

Inteligência Artificial

ACH2016

Aula 5: Busca informada

Profa. Karina Valdivia Delgado
EACH-USP

Slides baseados em:

RUSSEL, S.; NORVIG, P. Artificial Intelligence: A modern approach. Third Edition, 2010. Capítulo 3.

Slides da Profa. Leliane Nunes de Barros

Slides do Prof. Edirlei Soares de Lima

Métodos de busca

Como podemos modificar os algoritmos gerais de busca para incluir mais conhecimento sobre o problema?

isto é, conhecimento que vá além do que está na descrição de estado, ações, função custo e teste de estado meta

Métodos de busca

- **Busca sem informação ou cega ou exaustiva ou força bruta ou sistemática :**
 - Não sabe qual o melhor nó da fronteira a ser expandido. Apenas distingue o estado objetivo dos não objetivos.
- **Busca informada ou heurística:**
 - Estima qual o melhor nó da fronteira a ser expandido com base em funções heurísticas. Sabem se um estado não objetivo é “mais promissor”.
- **Busca local:**
 - Operam em um único estado e movem-se para a vizinhança deste estado.

Busca Heurística

- **Algoritmos de Busca Heurística:**
 - Busca Gulosa
 - A^*
- A busca heurística leva em conta o **objetivo** para decidir qual caminho escolher.
- Usa conhecimento específico para guiar o processo de busca, além da definição do problema em si.

Busca Heurística

- **Função Heurística (h)**

$h(n)$: custo **estimado** do caminho de menor custo do estado do nó n para um estado objetivo.

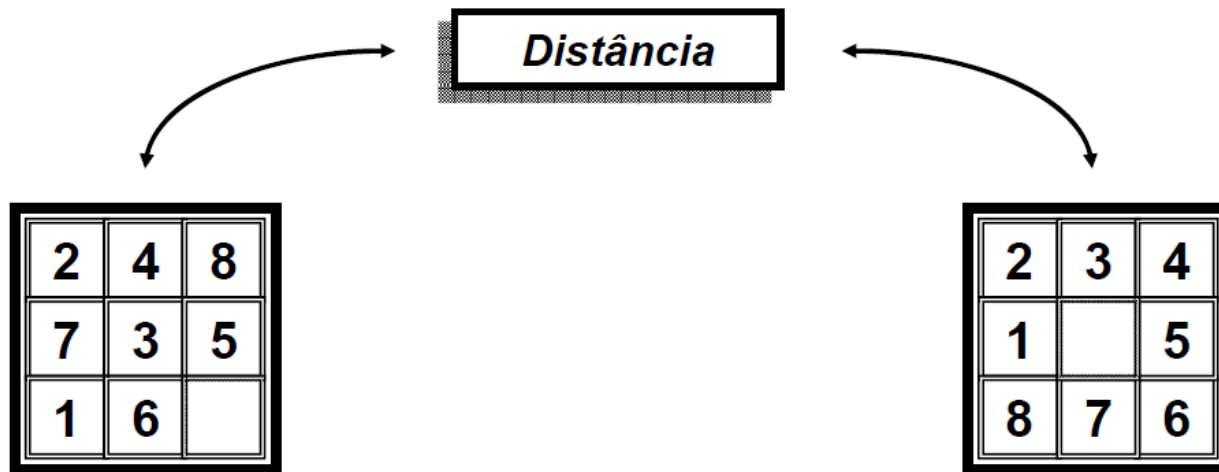
- São específicas para cada problema.

- **Exemplo:**

- Encontrar a rota mais curta entre duas cidades:

$h(n)$ = distância em linha reta direta entre o estado do nó n e o nó final.

Função Heurística para o 8-puzzle



$$h_1 \left(\begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & 6 & \\ \hline \end{array} \right) = ?$$

Busca Heurística

- **Algoritmos de Busca Heurística:**

- Busca Gulosa
- A^*

Para explicar os algoritmos, consideramos uma abordagem mais geral chamada de busca de melhor escolha.

Busca de melhor escolha

- Em que o nó é selecionado para expansão com base em uma **função de avaliação**, $f(n)$.
- O nó com a menor avaliação será expandido primeiro.
- A implementação da busca em grafos de melhor escolha é idêntica a busca de custo uniforme, exceto pelo uso de f em vez de g para ordenar a fila de prioridade.

Busca gulosa

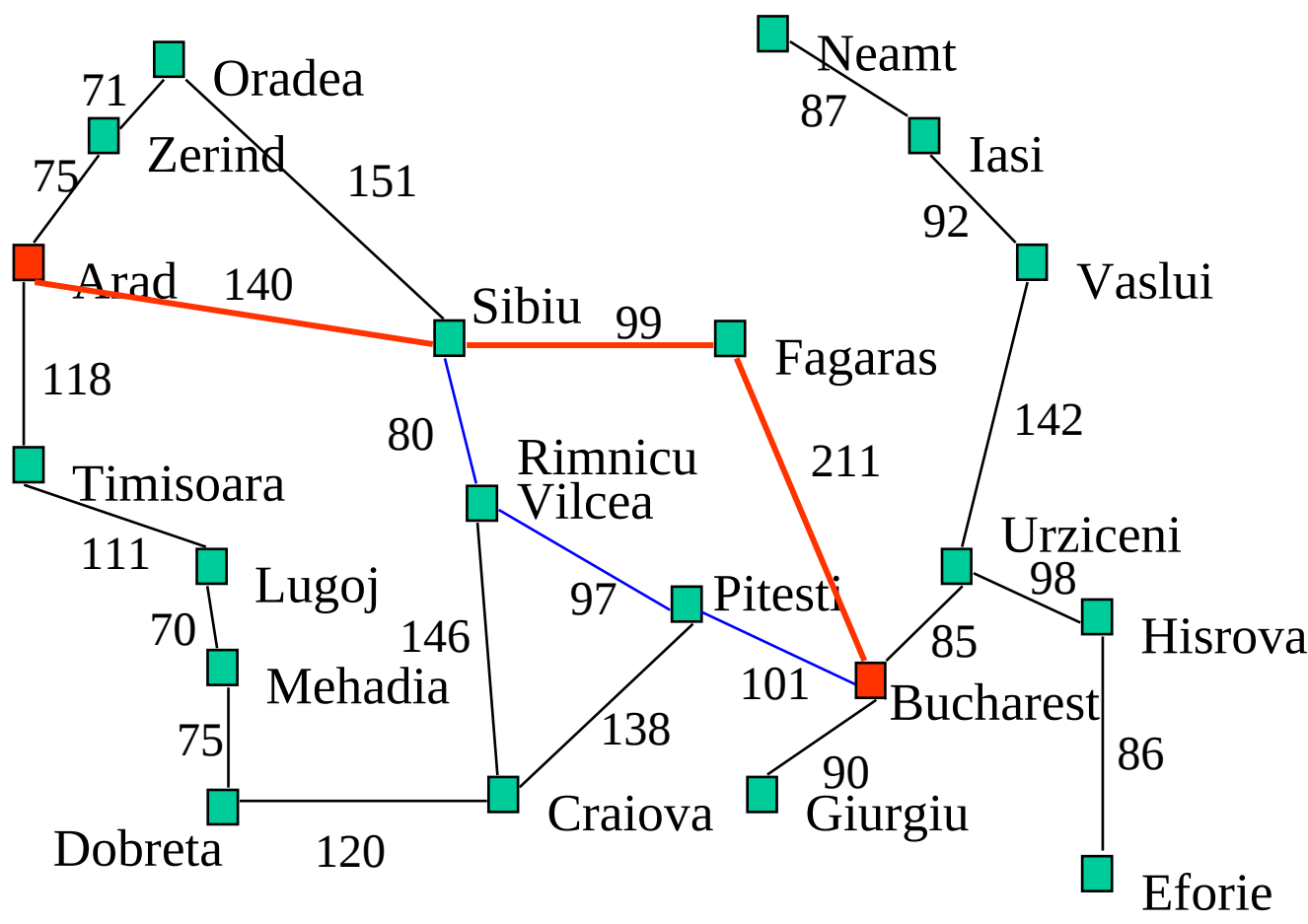
- **Estratégia:**

- Expande o nó que se encontra mais próximo do objetivo, desta maneira é provável que a busca encontre uma solução rapidamente.
- Avalia os nós usando apenas a função heurística:

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

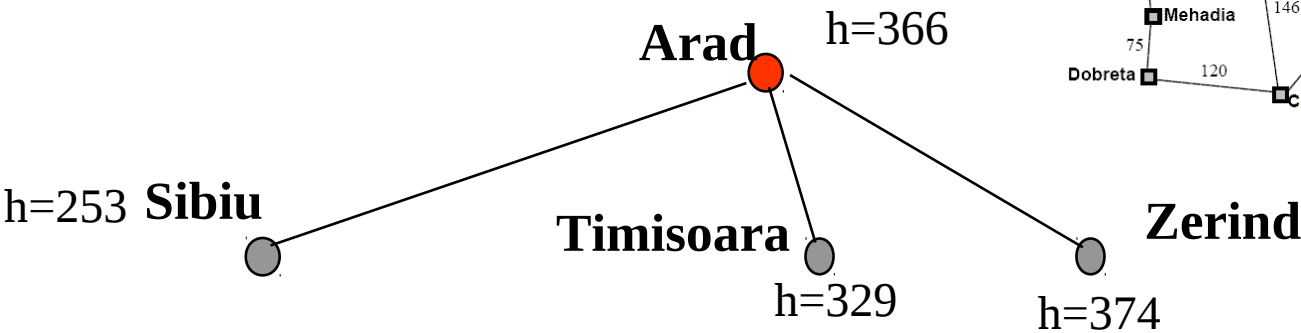
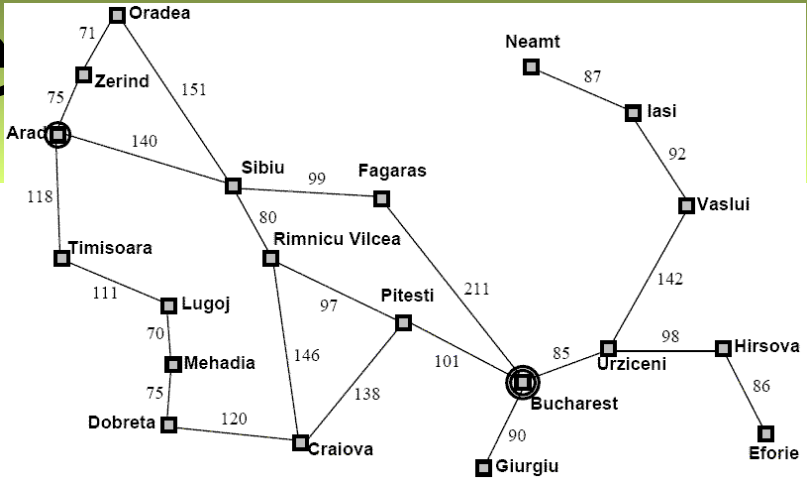
$h(n) = 0$ se n é o estado meta

Busca gulosa: exemplo



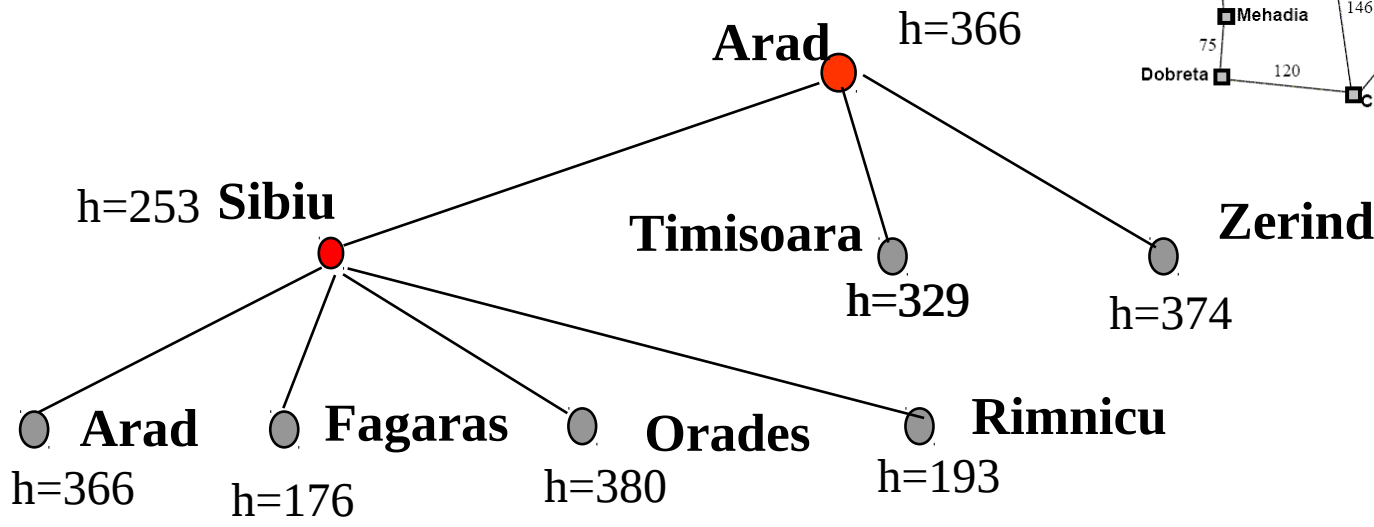
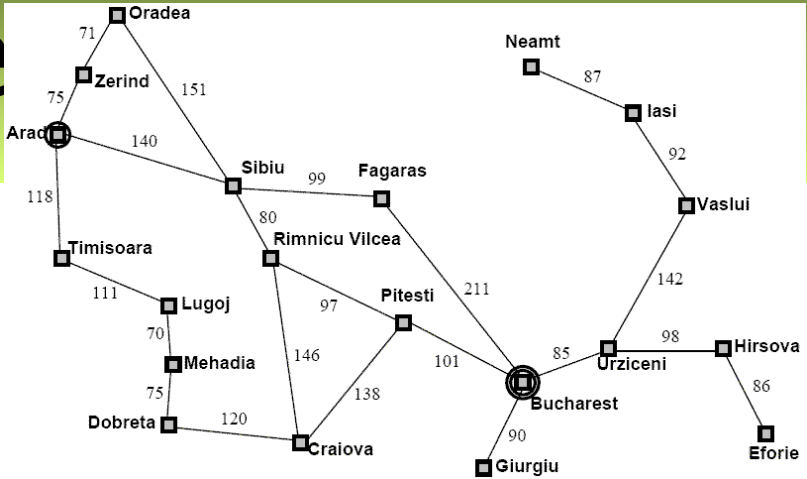
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Vaslui	199
Zerind	374

