# Artificial Intelligence

## *CSC348 Unit 3: Problem Solving and Search*

## Part 2: Informed Search Techniques

***Syedur Rahman***

***Lecturer, CSE Department***

***North South University***

***syedur.rahman@wolfson.oxon.org***

# *Artificial Intelligence: Lecture Notes*

The lecture notes from the introductory lecture and this unit will be available shortly from the following URL:

- http://www.geocities.com/syedatnsu/

**Acknowledgements**

These lecture notes contain material from the following sources

- *Logical Programming and Artificial Intelligence* by S. Kapetanakis, 2004
- *Artificial Intelligence: A modern approach* by S. Russell and P. Norvig, International Edition, 2nd edition
- *Intelligent Systems* by S.Clark, 2005

# *Best-first search*

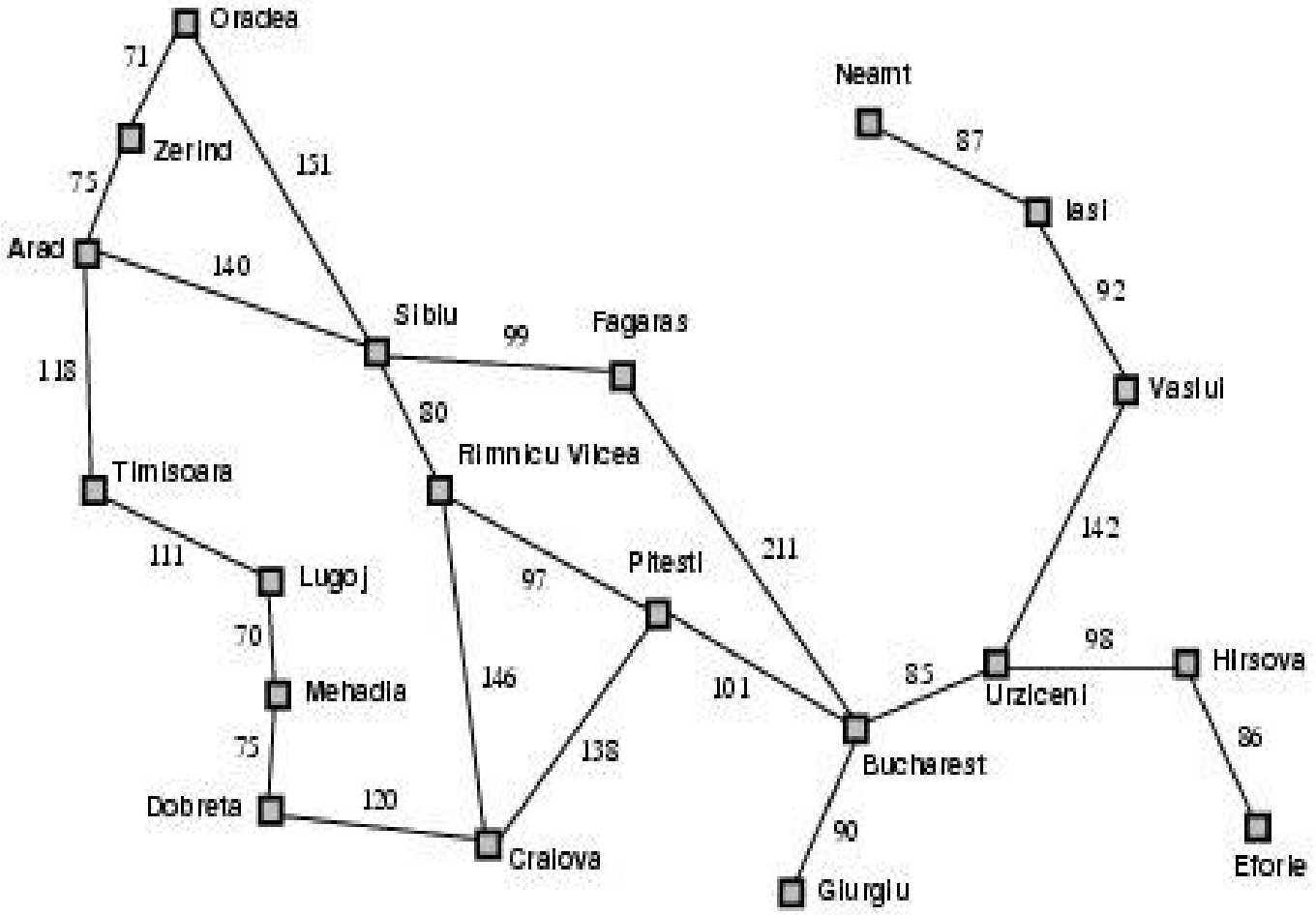
- Node is selected for expansion based on an *evaluation function f(n)*
- Evaluation function estimates distance to the goal
- Choose node which *appears* best
- Implementation:
  - fringe is a priority queue sorted in ascending order of *f*-values

