}—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

Best-First Search

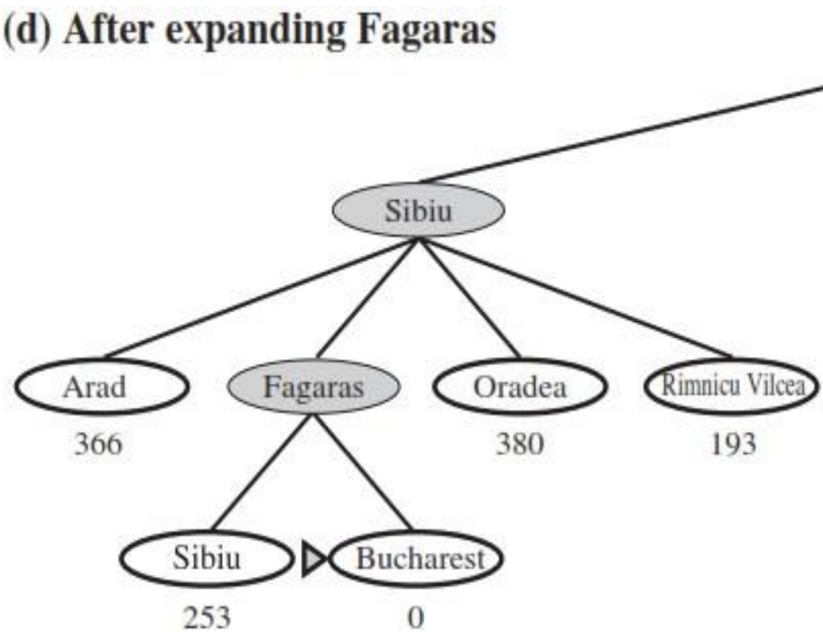
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



Best-First Search

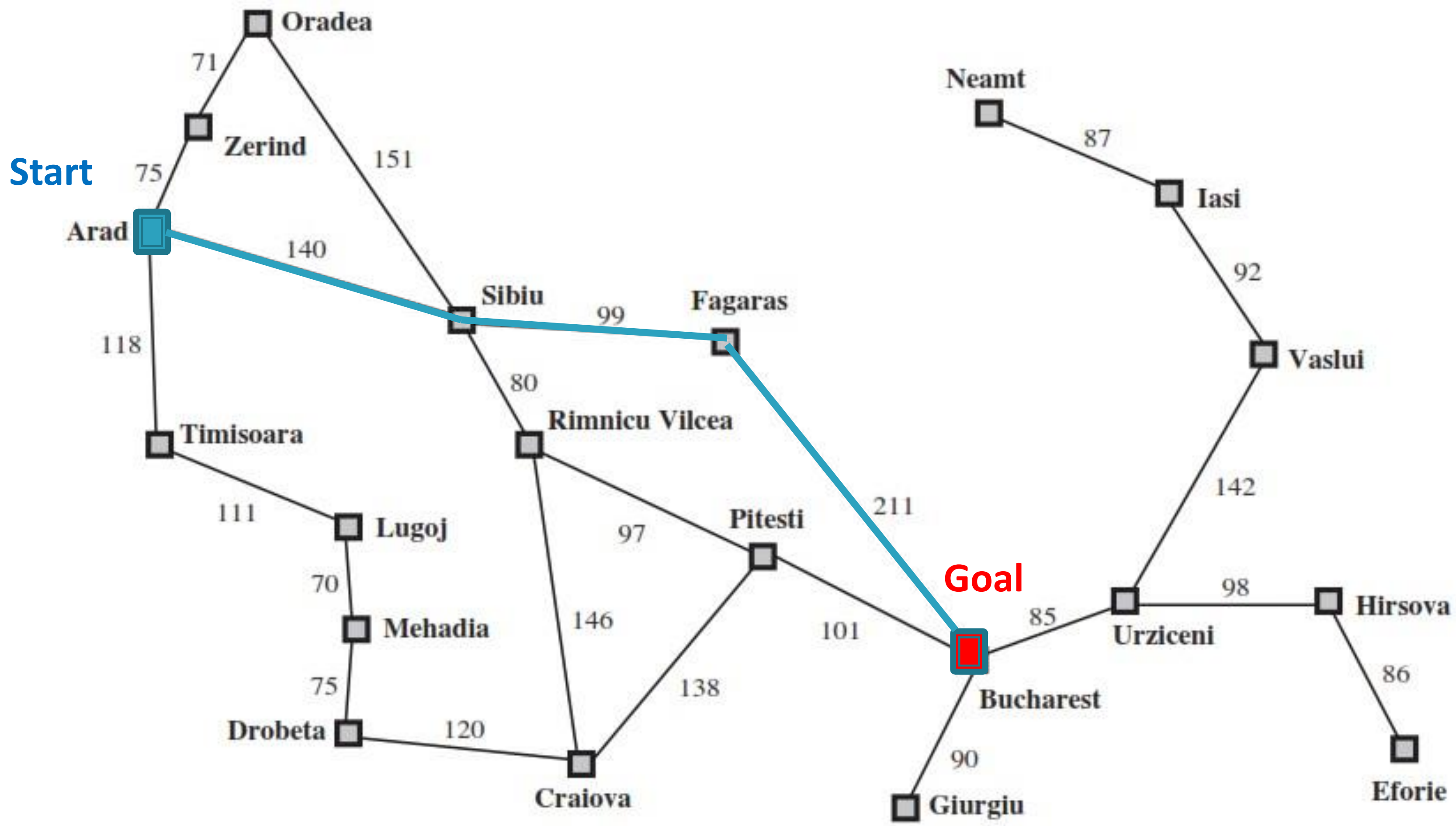
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

(d) After expanding Fagaras



Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic. Nodes are labeled with their h-values.

Example



A simplified road map of part of Romania.

Best-First Search

Completeness

- Best-first **tree search** is **incomplete** even in a finite state space, much like depth first search.
- The **graph search** version is **complete** in finite spaces, but not in infinite ones.

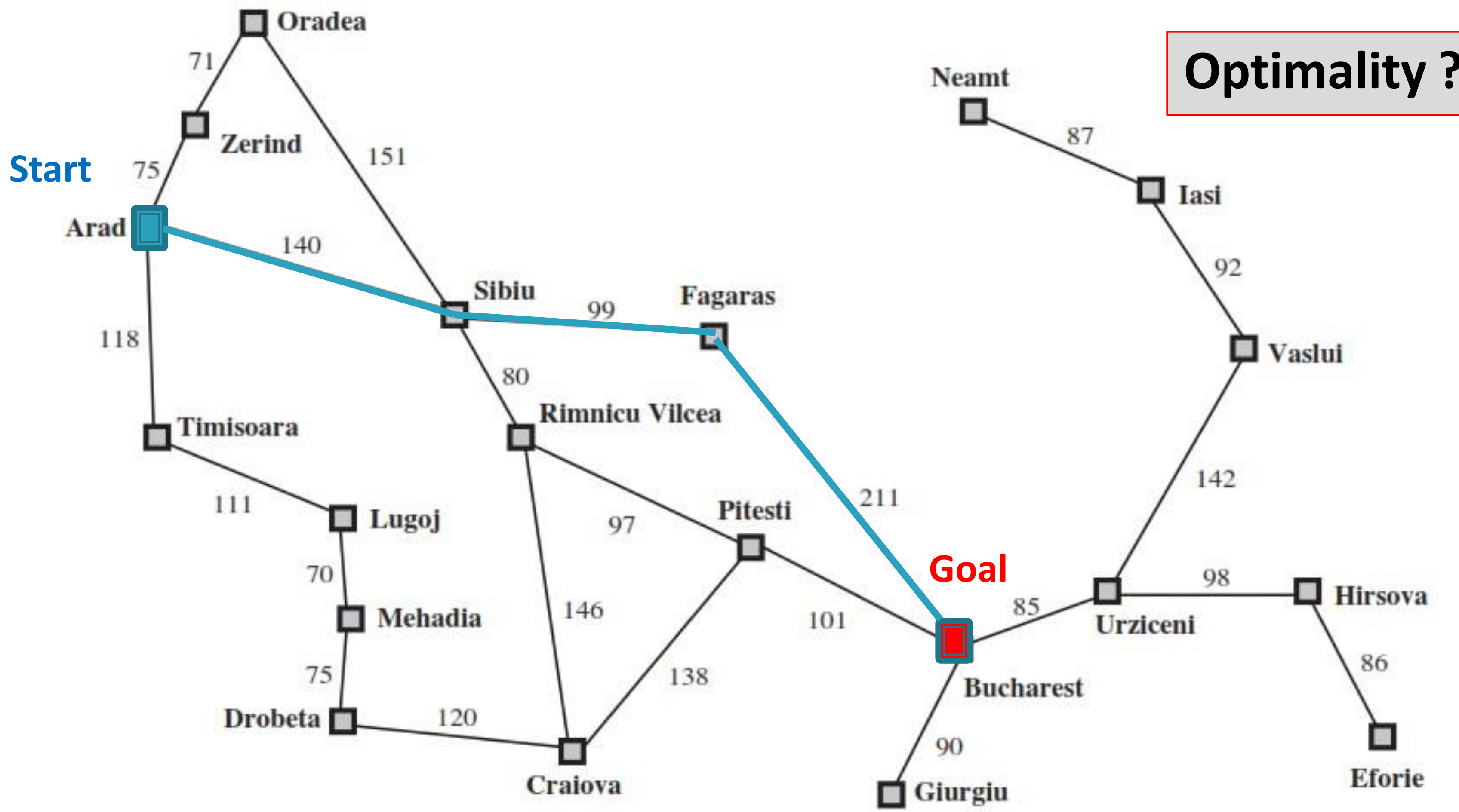
- **Time and Space Complexity**

- The *worst-case time and space* complexity for the tree version is $\boldsymbol{O(b^m)}$, where $\boldsymbol{m}$ is the maximum depth of the search space.
  - With a good heuristic function, however, the complexity can
  - be reduced substantially.

Example 1

Arad-Sibiu-Fagaras-Buchrest: 140+99+211=450
Arad-Sibiu-Vilcea-Pitesti-Buchrest: 140+80+97+101=418

Optimality ?

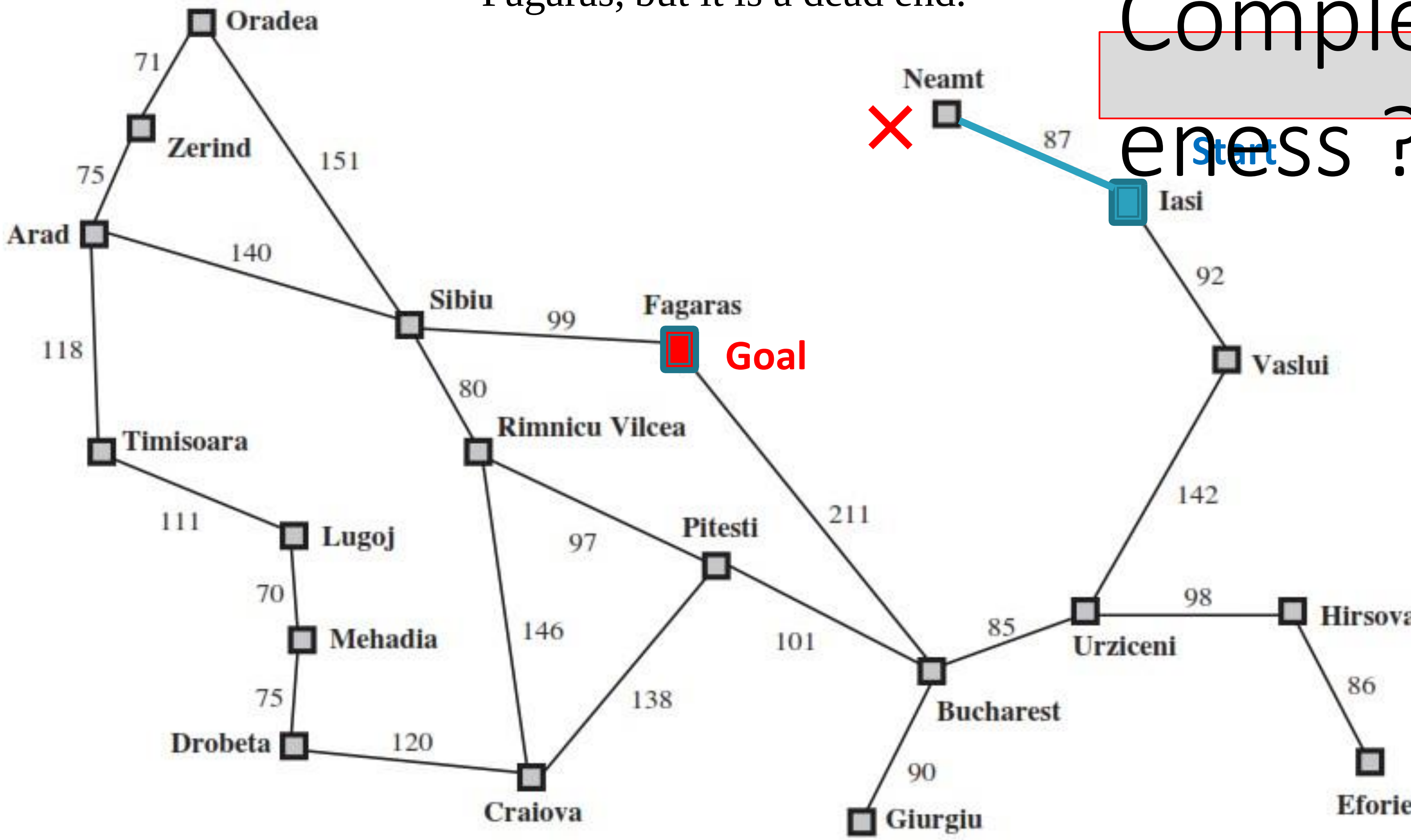


A simplified road map of part of Romania.

Example 2

The heuristic--- **straight-line distances ---**suggests that Neamt be expanded first because it is closest to Fagaras, but it is a dead end.

