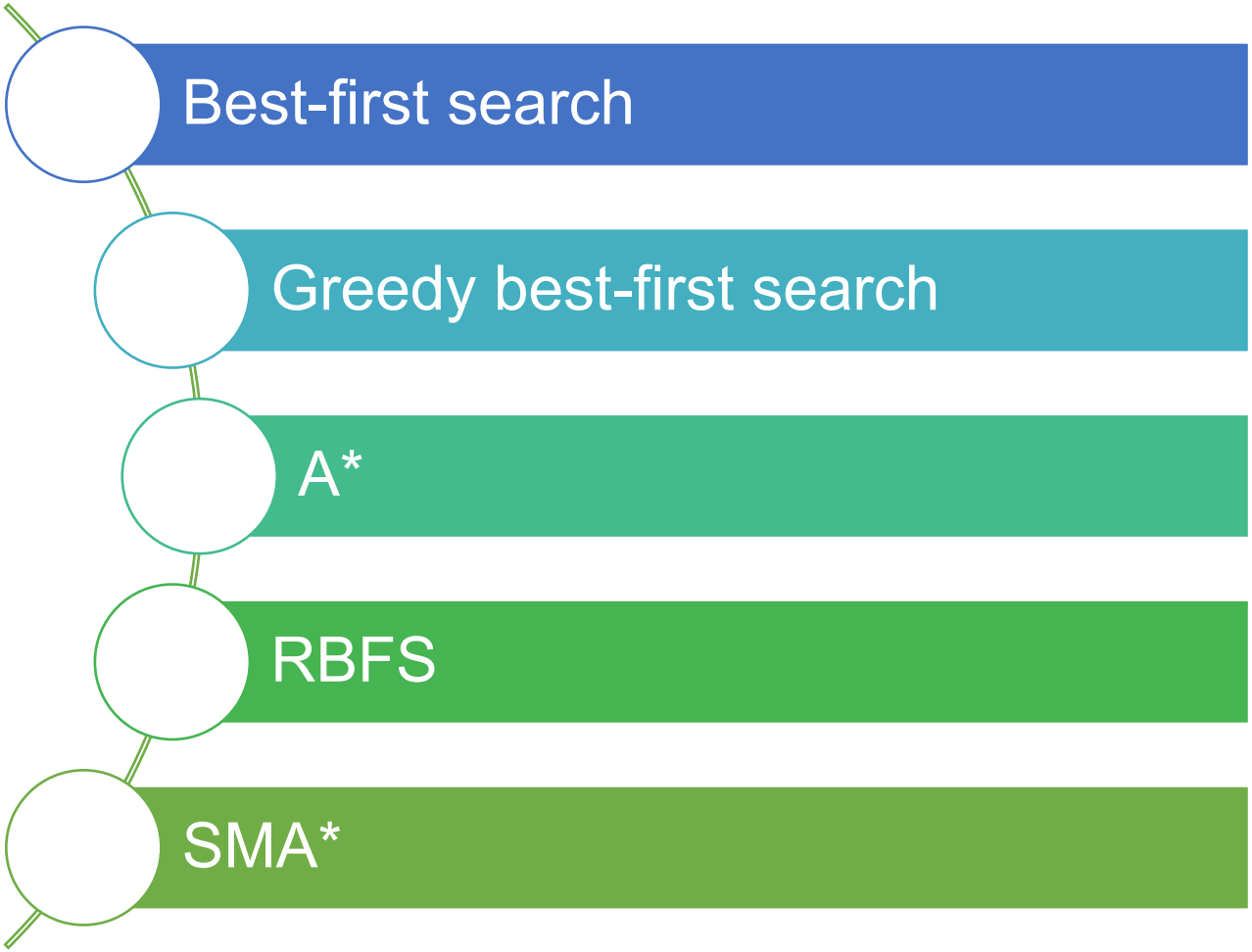
Informed search strategies



Best-first search

- An instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm
- A node is selected for expansion based on an **evaluation function**, $\boldsymbol{f(n)}$.
  - Node with the lowest $f(n)$ is expanded first
- The choice of f determines the search strategy.

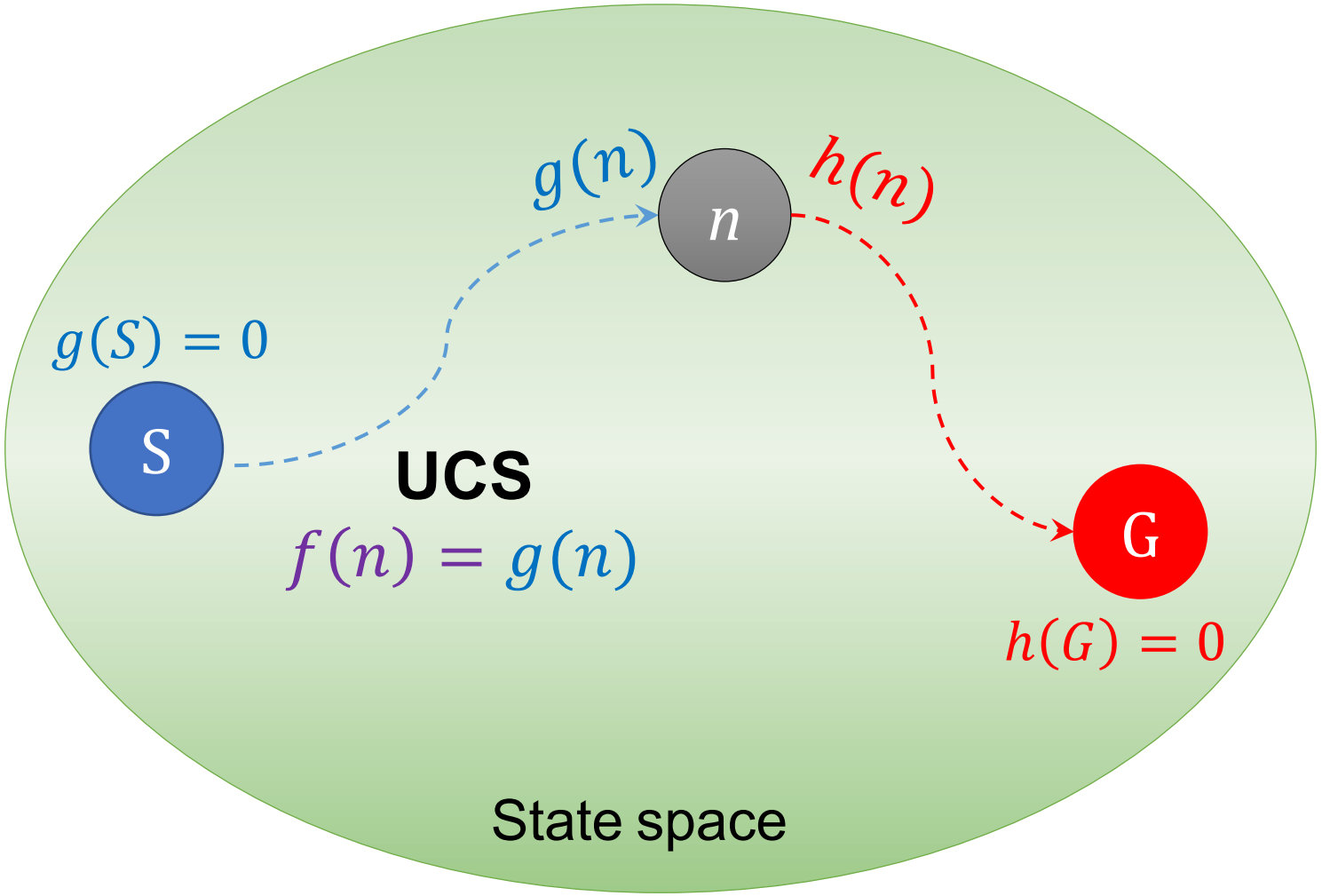
Heuristic function

- Most best-first algorithms include a **heuristic function $h(n)$** as a component of f.

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

- Unlike $g(n)$, $h(n)$ depends only on the state at that node
- Assumption of $h(n)$
  - Arbitrary, nonnegative, problem-specific functions
  - Constraint: if n is a goal node, then $h(n) = 0$

