有信息搜索（启发式搜索）

1 启发式函数：如何更快地接近目标？ 什么是启发式函数？ 什么是启发式搜索？

2 贪婪最佳优先搜索，A*搜索

3 可采纳性、一致性、准确性、松弛问题：什么是可采纳的启发函数？ 什么 是一致的启发函数?启发式函数的准确性。如何设计可采纳的启发函数?

Blind Strategies

- Breadth-first: depth(n)
- Uniform-cost: g(n)
- Depth-first: -depth(n)
- Depth-limited: -depth(n), depth(n)<=limit
- Iterative deepening
- Speeding up techniques
  - Avoiding repetitive states
  - Bi-directional search

- Blind strategies do not know which node in frontier is closer to the goal.

启发式函数

An Idea: *estimating the distance to the goal*

- It would be better if the agent knew whether or not the city it is travelling to gets it closer to Bucuresti
- Of course, the agent doesn't know the exact distance or path to Bucuresti (it wouldn't need to search otherwise!)
- The agent must estimate the distance to Bucuresti

Informed (Heuristic) search {3.5}

- More generally
  - We want the search algorithm to be able to estimate the path cost from the current node to the goal
- This estimate is called a heuristic(启发式) function
- Can't be done based on problem formulation
  - Need to add additional information
  - Informed search

Heuristic Function {3.5}

- Heuristic function $h(n)$
  - $h(n)$: estimated cost from node n to goal
  - $h(n_1) < h(n_2)$ means it's probably cheaper to get to the goal from n_1
  - $h(n_{goal}) = 0$
- Path cost $g(n)$
  - $g(n)$: cost from the initial node to n
- Evaluation function $f(n)$
  - $f(n) = g(n)$ Uniform Cost
  - $f(n) = h(n)$ Greedy Best-First
  - $f(n) = g(n) + h(n)$ A*

Heuristic Function {3.5.1}

- The heuristic function $h(n) \geq 0$ estimates the cost to go from STATE(n) to a goal state.
  - independent of the current search tree;
  - depends only on STATE(n) and the goal test GOAL.
- Example:

5		8
4	2	1
7	3	6

STATE(n)

1	2	3
4	5	6
7	8	

Goal state

  - How to estimate the cost to the goal?

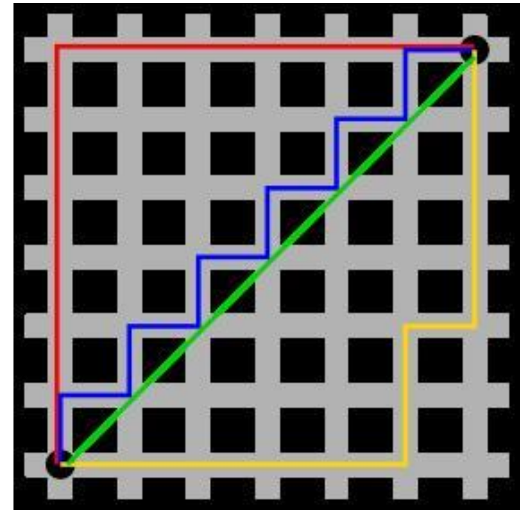
Heuristic Function {3.5.1}

5		8
4	2	1
7	3	6

STATE(n)

1	2	3
4	5	6
7	8	

Goal state



P1(x1, y1)到P2 (x2, y2)的曼哈顿距离:
|x1 - x2| + |y1 - y2|

- $h_1(N)$ = number of misplaced numbered tiles = 6
- $h_2(N)$ = sum of the (Manhattan) distance of every numbered tile to its goal position
  = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13

贪婪最佳优先搜索、A*搜索

Greedy best-first search{3.5.1}

- Greedy best-first search expands the node that is closest to the goal. It evaluates nodes by using just the heuristic function; that is, $f(n)=h(n)$.

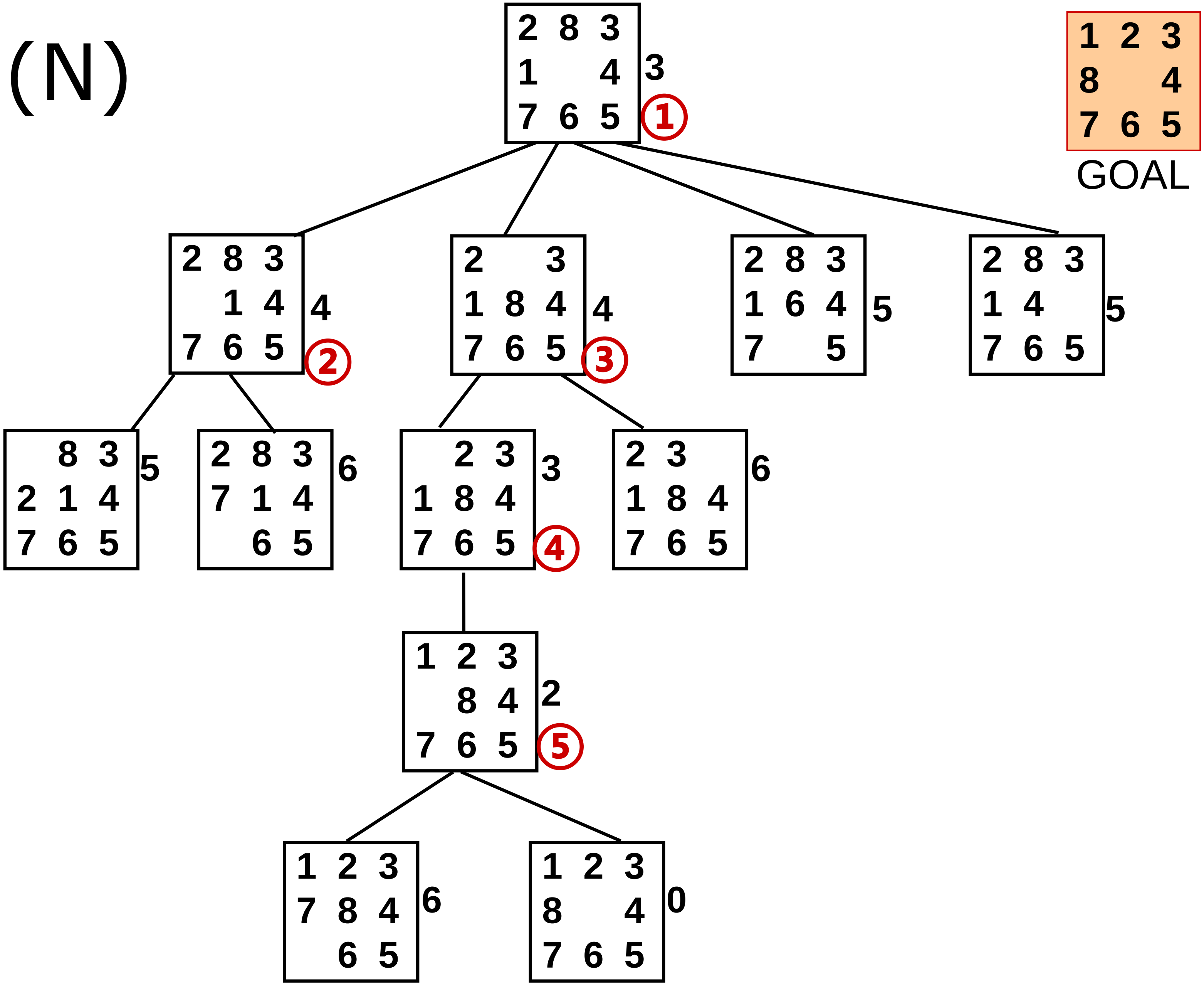
5		8
4	2	1
7	3	6

STATE(N)

1	2	3
4	5	6
7	8	

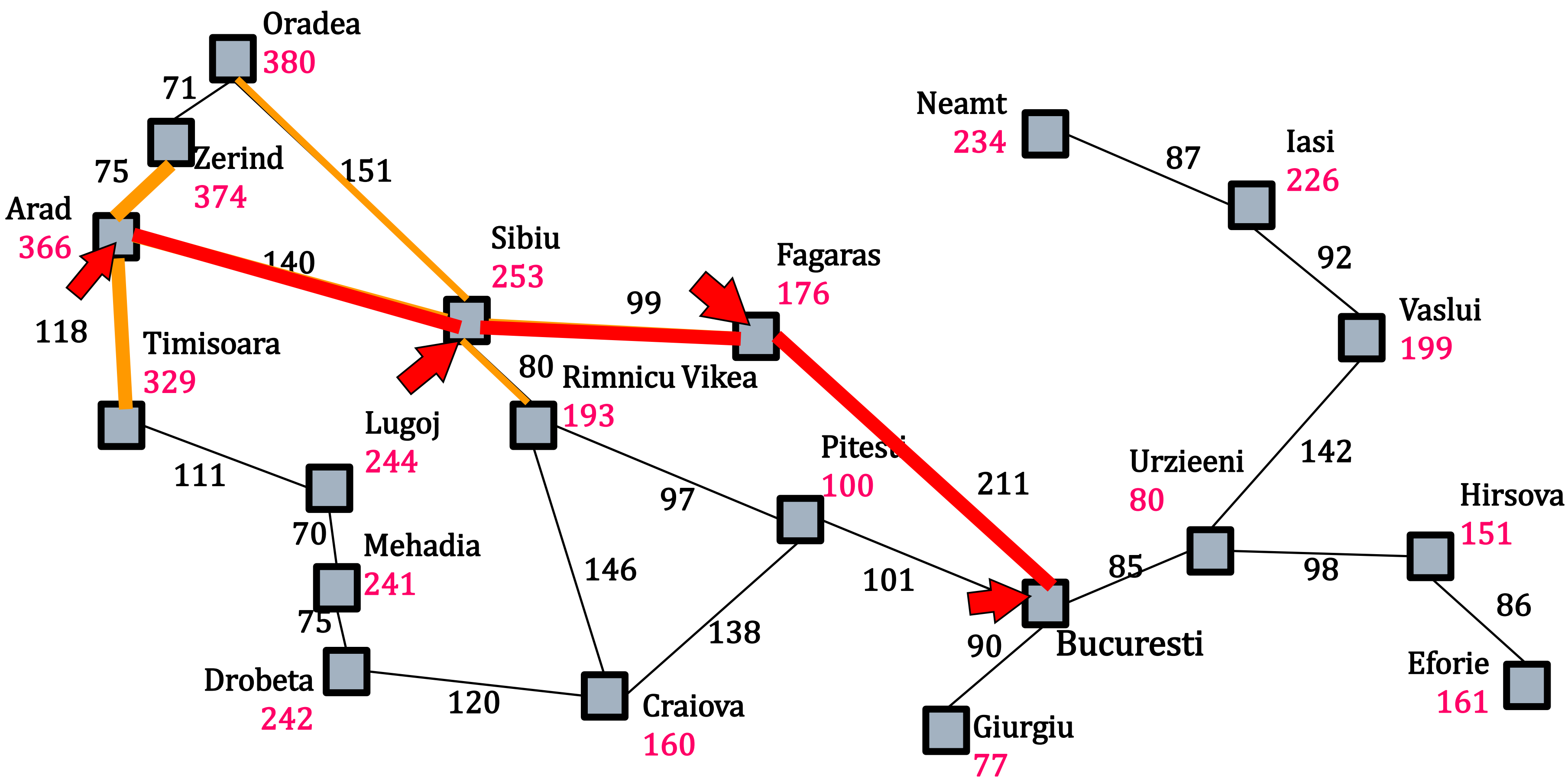
Goal state

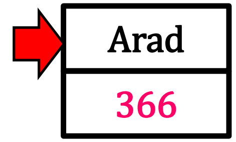
$h_1(N)$

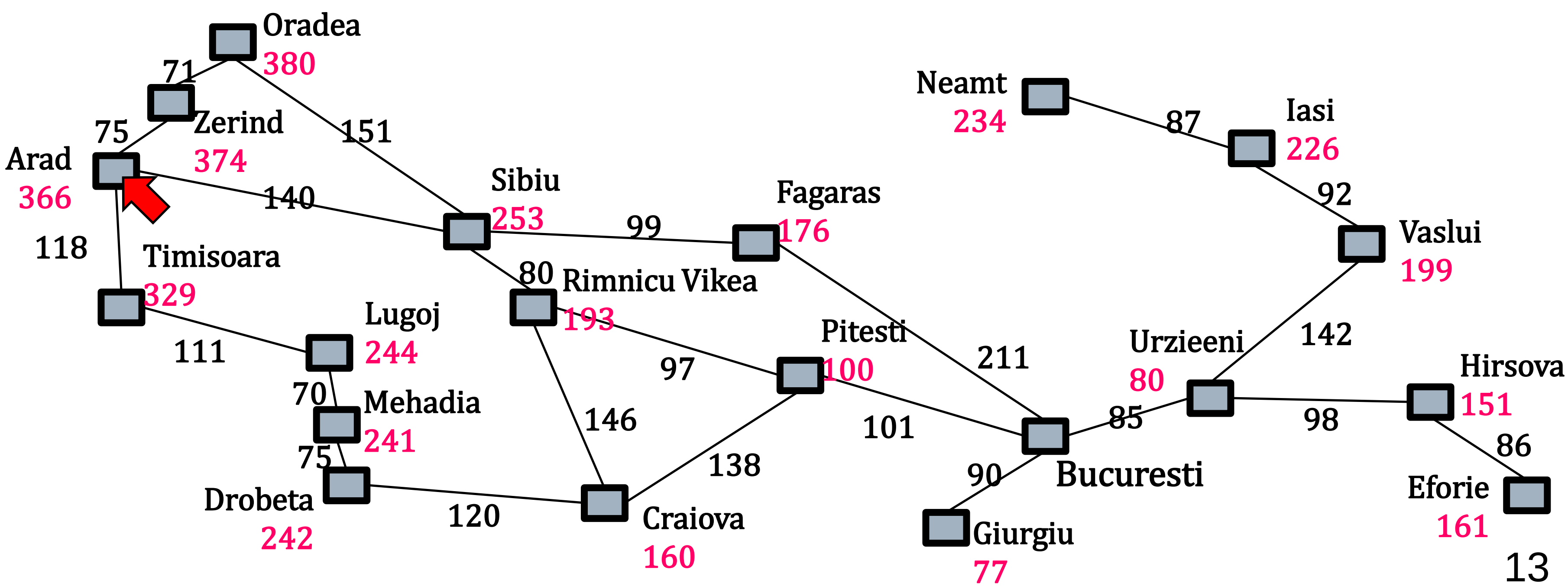

2 8 3
1 4
7 6 5
3
1
1 2 3
8 4
7 6 5
GOAL
2 8 3
1 4
7 6 5
4
2
2 3
1 8 4
7 6 5
4
3
2 8 3
1 6 4
7 5
5
2 8 3
1 4
7 6 5
5
8 3
2 1 4
7 6 5
5
2 8 3
7 1 4
6 5
6
2 3
1 8 4
7 6 5
3
4
2 3
1 8 4
7 6 5
6
1 2 3
8 4
7 6 5
2
5
1 2 3
7 8 4
6 5
6
1 2 3
8 4
7 6 5
0

