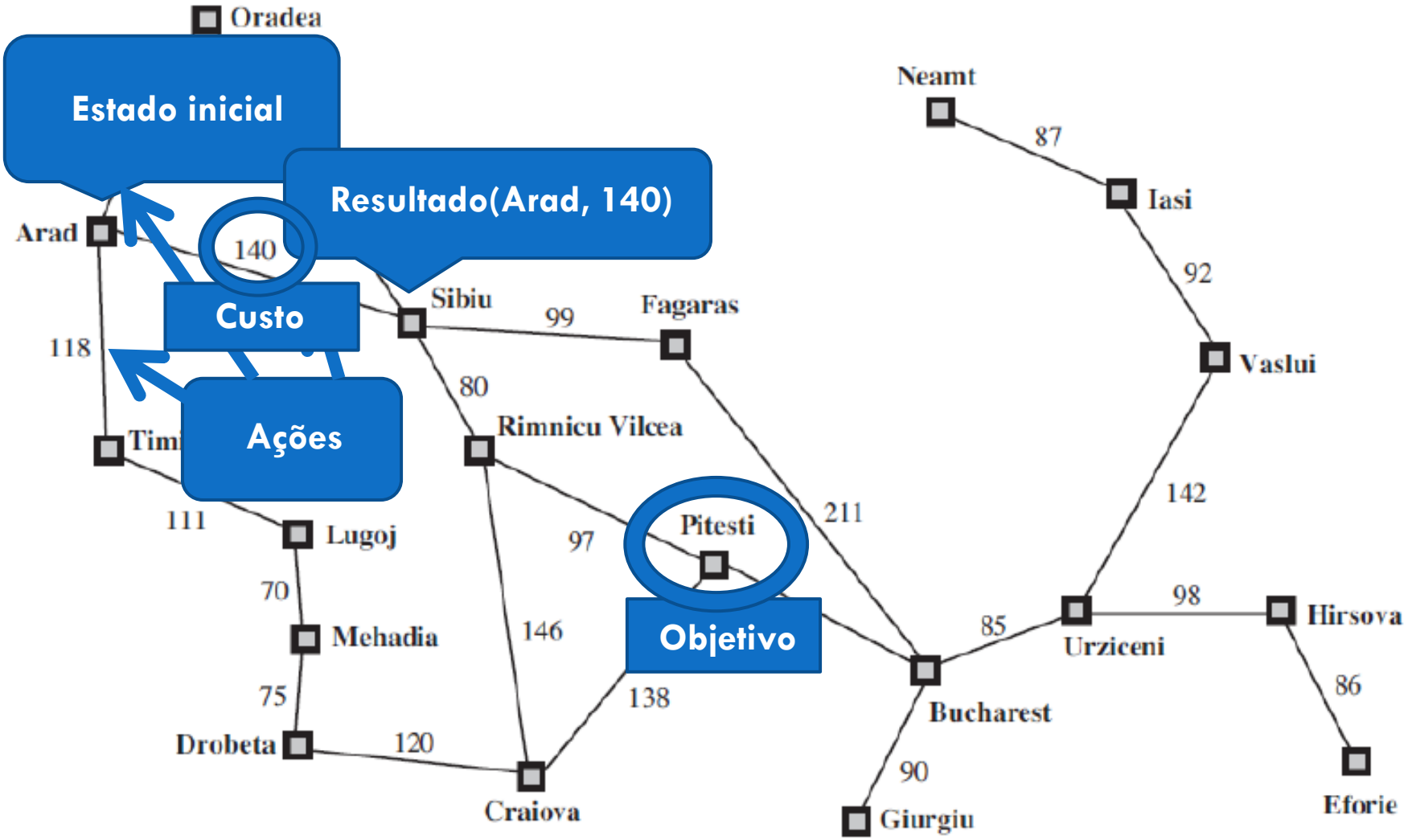
RESOLUÇÃO DE PROBLEMAS POR MEIO DE BUSCA

