



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

IT3160E

Introduction to Artificial Intelligence

Chapter 3 – Problem solving

Part 2: problem-solving by searching using informed search strategies

Lecturer:

Muriel VISANI

Acknowledgements:

Le Thanh Huong

Tran Duc Khanh

Department of Information Systems

School of Information and Communication Technology - HUST

Content of the course

- ❑ Chapter 1: Introduction
- ❑ Chapter 2: Intelligent agents
- ❑ Chapter 3: Problem Solving
 - Search algorithms, adversarial search
 - Constraint Satisfaction Problems
- ❑ Chapter 4: Knowledge and Inference
 - Knowledge representation
 - Propositional and first-order logic
- ❑ Chapter 5: Uncertain knowledge and reasoning
- ❑ Chapter 6: Advanced topics
 - Machine learning
 - Computer Vision

Outline

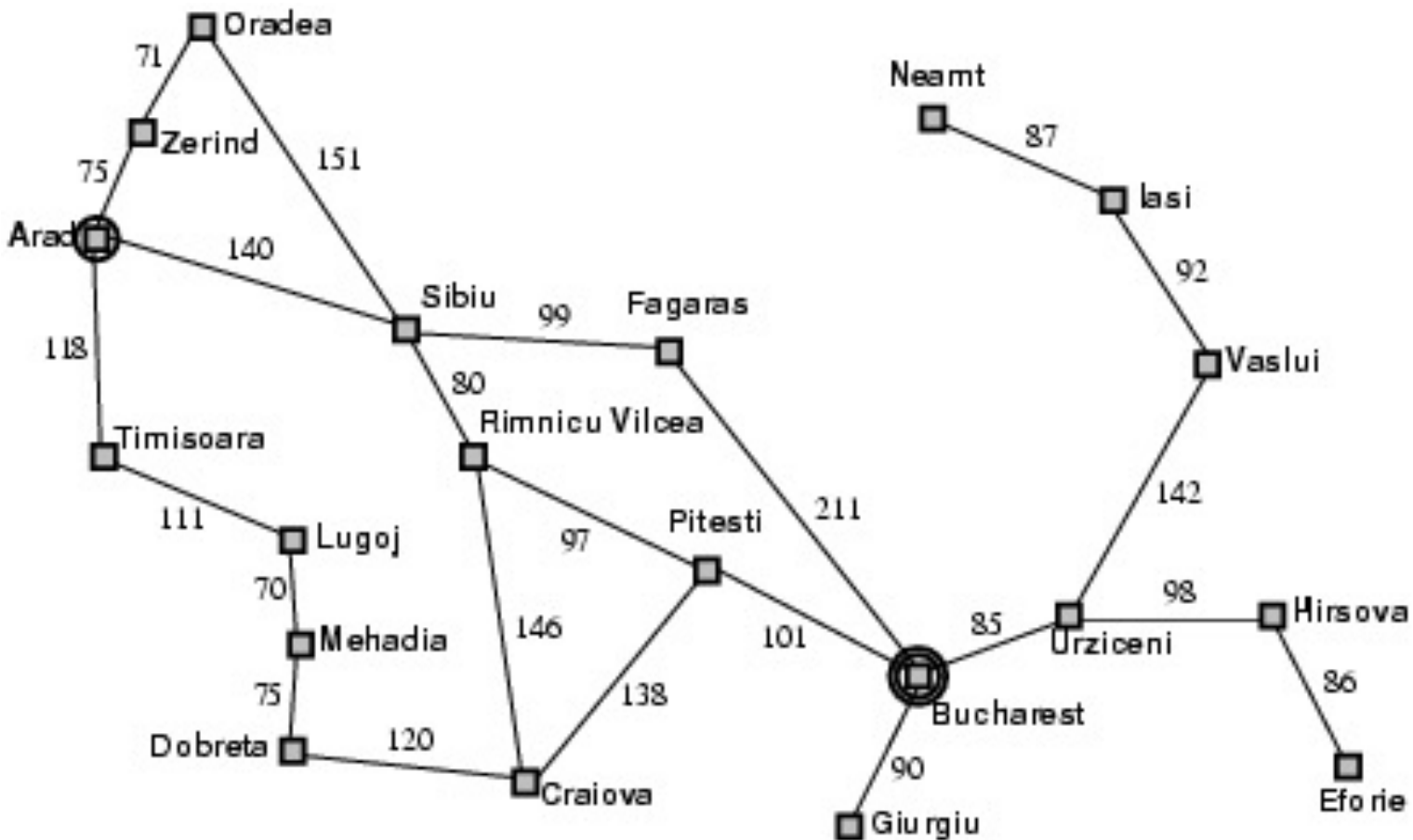
- Chapter 3 – part 1:
 - Introduction, Problem-solving agents, Problem formulation, Basic (uninformed) search algorithms, Homework
- Chapter3 - part 2: searching using informed search strategies
 - Graph search
 - Informed search strategies
 - Introduction
 - Best-first search
 - Greedy best-first search
 - Presentation of A* search
 - About heuristic quality
 - Contours of search
 - A* search evaluation
 - Summary
 - Homework

Goal of this Lecture

Goal	Description of the goal or output requirement	Output division/ Level (I/T/U)
M1	Understand basic concepts and techniques of AI	1.2
M4	Be able to identify research areas and potential developments of artificial intelligence	4.1-4.5

Informed search strategies

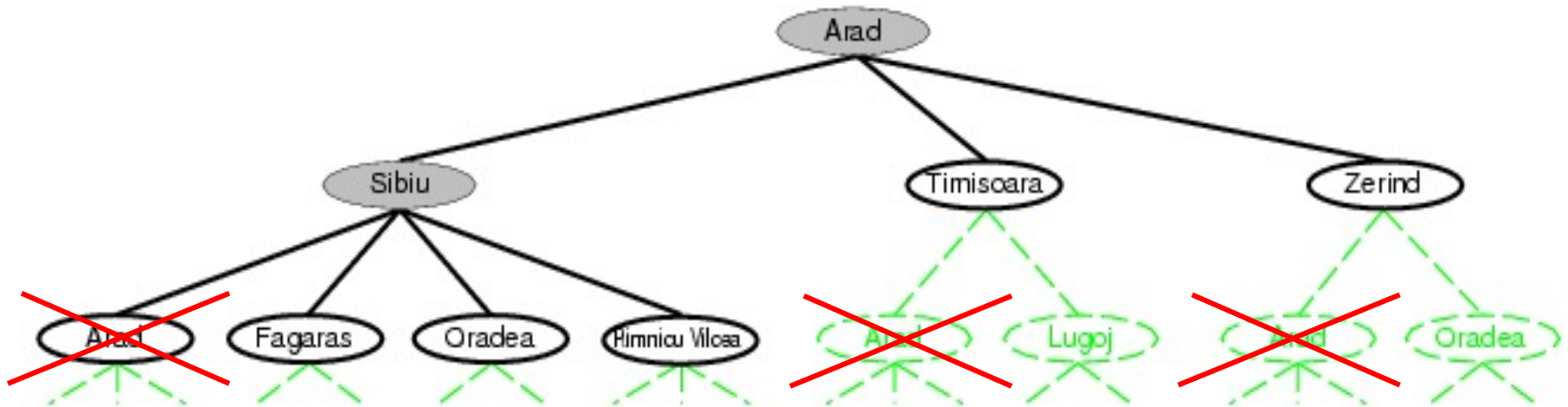
Graph search



Goal: Get from Arad to Bucharest with as little km as possible

Tree search

- ❑ Failure to detect / suppress repeated states can turn a linear problem into an exponential one!



- Very simple fix: avoid loopy paths
 - ❑ Not always necessary to do it explicitly, as their cost is always higher than the same path with the loop removed 😊

Graph search

```
function Graph-Search(problem, fringe) returns a solution, or failure
  fringe ← Insert(Make-Node(Initial-State(problem)), fringe);
  closed ← an empty set
  while (fringe not empty)
    node ← RemoveFirst(fringe);
    if (Goal-Test(problem, State(node))) then return Solution(node);
    if (State(node) is not in closed) then
      add State(node) to closed
      fringe ← InsertAll(Expand(node, problem), fringe);
    end if
  end
  return failure;
```

- By defaults, the graph search algorithm never expands a node twice (by closing nodes)

Informed search strategies

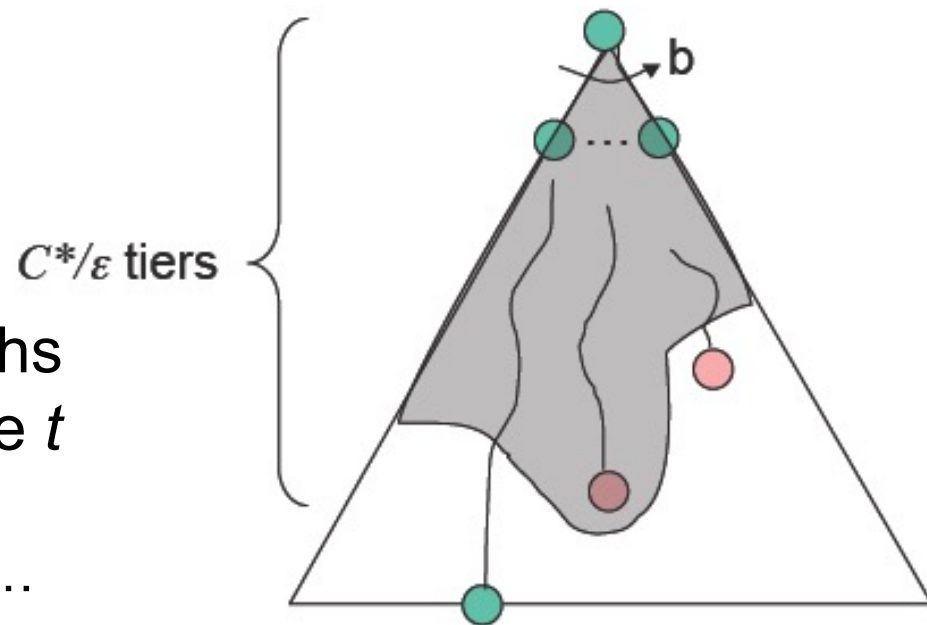
Introduction

Recall: uniform-cost search

- ❑ Task: Find the shortest/cheapest path from the start node to the goal
- ❑ Pick cheapest unexpanded node (node with minimum cost from the start node to it)