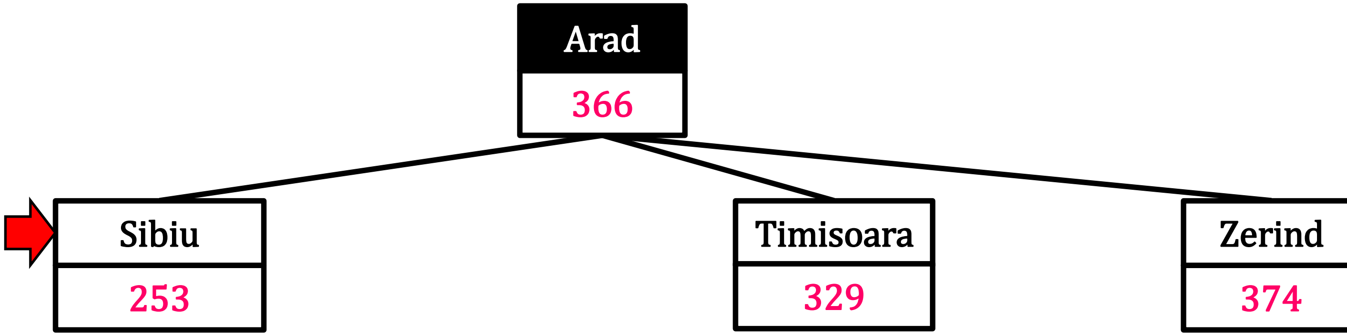
Example: Greedy Best-First Search {3.5.1}

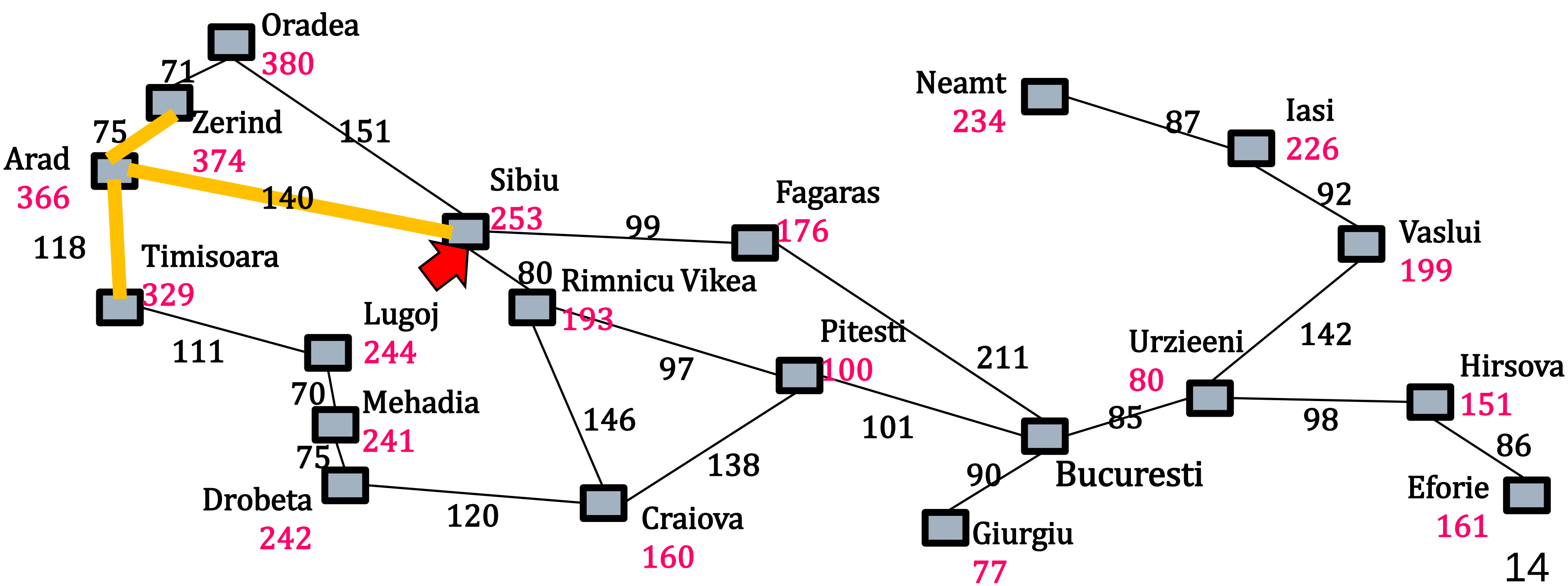
- Expand the node that seems closest to Bucuresti
- h(n) = straight-line distance

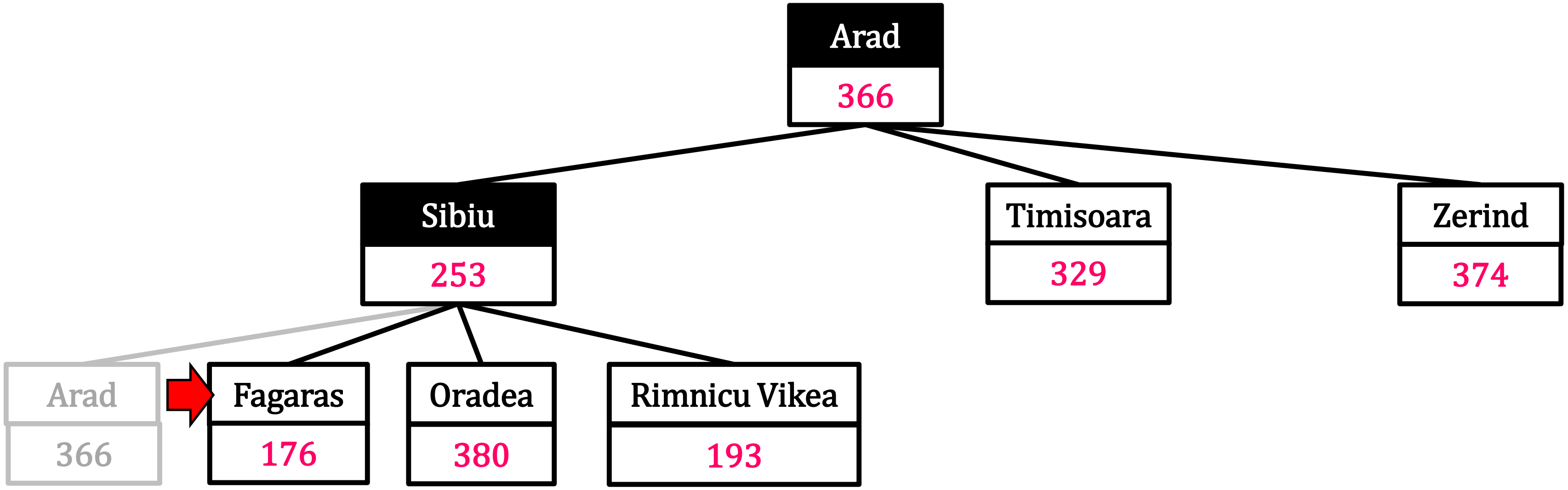


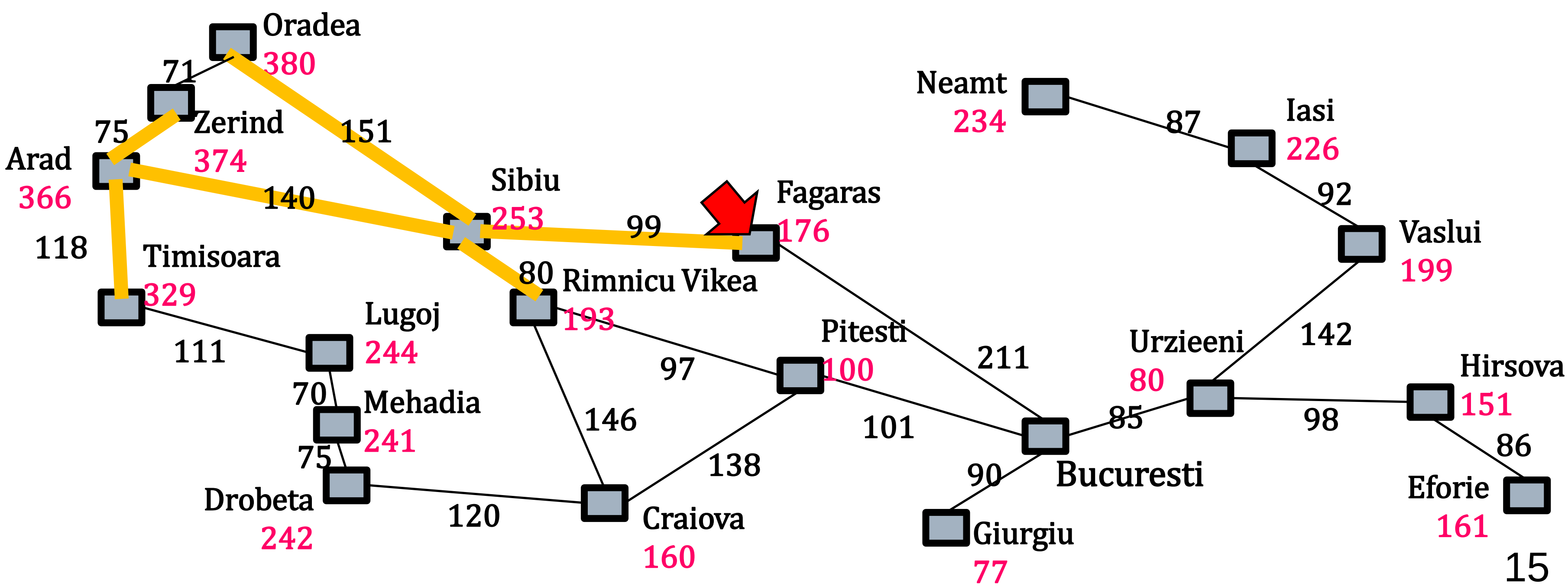

Arad
366

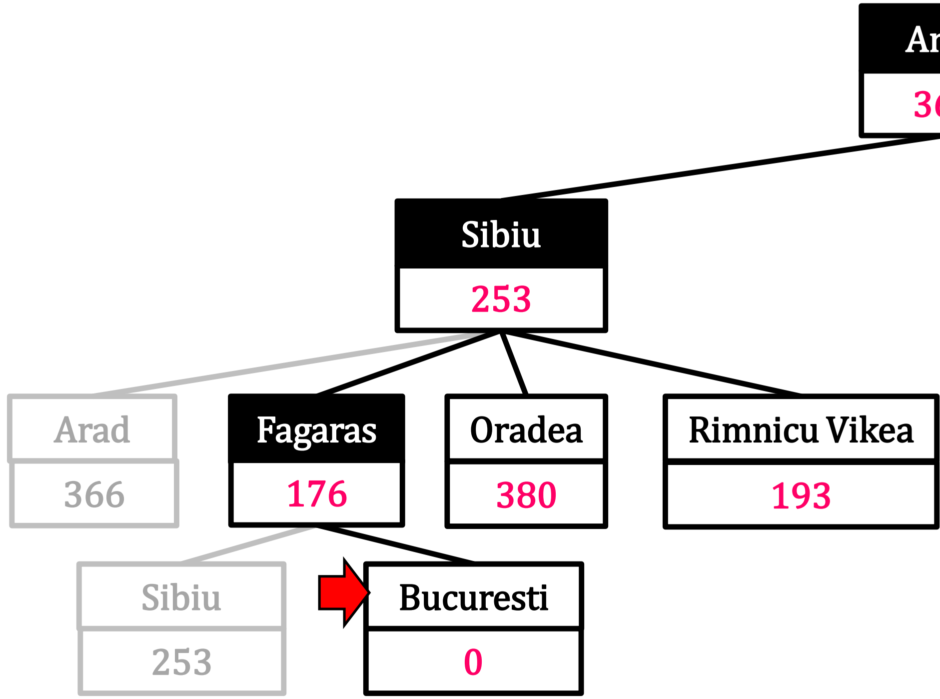

Oradea
380
71
Zerind
374
75
Arad
366
151
140
118
Sibiu
253
99
Fagaras
176
80
Rimnicu Vikea
193
Timisoara
329
111
Lugoj
244
70
Mehadia
241
75
Drobeta
242
120
Craiova
160
146
97
Pitesti
100
138
101
211
Bucuresti
90
Giurgiu
77
85
Urzieeni
80
Neamt
234
87
Iasi
226
92
Vaslui
199
142
98
Hirsova
151
86
Eforie
161