Formulação de um problema

- **Estado inicial:** onde o agente começa
- **Ações (Função Sucessor):** possíveis ações que o agente pode executar
- **Resultado (modelo de transição):** dado um estado e uma ação, especifica um novo estado
  - Result(S,a) = S'
- **Teste de Objetivo:** determina se o estado é objetivo ou não
- **Custo de Caminho:** dado um caminho (sequencia de transições estado/ação) determina um número que é o custo daquele caminho
  - Na maioria dos problemas, a custo será aditivo, sendo assim, o custo de um caminho é a soma dos passos individuais
- **Custo de Passo:** a implementação do custo de caminho envolve o custo de passo. Este recebe um estado, uma ação e um resultado e retorna um número que é o custo da ação
  - StepCost(S,a,S') = n

Mapa da Romênia

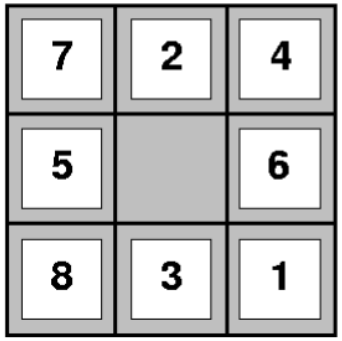


Tipos de solução

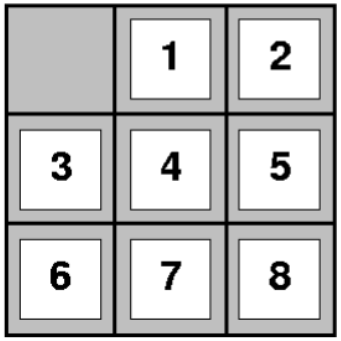
- Solução é um caminho desde o estado inicial até o estado final

- **Solução ótima:** tem o menor custo de caminho dentre todas as soluções existentes