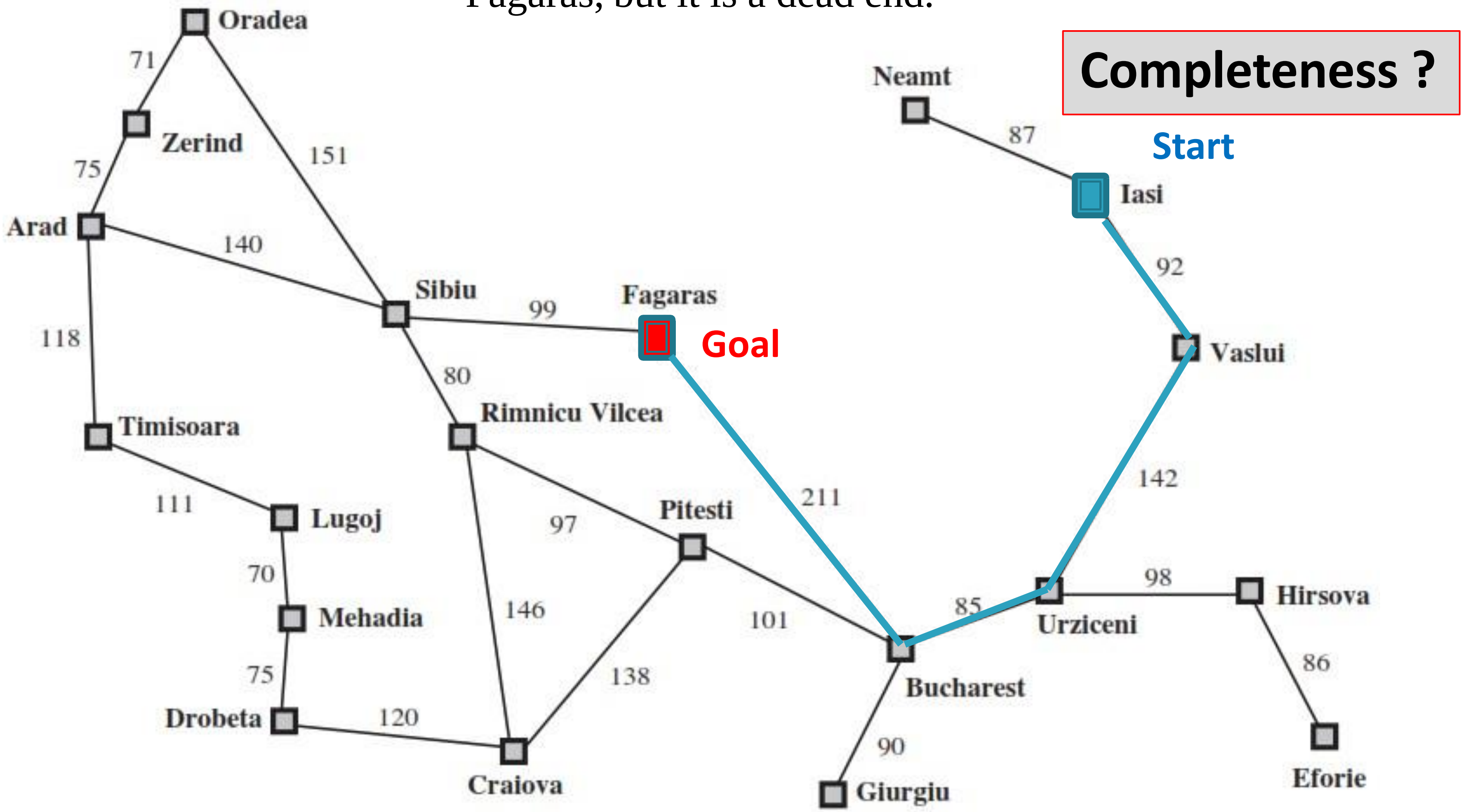
Complet eness ?



A simplified road map of part of Romania.

Example 2

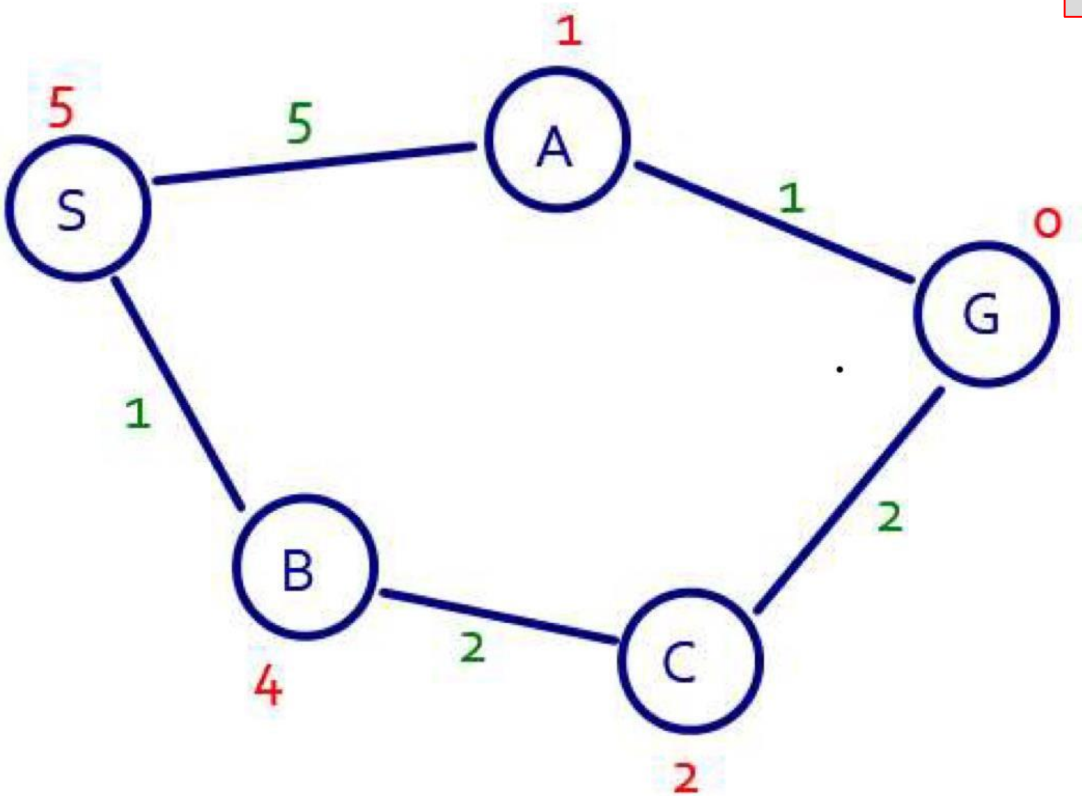
The heuristic--- straight-line distances ---suggests that Neamt be expanded first because it is closest to Fagaras, but it is a dead end.



A simplified road map of part of Romania.

Example 3

Optima
lity ?



A* Search

A* Search

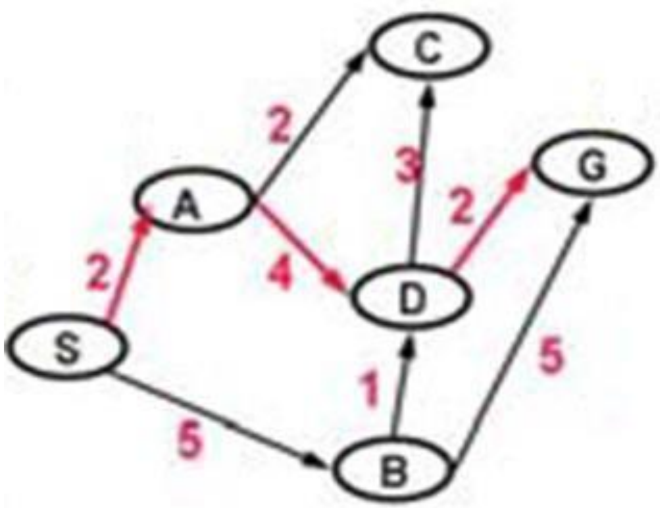
- We **can bias Uniform-cost search** to find the shortest path to the goal.
- In fact, we are interested in by using a **heuristic function $h(n)$** which is an estimate of the distance from a state to the goal.
- It evaluates nodes by combining $g(n)$, the cost to reach the node, and $h(n)$, the cost to get from the node to the goal:

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

A* Search

- Pick best (by **path length + heuristic value**) element of Q
- Add path extensions to Q

	Q
1	<u>(0 S)</u>
2	<u>**(4 A S)**</u> **(8 B S)**
3	<u>**(5 C A S)**</u> **(7 D A S)** (8 B S)
4	<u>(7 D A S)</u> (8 B S)
5	**(8 G D A S)** **(10 C D A S)** (8 B S)



Heuristic Values

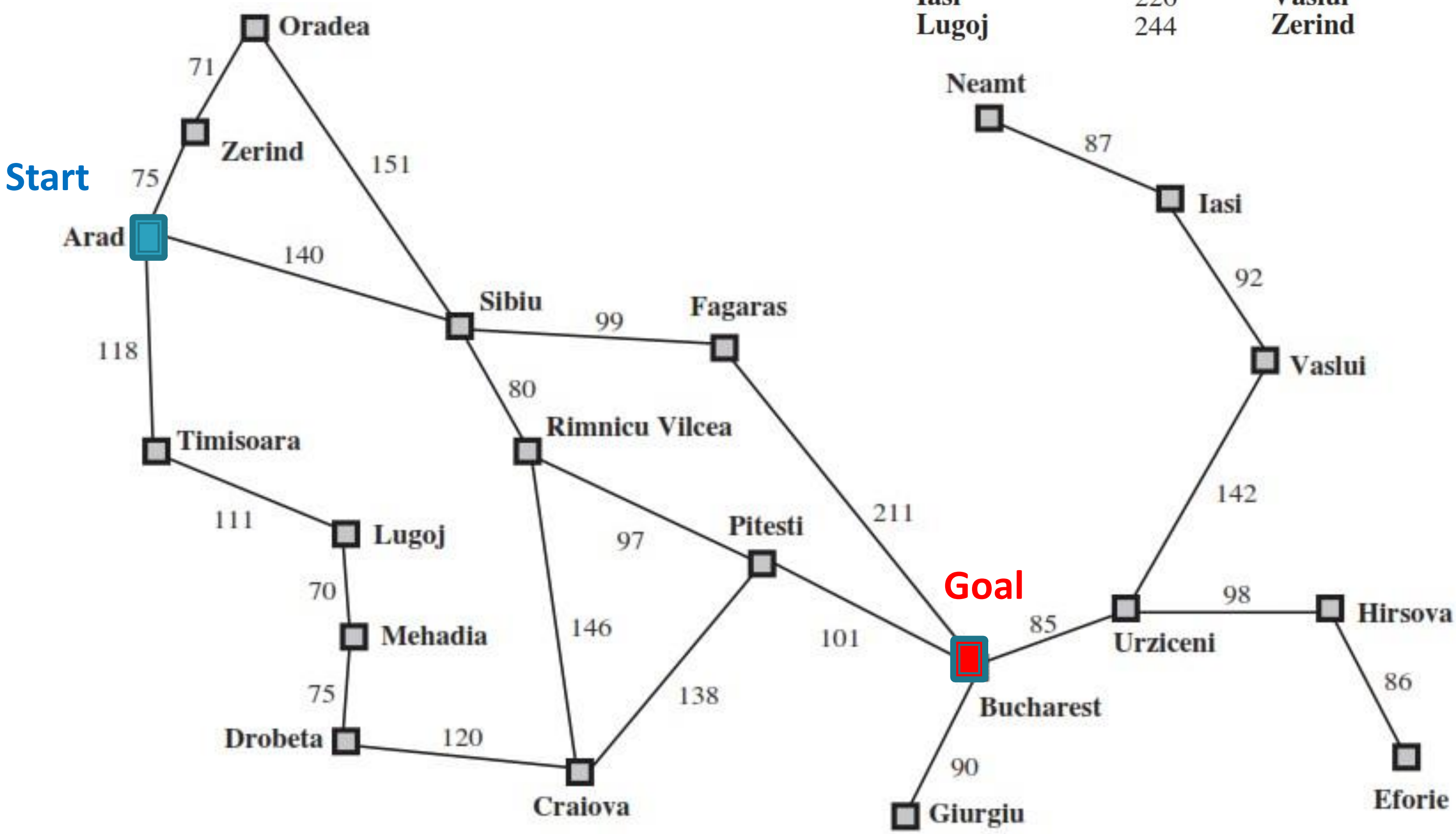
A=2 C=1 S=0

B=3 D=1 G=0

- **Blue Color represents added paths**
- <u>Underline</u> paths are selected for extension.

Example

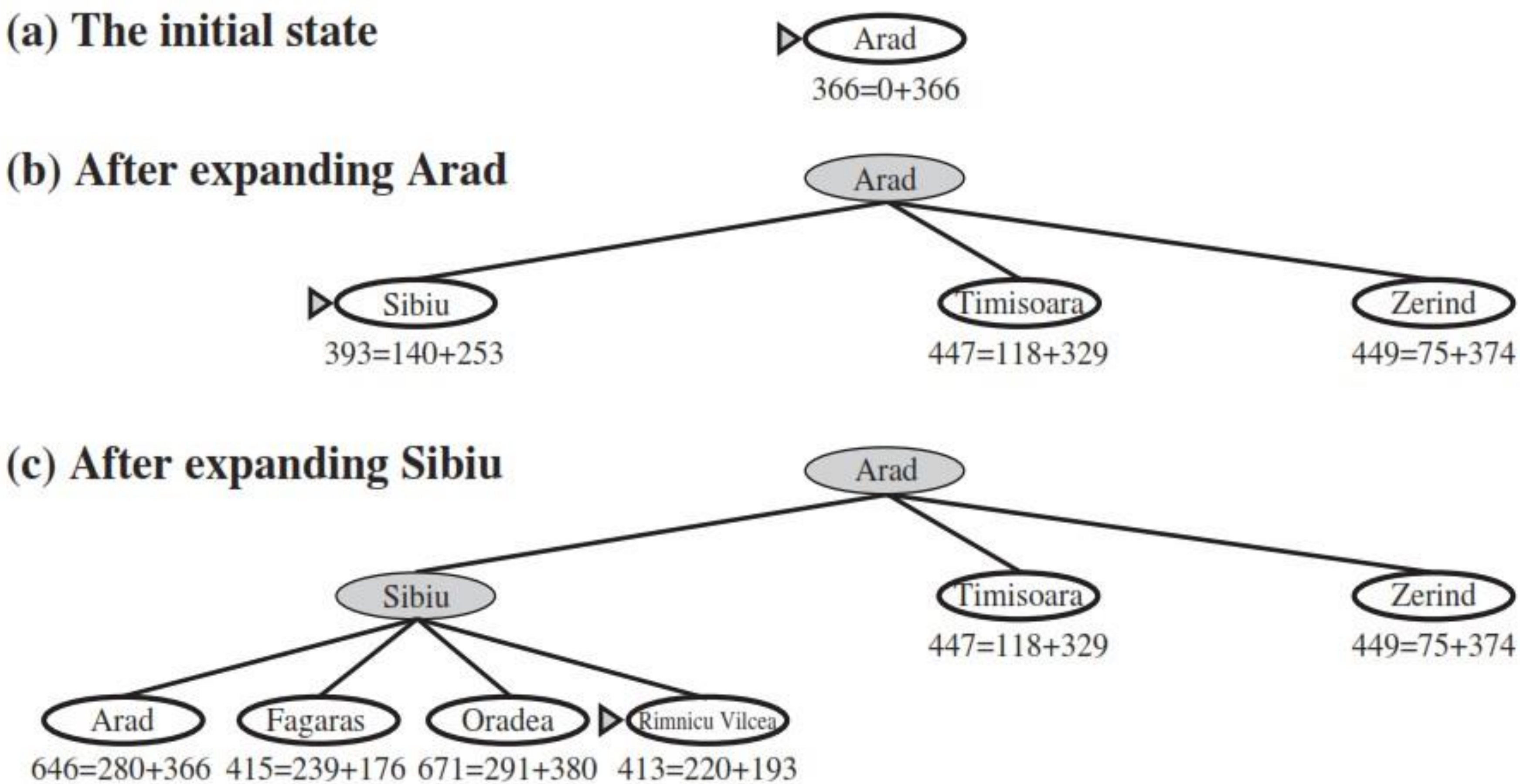
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



