CS 461
Artificial Intelligence

Dr. Hashim Yasin

A* Search

A* Search

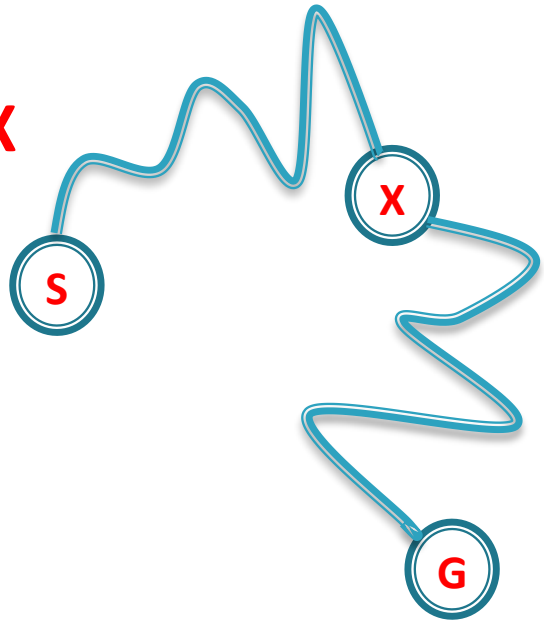
- We **can bias Uniform-cost search** to find the shortest path to the goal.
- In fact, we are interested in by using a **heuristic function $h(n)$** which is an estimate of the distance from a state to the goal.
- It evaluates nodes by combining $g(n)$, the cost to reach the node, and $h(n)$, the cost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

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

Optimality Principle

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**.



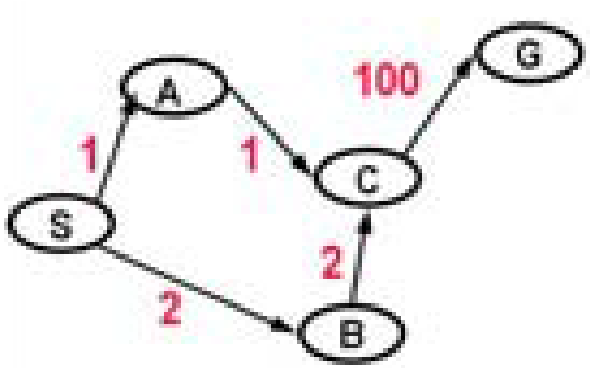
A* Search without Expanded List

A* Search

(without Expanded List)

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

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



- **Blue Color represents added paths**
- Underline paths are selected for extension

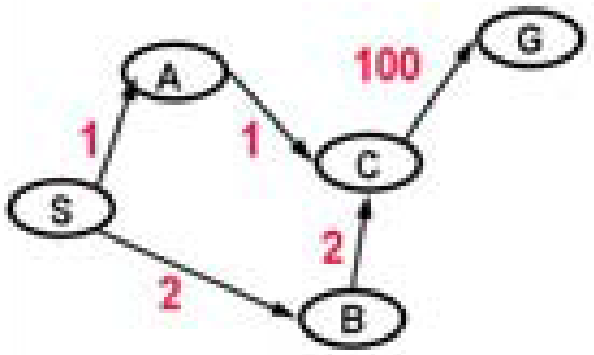
A* Search with Expanded List

A* Search

(with strict Expanded List)

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

	Q	Expanded
1	(90 S)	S
2	**(101 A S) (3 B S)**	S, B
3	**(94 C B S)** (101 A S)	S, B, C
4	(101 A S) **(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**
- Underline paths are selected for extension.

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

A* Search

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

(with strict Expanded List)

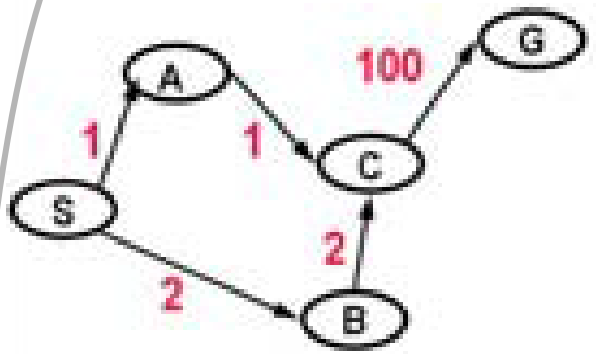
- Pick best (by **path length + heuristic value**) element of Q
- Add path extensions to Q

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

- **Blue Color represents added paths**
- Underline paths are selected for extension.

