Busca Informada

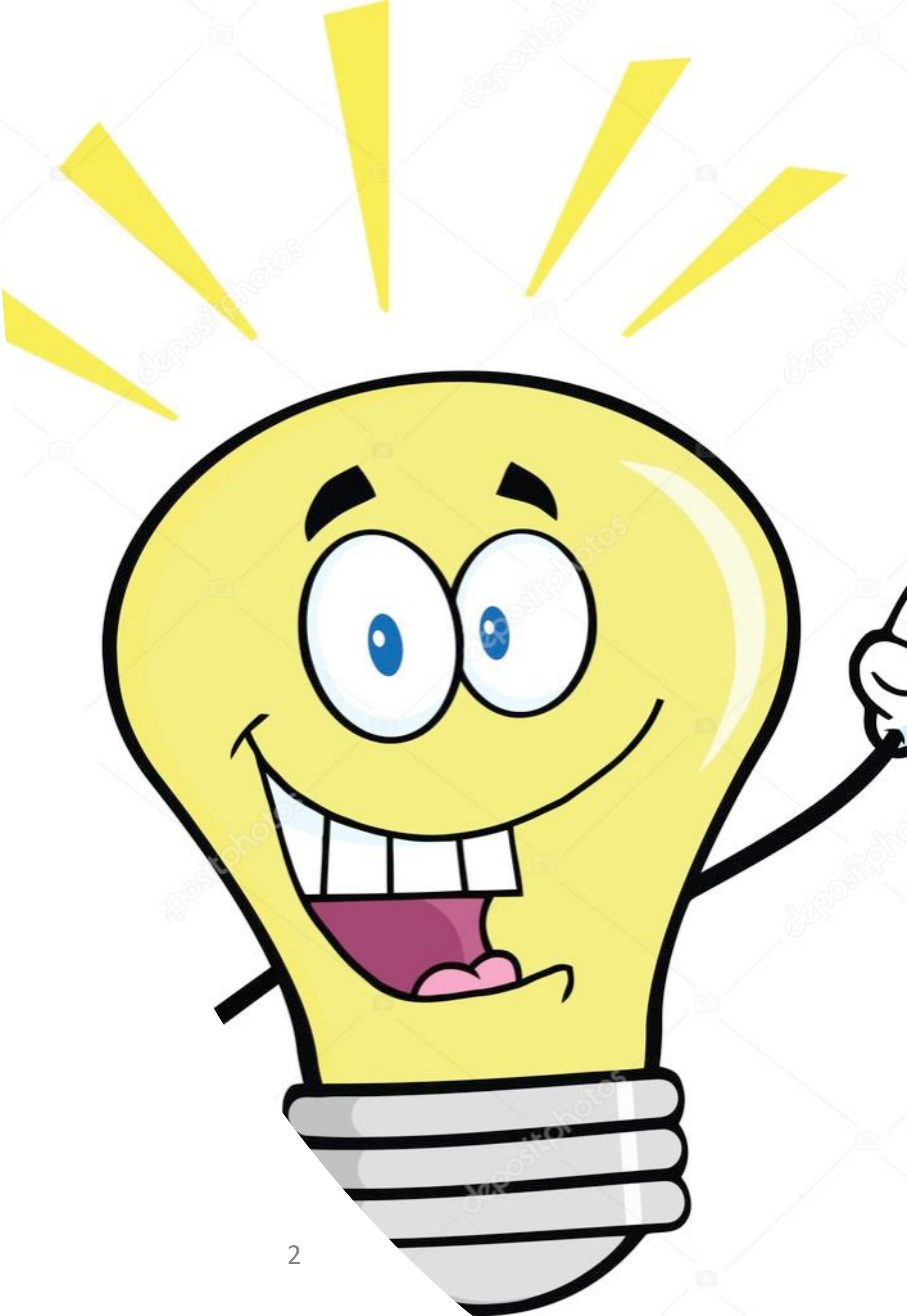
Inteligência Artificial
PCS3438

Escola Politécnica da USP

Engenharia de Computação (PCS)

Busca Informada

- Usa conhecimento específico do problema na busca da solução
- Também chamadas de **Busca Heurística**
- Mais eficientes que busca não informada



Busca Informada (ou Busca Heurística): Ideia básica

Busca pela Melhor Escolha (BME)

(**Best-first search**)

- Seleciona para expansão o nó que tiver o <u>mínimo custo estimado</u> até a meta, segundo uma função de avaliação f(n).
- Tipicamente f(n) usa uma função heurística h(n) que estima o custo da solução a partir de n.
- Na meta, h(n) = 0.