Cost function vs. Heuristic function

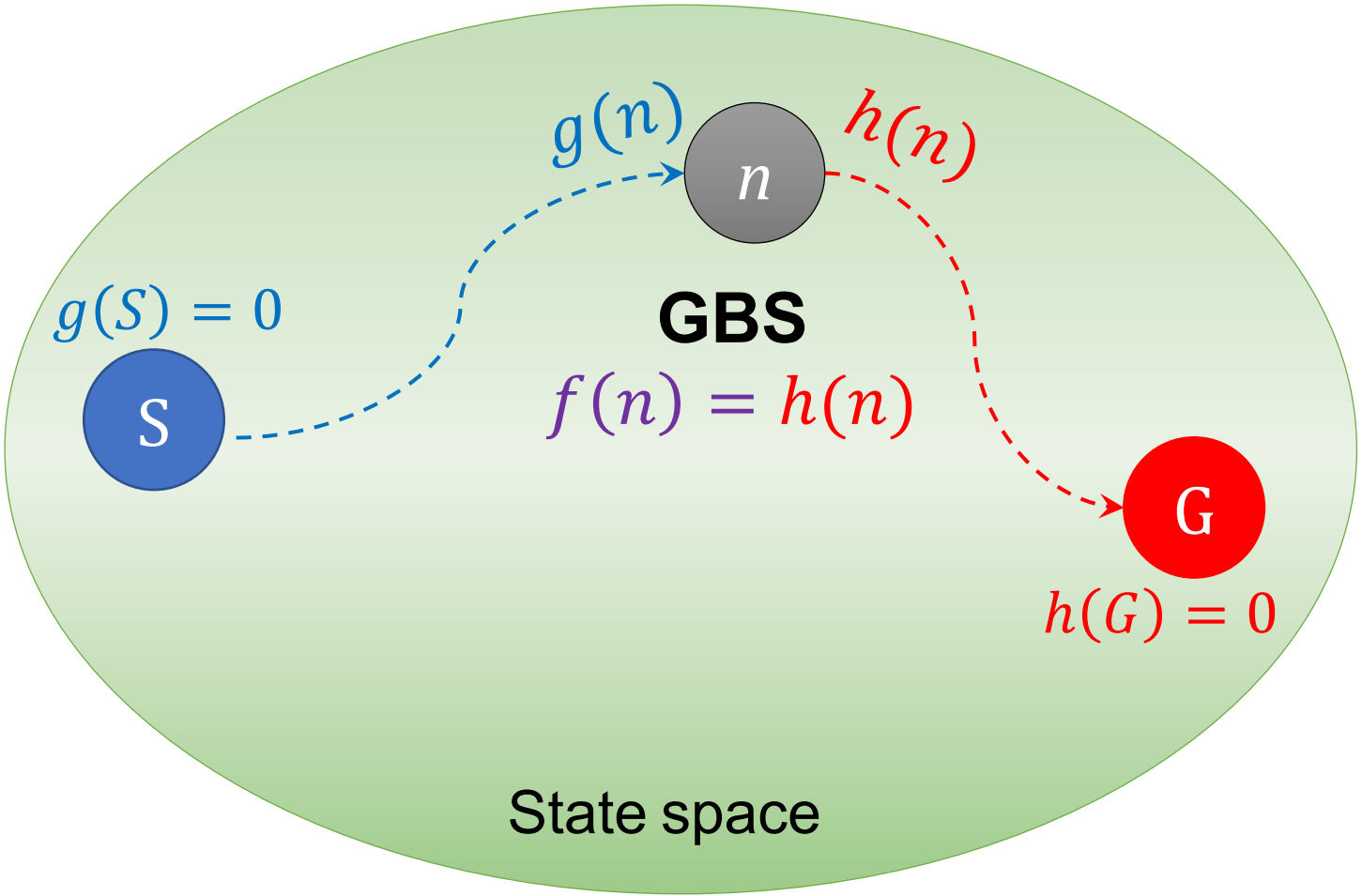


Greedy Best-First Search

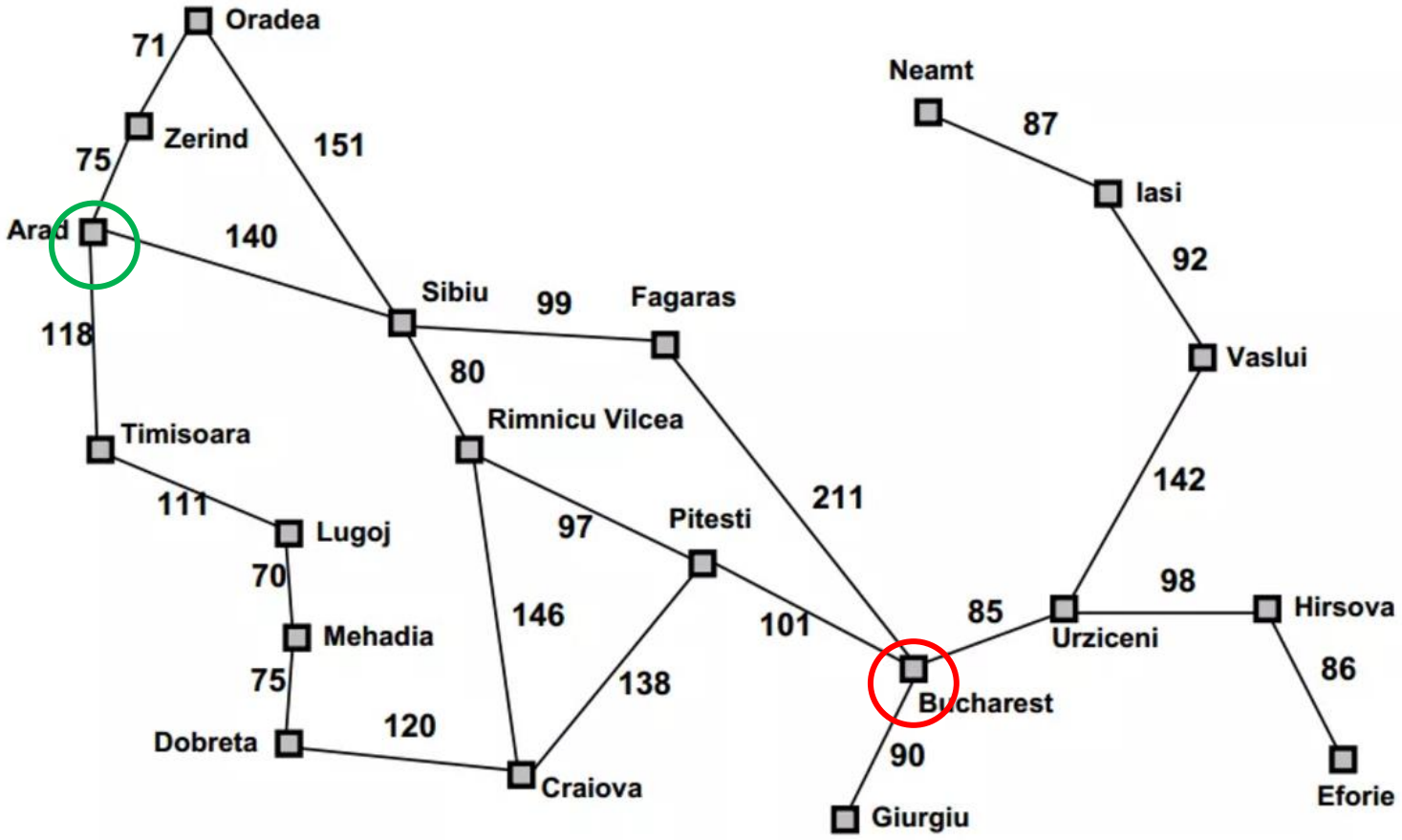
Greedy best-first search

- Expand the node that appears to be closest to goal using

$$f(n) = h(n)$$



Straight-line distance heuristic h_{SLD}



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search: An example

(a) The initial state

366

h_{SLD}(Arad)

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

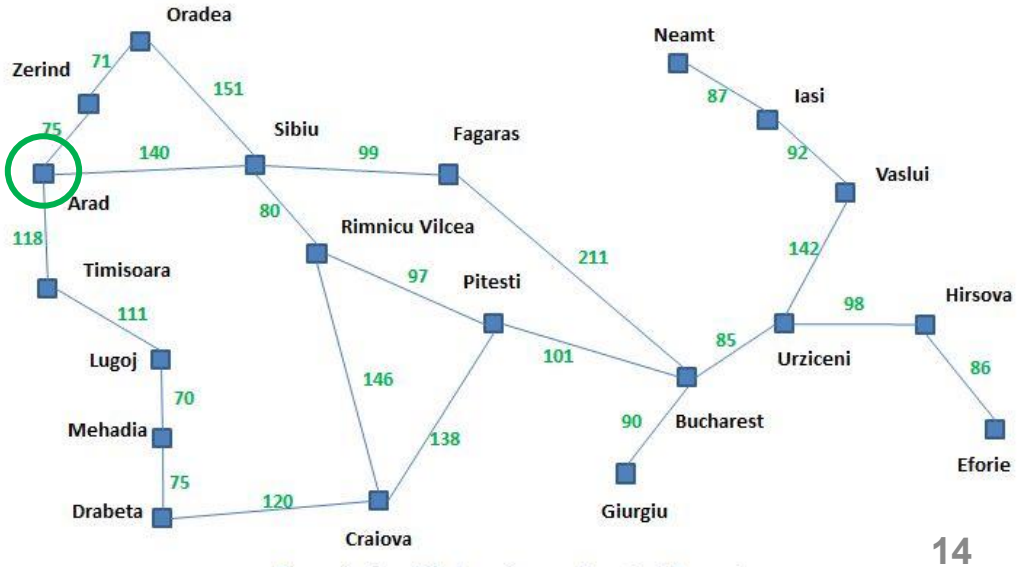
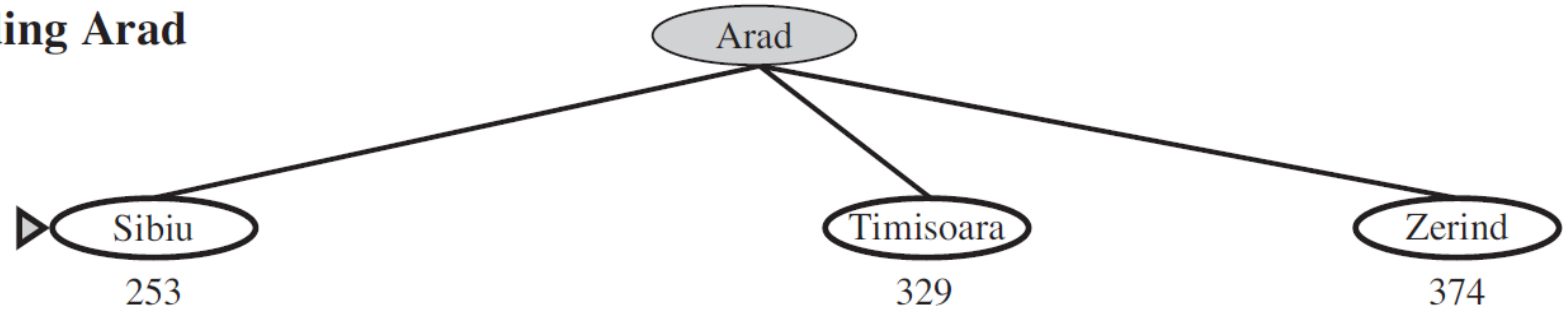


Figure A simplified road map of part of Romania.