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

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



Heuristic Values

A=100 C=90 S=90

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

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

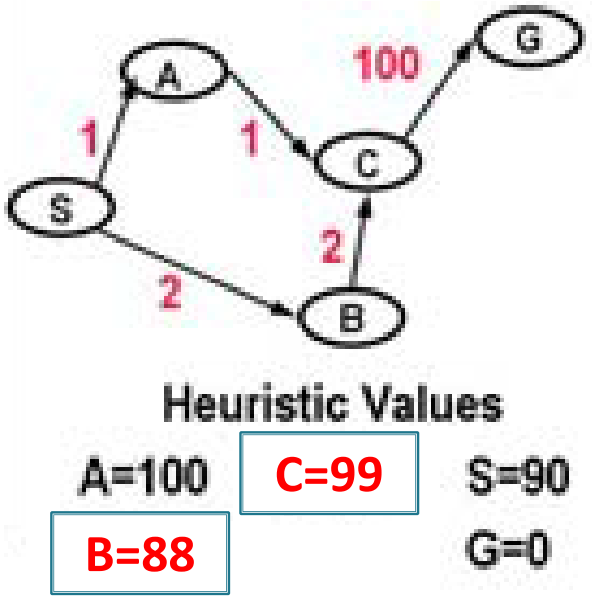
$$h(A) - h(C) \leq c(A, C)$$
$$100 - 99 \leq 1$$

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	(90 S)	S
2	(101 A S) (90 B S)	S, B
3	(103 C B S) (101 A S)	S, B, A
4	(101 C A S) ~~(103 C B S)~~	S, B, A, C,
5	(102 G C A S)	S, B, A, C, G



- **Blue Color represents added paths**
- Underline paths are selected for extension

A* Search with Pathmax

A* Search

(with pathmax and Expanded List)

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

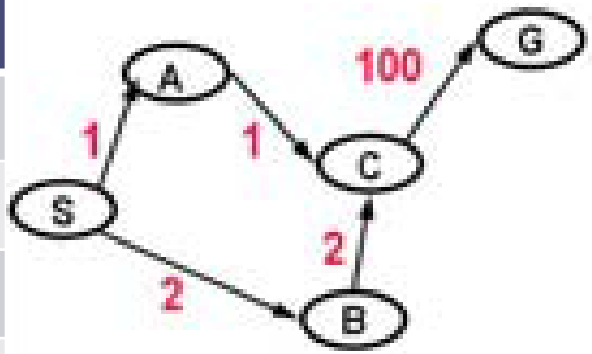
$$h(S) - h(A) \leq c(S, a, A)$$
$$90 - 100 \leq 1$$
$$-10 \leq 1$$

