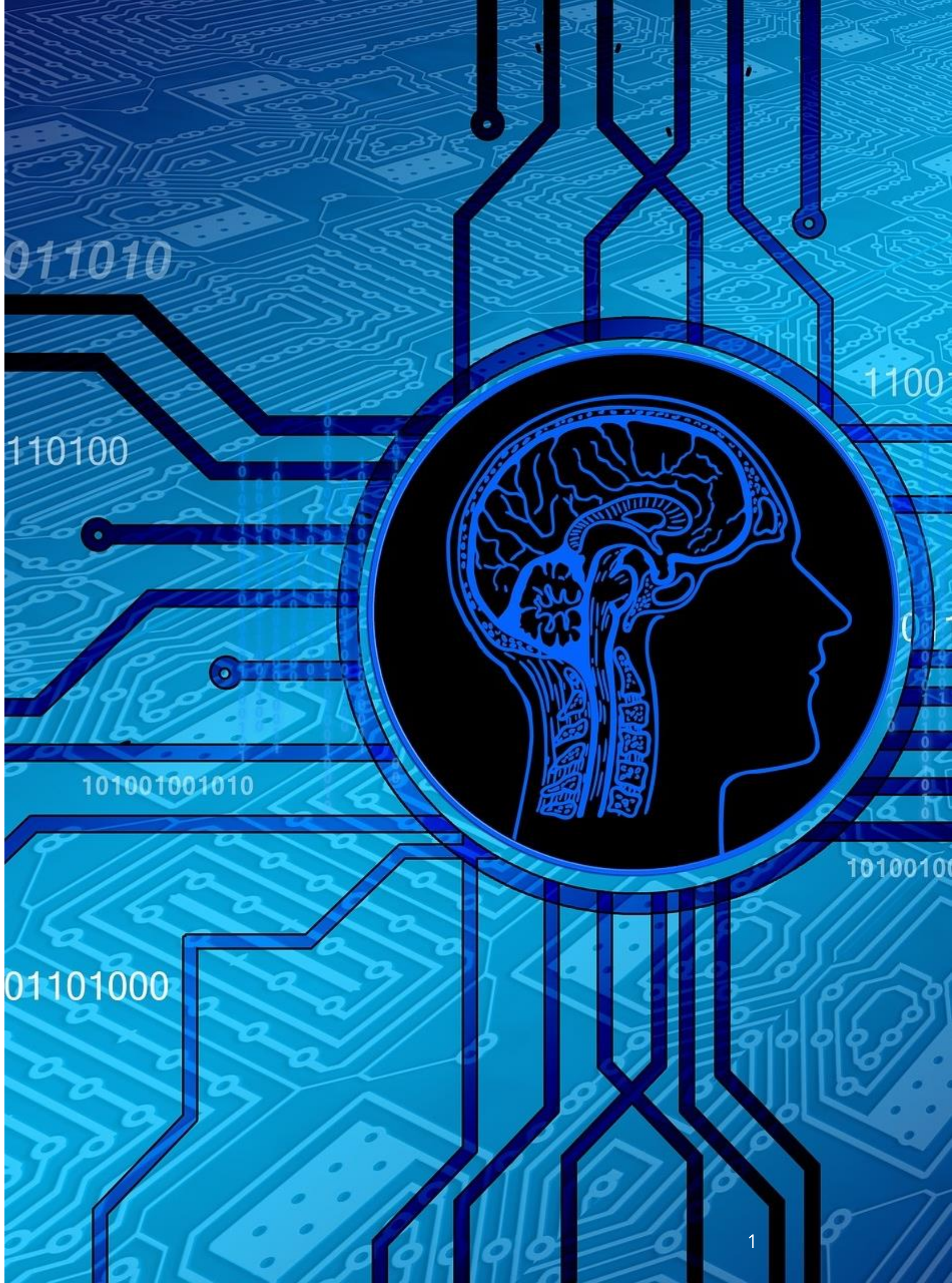
Informed Search

Informatics 2D: Reasoning and Agents

Lecture 4



Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Review: Summary of algorithms

Review: Tree search

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

A search strategy is defined by picking the order of node expansion from the ***frontier.***

Review: Graph search

```
function GRAPH-SEARCH(problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  initialize the explored set to be empty
  loop do
     if the frontier is empty then return failure
     choose a leaf node and remove it from the frontier
     if the node contains a goal state then return the corresponding solution
     add the node to the explored set
     expand the chosen node, adding the resulting nodes to the frontier
        only if not in the frontier or explored set
```

A search strategy is defined by picking the order of node expansion from the ***frontier.***

Making search 'informed'

- Tree-Search
- Graph Search

What if we **order** the nodes in the frontier by decreasing *desirability*?

A search strategy is defined by picking the order of node expansion from the ***frontier.***

Best-first search

An instance of general TREE-SEARCH or GRAPH-SEARCH

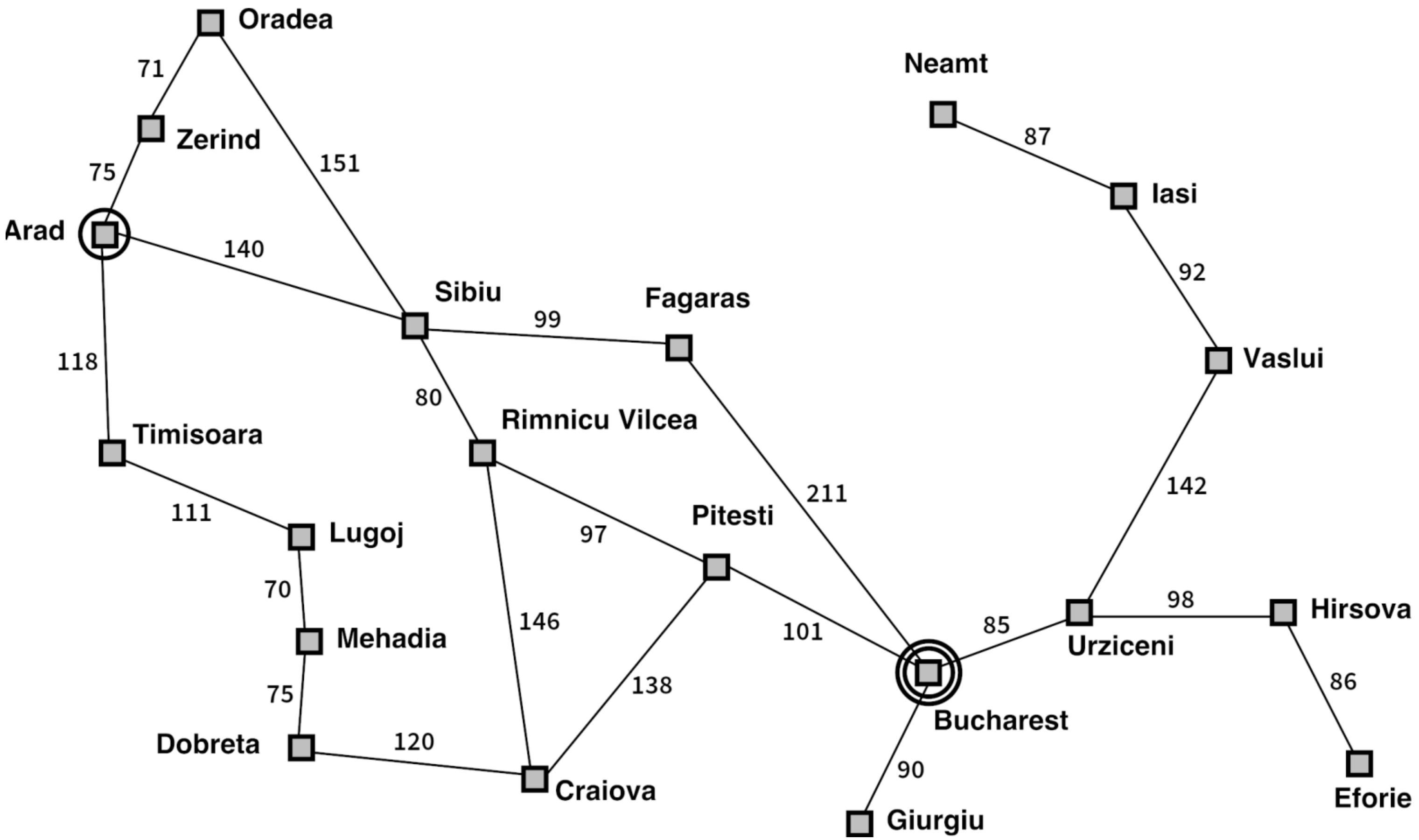
→ Use an evaluation function *f(n)* for each node *n*

- estimate of "desirability"

→ Expand most desirable unexpanded node, usually the node with the lowest evaluation

Heuristics

- Any method that is believed or practically proven to be useful for the solution of a given problem.
  - No guarantee that it will always work or lead to an optimal solution!
- We use heuristics to guide search.
  - This may not change the worst-case complexity of the algorithm, but can help in the average case.
- We will introduce admissibility, consistency conditions to identify good heuristics.



