Inteligência Artificial
Tópico 03 - Parte 03
Resolução de Problemas por Buscas
Busca com Informação

Profa. Dra. Priscila Tiemi Maeda Saito
priscilasaito@ufscar.br

Roteiro

1. Resolução de Problemas Buscas

2. Estratégias de Busca
   - Busca sem Informação
   - Busca com Informação

Estratégias de Busca

- **Buscas sem Informação, Não Informada, Exaustiva ou Cega**
  - ineficientes na maioria dos casos
  - encontram soluções gerando sistematicamente novos estados e
  - comparando-os com o objetivo

Estratégias de Busca

- **Buscas sem Informação, Não Informada, Exaustiva ou Cega**
  - Busca em Largura
  - Busca de Custo Uniforme
  - Busca em Profundidade
  - Busca em Profundidade Limitada
  - Busca em Profundidade com Aprofundamento Iterativo

Estratégias de Busca

- **Buscas com Informação** ou **Heurística**
  - Utilizam conhecimento específico sobre o problema
  - Podem encontrar soluções de forma mais eficiente do que a busca cega
    - ★ conhecimento específico além da definição do problema
  - Heurística para encontrar os caminhos mais promissores primeiro
    - ★ estima distância ao objetivo

Buscas com Informação

- **Busca Pela Melhor Escolha** ou **Best-First**
  - Abordagem geral de busca com informação
  - Usa uma função de avaliação $f(n)$ para selecionar o nó a ser expandido
    - ⋆ função de custo, cujo nó que apresentar menor $f(n)$ é expandido primeiro
- Implementação é idêntica ao da busca de custo uniforme
  - substituindo-se g(n) por $f(n)$ - busca em largura
  - introduz na fila de nós a serem expandidos de acordo com $f(n)$ - fila de prioridades

Busca Best-First

- Ideia: usar uma função de avaliação f(n) para cada nó
  - estimar o grau em que um nó é "desejável" como caminho
  - expandir os nós mais desejáveis

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

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

$h(n)$ = Custo estimado de n ao estado objetivo pelo caminho mais barato

Busca Gulosa Pela Melhor Escolha

- Greedy Best-First Search
- A cada passo tenta chegar mais perto do estado objetivo sem se preocupar com os passos futuros
- Supondo que provavelmente levará a uma solução rápida
- Utiliza somente a componente heurística da função f(n)

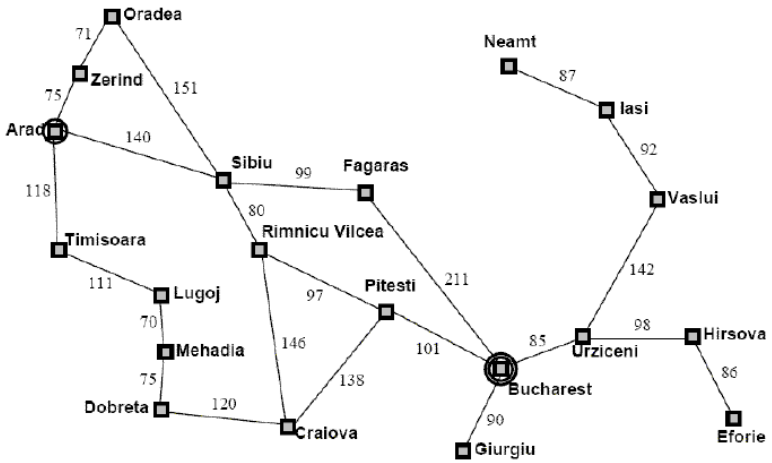
$$f(n) = h(n)$$

Busca Gulosa Pela Melhor Escolha

- Exemplo: encontrar a melhor rota (rota mais curta) de uma cidade a outra em um mapa
  - h(n) = distância em linha reta entre as cidades e a cidade meta

Busca Gulosa Pela Melhor Escolha

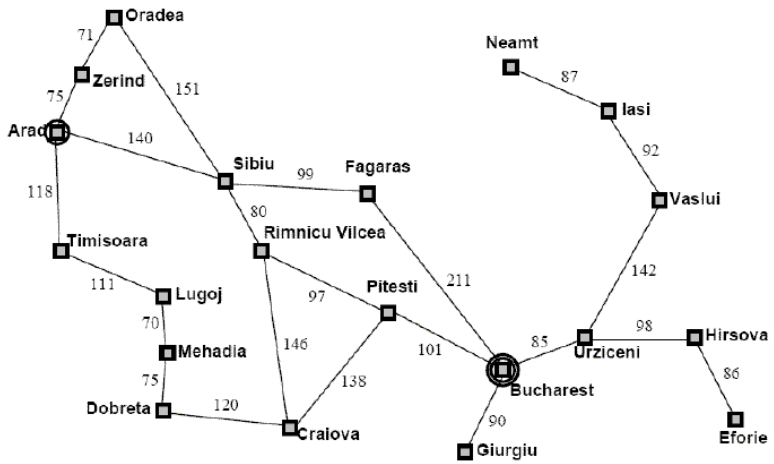
- Exemplo: ir de Arad a Bucareste
- Heurística: distância em linha reta - h_{DLR}



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

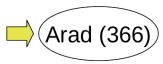
Busca Gulosa Pela Melhor Escolha

- Exemplo: ir de Arad a Bucareste
- Heurística: distância em linha reta - h_{DLR}



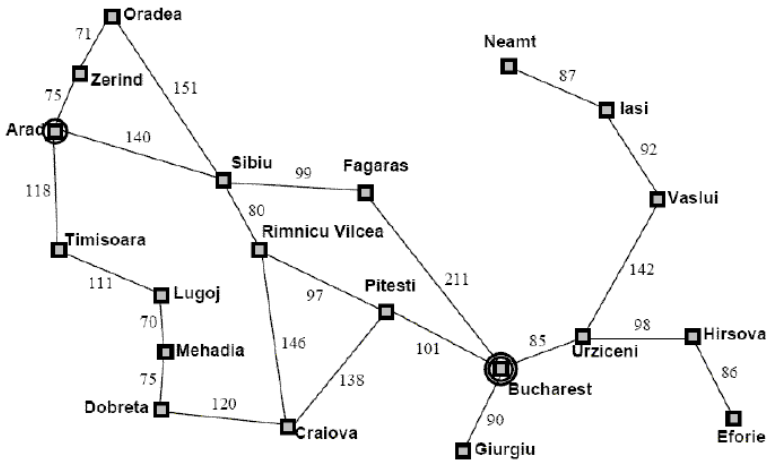
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374
Hirsova	151	Urziceni	80

- Estado Inicial



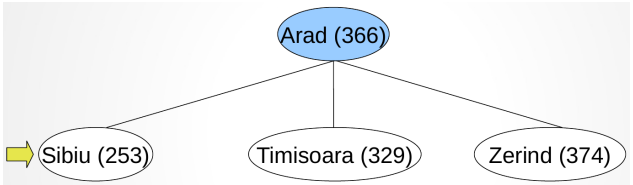
Busca Gulosa Pela Melhor Escolha

- Exemplo: ir de Arad a Bucareste
- Heurística: distância em linha reta - h_{DLR}