Exemplos de ambiente quebra-cabeças 8 peças



Start State



Goal State

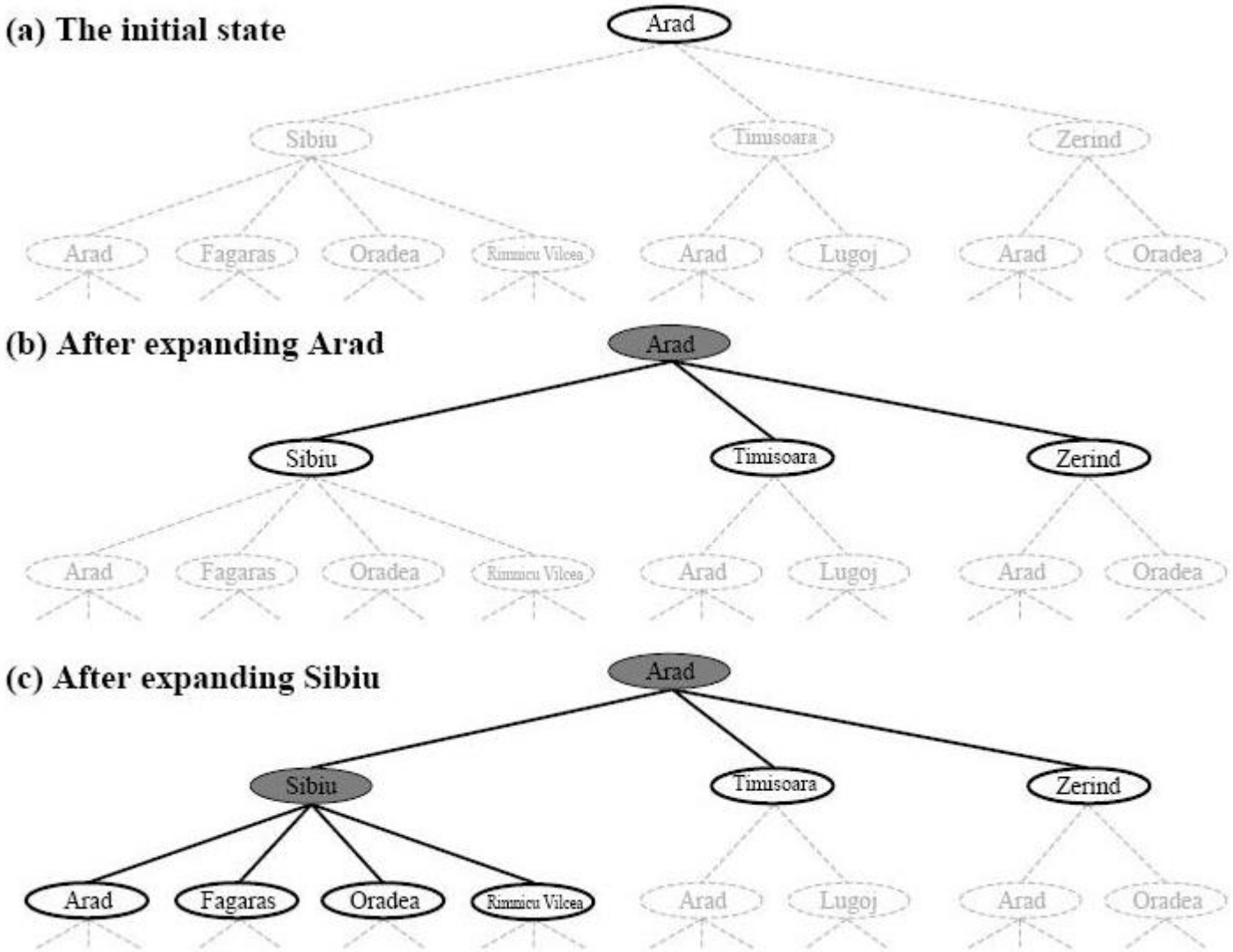
9!/2 =
181.440 estados

15 peças = 1,3 trilhão de estados
24 peças = 10^{25} estados

- Estados: posição das peças e do espaço vazio
- Estado inicial e objetivo representado na imagem
- Função sucessor: esquerda, direita, acima e abaixo
- Teste: verifica se o estado corrente é o objetivo
- Custo: contagem de passos

Outros exemplos: http://centurion2.com/AIHomework/AI120/ai120.php

Árvore de busca



Desempenho de um algoritmo

- **Completeza:** encontra uma solução se esta existir;
- **Otimização:** encontra a solução ótima
- **Complexidade de tempo:** tempo gasto para encontrar uma solução
- **Complexidade de espaço:** memória necessária para encontrar uma solução.

Estratégias de busca sem informação

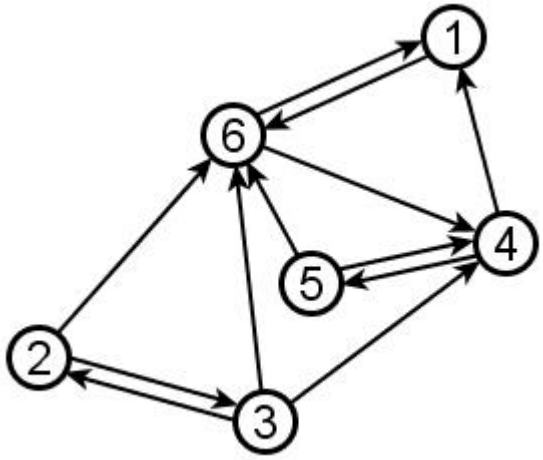
Busca em largura e busca em profundidade

