# Busca com informação

- Busca em espaço de estados de problemas usando heurísticas (conhecimento específico sobre o domínio do problema)
- Russel & Norvig: capítulo 4

# Última aula

- Busca sem informação: percorrer sistematicamente o espaço de estados, sem conhecimento específico sobre o problema
- Estratégias de busca sem informação:

- Busca em extensão
- Busca em profundidade
- Busca em profundidade limitada
- Busca com custo uniforme
- Busca de aprofundamento iterativo

# Heurística

- Função que descreve, de modo imperfeito, quão bom é um estado (ou "quão perto do objetivo" ele está)
- $h(n)$: estimativa de custo do melhor caminho de $n$ até o objetivo
- Uma heurística pode acelerar algoritmos de busca, priorizando nós que parecem ser melhores
- Heurística deve ser escolhida para cada problema

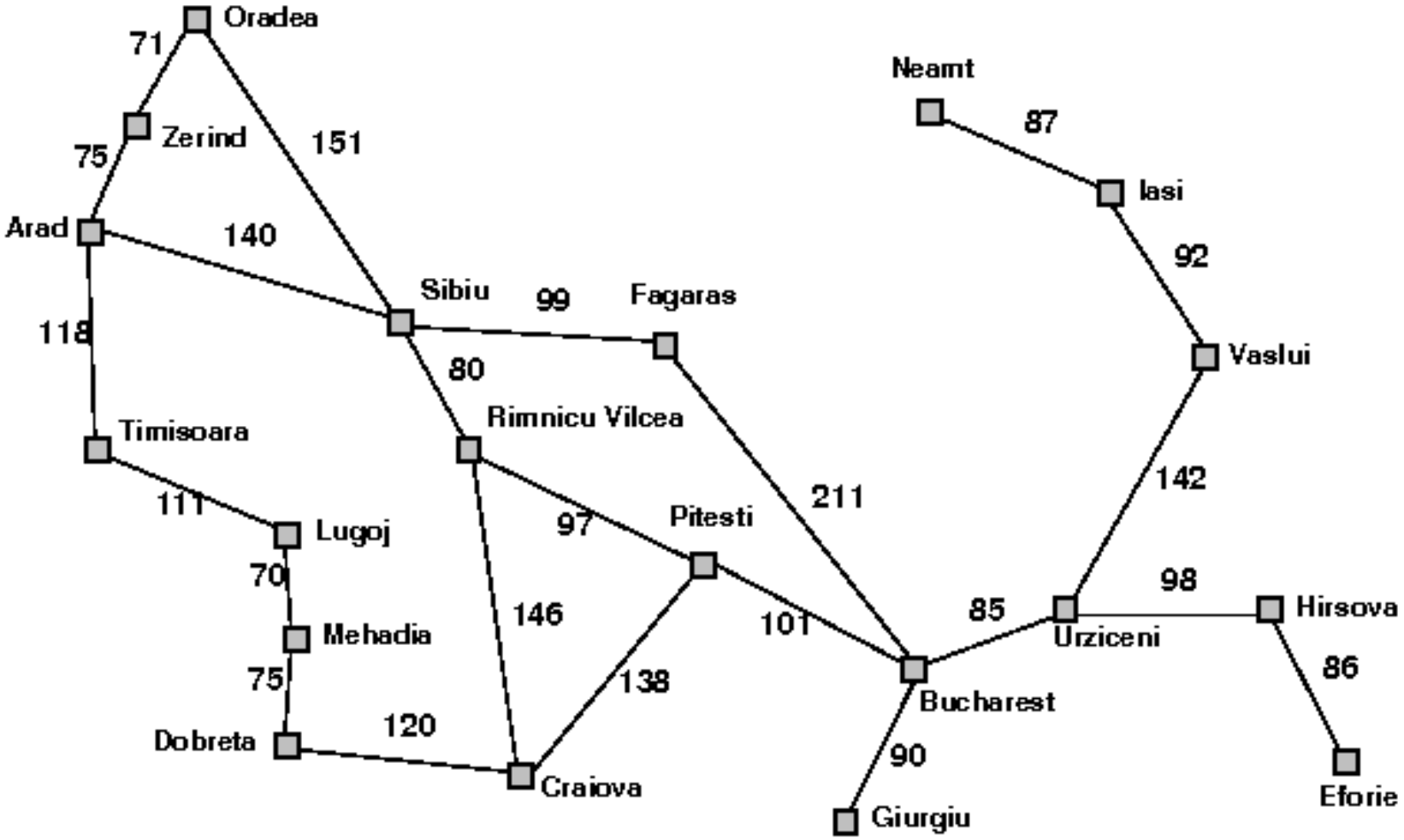
# Estratégias de busca com informação

- Usam uma função de avaliação $f$ para os nós
- Busca gulosa pela melhor escolha
- A$^*$