Values of $\boldsymbol{h_{SLD}}$—straight-line distances to Bucharest

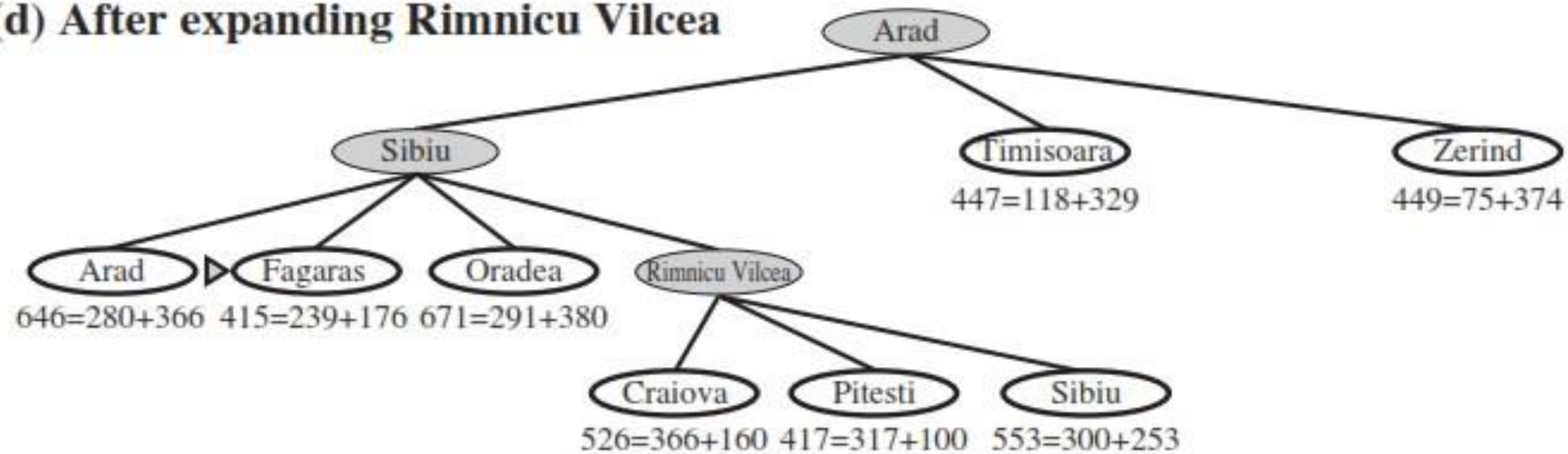
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Example

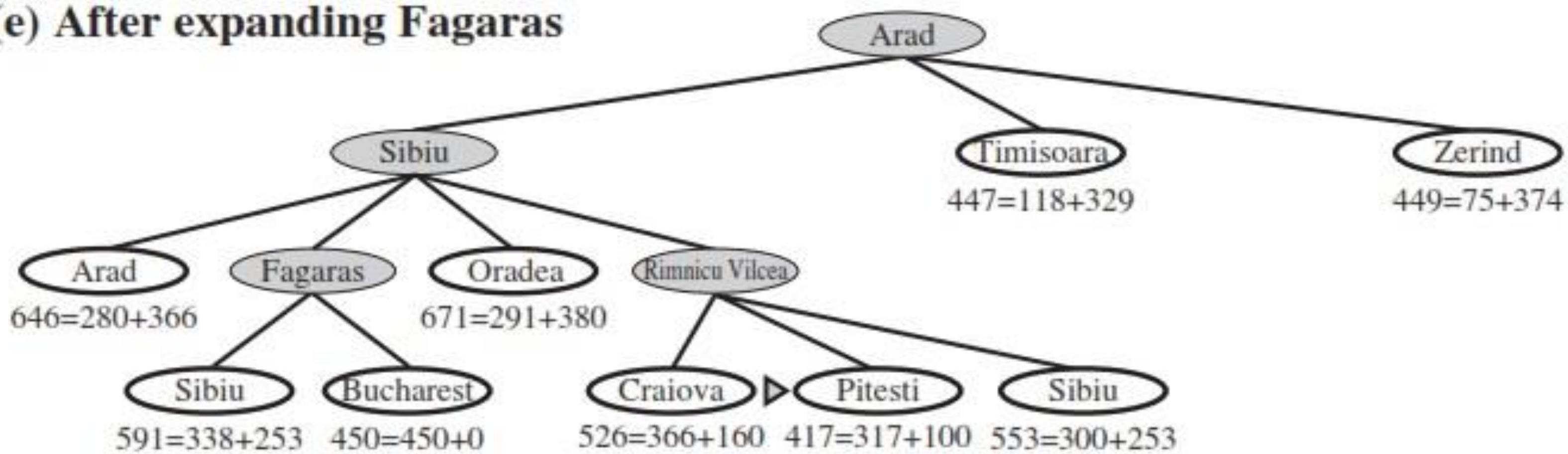

(a) The initial state
Arad
366=0+366
(b) After expanding Arad
Arad
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
413=220+193

Example

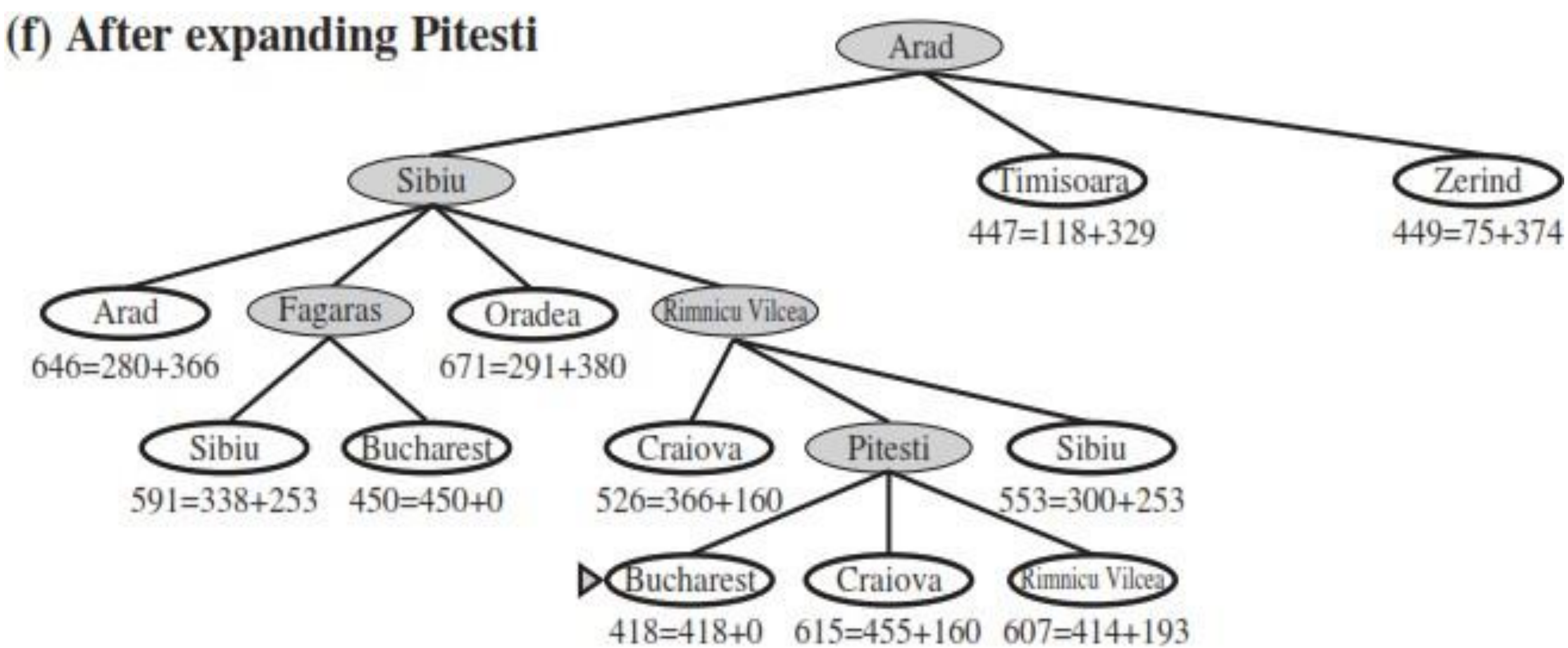
(d) After expanding Rimnicu Vilcea



(e) After expanding Fagaras

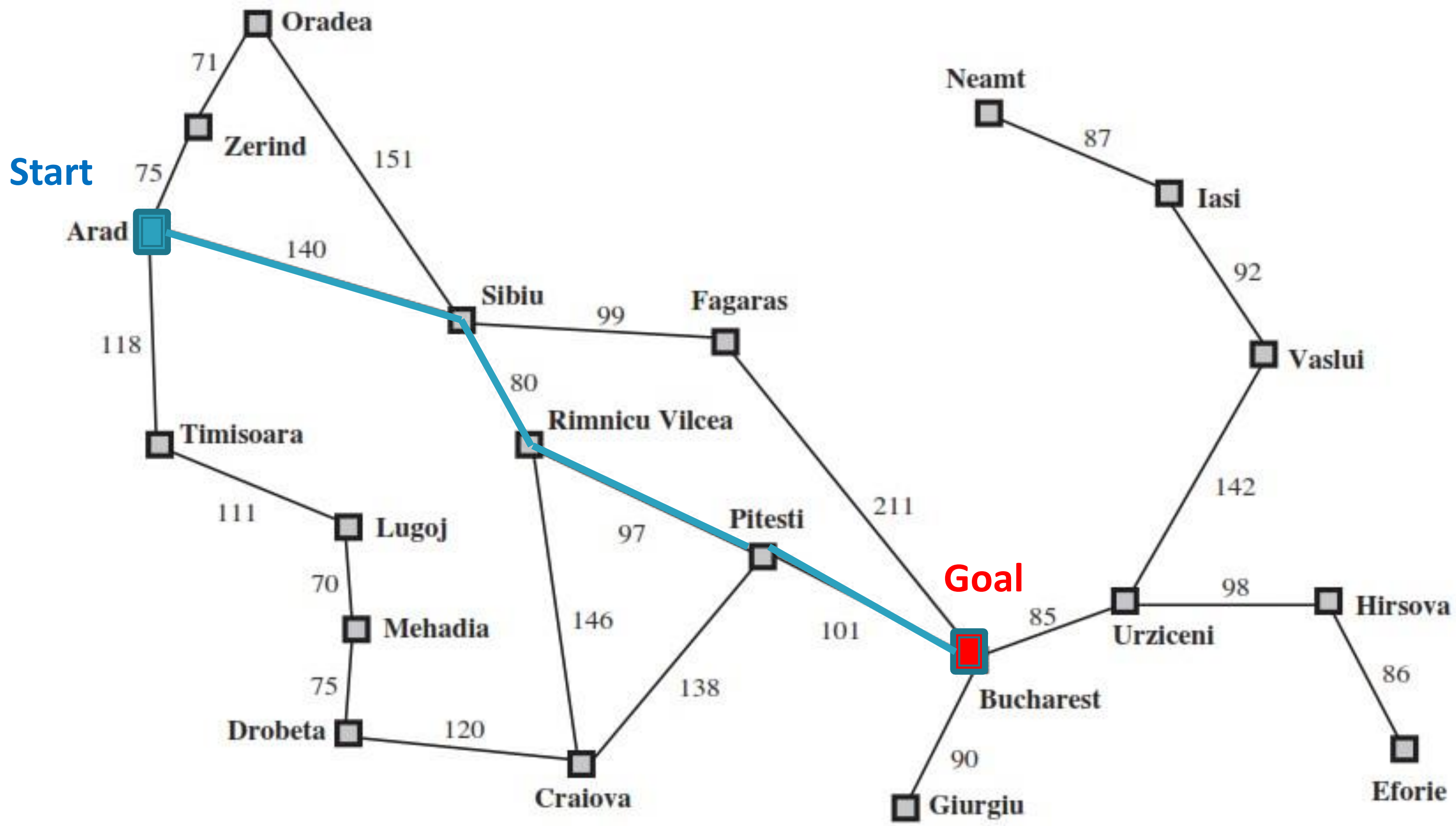


Example



A* search for Bucharest: Nodes are labeled with $f = g + h$. The h values are the straight-line distances to Bucharest.

Example



A simplified road map of part of Romania.

A* Search

- We **can bias Uniform-cost search** to find the shortest path to the goal.
- In fact, we are interested in by using a **heuristic function $h(n)$** which is an estimate of the distance from a state to the goal.
- It evaluates nodes by combining $g(n)$, the cost to reach the node, and $h(n)$, the cost to get from the node to the goal:

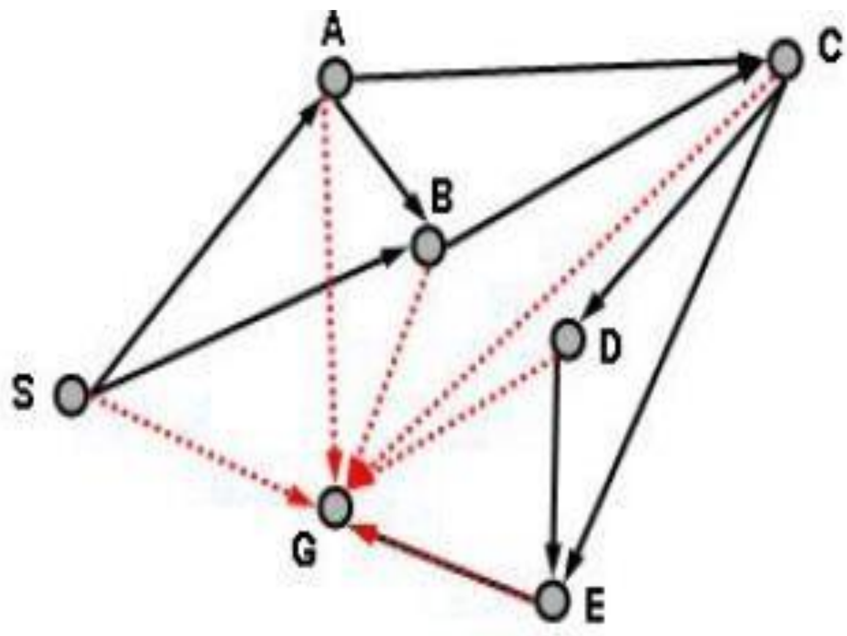
$$f(n) = g(n) + h(n)$$

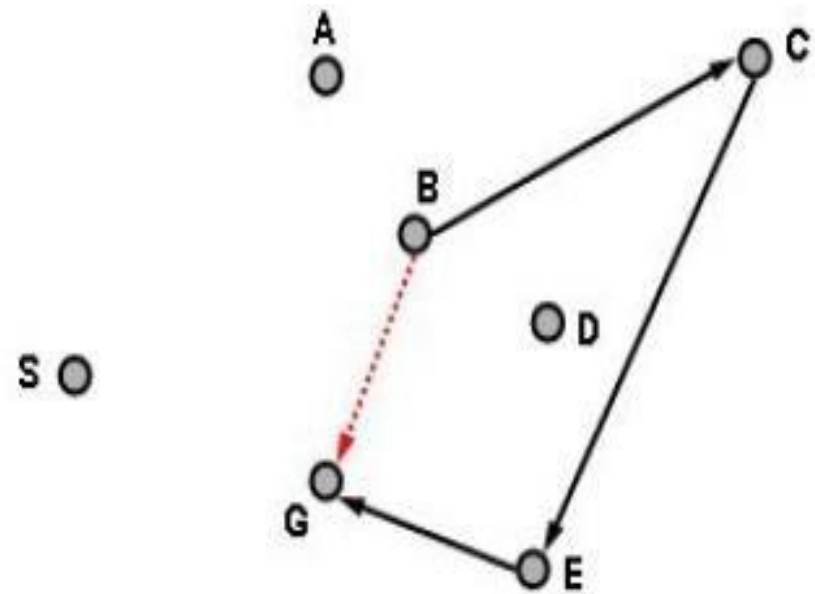
Admissible Heuristic

- An estimate that **always underestimates the real path length to *the goal*** is called **admissible estimate (heuristic).**
  - **Straight line distance** is an admissible estimate for path length in euclidian space.
- Uniform cost search is an instance of A*, If we set $h(n) = 0$.
  - Use of an admissible estimate gaurantee that Uniform-cost search will find the shortest path.

Uniform-cost search with admissible estimate (heuristic) is known as A* search.

Straight line Distance





Are all heuristics admissible?

- Given the **heuristic values** and **lenghts** in the Figure, are the heuristic values in the table addmissible?