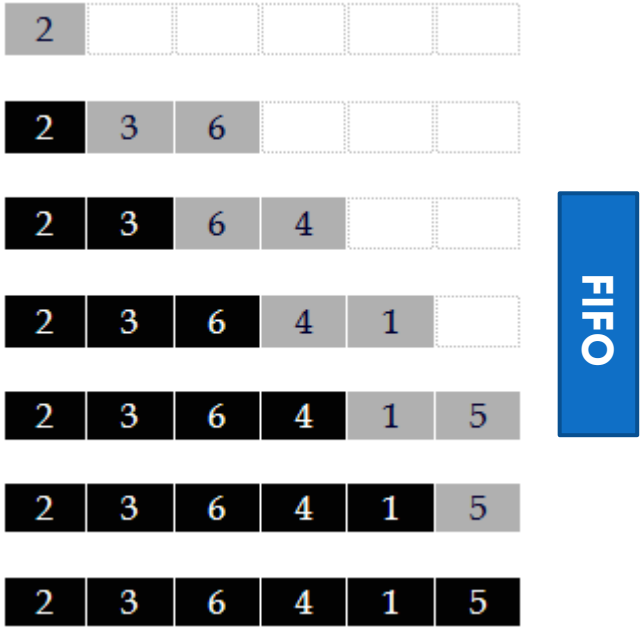
Buscas sem informação

- Também chamada de **busca cega**
- Não é fornecida informação adicional sobre estados além nas definidas no problema

Busca em largura

- BFS – Breadth-first search
- Expande o nó raiz e em seguida todos os sucessores do nó raiz. Depois, os sucessores desses nós, e assim por diante





Desempenho da BFS

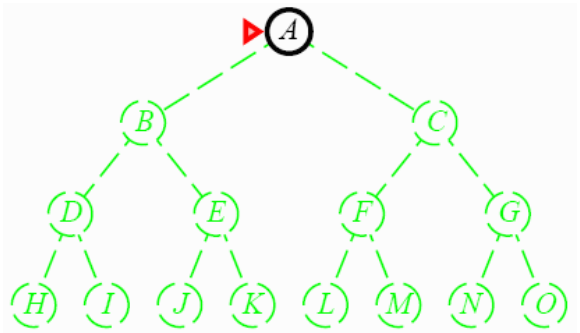
- Completa desde que o fator de ramificação seja finito
- Não é ótimo pois encontra o objetivo mais raso
- Tempo: com fator de ramificação uniforme *b* serão gerados $b + b^2 + b^3 + \ldots + b^d = O(b^d)$
- Espaço: como são mantidos os nós em memória existirão $O(b^{d-1})$ nós explorados e $O(b^d)$ na borda

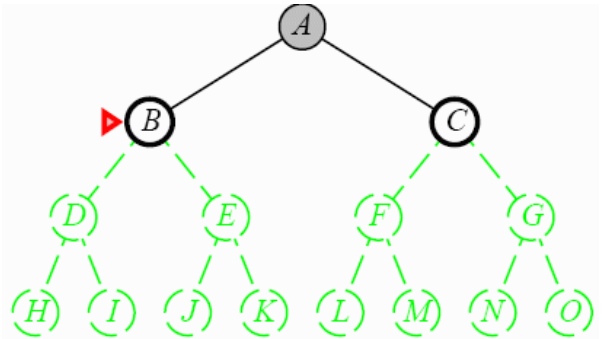
Busca de custo uniforme

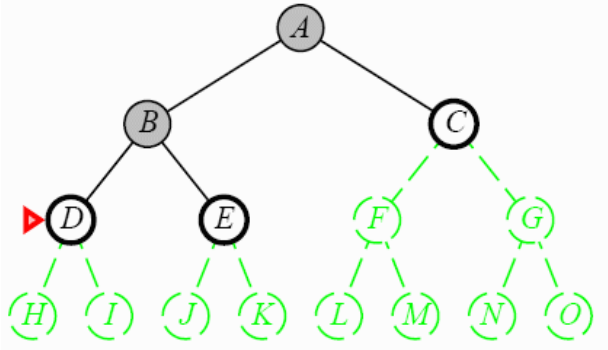
- Expande o nó *n* com o *custo de caminho* *g(n)* mais baixo
  - Armazenamento da borda como um fila de prioridade ordenada por *g*