# *A heuristic function* h(n)

- Dictionary defn: *"A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood"*
- For best-first search:

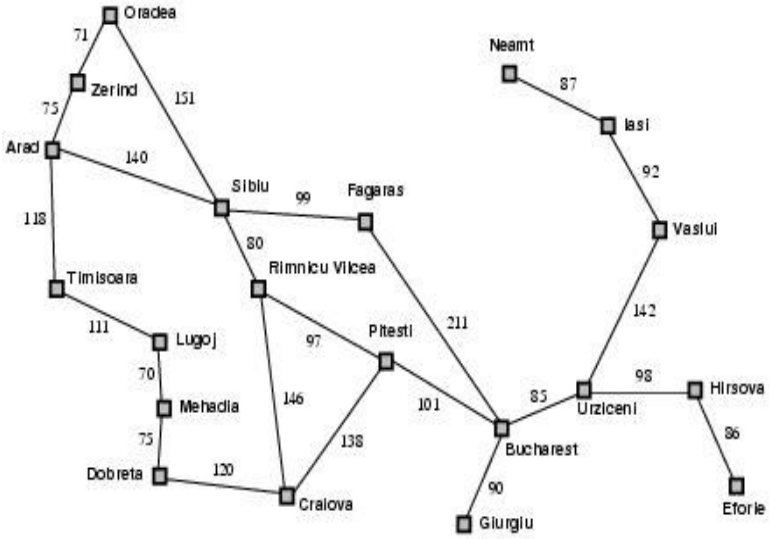
  $h(n)$ = estimated cost of the cheapest path from node $n$ to goal node

# *Example: Romania*



# *Romania with step costs in km*

| | | | |
|---|---|---|---|
| Arad | 366 | Mehadia | 241 |
| Bucharest | 0 | Neamt | 234 |
| Craiova | 160 | Oradea | 380 |
| Dobreta | 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 |

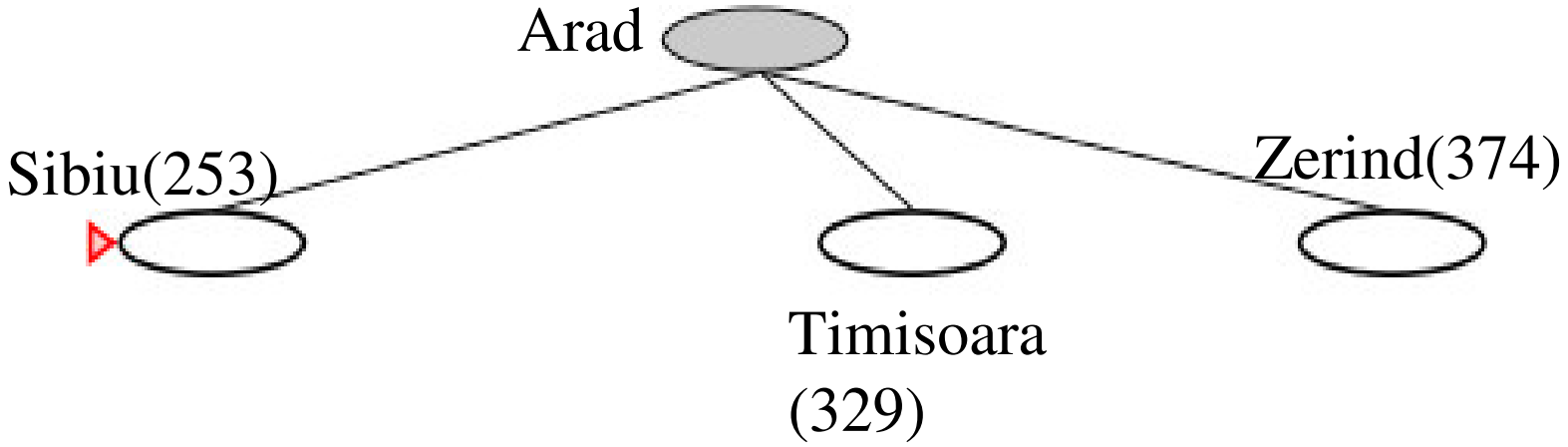



- $h_{SLD}$=straight-line distance heuristic
- $h_{SLD}$ cannot be computed from the problem description itself
- In **greedy best-first search** $f(n)=h(n)$
  - Expand node that is closest to goal