■ Problem

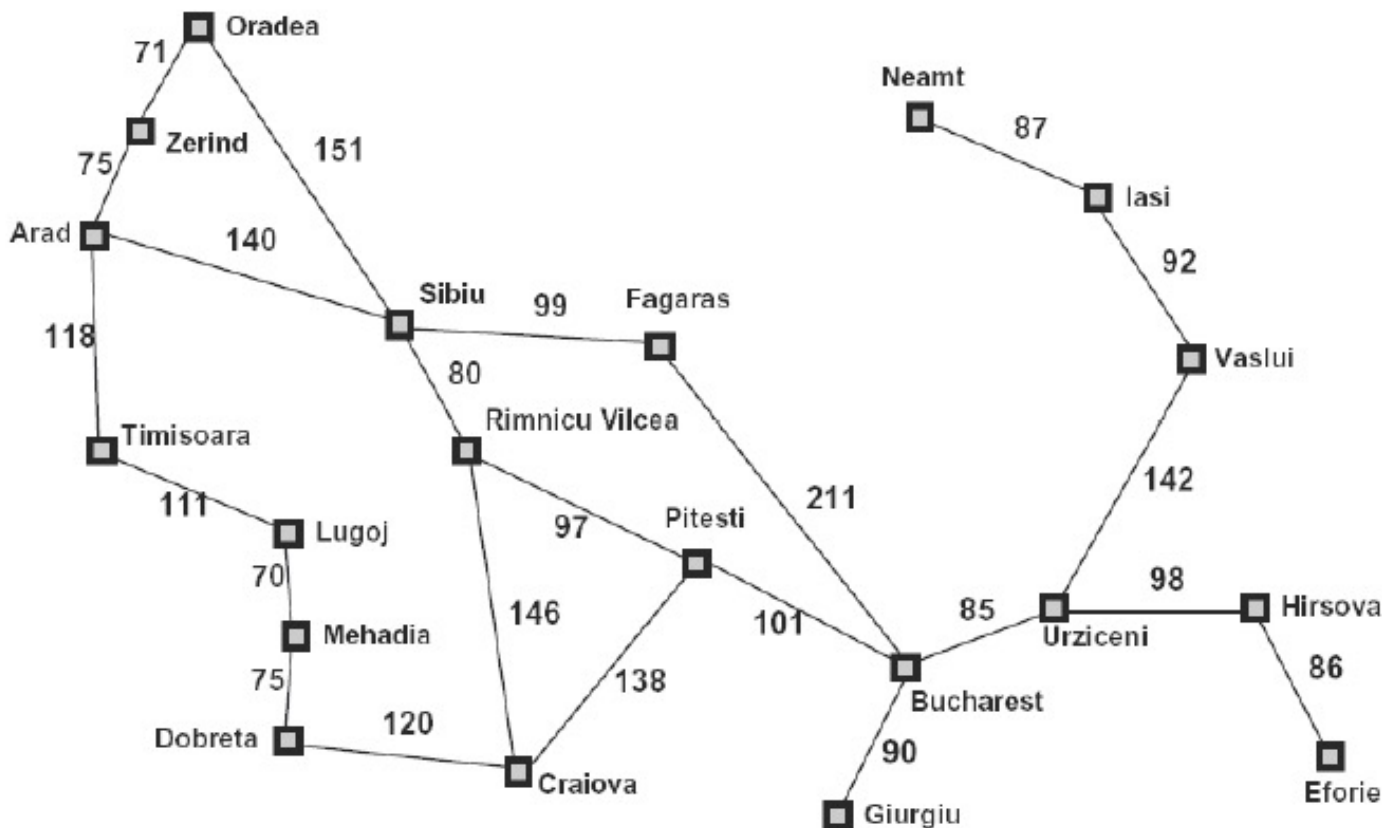
- ❑ Uniform-cost search only expands “cheap” paths based on their cost at time t
 - it pays no attention to the future (no long-term view)...



Example: Straight Line Distances

- Idea: take into account the problem-specific knowledge in order to make search more efficient

problem-specific knowledge



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

Definition of informed search strategies

- Different from **uninformed search strategies...**
 - (all algorithms presented during the last lecture, including uniform cost strategy are uninformed search strategies)...
- ... **informed search strategies** use **problem-specific knowledge**
 - *i.e.* knowledge beyond the definition of the problem itself
- Informed search strategies can find solutions **more efficiently** than uninformed search strategies

Informed search strategies

Best-first search

Best-first search

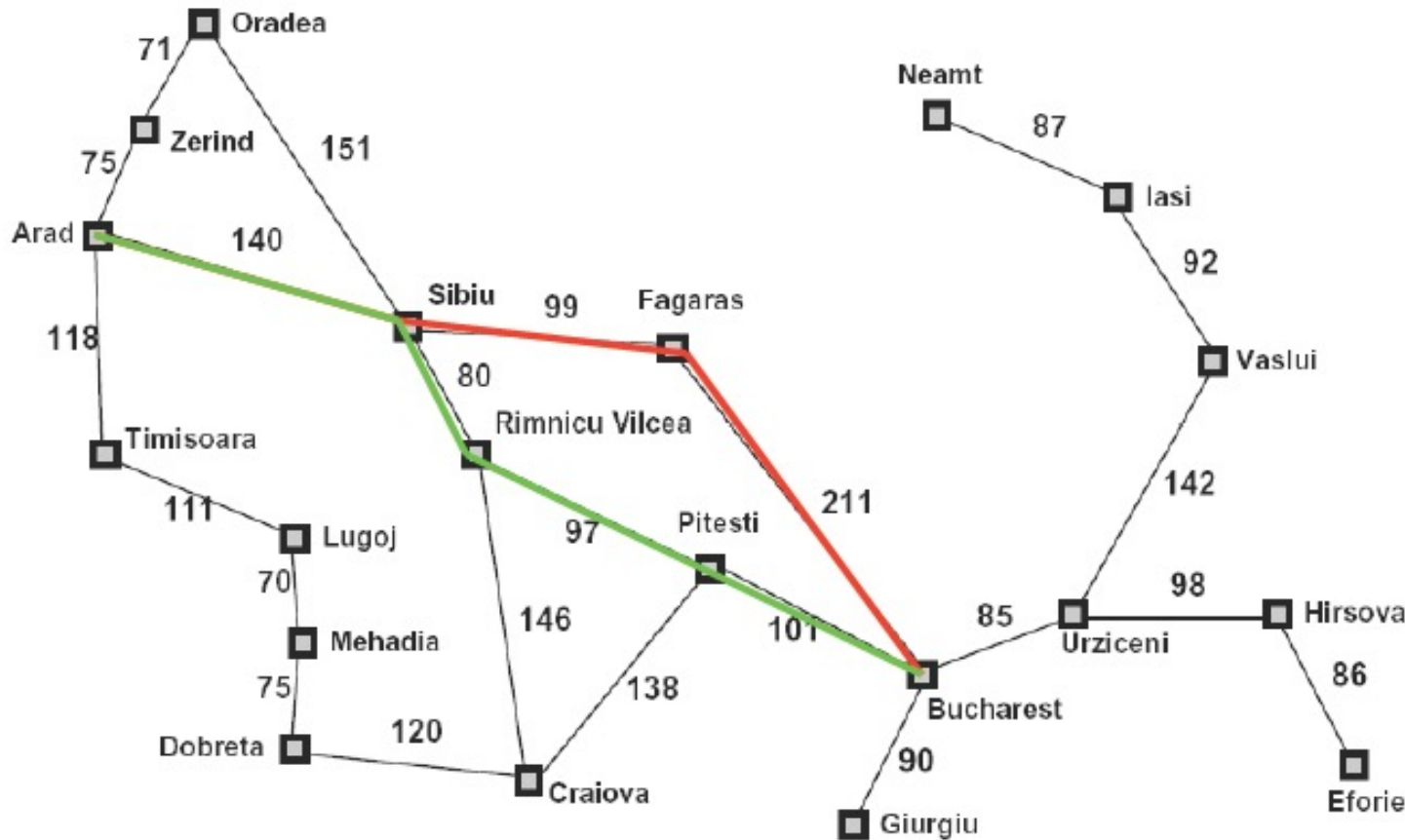
- Idea: use an **evaluation function** $f(n)$ for each node
 - estimate of "desirability", based on the problem-specific knowledge
 - Expand most desirable unexpanded node
- Nodes in the fringe are ordered in decreasing order of evaluation function value

Best-first search

problem-specific knowledge

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



- **Idea of best-first search:** using an evaluation function that makes the agent pick the green or red paths quicker
 - Using extra knowledge (compared to only the map contained in problem formulation):
 - Called problem-specific knowledge
 - Here, the problem-specific knowledge is the straight-line distances to Bucharest


Best-first search

- Special cases of best-first search:
 - Greedy best-first search
 - A^* search

Informed search strategies

Greedy best-first search

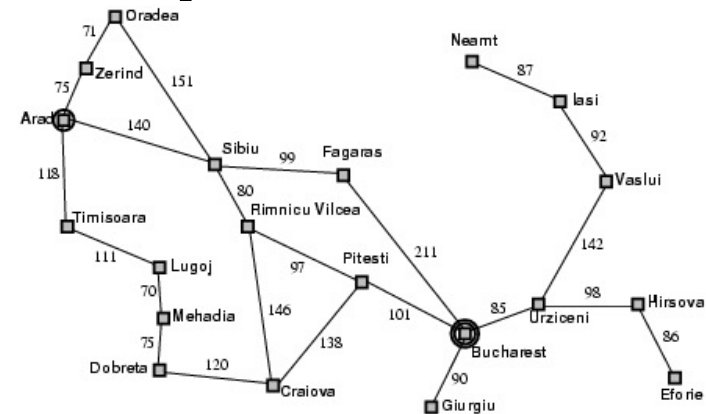
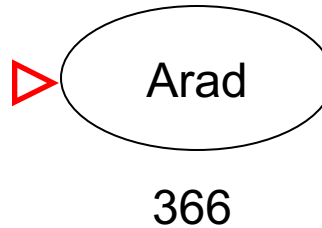
Greedy best-first search

- Evaluation function $f(n) = h(n)$ (**h**euristic)
= estimate of cost from n to *goal*
 - Estimate \neq real cost
- e.g., $h_{SLD}(n)$ = straight-line distance from n to Bucharest
-  □ Greedy best-first search expands the node that **is estimated** to be closest to goal
 - Not necessarily the closest :-/

Greedy best-first search example

Straight-line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178 176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98 100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