Arad
366
Sibiu
253
Timisoara
329
Zerind
374


Oradea
380
71
Zerind
374
75
151
Arad
366
140
118
Sibiu
253
99
Fagaras
176
Timisoara
329
80
Rimnicu Vikea
193
Lugoj
244
111
70
Mehadia
241
97
Pitesti
100
146
75
Drobeta
242
120
138
101
211
Craiova
160
Neamt
234
87
Iasi
226
92
Vaslui
199
142
Urzieeni
80
85
Bucuresti
90
Giurgiu
77
98
Hirsova
151
86
Eforie
161

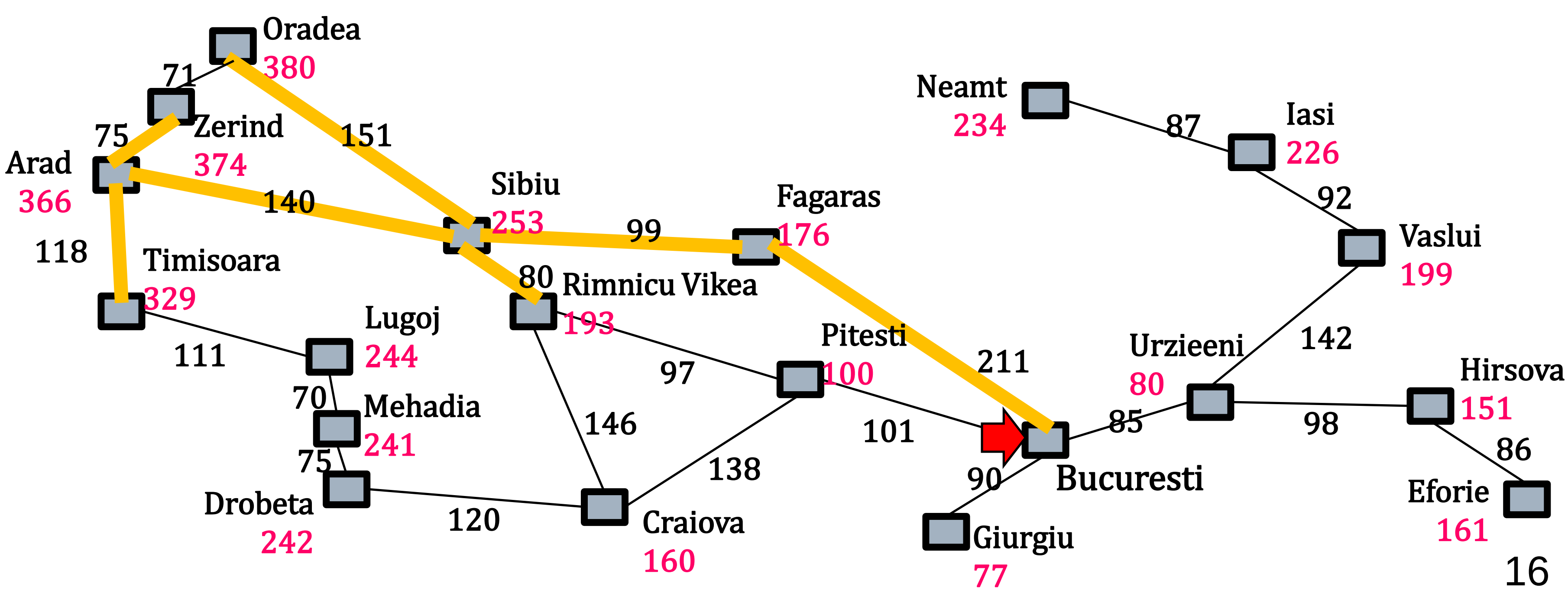

Arad
366
Sibiu
253
Timisoara
329
Zerind
374
Arad
366
Fagaras
176
Oradea
380
Rimnicu Vikea
193


Oradea
380
71
Zerind
374
75
Arad
366
151
140
118
Sibiu
253
99
Fagaras
176
80
Rimnicu Vikea
193
Timisoara
329
111
Lugoj
244
70
Mehadia
241
75
Drobeta
242
120
146
97
Pitesti
100
138
Craiova
160
211
101
Bucuresti
90
Giurgiu
77
85
Urzieeni
80
Neamt
234
87
Iasi
226
92
Vaslui
199
142
98
Hirsova
151
86
Eforie
161



贪婪最佳优先树搜索

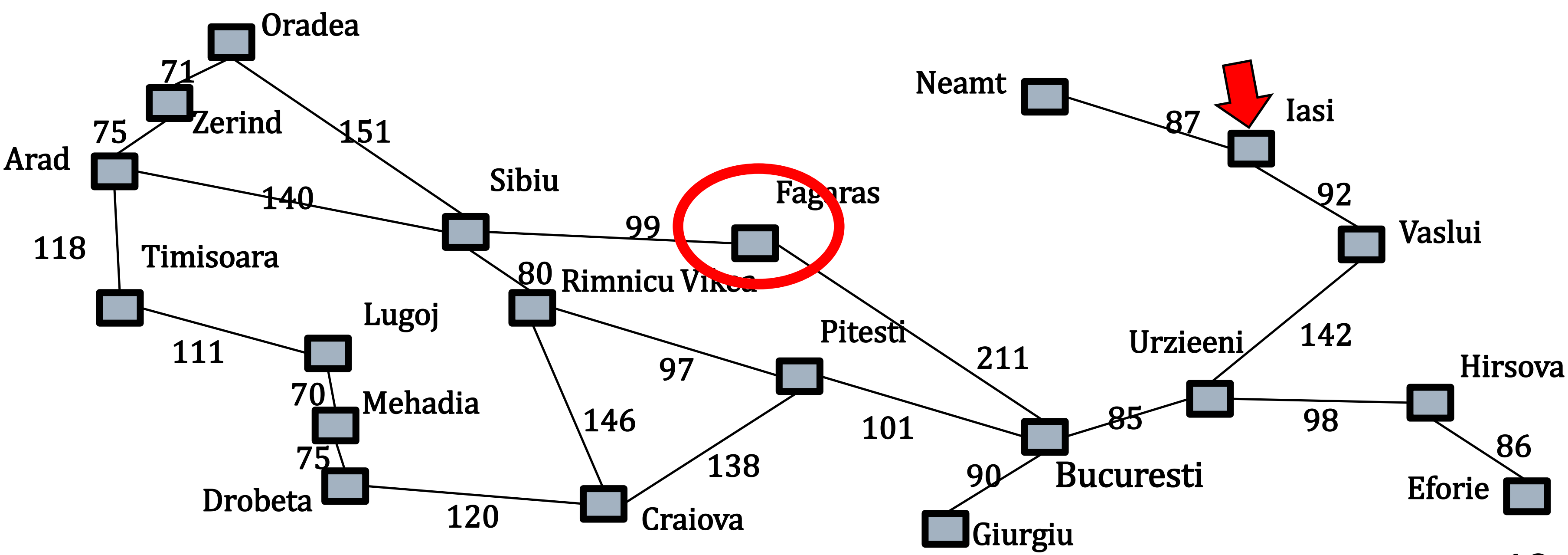
贪婪最佳优先图搜索



贪婪最佳优先树搜索
有何问题？

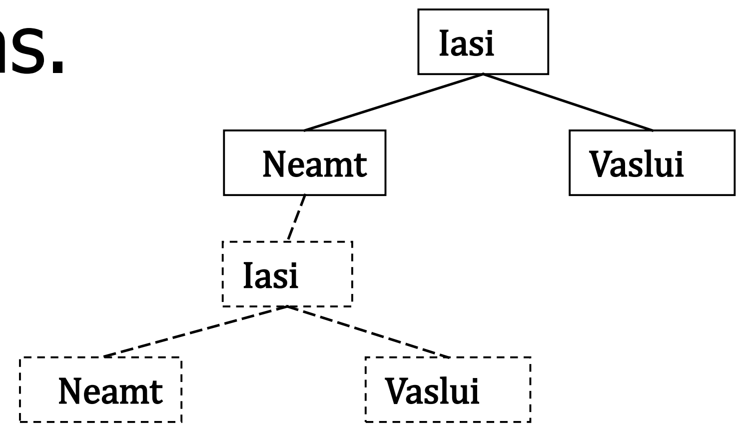
Example: Greedy Best-First Search {3.5.1}

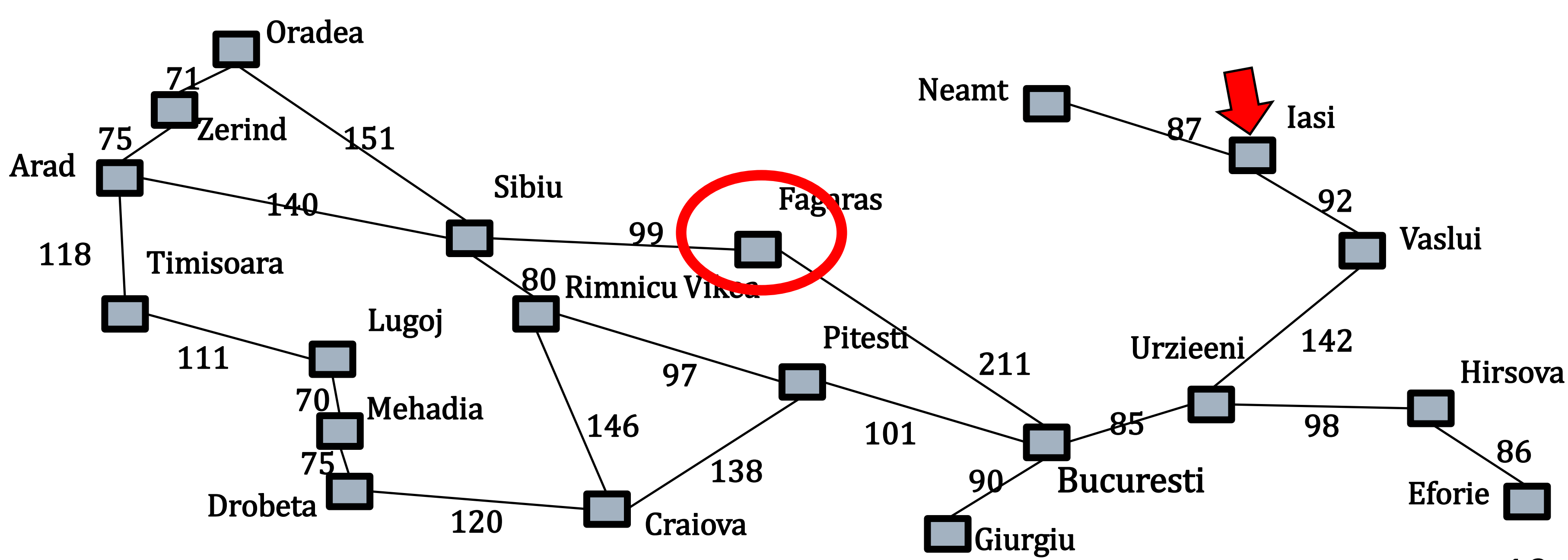
- try: from Iasi to Fagaras.