Greedy best-first search

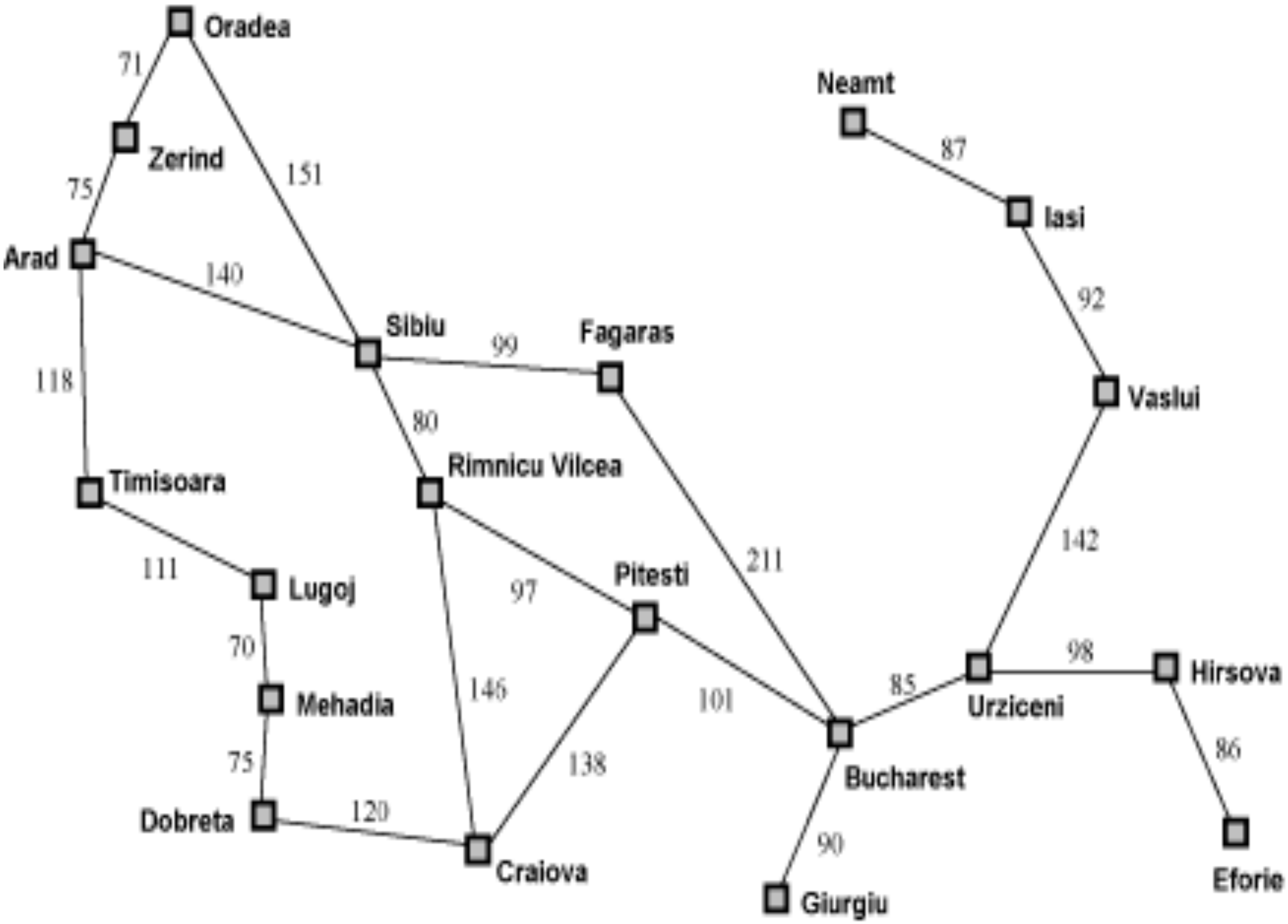
Busca gulosa pela melhor escolha

Avalia nós para expandir com base unicamente na função heurística:

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

Semelhante à busca em profundidade com retrocesso (*backtracking)*

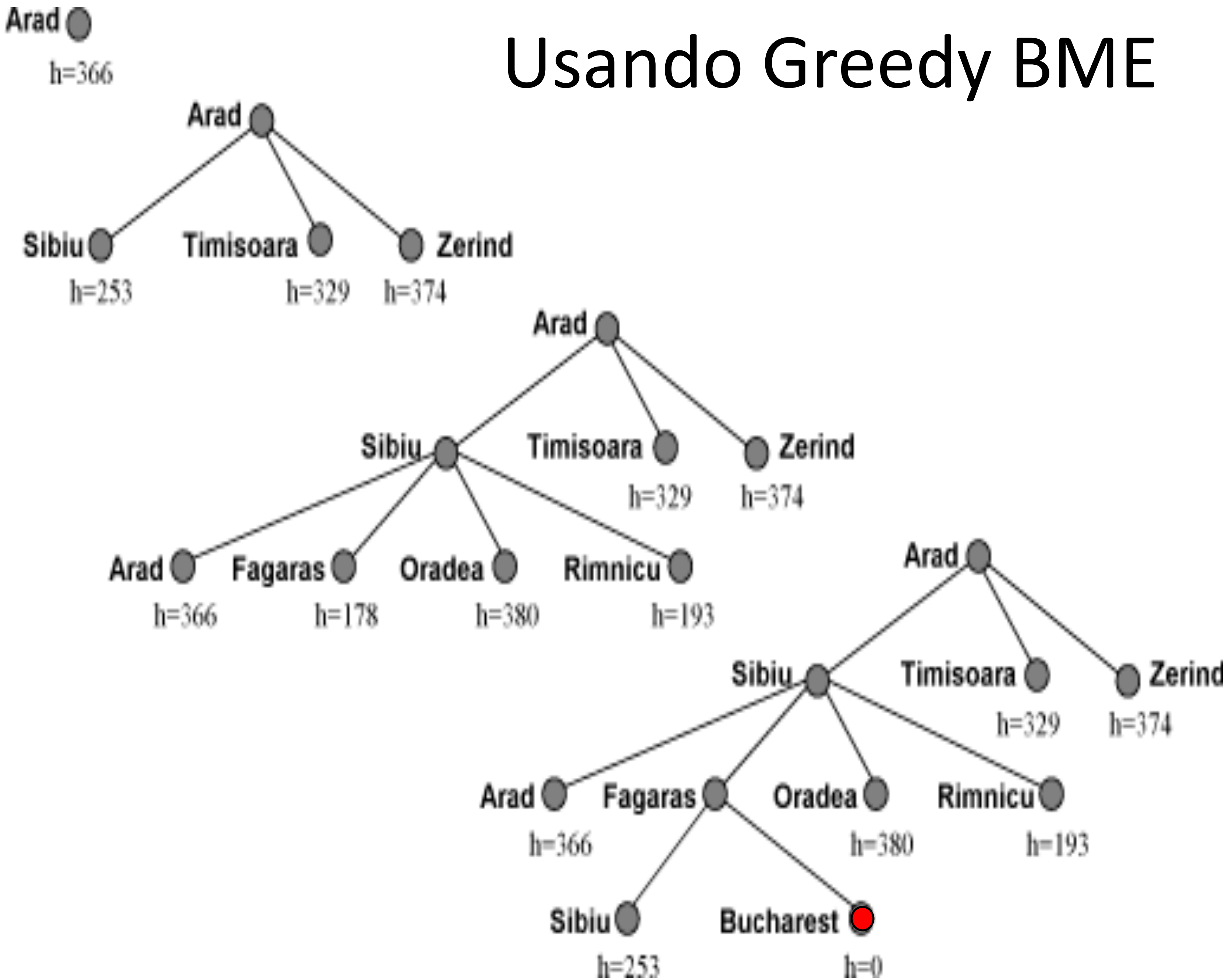
Exemplo: ir de Arad a Bucharest usando Greedy BME



Straight□line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Usando Greedy BME