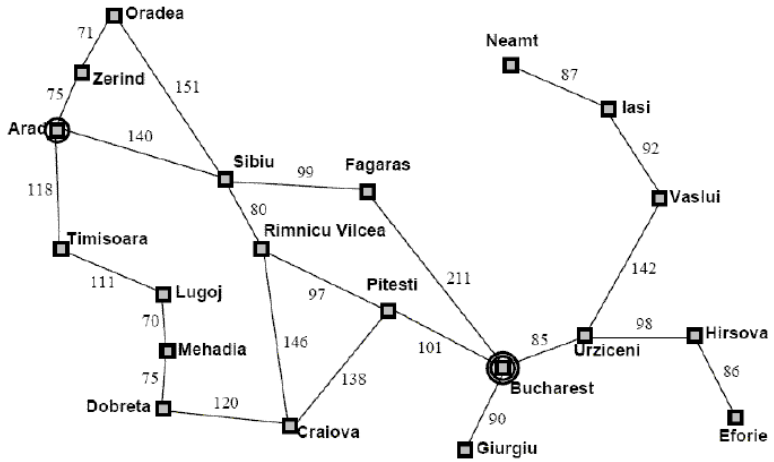
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374
Hirsova	151	Urziceni	80

- Expansão de Arad



Busca Gulosa Pela Melhor Escolha

- Exemplo: ir de Arad a Bucareste
- Heurística: distância em linha reta - h_{DLR}



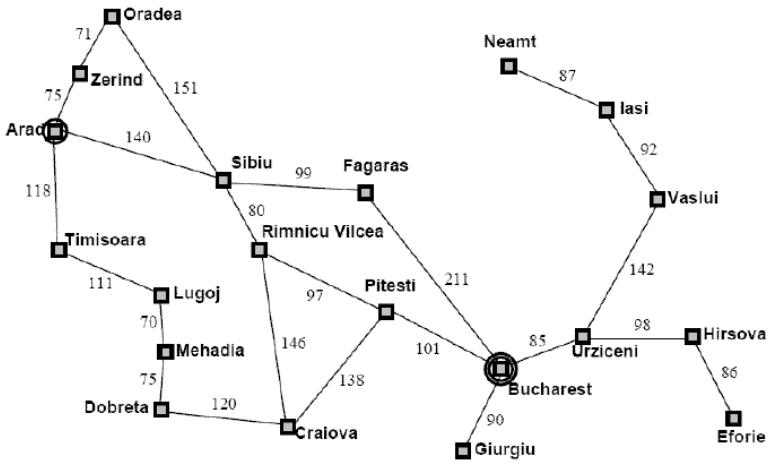
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374
Hirsova	151	Urziceni	80

Exercício

Continuar a aplicar a busca gulosa

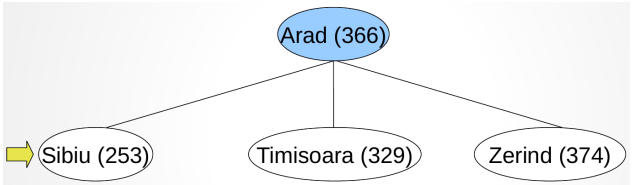
Busca Gulosa Pela Melhor Escolha

- Exemplo: ir de Arad a Bucareste
- Heurística: distância em linha reta - h_{DLR}



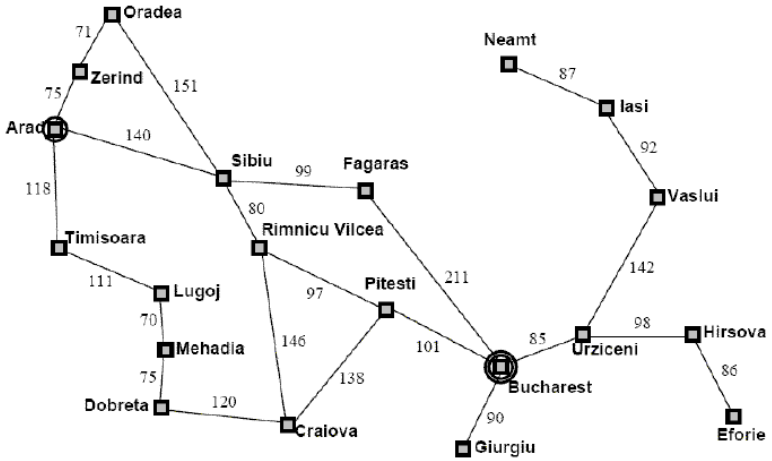
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374
Hirsova	151	Urziceni	80

- Expansão de Arad



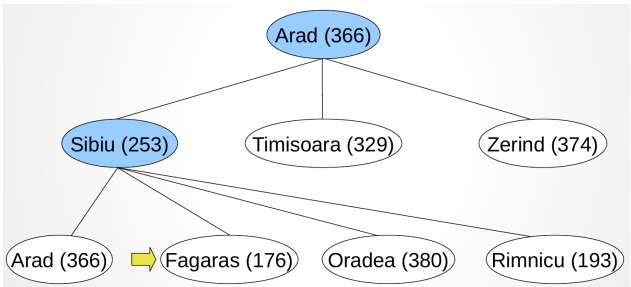
Busca Gulosa Pela Melhor Escolha

- Exemplo: ir de Arad a Bucareste
- Heurística: distância em linha reta - h_{DLR}



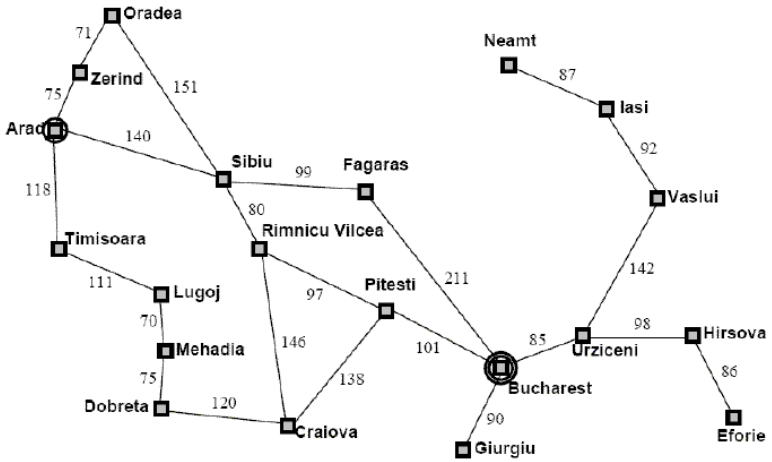
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374
Hirsova	151	Urziceni	80

- Expansão de Sibiu



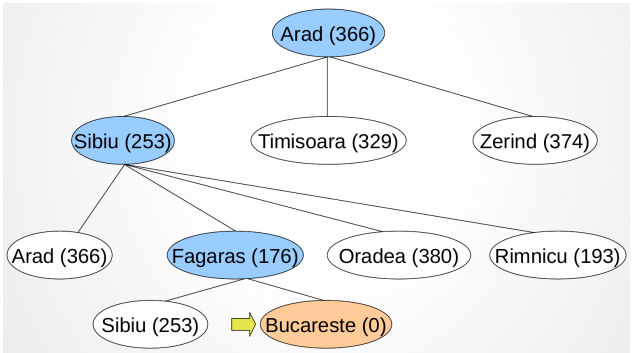
Busca Gulosa Pela Melhor Escolha

- Exemplo: ir de Arad a Bucareste
- Heurística: distância em linha reta - h_{DLR}