Example: Greedy Best-First Search {3.5.1}

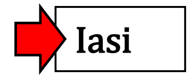
- try: from Iasi to Fagaras.
- can stuck in loops

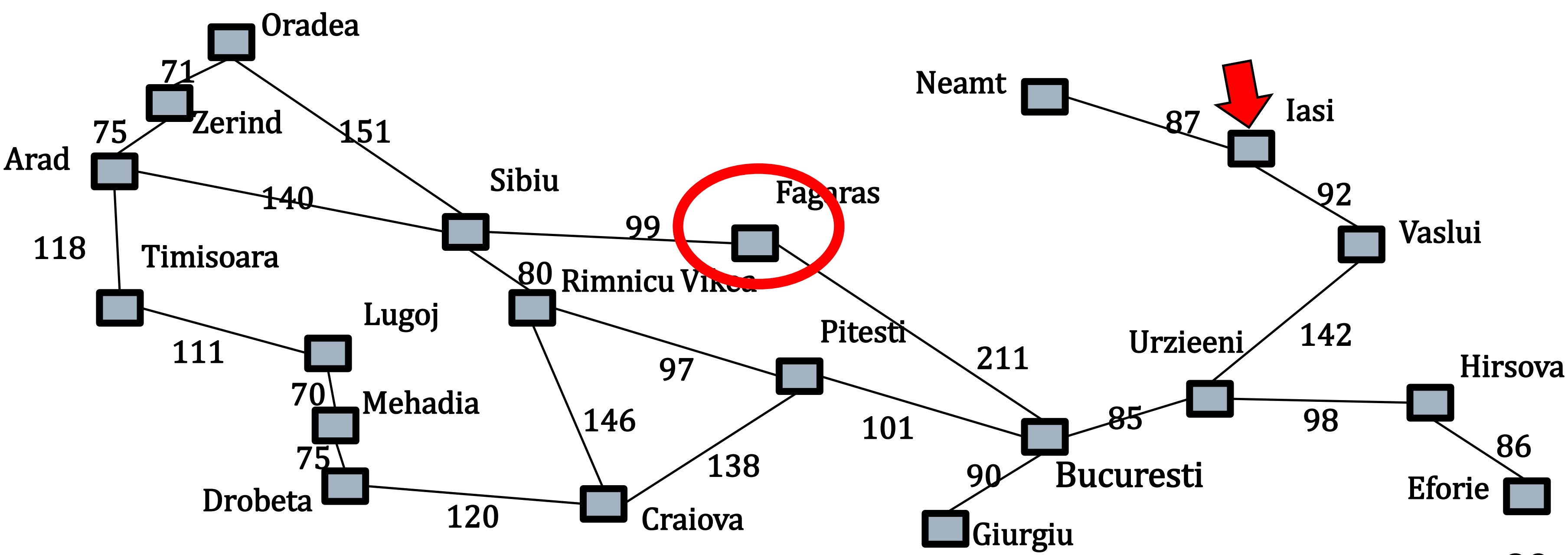




Example: Greedy Best-First Search {3.5.1}

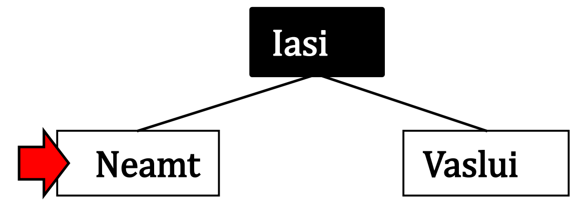
- try: from Iasi to Fagaras.
- consider checking repeated states

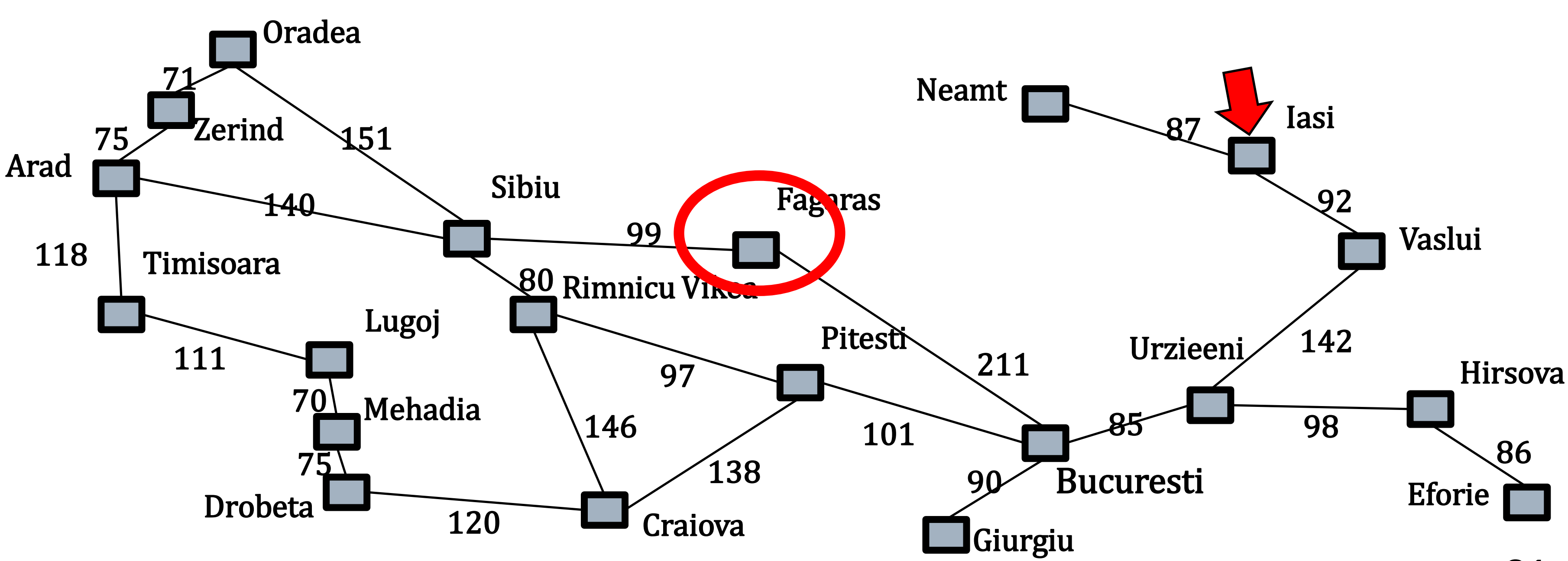




Example: Greedy Best-First Search {3.5.1}

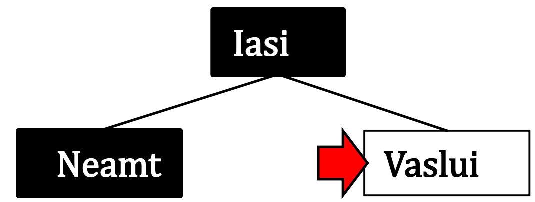
- try: from Iasi to Fagaras.
- consider checking repeated states

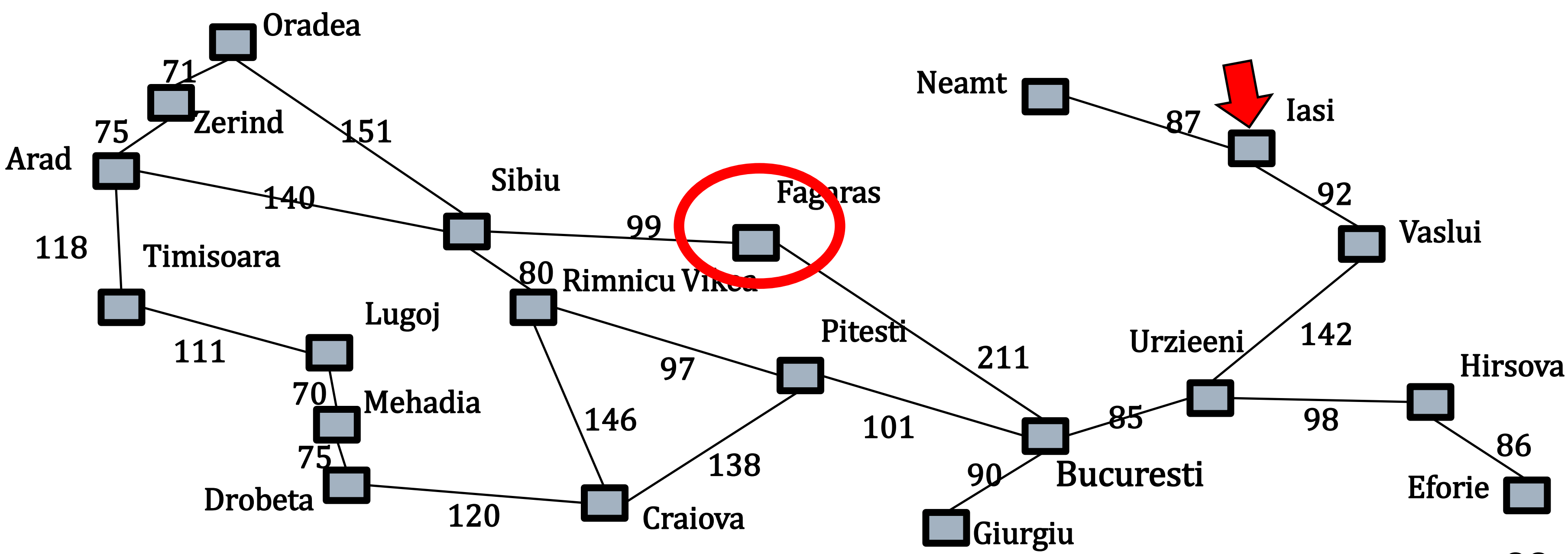




Example: Greedy Best-First Search {3.5.1}

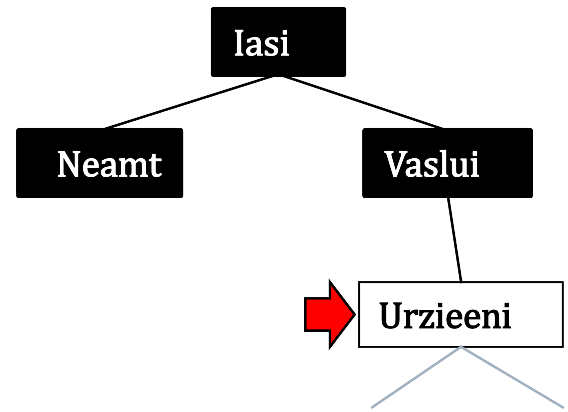
- try: from Iasi to Fagaras.
- consider checking repeated states

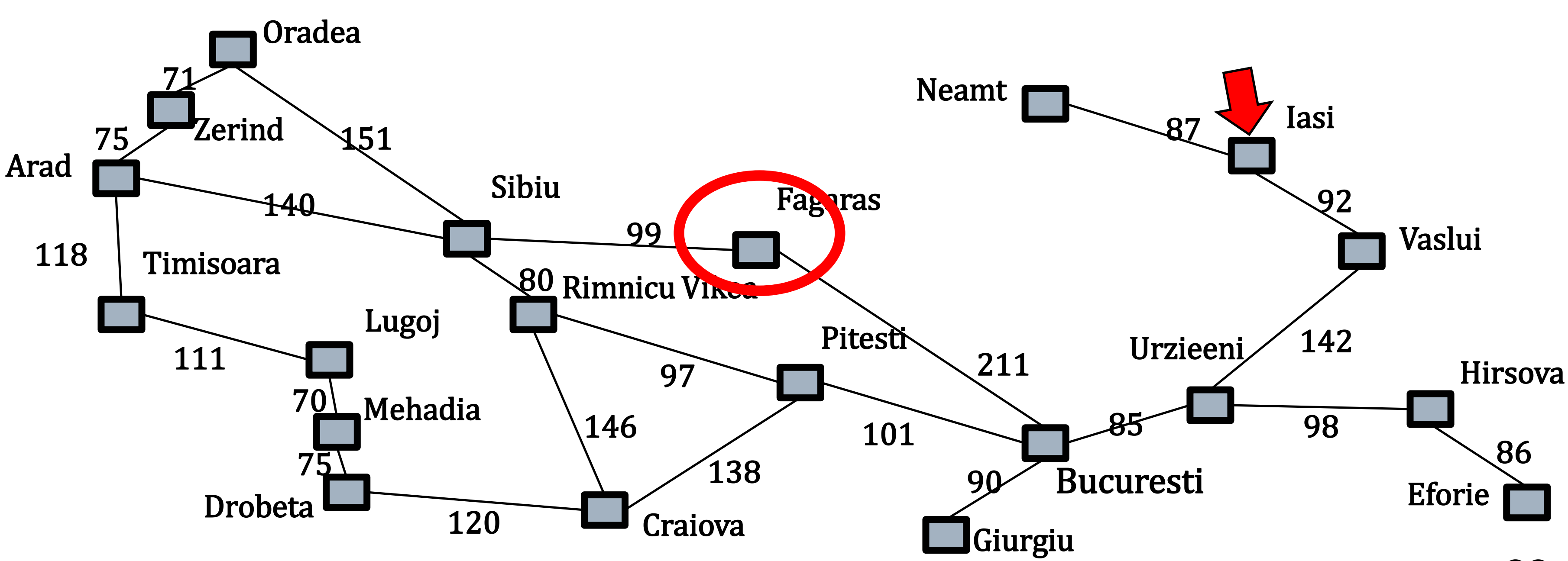




Example: Greedy Best-First Search {3.5.1}

- try: from Iasi to Fagaras.
- consider checking repeated states





complete? optimal? complexity{3.5.1}

- Complete?

	Finite space	Infinite space
GBFS tree search		
GBFS graph search		

complete? optimal? complexity{3.5.1}

- Optimal?
  - No
- m: maximal depth of a leaf node
- Time
  - $O(b^m)$, but a good heuristic can give dramatic improvement
- Space
  - $O(b^m)$ -- keeps all nodes in memory

Greedy Best First (f(n)=h(n)) is not complete (without checking repeated states)

and is not optimal.

How to solve this problem?