# *Greedy search example*

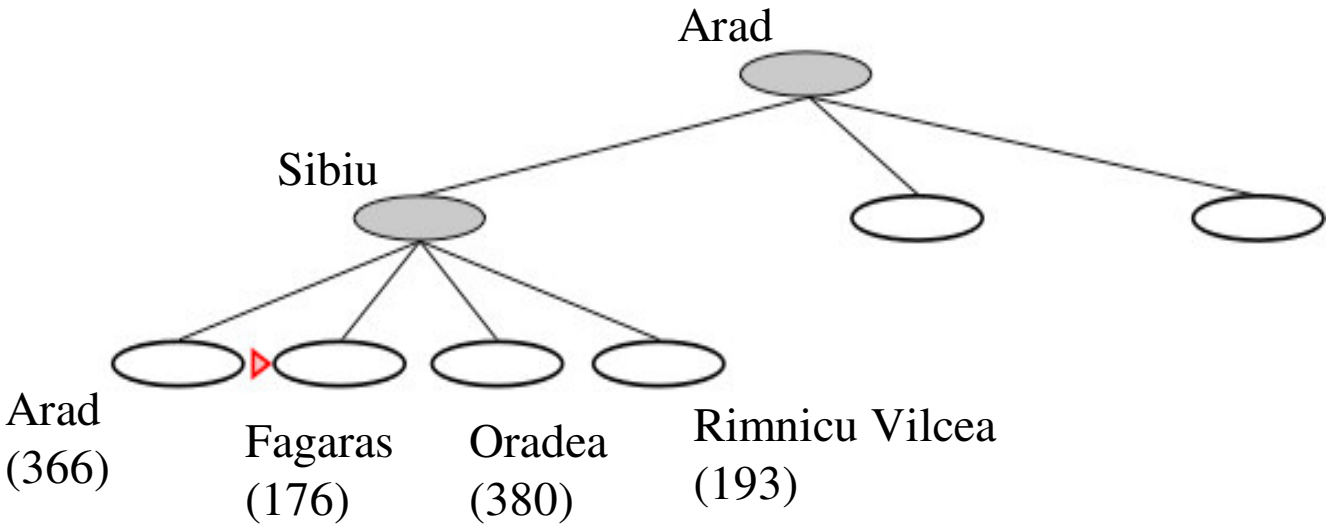

- Greedy search to solve the Arad to Bucharest problem

# *Greedy search example*



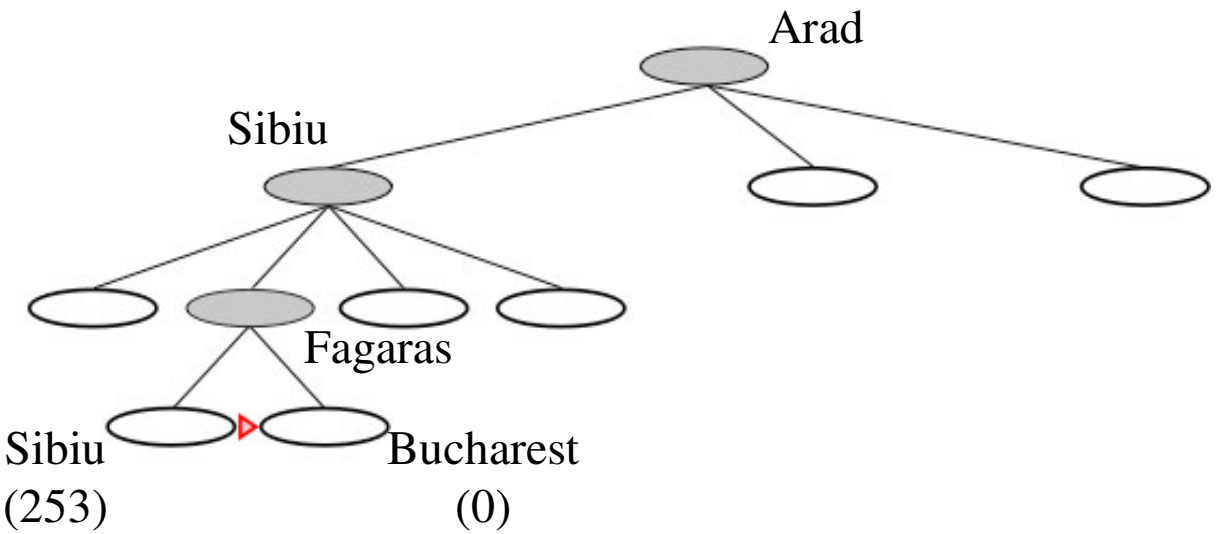

- Greedy best-first search will select Sibiu

# *Greedy search example*




- Greedy best-first search will select Fagaras

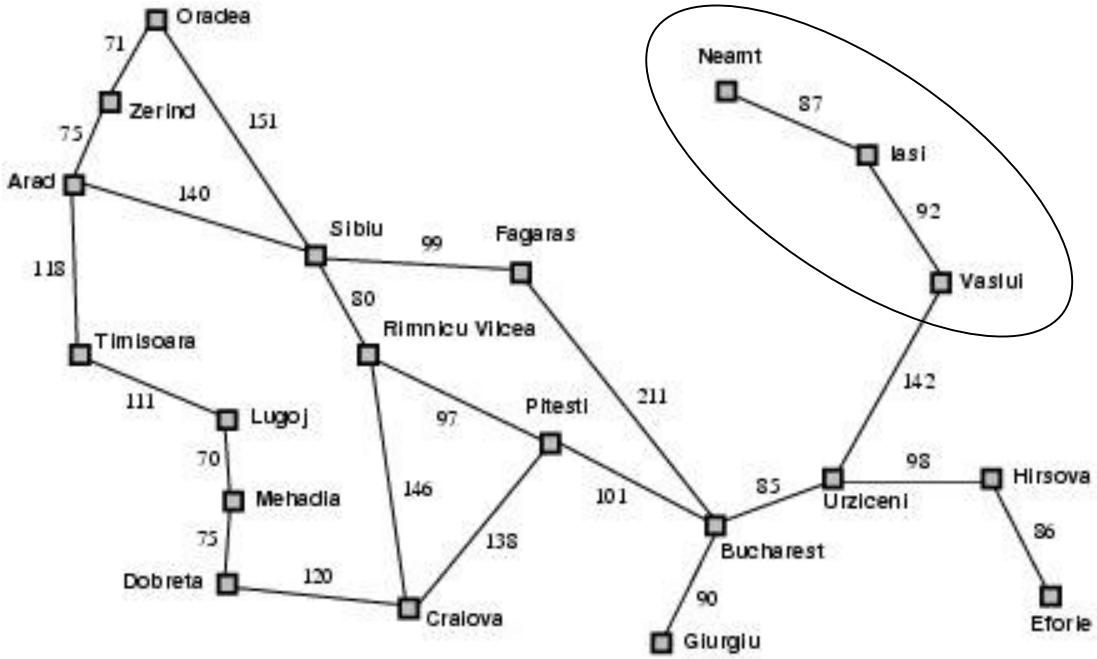
# *Greedy search example*




- Goal reached
  - For this example no node is expanded that is not on the solution path
  - But not optimal (see Arad, Sibiu, Rimnicu Vilcea, Pitesti)

# *Greedy search: evaluation*

- Complete or optimal: no
  - Minimizing h(n) can result in false starts, e.g. Iasi to Fagaras
  - Check on repeated states



# *Greedy search: evaluation*

- Time and space complexity:
  - In the worst case all the nodes in the search tree are generated: $O(b^m)$

    (m is maximum depth of search tree and b is branching factor)
  - But: choice of a good heuristic can give dramatic improvement