Busca gulosa: exe



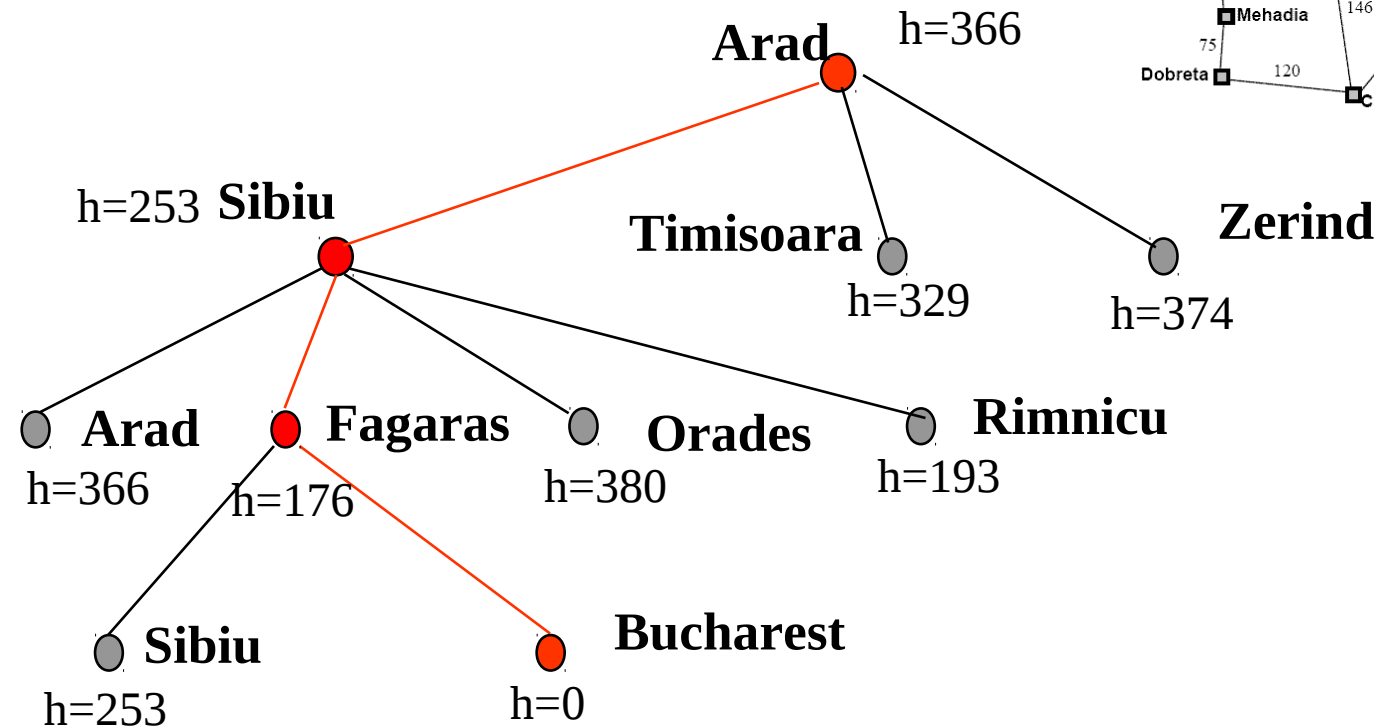
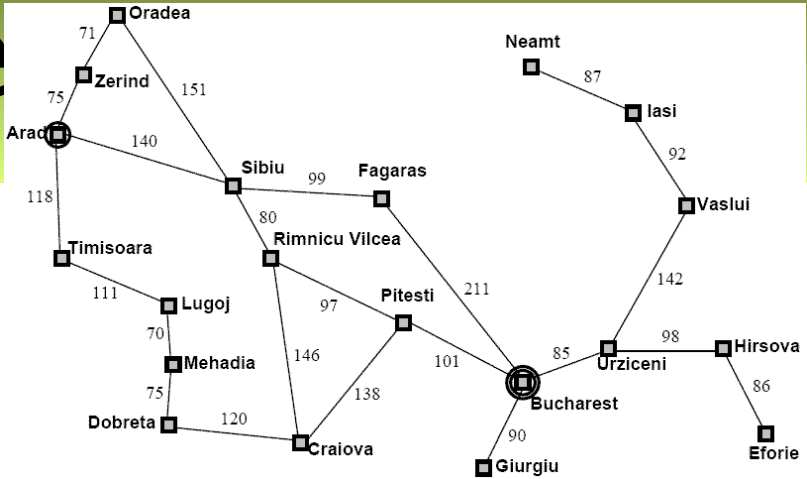
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Vaslui	199
Zerind	374

Busca gulosa: exe



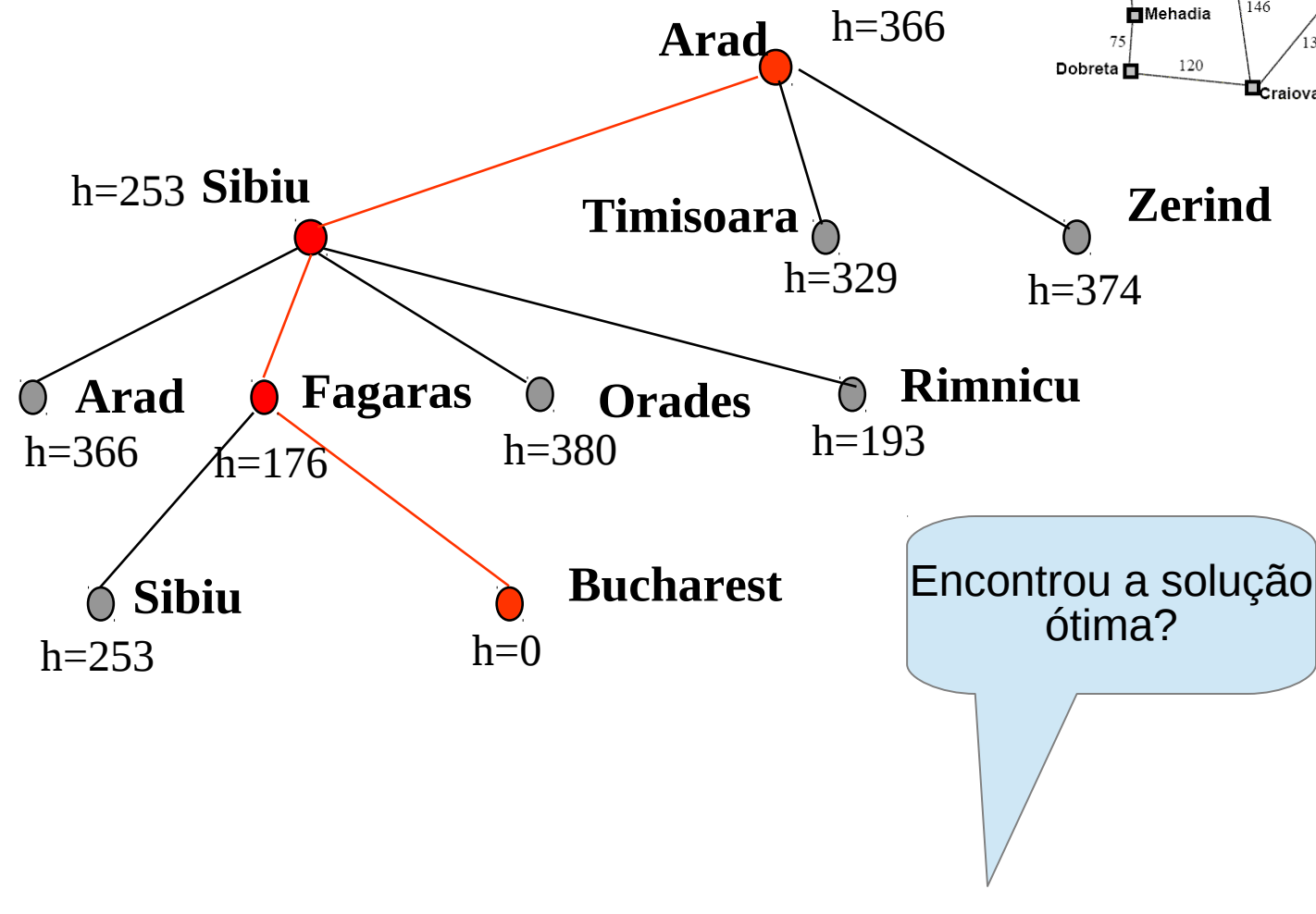
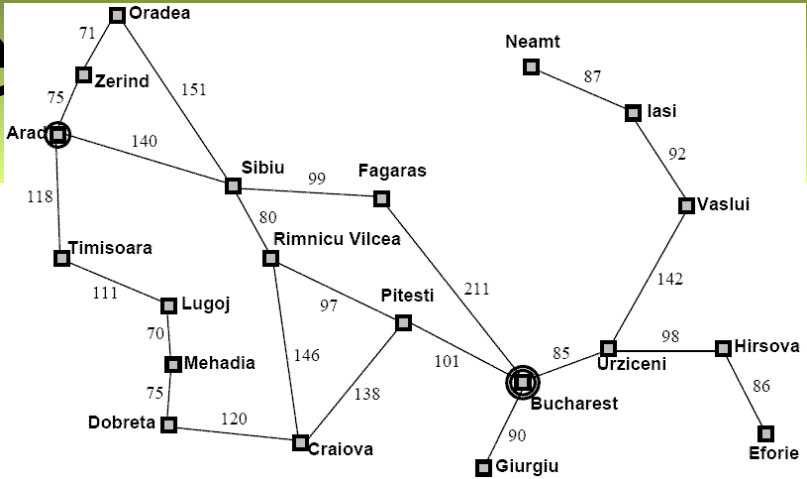
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Vaslui	199
Zerind	374

Busca gulosa: exe



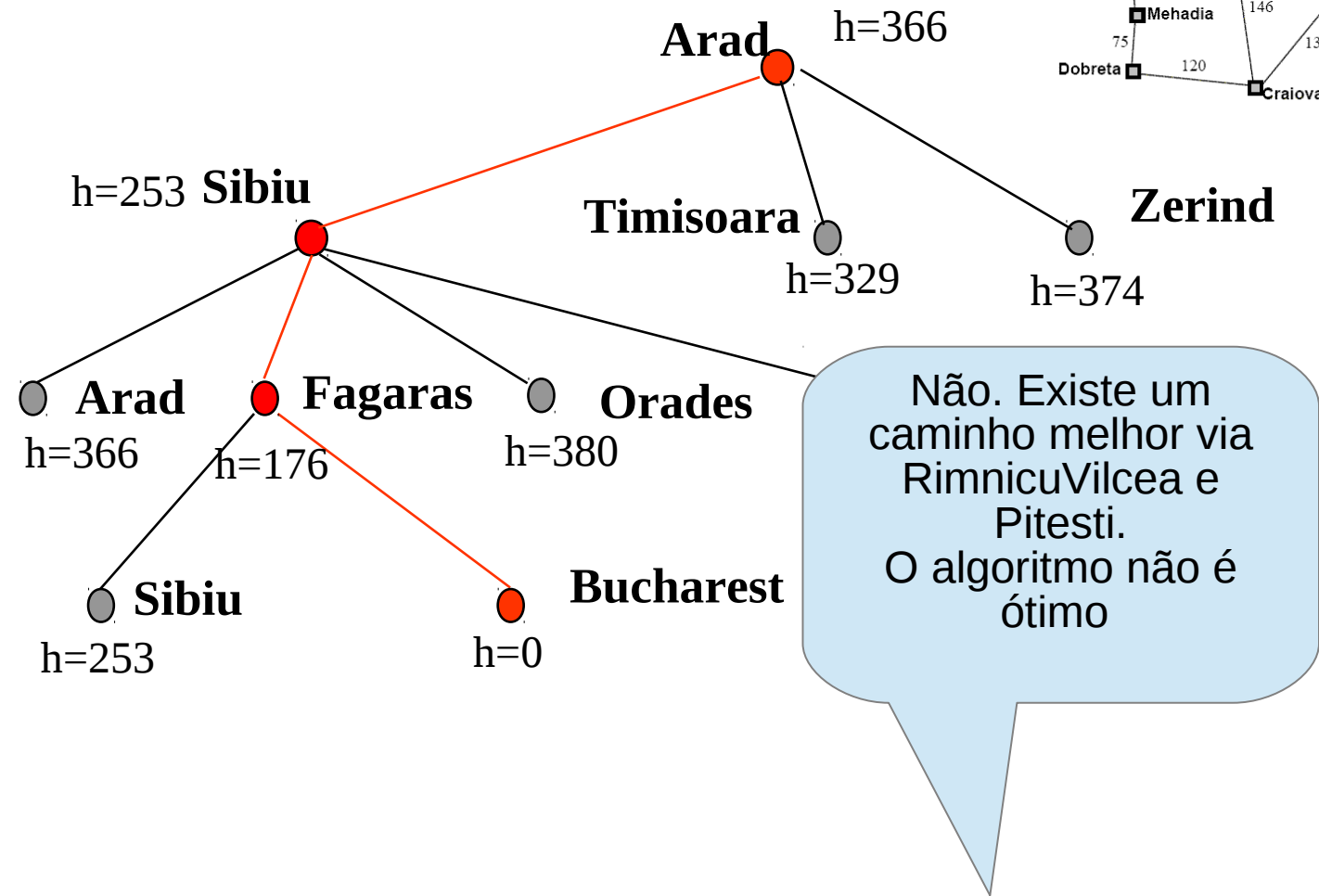
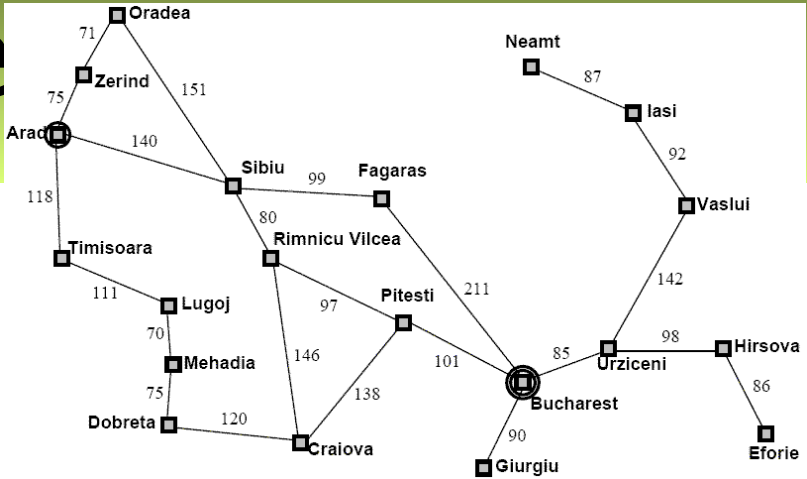
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Vaslui	199
Zerind	374

Busca gulosa: exe



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	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Vaslui	199
Zerind	374