A* search {3.5.2}

- Idea
  - avoid expanding paths that are already expensive
- 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

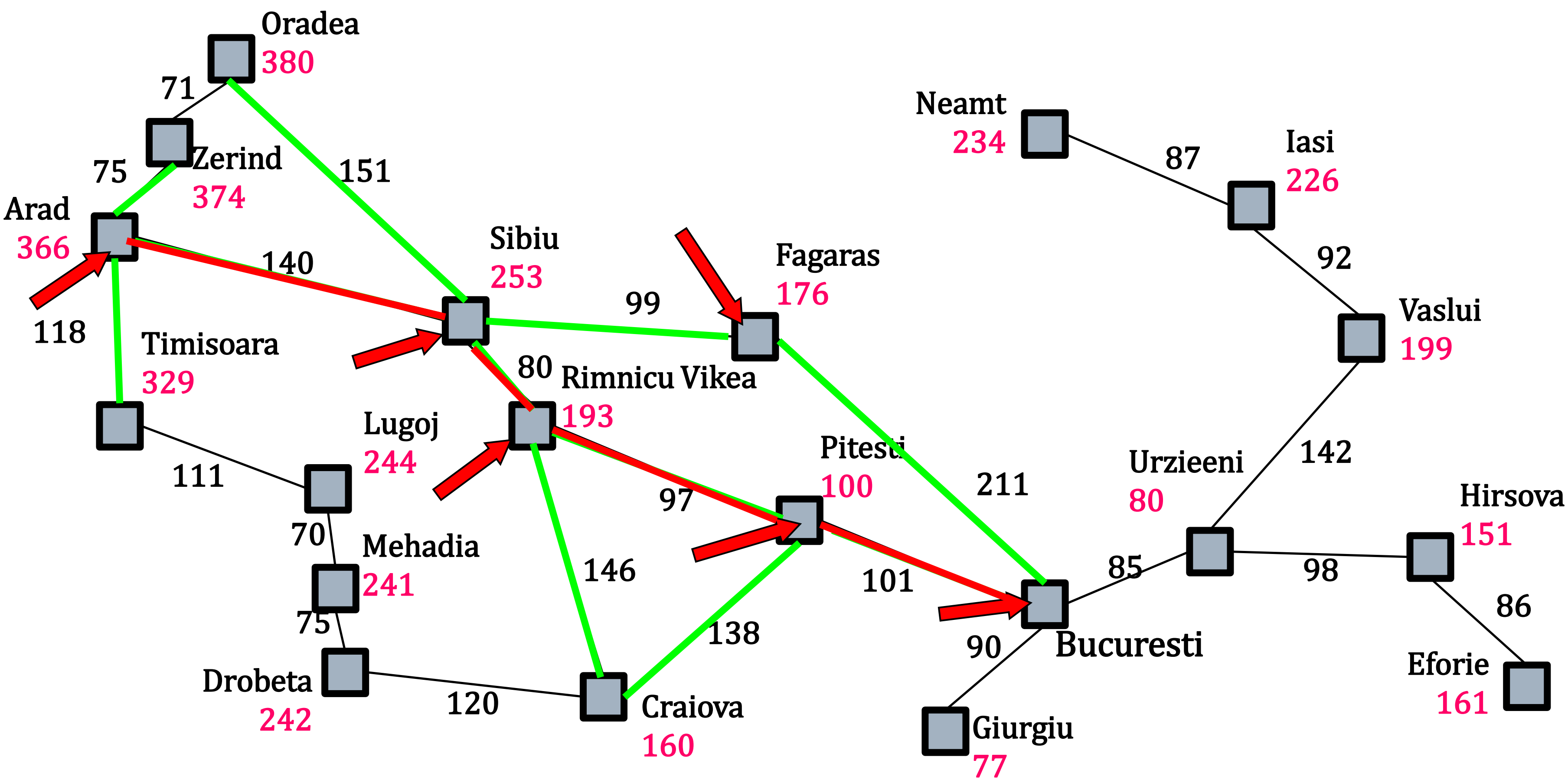
```
function A-Star-Search(problem) returns a solution, or failure
    node ← a node with State=problem.Initial-State, Path-Cost=0
    if problem.Goal-Test(node.State) then return Solution(node)
    frontier←{node}
    explored←an empty set
    loop do
        if Empty?(frontier) then return failure
        node ← POP the node with the lowest g(n)+h(n) in frontier
        if problem.Goal-Test(child.State) then return Solution(child)
        add node.State to explored
        for each action in problem.Actions(node.State) do
            child ← Child-Node(problem, node, action)
            if child.State is not in explored or frontier then
                frontier ← Insert(child, frontier)
            else if child.State is in frontier with higher g+h then
                replace that frontier node with child
```

//Tree search: frontier ← Insert(child, frontier)

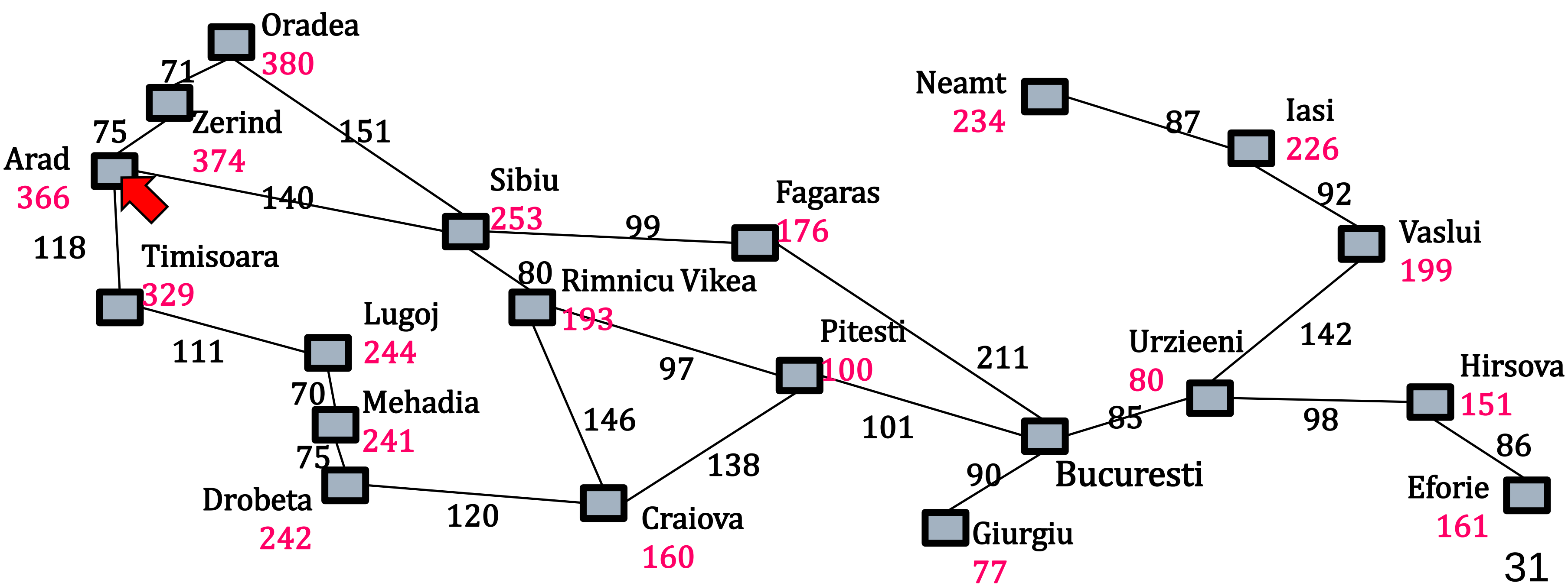
similar to uniform cost search

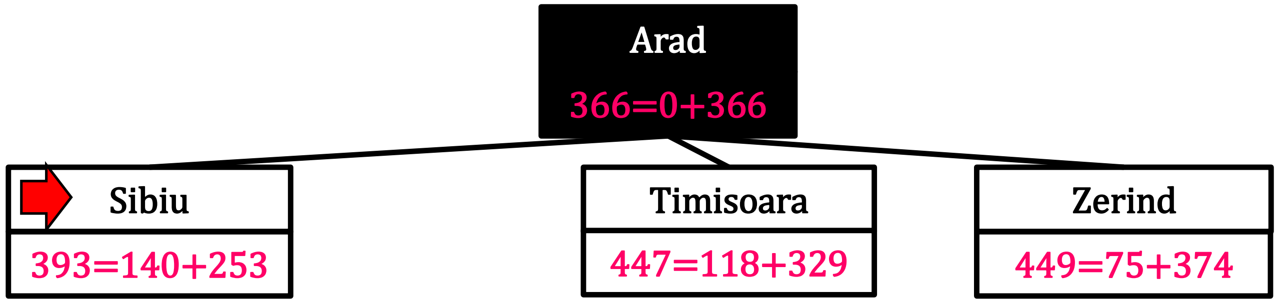
Example: A* Search {3.5.2}

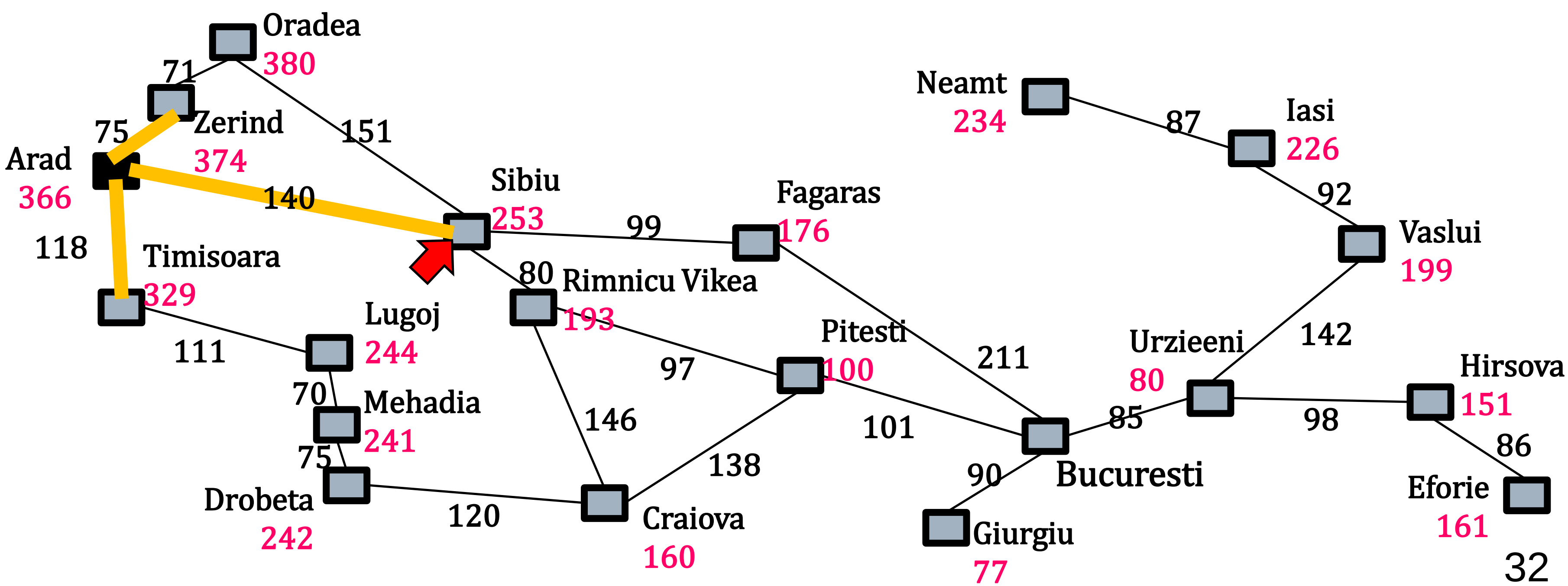
- $f(n) = g(n) + h(n)$
- h(n) = straight-line distance