Greedy best-first search: An example

(b) After expanding Arad



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

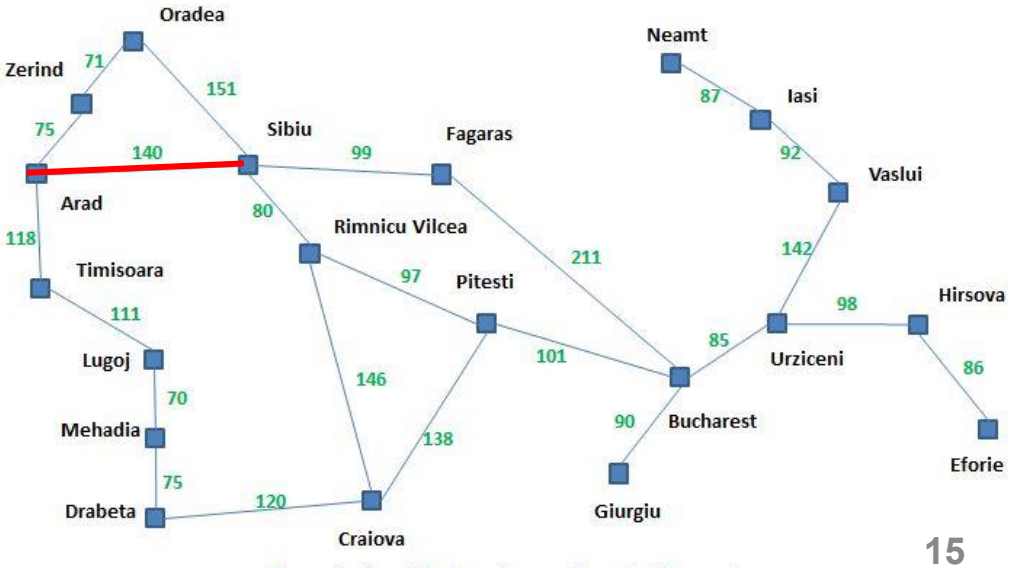
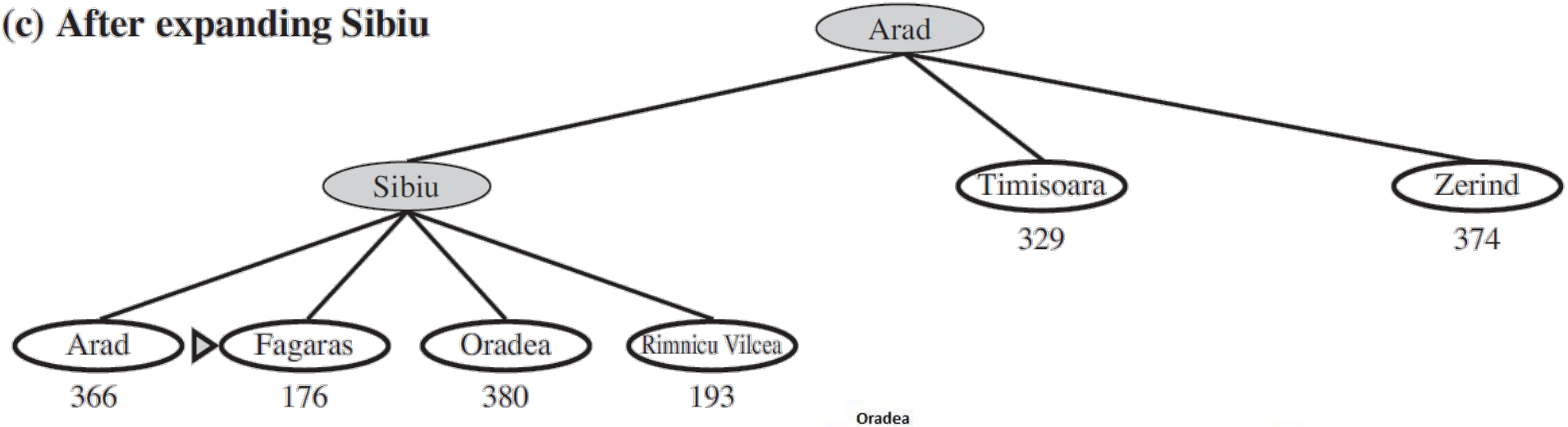


Figure A simplified road map of part of Romania.

Greedy best-first search: An example

(c) After expanding Sibiu



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

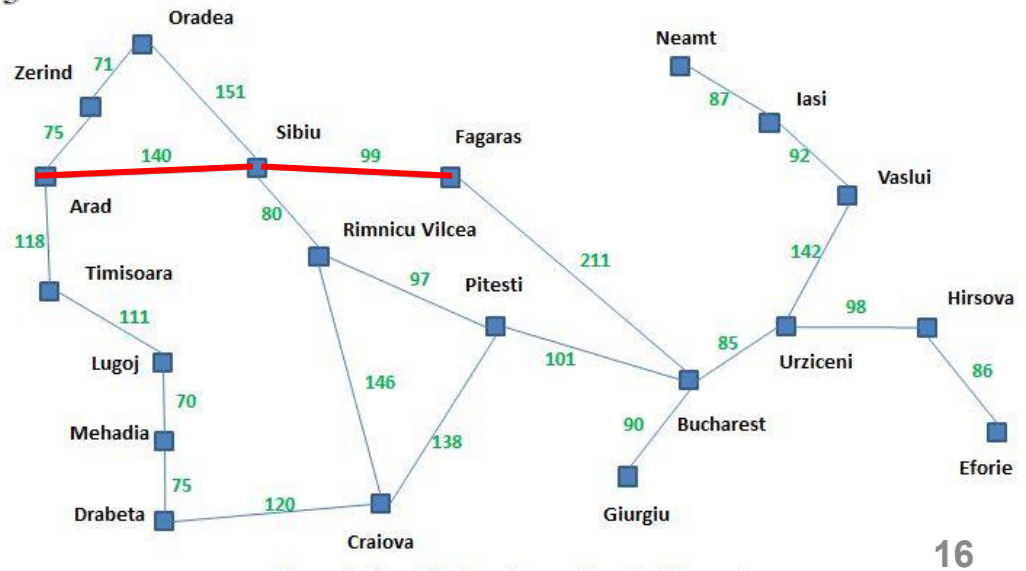
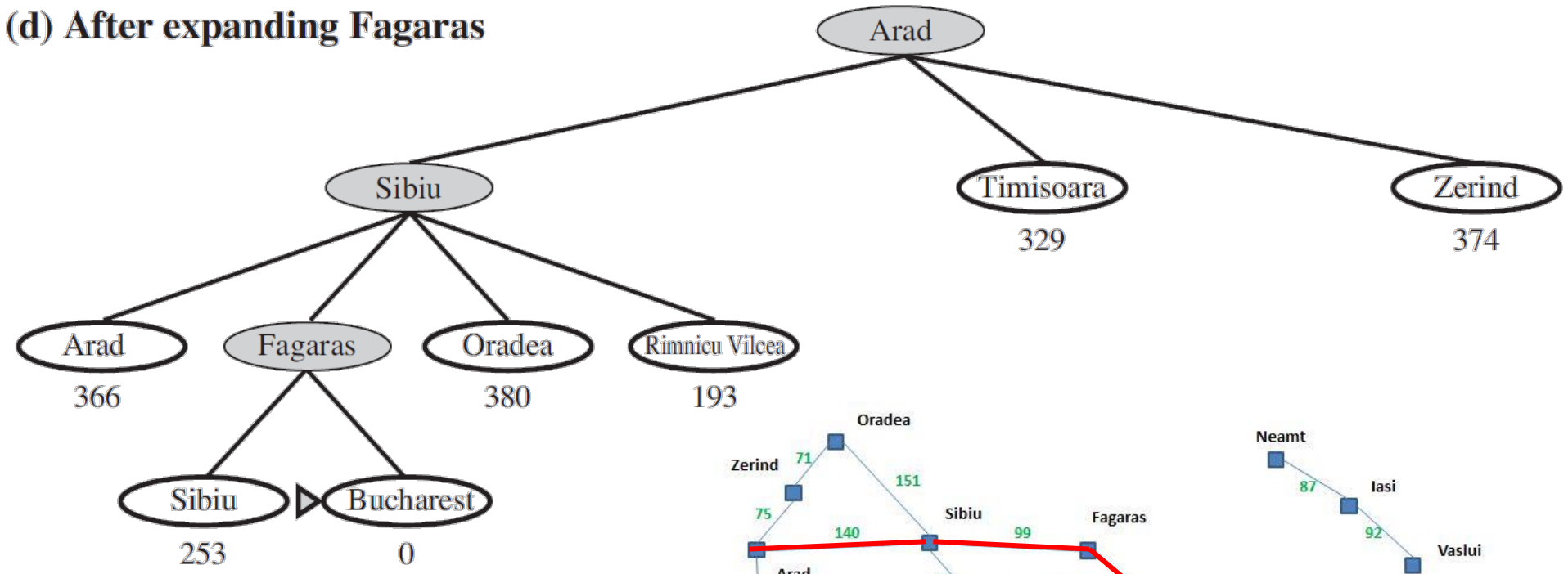


Figure A simplified road map of part of Romania.

Greedy best-first search: An 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