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

- 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



Heuristic Values

A=100 C=90 S=90

B=1 G=0

B=88

A* Search

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

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

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

$88 - 90 \leq 2$

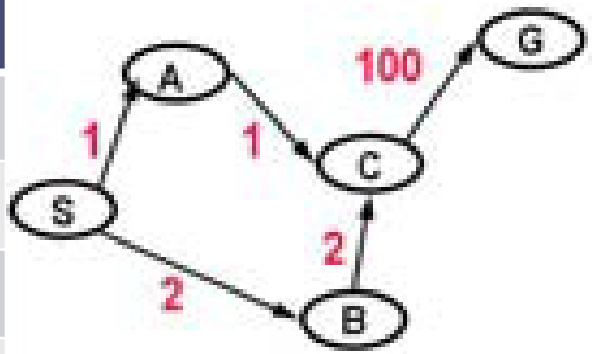
$-2 \leq 2$

(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

Pathmax changes f value from 3 to 90



Heuristic Values

A=100 C=90 S=90

B=1 G=0

B=88

A* Search

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

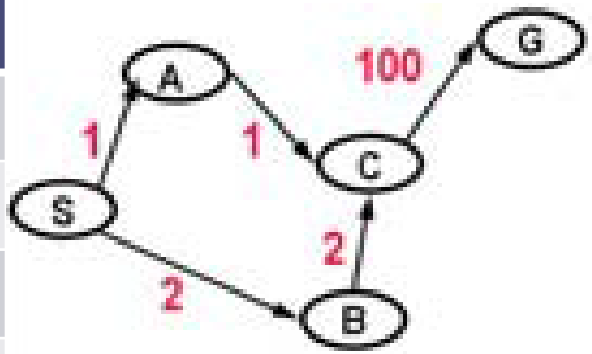
$$h(C) - h(G) \leq c(B, a, C)$$
$$90 - 0 \leq 100$$
$$90 \leq 100$$

(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

Pathmax changes f value from 3 to 90



Heuristic Values

A=100 C=90 S=90

B=1 G=0

B=88

A* Search

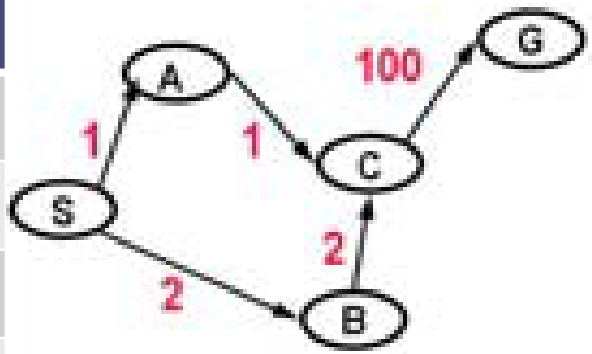
$$h(n) - h(n') \leq c(n, a, n')$$
$$h(n) \leq c(n, a, n') + h(n')$$

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

(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

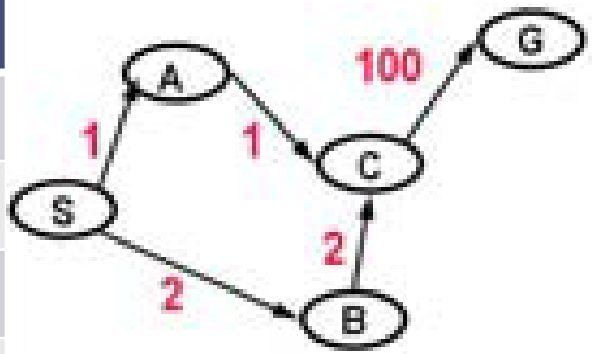
Pathmax changes f value from 92 to 101, node is added to Q even though C is on expanded list (and C is removed from expanded).