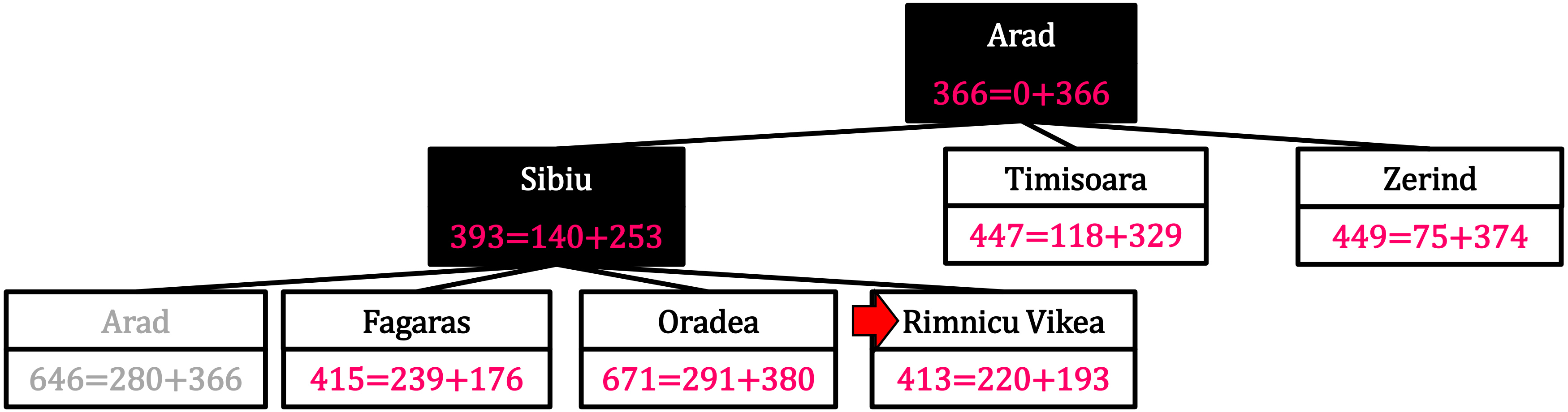


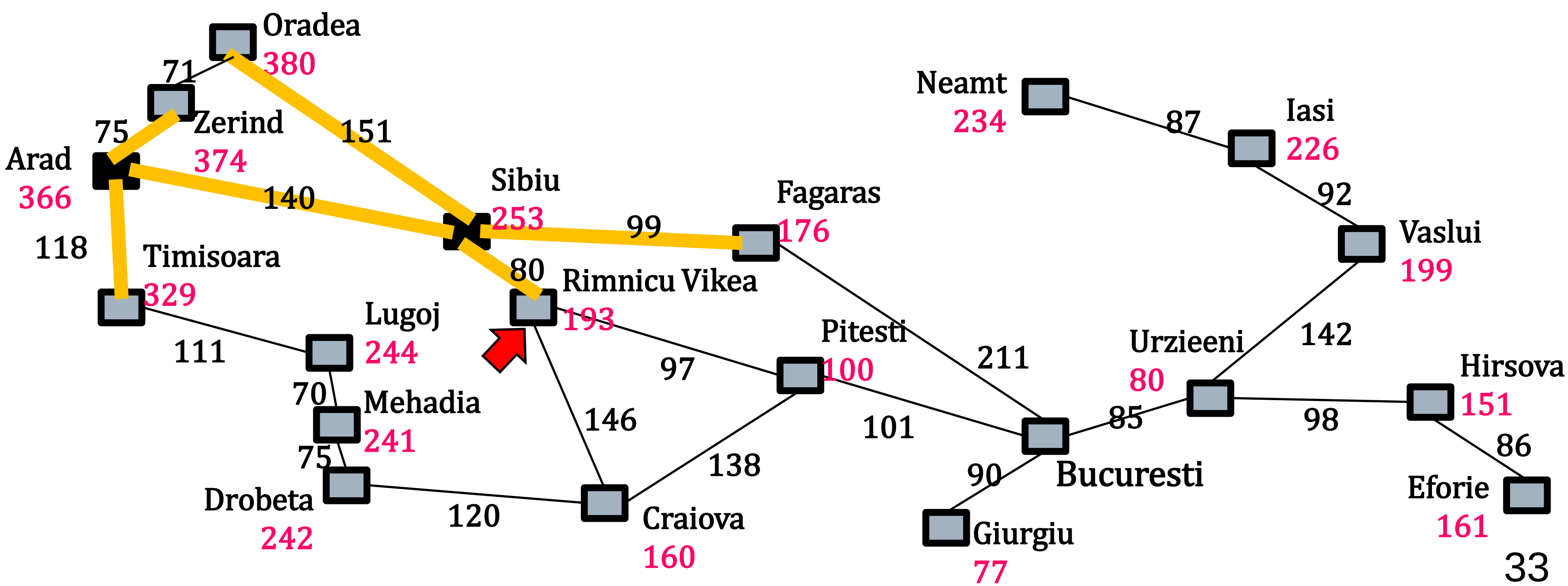

Arad
366=0+366

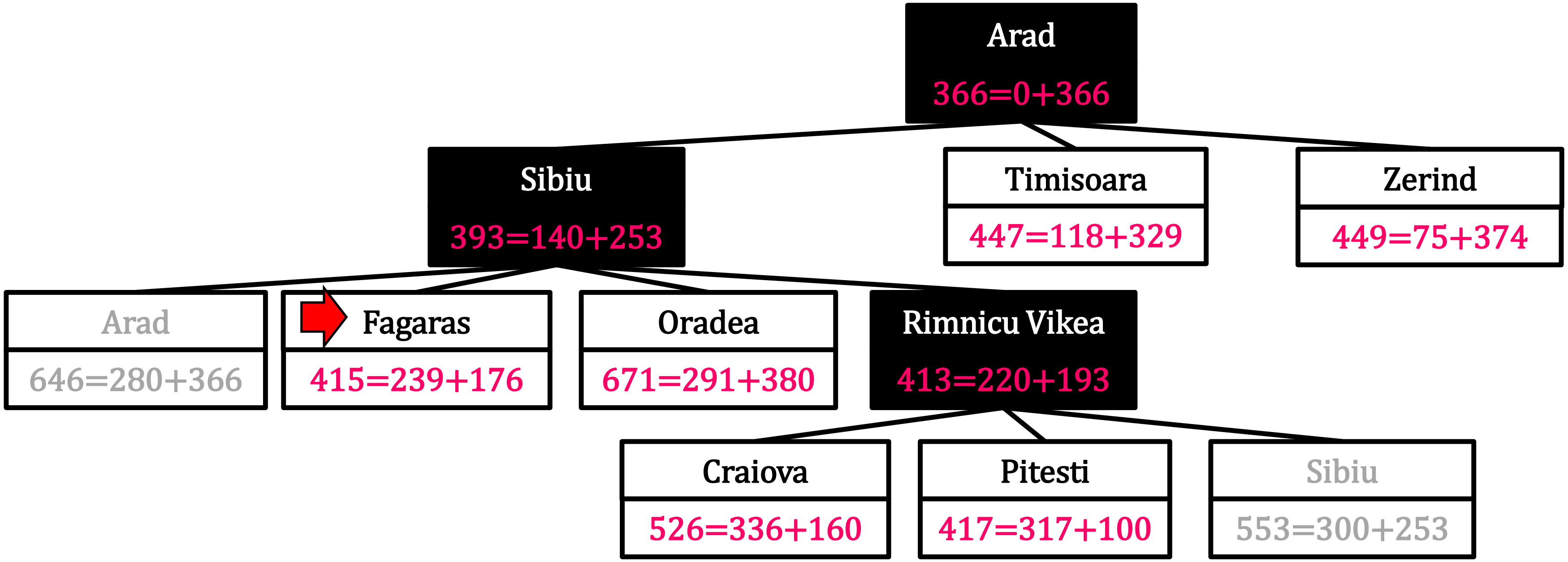

Oradea
380
Zerind
374
Arad
366
Sibiu
253
Fagaras
176
Timisoara
329
Rimnicu Vikea
193
Lugoj
244
Pitesti
100
Mehadia
241
Drobeta
242
Craiova
160
Neamt
234
Iasi
226
Vaslui
199
Urzieeni
80
Hirsova
151
Bucuresti
Giurgiu
77
Eforie
161
71
75
151
140
118
99
80
111
97
70
146
75
120
138
101
211
90
85
87
92
142
98
86

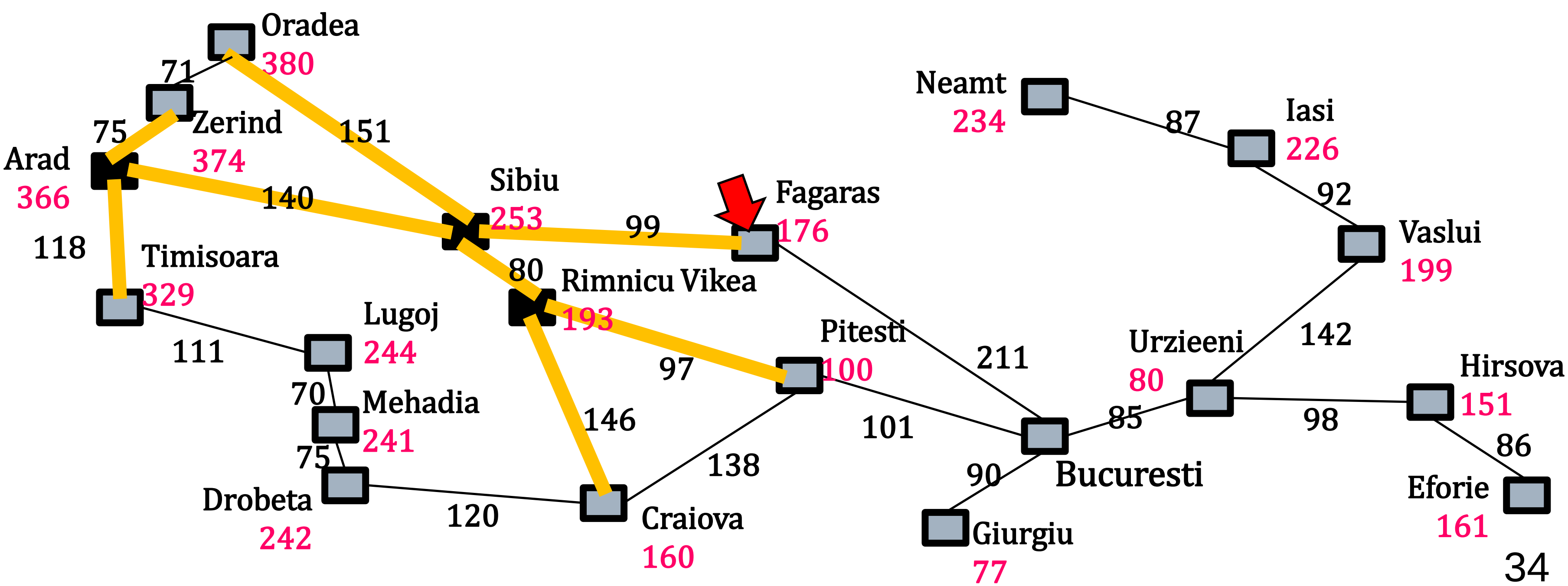

Arad
366=0+366
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374

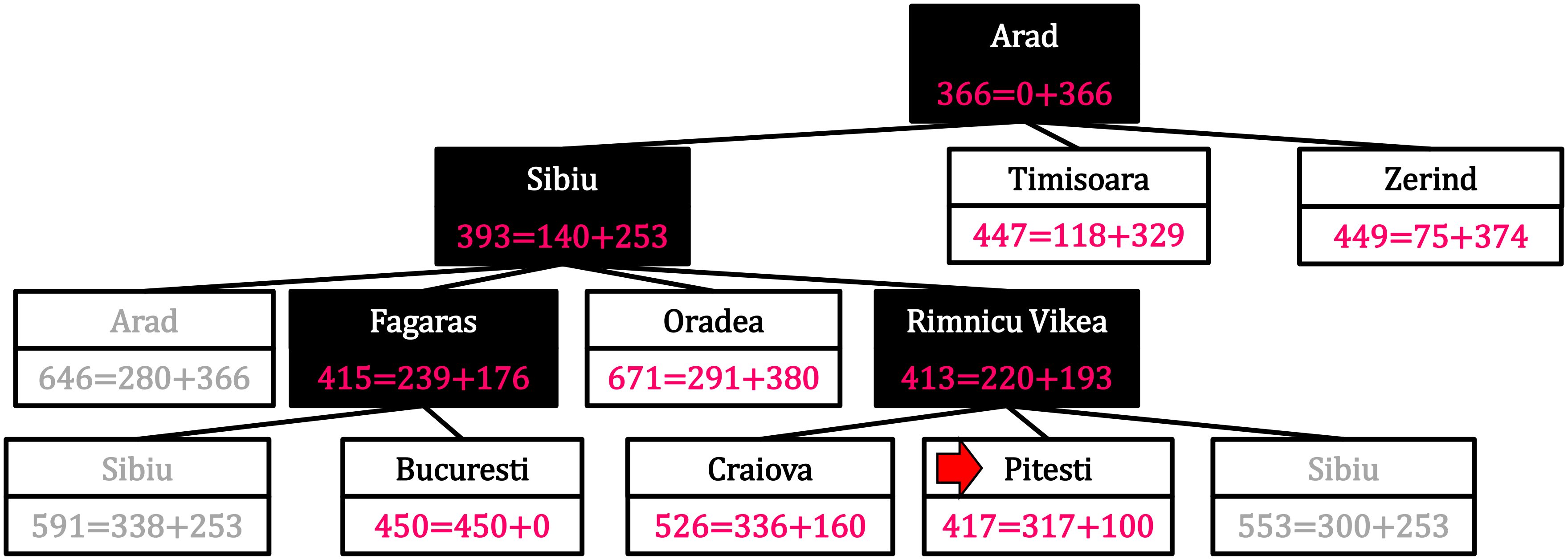

Oradea
380
71
Zerind
374
75
151
Arad
366
140
Sibiu
253
99
Fagaras
176
118
Timisoara
329
80
Rimnicu Vikea
193
Lugoj
244
111
70
Mehadia
241
75
Drobeta
242
120
146
97
Pitesti
100
138
Craiova
160
101
211
Neamt
234
87
Iasi
226
92
Vaslui
199
142
Urzieeni
80
85
Bucuresti
90
Giurgiu
77
98
Hirsova
151
86
Eforie
161

