

Pesquisa Informada (Heurística)

Msc. Eng. Solander Patrício Lopes Agostinho

Avaliação em algoritmos de pesquisa

- **Complexidade de tempo**
 - Quantidade de nós percorridos
- **Complexidade de espaço**
 - Número máximo de nós armazenados em memória
- **Completeza**
 - Se existe solução, algoritmo atinge?
- **Admissibilidade**
 - Garantia de atingir solução ótima, quando a mesma existe

Ideia base (Heurística)

- *“Utilizar informações específicas do **domínio** para ajudar na decisão do processo de busca ou pesquisa”*

Problemas com pesquisas não informadas

- Complexidade elevada devido ao número de alternativas
 - Grafos
 - DFS
 - BFS

Pesquisa informada

- Utilizam informação heurística sobre o problema
- Objectivo: Encurtar a busca no espaço de estados
- Estimativa:
 - Quanto o nó é promissor em função da meta a que se pretende atingir.

Avaliando o método...

- Uso conhecimento específico do problema
 - **Objectivo**: Escolher o próximo nó a ser expandido
 - Aplica-se uma **função de avaliação** a cada nó na fronteira do espaço de estados.

Função Heurística

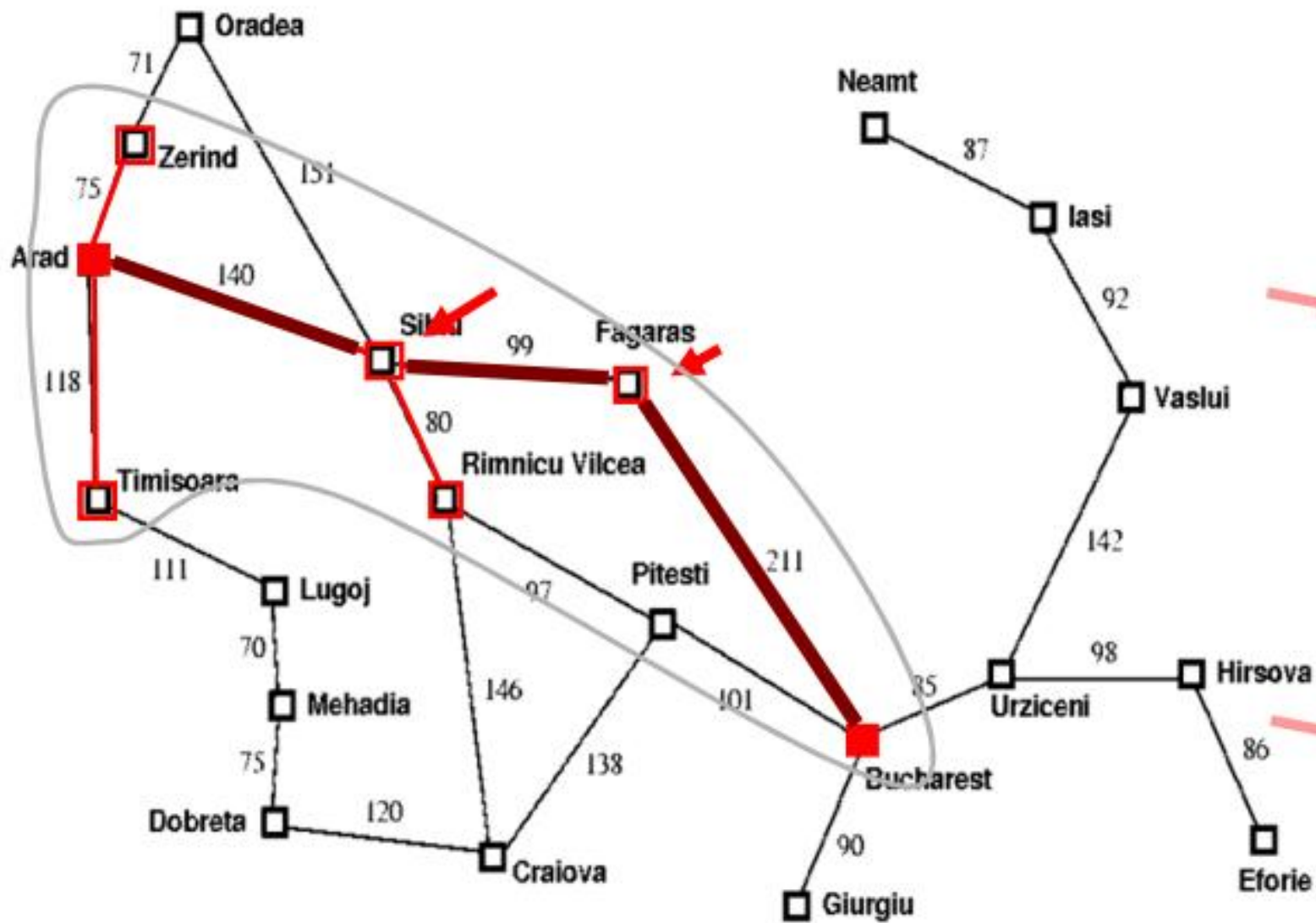
- **estima** o custo do caminho entre o nó n e o objetivo
- Muda de comportamento dependente do problema
 - Ex. encontrar a rota mais próxima entre Luanda e Benguela
 - $h_{dd}(n)$ = Distância entre nó inicial e objetivo
- Critérios para determinação da Função heurística
 - ela deve ser admissível e nunca *superestimar* o custo real da solução
 - **$h(n) \leq h^*$ (h^* é o custo real da solução)**
 - *Hdd*: *admissível* porque o caminho mais curto entre dois pontos é sempre uma linha reta