Desempenho da Greedy BME

7

- **Não é completa**
  - pode entrar em ciclos e não encontrar a solução se não detectar estados repetidos (idem BP)
  - pode se perder em um caminho infinito e nunca retroceder para tentar outras opções (idem BP)
- **Não é ótima**
  - No ex: encontrou caminho (Arad, Sibiu, Fagaras, Bucharest) que é 32km maior que (Arad, Sibiu, Rimnicu Vilcea, Pitesti, Bucharest)
- **Complexidade de tempo e espaço no pior caso: O(bm)**
  - m é a máxima profundidade do espaço de busca

- **Dependendo do problema e da qualidade da heurística a complexidade pode ser substancialmente reduzida.**

BME mais “famoso”: A*

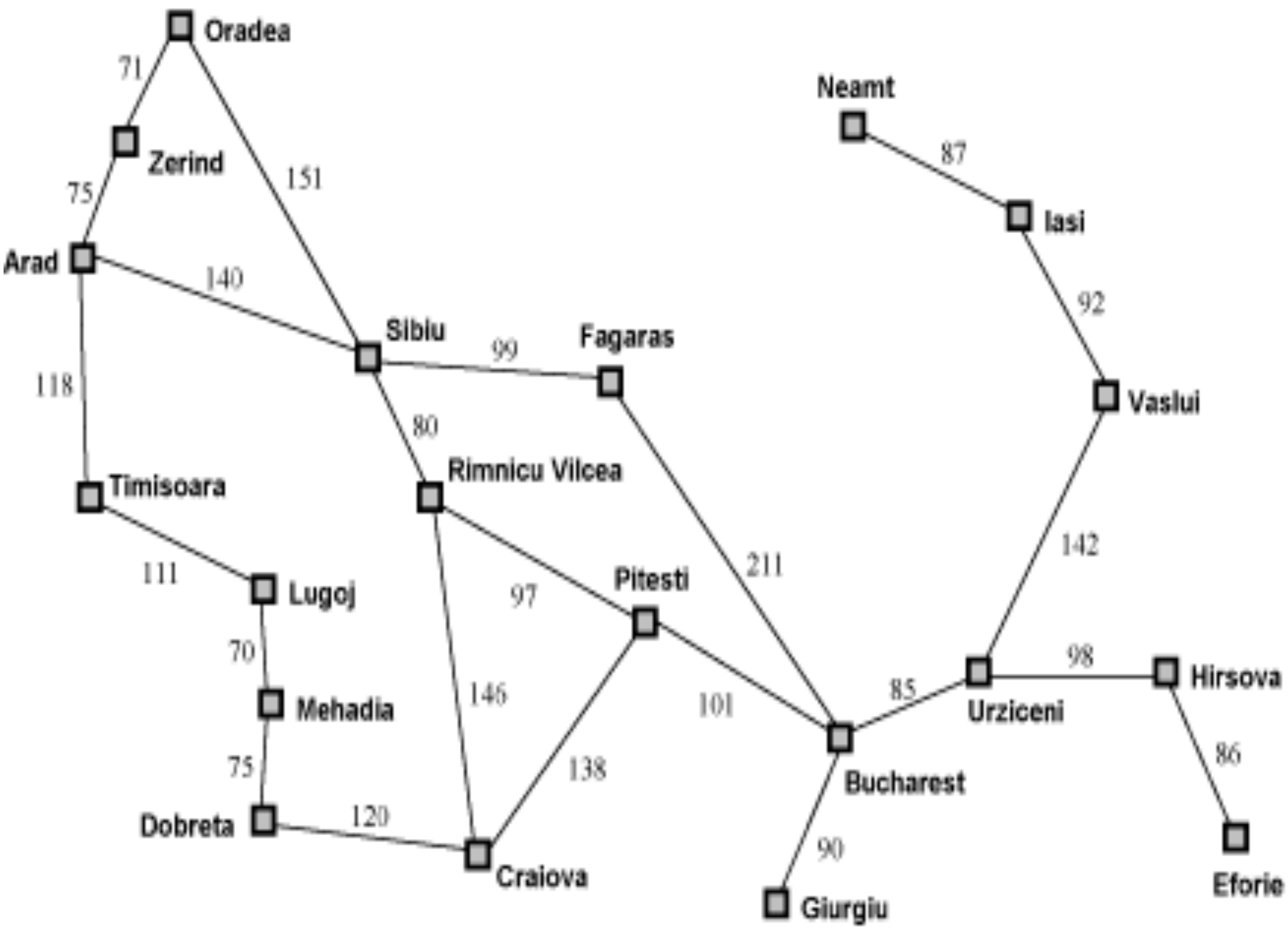
- Função de avaliação:

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

  - *g(n)* = distância (custo) **real** do nó inicial ao nó *n*
  - *h(n)* = distância (custo) **estimada** de *n* ao nó final
  - f(n) **estima** o custo da melhor solução que passa por n

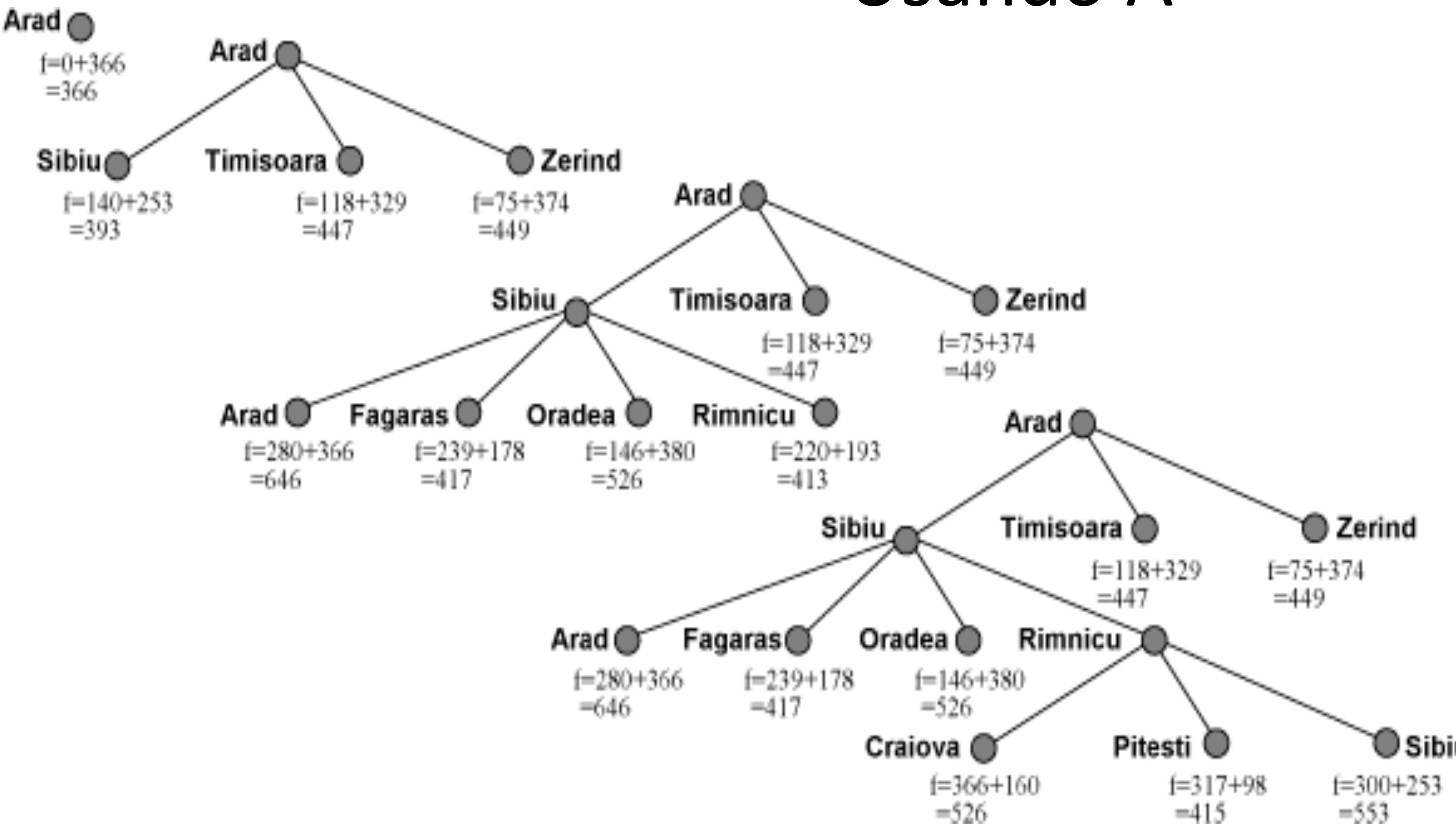
- A* expande o nó de menor valor de *f* na fronteira do espaço de estados
  - idêntico à Busca de Custo Uniforme, só que usa *g+h* em vez de *g*