  A = OK

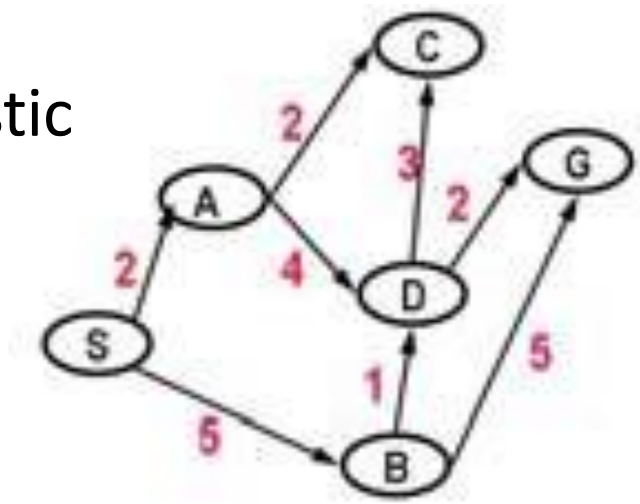
  B = OK

  C = OK

  D = not OK, needs to be <= 2

  S = not OK, should be 0.

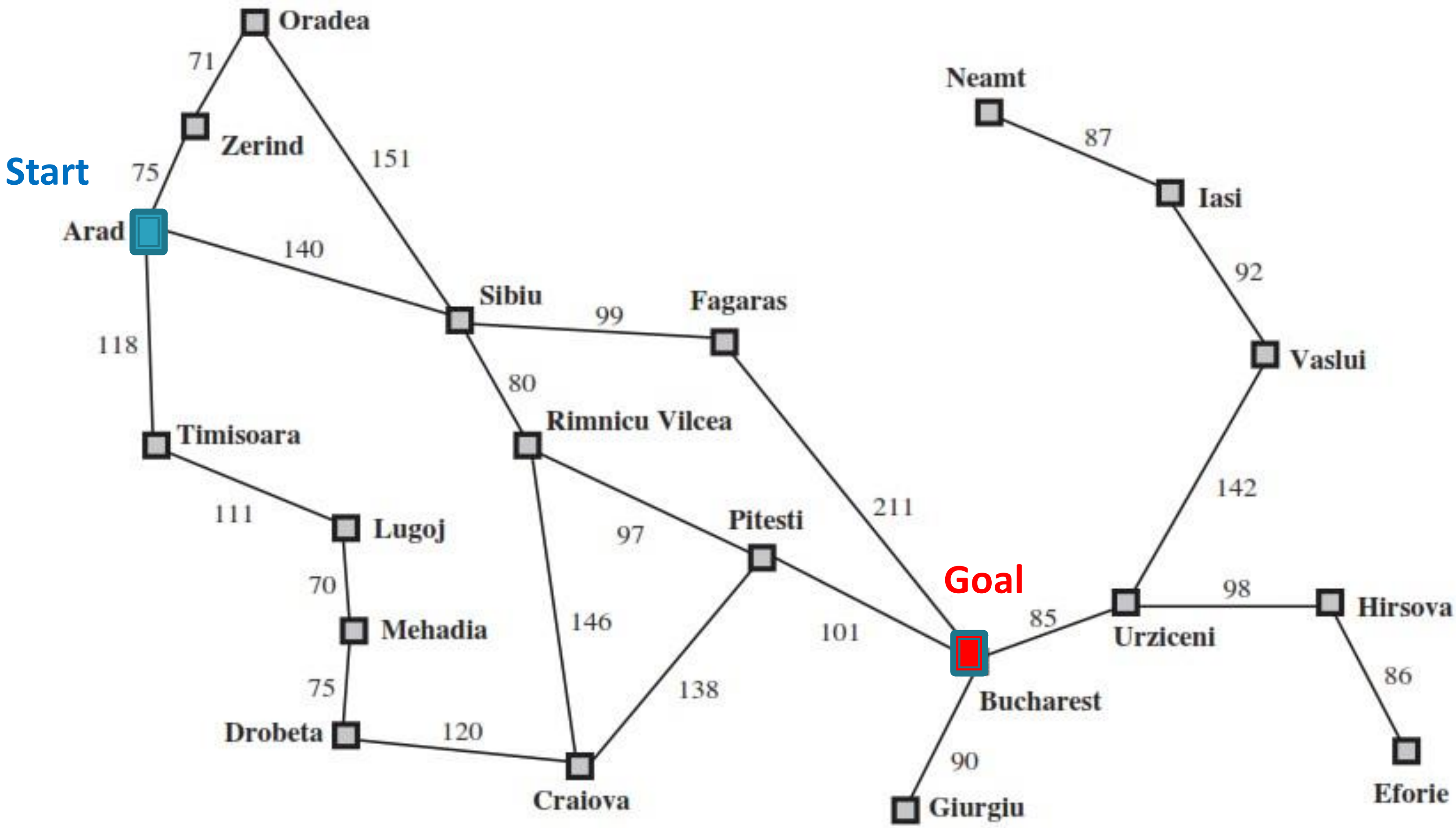
  but the value of S would have no ill effect



Heuristic Values

A=2	C=1	S=10
B=3	D=4	G=0

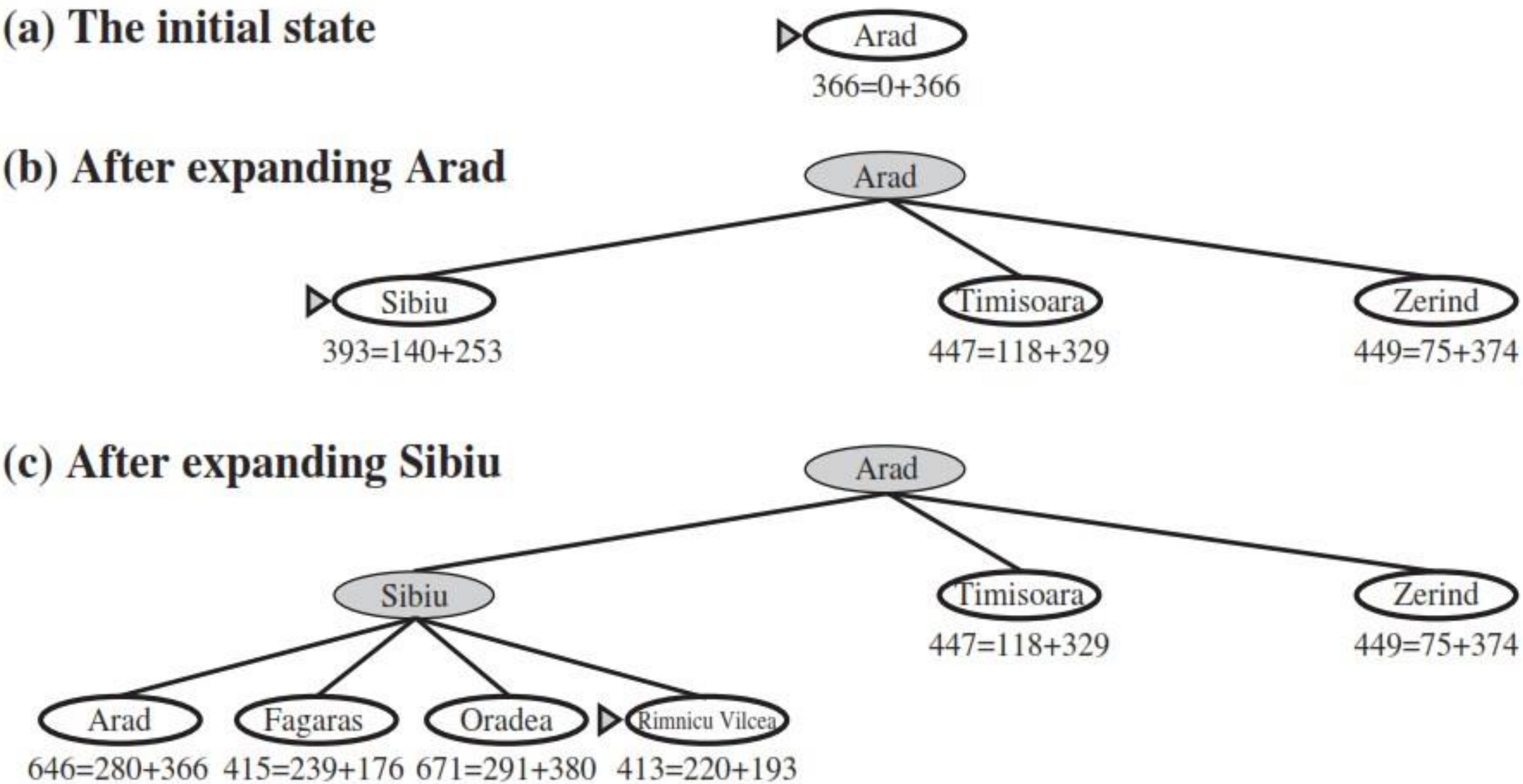
Example


Oradea
71
Zerind
151
Start
75
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
Goal
85
Urziceni
98
Hirsova
86
Bucharest
90
Giurgiu
Eforie

Values of $\boldsymbol{h_{SLD}}$—straight-line distances to Bucharest

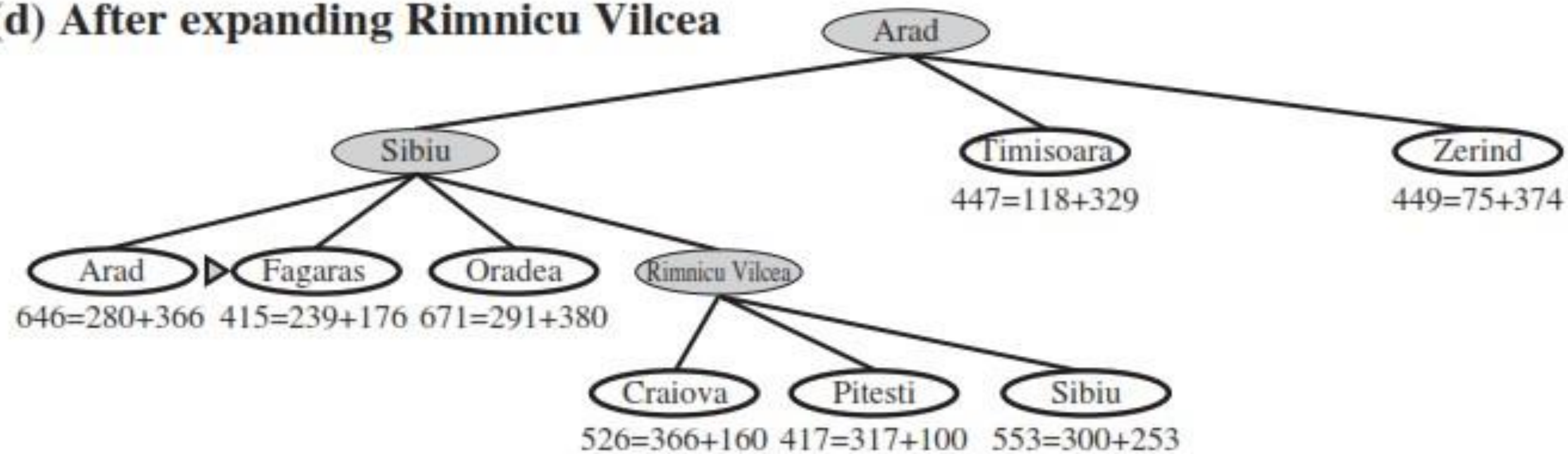
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Example

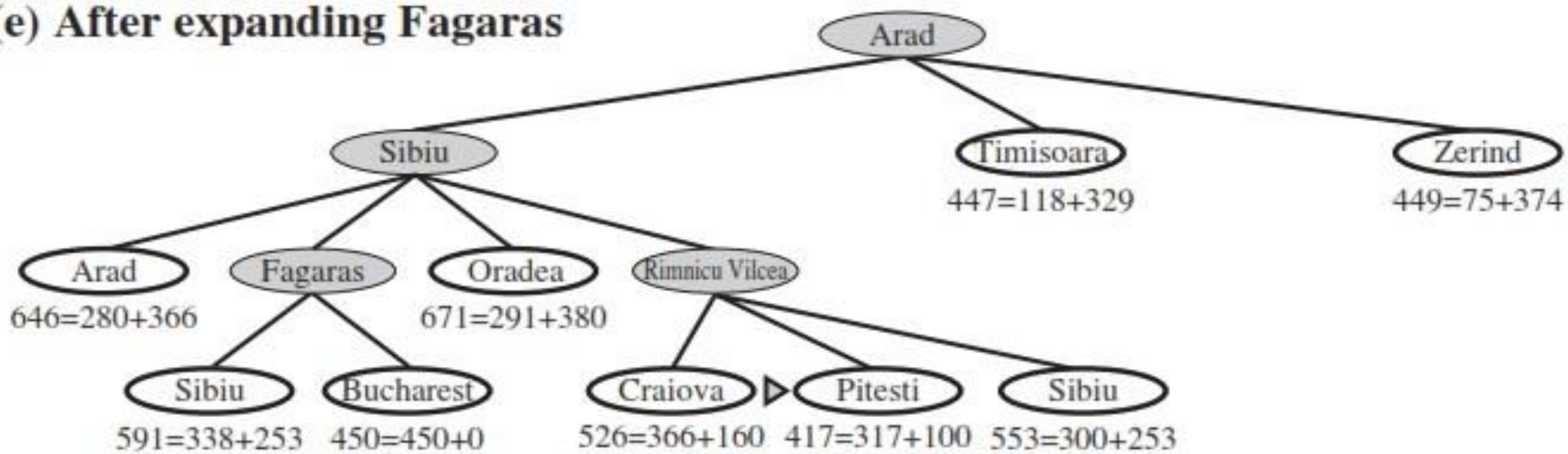

(a) The initial state
Arad
366=0+366
(b) After expanding Arad
Arad
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
413=220+193