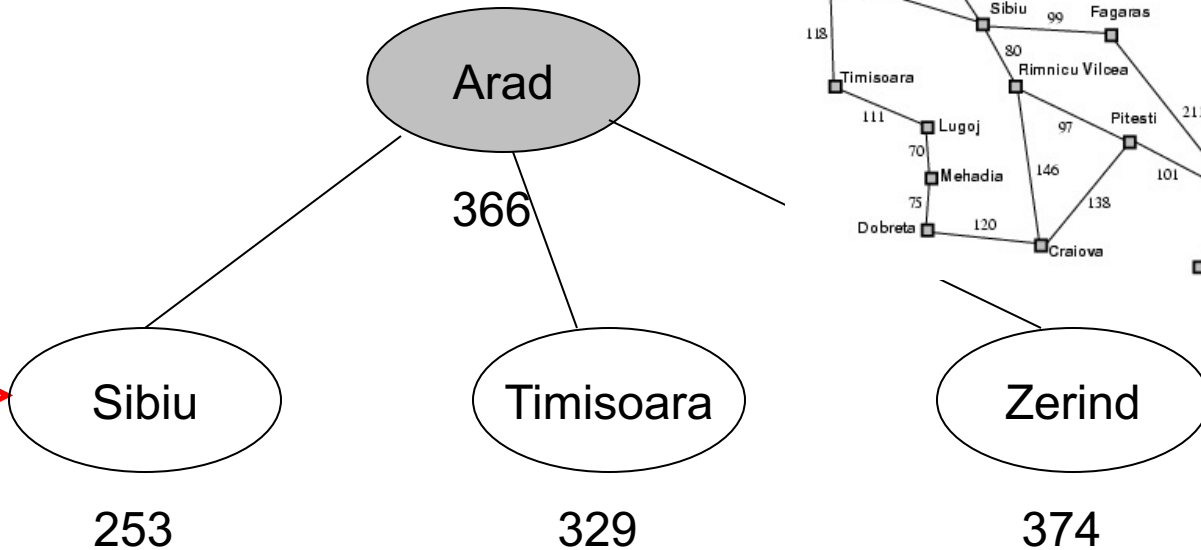
Greedy best-first search example

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

176

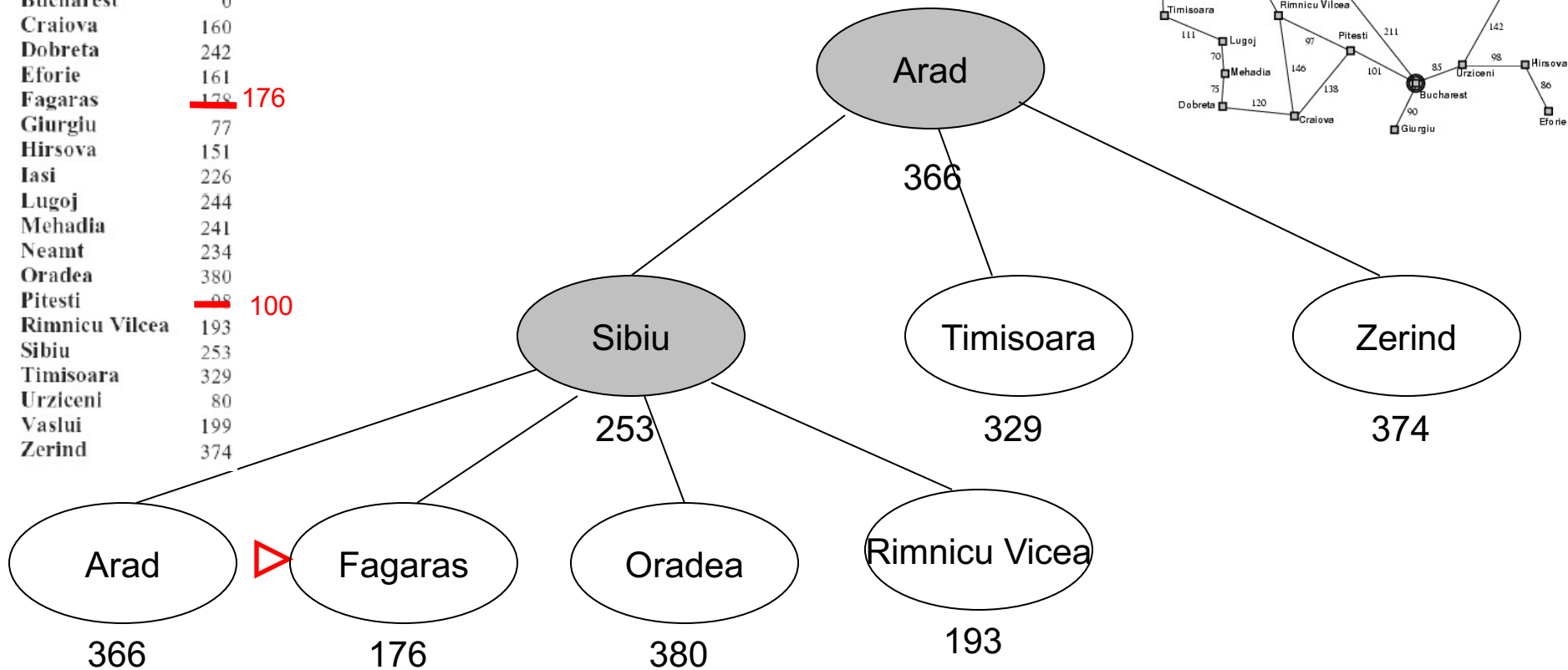
100



Greedy best-first search example

Straight-line distance
to Bucharest

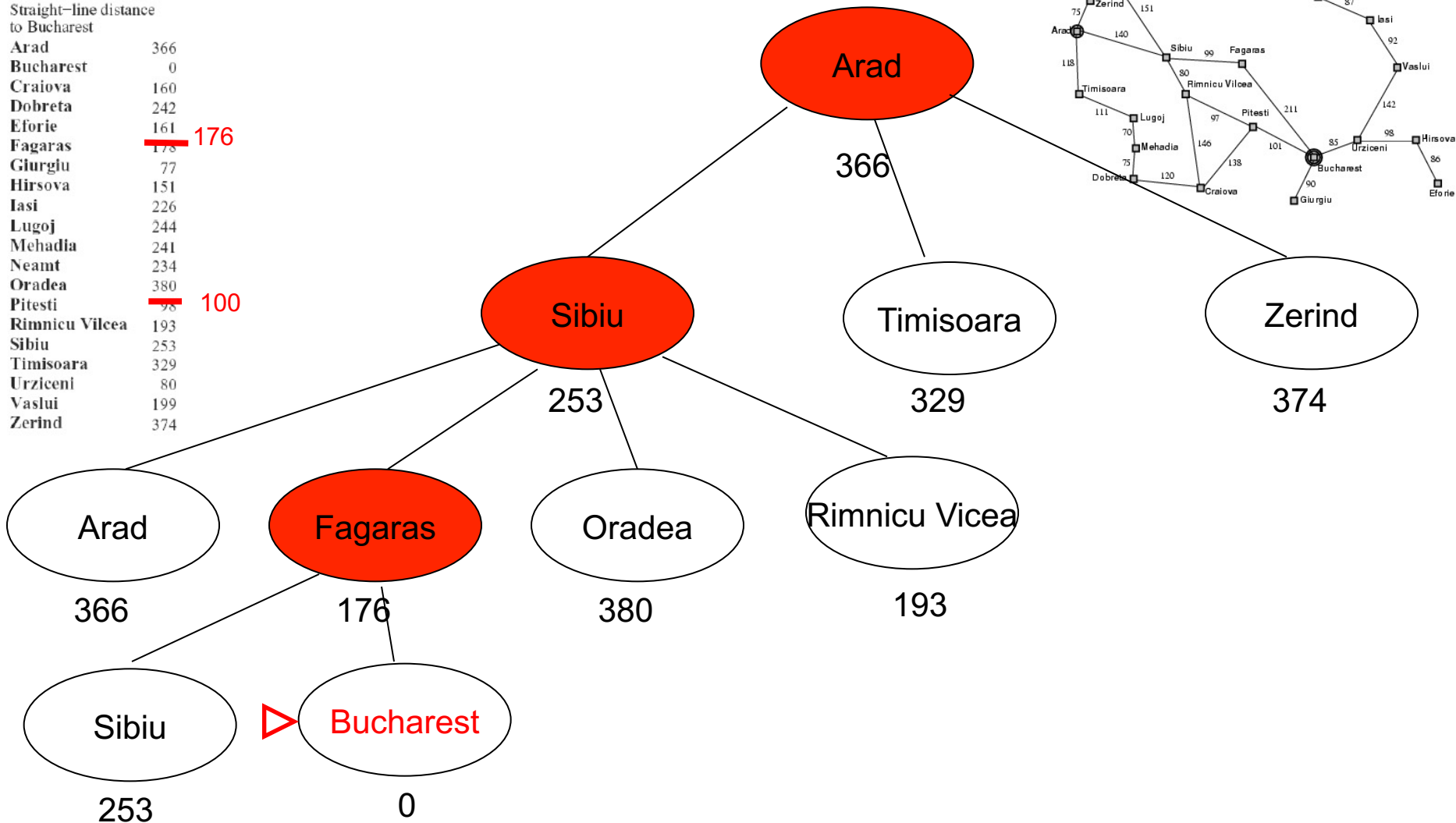
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178 176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98 100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Greedy best-first search example

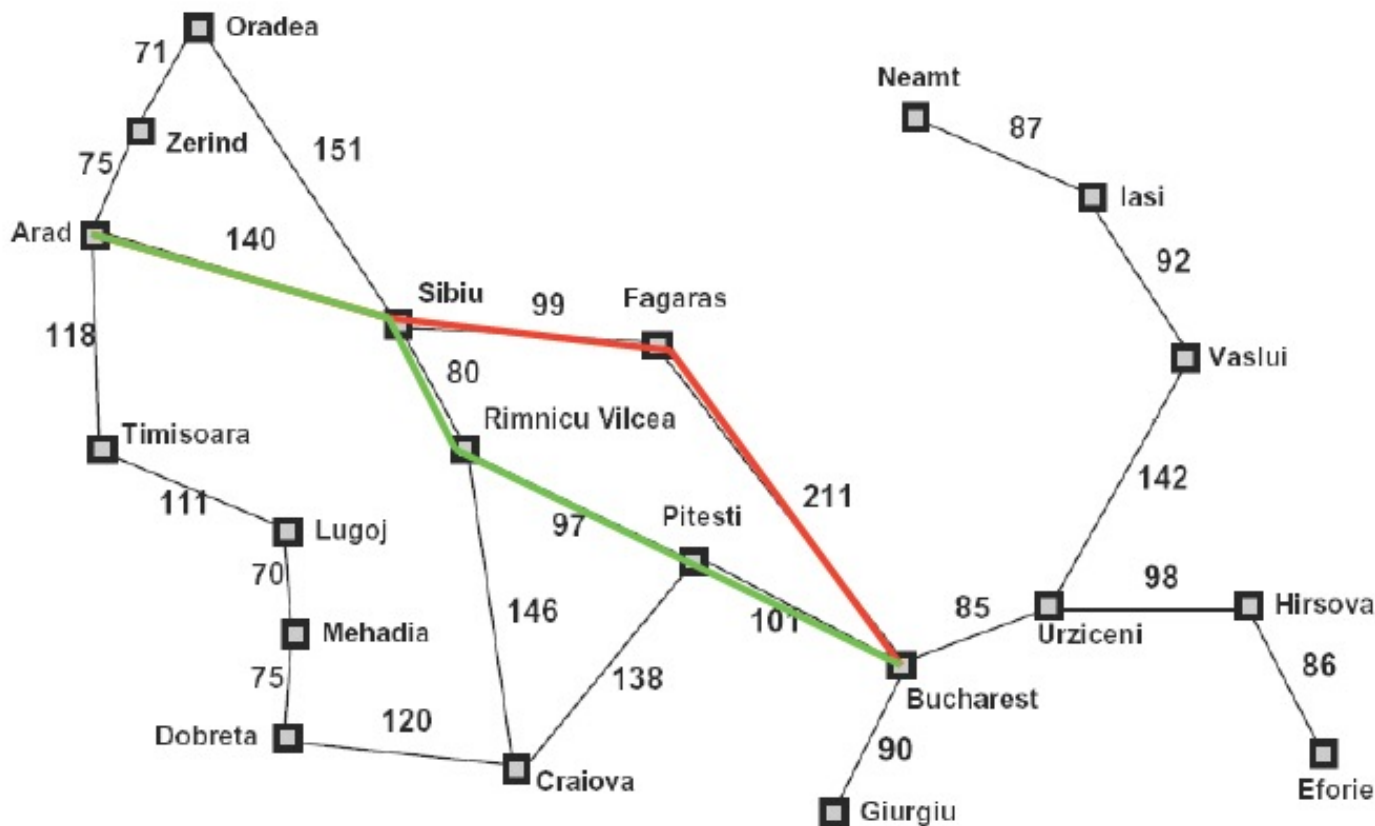
Straight-line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178 176
Glurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98 100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Greedy best-first search example

- Solution found by the greedy best-first search
 - The red trajectory, not optimal... but not too bad...
 - Real cost = $140 + 99 + 211 = 450$