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

Romania Example

Greedy best-first search

Greedy best-first search

- Evaluation function *f(n) = h(n)* (heuristic)
  - estimated cost of cheapest path from state at node *n* to a *goal* state
  - e.g., $h_{SLD}(n)$ = straight-line distance from *n* to Bucharest

- Greedy best-first search expands the node that appears to be closest to goal.

Arad							

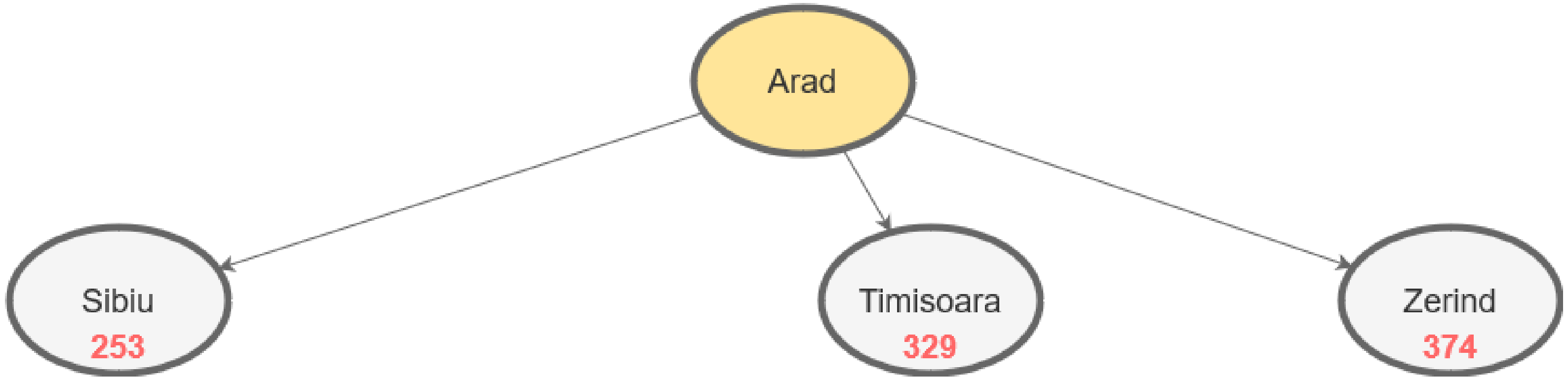
Greedy best-first search

Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search

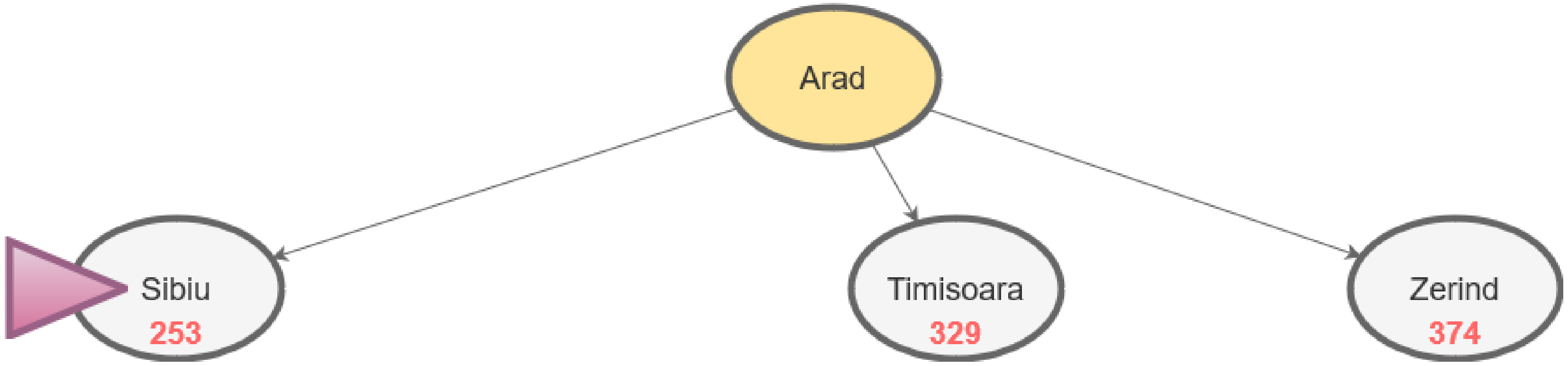
Sibiu	Timiso-ara	Zerind					

Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Greedy best-first search

Sibiu	Timiso-ara	Zerind					



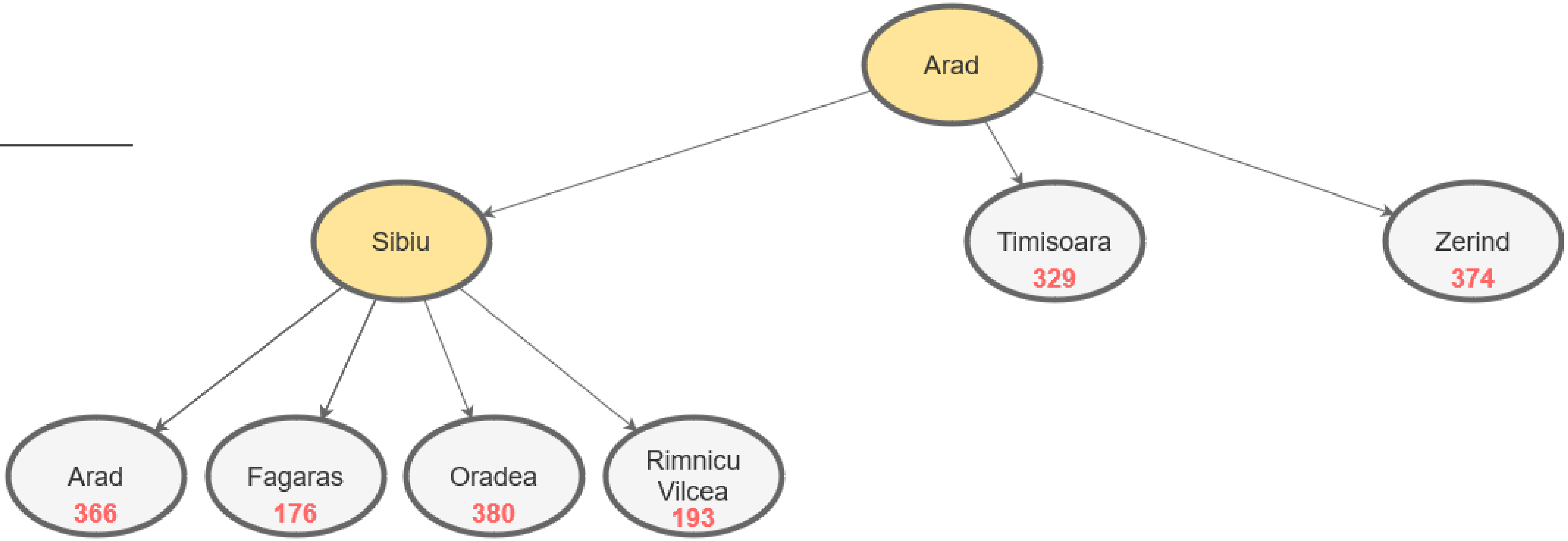
Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Timiso-ara	Zerind	Arad	Fagaras	Oradea	RV		