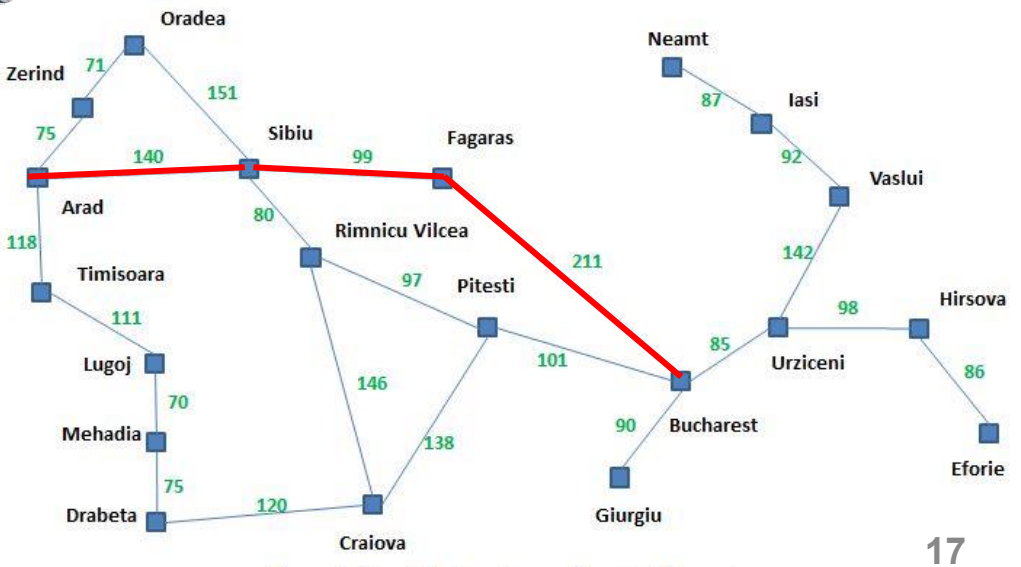


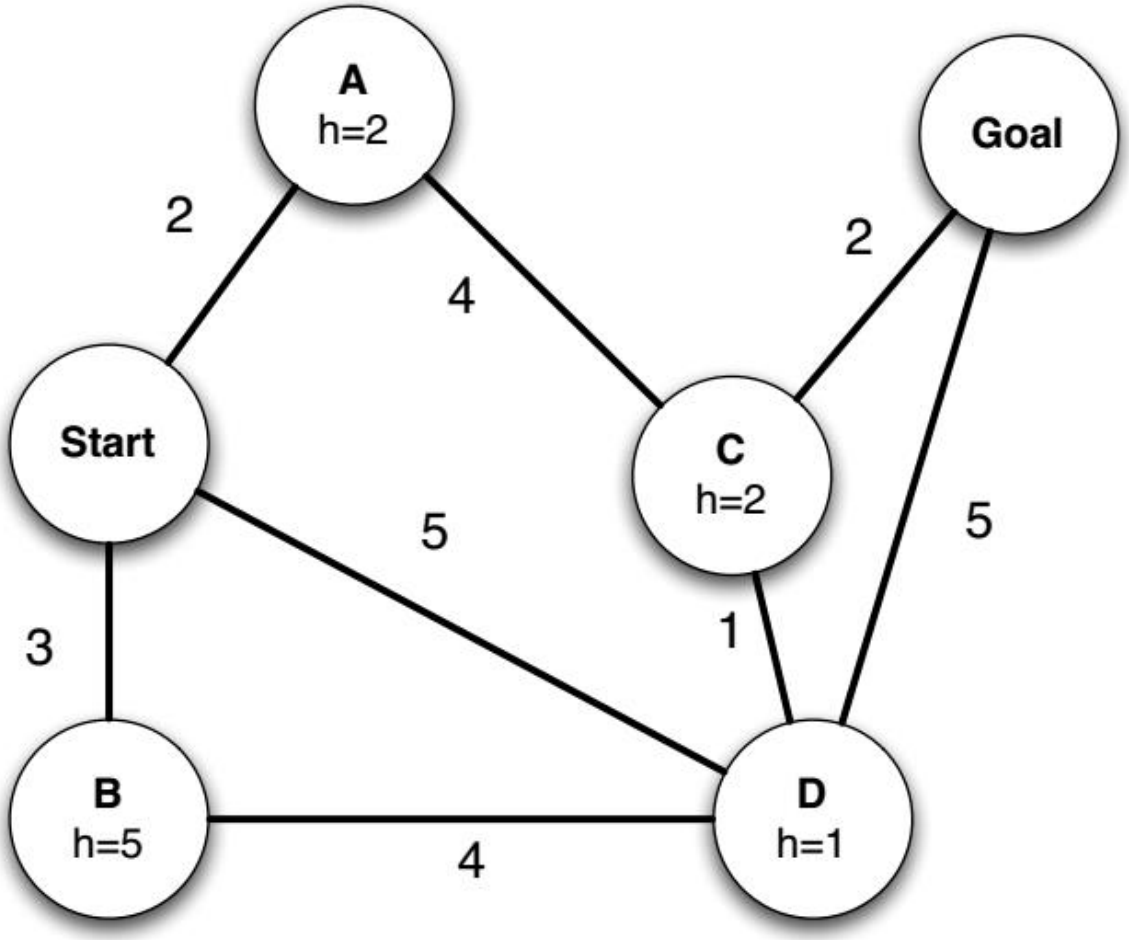
Figure A simplified road map of part of Romania.

Evaluation of Greedy best-first search

- Completeness
  - NO – may get stuck forever
  - E.g., Iasi $\rightarrow$ Neamt $\rightarrow$ Iasi $\rightarrow$ Neamt $\rightarrow$ …
- Time complexity
  - $O(b^m) \rightarrow$ reduced substantially with a good heuristic
- Space complexity
  - $O(b^m)$ – keeps all nodes in memory
- Optimality
  - NO

Quiz 01: Greedy best-first search

- Work out the order in which states are expanded, as well as the path returned by graph search. Assume ties resolve in such a way that states with earlier alphabetical order are expanded first.



A* Search

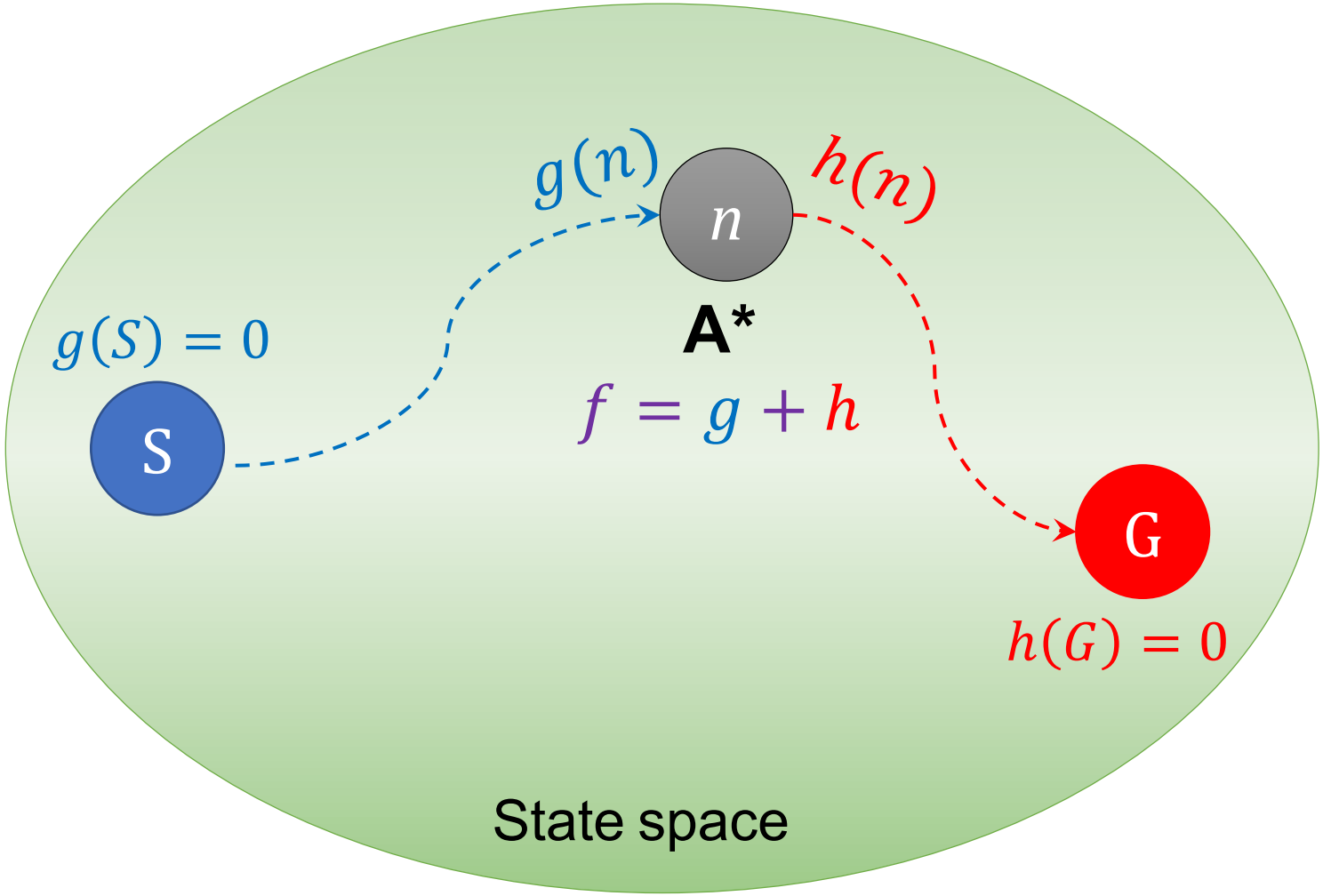
A* search

- The most widely known form of best-first search
- Ideas
  - Use heuristic to guide search, but not only
  - Avoid expanding paths that are already expensive
  - Ensure to compute a path with minimum cost

- Evaluate nodes by $\boldsymbol{f(n) = g(n) + h(n)}$
  - where $g(n)$ is the cost to reach the node n and $h(n)$ is the cost to get from n to the goal
  - $f(n)$ = estimated cost of the cheapest solution through n

.

A* search



(a) The initial state

366=0+366

$$f = g + n$$

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

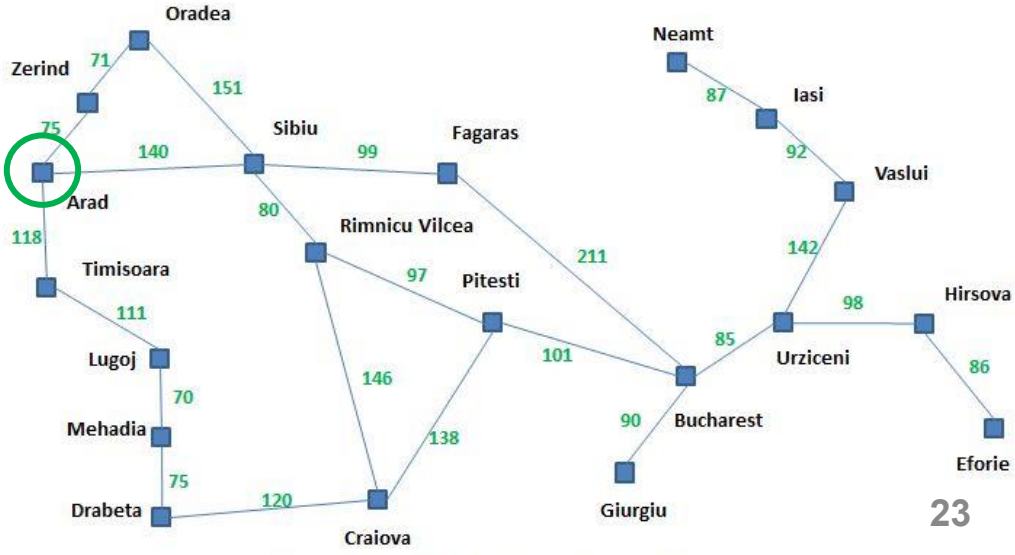
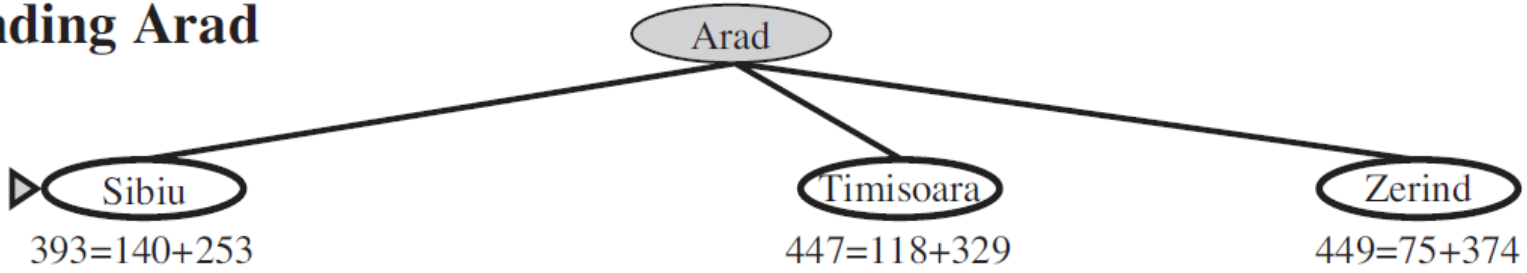


Figure A simplified road map of part of Romania.