Best First- Search

- Greedy search (Busca gulosa)
- A^*

Greedy search

- Semelhante à busca em profundidade com backtracking
 - Tenta expandir o nó mais próximo ao nó final com base na estimativa feita pela função heurística h
- Custo de busca é minimizado
 - não expande nós fora do caminho
- Escolhe o caminho que é mais econômico à primeira vista
- Não é ótima... (semelhante à busca em profundidade)
 - porque só olha para o futuro!
- ... nem é completa:
 - pode entrar em “loop” se não detectar a expansão de estados repetidos
 - pode tentar desenvolver um caminho infinito
- Custo de tempo e memória: $O(bd)$
 - guarda todos os nós expandidos na memória



Distância em linha
reta para 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

Greedy search

- Em muitas situações não escolhe o melhor caminho
 - PORQUE ESCOLHE O CAMINHO MAIS “VANTAJOSO”
- Totalmente “Míope”
- Tomada de decisão com base nas informações disponíveis nas iterações
 - Não mede consequências

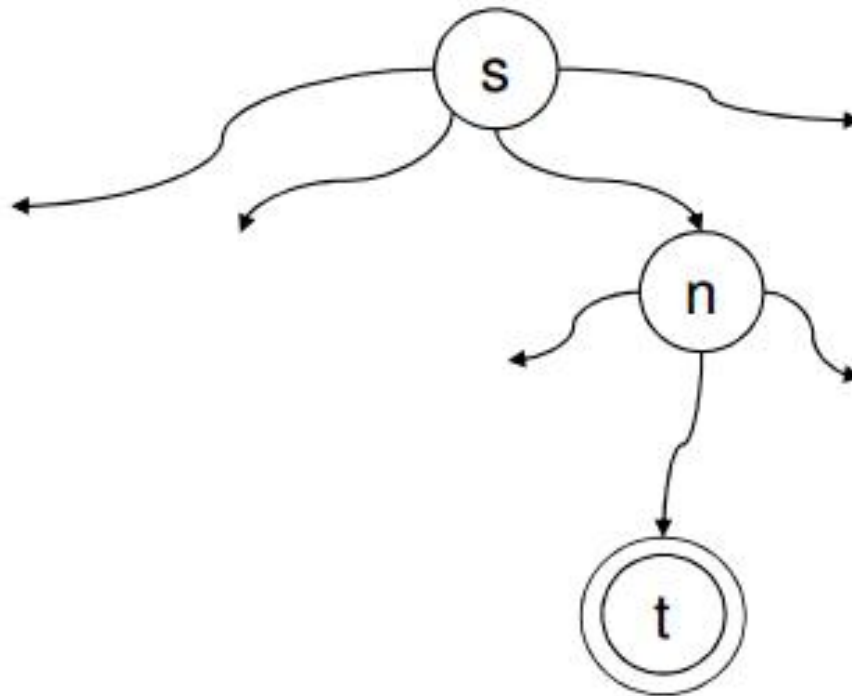
Visão humana

- Como fazer para dar troco de 36 usando apenas moedas de $\{1,5,10,20,40\}$?
- Começar sempre pela moeda com maior valor (menor que o valor a ser entregue)

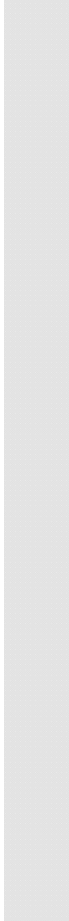
Análise em prolog...

- Assumindo o custo envolvido em cada arco:
 - $S(X,Y,C)$: movimento de X para Y usando custo C
- Seja s um nó inicial e t um nó final
 - Estimador heurístico f , tal que para cada nó n no espaço, $f(n)$ é a estimativa do custo do caminho mais barato de s até t , via n

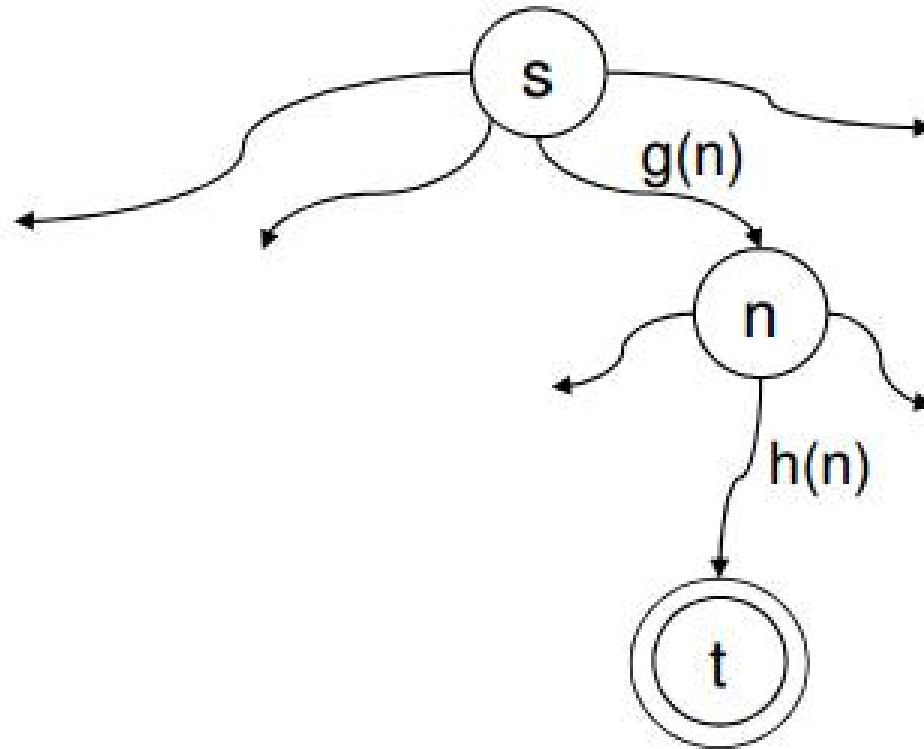
Análise em
prolog...




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

- $g(n)$ é uma estimativa do custo do caminho ótimo de s até n
 - $h(n)$ é uma estimativa do custo do caminho ótimo de n até t
- 

Graficamente..



Problema???