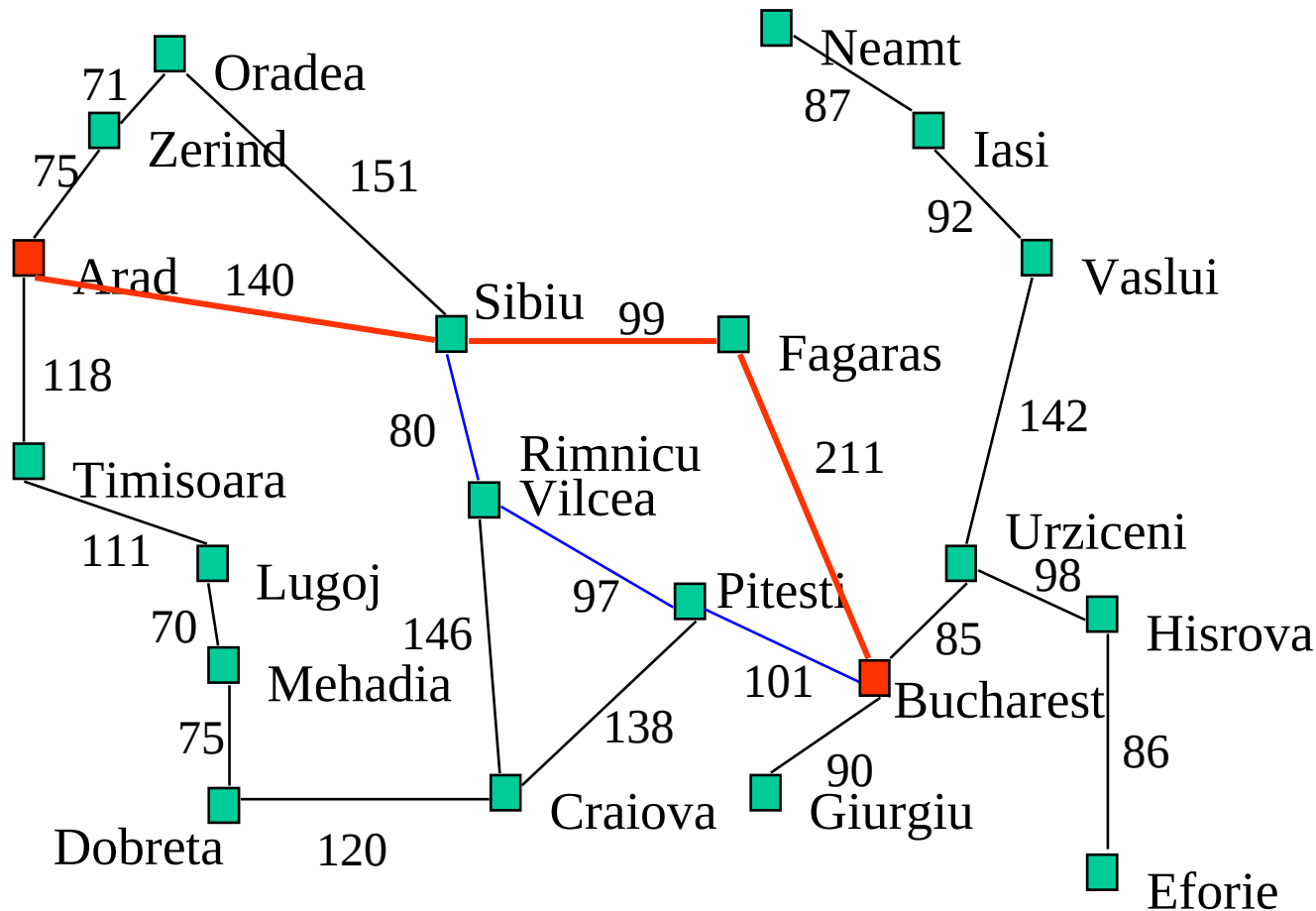
Busca gulosa: exe



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	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Vaslui	199
Zerind	374

Busca gulosa: exemplo

Iasi para Fagaras

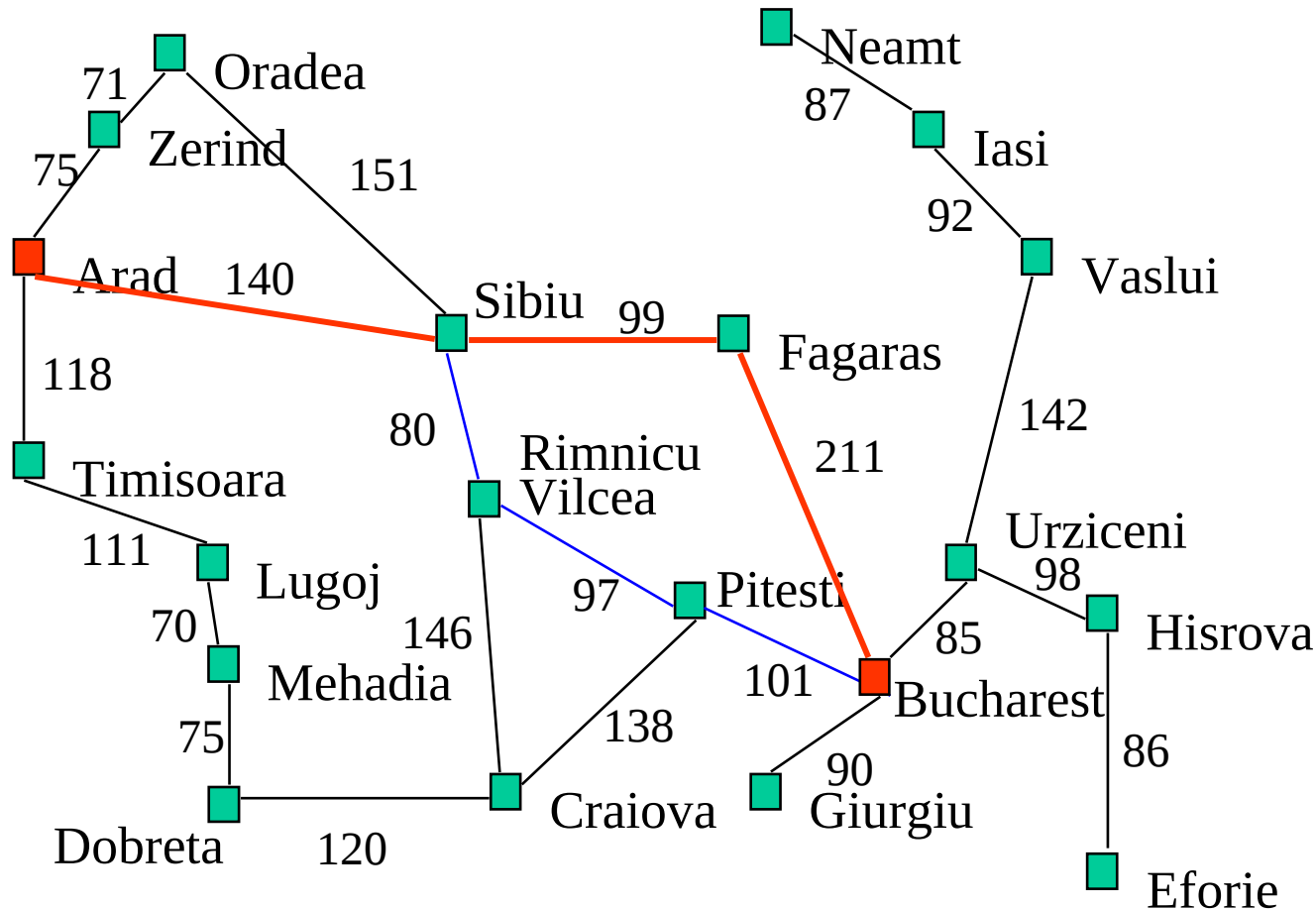


Arad	160
Bucharest	176
Iasi	50
Neamt	80
Vaslui	90
...	

Busca gulosa: exemplo

Se usamos busca em árvore (sem detecção de estados repetidos), o algoritmo ficará em loop

Iasi para Fagaras



Arad	160
Bucharest	176
Iasi	50
Neamt	80
Vaslui	90
...	

Busca gulosa

- **Não é ótima**
- **Não é completa:** Pode entrar em loop se não detectar a expansão de estados repetidos.

Busca A*

- **Estratégia:**

- Combina o custo do caminho $g(n)$ com o valor da heurística $h(n)$:

$$f(n) = g(n) + h(n)$$

$g(n)$ = custo do caminho do nó inicial até o nó n

$h(n)$ = valor da heurística do nó n até um nó objetivo

$f(n)$ = custo total estimado do caminho que passa por n para atingir o estado meta

- **É a técnica de busca mais conhecida de busca de melhor escolha.**

Busca A^*

- **Estratégia:**

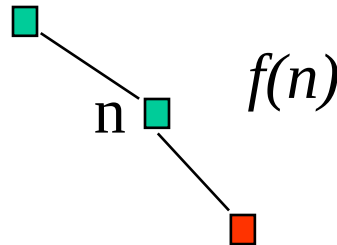
- Combina o custo do caminho $g(n)$ com o valor da heurística $h(n)$:

$$f(n) = g(n) + h(n)$$

$g(n)$ = custo do caminho do nó inicial até o nó n

$h(n)$ = valor da heurística do nó n até um nó objetivo

$f(n)$ = custo total estimado do caminho que passa por n para atingir o estado meta



- **É a técnica de busca mais conhecida de busca de melhor escolha.**

mplo

$$f(n) = g(n) + h(n)$$

$$f=140+253$$
$$=393$$

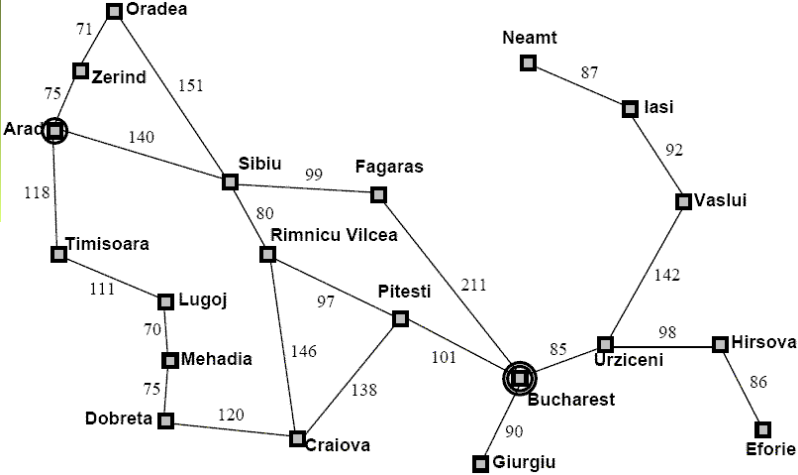
Sibiu

Timisoara

$$f=118+329$$
$$=447$$

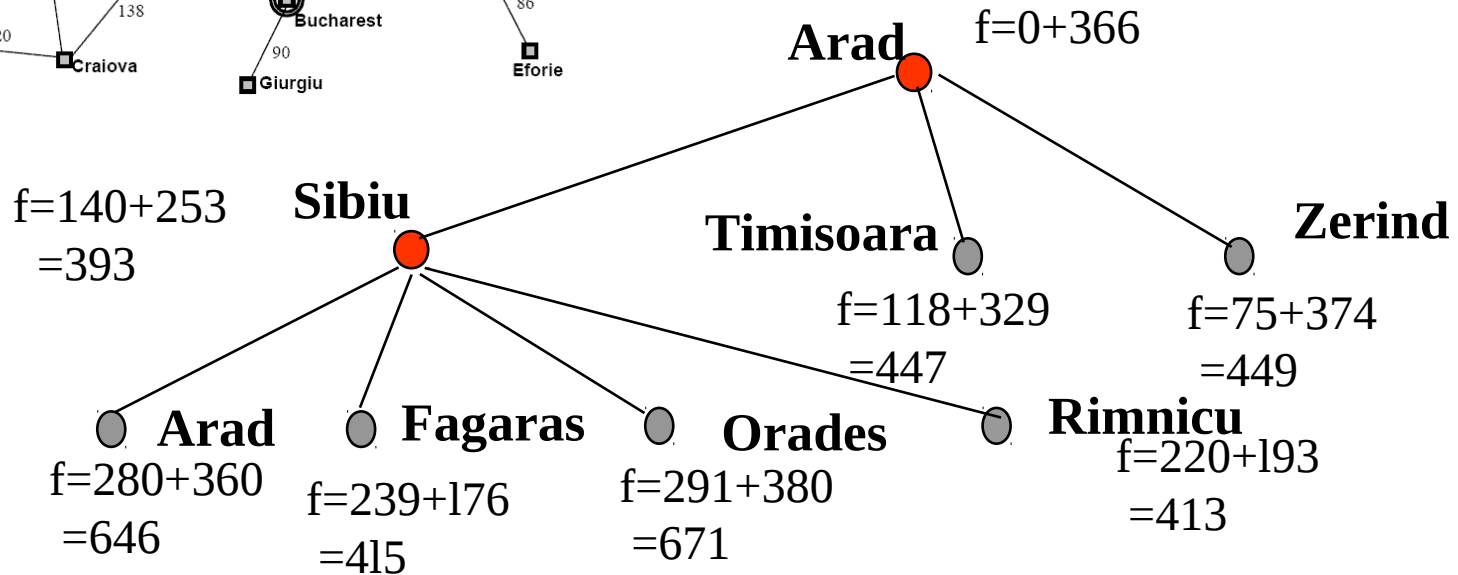
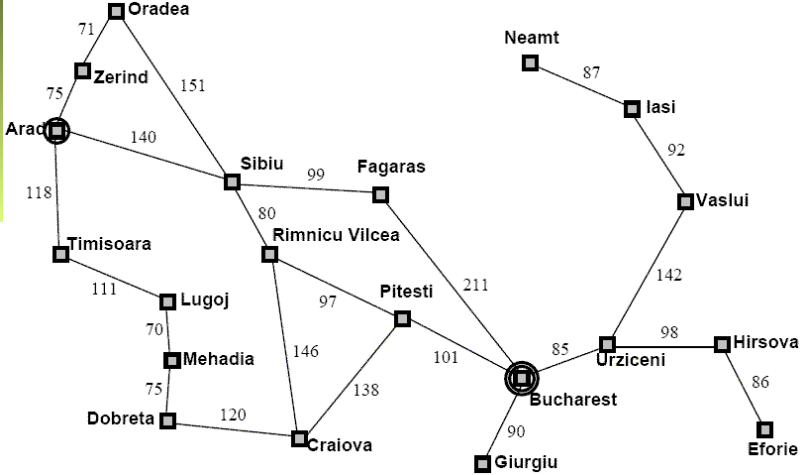
Zerind