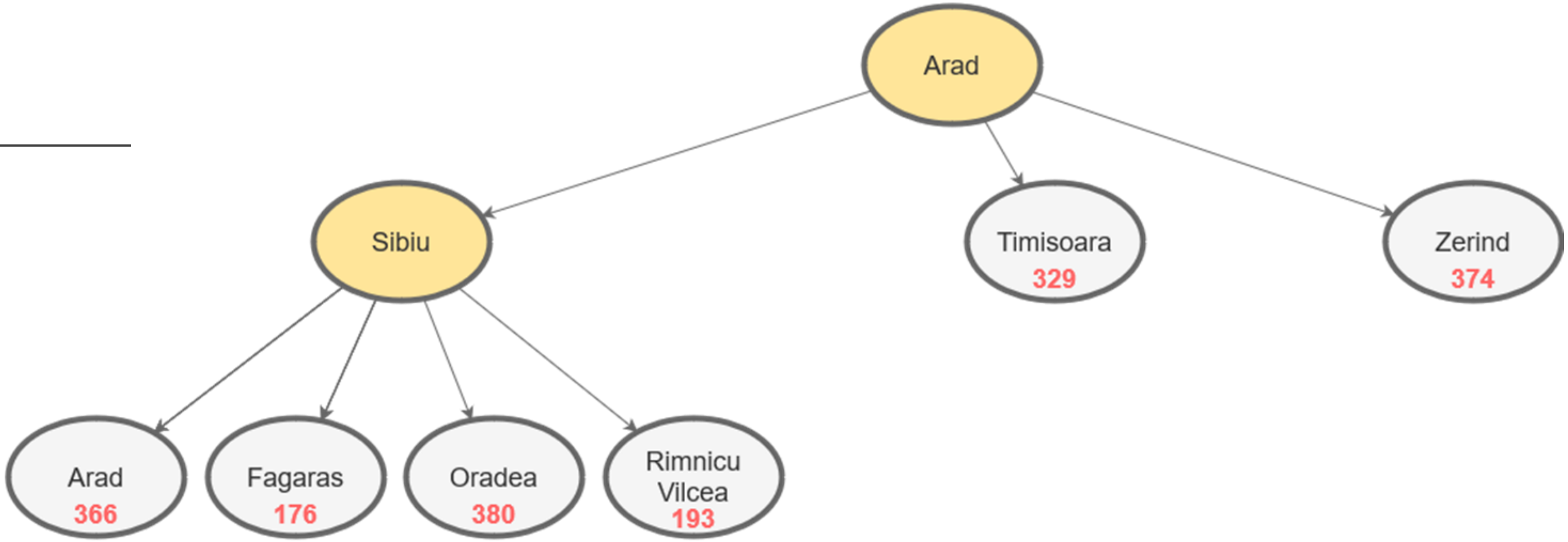
Greedy best-first search

Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Greedy best-first search

Fagaras	RV	Timiso-ara	Arad	Zerind	Oradea		



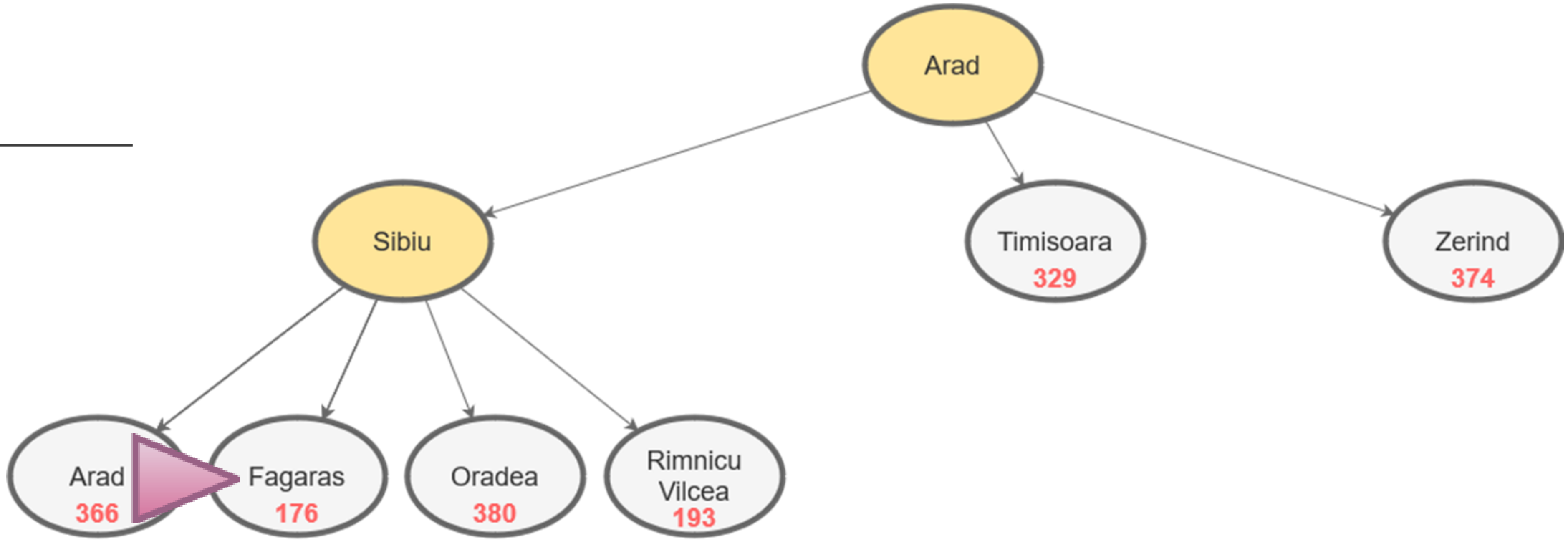
Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Greedy best-first search