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

- Expansão de Fagaras

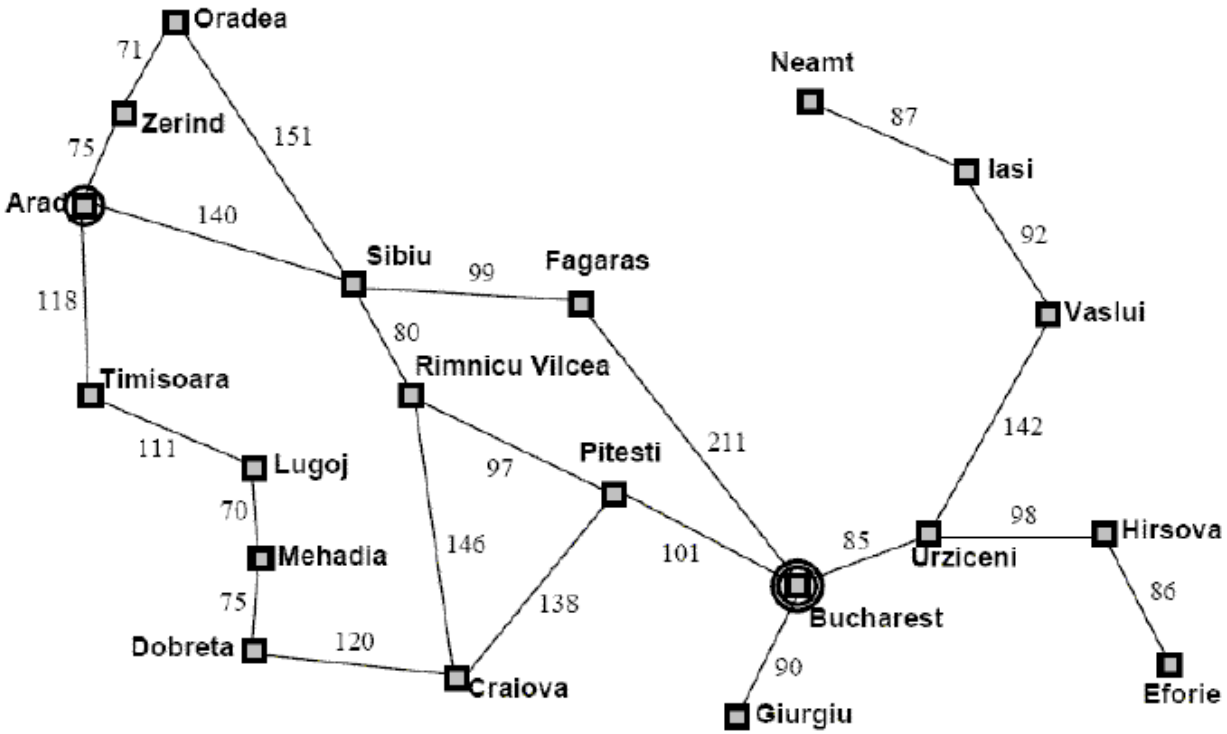


Busca Gulosa Pela Melhor Escolha

- Encontrou a solução sem expandir nenhum nó que não estivesse no caminho da solução
- No entanto, solução não é ótima
  - 32 km mais longo do que por Rimnicu e Pitesti
- Nomenclatura guloso
  - em cada passo, tenta chegar o mais perto possível do objetivo
  - não é ótima, pois segue o melhor passo **considerando somente o momento atual**
  - pode haver um caminho melhor seguindo algumas opções piores em alguns pontos da árvore de busca

Busca Gulosa Pela Melhor Escolha

- Ir de **Iasi** a **Fagaras**?



Busca Gulosa Pela Melhor Escolha

- Minimizar $h(n)$ é suscetível a produzir laços infinitos
  - Ex.: ir de Iasi a Fagaras fica-se preso indefinidamente entre Iasi e Neamt
    - busca gulosa com h_{DLR}: Iasi $\rightarrow$ Neamt $\rightarrow$ Iasi $\rightarrow$ Neamt $\rightarrow$ ...
  - solução: Iasi $\rightarrow$ Vaslui $\rightarrow$ Urziceni $\rightarrow$ Bucareste $\rightarrow$ Fagaras
- Vaslui é mais distante que Neamt do objetivo de acordo com a heurística
- No entanto, é o caminho que liga a Fagaras (por Neamt não tem caminho)

Busca Gulosa Pela Melhor Escolha

- Semelhante à busca em profundidade
  - prefere seguir em um único caminho
- É incompleta
  - pode entrar em caminho infinito
- Não é ótima
- Complexidade tempo no pior caso: $O(b^m)$
  - m é a profundidade máxima do espaço de busca
  - com boa heurística pode ter redução substancial

Busca A^*

- Técnica de busca mais utilizada
- Minimiza custo total estimado da solução
- Avalia nós combinando:
  - g(n): custo real do caminho para alcançar cada nó
    - ★ custo de nó inicial até o nó *n* (valor exato)
  - h(n): custo estimado para ir do nó até o objetivo
    - ★ custo estimado do caminho de n ao objetivo

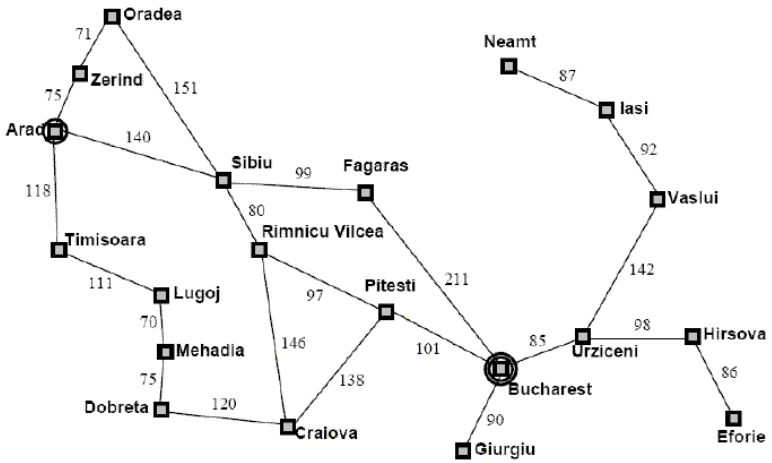
Custo estimado da solução "mais barata" passando por n

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

Ideia: evitar expandir caminhos que já ficaram caros
A^* expande o nó de menor valor de f na fronteira do espaço de estados

Busca A^*

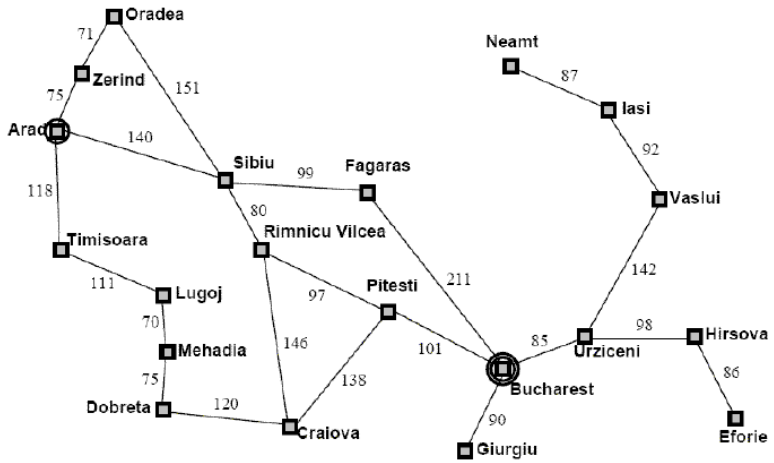
- Exemplo: ir de Arad a Bucareste
- Função de avaliação f(n) = g(n) + h(n)
  - g(n) = custos das estradas (na figura)
  - h(n) = h_{DLR}, distância em linha reta