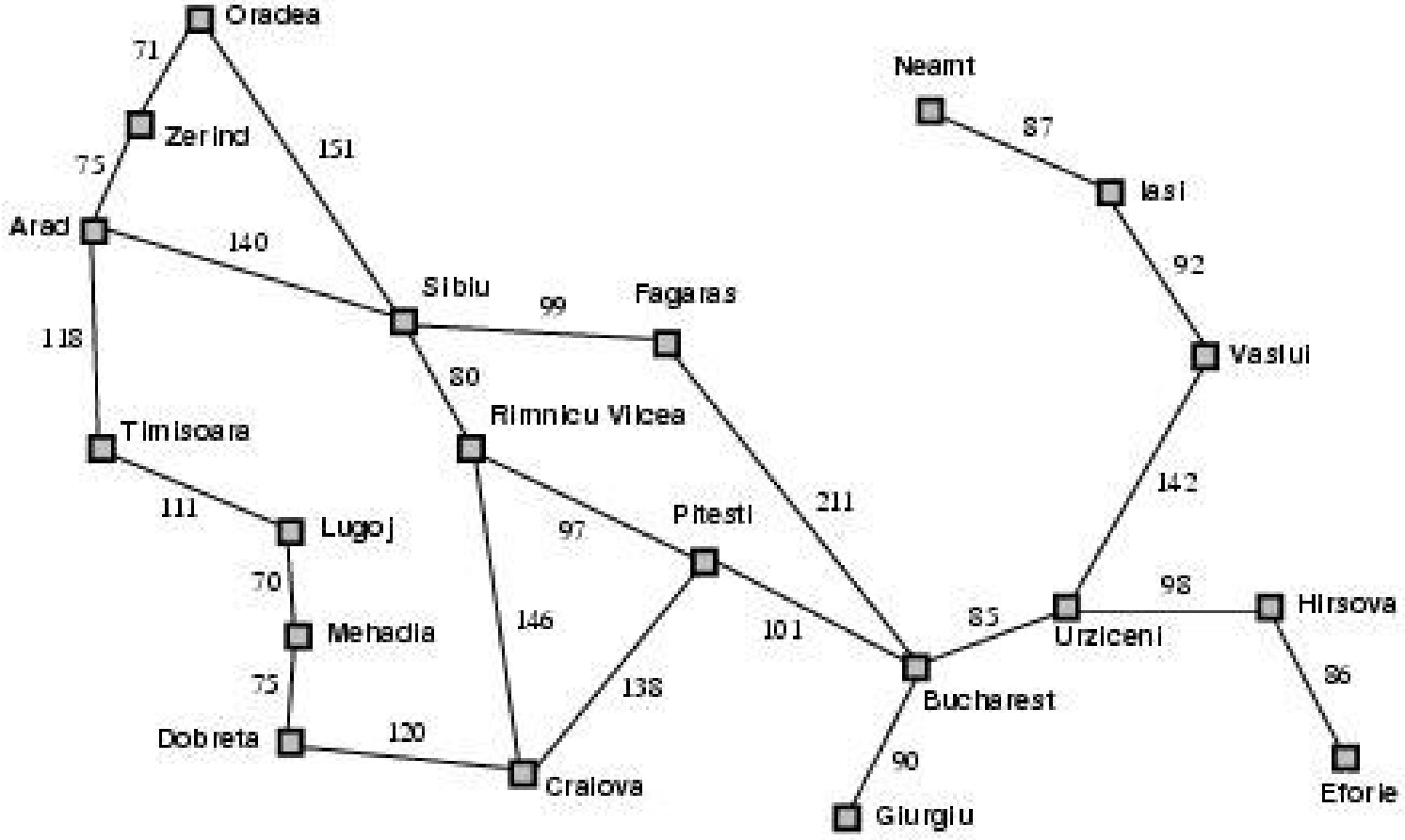
# *A* search*

- Best-known form of best-first search
- Idea: avoid expanding paths that are already expensive
- Evaluation function $f(n)=g(n) + h(n)$
  - $g(n)$: the cost (so far) to reach the node
  - $h(n)$: estimated cost to get from the node to the goal
  - $f(n)$: estimated total cost of path through $n$ to goal
- A* search is both complete and optimal if $h(n)$ satisfies certain conditions

# *A* search*

- A* search is optimal if h(n) is an **admissible heuristic**
- A heuristic is admissible if it *never overestimates* the cost to reach the goal
  - $h(n) <= h^*(n)$ where $h^*(n)$ is the true cost from $n$
- Admissible heuristics are optimistic about the cost of solving the problem
- e.g. $h_{SLD}(n)$ never overestimates the actual road distance