Fagaras	RV	Timiso-ara	Arad	Zerind	Oradea		

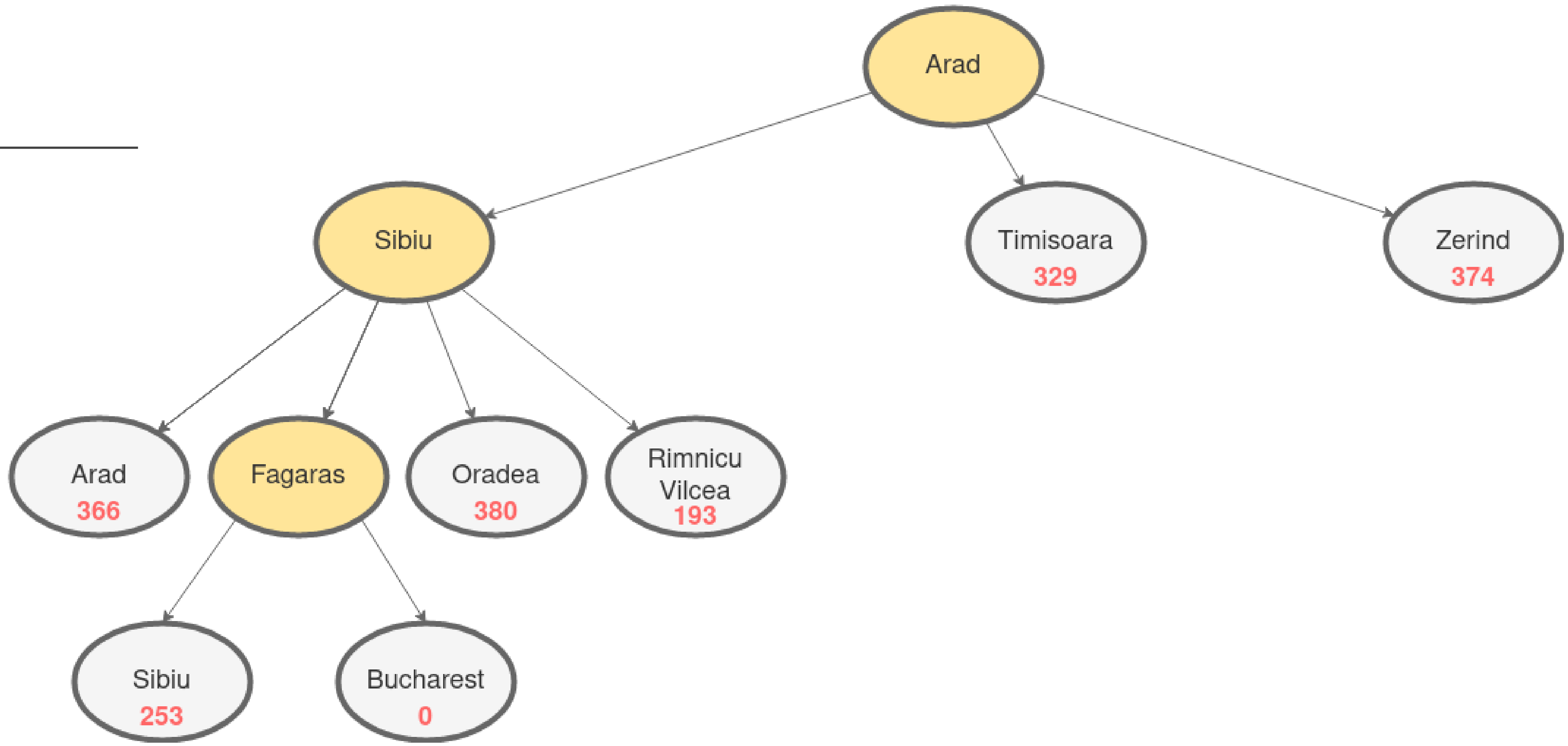
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Greedy best-first search

Fagaras	RV	Timiso-ara	Arad	Zerind	Oradea	Sibiu	Bucha-rest

Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Properties of best-first search

- **Complete?**
- **Time complexity?**
- **Space complexity?**
- **Optimal?**

Properties of best-first search

Complete?
No! Can get stuck in loops.

Time complexity?

Space complexity?

Optimal?

Properties of best-first search

Complete?
No! Can get stuck in loops.

GRAPH-SEARCH is complete in finite spaces.

Time complexity?

Space complexity?

Optimal?

Properties of best-first search

Complete?
No! Can get stuck in loops.

Time complexity?
$O(b^m)$ for tree version

Space complexity?

Optimal?

Properties of best-first search

Complete?
No! Can get stuck in loops.

Time complexity?
$O(b^m)$ for tree version

Space complexity?

Optimal?

A good heuristic can lead to dramatic improvement!

Properties of best-first search

Complete?
No! Can get stuck in loops.

Time complexity?
$O(b^m)$ for tree version

Space complexity?
$O(b^m)$ – keeps all nodes in memory

Optimal?

Properties of best-first search

Complete?
No! Can get stuck in loops.

Time complexity?
$O(b^m)$ for tree version

Space complexity?
$O(b^m)$ – keeps all nodes in memory

Optimal?
No

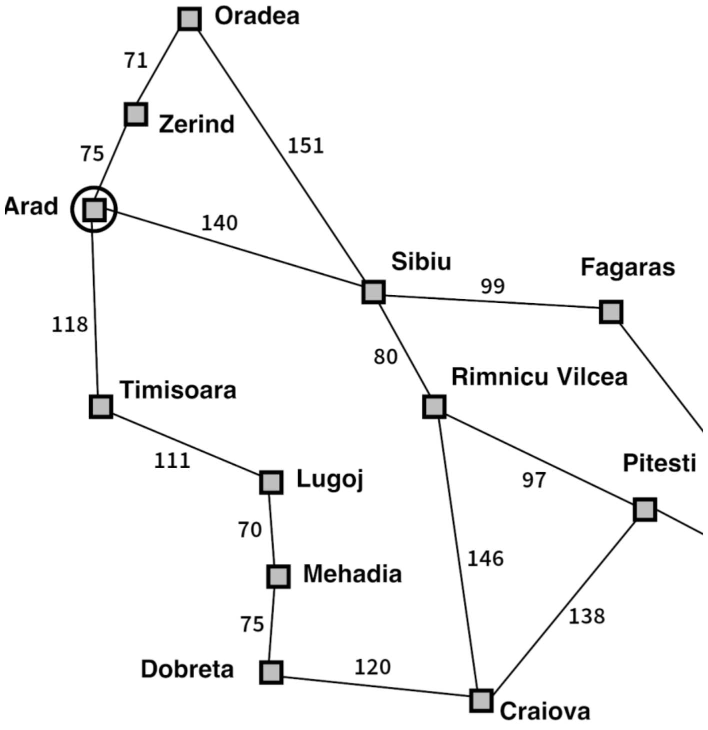
A* Search

A* search

- Evaluation function $f(n) = g(n) + h(n)$
  - $g(n)$ = cost so far to reach n
  - $h(n)$ = estimated cost from n to goal
  - $f(n)$ = estimated total cost of path through n to goal

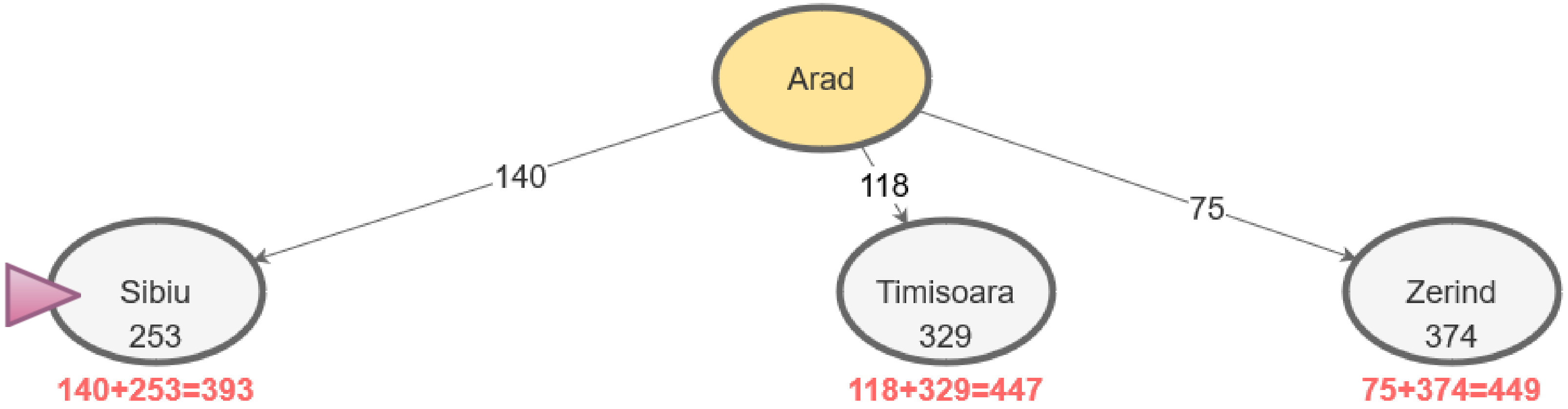
- Avoid expanding paths that are already expensive