# Busca gulosa

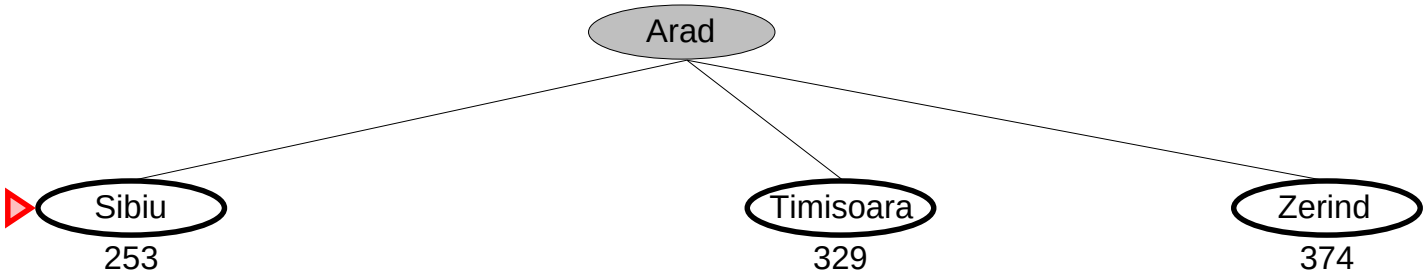
- Usa apenas a heurística $h$
- Seleciona o nó que parece ser imediatamente melhor:
- Dentre os nós não expandidos, expanda o nó $n$ com melhor $h(n)$