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,

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

- For constant step costs, we can write this as

$$O(b^{\varepsilon d})$$

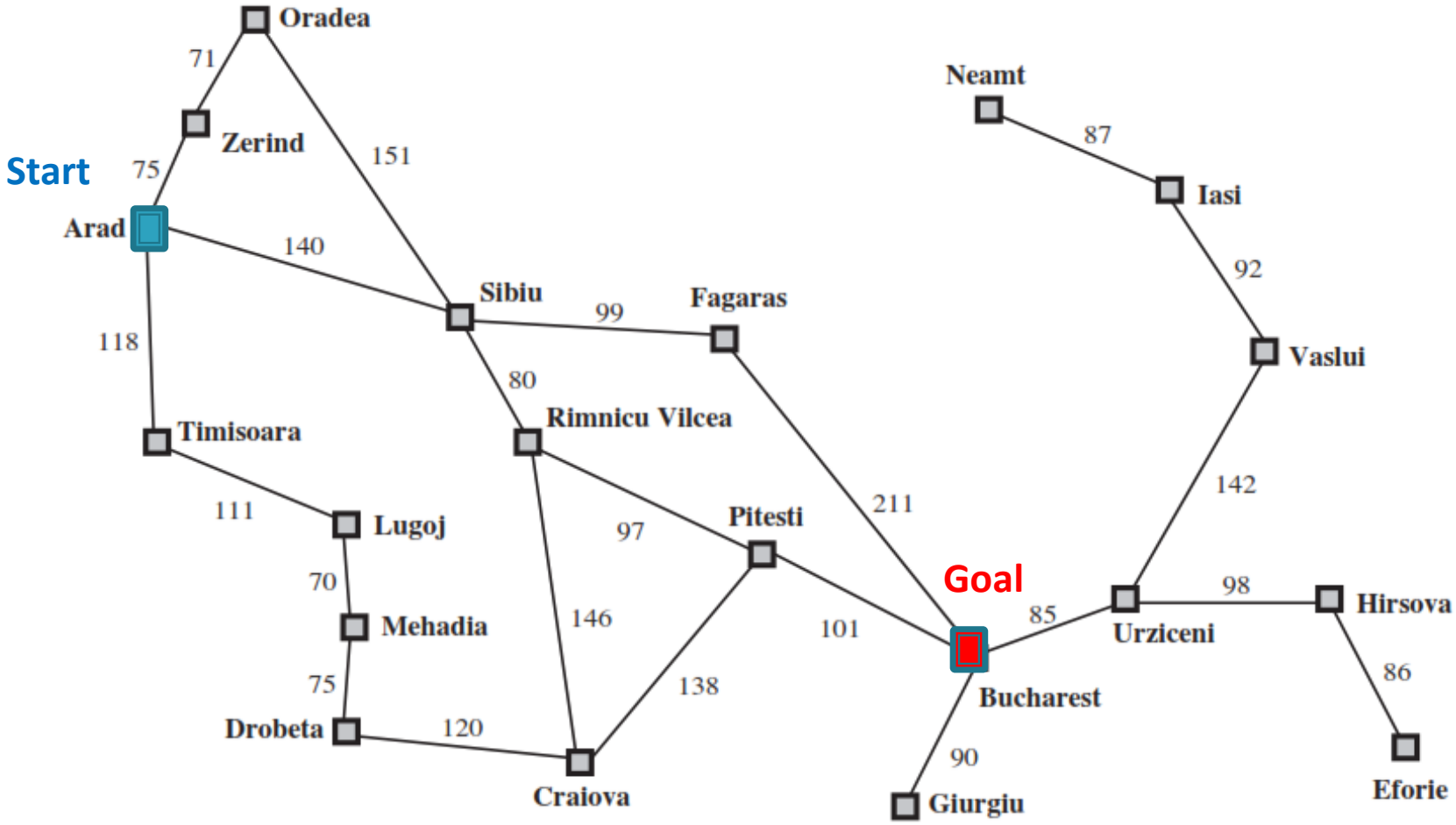
where d is the solution depth.

Memory-bounded Heuristic Search

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 f_limit variable to keep track of the 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 f_value of each node along the path with the best f_value of its children.

RBFS Example



A simplified road map of part of Romania.