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

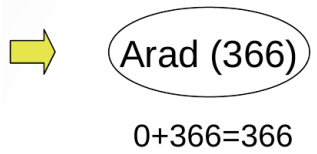
Busca A^*

- Exemplo: ir de Arad a Bucareste



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

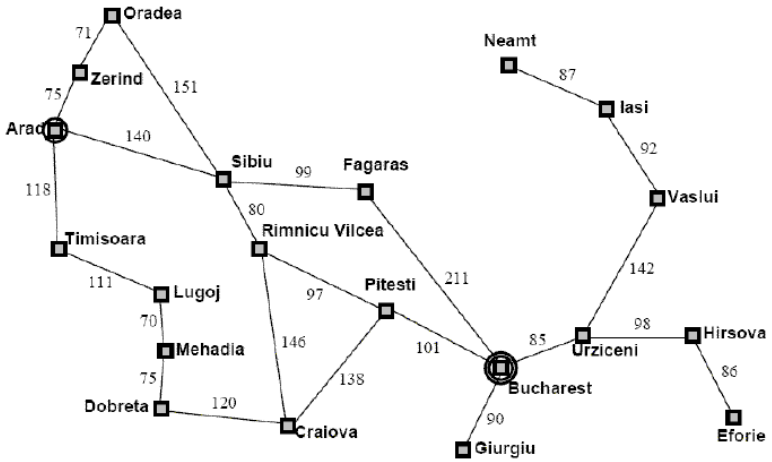
- Estado inicial



- $f(\text{Arad}) = 0 + h(\text{Arad}) = 0 + 366 = 366$

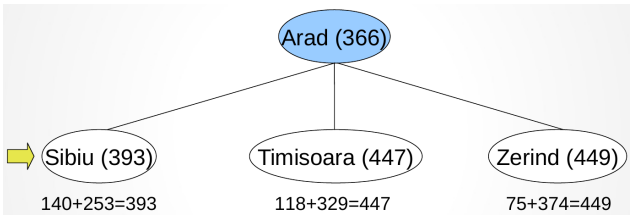
Busca A^*

- Exemplo: ir de Arad a Bucareste



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

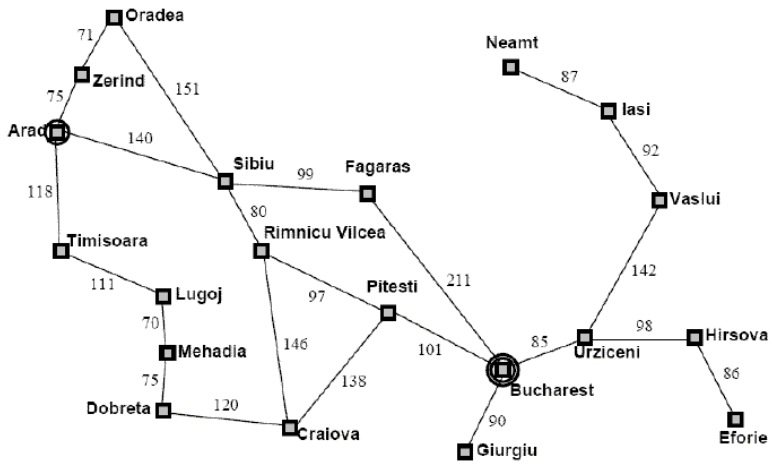
- Expansão de Arad



- f(Sibiu) = c(Arad,Sibiu) + h(Sibiu) = 140 + 253 = 393
- f(Timisoara) = c(Arad,Timisoara) + h(Timisoara) = 118 + 329 = 447
- f(Zerind) = c(Arad,Zerind) + h(Zerind) = 75 + 374 = 449
- Melhor escolha = Sibiu

Busca A^*

- Exemplo: ir de Arad a Bucareste
- Função de avaliação f(n) = g(n) + h(n)
  - g(n) = custos das estradas (na figura)
  - h(n) = h_{DLR}, distância em linha reta



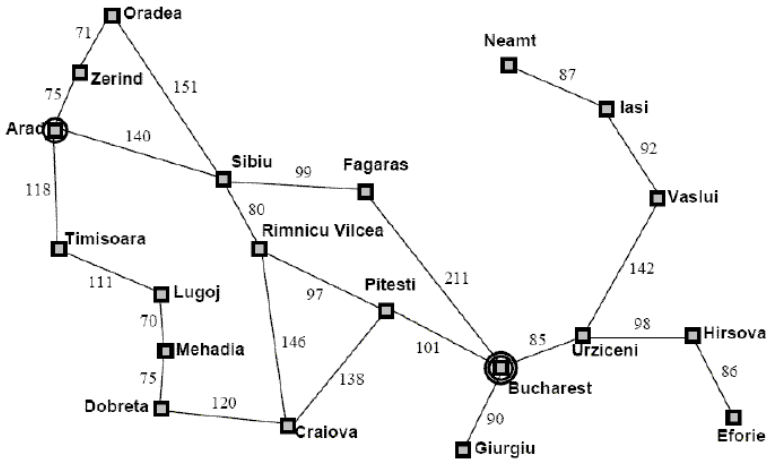
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374
Hirsova	151	Urziceni	80

Exercício

Continuar a aplicar a busca A^*

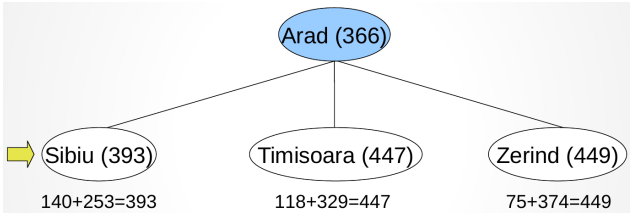
Busca A^*

- Exemplo: ir de Arad a Bucareste



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

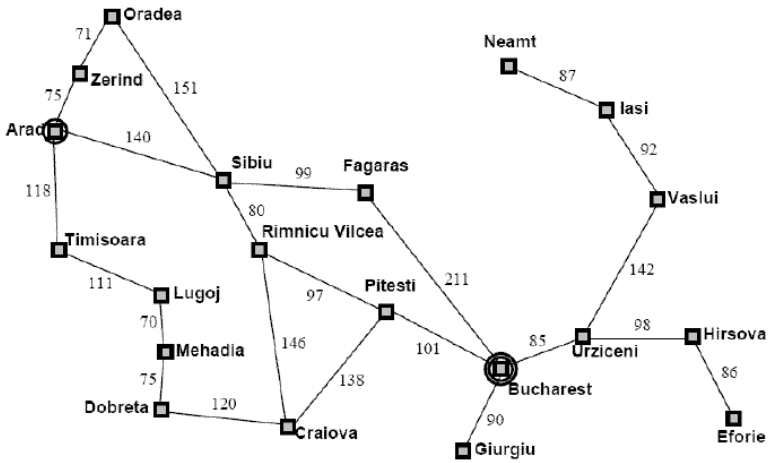
- Expansão de Arad



- f(Sibiu) = c(Arad,Sibiu) + h(Sibiu) = 140 + 253 = 393
- f(Timisoara) = c(Arad,Timisoara) + h(Timisoara) = 118 + 329 = 447
- f(Zerind) = c(Arad,Zerind) + h(Zerind) = 75 + 374 = 449
- Melhor escolha = Sibiu