# Lembrando: turista na Romênia

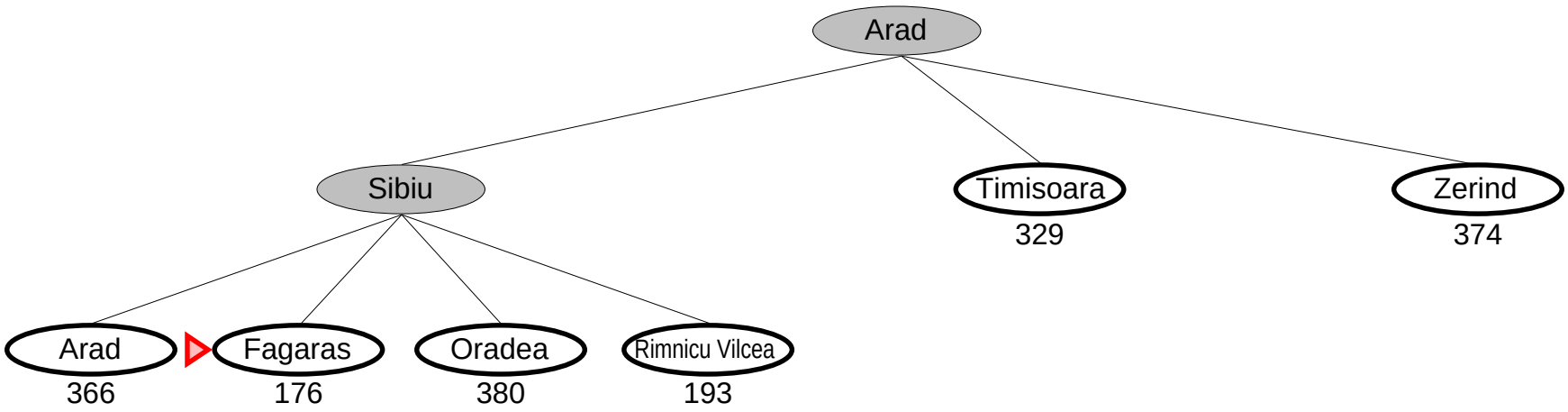


# Busca gulosa: exemplo

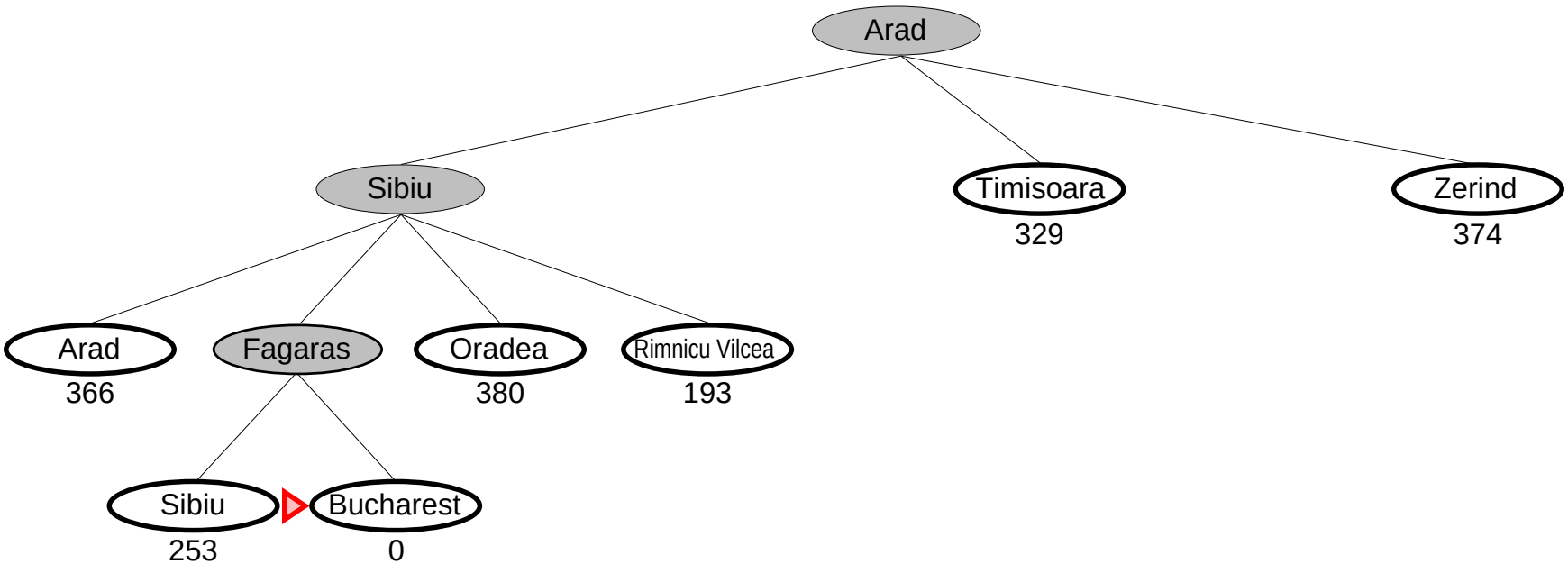
# Busca gulosa: exemplo



# Busca gulosa: exemplo

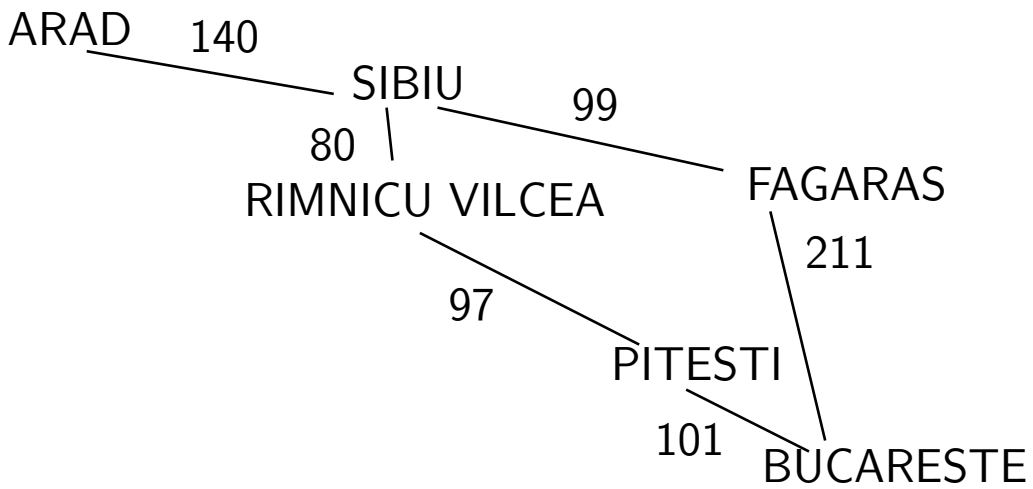


# Busca gulosa: exemplo



# Busca gulosa: a solução não é ótima

- Solução encontrada pelo algoritmo:
  Arad $\rightarrow$Sibiu $\rightarrow$Fagaras $\rightarrow$Bucareste (450 Km)
- Melhor:
  Arad $\rightarrow$Sibiu $\rightarrow$Rimnicu Vilcea $\rightarrow$Pitesti $\rightarrow$Bucareste (418 Km)



# Busca gulosa: análise

- Ótima: NÃO
- Completa: NÃO (pode ficar presa em ciclos)
- Tempo: $O(b^m)$, mas uma boa heurística pode melhorar muito