Example

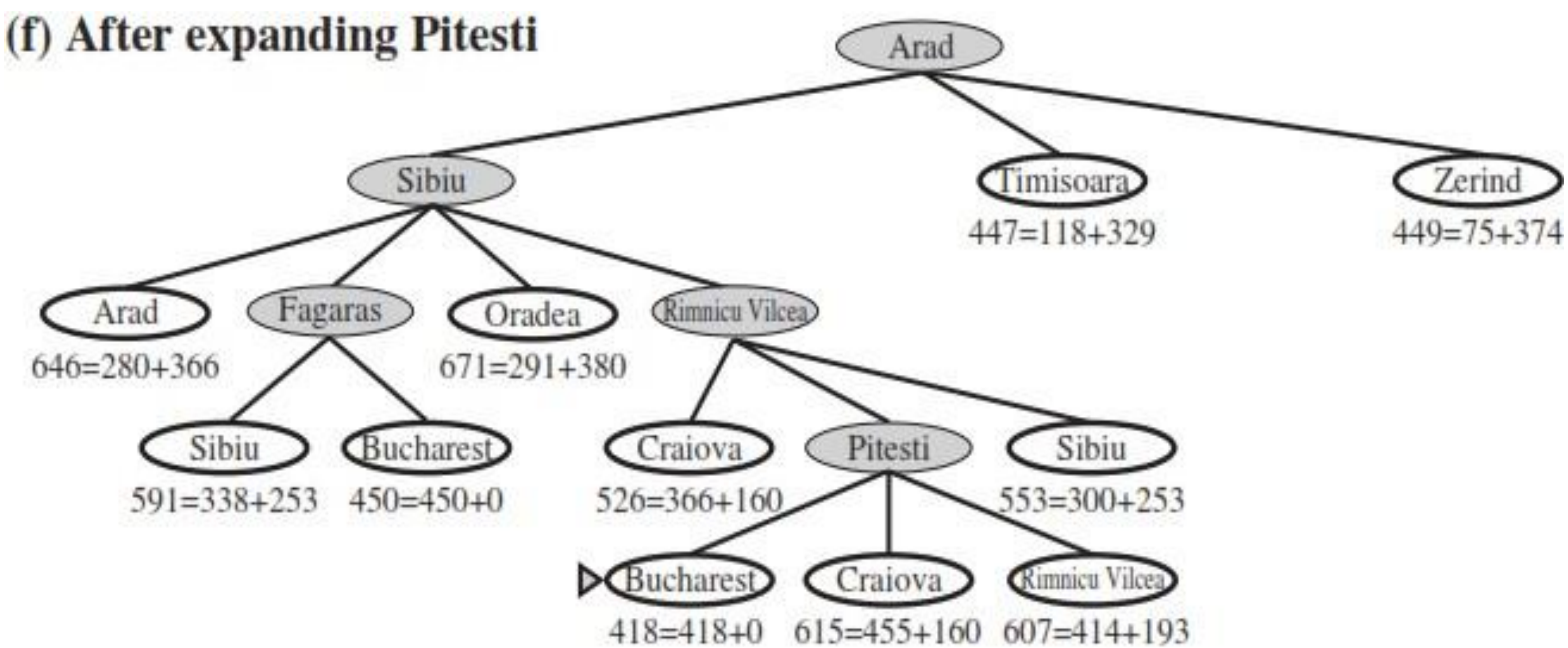
(d) After expanding Rimnicu Vilcea



(e) After expanding Fagaras



Example

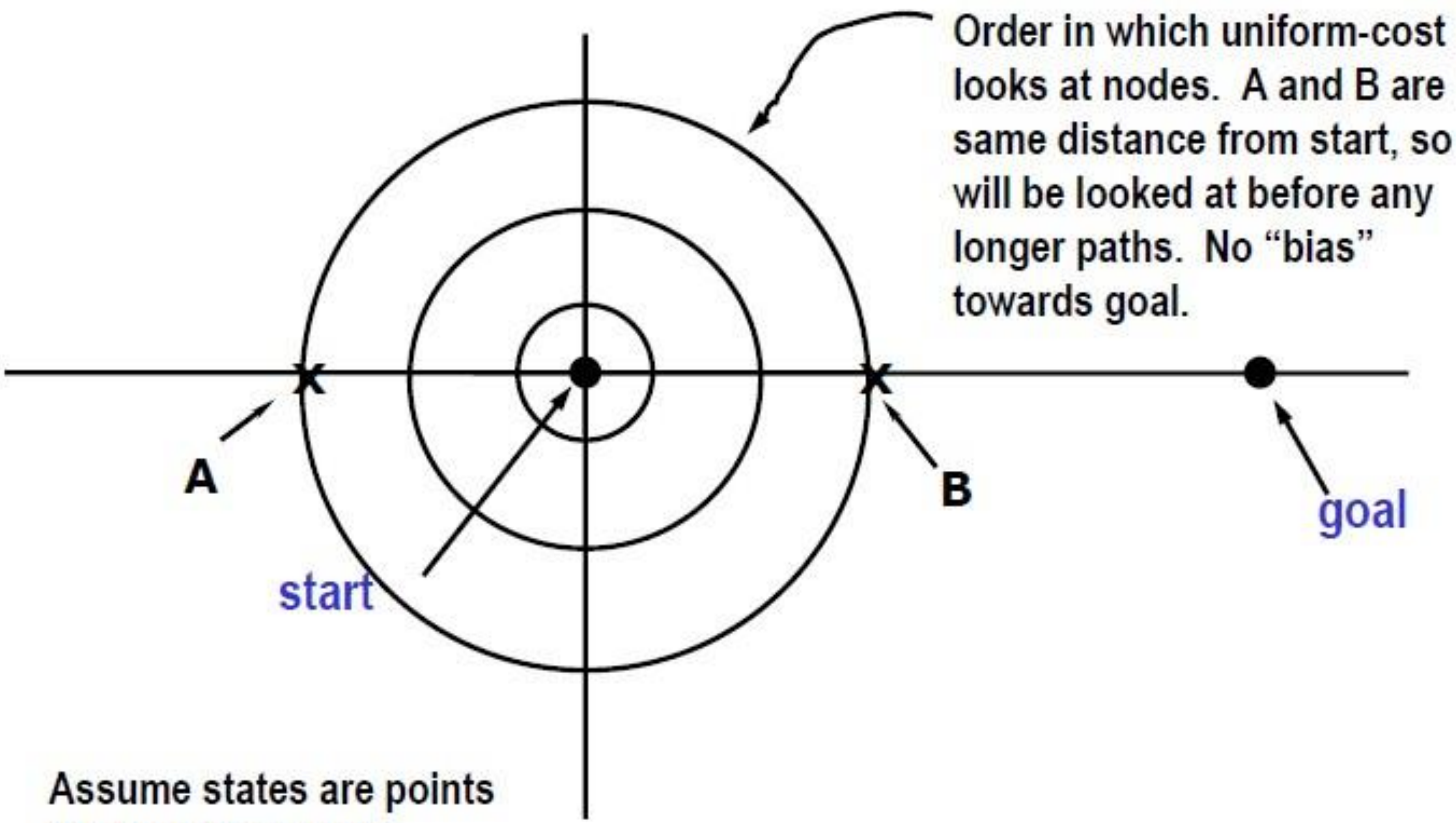


A* search for Bucharest: Nodes are labeled with $f = g + h$. The h values are the straight-line distances to Bucharest.

Why use estimate of goal distance?

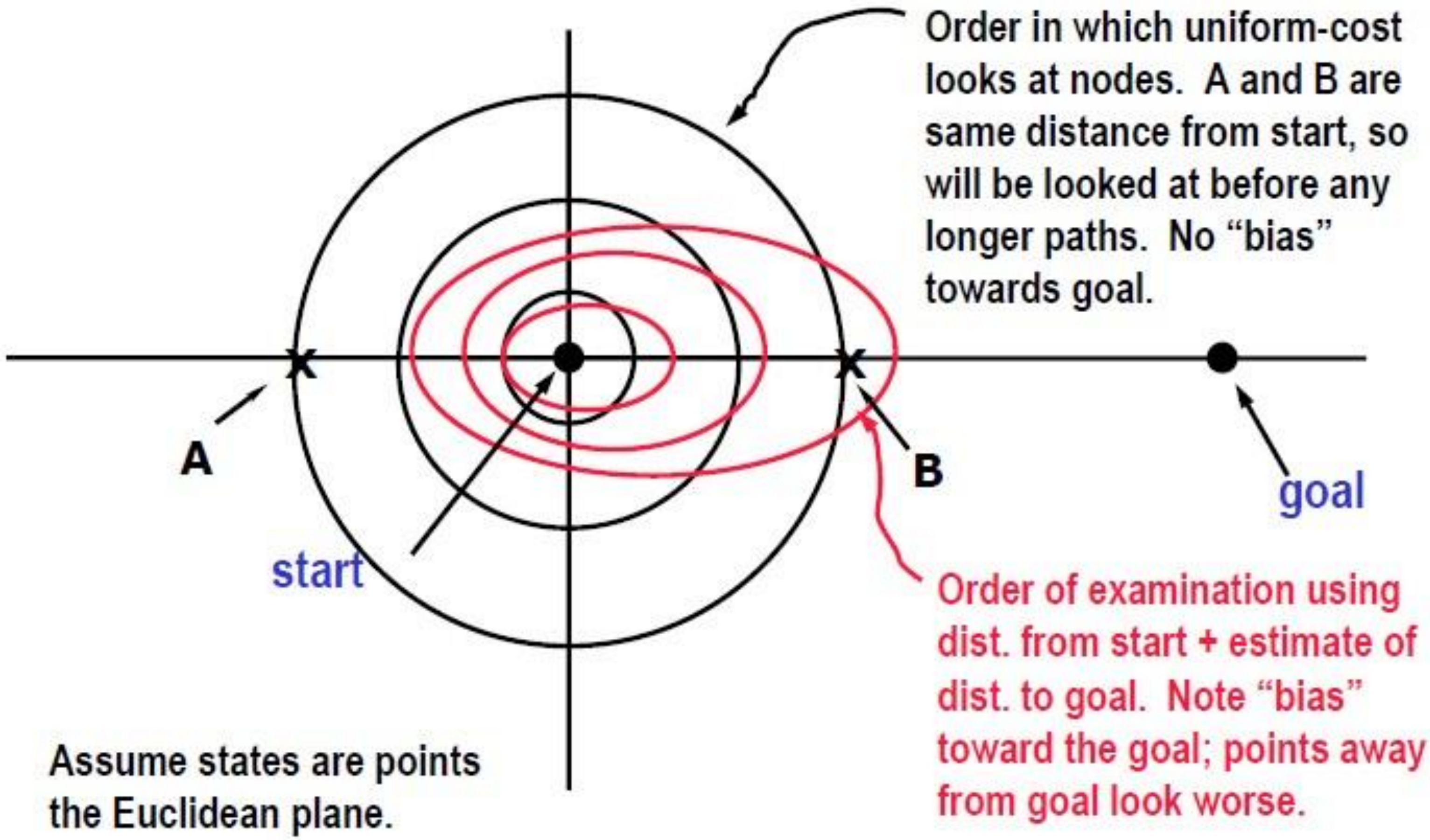
- You can think of A* as searching **contours** of distance from the **start state + estimated distance to the goal**.
- The **estimated/heuristic distance** term should **skew** the search in the direction of the goal.
- Heuristic doesn't mislead.
- How do you find a heuristic?
  - In the path-planning problem, it wasn't too hard to think of the shortest-line distance.

Why use estimate of goal distance?

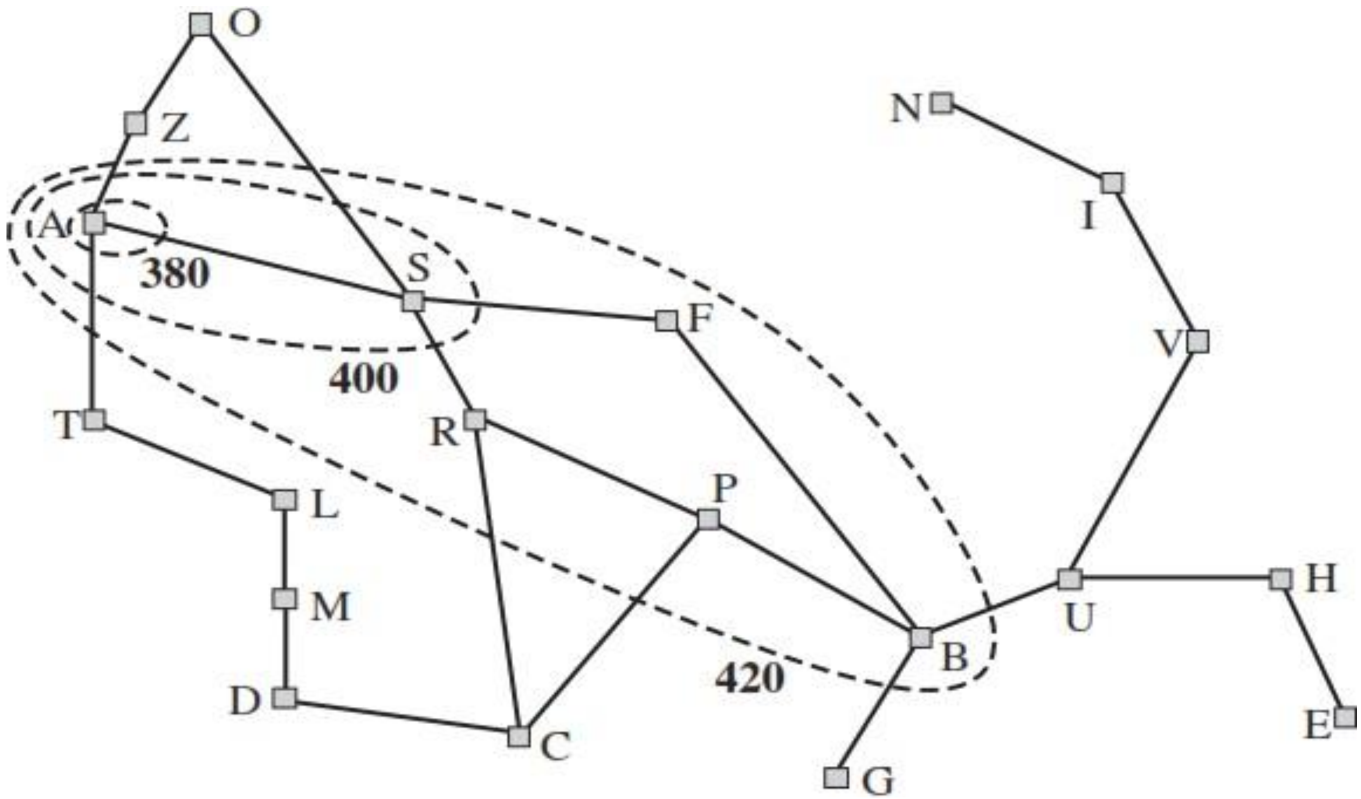


Assume states are points the Euclidean plane.

Why use estimate of goal distance?

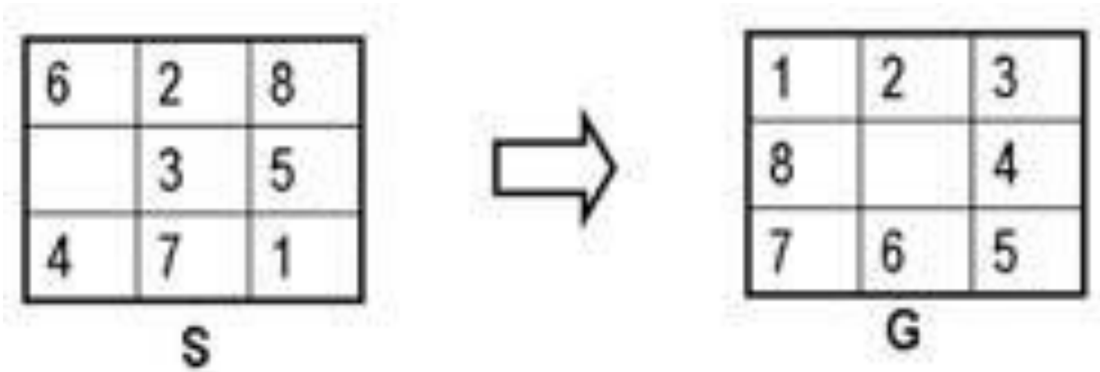


Why use estimate of goal distance?



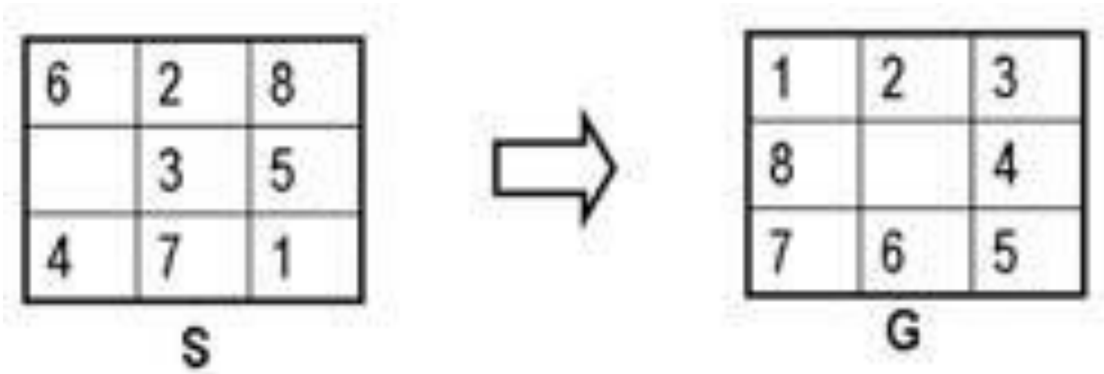
Map of Romania showing contours at $f = 380$, $f = 400$, and $f = 420$, with Arad as the start state.