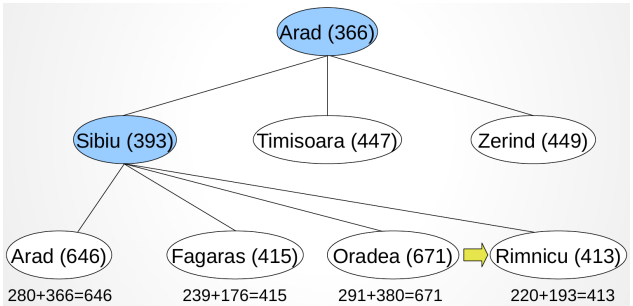
Busca A^*

- Exemplo: ir de Arad a Bucareste



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

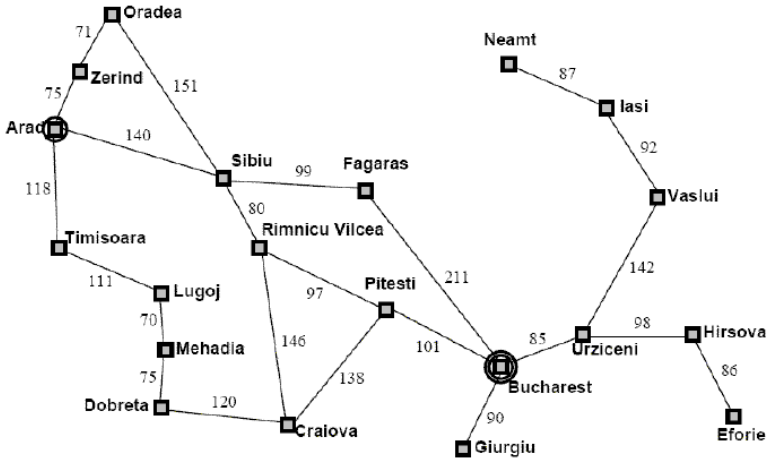
- Expansão de Sibiu



- f(Arad) = g(Arad) + h(Arad) = 280 + 366 = 646
- f(Fagaras) = g(Fagaras) + h(Fagaras) = 239 + 176 = 415
- f(Oradea) = g(Oradea) + h(Oradea) = 291 + 380 = 671
- f(Rimnicu Vilcea) = g(Rimnicu Vilcea) + h(Rimnicu Vilcea) = 220 + 193 = 413
- Melhor escolha = Rimnicu Vilcea

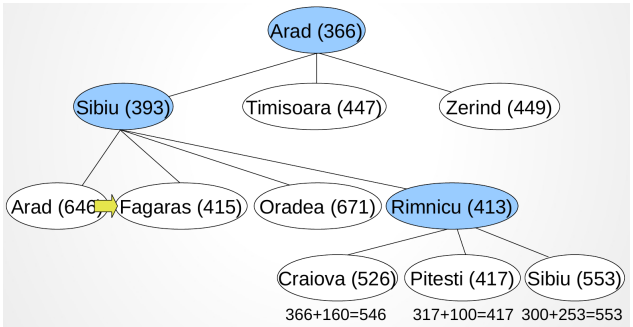
Busca A^*

- Exemplo: ir de Arad a Bucareste



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

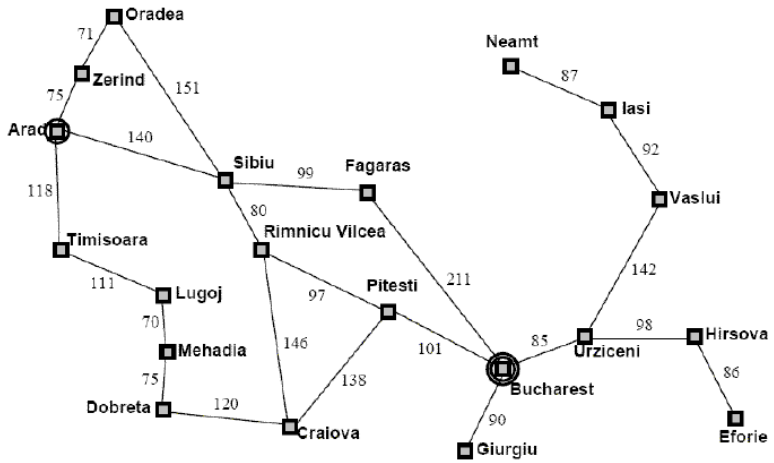
- Expansão de Rimnicu



- f(Craiova) = g(Craiova) + h(Craiova) = 366 + 160 = 526
- f(Pitesti) = g(Pitesti) + h(Pitesti) = 317 + 100 = 417
- f(Sibiu) = g(Sibiu) + h(Sibiu) = 300 + 253 = 553
- Melhor escolha = Fagaras

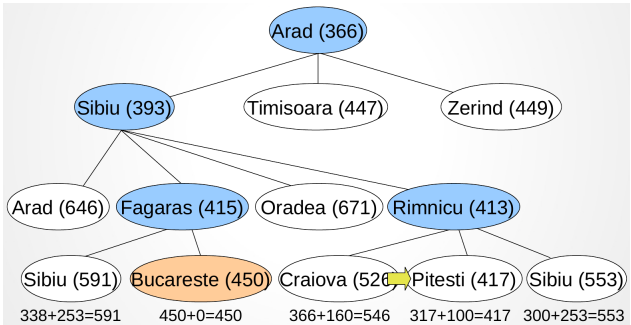
Busca A^*

- Exemplo: ir de Arad a Bucareste



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

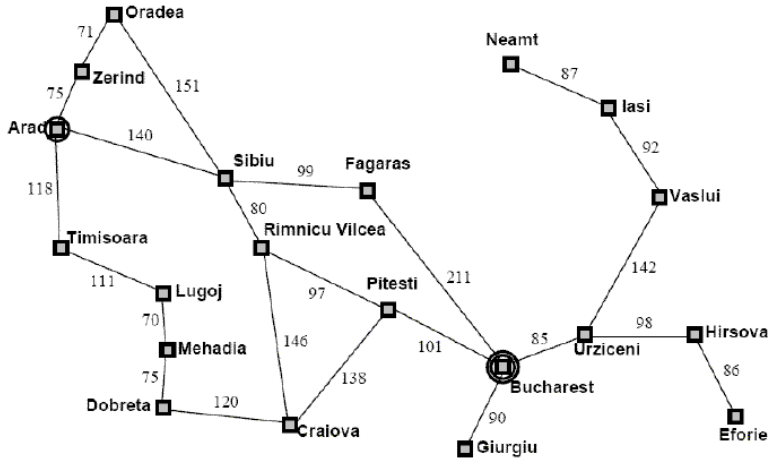
- Expansão de Fagaras



- f(Sibiu) = g(Sibiu) + h(Sibiu) = 338 + 253 = 591
- f(Bucareste) = g(Bucareste) + h(Bucareste) = 450 + 0 = 450
- Melhor escolha = Pitesti

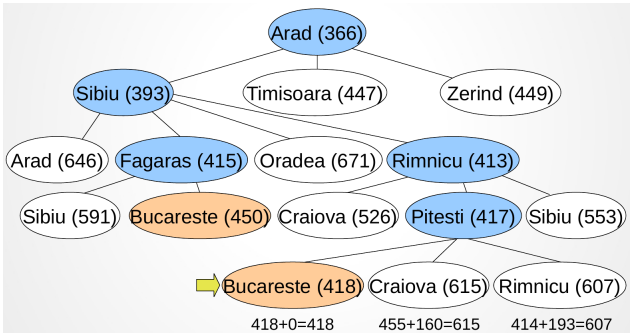
Busca A^*

- Exemplo: ir de Arad a Bucareste



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

- Expansão de Pitesti



- f(Bucareste) = g(Bucareste) + h(Bucareste) = 418 + 0 = 418
- Melhor escolha = Bucareste (Solução Ótima)

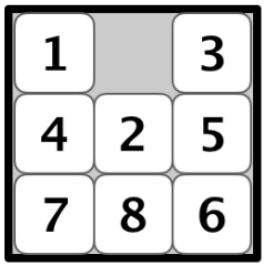
Busca A^*

- A estratégia é **completa** e **ótima**
- **Custo de tempo**
  - $O(b^d) \rightarrow$ exponencial com o comprimento da solução
  - boas funções heurísticas diminuem significativamente esse custo
- **Custo de memória**
  - guarda todos os nós expandidos na memória