- Espaço: $O(b^m)$, todos os nós na memória
- Lembrando:
  - $b$: fator de ramificação *(branching)*
  - $m$: profundidade máxima da árvore

# A*

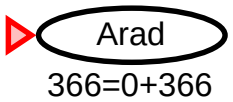
- Busca gulosa ignora caminho já percorrido
- A*: usa função $f(n) = g(n) + h(n)$
- $g(n)$ = custo até o nó $n$
- $h(n)$ = custo estimado até o objetivo (mesmo $h$ da busca gulosa)
- A* escolhe sempre o nó com menor $f(n)$ ainda não expandido

# A*: heurística admissível

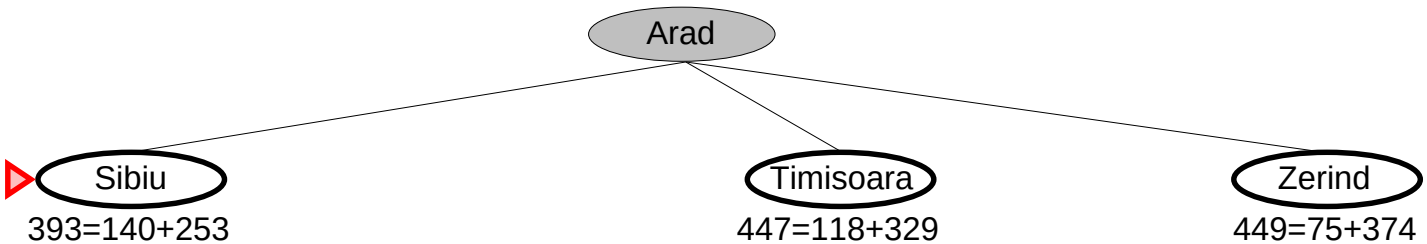
- O A* depende de uma heurística admissível.

- *h* é admissível se *nunca superestima* o custo real do caminho ao objetivo
- Exemplo: distância em linha reta (é o mínimo que precisamos andar de um ponto a outro)