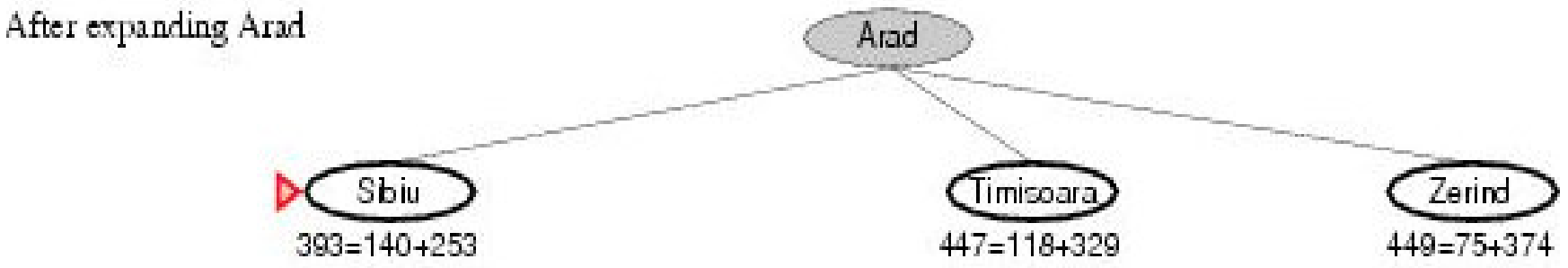
# *Romania example*



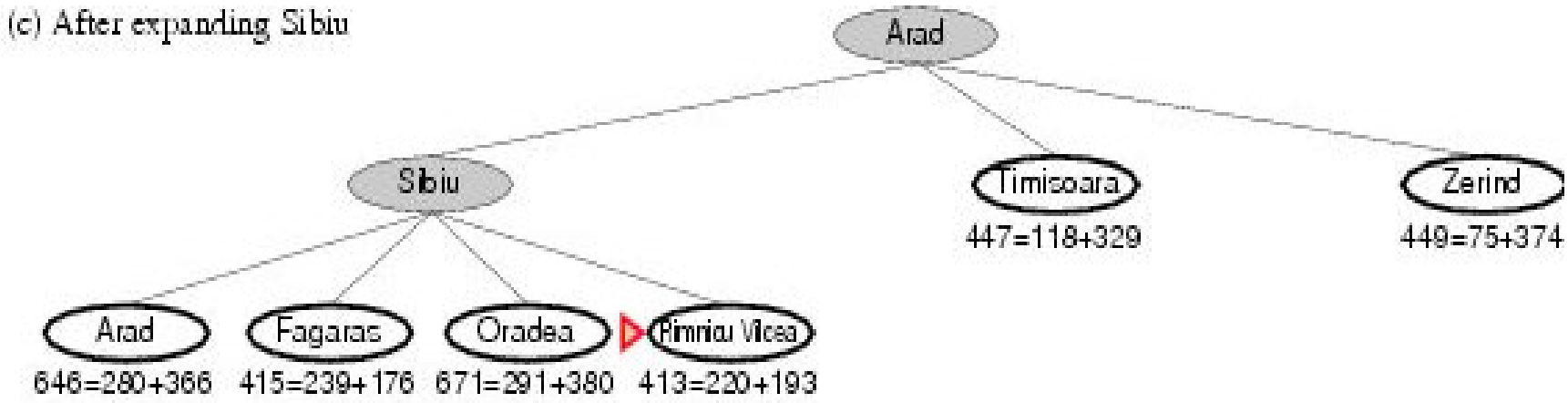
# A* search example

(a) The initial state

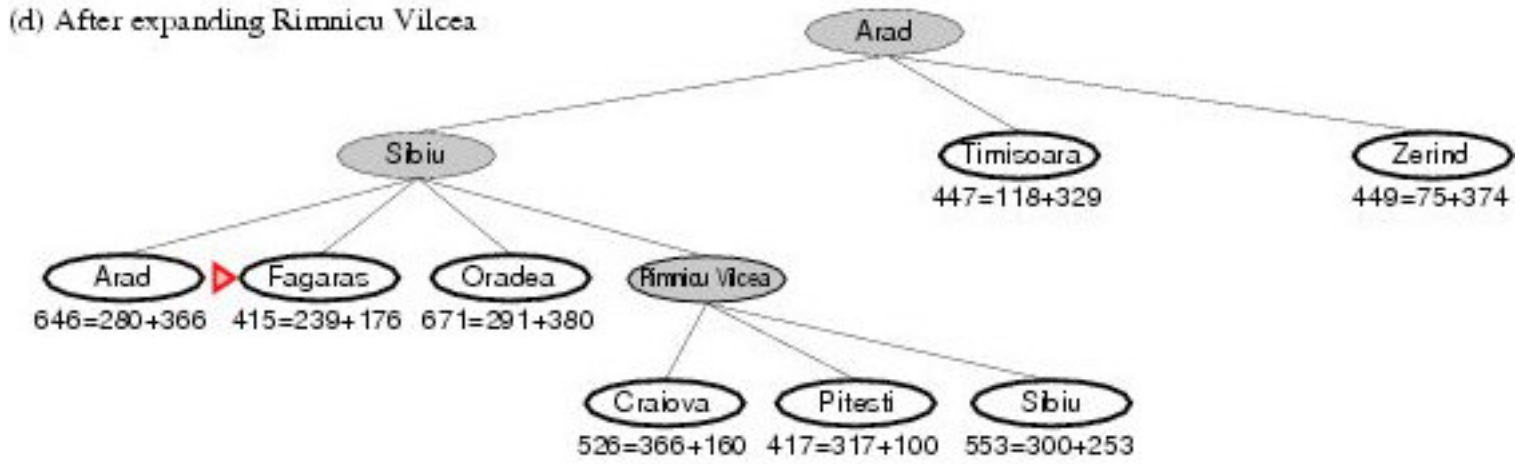
# A* search example



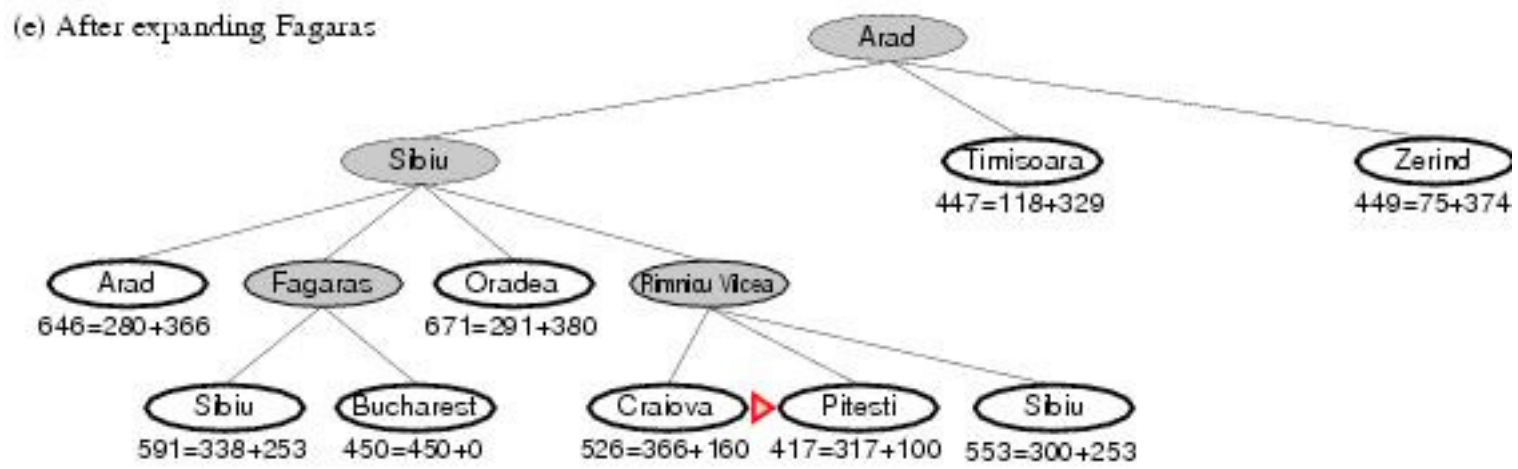
# A* search example



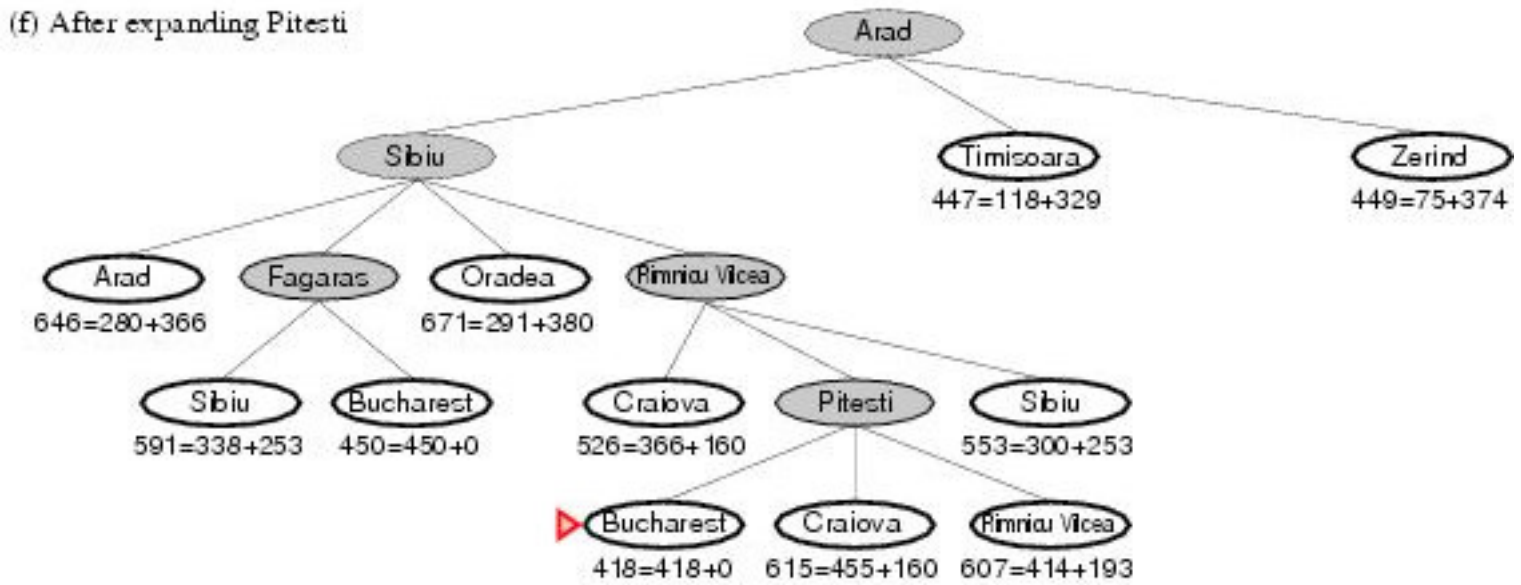
# A* search example



# A* search example



# A* search example



# *A* and GRAPH-SEARCH*

- GRAPH-SEARCH discards new paths to a repeated state
  - So may discard the optimal path
- Solutions:
  - Remove more expensive of the two paths