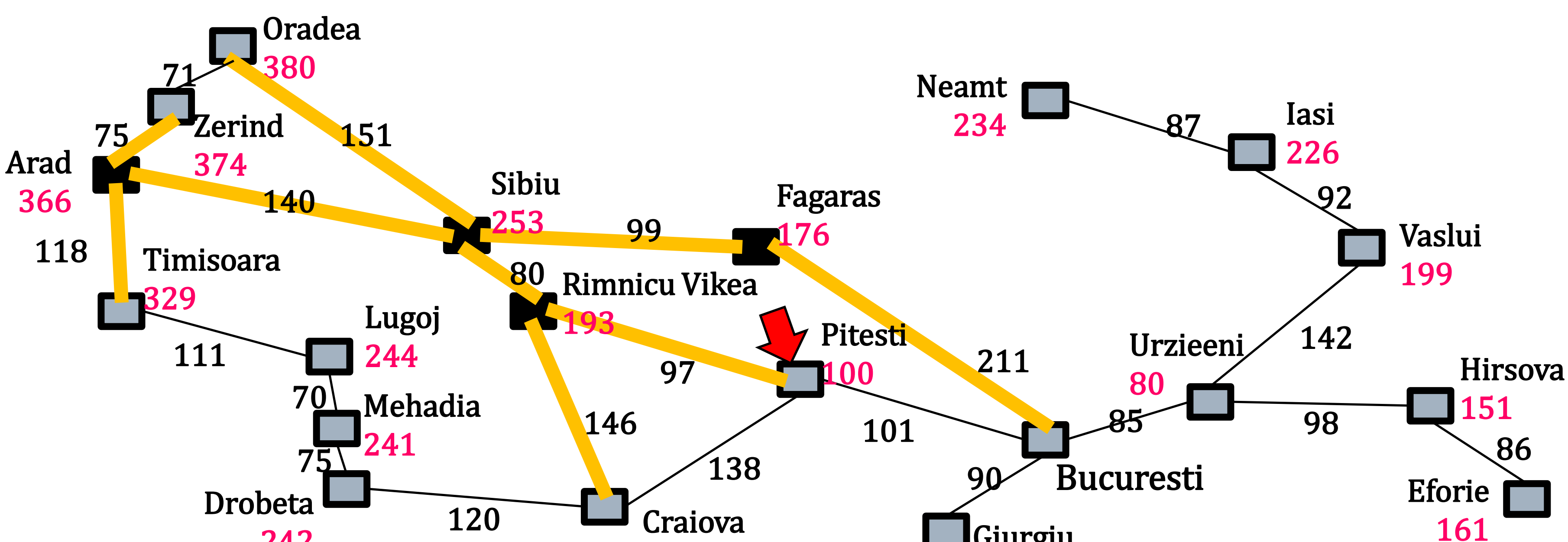

Arad
366=0+366
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vikea
413=220+193

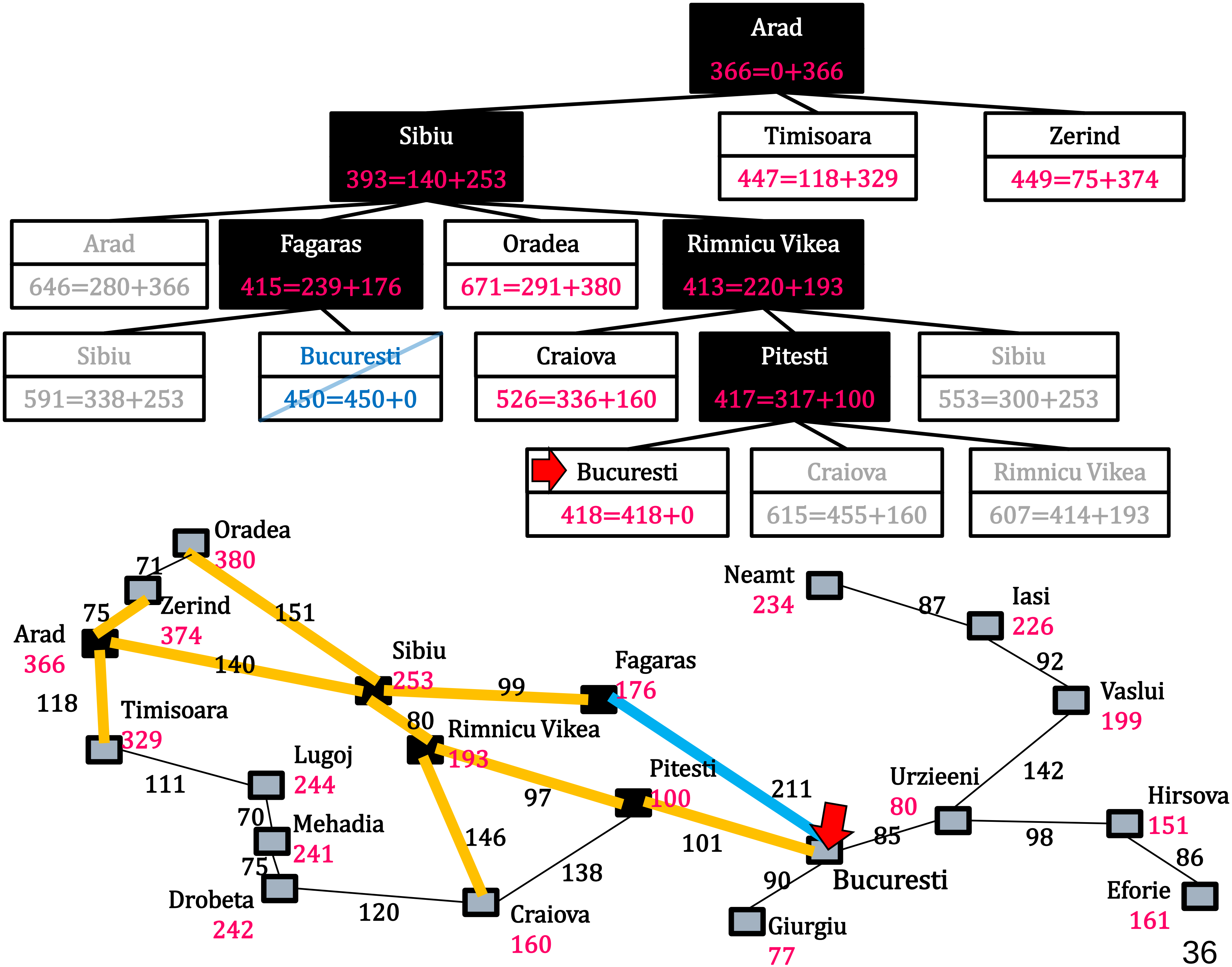

Oradea
380
71
Zerind
374
75
Arad
366
151
140
118
Sibiu
253
99
Fagaras
176
Timisoara
329
80
Rimnicu Vikea
193
111
Lugoj
244
70
Mehadia
241
75
Drobeta
242
120
146
97
Craiova
160
138
Pitesti
100
101
211
Neamt
234
87
Iasi
226
92
Vaslui
199
142
Urzieeni
80
85
Bucuresti
90
Giurgiu
77
98
Hirsova
151
86
Eforie
161


Arad
366=0+366
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vikea
413=220+193
Craiova
526=336+160
Pitesti
417=317+100
Sibiu
553=300+253


Oradea
380
71
Zerind
374
75
Arad
366
151
140
118
Sibiu
253
99
Fagaras
176
Timisoara
329
80
Rimnicu Vikea
193
111
Lugoj
244
70
Mehadia
241
75
Drobeta
242
120
97
146
Craiova
160
138
Pitesti
100
101
211
Neamt
234
87
Iasi
226
92
Vaslui
199
142
Urzieeni
80
85
Bucuresti
90
Giurgiu
77
98
Hirsova
151
86
Eforie
161


Arad
366=0+366
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vikea
413=220+193
Sibiu
591=338+253
Bucuresti
450=450+0
Craiova
526=336+160
Pitesti
417=317+100
Sibiu
553=300+253

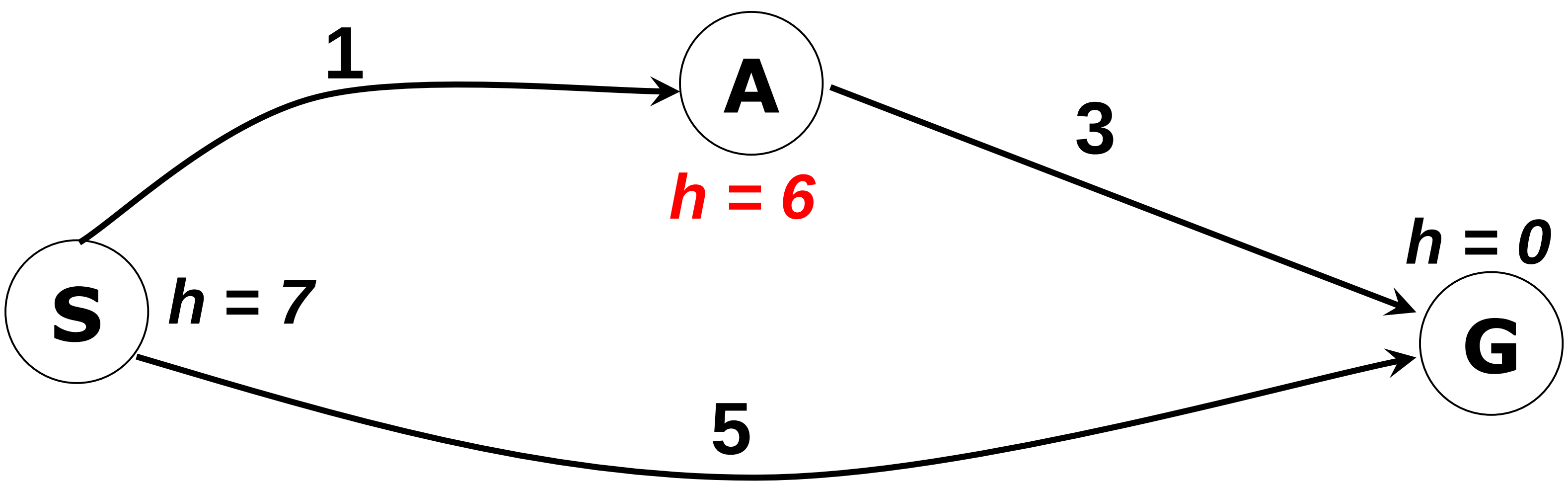

Oradea
380
71
Zerind
374
75
Arad
366
151
140
118
Sibiu
253
99
Fagaras
176
80
Rimnicu Vikea
193
Timisoara
329
111
Lugoj
244
70
Mehadia
241
75
Drobeta
242
120
146
97
Pitesti
100
138
101
211
Craiova
160
Neamt
234
87
Iasi
226
92
Vaslui
199
142
Urzieeni
80
85
98
Hirsova
151
86
Eforie
161
90
Bucuresti
Giurgiu
77



Arad
366=0+366
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vikea
413=220+193
Sibiu
591=338+253
Bucuresti
450=450+0
Craiova
526=336+160
Pitesti
417=317+100
Sibiu
553=300+253
Bucuresti
418=418+0
Craiova
615=455+160
Rimnicu Vikea
607=414+193
Oradea 380
Zerind 374
Arad 366
Timisoara 329
Lugoj 244
Mehadia 241
Drobeta 242
Sibiu 253
Rimnicu Vikea 193
Craiova 160
Fagaras 176
Pitesti 100
Bucuresti
Giurgiu 77
Urzieeni 80
Neamt 234
Iasi 226
Vaslui 199
Hirsova 151
Eforie 161
71
75
151
140
118
111
70
75
120
99
80
97
146
138
211
101
90
85
98
86
142
92
87

可采纳性、一致性、准确性、松弛问题

admissible heuristic{3.5.2}



- Try A* search and find what went wrong?
- Estimated goal cost > actual good goal cost
- We need estimates to be less than actual costs!

Admissible Heuristic {3.5.2}

- Let h*(N) be the cost of the optimal path from N to a goal node.
- The heuristic function h(N) is admissible (可采纳的) if:

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

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

G is a goal node → h(G) = 0

给定可采纳的启发式，

A*的树搜索和图搜索都具有最优性吗？

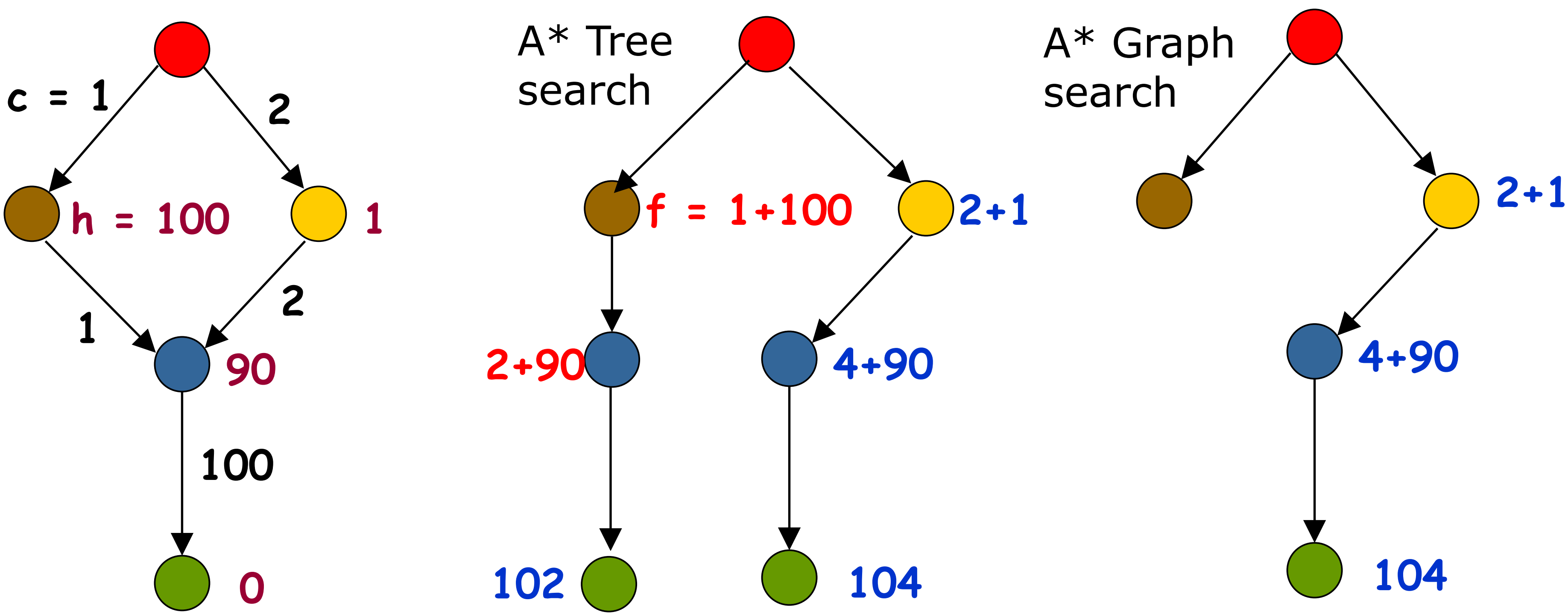
Properties of A* {3.5.2}

- optimal?

	inadmissible	admissible
A* tree search		
A* graph search		

Consistent Heuristic {3.5.2}

Try A* tree search and graph search respectively, and find what went wrong:



The heuristic h is clearly admissible.

h(n)<=c(n,n')+h(n') is required!!

consistent → admissible

- proof: consistent heuristic is admissible.
  - $h(n) <= c(n,n') + h(n') \rightarrow h(n) <= h^*(n)$

Properties of A* {3.5.2}

- optimal?

	inadmissible	admissible, inconsistent	consistent
A* tree search			
A* graph search			

yes? no?