Exemplo: ir de Arad a Bucharest usando A*

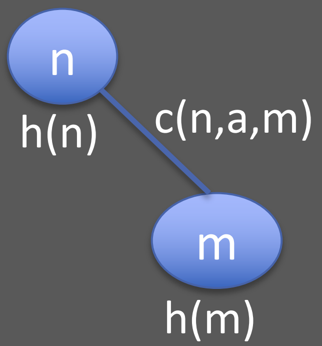


Straight‐line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Usando A*


Arad
f=0+366
=366
Arad
Sibiu
Timisoara
Zerind
f=140+253
=393
f=118+329
=447
f=75+374
=449
Arad
Sibiu
Timisoara
Zerind
f=118+329
=447
f=75+374
=449
Arad
Fagaras
Oradea
Rimnicu
f=280+366
=646
f=239+178
=417
f=146+380
=526
f=220+193
=413
Arad
Sibiu
Timisoara
Zerind
f=118+329
=447
f=75+374
=449
Arad
Fagaras
Oradea
Rimnicu
f=280+366
=646
f=239+178
=417
f=146+380
=526
Craiova
Pitesti
Sibiu
f=366+160
=526
f=317+98
=415
f=300+253
=553

Desempenho do A*



- **A* é completo e ótimo se h(n) for admissível e consistente**
  - **h admissível**: nunca superestima o custo de atingir a meta **(1)**
  - **h consistente (ou monotônica) (2)**

$$h(n) \leq c(n,a,m) + h(m), \quad \forall n, m$$

$$\text{ou } (h(n) - h(m)) \leq c(n,a,m)$$

- m é sucessor de n, gerado pela ação a; c(n,a,m) é o custo de sair de n e atingir m usando a.
- Se h é consistente, os valores de f(n) através de qualquer caminho são crescentes.

Busca A*: comentários

- A* é otimamente eficiente
  - nenhum outro algoritmo ótimo garante expandir menos nós que A*.
- Geralmente há crescimento exponencial do número de nós com o comprimento da solução (complexidade temporal).
- Mas o maior problema é a complexidade espacial: A* armazena todos os nós gerados!
- Função de avaliação: compromisso (conflito) entre:
  - tempo gasto na seleção de nó (computar h)
  - redução do espaço de busca

Busca Heurística com Memória Limitada

IDA* (Iterative Deepening A*)

- Similar ao BAI, porém seu limite é dado pela função de avaliação (f), e não pela profundidade (d).
- usa menos memória que A*

SMA* (Simplified Memory-Bounded A*)

- O número de nós guardados em memória é fixado previamente

HEURÍSTICAS – COMO DEFINIR?

Inventando Funções Heurísticas

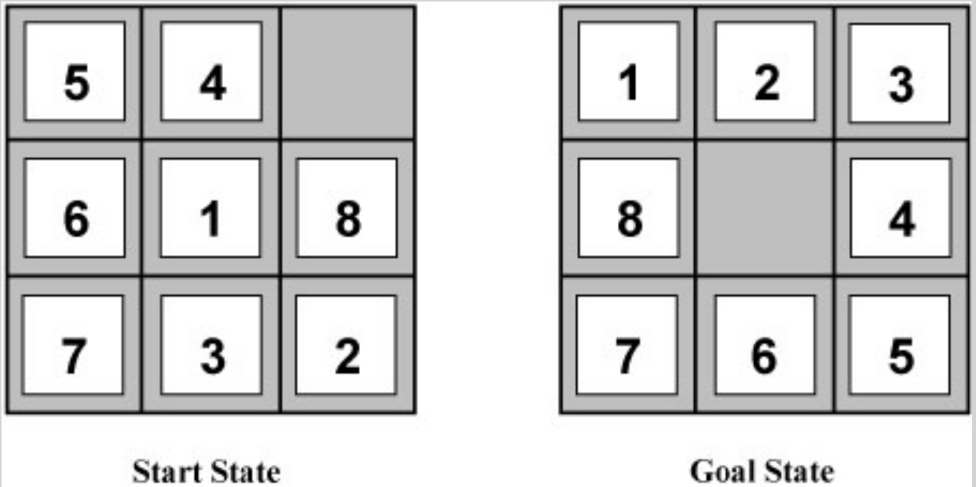
- Como escolher uma boa função heurística *h*?
  - *h* depende de cada problema particular.
  - *h* deve ser *admissível:* não superestimar o custo real da solução