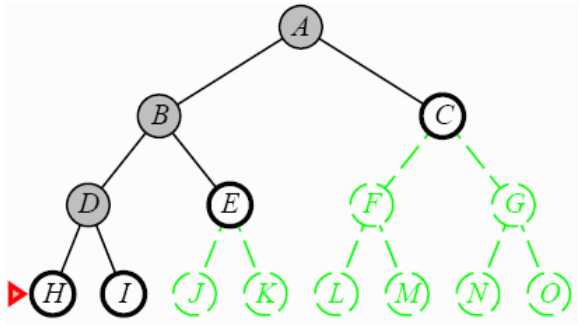
Busca em profundidade

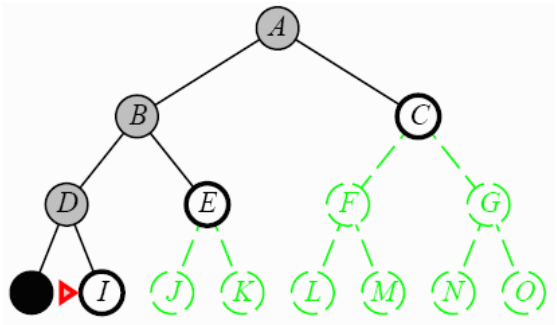
- DFS – Depth-first search
- Expande o nó mais profundo na borda atual da árvore
- Não havendo mais sucessores, a busca retorna à próxima profundidade acima que não foi explorada