(b) After expanding Arad



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

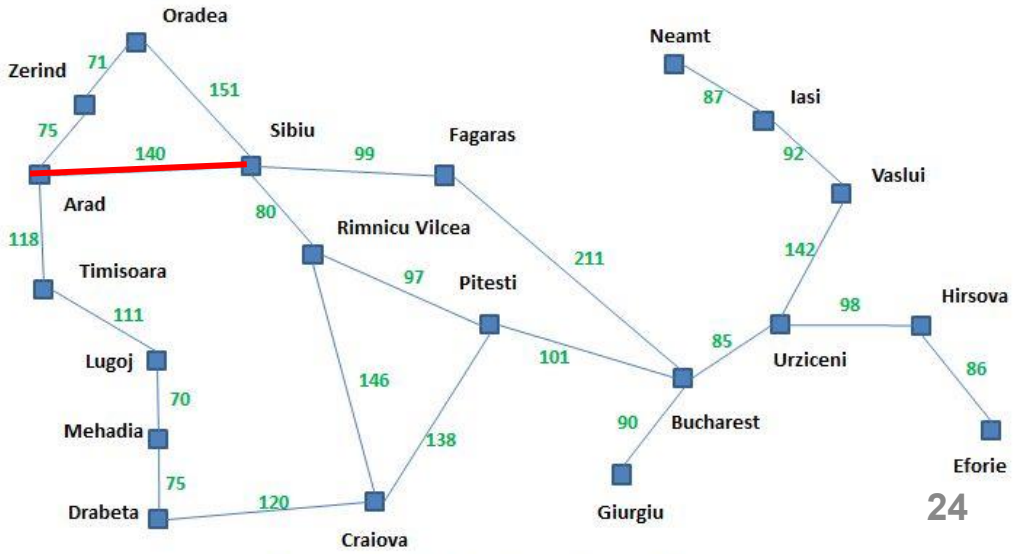
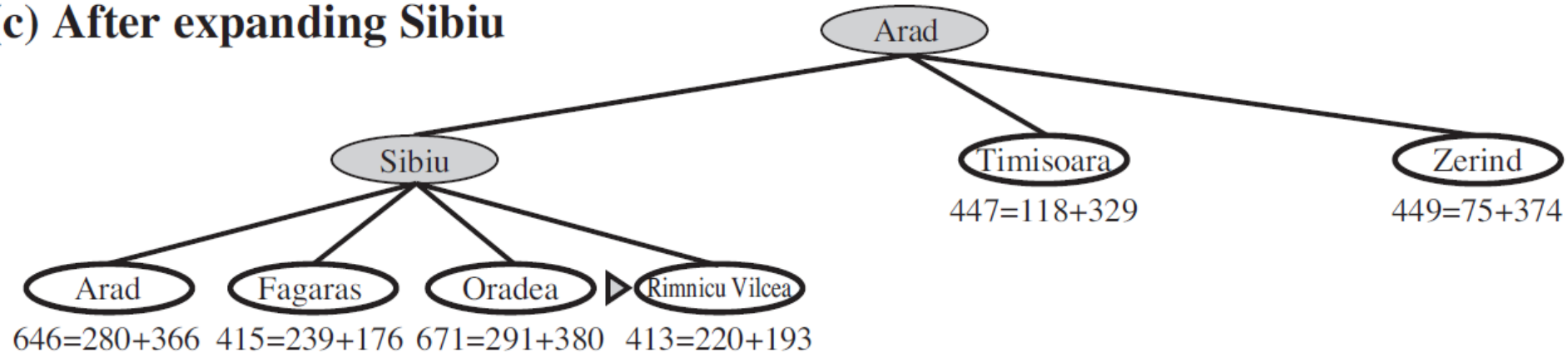


Figure A simplified road map of part of Romania.

(c) After expanding Sibiu



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

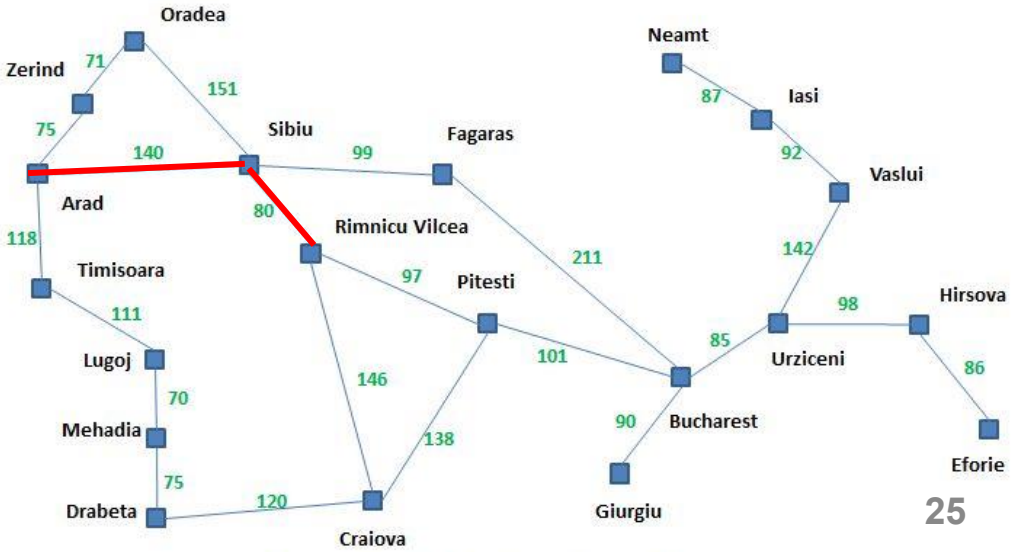
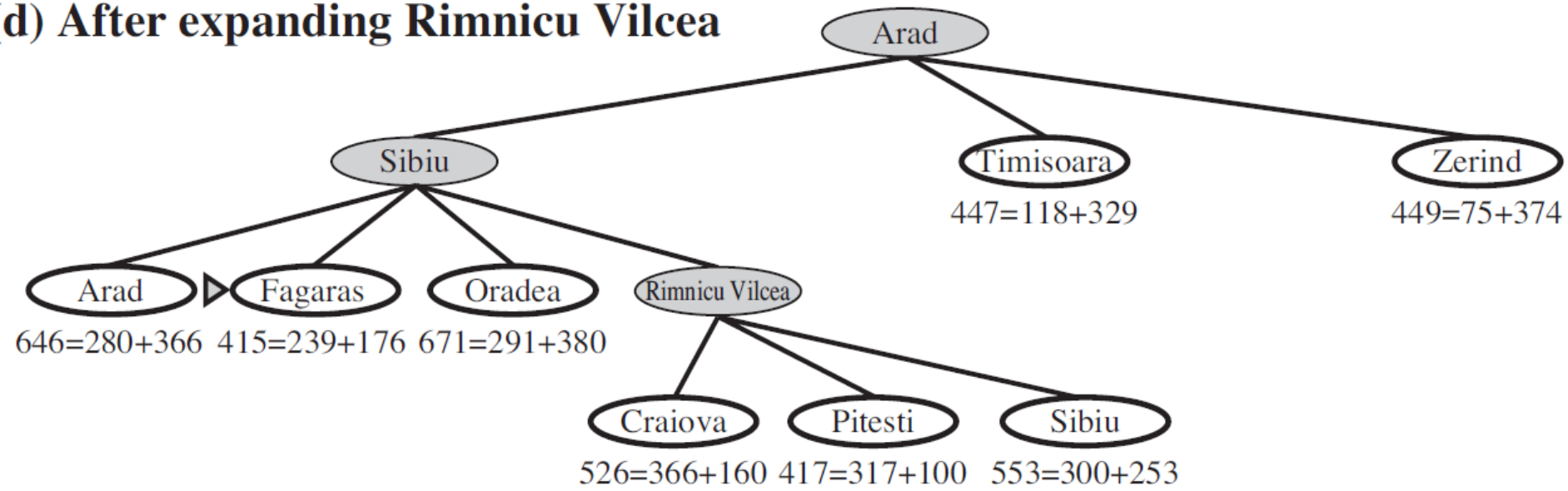


Figure A simplified road map of part of Romania.