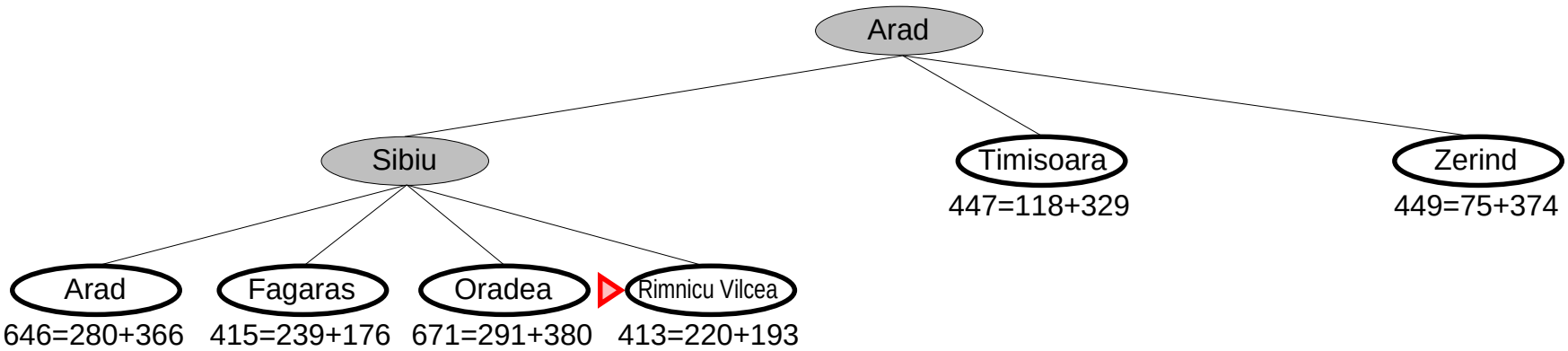
# A*: exemplo



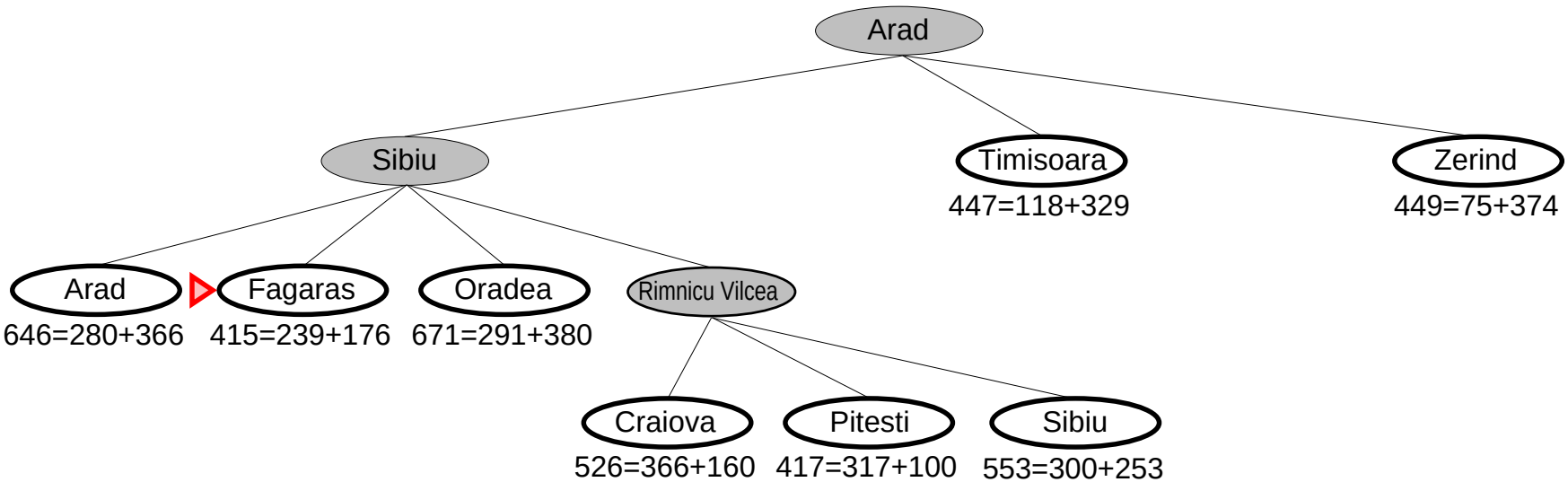
# A*: exemplo


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

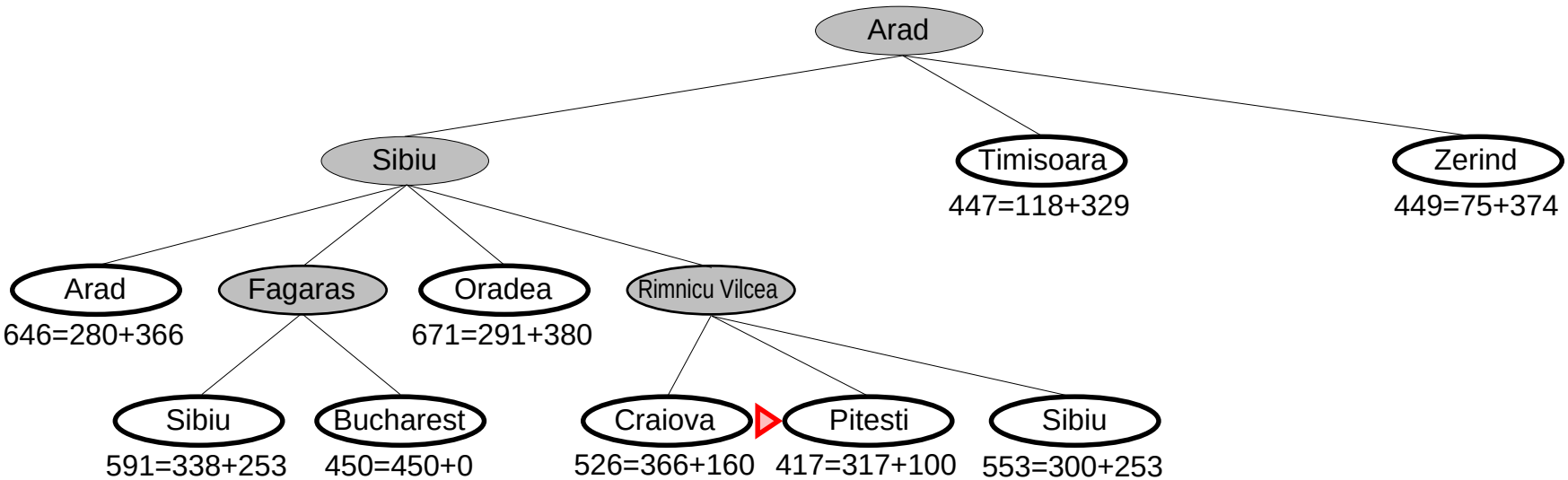
# A*: exemplo


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

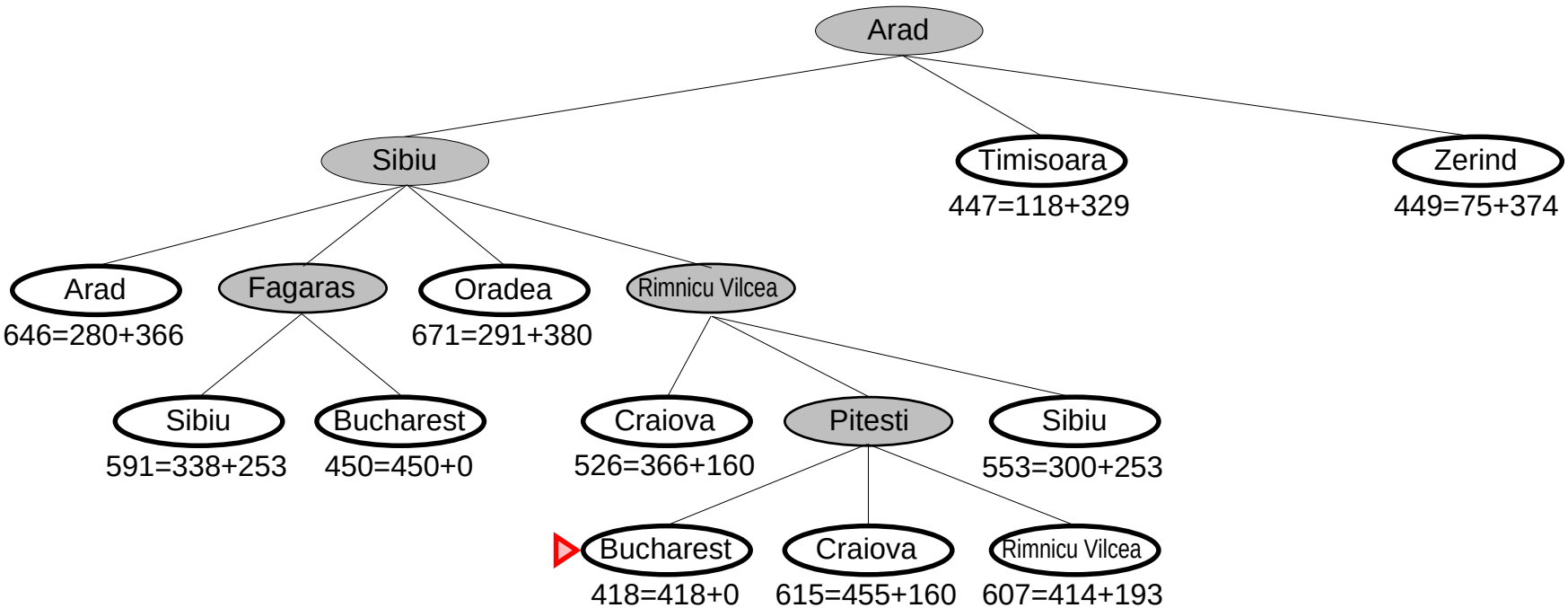
# A*: exemplo


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

# A*: exemplo


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

# A*: exemplo


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
Sibiu
553=300+253
Bucharest
418=418+0
Craiova
615=455+160
Rimnicu Vilcea
607=414+193