    - But requires extra book-keeping
  - Ensure that the optimal path to any repeated state is always the first one followed
    - Requires extra condition on *h(n)*: **consistency** (or **monotonicity**)

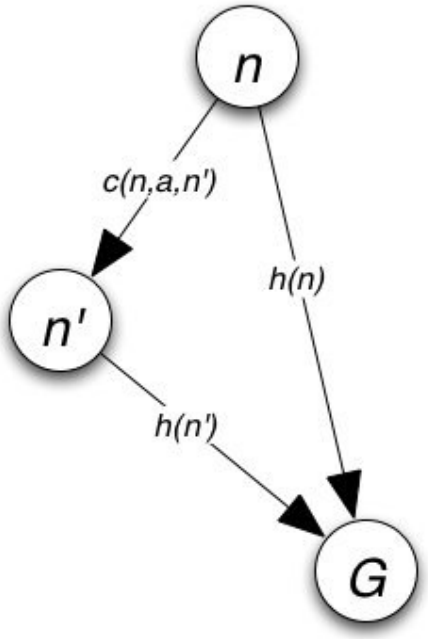
# Consistency

- A heuristic is consistent if

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

- If h is consistent, we have

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




- i.e. f(n) is nondecreasing along any path
  - and so A* using GRAPH-SEARCH expands nodes in non-decreasing order of f(n)

# *A* search: evaluation*

- Complete: yes
  - Unless there are infinitely many nodes with $f<f(G)$
  - Since bands of increasing $f$ are added
- Optimal: yes
  - A* is **optimally efficient** for any given h(n): no other optimal algorithm is guaranteed to expand fewer nodes

# *A* search: evaluation*

- Time complexity:
  - number of nodes expanded is still exponential in length of solution
- Space complexity:
  - All generated nodes are kept in memory
  - A* usually runs out of space before running out of time