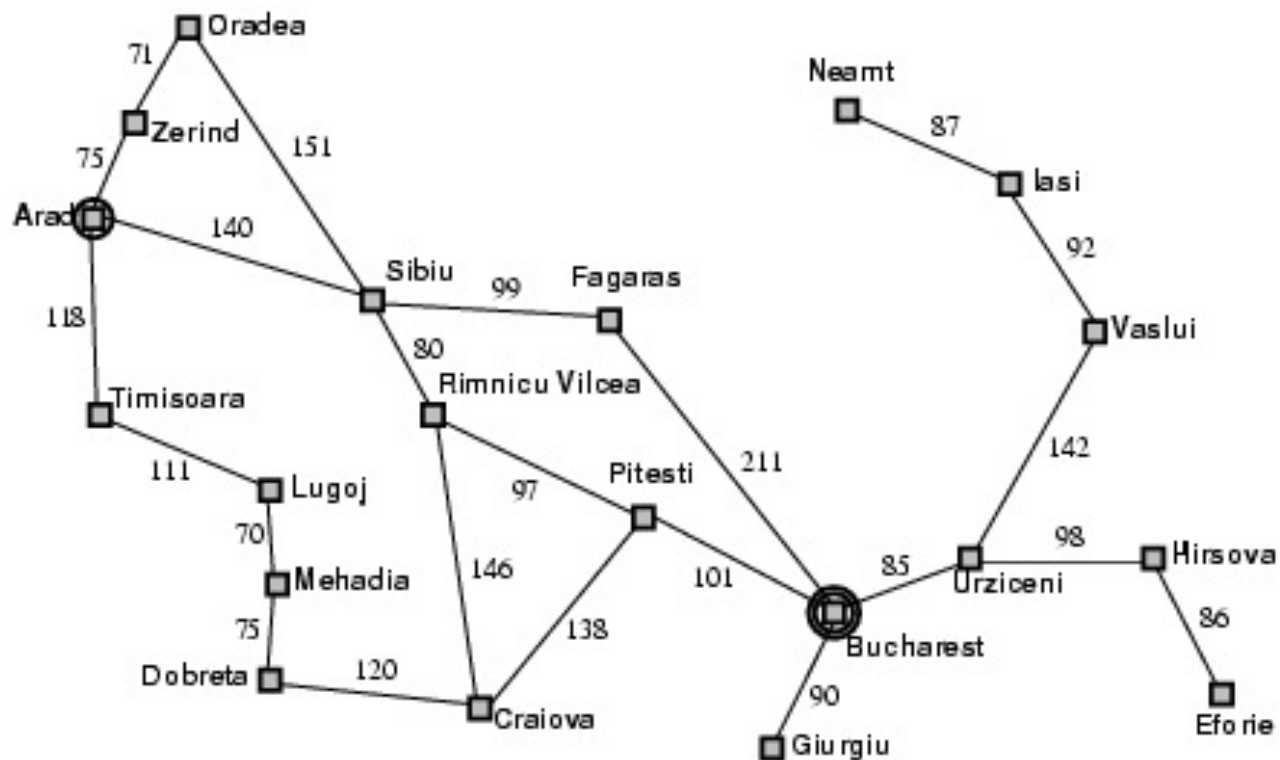


Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176 176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98 100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

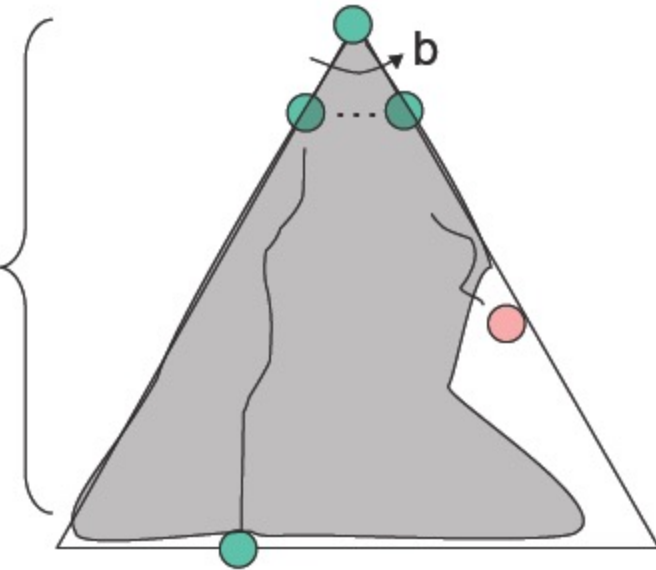
Greedy best-first search example

- Is greedy best-first search complete in general?
- Exercise: We now want to go from Iasi to Fagaras.
 - Will the greedy best-first search succeed to reach the goal?
 - Answer:



Greedy Best-First Search (for trees)

- ❑ Complete? No – can get stuck in local minima and plateau (due to loops)
 - But, it is complete in **finite** search spaces
- ❑ Time? $O(b^m)$, but a good heuristic can give dramatic improvement
- ❑ Space? $O(b^m)$ -- keeps all nodes in memory
- ❑ Optimal? No



- What do we need to do to improve this algorithm
 - ⇒ Improve the heuristic, by estimating the cost **to the goal**
 - ⇒ A* search

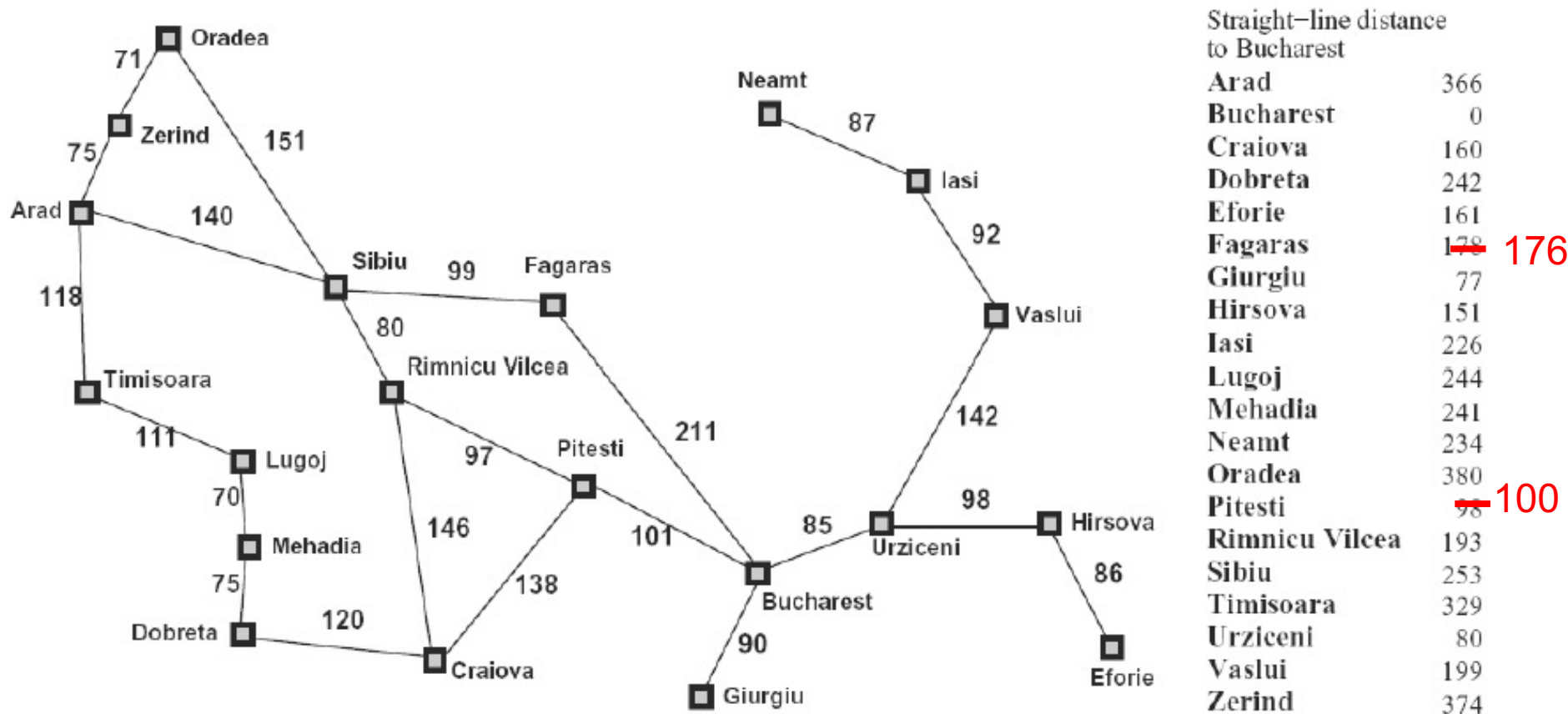
Informed search strategies

Presentation of A* search

A* search

- ❑ Idea: Expand unexpanded node with lowest evaluation value, using an evaluation value that:
 - is closer to the real cost value
 - avoids expanding paths that are expensive from the beginning
 - focuses on paths that seem the most promising
- ❑ In short, A* search tries to make the best use of all the available information
 - The information in the problem formulation (used by uniform-cost search)
 - The problem-specific knowledge (used by greedy-first 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
- ❑ Nodes in the fringe are ordered according to $f(n)$
 - Because it is a best-first algorithm

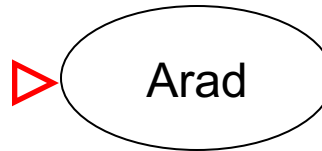
A* search example



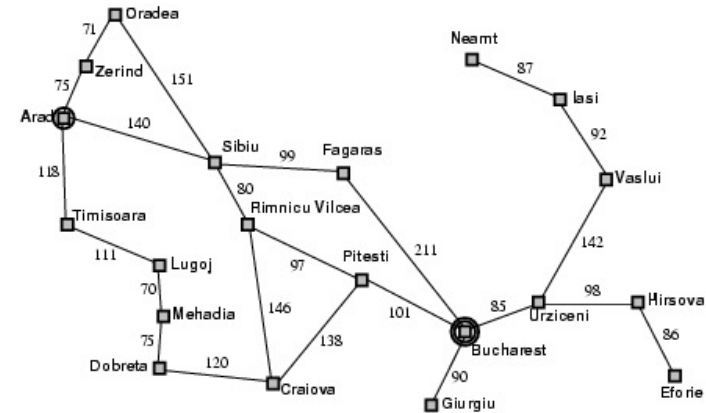
A* search example

Straight-line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	<u>178</u> 176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	<u>98</u> 100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