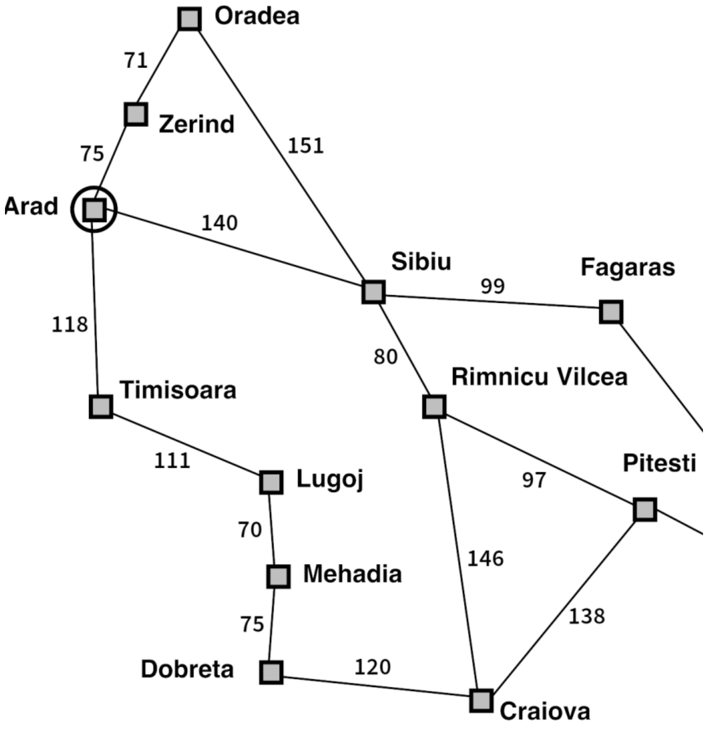
A* search



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search

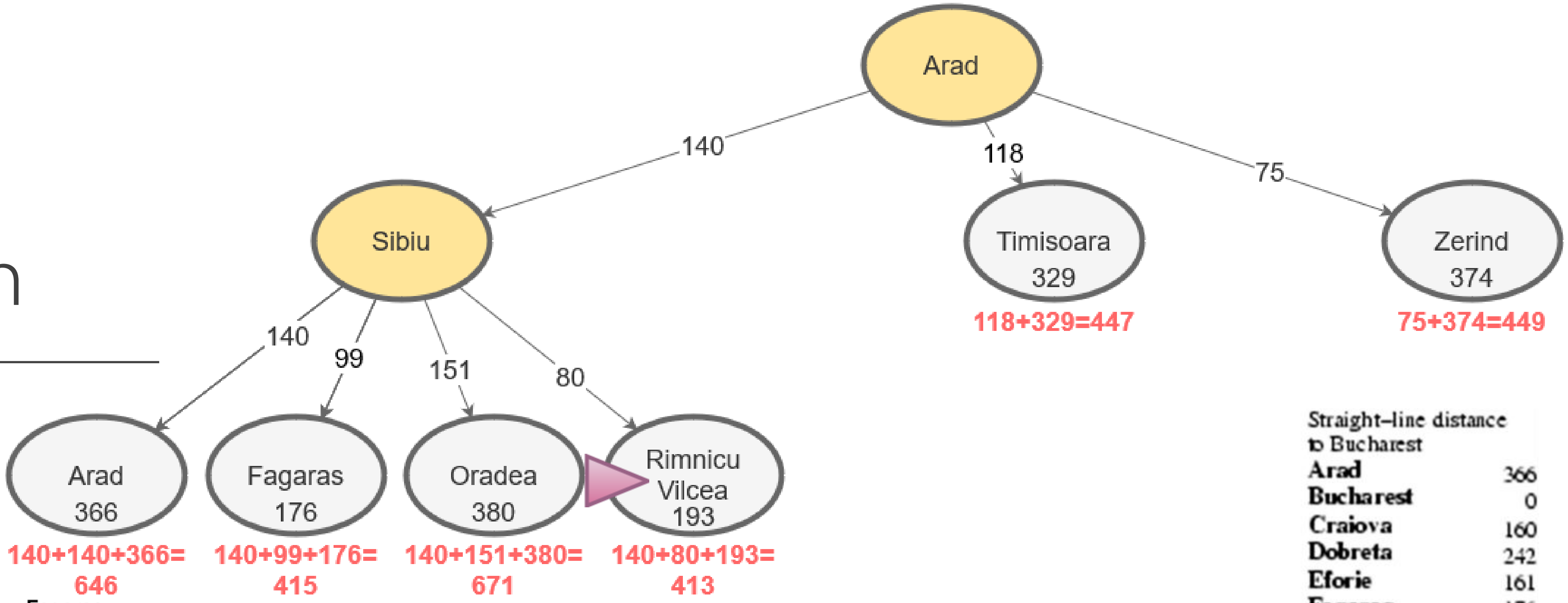


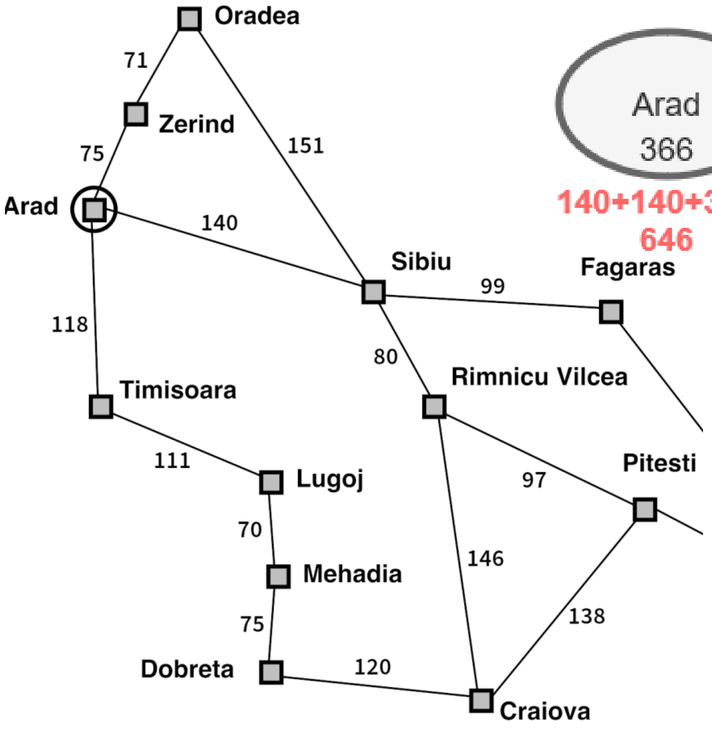


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search

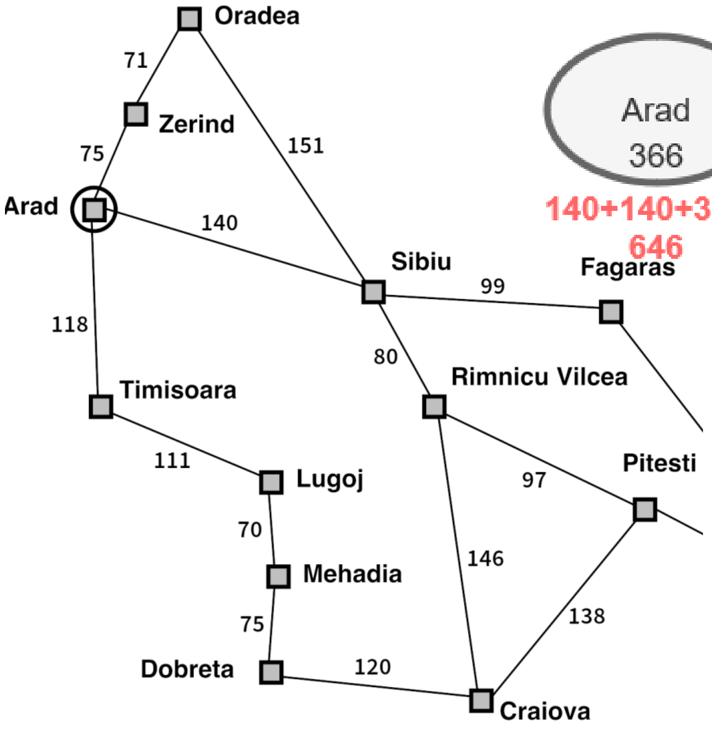


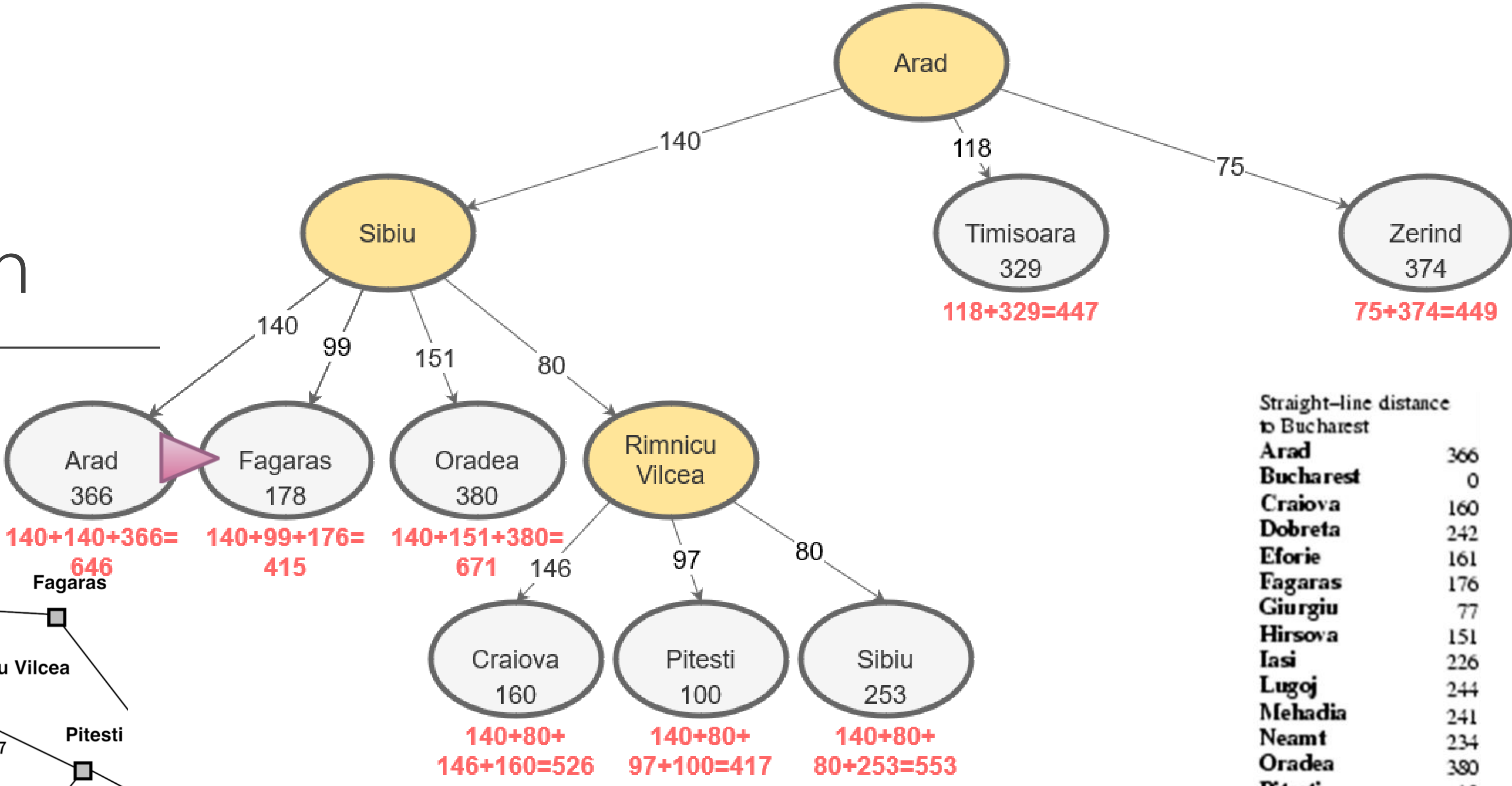


Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search

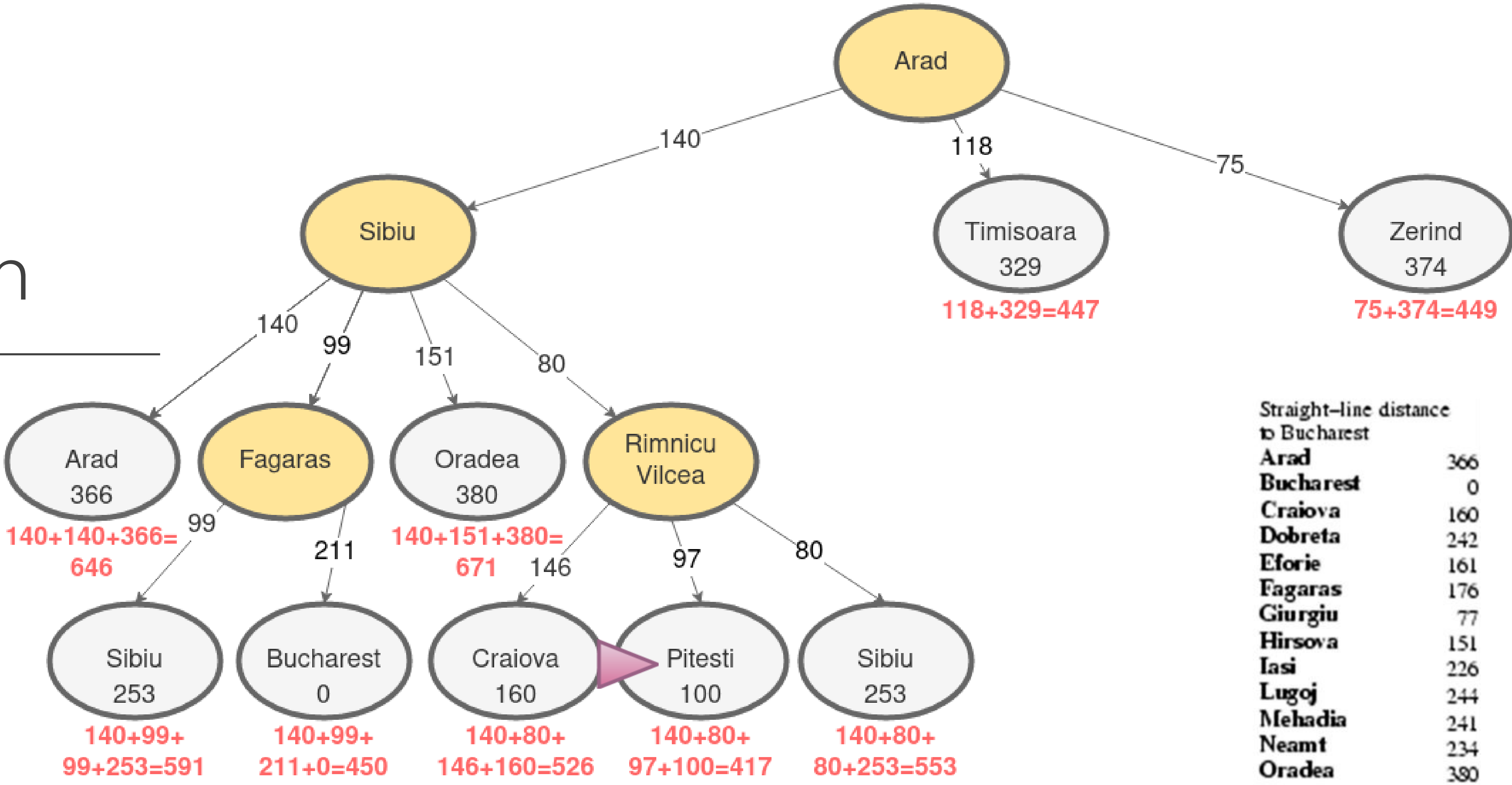




Straight–line distance to Bucharest

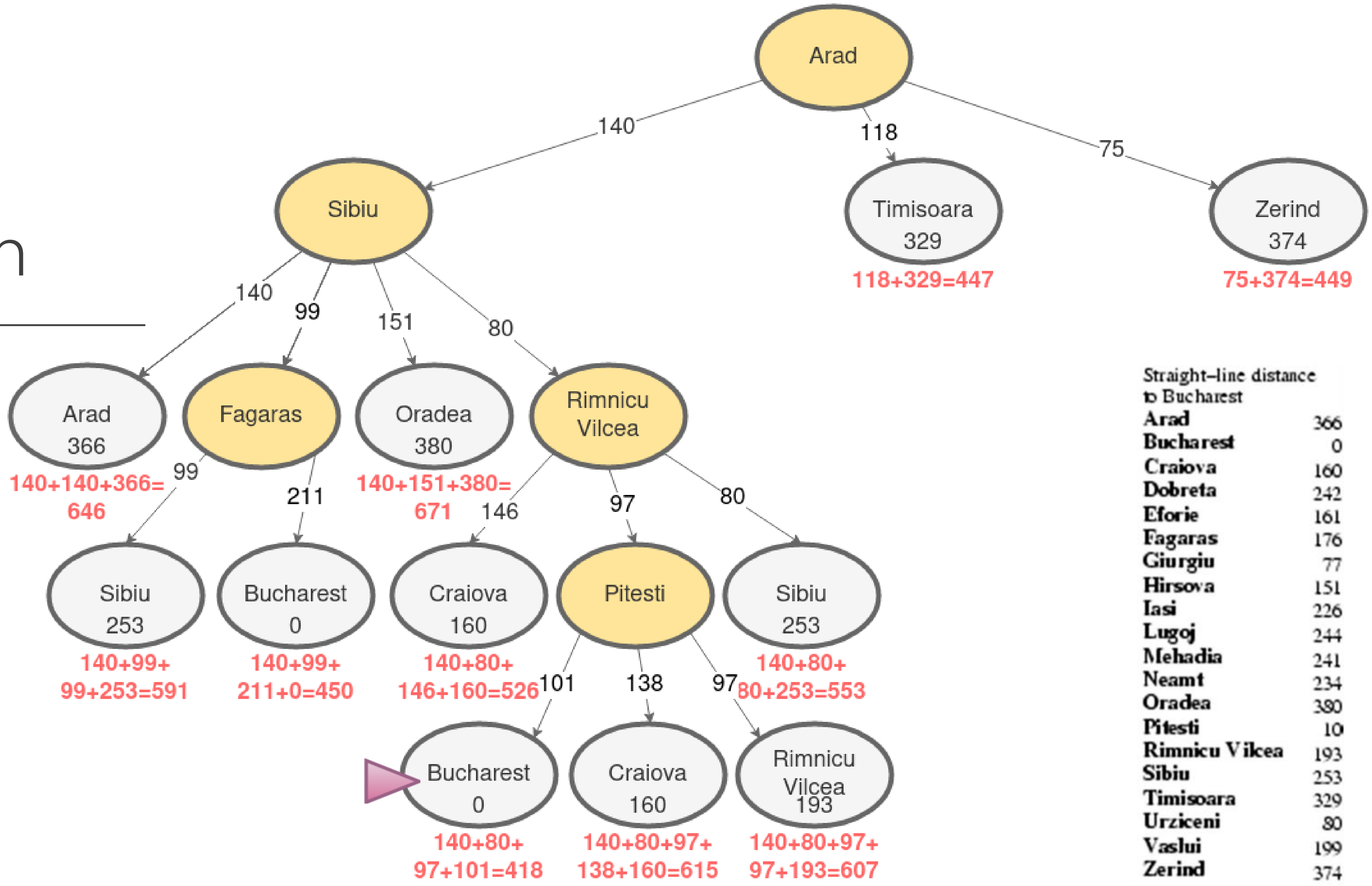
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search



Straight–line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Heuristics

Admissible heuristics