$$f=75+374$$
$$=449$$



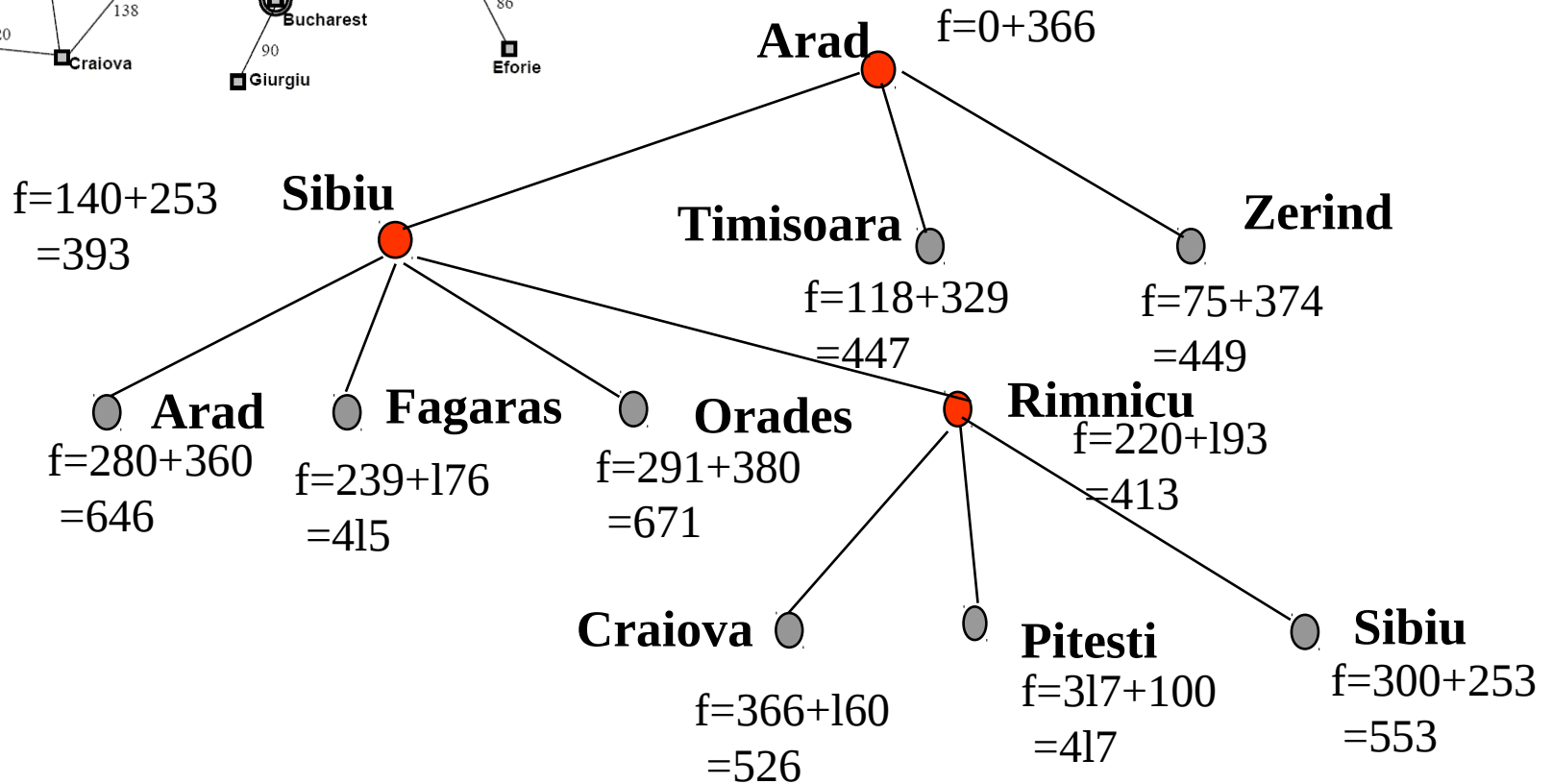
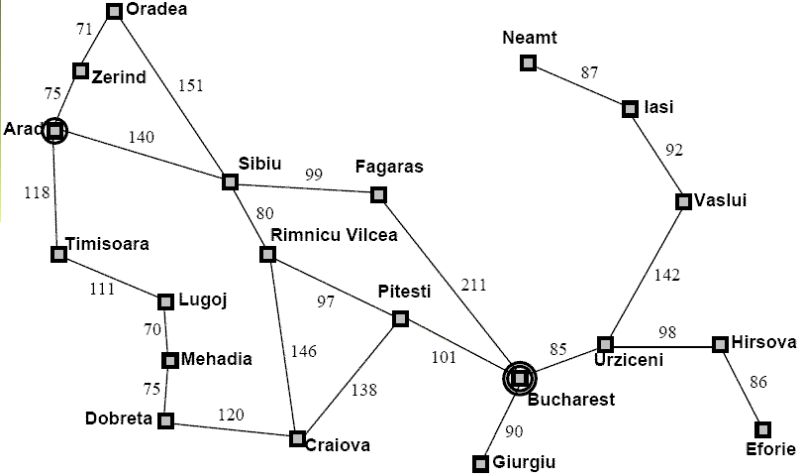
mplo

$$f(n) = g(n) + h(n)$$



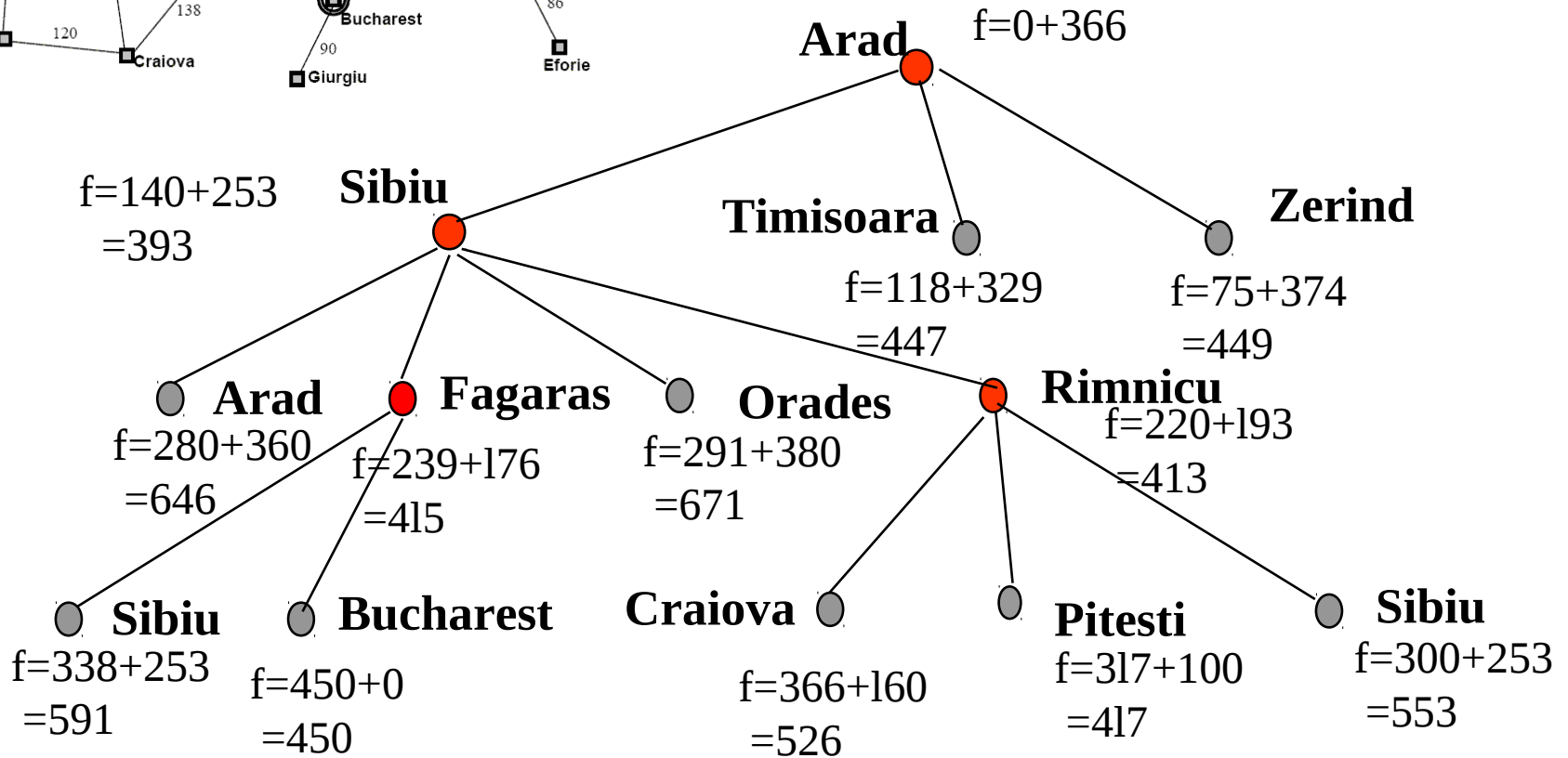
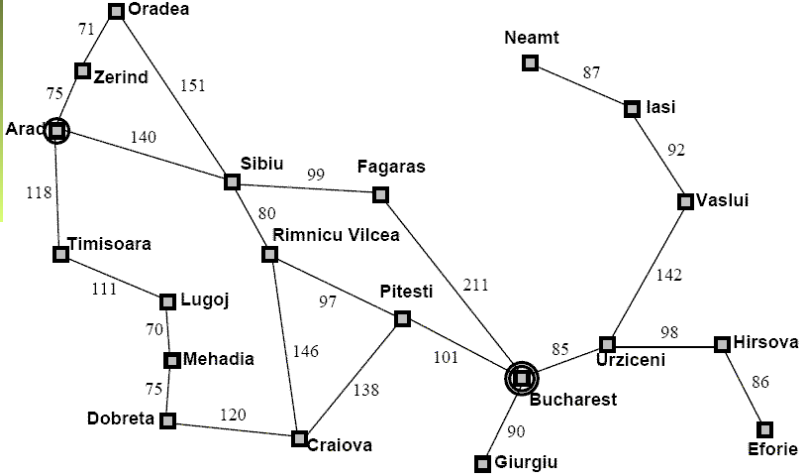
em plo

$$f(n) = g(n) + h(n)$$



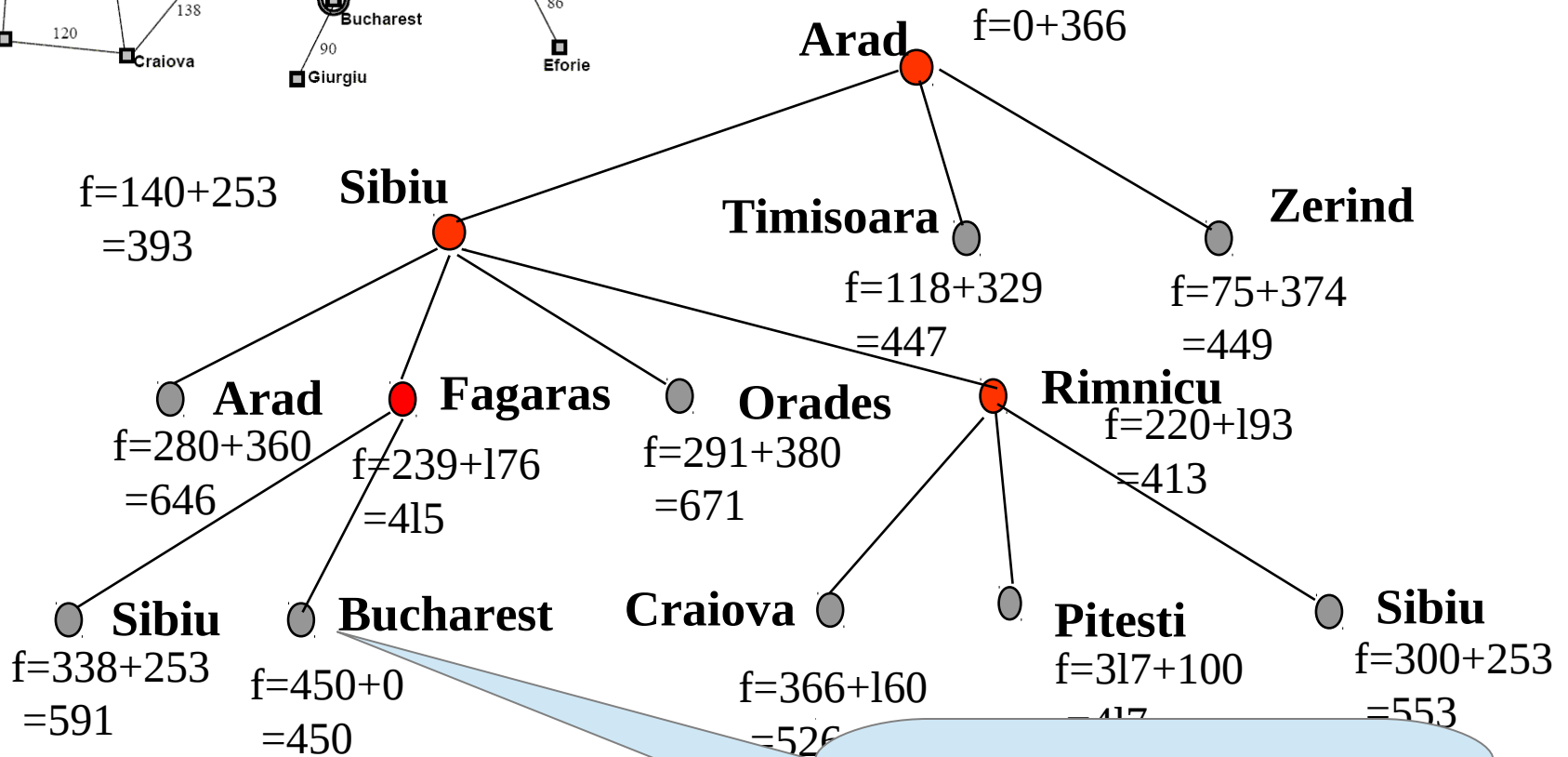
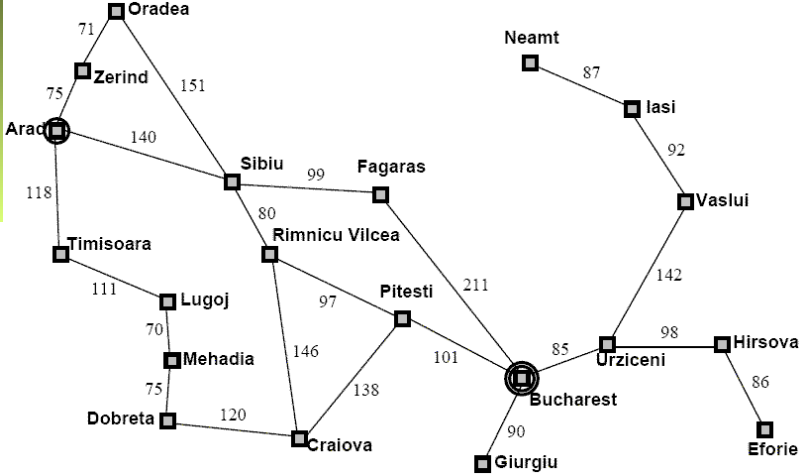
mplo