# A*: otimalidade

- Suponha que há um $G_2$ sub-ótimo na borda
- $n$ não foi expandido, mas está no caminho de um objetivo ótimo $G$

$$\begin{aligned} f(G_2) &= g(G_2) && \text{porque } h(G_2) = 0 \\ &> g(G) && \text{porque } G_2 \text{ é sub-ótimo} \\ &\geq f(n) && \text{porque } h \text{ é admissível} \end{aligned}$$

- Como $f(G_2) > f(n)$, A* nunca expandirá $G_2$

# $A^*$: análise

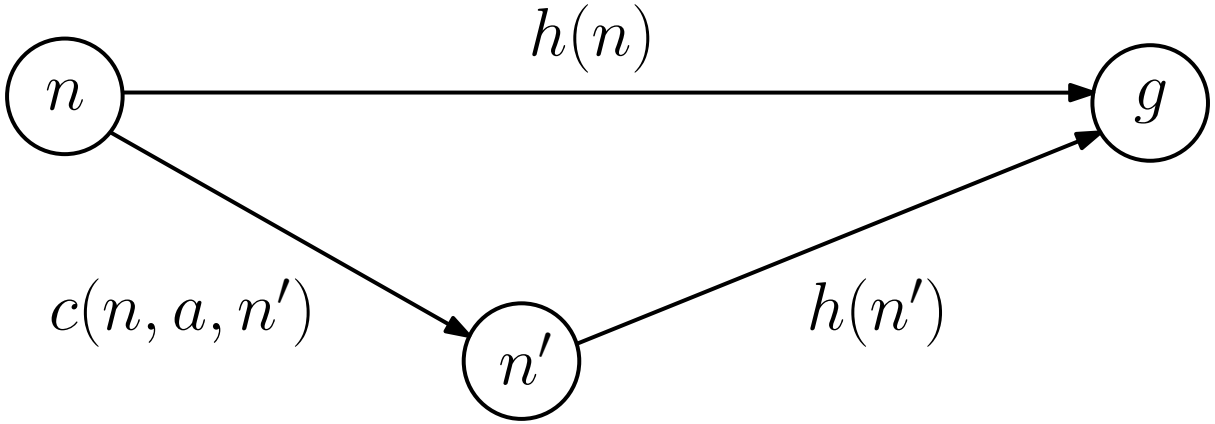
- Ótima: SIM (em árvores)
- Completa: SIM
- Tempo: exponencial
- Espaço: todos os nós na memória
  - Esgota a memória rapidamente (antes de o tempo de execução se tornar um problema)

# $A^*$ em grafos

- Heurística admissível não é suficiente:
- Caminho ótimo para um estado pode ser descartado se não for o primeiro caminho pelo qual ele é visitado
- Otimalidade depende de heurística consistente (garante que o primeiro caminho chegando a cada nó será o caminho ótimo)

# Consistência

- Para um nó $n$ e seu sucessor $n'$:
- O custo de $n'$ não pode ser maior que o custo de $n$




- Toda heurística consistente é admissível
- Freqüentemente heurísticas admissíveis são consistentes

# AIA* (IDA*)

- Aprofundamento Iterativo A* *(Iterative Deepening A*)*
- Semelhante ao aprofundamento iterativo
- Usa $f$ para corte ao invés de profundidade
- Ótimo se $h$ é admissível

# BRPM (RBFS)

- Busca Recursiva Pelo Melhor *(Recursive Best First Search)*
- Lembra o melhor $f$ dos ancestrais
- Se este $f$ é excedido, volta ao ponto onde era ótimo e um caminho alternativo é escolhido
- Ótimo se $h$ é admissível