- A heuristic *h(n)* is admissible if for every node *n*:

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

where $\boldsymbol{h^*(n)}$ is the true cost to reach the goal state from *n*.

- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic

Example: $h_{SLD}(n)$ (never overestimates the actual road distance)

Admissible heuristic = optimal A*

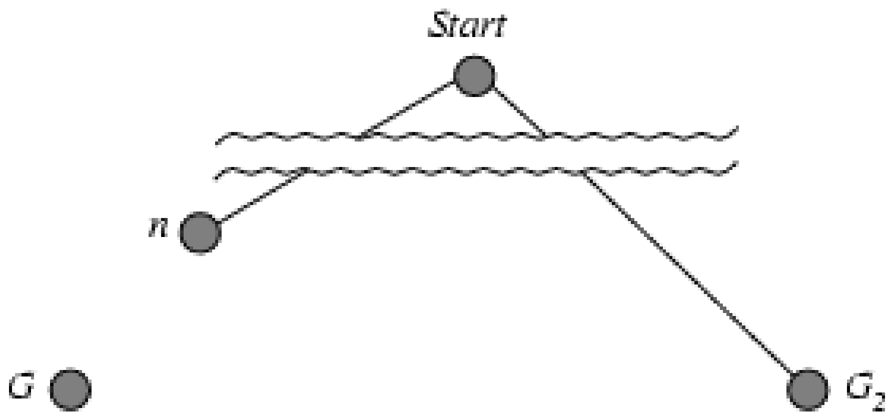
h(n) never overestimates the cost to reach the goal

Thus, $f(n) = g(n) + h(n)$ never overestimates the true cost of a solution

THEOREM

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

Proof: Optimality of A^*

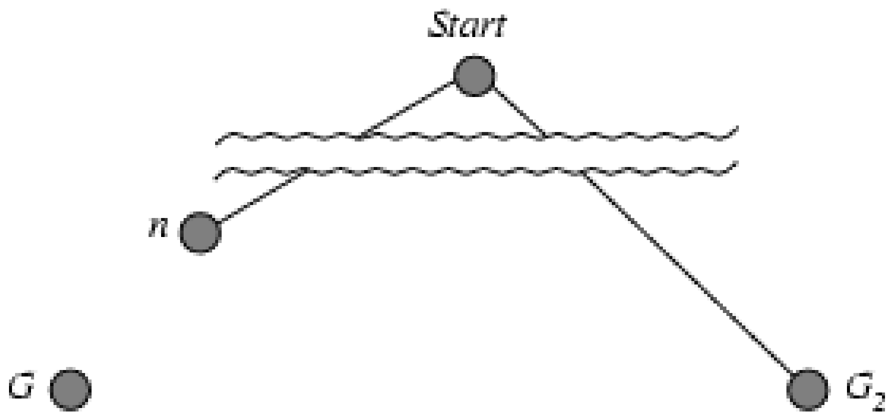


Suppose some suboptimal goal G_2 has been generated and is in the frontier.