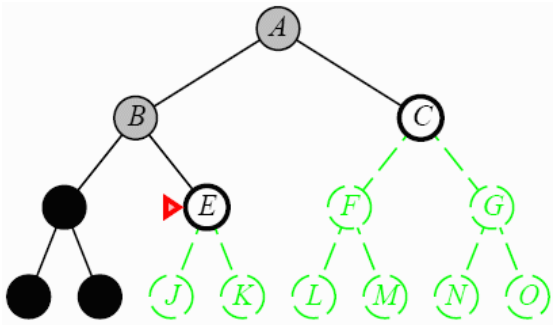

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O

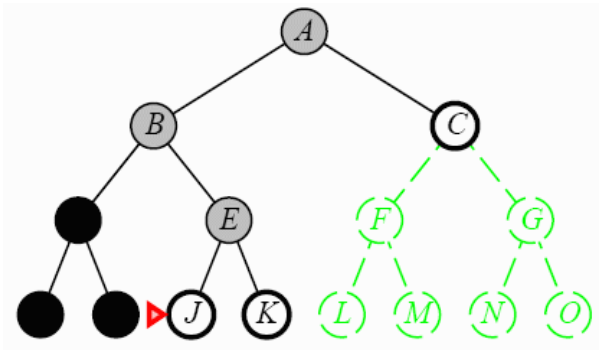

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O

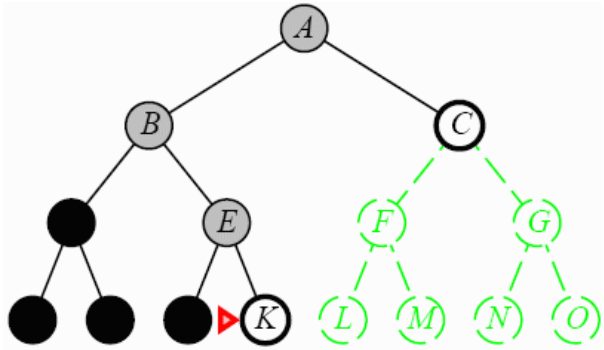

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O

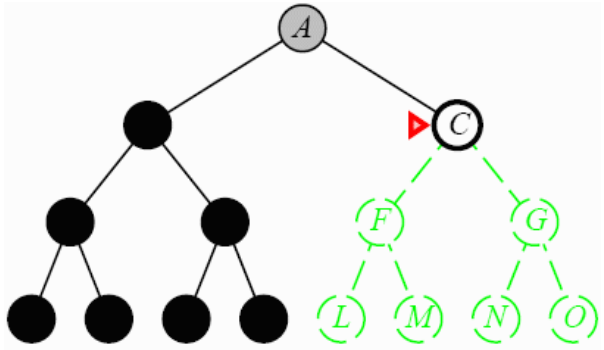

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O


A
B
C
D
E
F
G
I
J
K
L
M
N
O


A
B
C
E
F
G
J
K
L
M
N
O


A
B
C
E
F
G
J
K
L
M
N
O


A
B
C
E
F
G
K
L
M
N
O


A
C
F
G
L
M
N
O

Desempenho da DFS

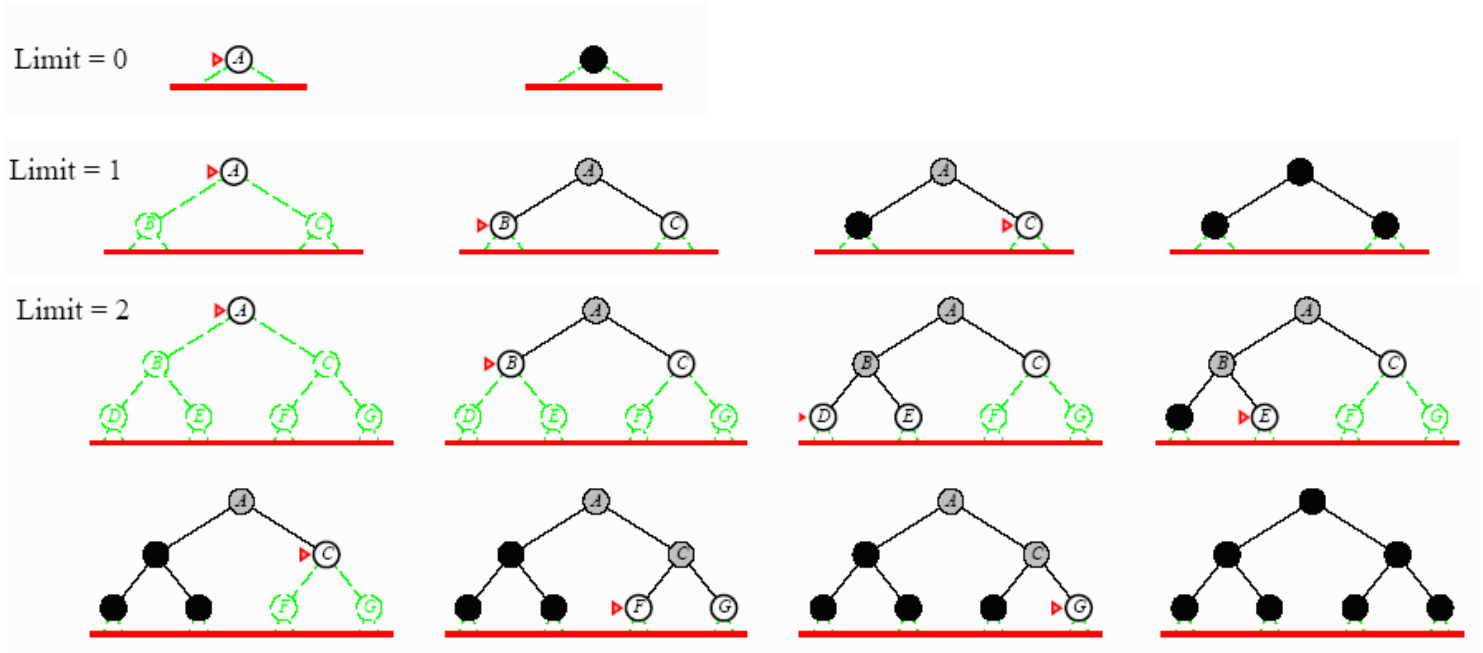
- Pode percorrer um caminho muito longo (as vezes infinito), portanto, a busca em profundidade não é completa e nem ótima
- Complexidade temporal: no pior caso, todos os nós são gerados, portanto: $O(b^d)$
- Só precisa armazenar um único caminho da raiz até um nó folha, e os nós irmãos não expandidos
  - Para ramificação *b* e profundidade máxima *d*, a complexidade espacial é O(bd)