C*是最优解路径的代价
A* cannot expand f_{i+1} until f_i is finished
A* expands _____ nodes with $f(n) < C^*$
A* expands _____ nodes with $f(n) = C^*$
A* expands _____ nodes with $f(n) > C^*$

all? some? no?

Properties of A* {3.5.2}

- Complete??
  - Yes, unless there are infinitely many nodes with f≤f(G)
- Time??
  - Exponential
- Space??
  - Keeps all nodes in memory

给定两个可采纳的启发式，
哪个更好？

Heuristic Accuracy{ 3.6.1}

5		8
4	2	1
7	3	6

STATE(N)

1	2	3
4	5	6
7	8	

Goal state

two admissible heuristics:

- $h_1(N)$ = number of misplaced tiles = 6
- $h_2(N)$ = sum of the (Manhattan) distances of every tile to its goal position = 3+1+3+0+2+1+0+3 = 13
- which one is better?

Heuristic Accuracy {3.6.1}

- Let h_1 and h_2 be two consistent heuristics such that for all nodes N:

$$h_1(N) \leq h_2(N)$$

- h_2 is said to be more accurate (or more informed) than h_1 and is better for search，h_2 dominates(占优势) h_1

5		8
4	2	1
7	3	6

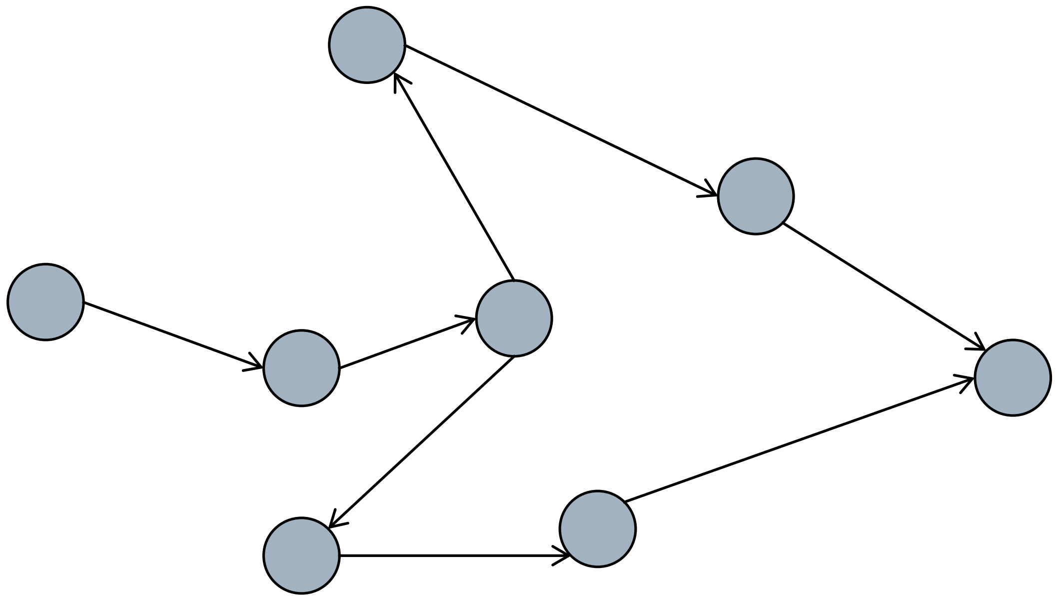
STATE(N)

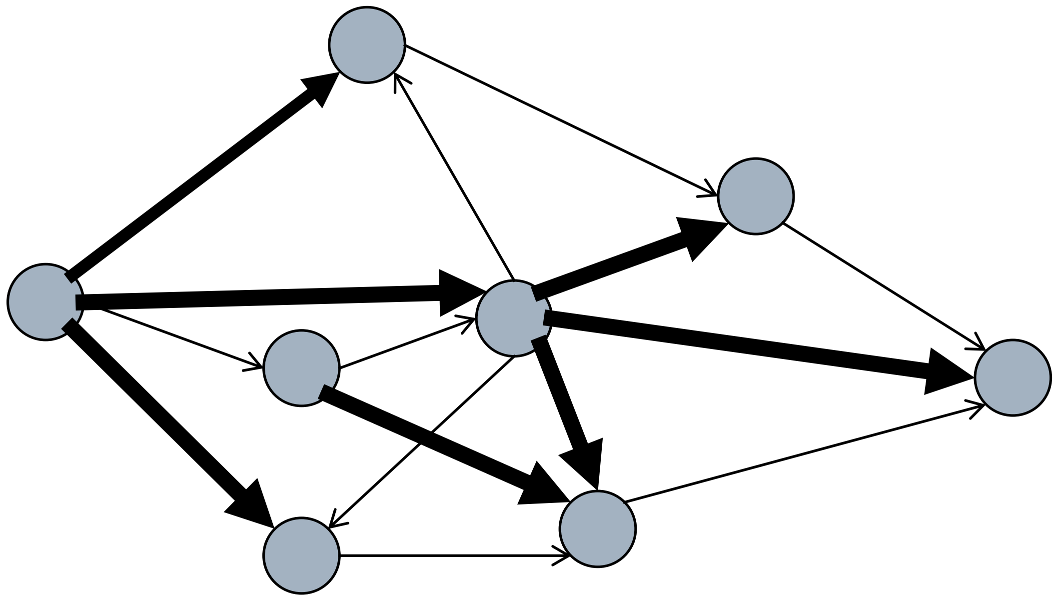
1	2	3
4	5	6
7	8	

Goal state

- $h_1(N)$ = number of misplaced numbered tiles
- $h_2(N)$ = sum of the (Manhattan) distance of every numbered tile to its goal position
- h_2 is more accurate than h_1

How to get admissible heuristic?





Relaxed problems {3.6.2}

- 减少了行动限制的问题称为**松弛问题**。松弛问题的状态空间图是原有状态空间的*超图*，原因是减少限制导致图中边的增加。
- 例如，如果八数码问题的行动描述如下：

  棋子可以从方格A移动到方格B，如果

  A与B水平或竖直相邻 **而且** B是空的，
- 去掉其中一个或者两个条件，生成三个松弛问题：

  (a) 棋子可以从方格A移动到方格B，如果A和B相邻。 (h2)

  (b) 棋子可以从方格A移动到方格B，如果B是空的。

  (c) 棋子可以从方格A移动到方格B。 (h1)

Relaxed problems {3.6.2}

- Key point
  - the optimal solution cost of a relaxed problem is no greater ($\leq$) than the optimal solution cost of the real problem

Relaxed problems {3.6.2}

- **松弛问题增加了状态空间的边。所以，一个松弛问题的最优解代价是原问题的可采纳的启发式。**
- 松弛问题本质上要能够不用搜索就可以求解
- **是一致的吗？**

Relaxed problems {3.6.2}

- 松弛问题增加了状态空间的边。所以，一个松弛问题的最优解代价是原问题的可采纳的启发式。
- 松弛问题本质上要能够不用搜索就可以求解
- 更进一步，由于得出的启发式是松弛问题的确切代价，那么它一定遵守三角不等式，因而是一致的

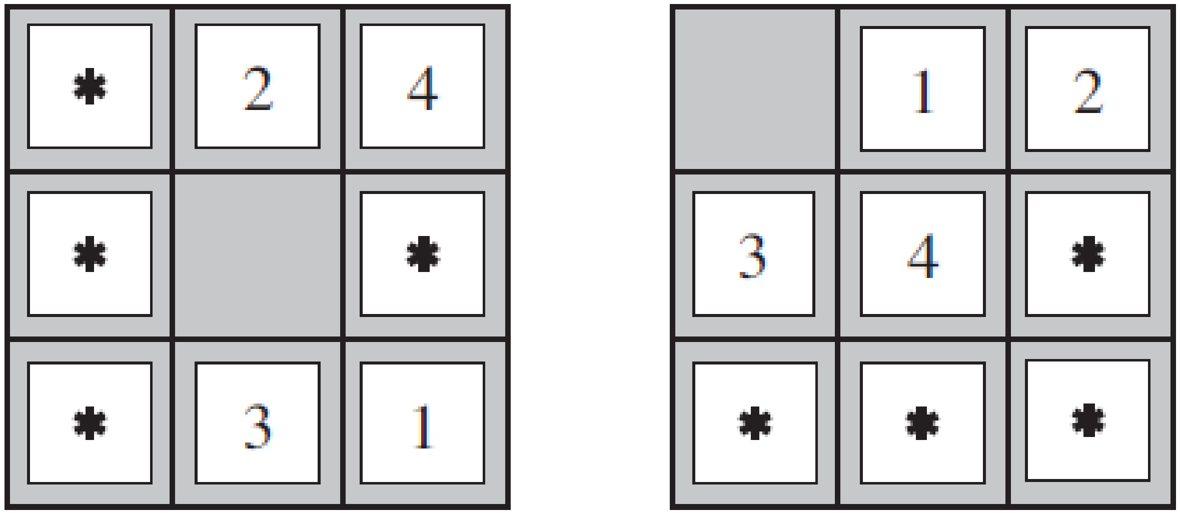
Relaxed problems {3.6.2}

- 如果可采纳启发式的集合h_1 ... h_m对问题是有效的，并且其中没有哪个比其他的更有优势，我们应该怎样选择？

Relaxed problems {3.6.2}

- 如果可采纳启发式的集合h_1 … h_m对问题是有效的，并且其中没有哪个比其他的更有优势，我们应该怎样选择？
- 可以定义新的启发式从而得到其中最好的：
- $h(n) = max\{h_1(n), \ldots, h_m(n)\}$。

sub-problem



Start State　　Goal State

- **子问题**的解代价(计算数字的移动以及*的移动总次数)小于完整问题的解代价。某些情况下比曼哈顿距离更准确。

sub-problem

- **模式数据库**：对每个可能的子问题实例存储解代价。八码问题中，每个子问题实例可以是4个棋子和一个空位组成的可能状态。另外4个棋子的移动也计算在解代价里。

sub-problem

- 可以构造多个模式数据库，例如1-2-3-4的模式数据库，5-6-7-8的模式数据库，2-4-6-8等的模式数据库。
- 每个数据库都能产生一个可采纳的启发式，这些启发式可以像前面所讲的那样取最大值的方式组合使用。
- 这种组合的启发式比曼哈顿距离要精确；求解随机的15数码问题时所生成的结点数要少1000倍。

sub-problem

- 考虑1-2-3-4数据库和5-6-7-8数据库，从它们得到的启发式是否可以相加？相加得到的启发式是否还是可采纳的？
- 答案是否定的，因为对于一给定状态，1-2-3-4子问题的解和5-6-7-8子问题的解可能有一些重复的移动

sub-problem

- 如果我们只记录涉及1-2-3-4的移动次数。这样很容易得出，两个子问题的代价之和仍然是求解整个问题的代价的下界。这就是**不相交的模式数据库**的思想。
- 用这样的数据库，我们可以在几毫秒内解决一个随机的15数码问题——与使用曼哈顿距离启发式相比生成的结点数减少了10,000倍。对于24数码问题减少的结点数以百万倍计。

learning

- 另一个方案则是从经验里**学习**。每个八数码问题的最优解都可提供学习实例。每个实例都包括解路径上的一个状态和从这个状态到达解的代价。一个学习算法从实例构造$h(n)$，预测搜索过程中所出现的其他状态的解代价。
- 如果能刻画给定状态的**特征**，**归纳学习方法**则是最可行的。例如，特征“不在位的棋子数”$x_1(n)$，“现在相邻但在目标状态中不相邻的棋子对数”$x_2(n)$，用线性组合构造$h(n)$：
  - $h(n) = c_1 x_1(n) + c_2 x_2(n)$。

Summary {3.7}

- Simplest form of problem specific knowledge is *heuristic*.
- Heuristic functions estimate costs of shortest paths
- Good heuristics can dramatically reduce search cost
- Greedy best-first search expands lowest h
  - incomplete and not always optimal
- A* with a consistent heuristic function has nice properties: completeness, optimality, no need to revisit states
- Admissible heuristics can be derived from exact solution of relaxed problems

summary

1 启发函数：什么是启发函数? 什么是启发式搜索?

2 贪婪最佳优先搜索，A*搜索

3 可采纳性、一致性、准确性、松弛问题：什么是可采纳的启发函数? 什么 是一致的启发函数? 如何设计可采纳的启发函数?