(d) After expanding Rimnicu Vilcea



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

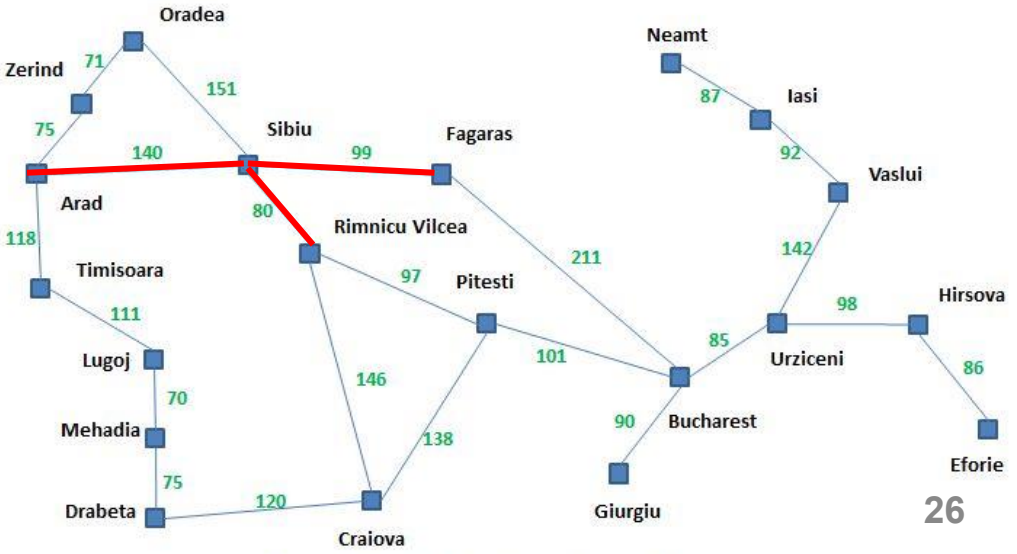
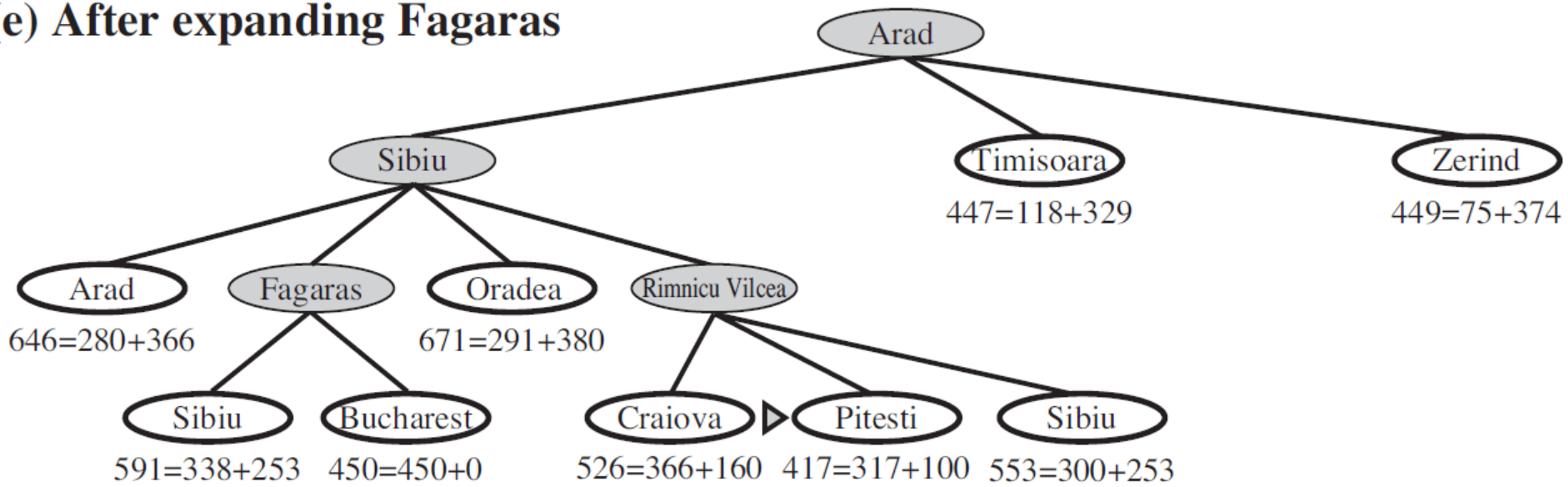


Figure A simplified road map of part of Romania.

(e) After expanding Fagaras



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

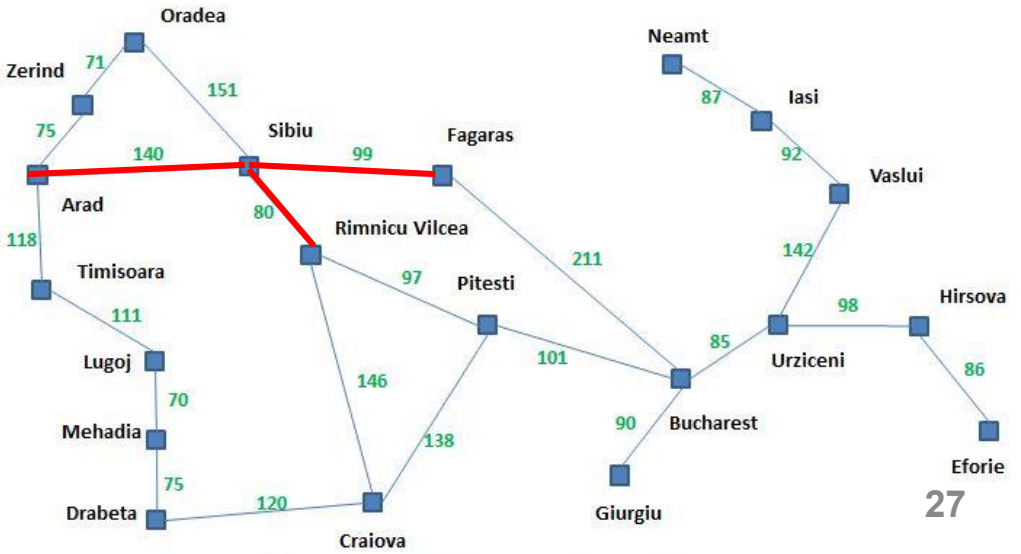
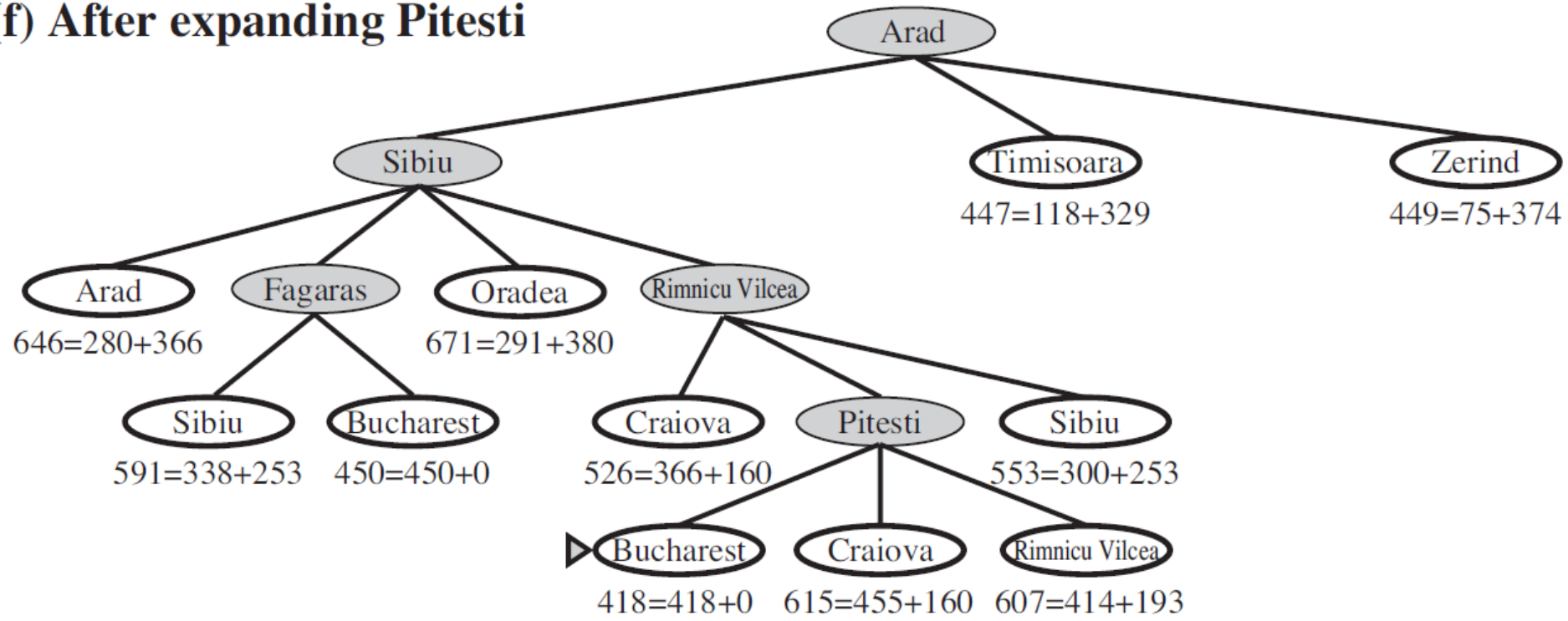


Figure A simplified road map of part of Romania.