Admissible Heuristic ...8-puzzle



- **The goal** is to arrange the pieces as in the goal state on the right.
- **A move** in this game is as sliding the **"empty"** space to one of its **nearest vertical or horizontal neighbours.**
- Move tiles to reach goal

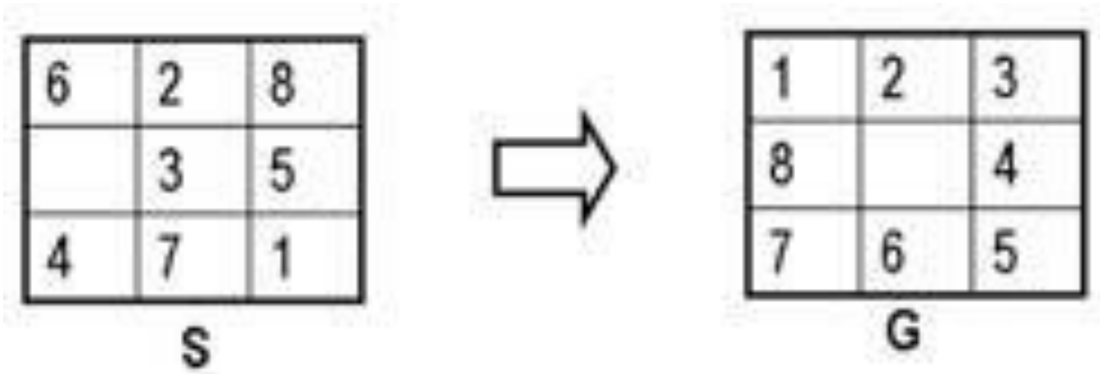
Admissible Heuristic ...8-puzzle



- Each move can at best decrease by one, the **"Manhattan distance"** of a tile from its goal.
- Manhattan distance, the metric of the Euclidean plane, is defined as

$$m\big((x_1, y_1)(x_2, y_2)\big) = |x_1 - x_2| + |y_1 - y_2|$$

Admissible Heuristic …8-puzzle



The alternative underestimates of "distance" (the number of moves) to goal:

- Number of misplaced tiles (7 in this case)
- Sum of manhattan distance, which is 17
  - **tile 1 = 4, tile 2 = 0, tile 3 = 2, tile 4 = 3, tile 5 = 1, tile 6 = 3, tile 7 = 1, tile 8 = 3**

Admissible Heuristics … Dominance

- If $\boldsymbol{h_2(n)} \geq \boldsymbol{h_1(n)}$ for all n (both admissible) then h_2 **dominates** h_1 and is better for search

$$
\begin{array}{ll}
d = 14 & \text{IDS} = 3{,}473{,}941 \text{ nodes} \\
 & \text{A}^*(h_1) = 539 \text{ nodes} \\
 & \text{A}^*(h_2) = 113 \text{ nodes} \\
d = 24 & \text{IDS} \approx 54{,}000{,}000{,}000 \text{ nodes} \\
 & \text{A}^*(h_1) = 39{,}135 \text{ nodes} \\
 & \text{A}^*(h_2) = 1{,}641 \text{ nodes}
\end{array}
$$

- Given admissible heuristics $h_1 \dots h_b$,

$$\boldsymbol{h(n) = max[h_1(n) \dots h_b(n)]}$$

is also admissible and dominates the other heuristics.

Admissible Heuristics … Dominance

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

	Search Cost (nodes generated)			Effective Branching Factor		
d	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Consistency

To implement A* Search, **heurstic $\boldsymbol{h}$ needs to be consistant** (sometimes monotonicity):

- The goal states have a heuristic estimate of zero.

$$h(n_i) = 0 \qquad if\ node\ n_i\ is\ the\ goal$$

- *The difference in the heuristic estimate between one state and its descendant must be less than or equal to the actual path cost.*

$$h(n) \leq c(n, a, n') + h(n')$$
$$h(n) - h(n') \leq c(n, a, n')$$
$$where\ n'\ is\ the\ successor\ of\ n.$$

Consistency Lemma

- ***If $h(n)$ is consistent, then the values of $f(n)$ along any path are nondecreasing.***

$$f(n') \geq f(n)$$

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

- **Proof:**

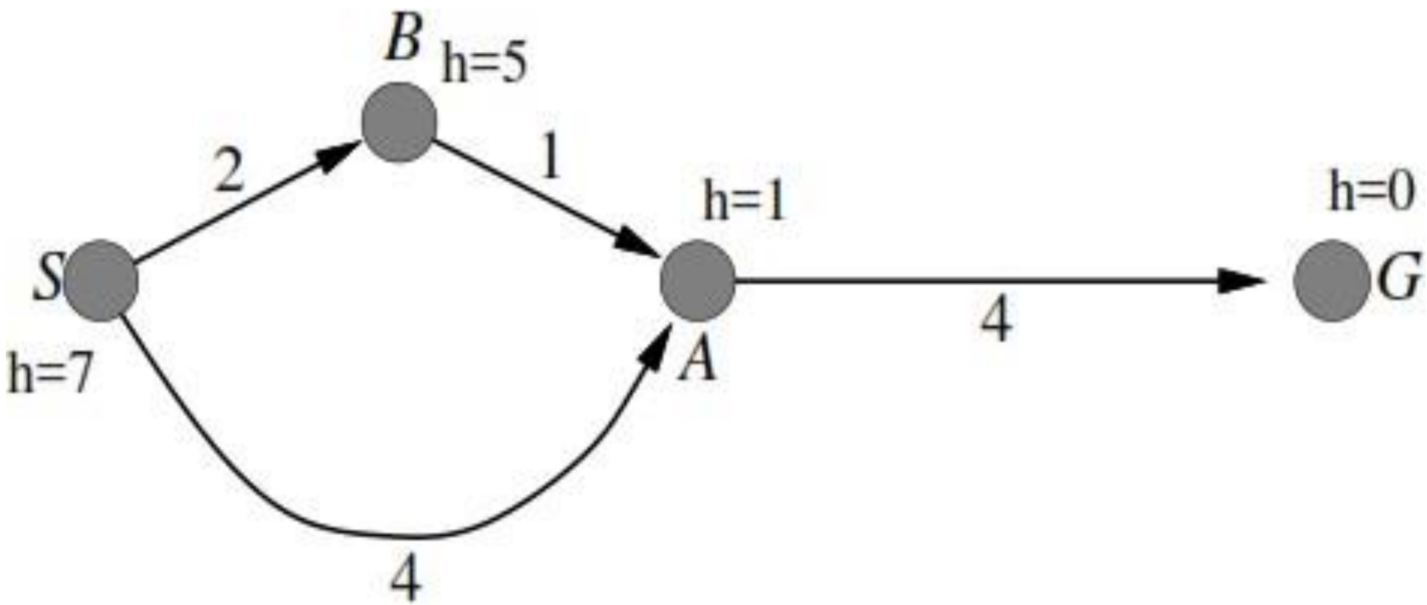
$$\text{Suppose } n' \text{ is the successor of } n$$

$$g(n') = g(n) + c(n, a, n')$$

$$f(n') = g(n') + h(n')$$
$$f(n') = g(n) + c(n, a, n') + h(n')$$
$$\geq g(n) + h(n)$$
$$f(n') \geq f(n)$$

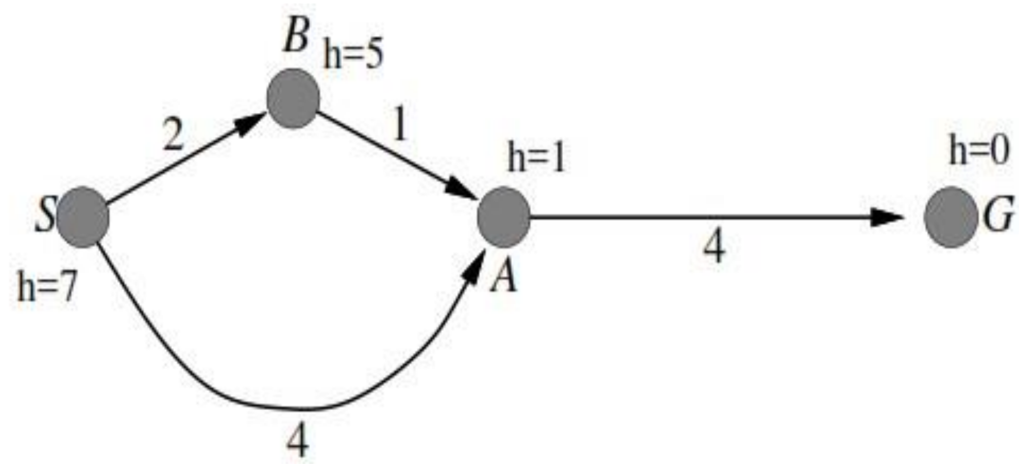
Example

- Are all heuristics admissible and consistent in this graph?



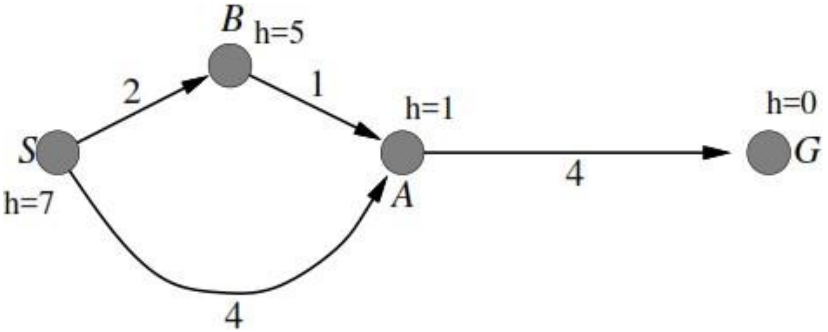
Example

- An estimate that always **underestimates the real path length to *<u>the goal</u>*** is called **admissible (heuristic) estimate.**
- **<u>Admissibility:</u>**
  - (S): $7 \le 7, 7 \le 8$
  - (A):$1 \le 4$
  - (B): $5 \le 5$



All heuristic values are admissible.

Example

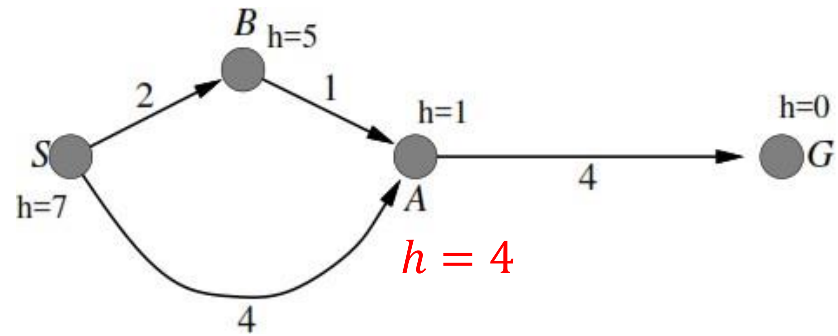


- **heurstic h is consistant, if**

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

- **<u>Consistency:</u>**
    - S→A: $7 - 1 \leq 4$ — Not Consistent
    - S→B: $7 - 5 \leq 2$ — Consistent
    - B→A: $5 - 1 \leq 1$ — Not Consistent
    - A→G: $1 - 0 \leq 4$ — Consistent

Example



- **heurstic h is consistant, if**

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

- To make them admissible and consistent, the heuristic value of ***A* is changed to 4**.
- **<u>Consistency:</u>**
  - S→A: $7 - 4 \leq 4$ Consistent
  - S→B: $7 - 5 \leq 2$ Consistent
  - B→A: $5 - 4 \leq 1$ Consistent
  - A→G: $4 - 0 \leq 4$ Consistent

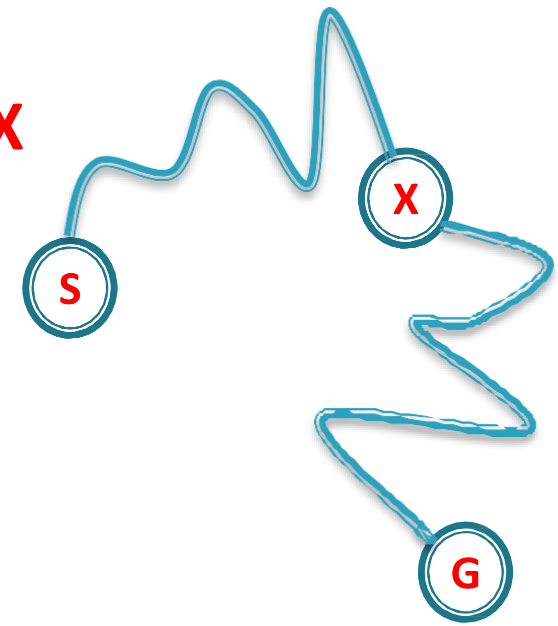
A* Search

- We **can bias Uniform-cost search** to find the shortest path to the goal.
- In fact, we are interested in by using a **heuristic function $h(n)$** which is an estimate of the distance from a state to the goal.
- It evaluates nodes by combining $g(n)$, the cost to reach the node, and $h(n)$, the cost to get from the node to the goal:

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

Dynamic Programming Optimality Principle

- *Given that **path length is additive**, the shortest path from **S** to **G** via a state **X** is made up of the shortest path from **S** to **X** and the shortest path from **X** to **G**.*
- **Only need to keep the shortest path**, discard all other paths.
- Once we expand path to state **X**, we do not need to expand any other path to state **X**.



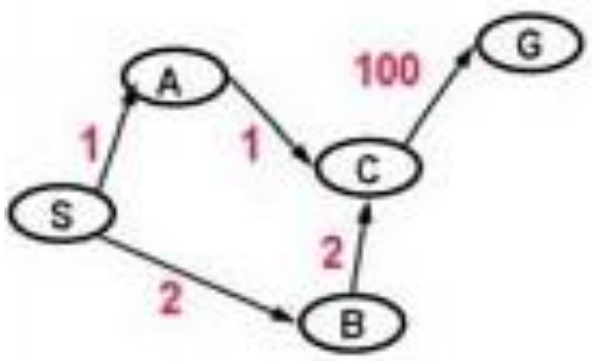
A* Search without Expanded List

A* Search

(without Expanded List)

- Pick best (by **path length + heuristic value**) element of Q
- Add path extensions to Q

	Q
1	<u>(90 S)</u>
2	**(101 A S)** **<u>(3 B S)</u>**
3	**<u>(94 C B S)</u>** (101 A S)
4	<u>(101 A S)</u> **(104 G C B S)**
5	**<u>(92 C A S)</u>** (104 G C B S)
6	**(102 G C A S)** (104 G C B S)



- **Blue Color represents added paths**
- <u>Underline</u> paths are selected for extension

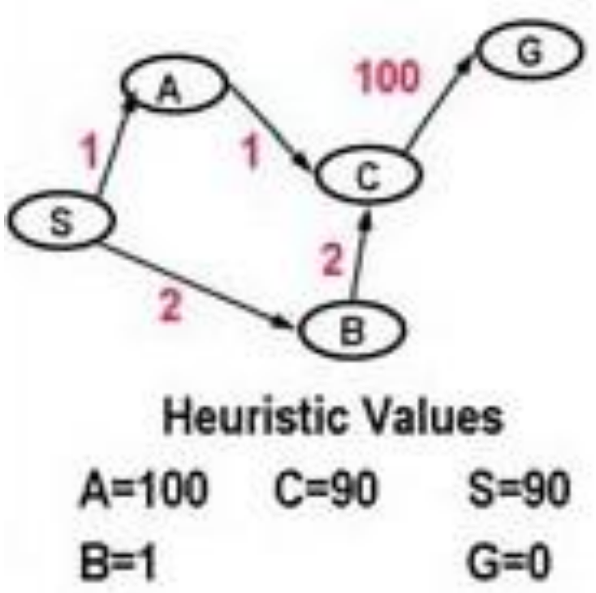
A* Search with Expanded List

A* Search

(with strict Expanded List)

- ❑ Pick best (by **path length + heuristic value**) element of Q
- ❑ Add path extensions to Q

	Q	Expanded
1	<u>(90 S)</u>	S
2	**(101 A S)** **<u>(3 B S)</u>**	S, B
3	**<u>(94 C B S)</u>** (101 A S)	S, B, C
4	<u>(101 A S)</u> **(104 G C B S)**	S, B, C, A
5	**(104 G C B S)** ~~(92 C A S)~~	S, B, C, A, **G**



- ❑ **Blue Color represents added paths**
- ❑ <u>Underline</u> paths are selected for extension.