Busca em profundidade limitada

- Resolve o problema de busca em profundidade em árvores infinitas
- Adiciona um limite para profundidade da busca
  - nós na profundidade limite são tratados como se não tivessem sucessores

Busca por aprofundamento iterativo

- Busca em profundidade aumentando gradualmente o limite de profundidade
- Usado quando se tem espaço de busca grande e profundidade não conhecida



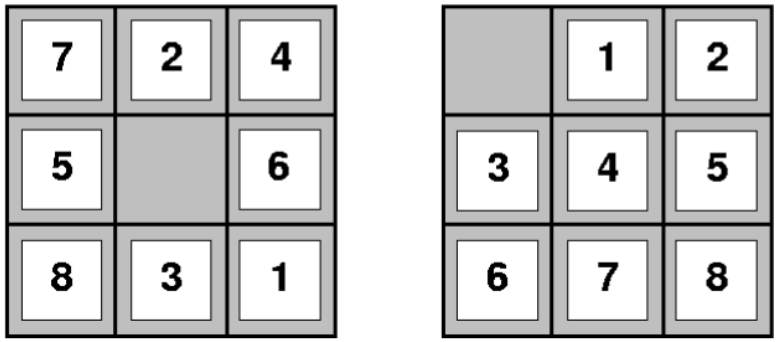
Busca informada (heurística)

Busca informada

- Busca a **melhor escolha** expandindo os nós com base em uma **função de avaliação**
- A função precisa ser uma heurística admissível
  - Heurística admissível é a que nunca superestima o custo de atingir o objetivo

Funções heurísticas

- H_1 = número de peças fora do lugar
  - $H_1 = 8$
- H_2 = soma das distâncias das peças de suas posições-objetivo (distância de Manhattan)
  - $H_2 = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$