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 $h(G) = 0$
$f(G_2) > f(G)$	from above

Proof: Optimality of A*



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.

$f(G) < f(G_2)$ from above

$h(n) \leq h^*(n)$ since h is admissible

$g(n) + h(n) \leq g(n) + h^*(n)$

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

Hence $f(n) < f(G_2)$, and A will never select G2 for expansion*

Consistent heuristics

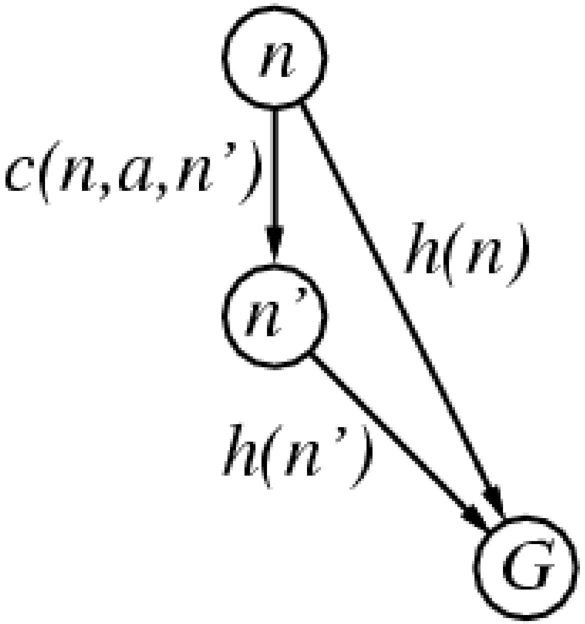
A heuristic *h(n)* is consistent if for every node *n*, every successor *n'* of *n* generated by any action *a*,

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

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 non-decreasing along any path.