RBFS Example

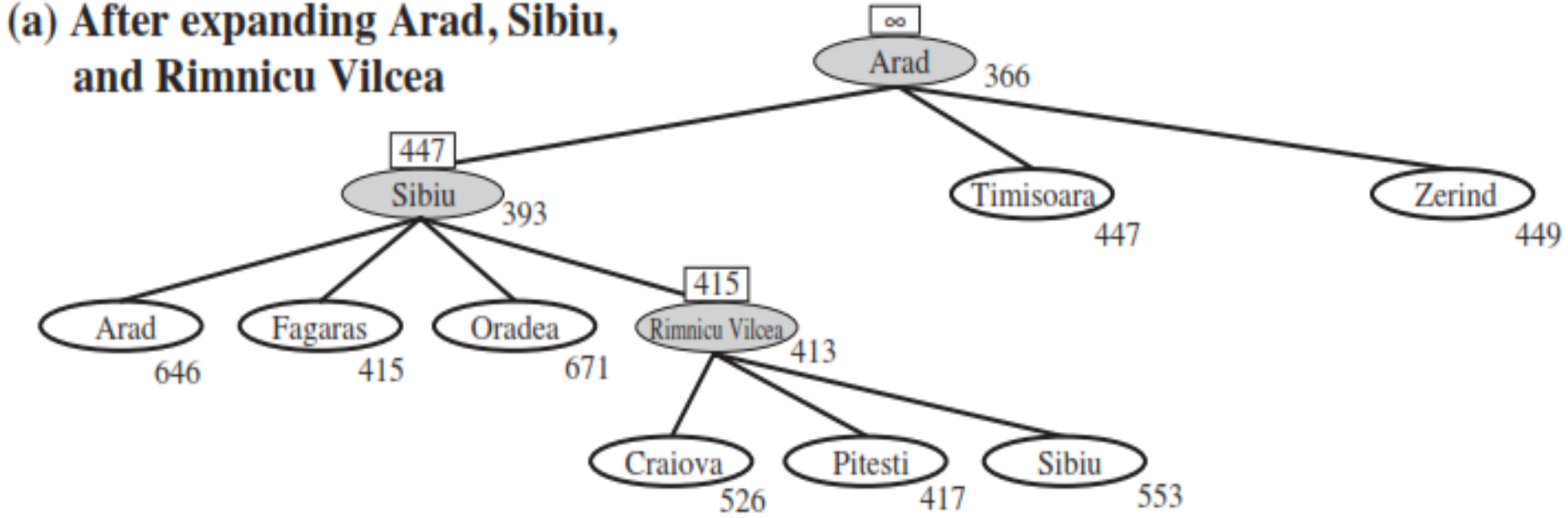
Values of h_{SLD}—straight-line distances to Bucharest

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

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

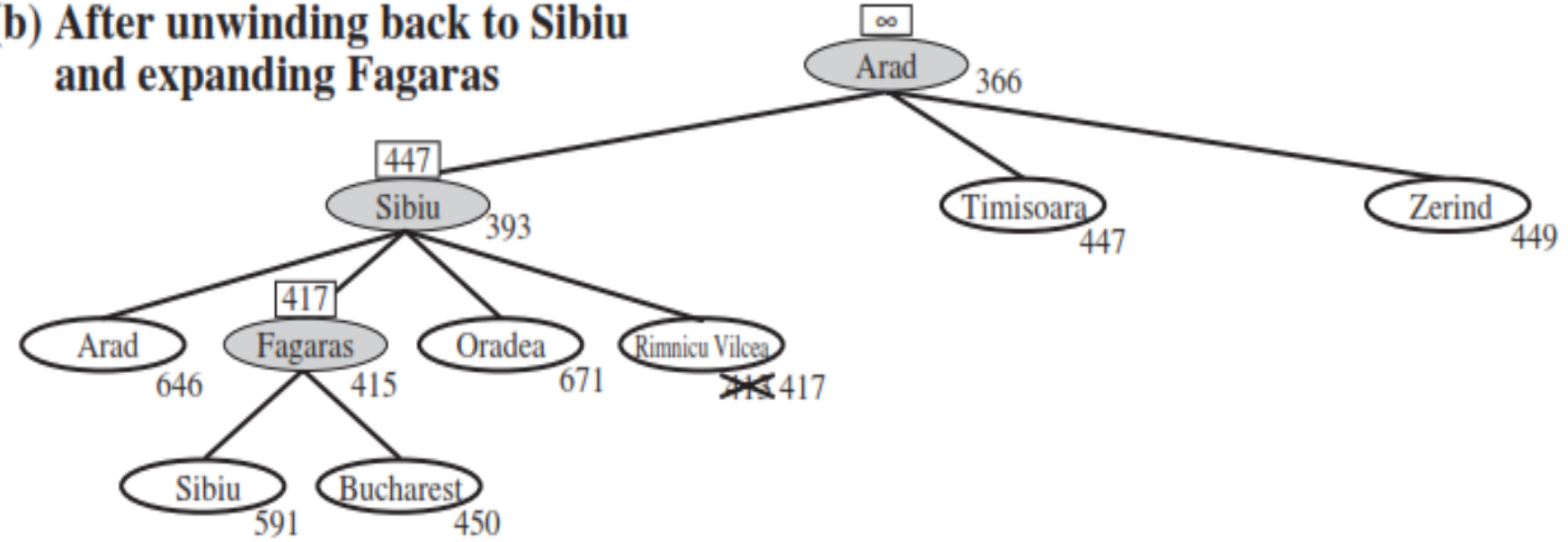
The f_limit value for each recursive call is shown on top of each current node, and every node is labeled with its f_cost.