$$f(n) = g(n) + h(n)$$



Exemplo

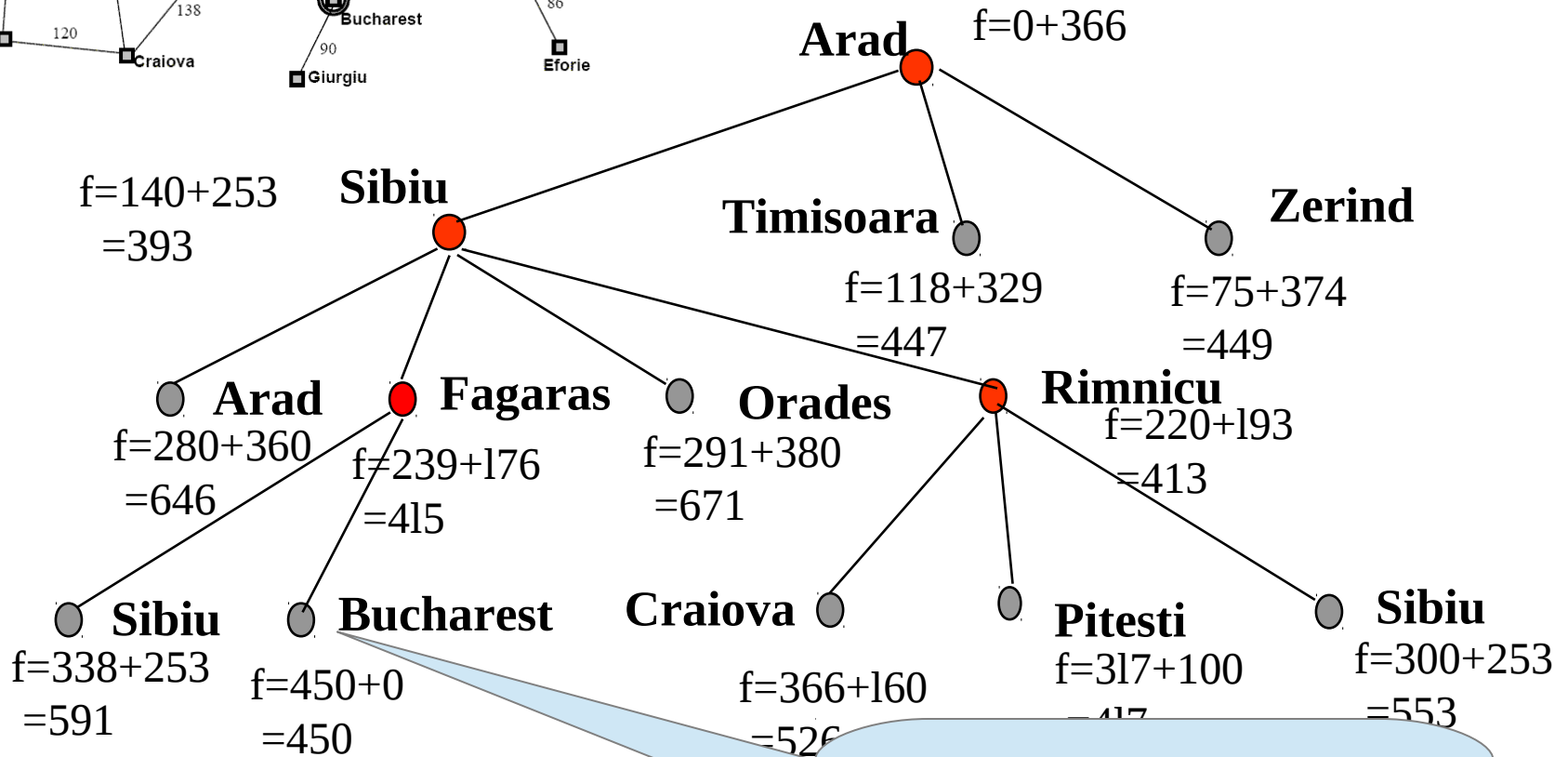
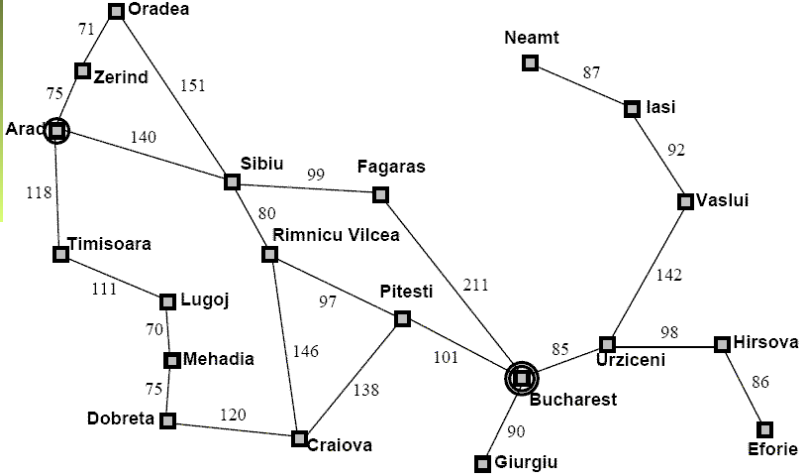
$$f(n) = g(n) + h(n)$$



Será que o algoritmo para?

Exemplo

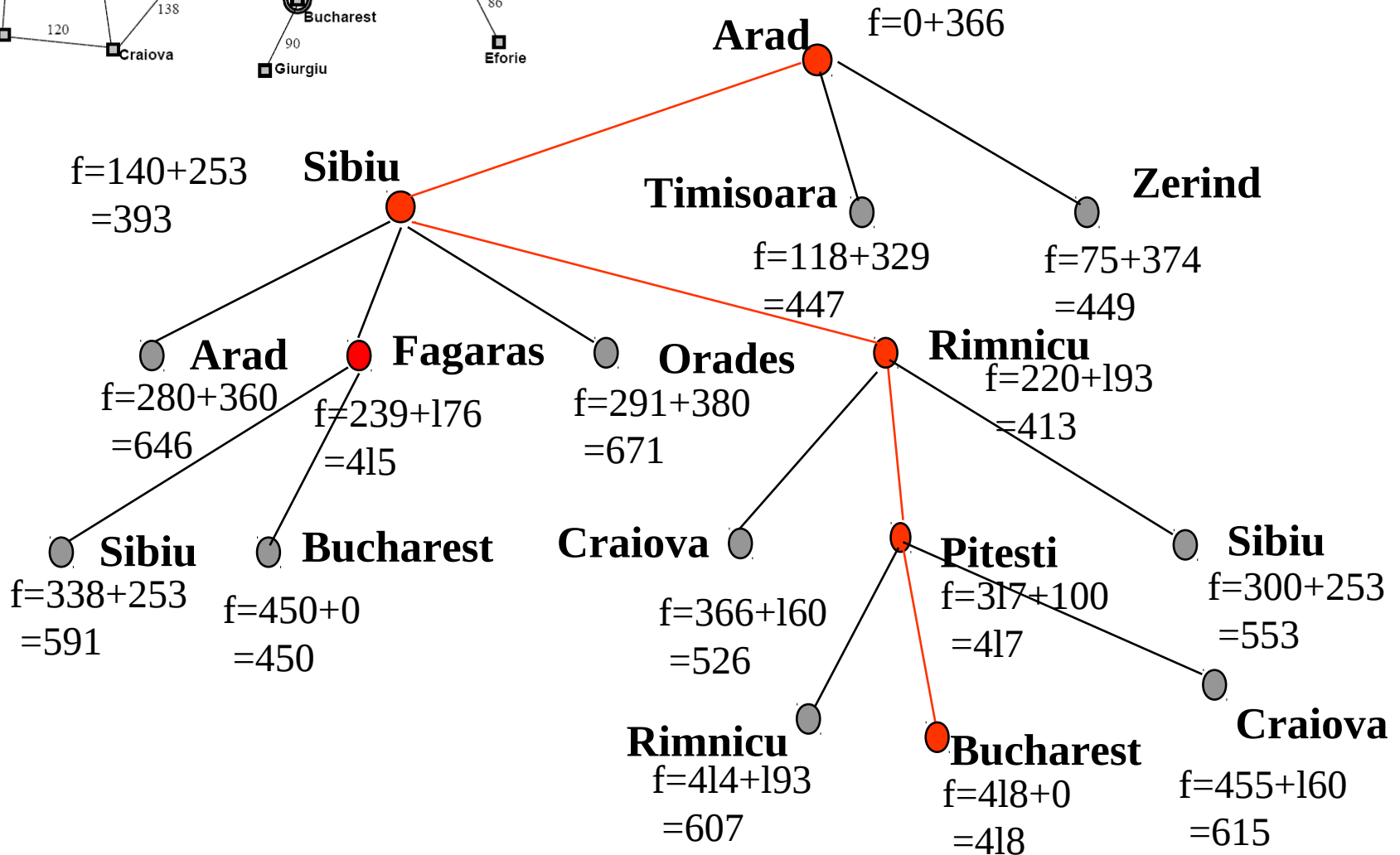
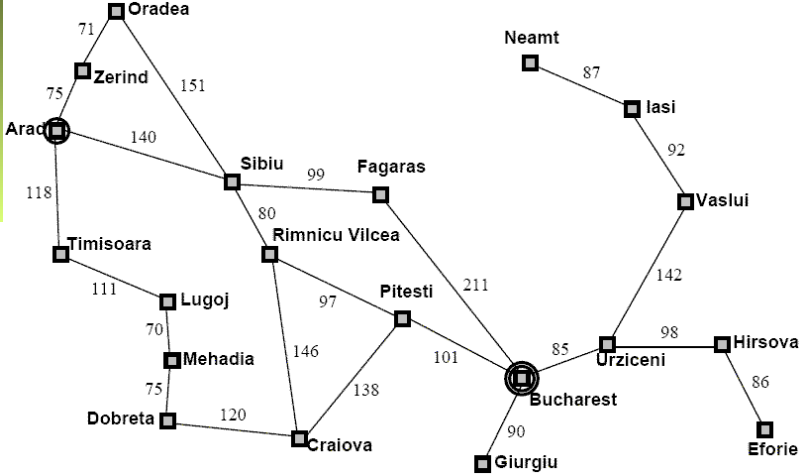
$$f(n) = g(n) + h(n)$$



Será que o algoritmo para?
Não, pode haver uma outra
solução através de Pitesti.

em plo

$$f(n) = g(n) + h(n)$$



Busca A^*

- A estratégia é **completa** e **ótima** desde que $h(n)$ satisfaça certas condições.
- O algoritmo é idêntico à busca de custo uniforme, exceto que A^* usa $g+h$ em vez de g .

Busca A*: condições de otimalidade

$h(n)$ deve ser uma **heurística admissível**:

- $h(n)$ não deve superestimar o custo real $h^*(n)$ para se alcançar a solução. Isto é:

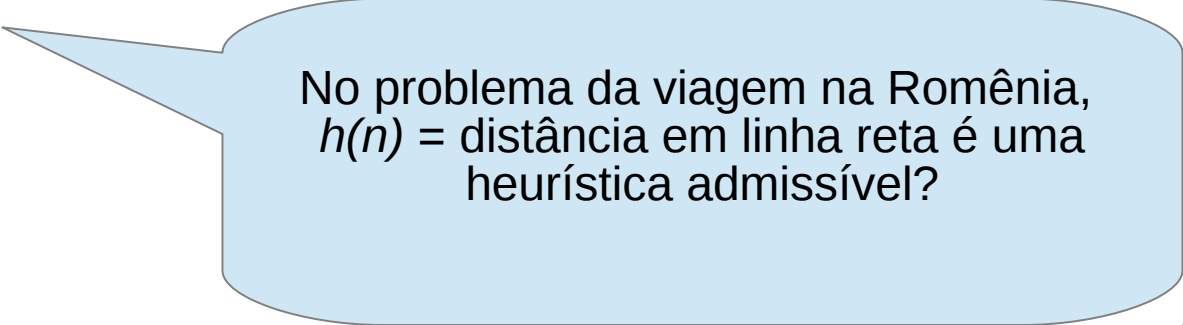
$h(n) \leq h^*(n)$, para todo n , sendo $h^*(n)$ o custo real a partir de n

Busca A*: condições de otimalidade

$h(n)$ deve ser uma **heurística admissível**:

- $h(n)$ não deve superestimar o custo real $h^*(n)$ para se alcançar a solução. Isto é:

$h(n) \leq h^*(n)$, para todo n , sendo $h^*(n)$ o custo real a partir de n

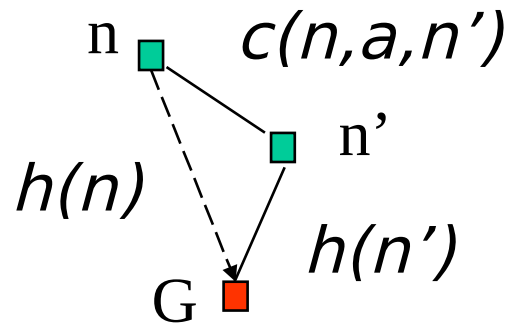


No problema da viagem na Romênia,
 $h(n)$ = distância em linha reta é uma
heurística admissível?