But the shortest path to goal is **G,C,A,S** and the cost is 102.

A* Search

(with strict Expanded List)

- ❑ Pick best (by **path length + heuristic value**) element of Q
- ❑ Add path extensions to Q

$$h(S) - h(A) \le c(S,a,A)$$
$$90 - 100 \le 1$$
$$-10 \le 1$$

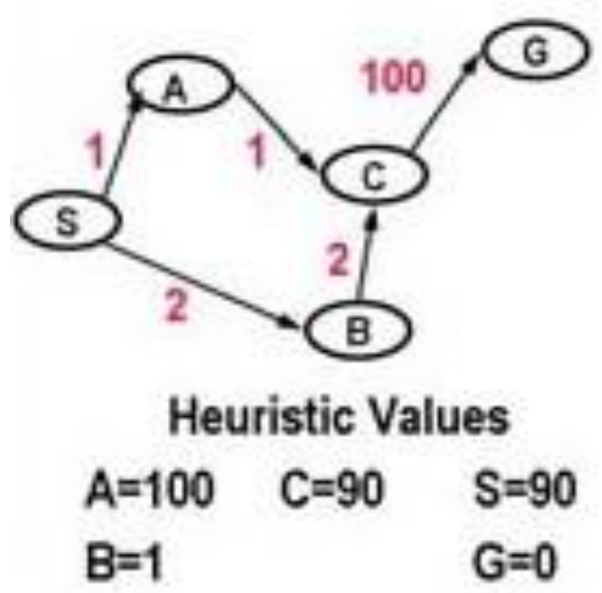
$$h(B) - h(C) \le c(S,a,A)$$
$$1 - 90 \le 1$$
$$-90 \le 1$$

$$\boldsymbol{h(S) - h(B) \le c(S,B)}$$
$$90 - 1 \le 2$$
$$89 \le 1$$

$$\boldsymbol{h(A) - h(C) \le c(A,C)}$$
$$100 - 90 \le 1$$
$$10 \le 1$$

$$\boldsymbol{h(S) - h(B) \le c(S,B)}$$
$$\boldsymbol{90 - 88 \le 2}$$

$$\boldsymbol{h(A) - h(C) \le c(A,C)}$$
$$\boldsymbol{100 - 99 \le 1}$$



A* Search

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

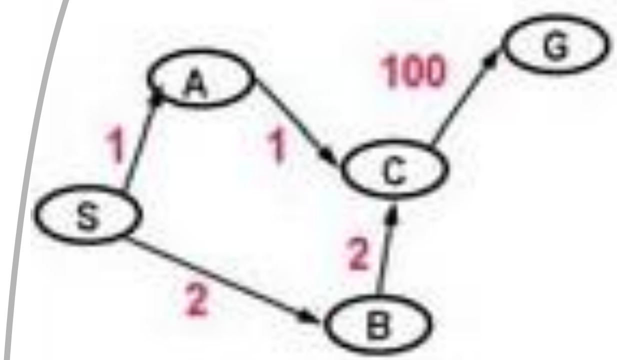
(with strict Expanded List)

$$h(S) - h(B) \leq c(S, B)$$
$$90 - 1 \leq 2$$
$$89 \leq 1$$

$$h(S) - h(B) \leq c(S, B)$$
$$90 - 88 \leq 2$$

- Pick best (by **path length + heuristic value**) element of Q
- Add path extensions to Q

	Q	Expanded
1	<u>(90 S)</u>	S
2	**(101 A S)** <u>**(3 B S)**</u>	S, B
3	<u>**(94 C B S)**</u> (101 A S)	S, B, C
4	<u>(101 A S)</u> **(104 G C B S)**	S, B, C, A
5	**(104 G C B S)** ~~(92 C A S)~~	G, S, B, C, A



Heuristic Values

A=100 C=90 S=90

B=88 B=1 C=99 G=0

$$h(A) - h(C) \leq c(A, C)$$
$$100 - 90 \leq 1$$
$$10 \leq 1$$

$$h(A) - h(C) \leq c(A, C)$$
$$100 - 99 \leq 1$$

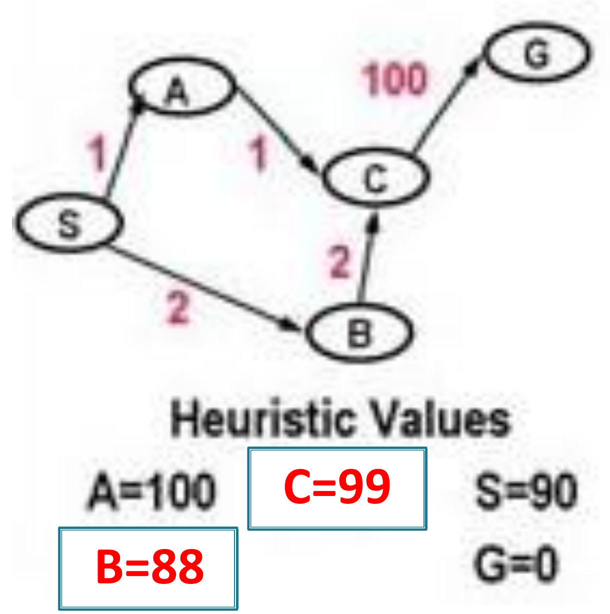
- **Blue Color represents added paths**
- <u>Underline</u> paths are selected for extension.

A* Search

(with strict Expanded List)

- Pick best (by **path length + heuristic value**) element of Q
- Add path extensions to Q
- **Heuristic is admissible and consistent**

	Q	Expanded
1	<u>(90 S)</u>	S
2	**(101 A S)** **<u>(90 B S)</u>**	S, B
3	**(103 C B S)** <u>(101 A S)</u>	S, B, A
4	**<u>(101 C A S)</u>** ~~(103 C B S)~~	S, B, A, C,
5	**(102 G C A S)**	S, B, A, C, **G**



- **Blue Color represents added paths**
- <u>Underline</u> paths are selected for extension

A* Search with Pathmax

$$h\ n\ - h\ n'\ \le c\ n, a, n'$$
$$h\ n\ \le c\ n, a, n'\ + h\ n'$$

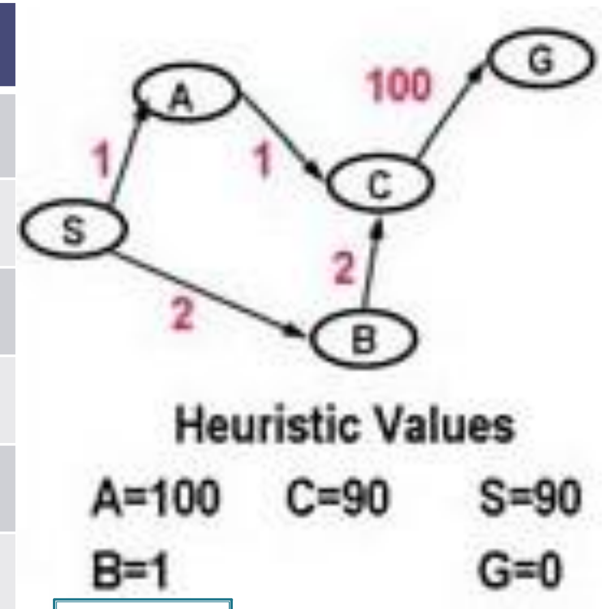
$$h(S) - h(A) \le c(S, a, A)$$
$$90 - 100 \le 1$$
$$-10 \le 1$$

$$h(S) - h(B) \le c(S, a, B)$$
$$90 - 1 \le 2$$
$$89 \le 1$$

- Pick best (**path length + heuristic value**) element of Q
- Add path extensions to Q
- **Heuristic is admissible but not consistent**

	Q	Expanded
1	(90 S)	[S, 90]
2	**(101 A S) (90 B S)**	[B, 90], S

Pathmax changes f value from 3 to 90



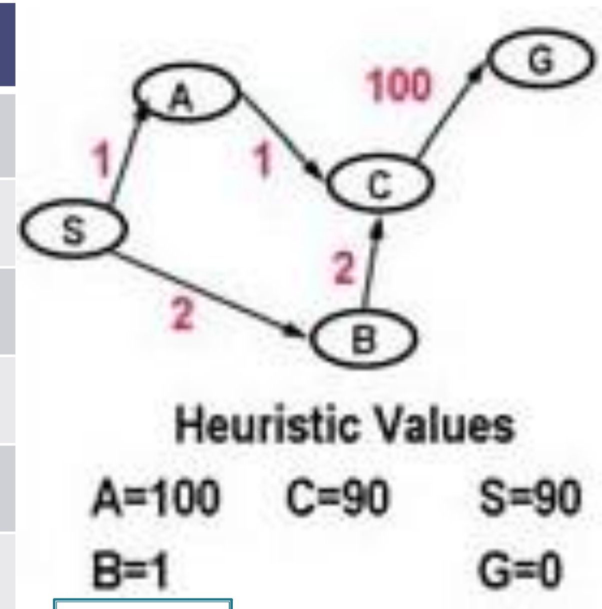
B=88

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

$$h(B) - h(C) \leq c(B, a, C)$$
$$88 - 90 \leq 2$$
$$-2 \leq 2$$

- Pick best (**path length + heuristic value**) element of Q
- Add path extensions to Q
- **Heuristic is admissible but not consistent**

	Q	Expanded
1	(90 S)	[S, 90]
2	**(101 A S)** **(90 B S)**	[B, 90], S
3	**(94 C B S)** (101 A S)	[C, 94], B, S



B=88

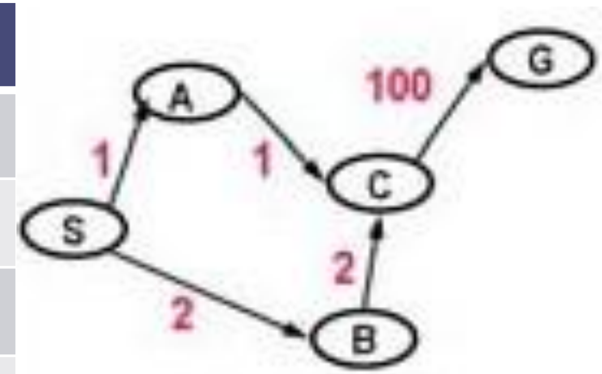
Pathmax changes f value from 3 to 90

$$h\ n\ - h\ n' \le c\ n, a, n'$$
$$h\ n\ \le c\ n, a, n' + h\ n'$$

$$h(C) - h(G) \le c(B, a, C)$$
$$90 - 0 \le 100$$
$$90 \le 100$$

- Pick best (by **path length + heuristic value**) element of Q
- Add path extensions to Q
- **Heuristic is admissible but not consistent**

	Q	Expanded
1	(90 S)	[S, 90]
2	**(101 A S)** **(90 B S)**	[B, 90], S
3	**(94 C B S)** (101 A S)	[C, 94], B, S
4	(101 A S) **(104 G C B S)**	[A, 101], B, S



Heuristic Values

A=100 C=90 S=90

B=1 G=0

B=88

Pathmax changes f value from 3 to 90

A* Search

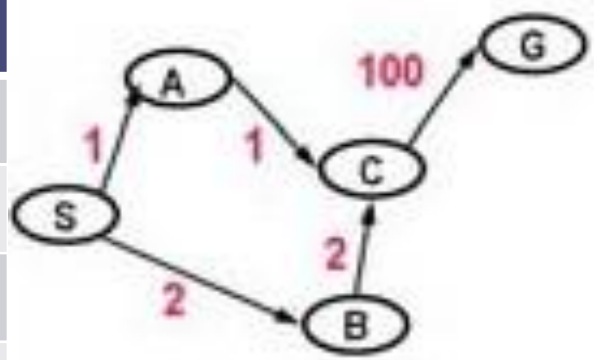
$$h(n) - h(n') \le c(n,a,n')$$
$$h(n) \le c(n,a,n') + h(n')$$

$$h(A) - h(C) \le c(B,a,C)$$
$$100 - 90 \le 1$$
$$10 \le 1$$

(with pathmax and Expanded List)

- Pick best (**path length + heuristic value**) element of Q
- Add path extensions to Q
- **Heuristic is admissible but not consistent**

	Q	Expanded
1	(90 S)	[S, 90]
2	**(101 A S)** **(90 B S)**	[B, 90], S
3	**(94 C B S)** (101 A S)	[C, 94], B, S
4	(101 A S) **(104 G C B S)**	[A, 101], B, S
5	**(101 C A S)** (104 G C B S)	[C, 101], A, B, S



Heuristic Values

A=100 C=90 S=90
B=1 C=99 G=0
B=88

Pathmax changes f value from 3 to 90

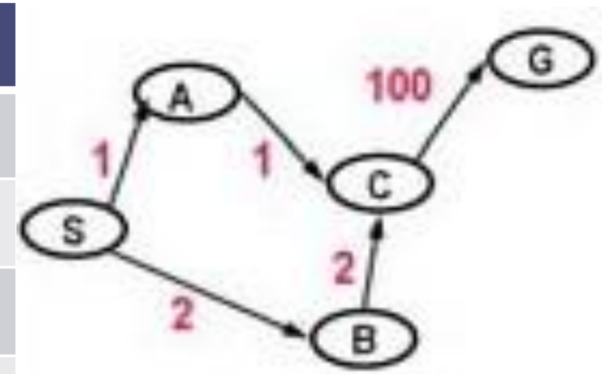
Pathmax changes f value from 92 to 101, node is added to Q even though C is on expanded list (and C is removed from expanded).

A* Search

(with pathmax and Expanded List)

- Pick best (by **path length + heuristic value**) element of Q
- Add path extensions to Q
- **Heuristic is admissible but not consistent**

	Q	Expanded
1	(90 S)	[S, 90]
2	**(101 A S)** **(90 B S)**	[B, 90], S
3	**(94 C B S)** (101 A S)	[C, 94], B, S
4	(101 A S) **(104 G C B S)**	[A, 101], B, S
5	**(101 C A S)** (104 G C B S)	[C, 101], A, B, S
6	**(102 G C A S)** ~~(104 G C B S)~~	[G, 0], A, B, S, C



Heuristic Values

A=100 C=90 S=90

B=1 **C=99** G=0

B=88

Pathmax changes f value from 3 to 90

Pathmax changes f value from 92 to 101, node is added to Q even though C is on expanded list (and C is removed from expanded).