$$366 = 0 + 366$$



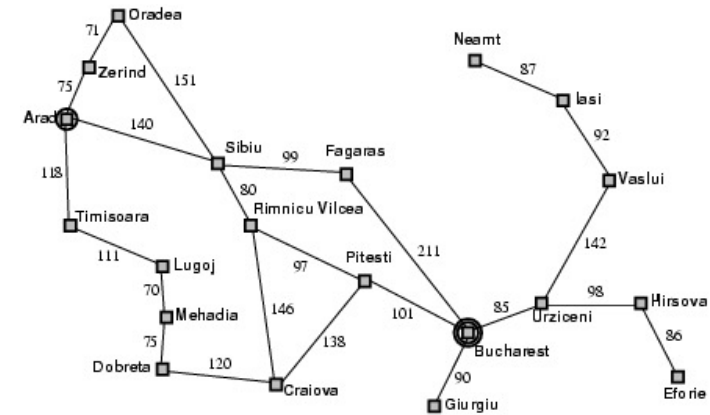
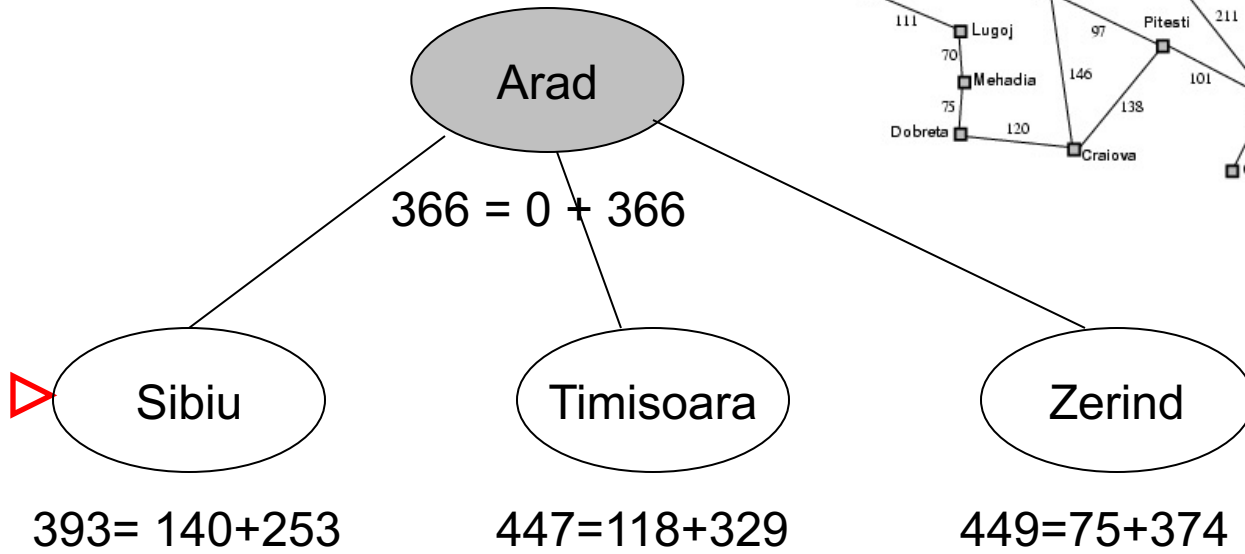
A* search example

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	101
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

176

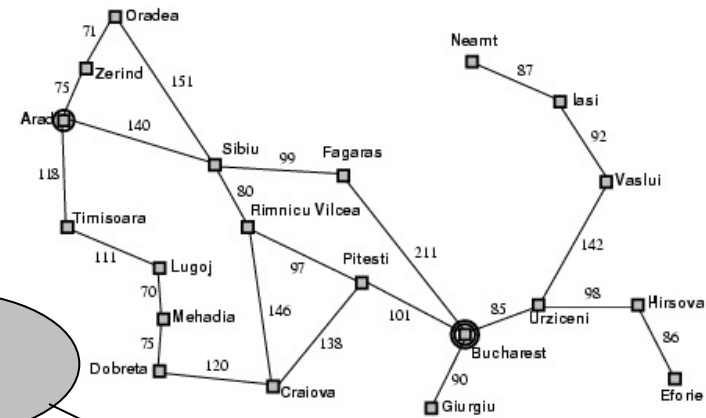
100



A* search example

Straight-line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	<u>176</u> 176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	<u>98</u> 100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



$$366 = 0 + 366$$

Sibiu

Timisoara

Zerind

$$393 = 140 + 253$$

$$447 = 118 + 329$$

$$449 = 75 + 374$$

Arad

Fagaras

Oradea

▶ Rimnicu Vicea

$$646 = 280 + 366$$

$$415 = 239 + 176$$

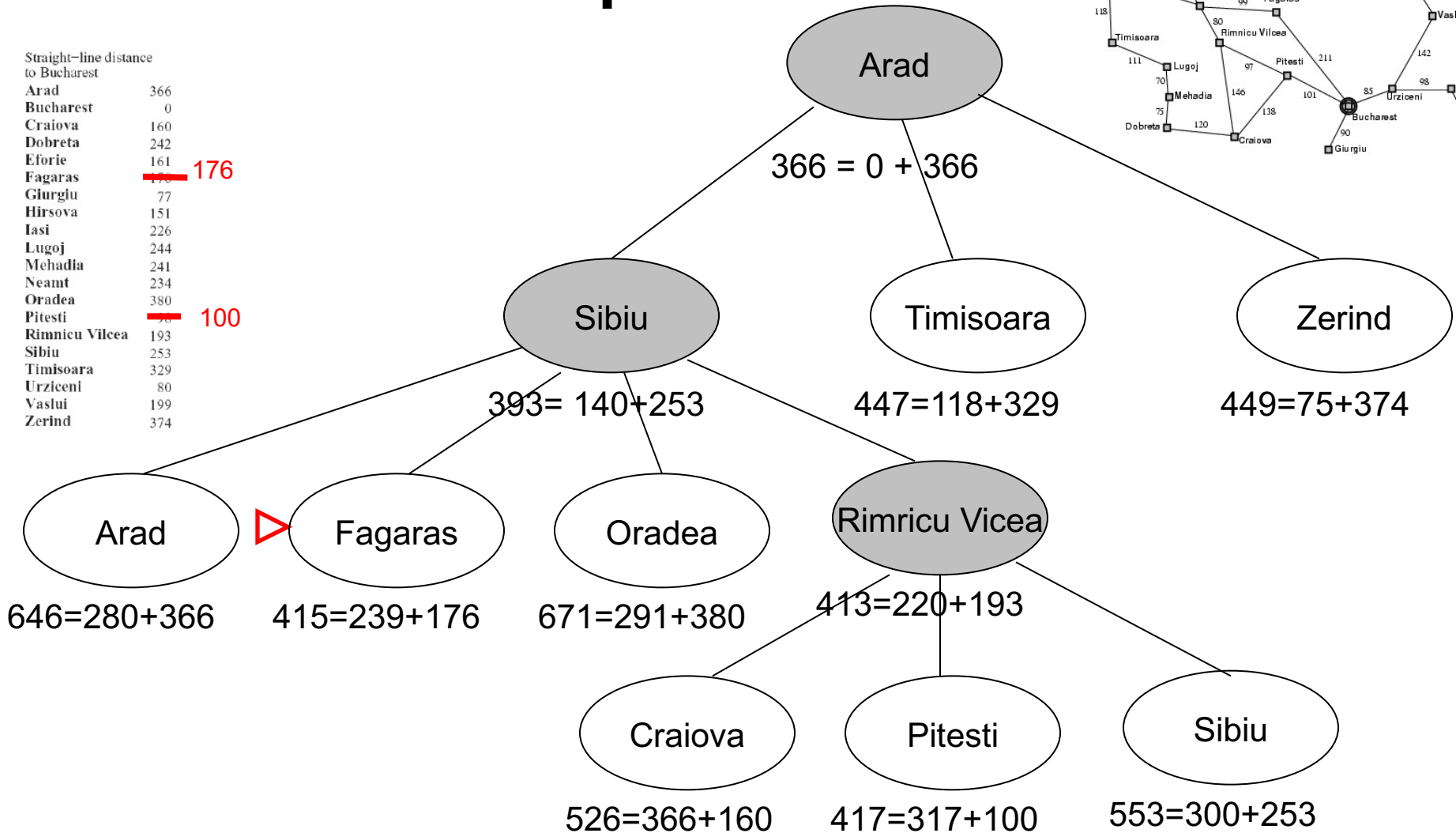
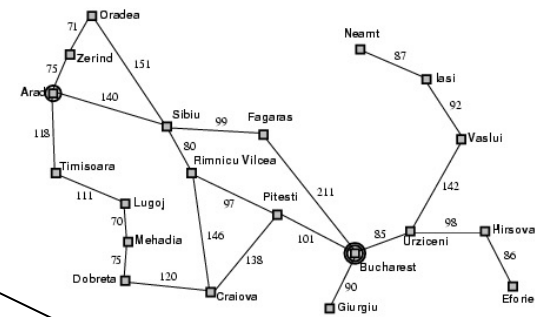
$$671 = 291 + 380$$

$$413 = 220 + 193$$

A* search example

Straight-line distance
to Bucharest

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



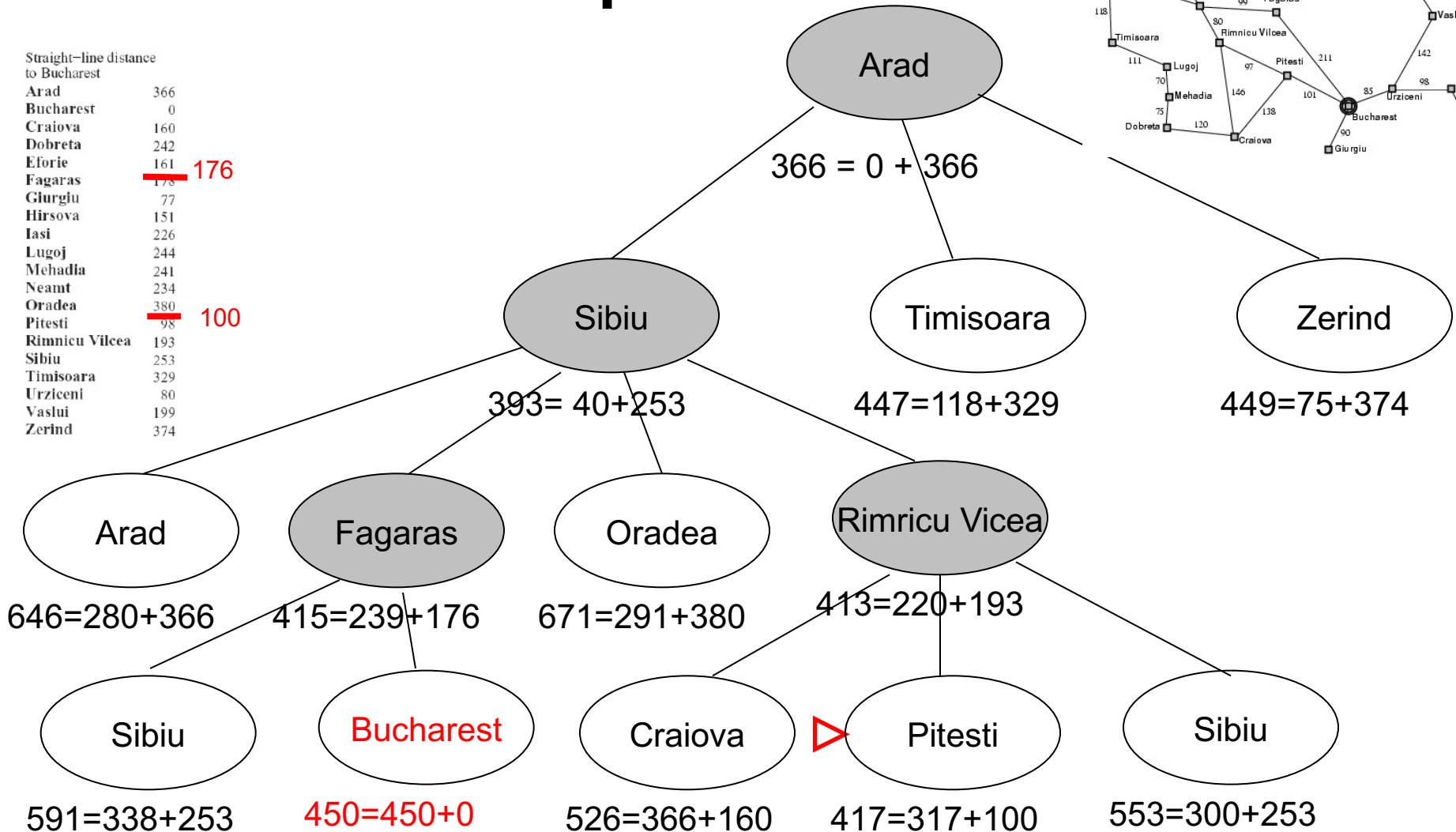
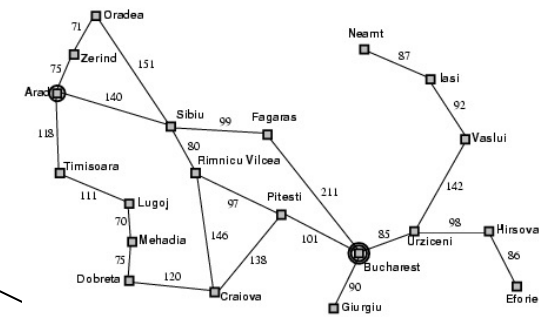
A* search example