(f) After expanding Pitesti



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

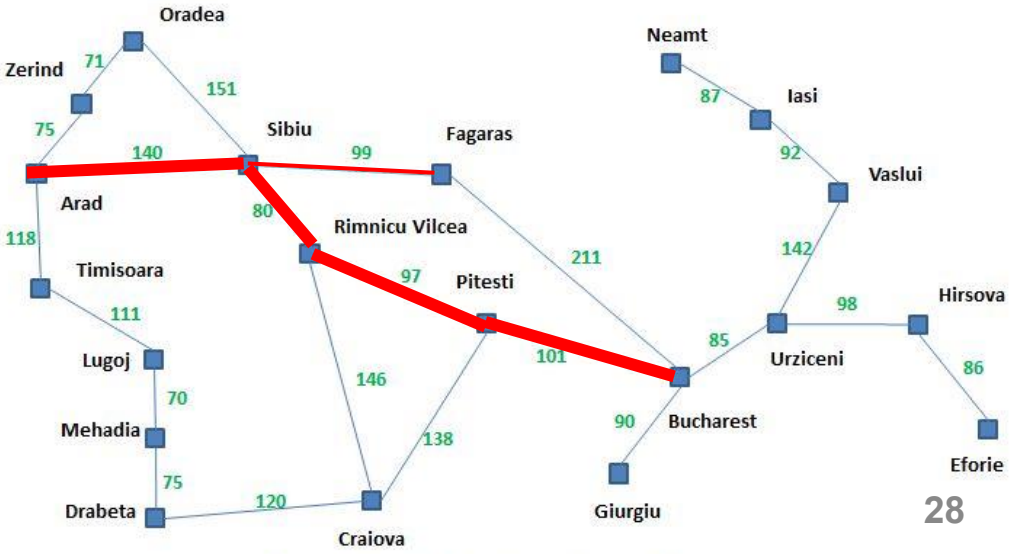
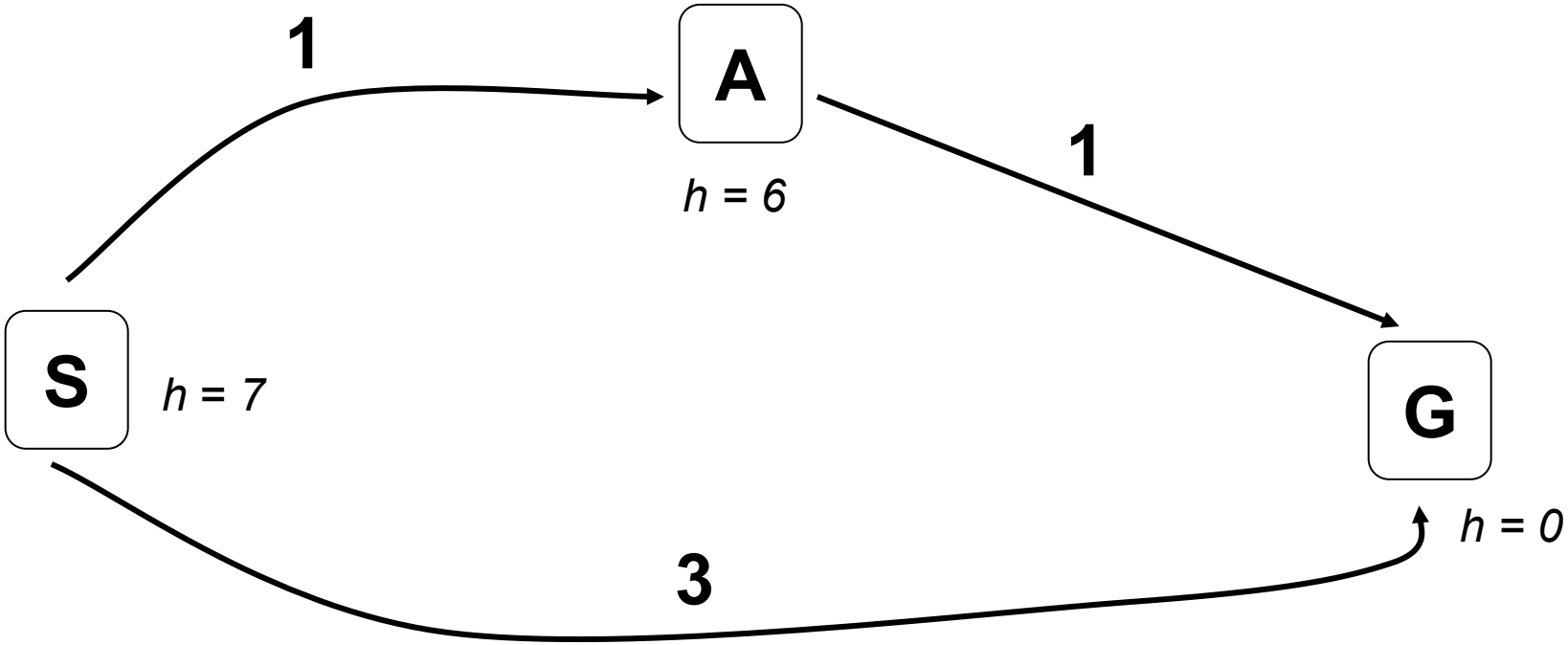


Figure A simplified road map of part of Romania.

Evaluation of A* search

- Completeness
  - YES if all step costs exceed some finite ϵ and if b is finite
  - (review the condition for completeness of UCS)
- Optimality
  - YES – with conditions on heuristic being used
- Time complexity
  - Exponential
- Space complexity
  - Exponential (keep all nodes in memory)

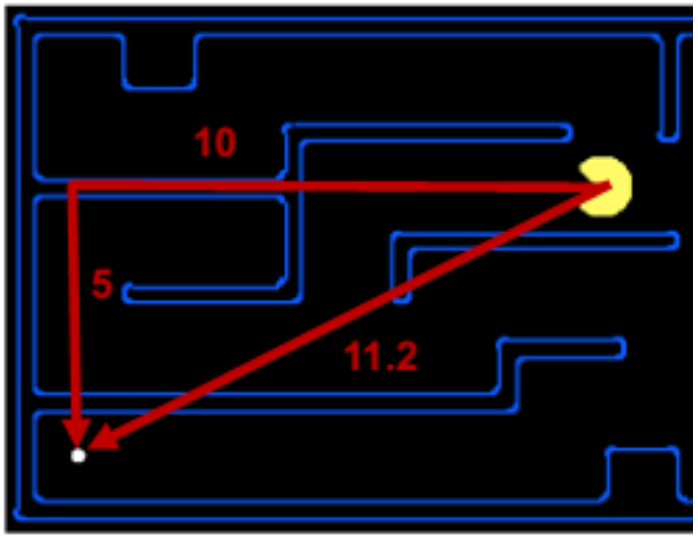
A* is not always optimal...



In what conditions, A* is optimal?

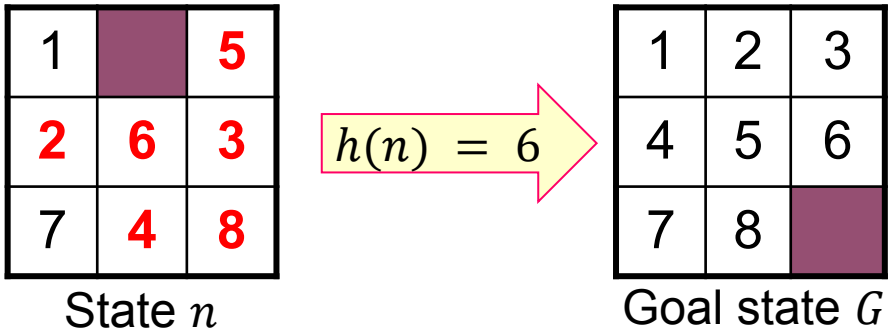
Conditions for optimality: Admissibility

- $h(n)$ must be an admissible heuristic
  - Never overestimate the cost to reach the goal $\rightarrow$ optimistic
  - E.g., the straight-line distance h_{SLD}

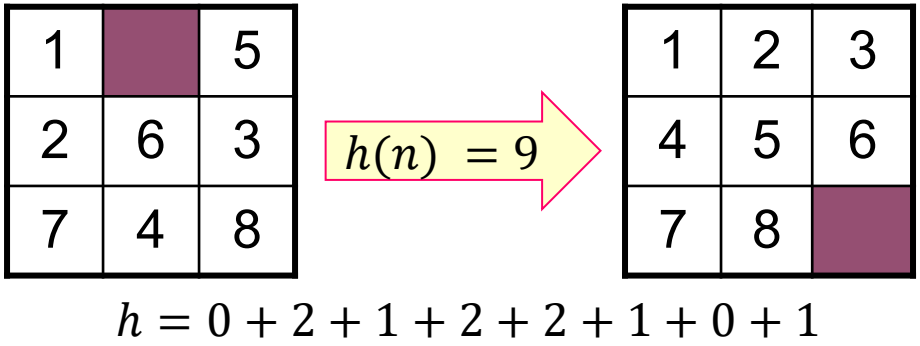


Admissible heuristics for 8-puzzle

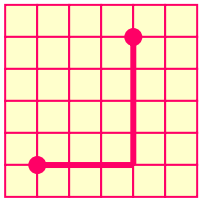
- $h(n)$ = number of misplaced tiles



- $h(n)$ = sum of the (Manhattan) distance of every numbered tile to its goal position



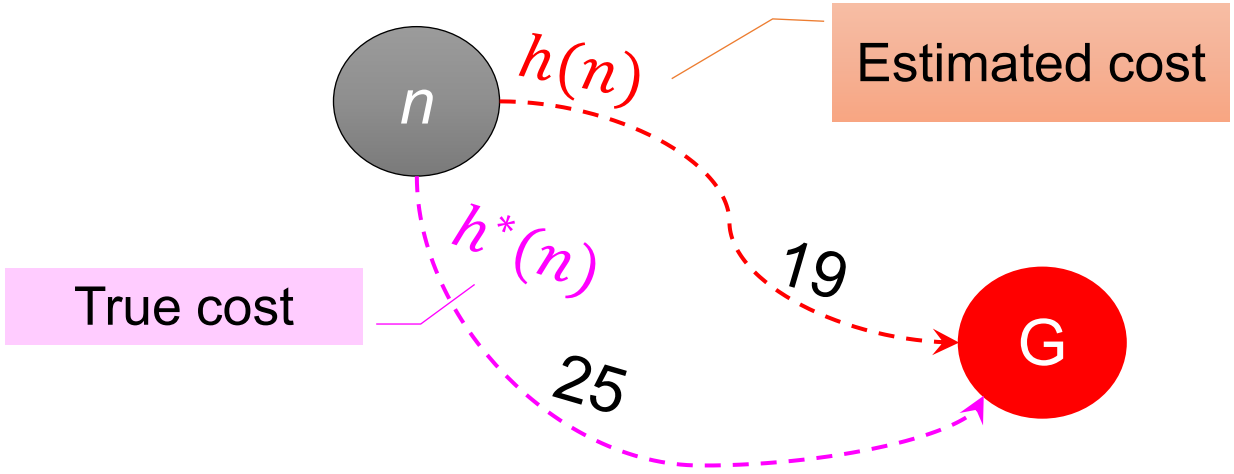
$h = 0 + 2 + 1 + 2 + 2 + 1 + 0 + 1$



$d = dx + dy$

Conditions for optimality: Admissibility

- $h(n)$ is admissible if for every node n, $\boldsymbol{h(n) \leq h^*(n)}$
  - where $h^*(n)$ is the true cost to reach the goal state from n



- Hence, $f(n)$ never overestimates the true cost of a solution along the current path through n.
  - $g(n)$ is the actual cost to reach n along the current path

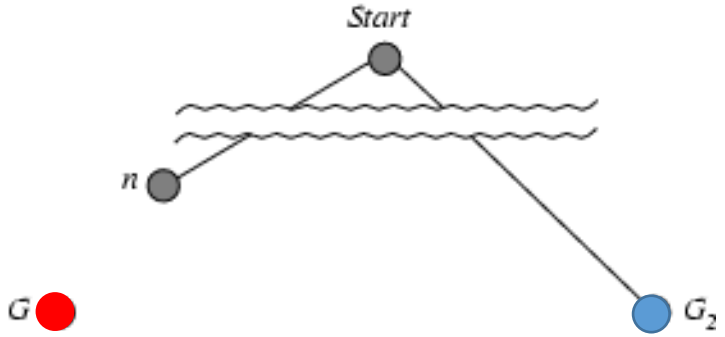
Conditions for optimality: Admissibility