Busca A*: condições de otimalidade

$h(n)$ deve ser **consistente** (mais forte que admissibilidade):

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



Otimidade de A^*

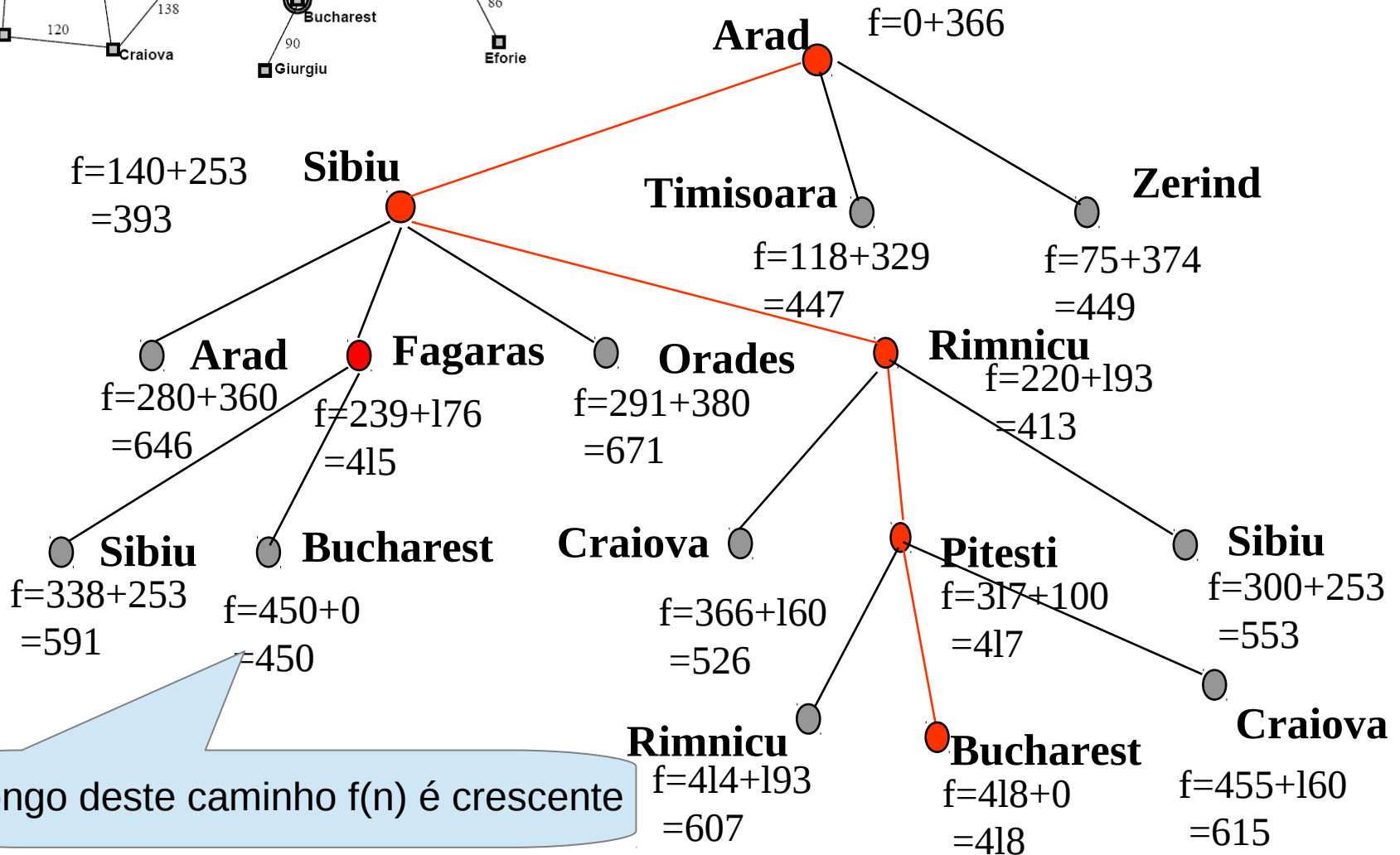
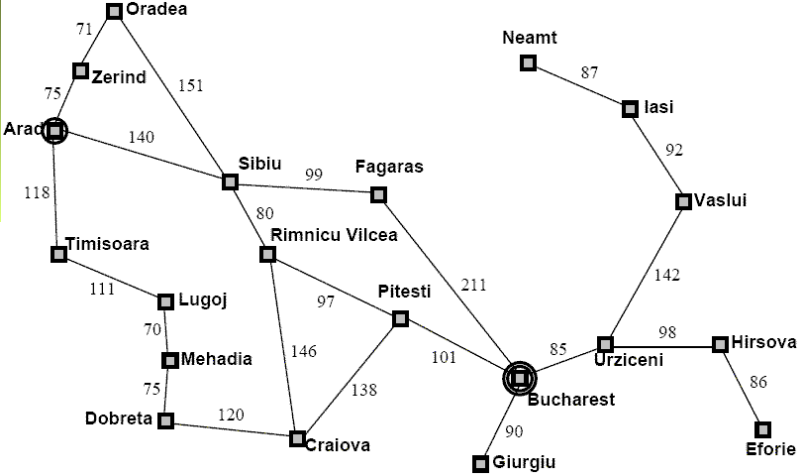
Busca A^* com detecção de estados repetidos é ótima se $h(n)$ for consistente.

Demonstração:

1. Se $h(n)$ for *consistente* \Rightarrow os valores de $f(n)$ ao longo de qualquer caminho serão *crescentes*

Exemple

$f(n) = g(n) + h(n)$



Ao longo deste caminho $f(n)$ é crescente

Otimidade de A^*

Busca com detecção de estados repetidos é ótima se $h(n)$ for consistente.

Demonstração:

1. Se $h(n)$ for *consistente* \Rightarrow os valores de $f(n)$ ao longo de qualquer caminho serão *crescentes*

Suponha que n' é sucessor de n : $g(n')=g(n)+c(n,a,n')$, teremos:

$$f(n')=g(n')+h(n')=g(n)+c(n,a,n')+h(n') \geq g(n)+h(n)=f(n)$$

$$f(n') \geq f(n)$$

Otimidade de A^*

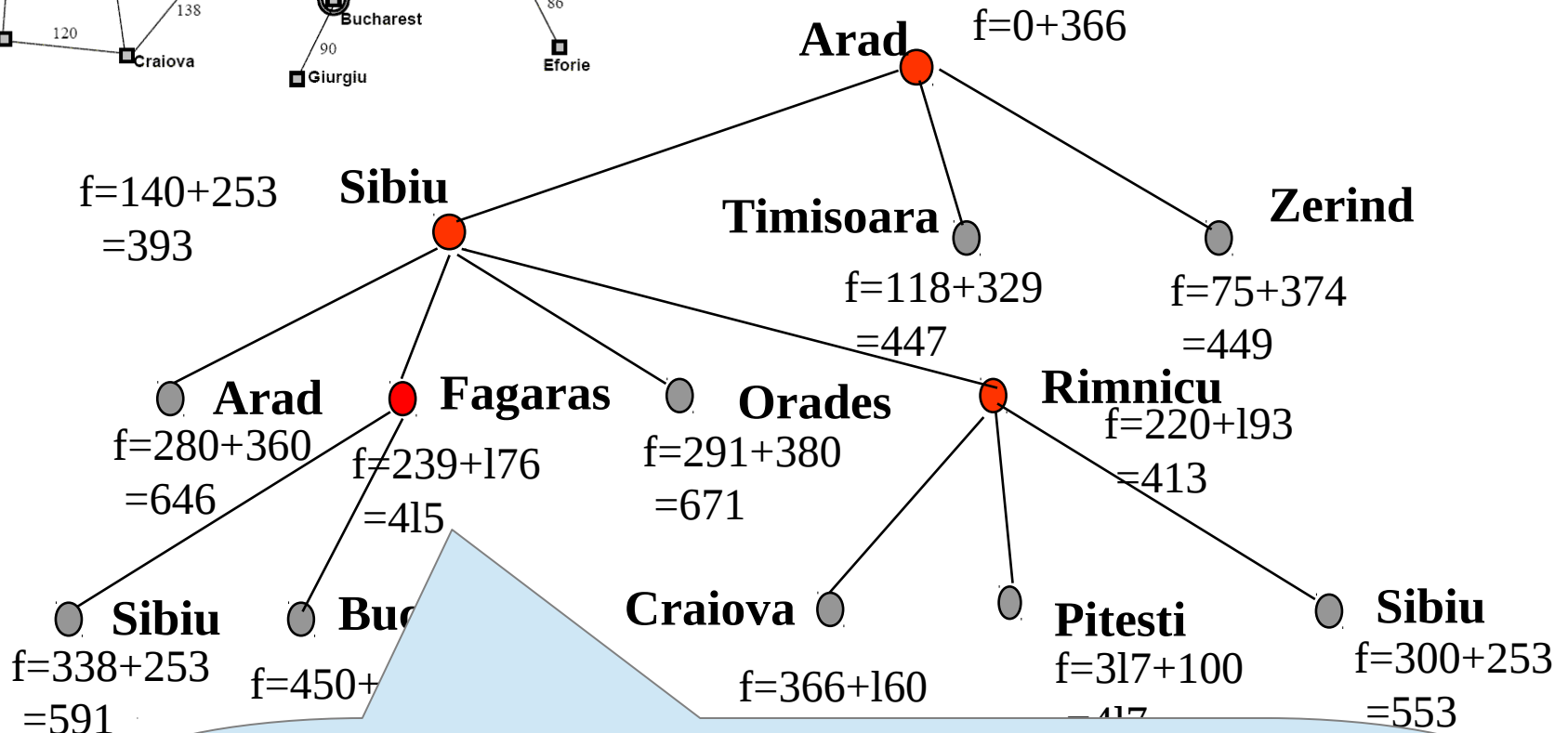
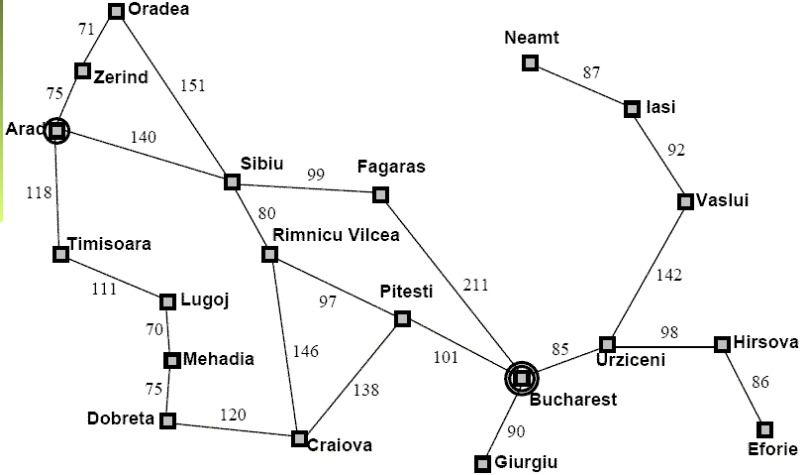
Busca com detecção de estados repetidos é ótima se $h(n)$ for consistente.

Demonstração:

2. Sempre que A^ seleciona um nó n para expansão, o caminho ótimo para aquele nó foi encontrado.*

Exemplo

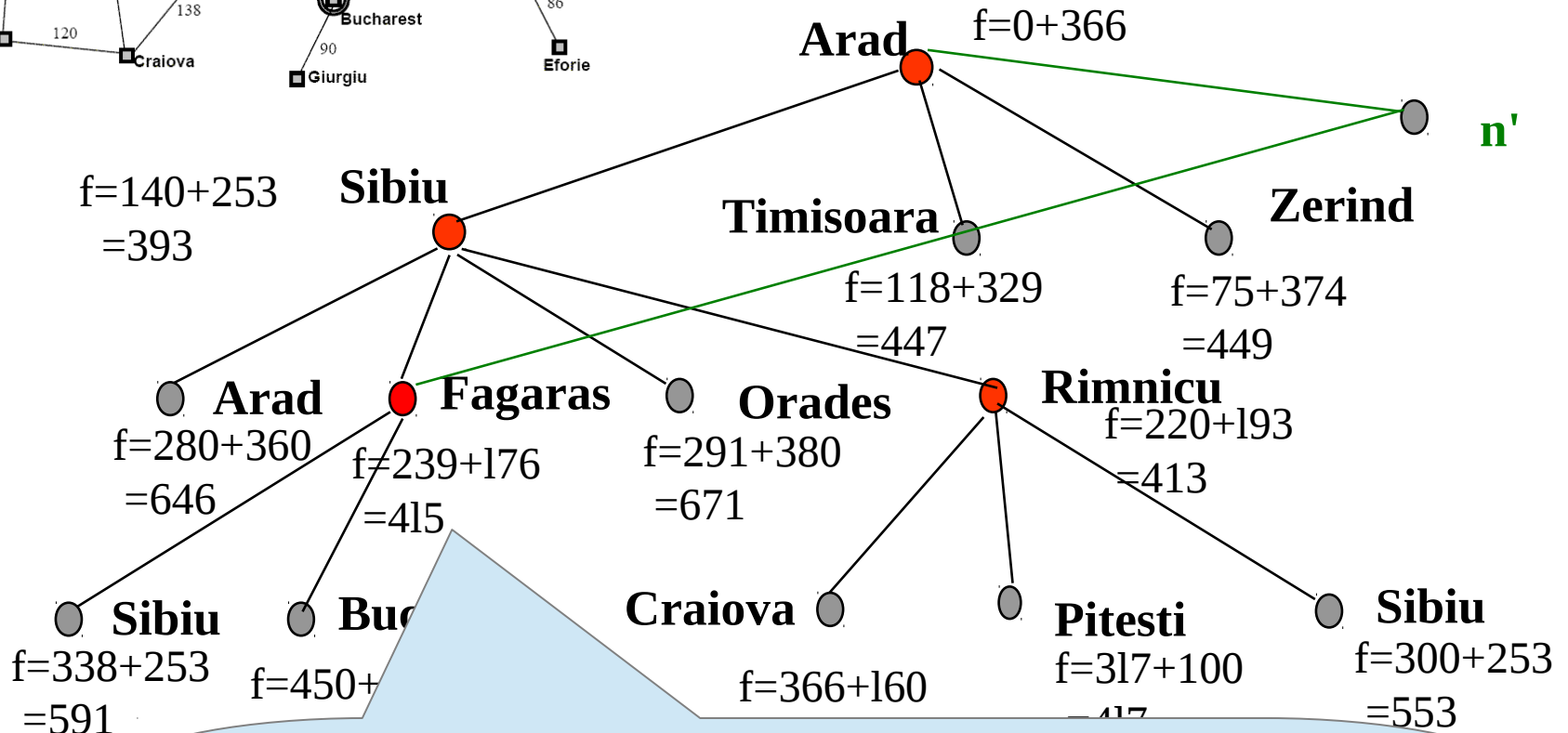
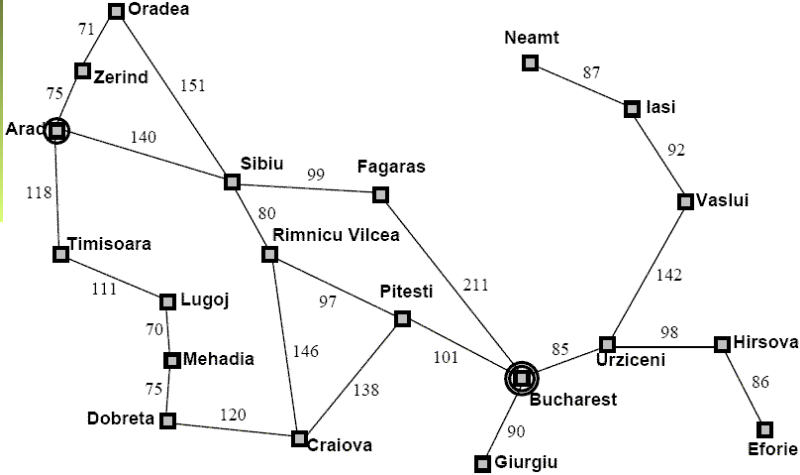
$$f(n) = g(n) + h(n)$$



Quando A* seleciona o nó Fagaras para expansão, o caminho ótimo para aquele nó foi encontrado.