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

176

100

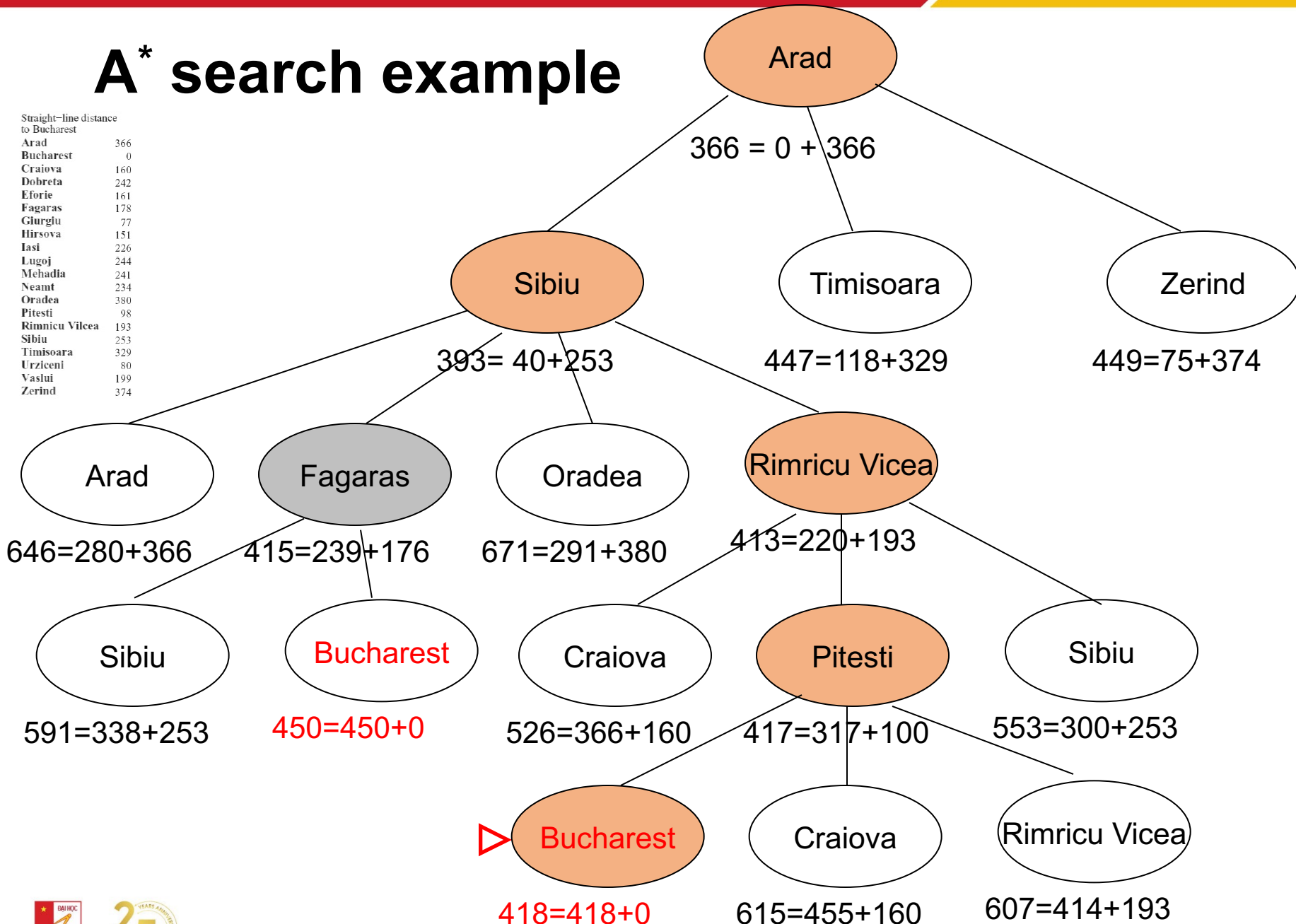


We reached the goal with $C=450$, but there is a cheaper node in the fringe (Pitesti) that is unexpanded \Rightarrow A* search will expand it

A* search example

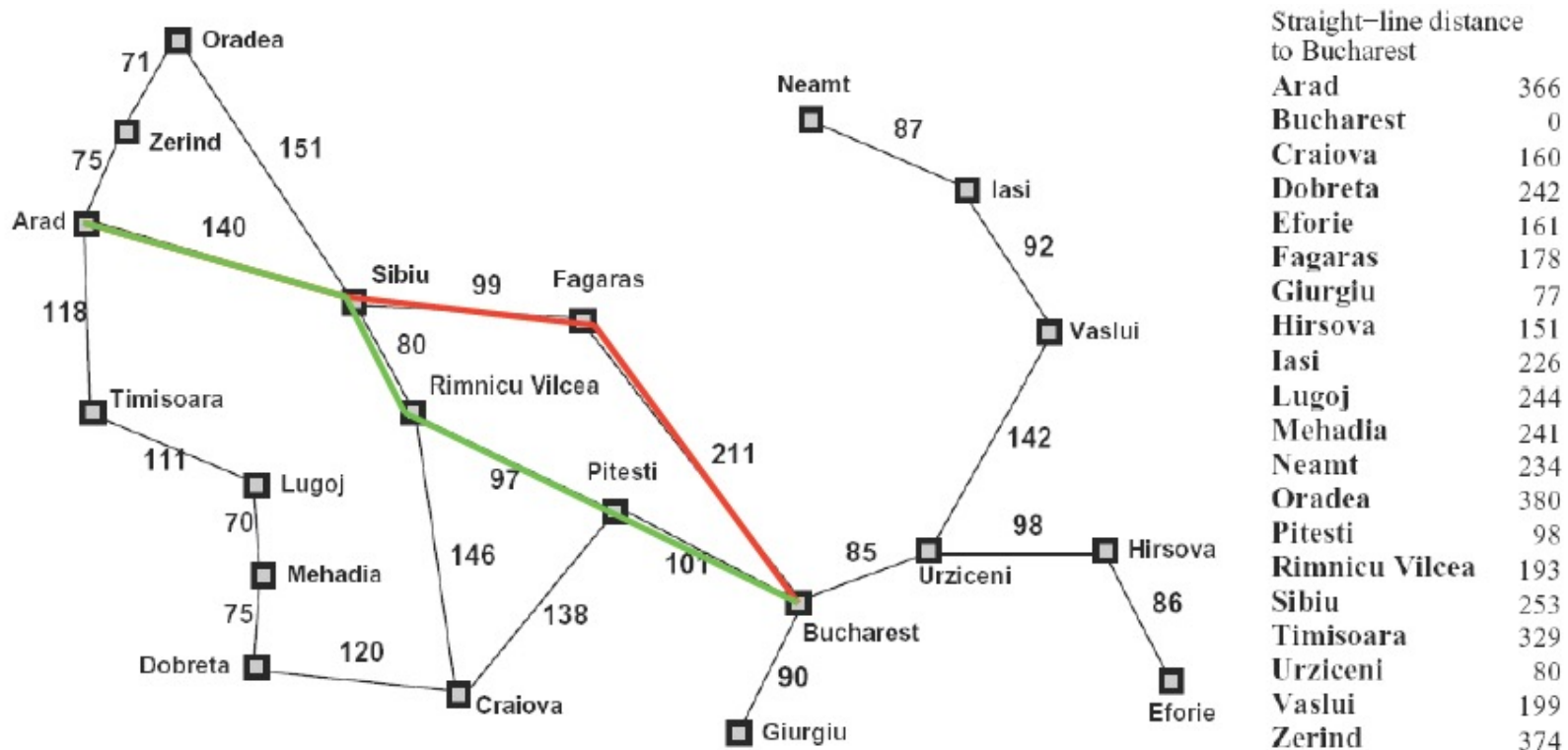
Straight-line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Glurgiu	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



A* search example

- Solution found by the A* search
 - The green trajectory, optimal!
 - Real minimal cost $C^* = 140 + 80 + 97 + 101 = 418$ (better than the greedy best-first solution, in red, with cost 450)



What about completeness/optimality of A*?

- If the state space is finite and we avoid repeated states and all costs are $> \epsilon$, then the search is **complete**
 - If the state space is infinite, the search is in general not complete
 - If the state space is finite and we do not avoid repeated states, the search is in general not complete

- But, even under all these conditions, it might **not** be optimal
 - ⇒ Need some “good properties” of the heuristic to ensure optimality

Informed search strategies

About heuristic quality

Admissible heuristics

□ Recall of definitions:

- Evaluation function $f(n) = g(n) + h(n)$
 - $g(n)$ = cost so far to reach n
 - $h(n)$ = estimated cost from n to goal = **heuristic**
- $f(n)$ = estimated total cost of path through n to goal
- A* search is a best-first search where the nodes in the fringe are ordered according to $f(n)$

□ Let $h^*(N)$ be the **true** cost of the optimal path from N to a goal node

□ Heuristic $h(N)$ is **admissible** if:

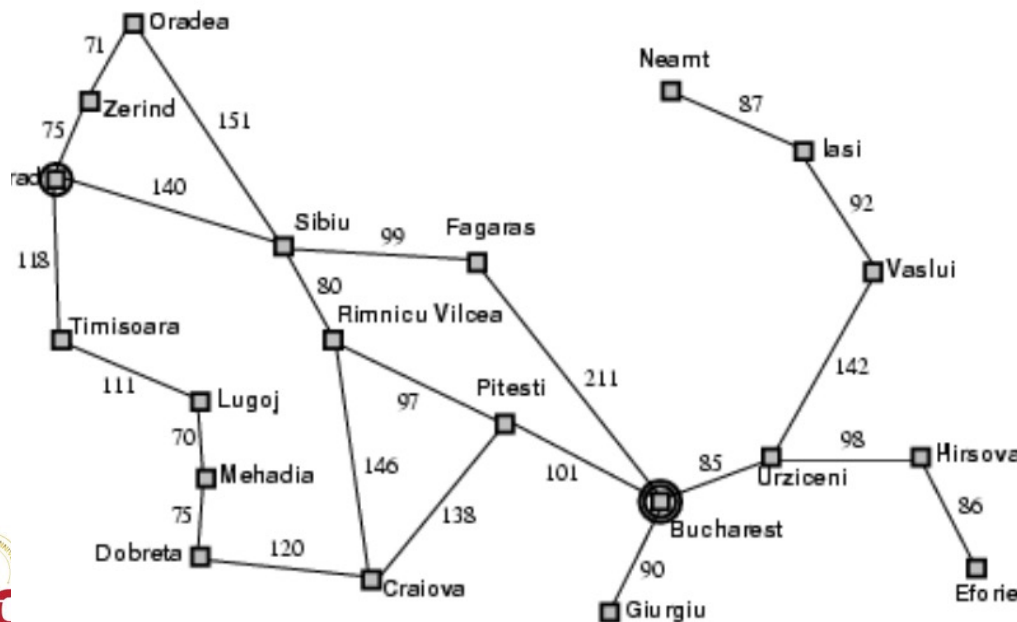
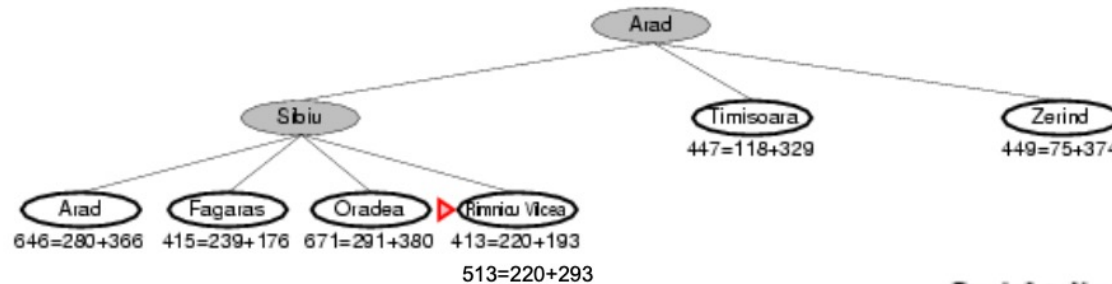
$$0 \leq h(N) \leq h^*(N)$$

□ An admissible heuristic is always **optimistic**

□ A* is **admissible** if it uses an admissible heuristic, **and** $h(\text{goal}) = 0$

Admissible heuristics

- Example of A* search with inadmissible heuristic



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	101
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199

→293

Admissible heuristics

Example: The 8-puzzle

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan (L_1) distance
 - (i.e. # of squares from desired location of each tile)
- Question 1: Are these two heuristics admissible? (no need to prove it)
 - Answer:
- Question 2: Is A* search using these two heuristics admissible?
 - Answer:
- Question 3:
 - $h_1(S) = ?$
 - $h_2(S) = ?$

5	4	
6	1	8
7	3	2

S Start State

1	2	3
8		4
7	6	5

Goal State

Admissible heuristics

□ Admissible heuristics

- Heuristics that **never** over-estimate the cost to reach the goal (optimistic)

□ Admissible heuristics can be derived from the exact solution cost of a **relaxed** version of the problem:

- Relaxed 8-puzzle for h_1 (number of misplaced tiles): a tile can move anywhere
- Relaxed 8-puzzle for h_2 (total Manhattan (L_1) distance): a tile can move to any adjacent square

□ Any optimal solution in the **original** problem is, by definition, also a solution in the **relaxed** problem

- The optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

Admissible heuristics

Example: The 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan (L_1) distance
(i.e. # of squares from desired location of each tile)

Both are admissible, but which one is best? → **dominance**

5	4	
6	1	8
7	3	2

Start State

1	2	3
8		4
7	6	5

Goal State

Heuristic **dominance**

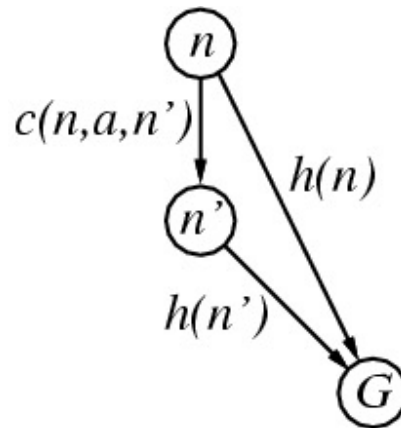
- If $h_j(n) \geq h_i(n)$ for all n that do not meet the goal, then h_j **dominates** h_i (better for search)
- **Question:** which heuristic dominates the other in the 8-puzzle problem?
 h_2 or h_1 ?
 - (No need to prove your answer)
 - Answer:
- In general, it is not possible to prove theoretically the dominance of one heuristic over another
 - We have to run numerous instances of the problem observe it in practice
 - Example: 1200 random problem instances of the 8-puzzle with solution lengths from 2 to 24 (average values over 100 instances / d)

How to invent good heuristics?

- ❑ One needs to distinguish between the two cases: tree and graph search
- ❑ **For tree search**
 - Admissibility is **not required** for A* **tree search completeness**
 - A* always terminates with a solution path (even if h is not admissible) if
 - costs on arcs are $> \epsilon > 0$
 - branching factor is finite
 - Admissibility is sufficient for A* **tree search optimality**
 - And, therefore, admissibility is sufficient for A* tree search to be complete
- ❑ **For graph search**
 - A* **graph search** is **complete** if
 - costs on arcs are $> \epsilon > 0$
 - The graph is **finite**
 - Admissibility is not sufficient for A* **graph search optimality**
 - Unless we can re-open the closed nodes (see algorithm on slide 8)
 - If we can re-open closed nodes, then admissibility is **sufficient** for A* **graph search optimality**
 - If we can't re-open the node, then we can guarantee optimality by requiring **consistency** property for $h(n)$

Consistent heuristics

- Consistent (or monotonous) heuristics verify the triangular inequality
 - $h(n) \leq c(n, a, n') + h(n')$
 - A heuristic $h(n)$ is consistent if, for every node n and every successor n' of n generated by any action a , the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n'

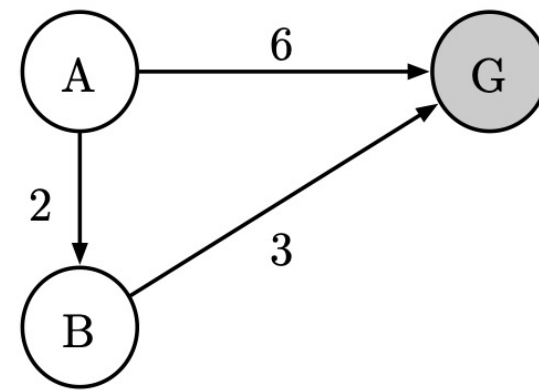


- If h is consistent and $h(\text{goal})=0$ then h is admissible
 - Proof: by induction of distance from the goal (see the reference book)
 - In practice, most admissible heuristics are also consistent (but not all)

Optimality

- ❑ **Consistency** is required for optimality of A* graph search **without** re-opening closed nodes
 - So that its evaluation function $f(n)$ is non-decreasing along any path (monotonic)
→ Optimality
- ❑ **Summary:** to reach **optimality**
 - A* Tree search
 - Admissibility is enough
 - Graph search, with re-opening closed nodes
 - Admissibility is enough
 - Graph search, without re-opening closed nodes (usual graph search)
 - Requires consistency
- ❑ For more explanations and proofs, look in the reference book
 - *Artificial Intelligence – A Modern Approach*. Stuart Russell and Peter Norvig. Prentice Hall, **THIRD EDITION**, pages 92-99

Exercise



	$h(A)$	$h(B)$	$h(G)$
I	4	1	0
II	5	4	0
III	4	3	0
IV	5	2	0

- Let us consider the following problem
 - with A the start node and G the goal node

 **Directed graph!**

- With 4 possible heuristics: I, II, III, IV

- Which ones are admissible?
- Which ones are consistent?
- Which ones dominate the others (pair by pair)?

- **Recall:**

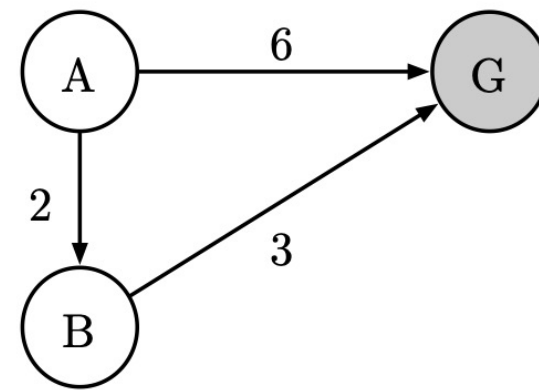
- $h(n)$ = **estimated** cost from n to goal
- In the graph are shown the **real costs** between nodes
- Heuristic $h(n)$ is **admissible** if:
 $0 \leq h(n) \leq h^*(n)$, where $h^*(n)$ is the **true** cost of the optimal path from n to a goal node
- A heuristic $h(n)$ is **consistent** if, for every node n and every successor n' of n generated by any action a :

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

- For one heuristic to **dominate** another, **all** of its values must be greater than or equal to the corresponding values of the other heuristic.

Exercise

- Exercise: for each line, circle the good answers in the table above

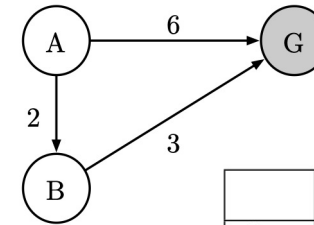


	Admissible?		Consistent?	
I	Yes	No	Yes	No
II	Yes	No	Yes	No
III	Yes	No	Yes	No
IV	Yes	No	Yes	No

	$h(A)$	$h(B)$	$h(G)$
I	4	1	0
II	5	4	0
III	4	3	0
IV	5	2	0

- Answer for admissibility:

Exercise



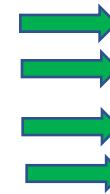
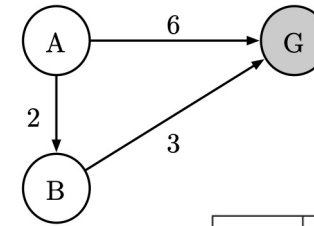
	$h(A)$	$h(B)$	$h(G)$
I	4	1	0
II	5	4	0
III	4	3	0
IV	5	2	0

- Exercise: for each line, circle the good answers in the table above

	Admissible?		Consistent?	
I	Yes	No	Yes	No
II	Yes	No	Yes	No
III	Yes	No	Yes	No
IV	Yes	No	Yes	No

- Answer for consistency:

Exercise



	$h(A)$	$h(B)$	$h(G)$
I	4	1	0
II	5	4	0
III	4	3	0
IV	5	2	0

	Admissible?		Consistent?	
I	Yes	No	Yes	No
II	Yes	No	Yes	No
III	Yes	No	Yes	No
IV	Yes	No	Yes	No

□ Answer for dominance:

Heuristic quality evaluation: useful definition

□ Effective branching factor b^*

- Is the branching factor that a **uniform** tree of depth d should have in order to contain $N+1$ nodes (N being the # of nodes expanded by the algorithm)
 - Uniform tree: all nodes have the same number of children

$$N + 1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$$

- A well-designed heuristic would have a value of b^* close to 1
- b^* is fairly constant for sufficiently hard problems
 - Can thus provide a good guide to the heuristic's overall usefulness.
 - Experimental measurements of b^* on a small set of problems can provide a good guide to the heuristic's overall usefulness

Heuristic quality: useful definition

■ Example: 8-puzzle problem

□ Recall:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan (L_1) distance
 - (i.e. # of squares from desired location of each tile)

- in this problem, both h_1 and h_2 are admissible, but h_2 dominates h_1

d	Search Cost			Effective Branching Factor		
	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