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

- Suppose some suboptimal goal G_2 has been generated and is in the frontier.
- Let n be an unexpanded node in the frontier such that n is on a shortest path to an optimal goal G.
- $f(G_2) = g(G_2)$ since $h(G_2) = 0$
- $g(G_2) > g(G)$ since G_2 is suboptimal
- $f(G) = g(G)$ since $h(G) = 0$

  $f(G_2) > f(G)$ (1)
- $h(n) \leq h^*(n)$ since *h* is admissible
- $g(n) + h(n) \leq g(n) + h^*(n)$

  $f(n) \leq f(G)$ (2)



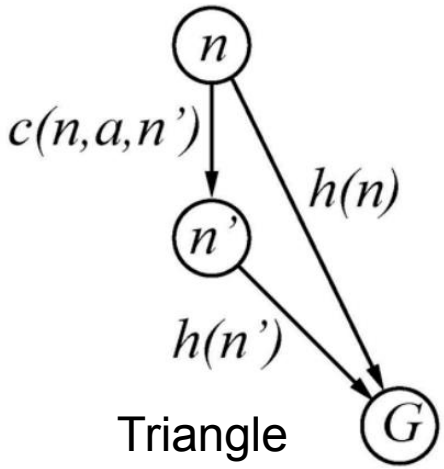
- From (1), (2): $f(G_2) > f(n)$ → A* will never select G_2 for expansion

Conditions for optimality: Consistency

- Admissibility is insufficient for graph search.
  - The optimal path to a repeated state could be discard if it is not the first one selected.
- $h(n)$ is consistent if for every node n, every successor n' of n generated by any action a,

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

- Every consistent heuristic is also admissible.



Triangle inequality

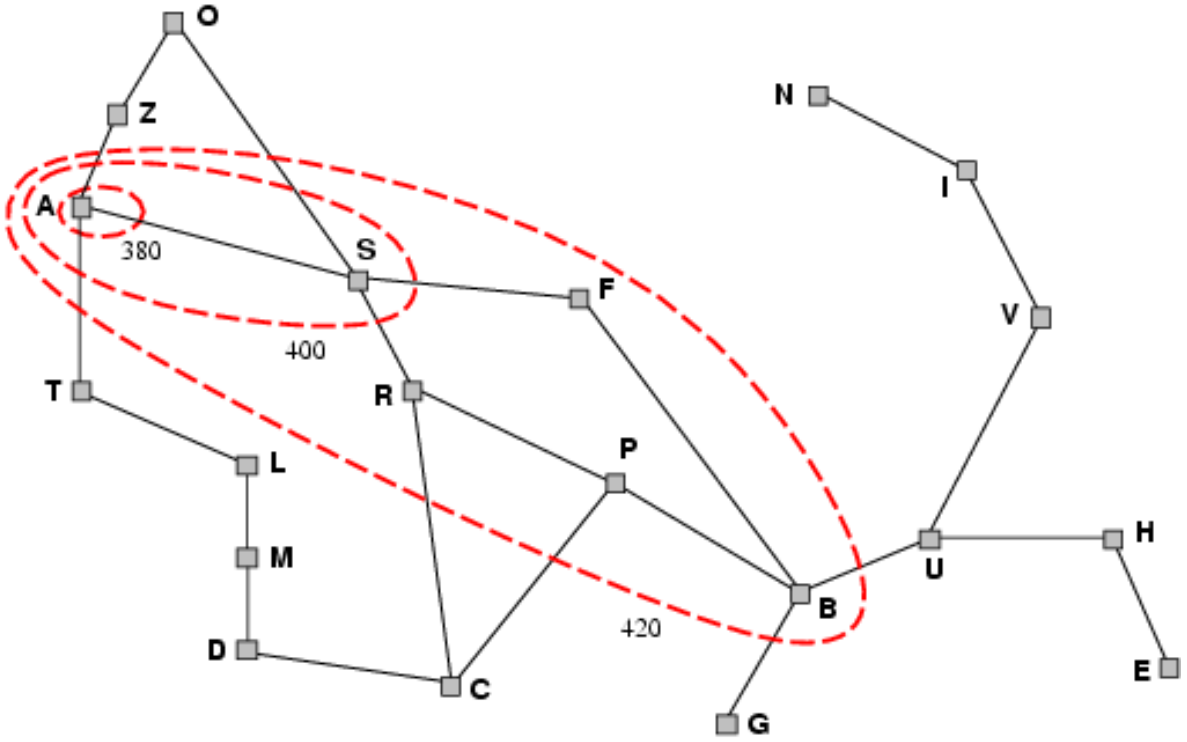
Conditions for optimality: Consistency

If $h(n)$ is consistent, A* using **GRAPH-SEARCH** is optimal

- If $h(n)$ is consistent, the values of $f(n)$ along any path are non-decreasing.
  - Suppose n' is a successor of $n \rightarrow g(n') = g(n) + c(n, a, n')$
  - $f(n') = g(n') + h(n') = g(n) + c(n, a, n') + h(n') \geq g(n) + h(n) = f(n)$
- Whenever A* selects a node n for expansion, the optimal path to that node has been found.
  - Proof by contradiction: There would have to be another frontier node n' on the optimal path from the start node to n (by the graph separation property)
  - f is nondecreasing along any path $\rightarrow f(n') < f(n) \rightarrow n'$ would have been selected first

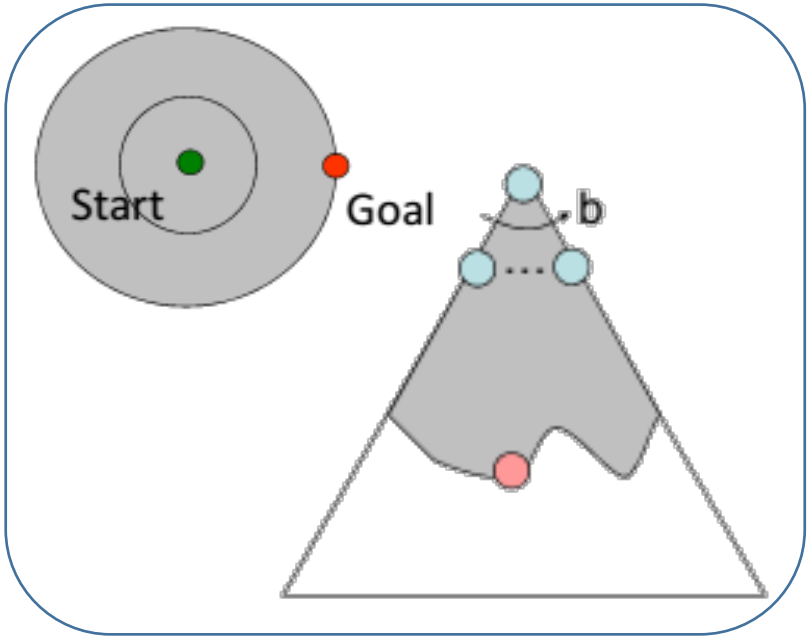
Contours of A* search

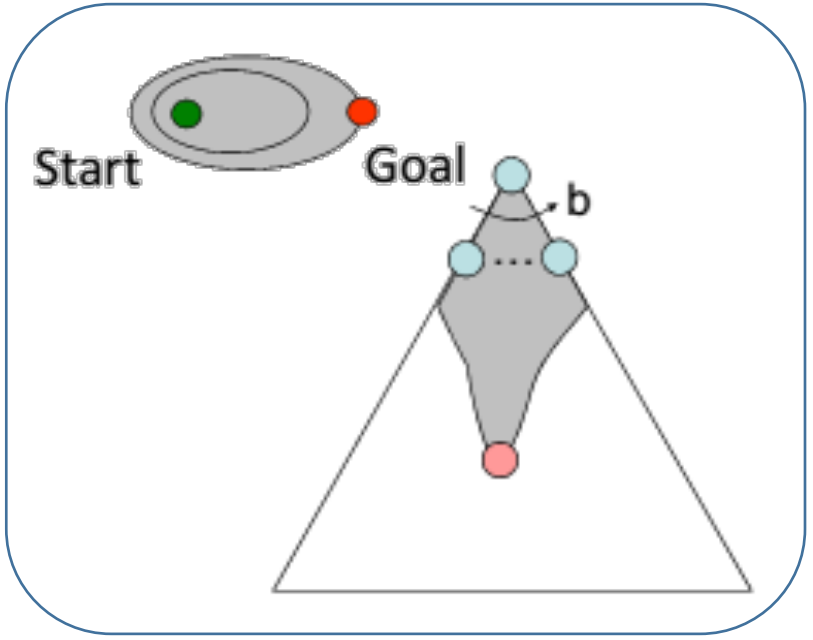
- A* expands nodes in order of increasing f-value
- Gradually adds "f-contours" of nodes such that contour i has all nodes with $f = f_i$ where $f_i < f_{i+1}$
- A* will expand all nodes with costs $f(n) < C^*$



A* contours vs. UCS contours

- The bands of UCS will be “circular” around the start state.





- The bands of A*, with more accurate heuristics, will stretch toward the goal state and become more narrowly focused around the optimal path.

Comments on A*: The good

- Never expand nodes with $f(n) > C^*$
  - All nodes like these are **pruned** while stile guaranteeing optimality
- Optimally efficient for any given consistent heuristic
  - No other optimal algorithm is guaranteed to expand fewer nodes

Comments on A*: The bad

- A* expands all nodes with $f(n) < C^*$ (and possibly some nodes with $f(n) = C^*$) with before selecting a goal node.
  - This can still be exponentially large
  - A* usually runs out of space before it runs out of time
- Exponential growth will occur unless error in $h(n)$ grows no faster than log(true path cost)
  - In practice, error is usually proportional to true path cost (not log)
  - So exponential growth is common

$\rightarrow$ Not practical for many large-scale problems

Quiz 02: A*

- Work out the order in which states are expanded, as well as the path returned by graph search. Assume ties resolve in such a way that states with earlier alphabetical order are expanded first.