Exemplo

$$f(n) = g(n) + h(n)$$



É impossível que exista outro nó na borda n' no caminho ótimo do nó inicial até n , porque f é crescente. Se existir ele teria sido escolhido antes.

Otimidade de A^*

Busca com detecção de estados repetidos é ótima se $h(n)$ for consistente.

Demonstração:

2. Sempre que A^* seleciona um *nó n para expansão, o caminho ótimo para aquele nó foi encontrado.*

É impossível que exista outro nó na borda n' no caminho ótimo do nó inicial até n , porque f é crescente. Se existir ele teria sido selecionado em primeiro lugar.

Otimidade de A^*

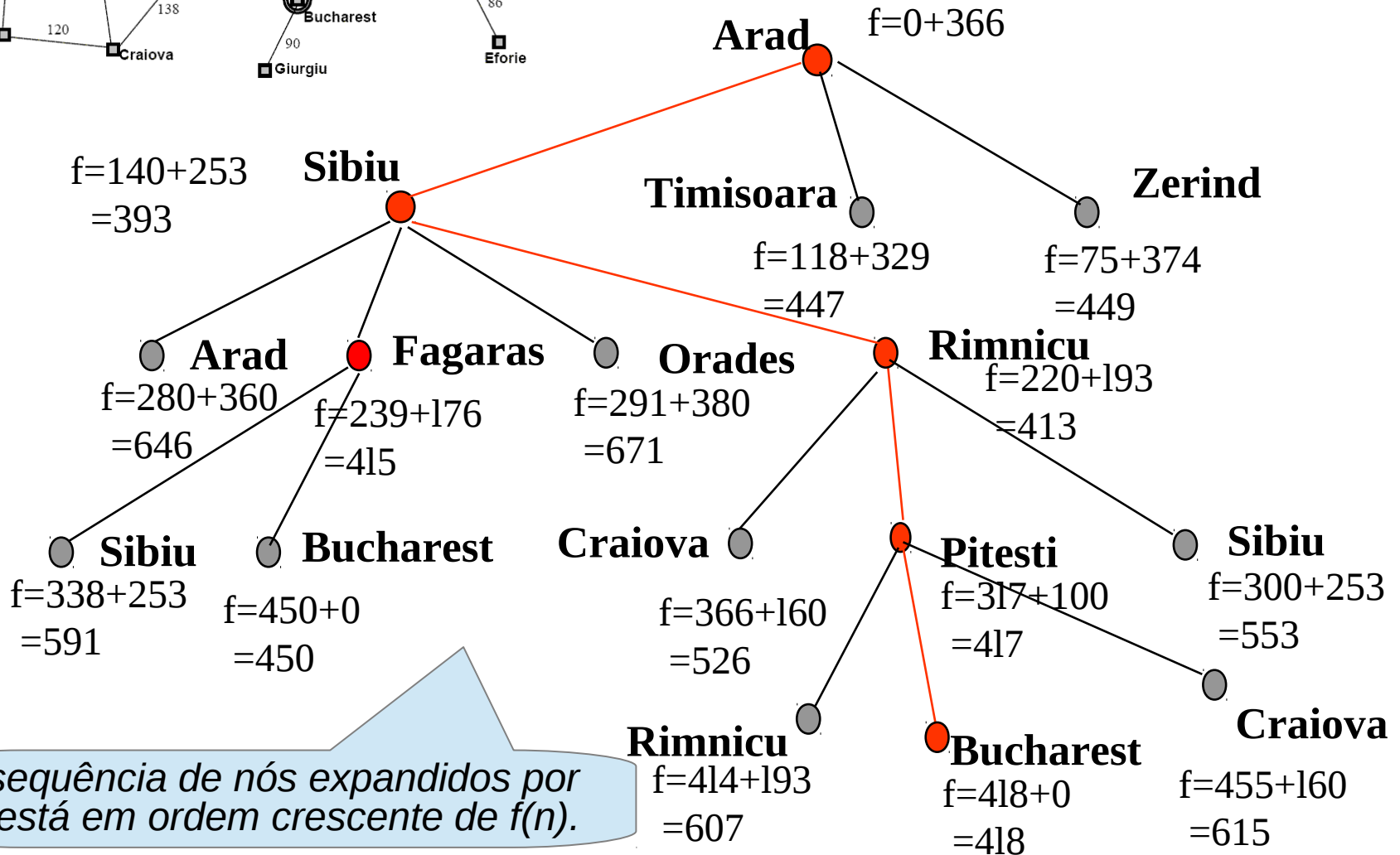
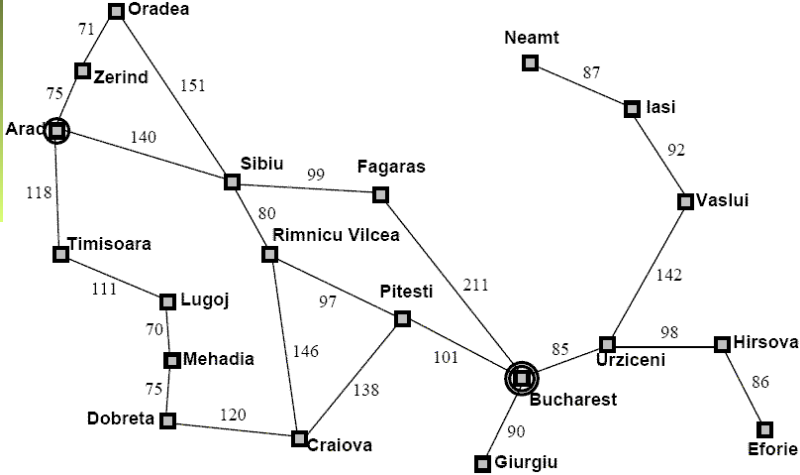
Busca com detecção de estados repetidos é ótima se $h(n)$ for consistente.

Demonstração:

3. Dos dois anteriores, segue que a sequência de nós expandidos por A^ está em ordem crescente de $f(n)$. O primeiro nó objetivo selecionado para expansão deve ser a solução ótima.*

Exemplo

$$f(n) = g(n) + h(n)$$



a sequência de nós expandidos por A* está em ordem crescente de $f(n)$.

Otimidade de A^*

Busca com detecção de estados repetidos é ótima se $h(n)$ for consistente.

Demonstração:

3. Dos dois anteriores, segue que a sequência de nós expandidos por A^ está em ordem crescente de $f(n)$. O primeiro nó objetivo selecionado para expansão deve ser a solução ótima.*

Porque f é o custo real para os nós objetivos

Os outros nós objetivo visitados depois terão custo \geq

Busca heurística limitada pela memória

- Apesar da existência de algoritmos de busca informada, alguns problemas são realmente difíceis por natureza
- Algoritmos utilizados para economizar memória.
 - **IDA*** extensão da busca de aprofundamento iterativo para o A^* . Em que o corte usado é f-cost em vez da profundidade.
 - **RBFS** imita a operação da busca de melhor escolha, usando espaço linear de memória.
 - **SMA***, é similar ao A^* , mas restringe o tamanho da lista de nós para caber na memória disponível.

SMA*

- para gerar um sucessor, se não existir espaço na memória, SMA* joga fora nós com alto *f-cost* (*nós esquecidos*)
- guarda nos nós ancestrais informação sobre a qualidade do melhor caminho da sub-árvore esquecida
- somente regenera a sub-árvore esquecida quando todos os outros caminhos se mostraram piores que o caminho esquecido

Definindo heurísticas

- Cada problema **exige** uma função heurística diferente.
- Não se deve superestimar o custo real da solução.
- Como escolher uma boa função heurística para o jogo 8-Puzzle?

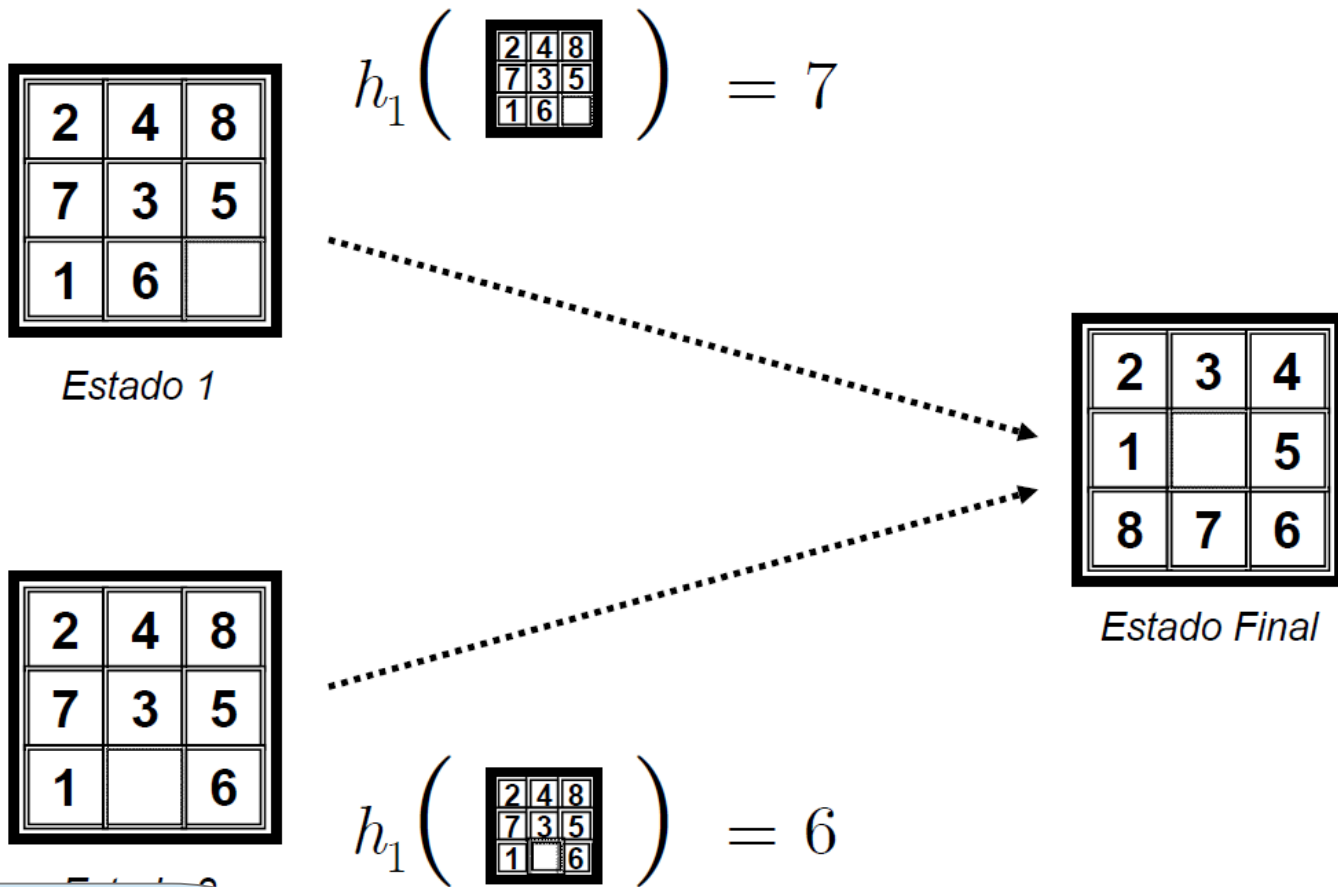
7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

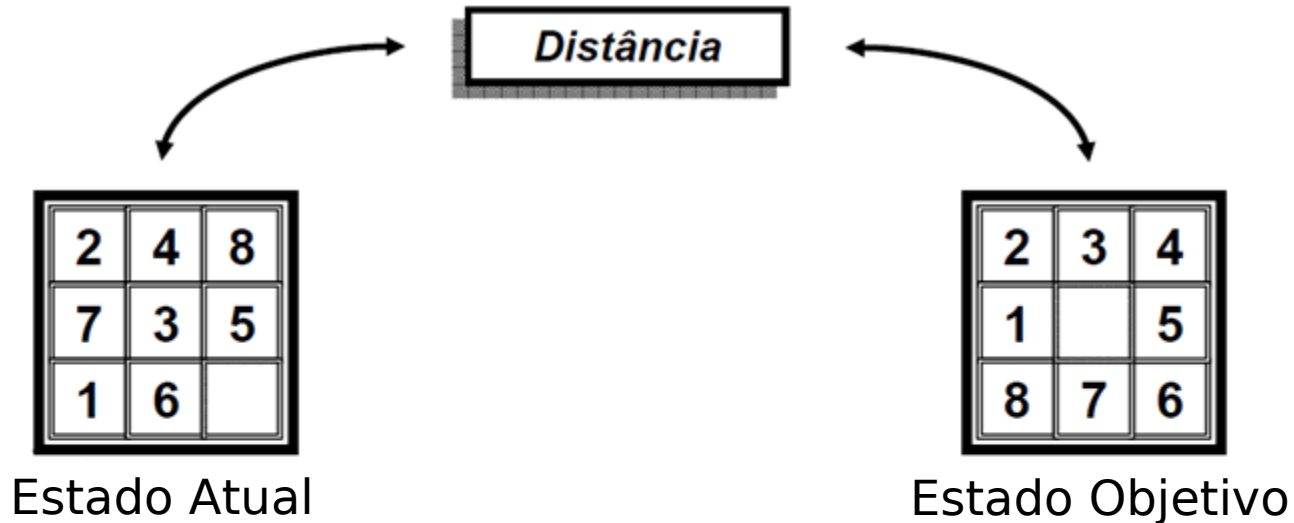
Goal State

Definindo heurísticas



É admissível?