# Funções heurísticas

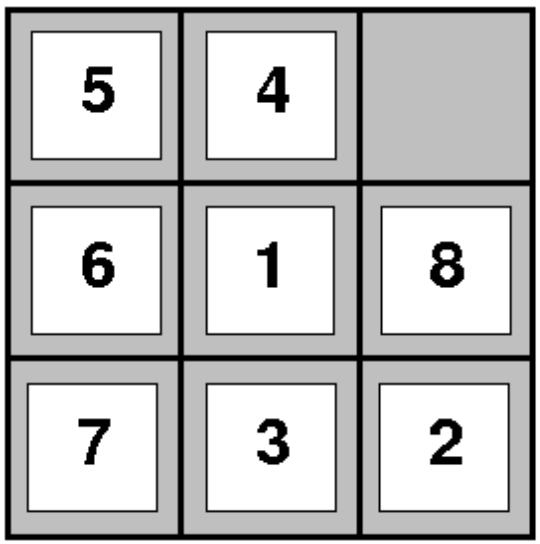
- A escolha de uma boa heurística é extremamente importante
- Fator de ramificação efetiva ($b^*$):
- $N$ nós gerados
- Profundidade $d$
- $b* =$ fator de ramificação da árvore uniforme de profundidade $d$ com $N + 1$ nós

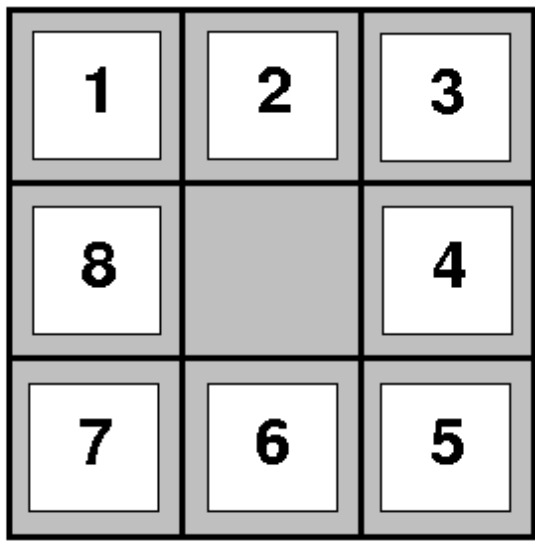
$$N + 1 = 1 + b^* + (b^*)^2 + \cdots + (b^*)^d$$

- Por exemplo, com $d = 5$ e $N = 25$, $b^* = 1,92$

# Funções heurísticas: exemplo

- Exemplo: quebra-cabeça de oito peças



# Funções heurísticas: exemplo

- $h_1(n)$ = número de blocos fora do lugar
- $h_2(n)$ = soma da distância de Manhattan de cada bloco até seu lugar correto
- Ambas são admissíveis
- Se quisermos usar o A$^*$, qual é melhor, $h_1$ ou $h_2$?

# Funções heurísticas: exemplo

- Comparando aprofundamento iterativo e A* com as duas heurísticas:
  - 1200 problemas gerados aleatoriamente (Russel & Norvig, p. 106)
  - Tamanho de solução variando entre 2 e 24

| | Custo da busca | | | Fator de ramificação | | |
|---|---|---|---|---|---|---|
| d | BAI | A*($h_1$) | A*($h_2$) | BAI | A*($h_1$) | A*($h_2$) |
| 2 | 10 | 6 | 6 | 2,45 | 1,79 | 1,79 |
| 16 | 3644035 | 227 | 73 | 2,78 | 1,42 | 1,24 |
| 24 | – | 39135 | 1641 | – | 1,48 | 1,26 |

# LMSA* (SMBA*)

- A* limitado pela memória *(Simplified Memory Bound A*)*
- Similar ao A*: expande nós folha repetidamente
- Quando a memória acaba, esquece um nó para poder expandir outro
- Nó esquecido é o pior
- $f$ do nó esquecido é copiado para seu pai
- A árvore do nó esquecido só será re-gerada se todos os outros nós forem piores que ele

# LMSA$^*$: análise

- Completo se a solução é alcançável ($d$ menor que a memória disponível)
- Se a solução ótima é alcançável, o algoritmo é ótimo
- Senão, retorna a melhor solução alcançável

- Pode acabar alternando entre caminhos diferentes, de forma similar ao *thrashing* em sistemas operacionais
  - e neste caso o A$^*$ pode resolver um problema que o LMSA$^*$ não resolve!

# Busca local

- Caminhos podem não ser relevantes (o que importa é o estado final)
- Busca local:

- Subida de encosta
- Têmpera simulada
- Feixe local
- Algoritmos evolutivos
  - Generalização de algoritmos genéticos, estratégias de evolução e programação genética
- Outros temas em Computação Natural
  - *Particle swarm*
  - Colônia de formigas