THEOREM

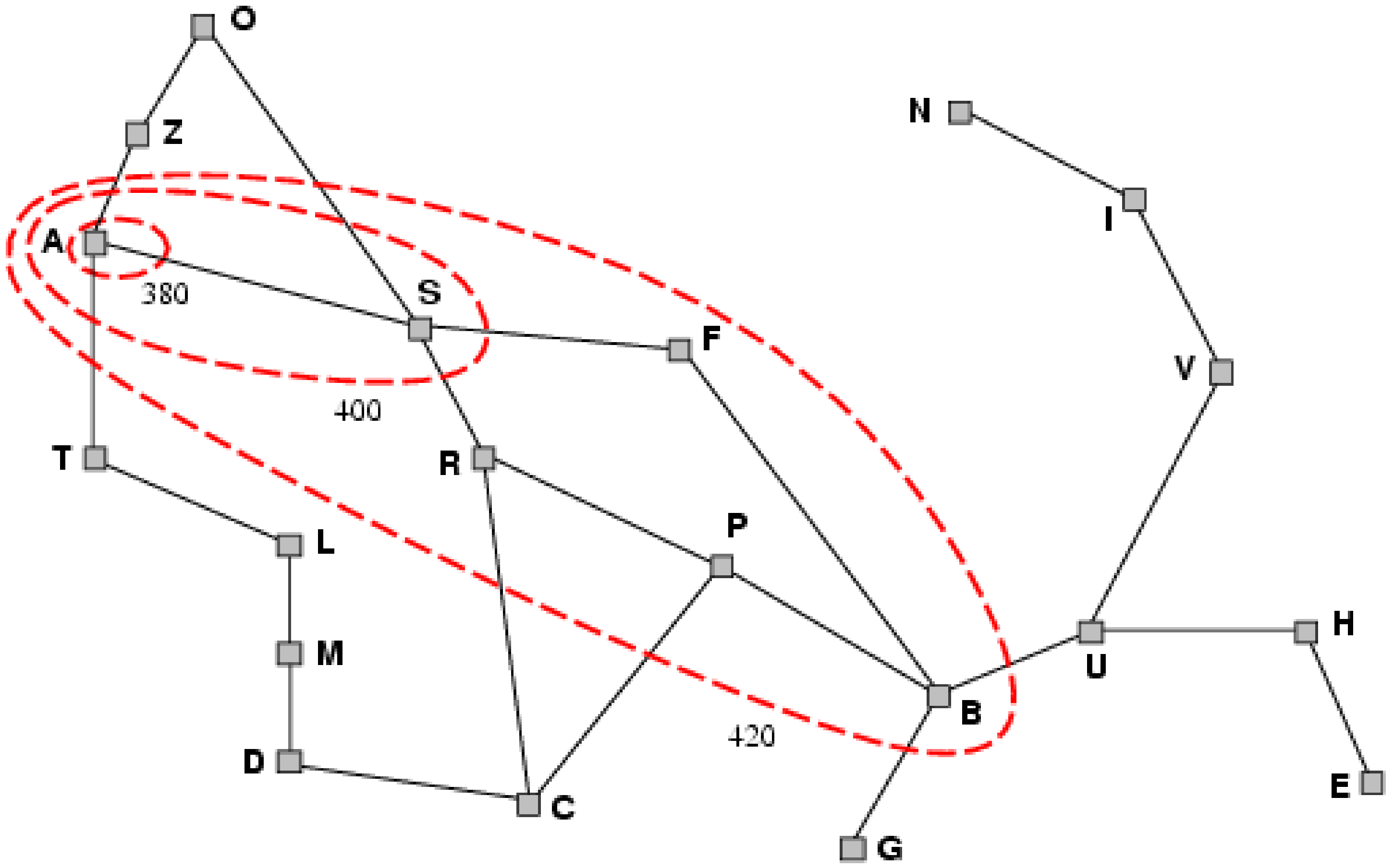
If $h(n)$ is consistent, A* using **GRAPH-SEARCH** is optimal

Optimality of A*

A* expands nodes in order of increasing *f* value

Gradually adds "*f*-contours" of nodes

Contour *i* has all nodes with $f=f_i$, where $f_i < f_{i+1}$



Properties of A*

- **Complete?**
- **Time complexity?**
- **Space complexity?**
- **Optimal?**

Properties of A*

Complete?
Yes (unless there are infinitely many nodes with $f \leq f(G)$)

Time complexity?

Space complexity?

Optimal?

Properties of A*

Complete?
Yes (unless there are infinitely many nodes with $f \leq f(G)$)

Time complexity?
Exponential

Space complexity?
Keeps all nodes in memory

Optimal?

Properties of A*

Complete?
Yes (unless there are infinitely many nodes with $f \leq f(G)$)

Time complexity?
Exponential

Space complexity?
Keeps all nodes in memory

Optimal?
Yes

Admissible heuristics

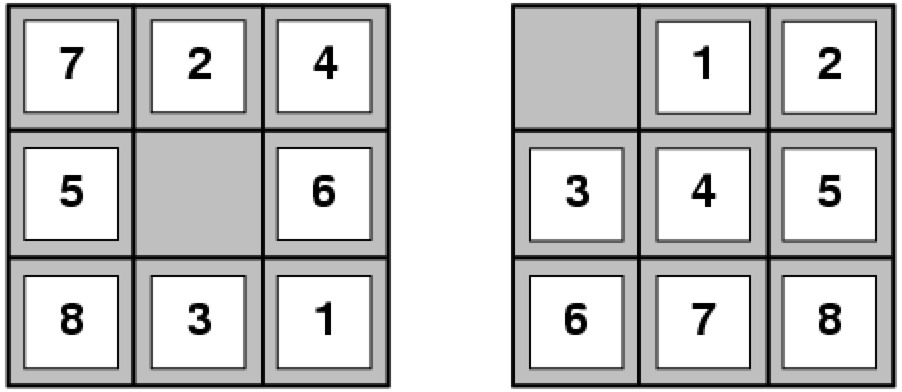
Example: 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)

$h_1(S) = ?$

$h_2(S) = ?$



Start State Goal State

Relaxed problems

- A problem with <u>fewer restrictions</u> on the actions is called a *relaxed problem.*
- The cost of an optimal solution to a relaxed problem is an *admissible heuristic* for the original problem.
- If the rules of the 8-puzzle are relaxed so that a tile can move <u>anywhere</u>,
  - then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to <u>any adjacent square</u>,
  - then $h_2(n)$ gives the shortest solution

Use relaxation to automatically generate admissible heuristics!

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible) then
  - h_2 dominates h1
  - h_2 is better for search
- Typical search costs (*average number of nodes expanded*):
  - d=12 IDS = 3,644,035 nodes
    $A^*(h_1)$ = 227 nodes
    $A^*(h_2)$ = 73 nodes
  - d=24 IDS ≈ 54,000,000,000 nodes
    $A^*(h_1)$ = 39,135 nodes
    $A^*(h_2)$ = 1,641 nodes

Summary

Smart search based on heuristic scores.

- Best-first search
- Greedy best-first search
- A* search
- Admissible heuristics and optimality.

Why?

- Informed search allows us to use domain knowledge to our advantage.
- Optimality over some utility can often be the top priority.
- A* is **very** popular! (e.g., pathfinding)
- A* is simple, yet very efficient.
- A* is *too* good sometimes (e.g., in games).