Note for the capstone project

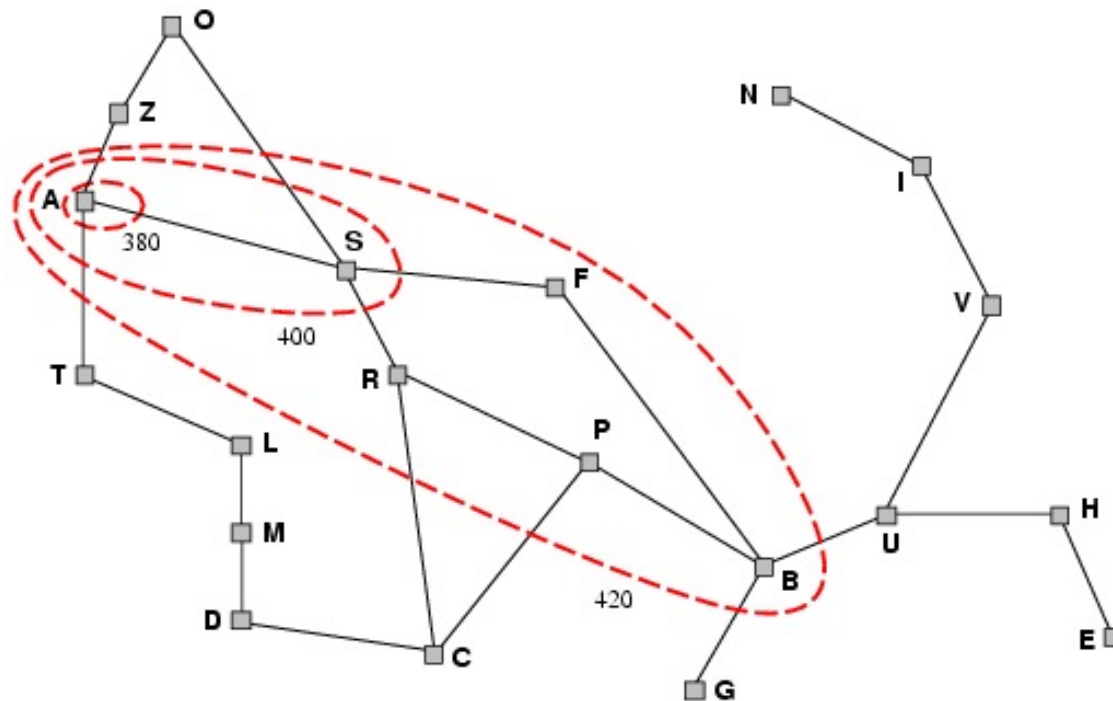
- For students who need to use search algorithms:
 - When possible, using informed search strategies should always give better (or same) results than un-informed search strategies
 - When possible, in the capstone project, please compare at least 1 un-informed search strategy with un-informed search strategies
 - In many cases, you will need to invent your own heuristic:
 - Pay attention to the “good properties” above (admissibility, consistency, dominance, effective branching factor close to 1) to choose the best heuristics you can...
 - ... AND...
 - ... **Explain** your choice of heuristics !!!
 - You can use variants of A^* , even if we did not see them in the lectures
 - In that case, you need to explain the variant’s algorithm in your presentation and report

Informed search strategies

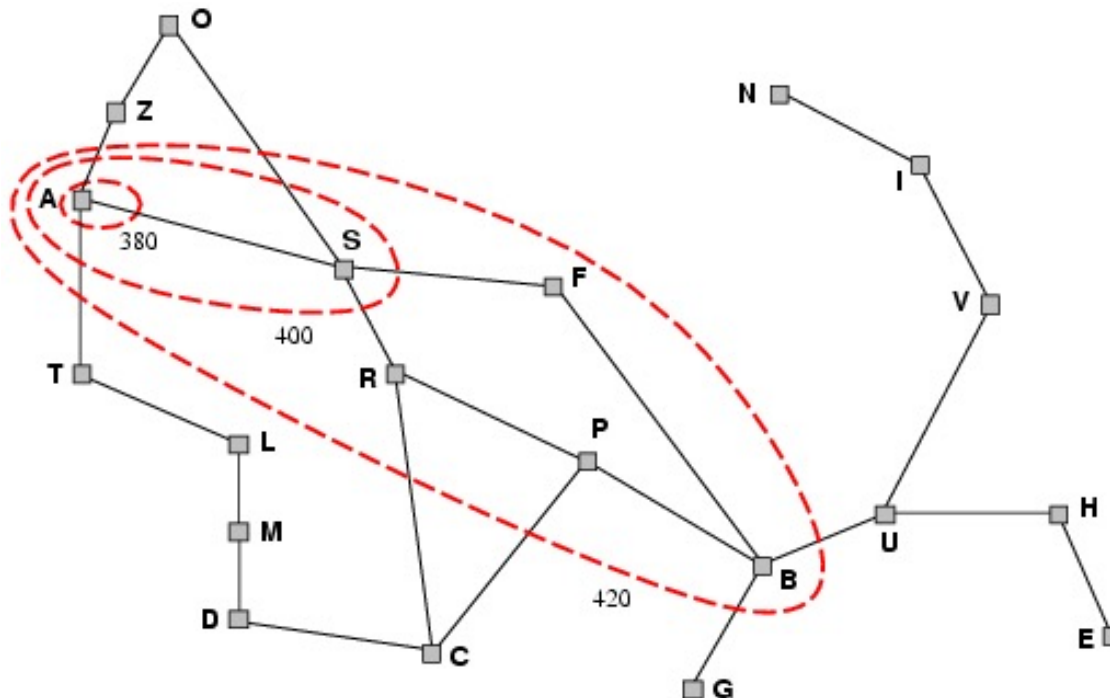
Contours of search

Contours of A* Search

- 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 $f_i < f_{i+1}$



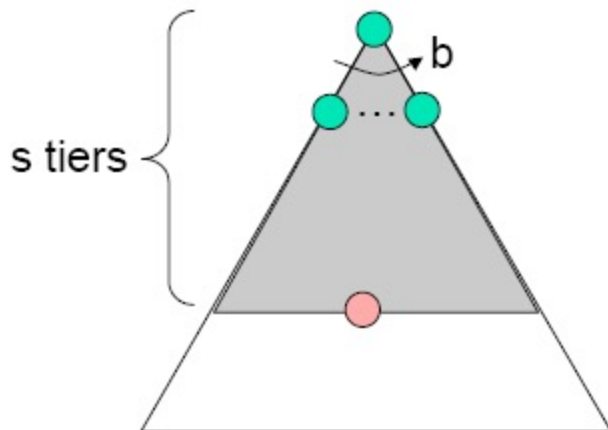
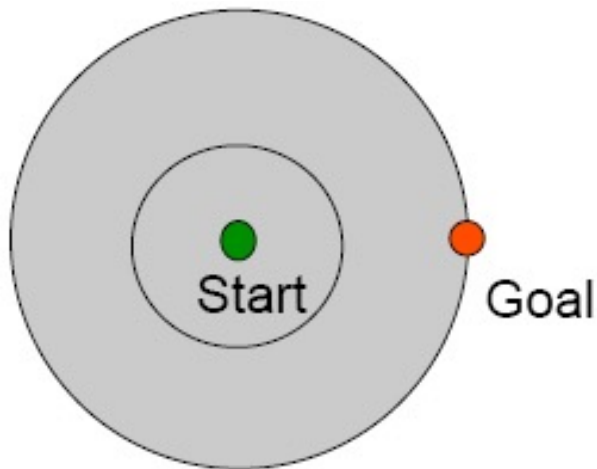
Contours of A* Search



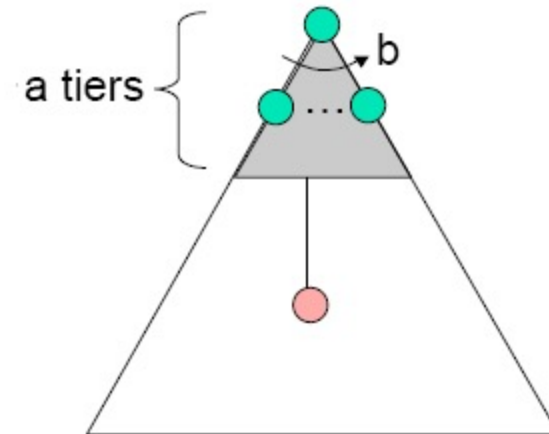
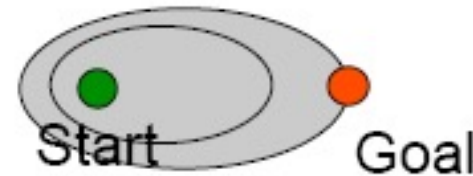
- With uniform-cost ($h(n) = 0$), contours will be circular
- With good heuristics, A* contours will be focused around optimal path
 - A* will expand all nodes with cost $f(n) < C^*$ (where C^* is the cost of the optimal solution)

Comparison of Uniform Cost and A*

- Uniform-cost expands in all directions



- A* expands mainly toward the goal, but enlarges its contours to ensure optimality



Informed search strategies

A* search evaluation

A* search, evaluation

- Completeness: YES (if the state space is finite and we avoid repeated states and all costs are $> \epsilon$ – see slide 36)
 - Since bands of increasing f are added

A* search, evaluation

- ❑ Completeness: YES (under certain conditions)
- ❑ Time complexity:
 - Number of nodes expanded is exponential in the length of the solution (path)

A* search, evaluation

- ❑ Completeness: YES (under certain conditions)
- ❑ Time complexity: exponential with path length
- ❑ Space complexity:
 - It keeps all generated nodes in memory
 - Hence space is the major problem here, not time

A* search, evaluation

- Completeness: YES
- Time complexity: exponential with path length
- Space complexity: exponential with path length
- Optimality: YES (if h is consistent)
 - Cannot expand f_{i+1} until f_i is finished.
 - A* expands all nodes with $f(n) < C^*$
 - A* expands some nodes with $f(n) = C^*$
 - A* expands no nodes with $f(n) > C^*$

Also *optimally efficient* (not including ties)

Main drawbacks of A^* , and A^* variants

- ❑ The time and space complexities of A^* are **big** in practice
- ❑ To reduce **time** complexity:
 - A^* is optimally efficient, *i.e.* it is quickest to find an optimal (cheapest) solution
 - But, it takes time
 - So, there are variants of A^* that find suboptimal solutions quicker
 - Possibility to design heuristics that are more accurate but not strictly admissible
 - N.B. In any case, the use of a good heuristic still provides enormous savings compared to the use of an uninformed search.
- ❑ To reduce **space** complexity
 - Its huge space complexity is A^* 's main drawback
 - For large state spaces, a normal computer will lack RAM quickly
 - => **Memory-bounded variants of A^***
 - Read pages 99-102 in the reference book
 - IDA* is one of these variants
 - Same idea as Iterative Deepening Search, but the cutoff is the f -cost ($g + h$) rather than the depth

Chapter 3 – part 2

Summary

Summary

- ❑ In this lecture, you've learnt about graph search...
 - And, in particular, about informed search strategies in graphs
- ❑ You've seen two best-first search algorithms:
 - Greedy best-first search
 - A* search
 - We've also talked briefly about some variants of A*
- ❑ You've learnt how to design a good heuristic for A* search
- ❑ You've learnt about the properties of A* search

Chapter 3 – part 2

Questions



Recall : homework

- ❑ Do the exercises 1 and 2 of Chapter 3-part1
 - Slides 85 and 86
- ❑ Assignment on Teams for Monday, 1st of Nov., 23h59



25 YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you
for your
attention!



soict.hust.edu.vn/



fb.com/groups/soict