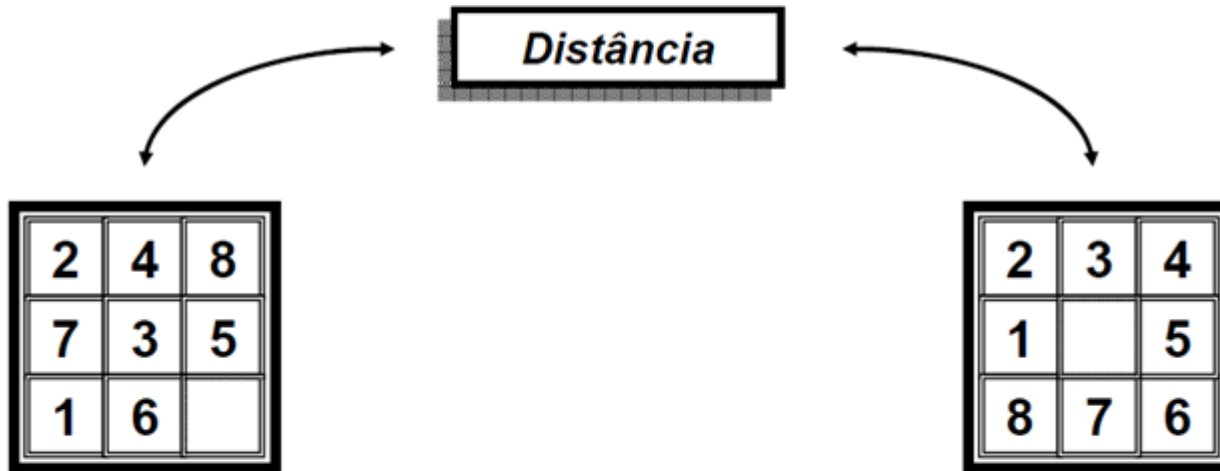
Definindo heurísticas



É admissível porque qualquer peça que esteja fora de lugar deverá ser movida pelo menos uma vez.

$$h_1 \left(\begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & 6 & \\ \hline \end{array} \right) = \begin{array}{l} \text{A quantidade de} \\ \text{peças fora do} \\ \text{lugar} \\ = 7 \end{array}$$

Definindo heurísticas



$$h_2\left(\begin{array}{|c|c|c|}\hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & 6 & \\ \hline\end{array}\right) = ?$$

Outra Heurística?

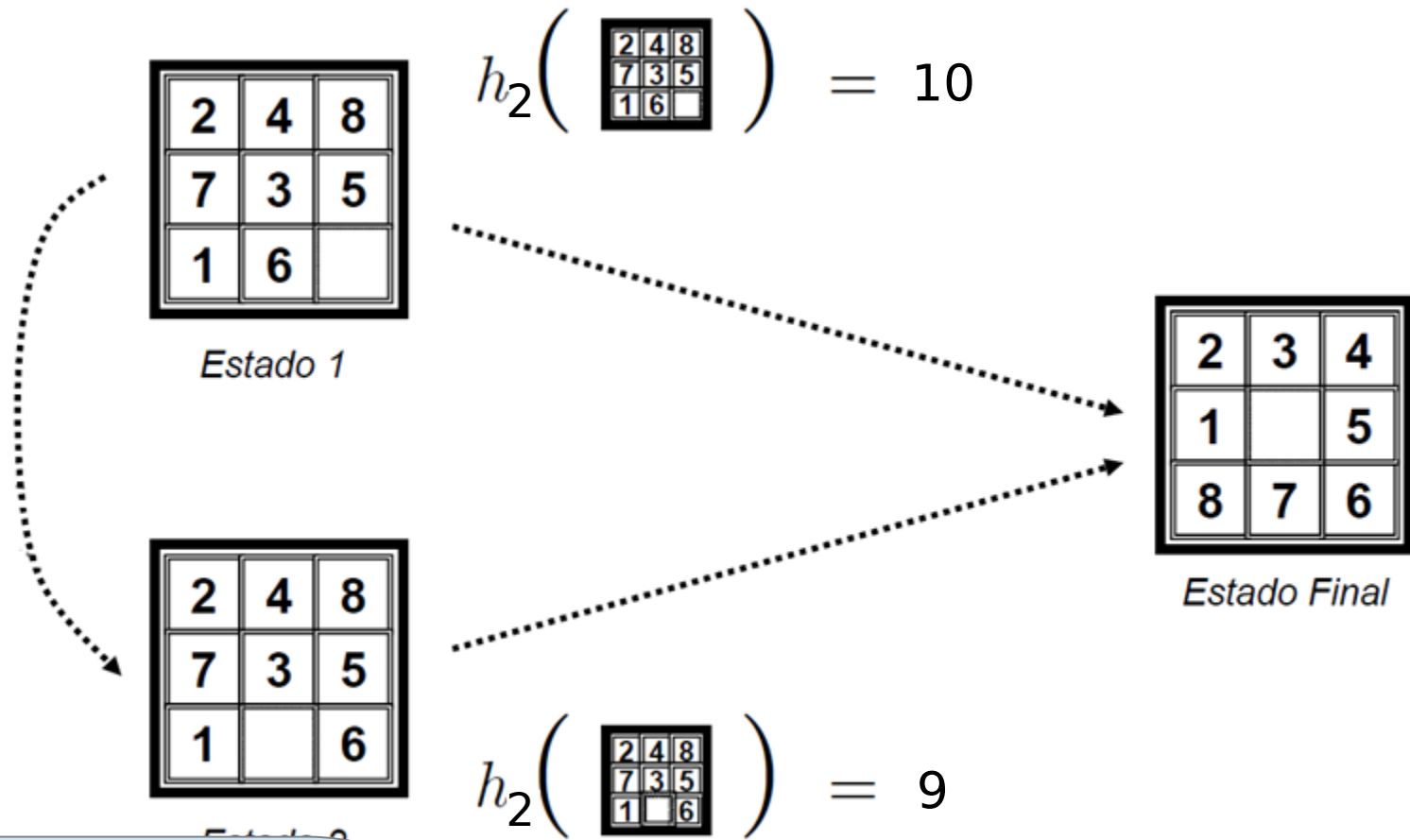
Definindo heurísticas



$$h_2 \left(\begin{array}{|c|c|c|} \hline 2 & 4 & 8 \\ \hline 7 & 3 & 5 \\ \hline 1 & 6 & \\ \hline \end{array} \right) =$$
$$= 10$$

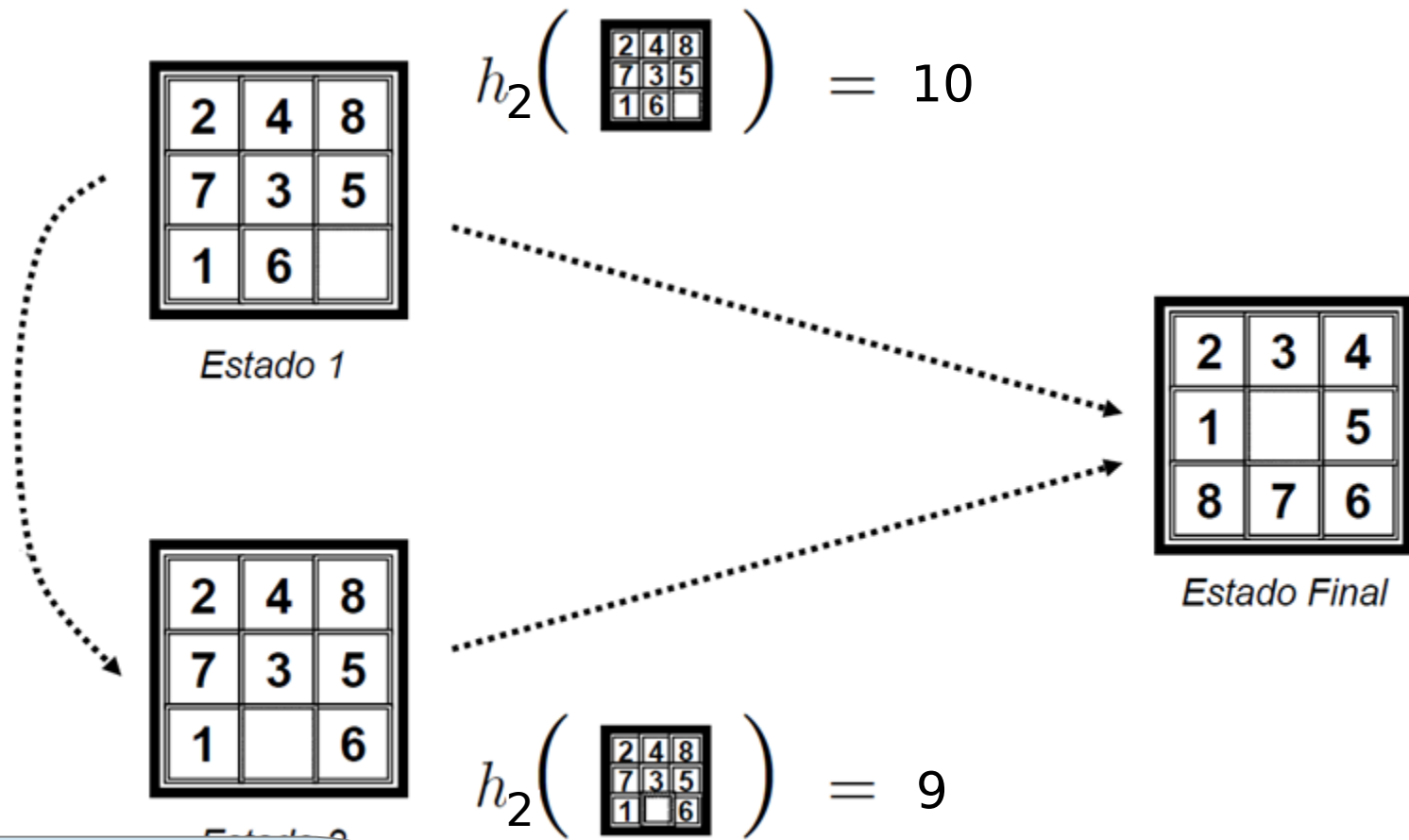
Soma das distâncias (horizontal e vertical) das peças de suas posições objetivo. Conhecida como distância de Manhattan.

Definindo heurísticas



É admissível?

Definindo heurísticas



É admissível.

Definindo heurísticas

- Como escolher uma boa função heurística para o jogo 8-Puzzle?
- Qual das duas heurísticas é melhor?

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Definindo heurísticas

Se $h_2(n) \geq h_1(n)$ para todo n então h_2 **domina** h_1 e é melhor para a busca (expande menos nós)

Exemplos :

$d = 14$ $A^*(h_1) = 539$ nós

$A^*(h_2) = 113$ nós

$d = 24$ $A^*(h_1) = 39.135$ nós

$A^*(h_2) = 1.641$ nós

Como “inventar” uma boa heurística?

Note que h_2 e h_1 são versões simplificadas do problema.

Heurísticas admissíveis podem ser derivadas a partir do custo da solução exata de uma versão relaxada do problema. Isto é, a partir um problema com menos restrições sobre as ações.

Como “inventar” uma boa heurística?

Podemos tentar descrever o problema em uma linguagem formal. Ex:

Uma peça pode se mover do quadrado A para B se
A for horizontal ou verticalmente adjacente a B e B estiver vazio.

Como “inventar” uma boa heurística?

Podemos tentar descrever o problema em uma linguagem formal. Ex:

Uma peça pode se mover do quadrado A para B se
A for horizontal ou verticalmente adjacente a B e B estiver vazio.



Podemos gerar 3 problemas relaxados, removendo restrições

Como “inventar” uma boa heurística?

Podemos tentar descrever o problema em uma linguagem formal. Ex:

Uma peça pode se mover do quadrado A para B se

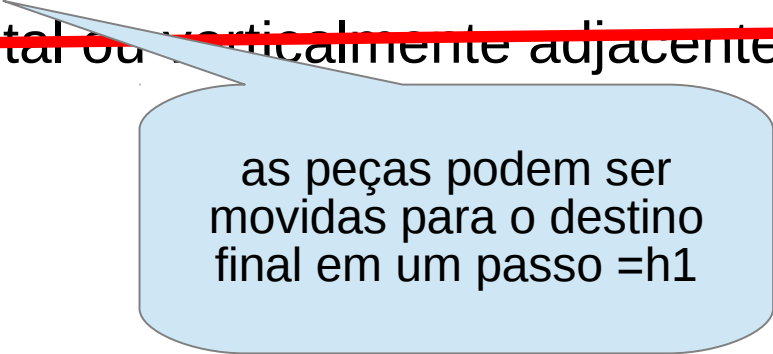
~~A for horizontal ou verticalmente adjacente a B e B estiver vazio.~~

Como “inventar” uma boa heurística?

Podemos tentar descrever o problema em uma linguagem formal. Ex:

Uma peça pode se mover do quadrado A para B se

~~A for horizontal ou verticalmente adjacente a B e B estiver vazio.~~



as peças podem ser movidas para o destino final em um passo =h1

Como “inventar” uma boa heurística?

Podemos tentar descrever o problema em uma linguagem formal. Ex:

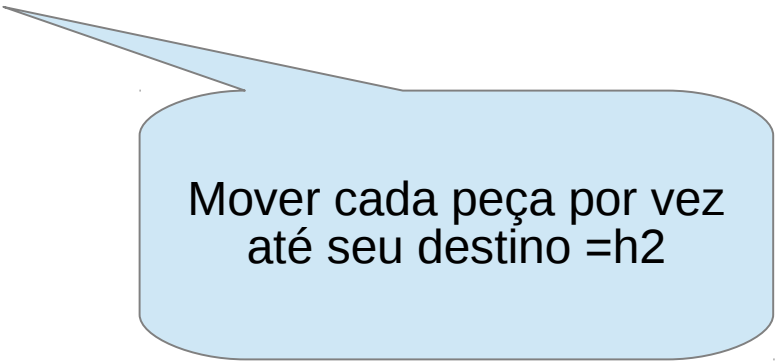
Uma peça pode se mover do quadrado A para B se

A for horizontal ou verticalmente adjacente a B e ~~B estiver vazio.~~

Como “inventar” uma boa heurística?

Podemos tentar descrever o problema em uma linguagem formal. Ex:

Uma peça pode se mover do quadrado A para B se
A for horizontal ou verticalmente adjacente a B e ~~B estiver vazio.~~



Mover cada peça por vez
até seu destino =h2

Como “inventar” uma boa heurística?

Podemos tentar descrever o problema em uma linguagem formal. Ex:

Uma peça pode se mover do quadrado A para B se

~~A for horizontal ou verticalmente adjacente a B~~ e B estiver vazio.

Como “inventar” uma boa heurística?

- É crucial que os problemas relaxados gerados possam ser resolvidos sem busca.
- custo de se calcular $h(n)$ deve ser menor que o custo de se expandir um nó
- *Uma boa heurística deve ser o mais precisa possível e deve ser calculada de maneira eficiente!*

Definindo heurísticas

Se uma coleção de heurísticas admissíveis estiver disponível:

$$h_1(n), \dots, h_m(n)$$

E nenhuma domina claramente as outras, podemos usar uma heurística composta

$$h(n) = \text{max}(h_1(n), \dots, h_m(n))$$

h também é admissível.