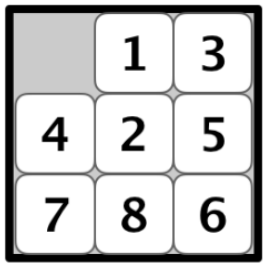
Definindo Heurísticas

Problema - Jogo 8-Puzzle

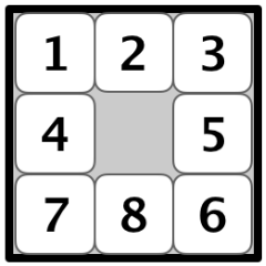
- Exemplo de movimentos possíveis a partir de um estado



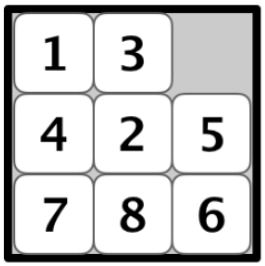
board



neighbor 1



neighbor 2



neighbor 3

- Exemplo de um estado (board) e sua respectiva representação de string (composta por n+1 linhas)
  - primeira linha indica o tamanho do grid
  - n linhas seguintes indicam as peças (zero indica o quadrado vazio)

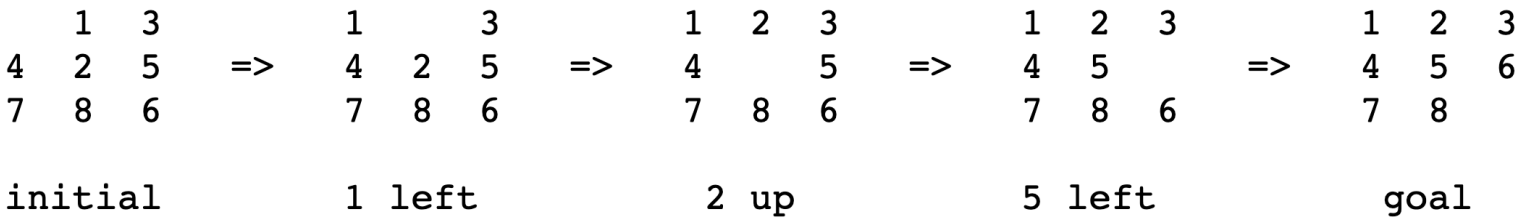
board

```
3
1 0 3
4 2 5
7 8 6
```
string representation

Definindo Heurísticas

Problema - Jogo 8-Puzzle

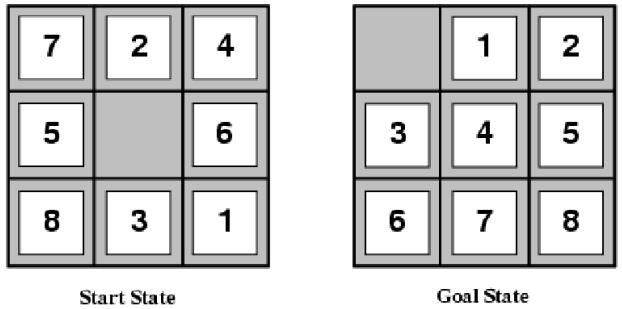
- Exemplo de sequência de movimentos a partir um estado inicial (esquerda) até atingir o estado objetivo (direita)



Definindo Heurísticas

Problema - Jogo 8-Puzzle

- Busca Exaustiva:
  - solução média em 22 passos
  - fator de ramificação médio: 3
  - $\approx 3^{22}$ estados possíveis
  - $\approx \frac{9!}{2}$ (controlando os estados repetidos)



Cada problema **exige** uma função heurística diferente

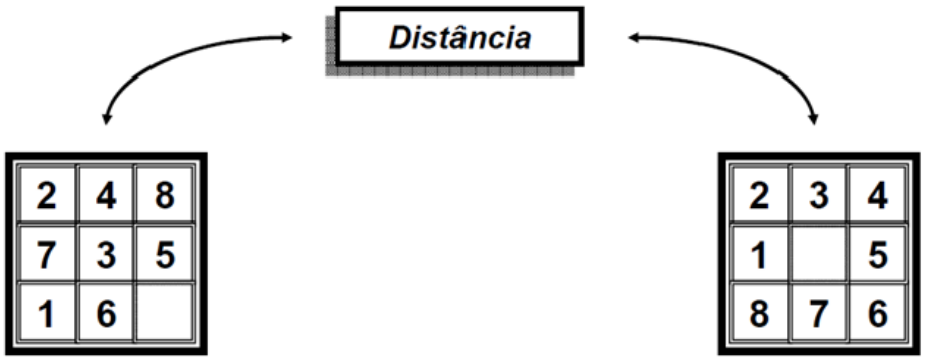
Como escolher uma boa função heurística para o jogo 8-Puzzle?

Definindo Heurísticas

h_1(2 4 8 / 7 3 5 / 1 6) = *A quantidade de peças for a do lugar*

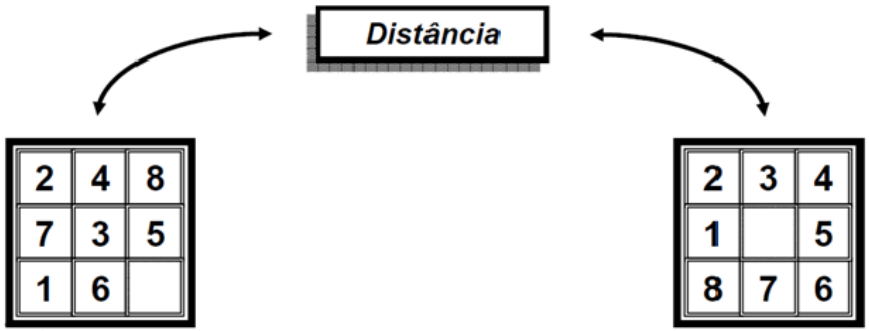
= 7

Definindo Heurísticas



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

Definindo Heurísticas



h_2() = *Número de movimentos necessários para colocar cada peça no seu lugar*

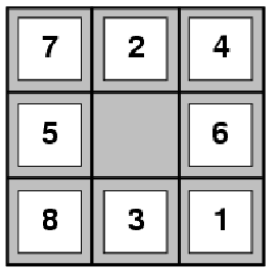
= 10

Definindo Heurísticas

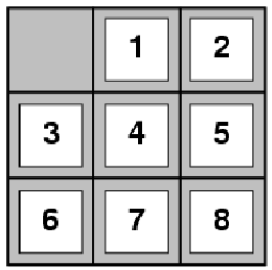
- Para o quebra-cabeça de 8 peças
  - h_1 = número de peças fora do lugar (função de prioridade Hamming)
  - h_2 = distância total à la Manhattan (função de prioridade Manhattan)
    - ★ número de quadrados da localização desejada de cada peça

Distância Manhattan

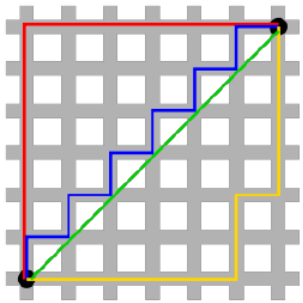
$$d_1(p,q) = \| p - q \|_1 = \sum_{i=1}^{n} |p_i - q_i|$$