A* Search

(with pathmax and expanded List)

- **In step 2**, when we generate a path to B, we need to modify the value of $\boldsymbol{h(B)}$ drastically,
  - The estimate at S is 90 and the edge length to B is 2, then the estimate at B is:

$$\boldsymbol{h(S) - h(B) \ \leq \ c(S, B)}$$
$$\boldsymbol{90 - ? \ \leq \ 2}$$

So, the lowest consistent value for $\boldsymbol{h(B)}$ is 88, and $\boldsymbol{f(B)}$ becomes,

$$\begin{aligned} \boldsymbol{f(B)} &= \boldsymbol{h(B) + c(S, B)} \\ &= \boldsymbol{88 + 2} \\ &= \boldsymbol{90\ (not\ 3).} \end{aligned}$$

A* Search

- **Complete:** Yes
  - A* is complete and optimal, provided that $h(n)$ is admissible (for TREE-SEARCH) or consistent (for GRAPH-SEARCH).
- **Optimal:** Yes, A* is **optimally efficient** for any given consistent heuristic.
- **Time:** Exponential
  - The complexity results depend very strongly on the **assumptions** made about the state space.
  - The complexity of A∗ often makes it **impractical** to insist on finding an optimal solution.
- **Space:** Keeps all nodes in memory,
  - A* usually runs out of space long before it runs out of time.

A* Search

- For problems with constant step costs, the growth in run time as a function of the optimal solution depth d is analyzed in terms of the **absolute error** or the **relative error** of the heuristic.
- The **absolute error** is defined as

$$\Delta \equiv h^* - h$$

where h^* is the actual cost of getting from the root to the goal

- The **relative error** is defined as

$$\varepsilon \equiv (h^* - h)/h^*$$

A* Search

The 8-puzzle problem with single goal:

- The **time complexity of A*** is exponential in the maximum absolute error, that is,

$$\boldsymbol{O}(\boldsymbol{b}^{\Delta})$$

- For constant step costs, we can write this as

$$\boldsymbol{O}(\boldsymbol{b}^{\epsilon \boldsymbol{d}})$$

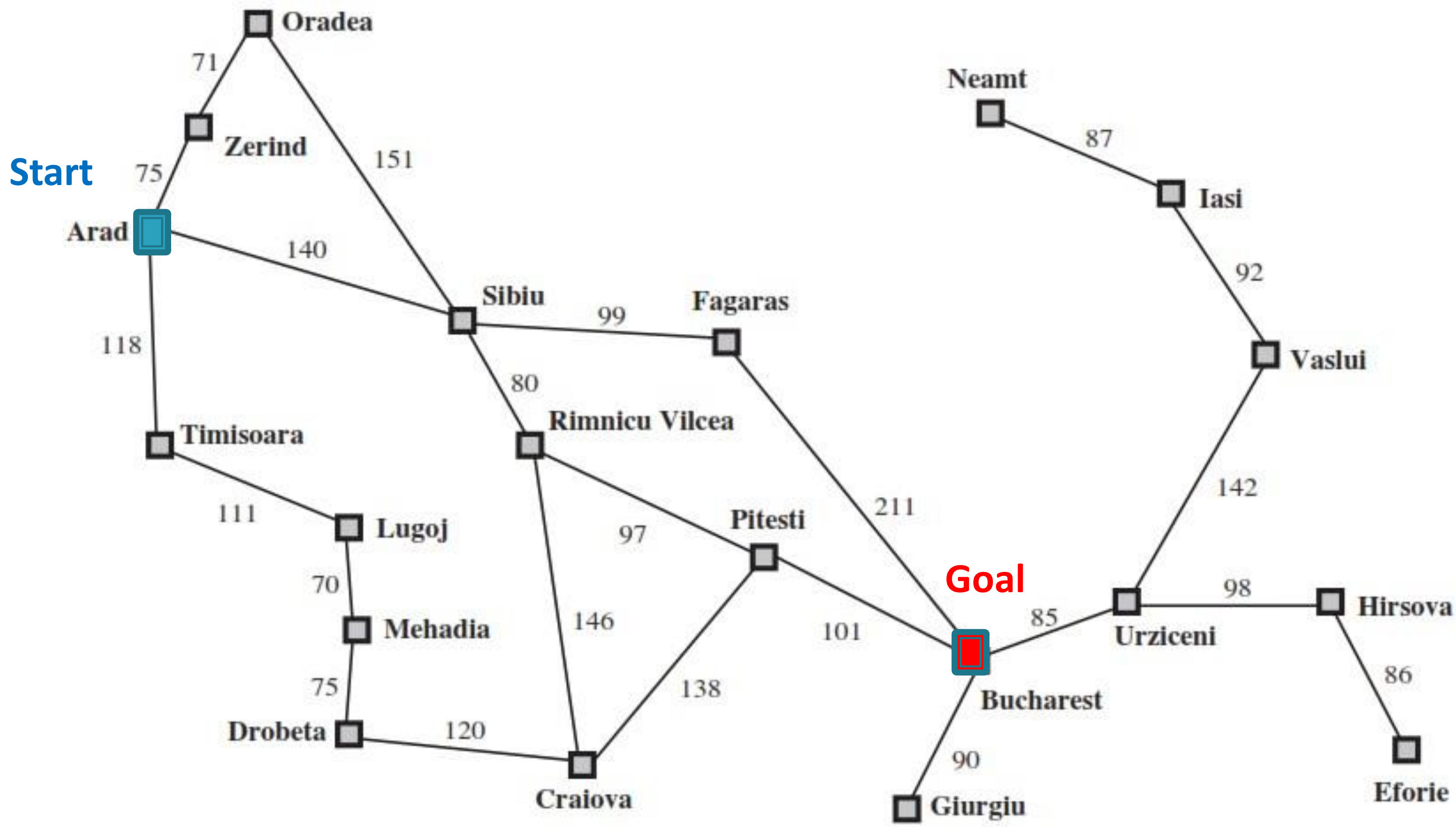
where d is the solution depth.

Memory-bounded Heuristic Search … (RBFS)

Recursive Best-First Search (RBFS)

- **Recursive Best-First Search** (RBFS) is a **simple recursive algorithm** that attempts to mimic the operation of standard best-first search and A* search,
  - but using only linear space
- It uses the $\boldsymbol{f\_limit}$ variable to keep track of the $\boldsymbol{f\_value}$ of the best *alternative* path.
- If the current node exceeds this limit, the recursion unwinds back to the alternative path.
  - As the recursion unwinds, RBFS replaces the $\boldsymbol{f\_value}$ of each node along the path with the best $\boldsymbol{f\_value}$ of its children.

RBFS Example



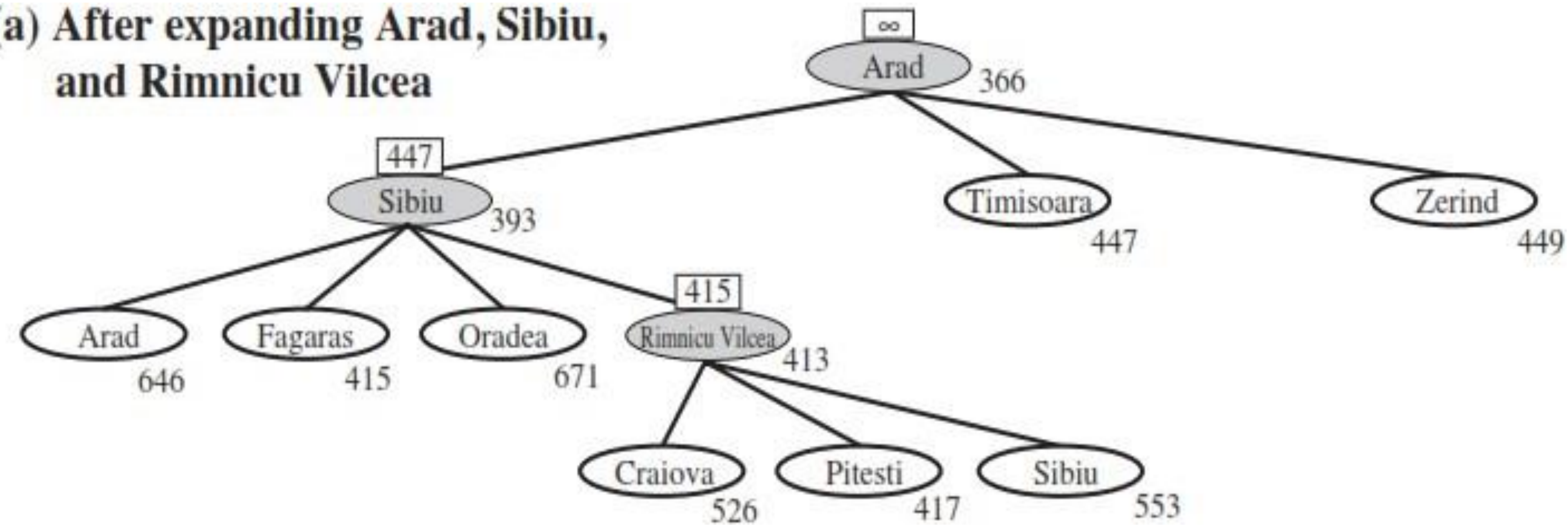
A simplified road map of part of Romania.

Values of h_{SLD}—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

The *f_limit* value fo…
recursive call is show…
each current node, a…
node is labeled with its ***f_cost***.

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

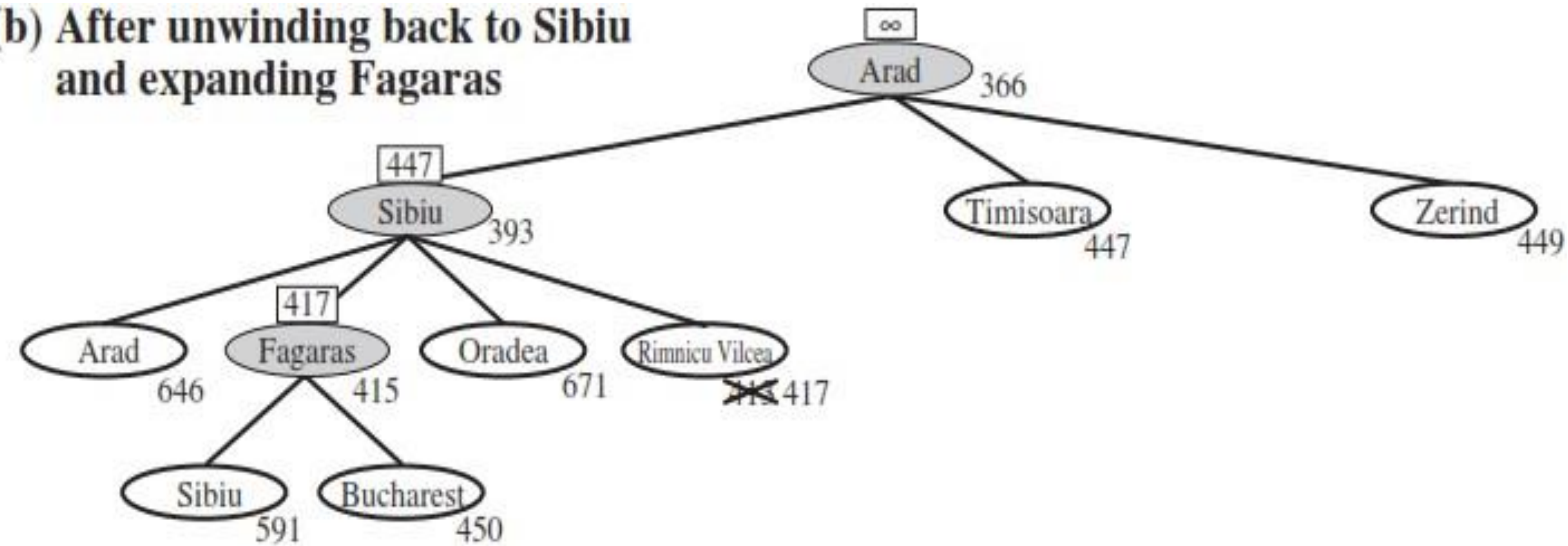


(a) After expanding Arad, Sibiu, and Rimnicu Vilcea

RBFS 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

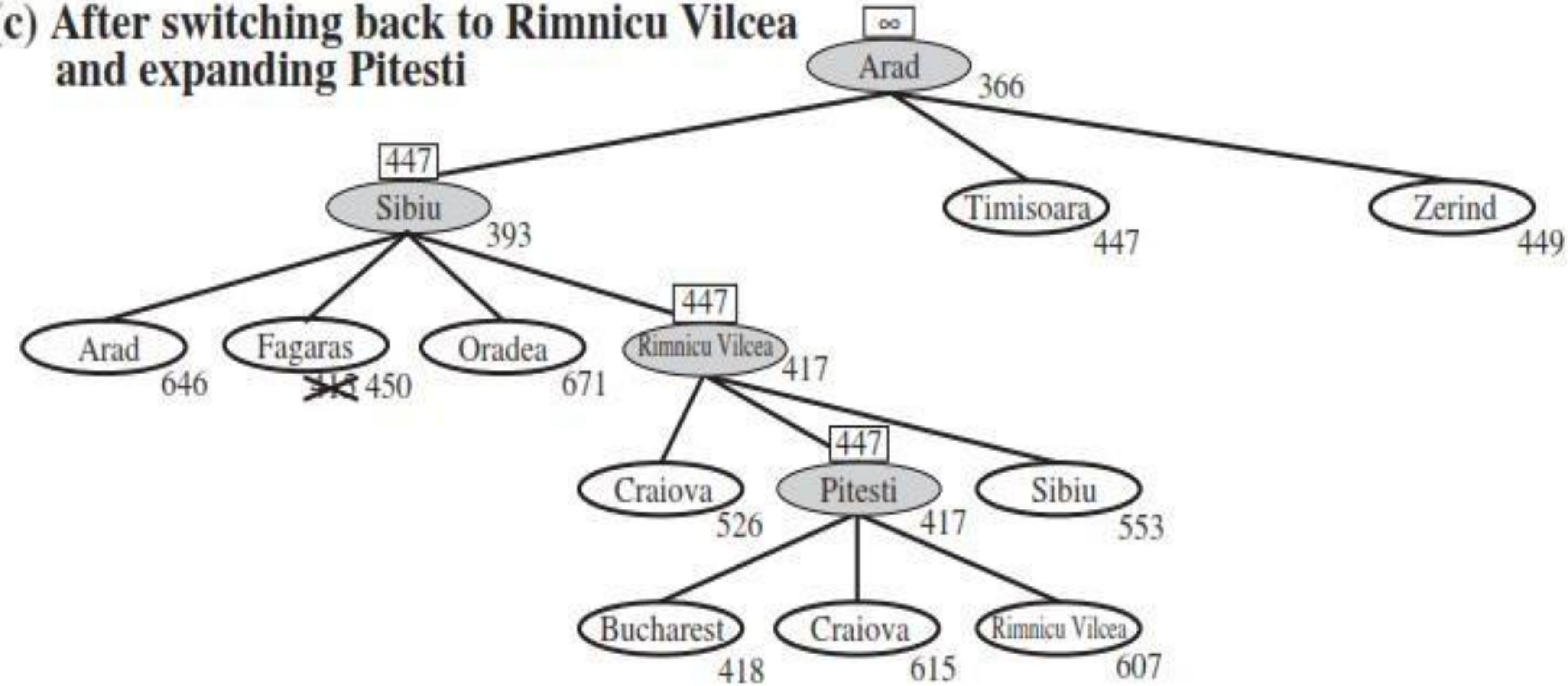
(b) After unwinding back to Sibiu and expanding Fagaras



RBFS 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

(c) After switching back to Rimnicu Vilcea and expanding Pitesti



Recursive Best-first Search (RBFS)

- Like A$*$ tree search, RBFS is an **optimal algorithm** if the *heuristic function* $h(n)$ *is admissible*.
- Its **space complexity** is *linear* in the depth of the deepest optimal solution,
- Its **time complexity** is rather difficult to characterize: it depends both on
  - The *accuracy of the heuristic function* and
  - *How often the best path changes* as nodes are expanded.