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

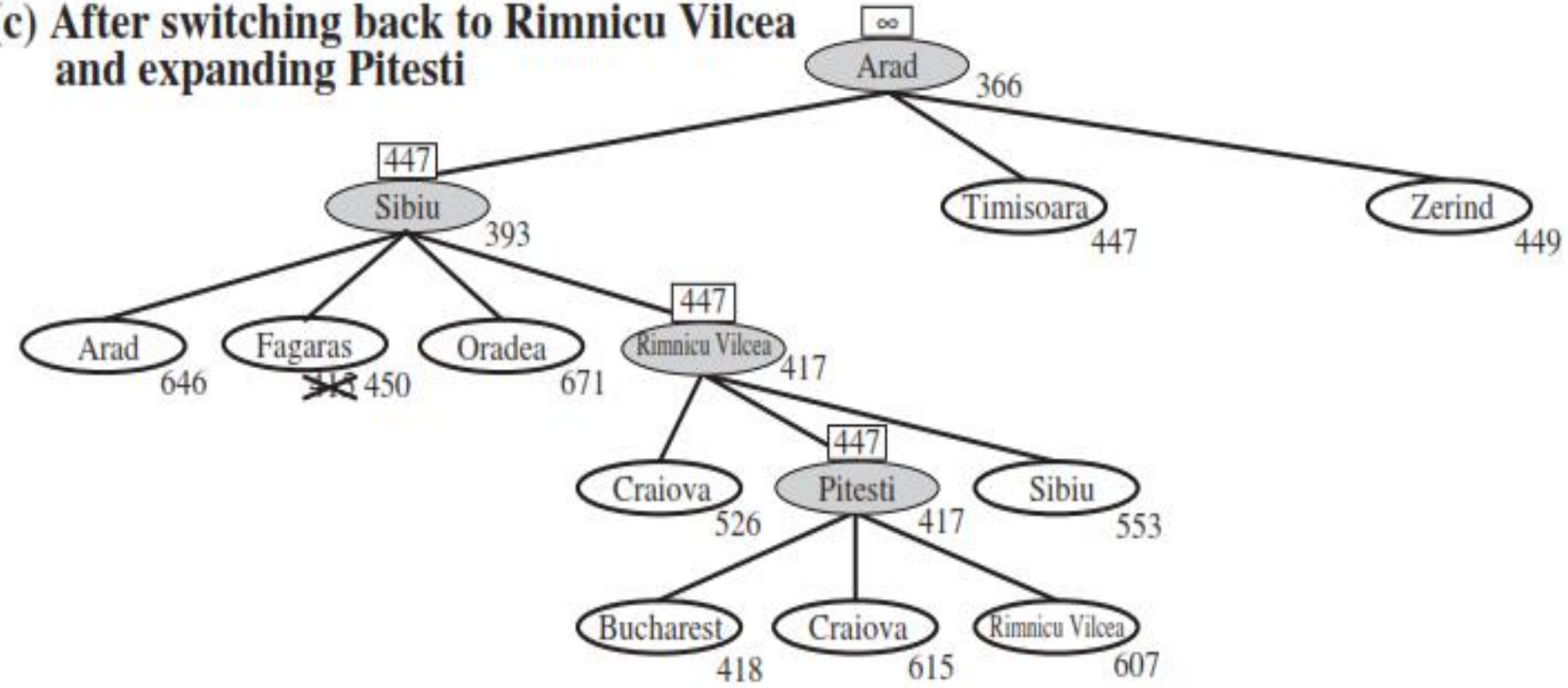
(b) After unwinding back to Sibiu and expanding Fagaras



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.

Reading Material

- **Artificial Intelligence,** A Modern Approach

  Stuart J. Russell and Peter Norvig

  - **Chapter 3.**