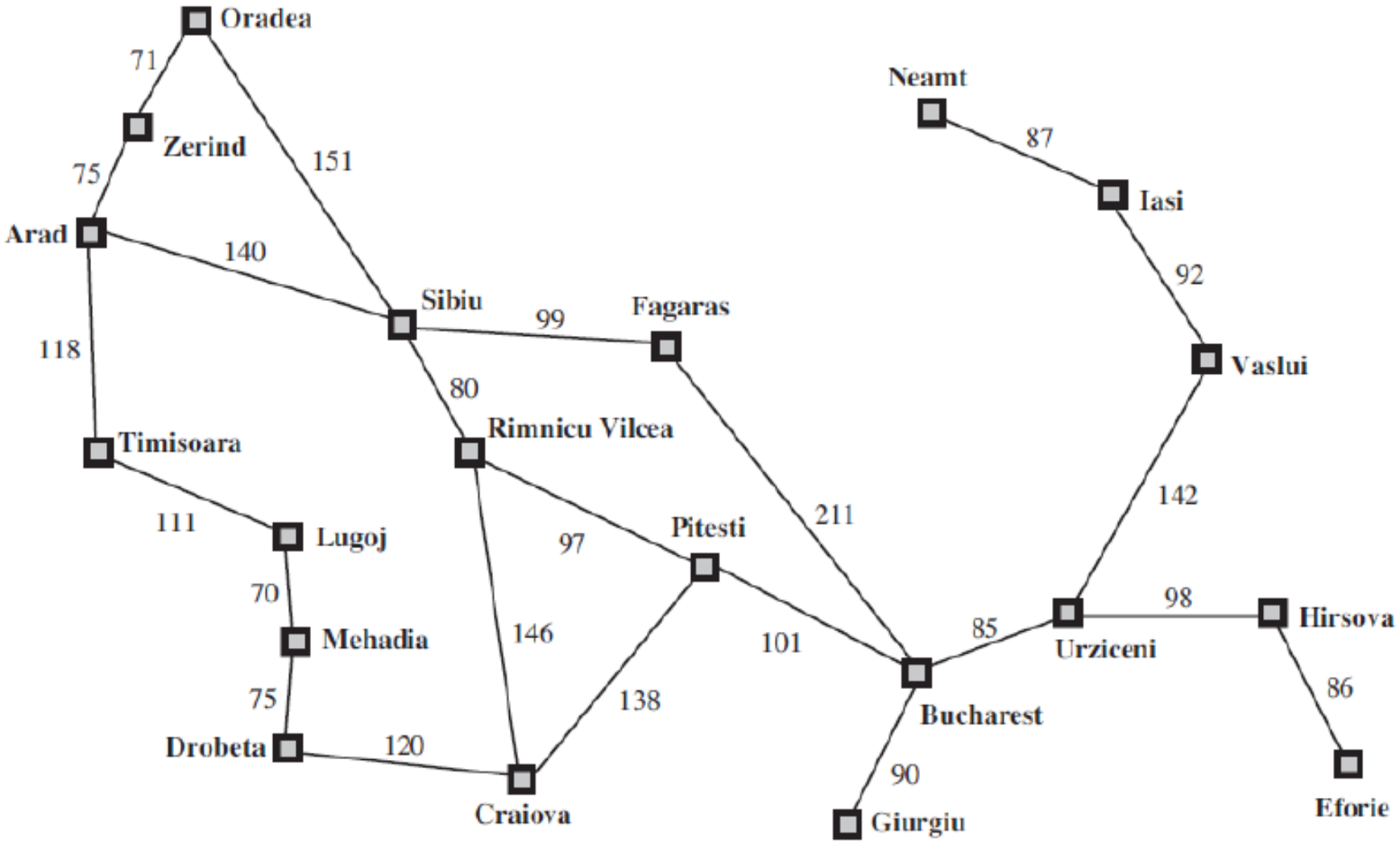
Busca gulosa de melhor escolha

- Expande o nó que está mais próximo do objetivo
- Por exemplo, no mapa da Romênia, poderia ser utilizado a distância em linha reta para o objetivo
- É uma busca incompleta
  - Avaliar ir de Iasi para Fagaras

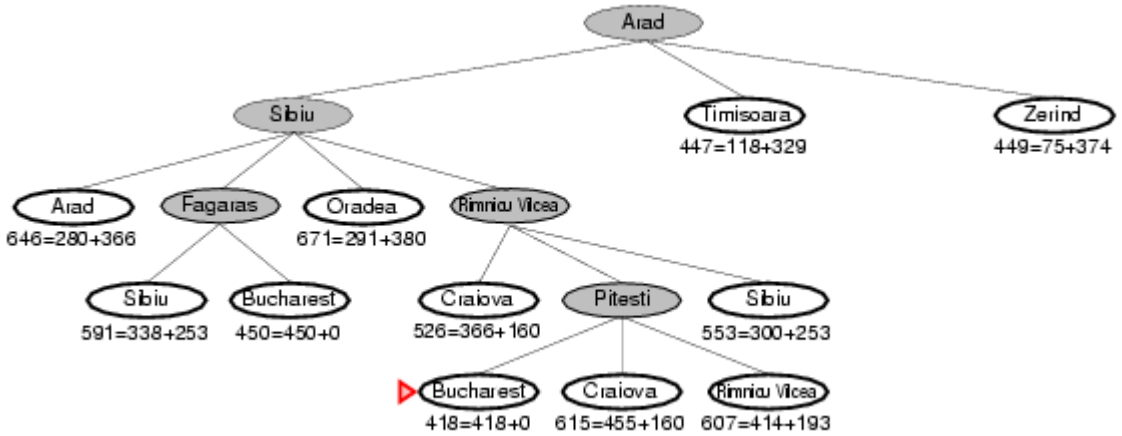
Busca A*

- Avalia os nós através da combinação de $g(n)$, o custo para alcançar o nó, e $h(n)$, o custo para ir do nó ao objeto
  - $F(n) = g(n) + h(n)$
- É um método de busca ótima

Mapa da Romênia



Busca A*



h(n) = distância em linha reta para o objetivo