- Exemplo: jogo dos 8 números
  - um número pode mover-se de A para B se A é adjacente a B e B está vazio
  - busca exaustiva (ou cega ou não informada):
    - solução média em 22 passos
    - fator de ramificação médio: 3
    - Assim $\approx 3^{22}$ estados possíveis

4	5	8
	1	6
7	2	3

Heurísticas para o jogo 8 números

- *h1* = no. de elementos fora do lugar
- *h2* = soma das distâncias de cada número à posição final (distância de Manhattan)



h1 = 7

h2 = 2+3+3+2+4+2+0+2 = 18

Qualidade da função heurística

- Mede-se empiricamente a partir do conjunto de valores experimentais de **N** e **d**.
- Dada através do fator de expansão efetivo b*
  - b* é o 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
      - Ex: N =52, d = 5 → b* = 1.92
- Uma boa função heurística terá o b* muito **próximo de 1**.

Escolhendo Funções Heurísticas

18

- Usar uma função heurística de valores mais altos, desde que seja admissível e que o tempo para computá-la não seja muito grande!
  - ex. h_2 melhor que h_1
- h_i **domina** $h_k \Rightarrow h_i(n) \geq h_k(n), \forall n$
  - ex. h_2 domina h_1.
- Se várias h e nenhuma domina a outra, usa-se uma **heurística composta**:
  - $h(n) = \max (h1(n), h2(n), \dots ,hm(n))$

Como inventar funções heurísticas admissíveis?

(1) Relaxar o problema (versão simplificada)
(2) Usar informação estatística
(3) Identificar atributos relevantes do problema e usar aprendizagem

(1) Relaxando o problema

- Operadores relaxados:
  - 1. Uma peça pode se mover para lugares adjacentes, mesmo que ocupados
    - h_2 seria o custo da solução “correta” neste jogo
  - 2. Uma peça pode se mover para qualquer lugar vazio, mesmo que não adjacente
  - 3. Uma peça pode se mover para qualquer lugar
    - h_1 seria o custo da solução “correta” neste jogo

O custo da solução ótima de um problema relaxado é uma heurística admissível para o problema original!!!

(2) Usando informação estatística

- Funções heurísticas podem ser “melhoradas” com informação estatística:
  - executar a busca com um conjunto de treinamento (ex., 100 configurações diferentes do jogo), e computar os resultados.
  - se, em 90% dos casos, quando $h(n) = 14$, a distância real da solução é 18, então, quando o algoritmo encontrar 14 para o resultado da função, pode substituir esse valor por 18.
- Informação estatística expande menos nós, porém elimina admissibilidade:
  - em 10% dos casos do problema acima, $h(.)$ poderá <u>superestimar</u> o custo da solução!

(3) Aprendendo heurísticas por experiência

- Resolve o jogo diversas vezes e computa o custo da solução, relacionando a algum atributo do problema.
  - Ex1: atributo $x_1(n)$ = número de peças fora do lugar no início do jogo; para cada valor de atributo, determinar experimentalmente o custo médio da solução.
    - Ex: para $x_1(n)=5$, resolvo 100 vezes o problema e concluo que o custo médio da solução é 14 passos
  - Ex2: atributo $x_2(n)$ = número de pares de peças adjacentes que também são adjacentes na configuração de solução

- Uso $x_1(n)$ e/ou $x_2(n)$ para estimar $h(n)$:
  - $h(n) = c_1.x_1(n) + c_2.x_2(n)$, ajustando (aprendendo) c_1 e c_2 da melhor forma para os dados de custo da solução.

Bibliografia

- Busca cega e heurística:
  - Capítulo 3 do livro texto (Russel & Norvig, Inteligência Artificial, 3a. Edição)