Start State



Goal State



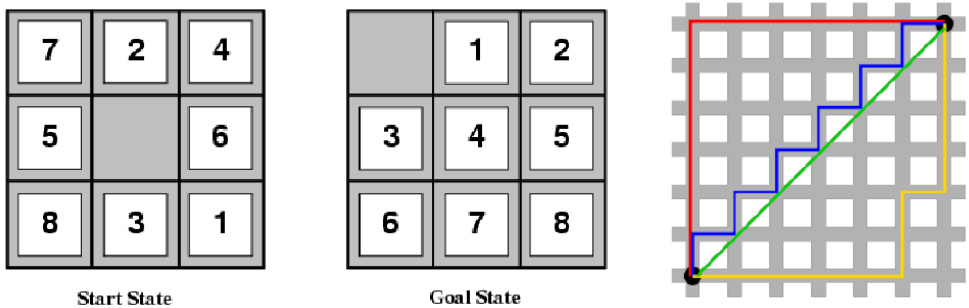
- $h_1(S) = ?$
- $h_2(S) = ?$

Definindo Heurísticas

- Para o quebra-cabeça de 8 peças
  - h_1 = número de peças fora do lugar (função de prioridade Hamming)
  - h_2 = distância total à la Manhattan (função de prioridade Manhattan)
    - ★ número de quadrados da localização desejada de cada peça

Distância Manhattan

$$d_1(p,q) = \| p - q \|_1 = \sum_{i=1}^{n} |p_i - q_i|$$



- $h_1(S) = 8$
- $h_2(S) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$

Definindo Heurísticas

- Heurísticas h_1 e h_2
  - $h_1 = 5$ (função de prioridade Hamming)
  - $h_2 = 10$ (função de prioridade Manhattan)

board

1	2	3	4	5	6	7	8
×	×	✓	✓	×	×	✓	×

Hamming = 5

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

Manhattan = 10
(1 + 2 + 2 + 2 + 3)

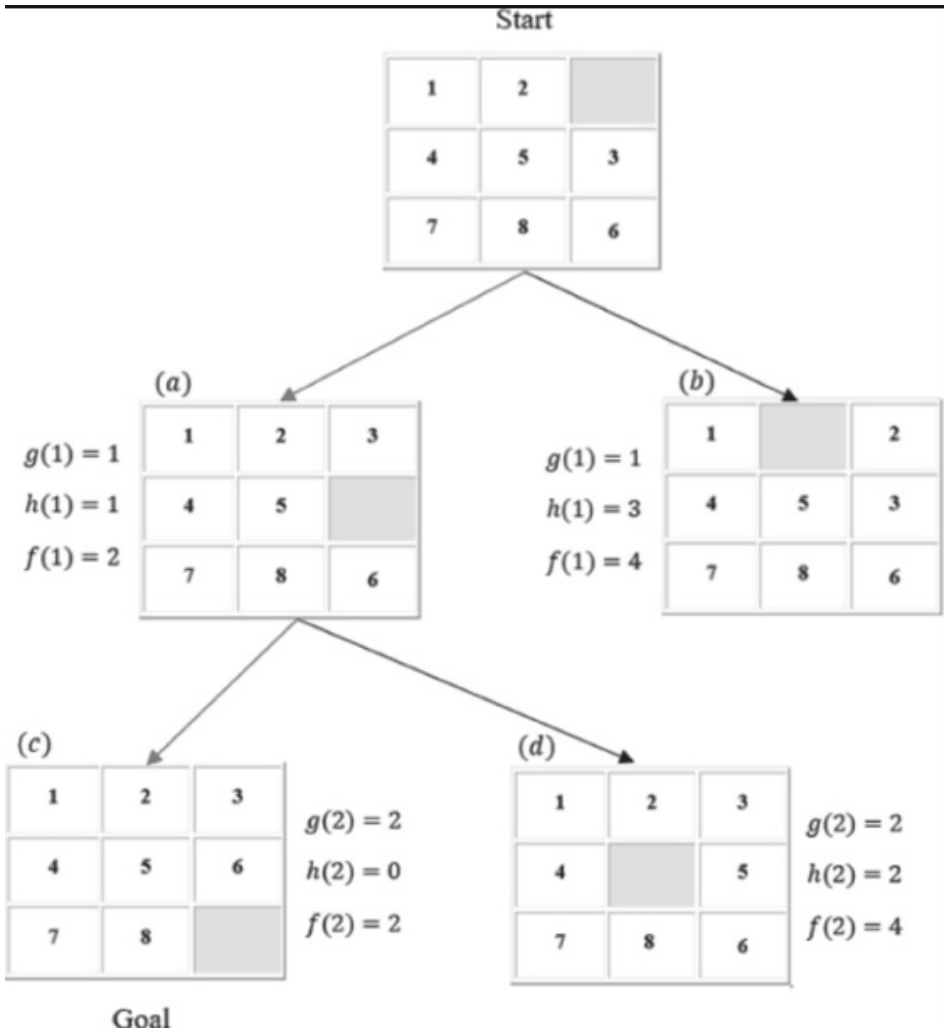
goal

```
8  1  3          1  2  3          1  2  3  4  5  6  7  8     1  2  3  4  5  6  7  8
4     2          4  5  6          ----------------------     ----------------------
7  6  5          7  8             1  1  0  0  1  1  0  1     1  2  0  0  2  2  0  3

initial             goal             Hamming = 5 + 0            Manhattan = 10 + 0
```

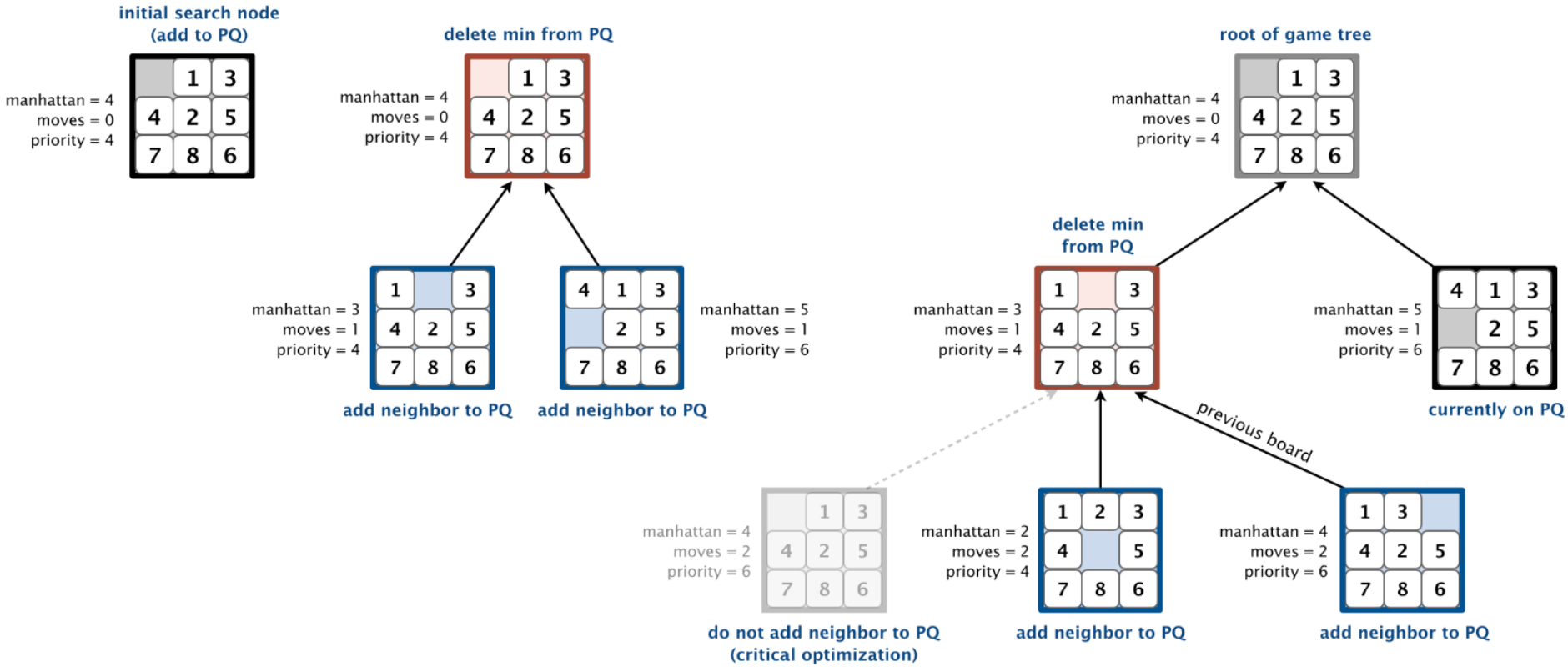
Definindo Heurísticas

- Exemplo de espaço de estados (parcial) considerando as heurísticas (prioridades de Hamming e de Manhattan) para o algoritmo A^*



Definindo Heurísticas

- Exemplo de espaço de estados (parcial) considerando as heurísticas (prioridades de Hamming e de Manhattan) para o algoritmo A^*



Definindo Heurísticas

- Como escolher uma boa função heurística para o quebra-cabeça de 8 peças?

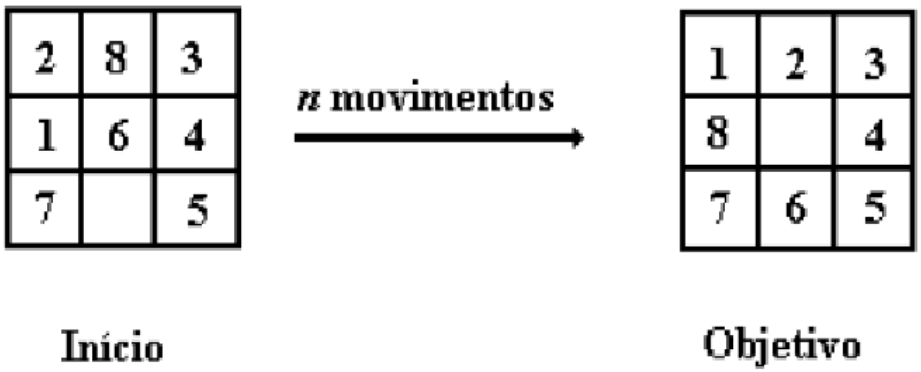
h_1 = número de elementos fora do lugar

h_2 = soma das distâncias de cada número à sua posição final

Qual das heurísticas é melhor?

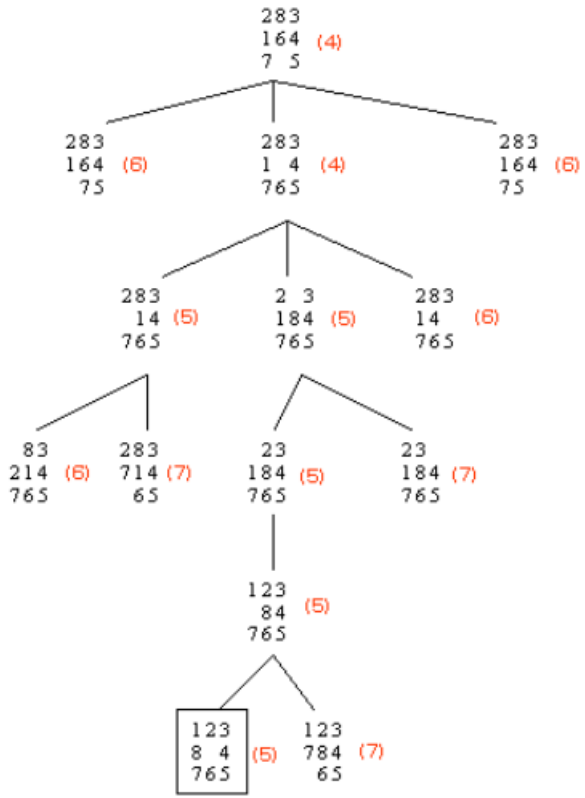
Definindo Heurísticas

- Função $h_2(n)$
  - espaço de estados gerado é menor
  - algoritmo acha mais rapidamente a solução



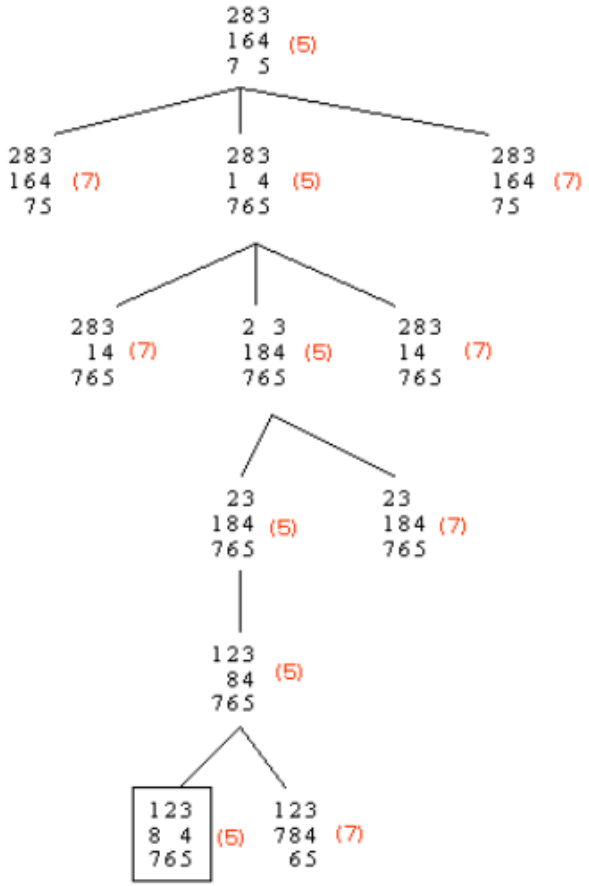
Definindo Heurísticas

- Espaço de estados gerado com h_1
  - para cada estado está indicado entre parênteses o valor da função heurística
  - não são considerados os nós que aparecem por mais de uma vez



Definindo Heurísticas

- Espaço de estados gerado com h_2



Medindo a Qualidade de uma Heurística

- Medida por meio do fator de expansão efetivo ou fator de ramificação efetiva ou b^*
  - fator de expansão de uma árvore uniforme com $n+1$ nós e nível de profundidade d

$$n + 1 = 1 + b^* + (b^*)^2 + ... + (b^*)^d$$

  - n = total de nós gerados pelo A^* para um problema
  - d = profundidade da solução
- Mede-se empiricamente a qualidade de h a partir do conjunto de valores experimentais de n e d
  - uma boa função heurística terá o b^* muito próximo de 1

Medindo a Qualidade de uma Heurística

- É sempre melhor utilizar uma função heurística com valores mais altos, desde que o tempo para computá-la não seja muito grande!
- Exemplo: h_2 melhor que h_1
- h_i domina $h_k \rightarrow h_i(n) \geq h_k(n)$, $\forall n$ no espaço de estados
- No exemplo anterior: h_2 domina h_1
- Isso pode ser traduzido na forma
  - a heurística 2 é melhor que a heurística 1, pois terá um menor fator de ramificação

Referências e Leituras Complementares

- Cap. 03 $\rightarrow$ livro Russel e Norvig
- Cap. 04 $\rightarrow$ livro Ben Coppin