- $H(n)$
 - O caminho entre **n** e **t** não é conhecido
 - Então: $h(n)$ é uma heurística.
 - Não há um método universal para determinar **h**

Exercício

- Desenvolver um algoritmo que dado uma árvore, a mesma atinja um resultado usando ***greedy search***

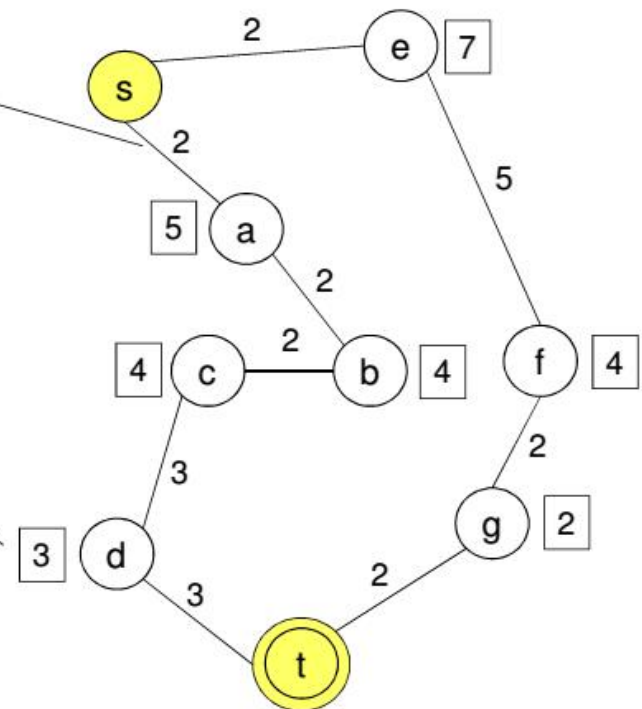
A*

- Conjunto de sub-processos para explorar todas as alternativas
- Sub-árvores têm sub-árvores que são exploradas por subprocessos dos subprocessos
 - Recursividade
- Dentre todos os processos apenas um encontra-se ativo a cada momento.
- Manter em funcionamento aquele que lida com a alternativa atual mais promissora
- Os outros processos aguardam, até que a estimativa atual se altere e alguma outra alternativa se torne mais promissora
- Por fim processo de comutação entre os processos

A*

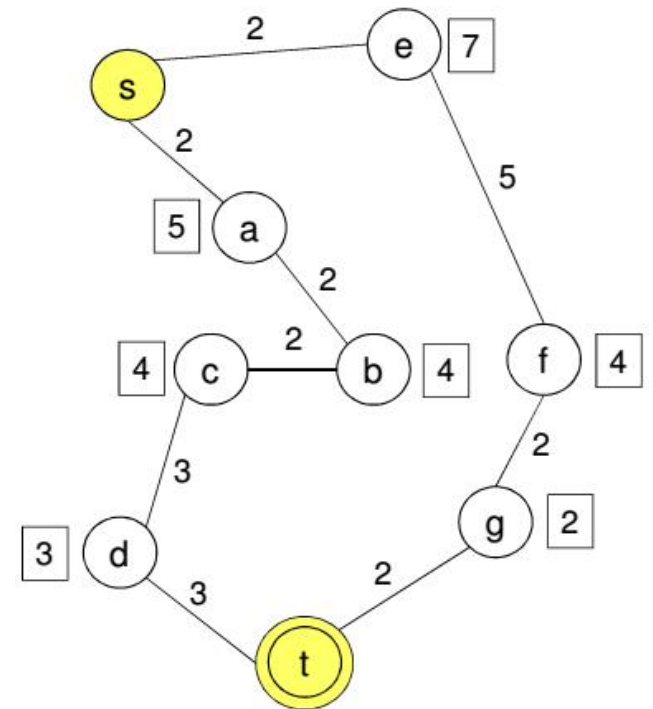
Distância entre duas
cidades através de um
caminho (rodovia)

Distância entre a
cidade em questão e a
cidade destino (t) em
linha reta



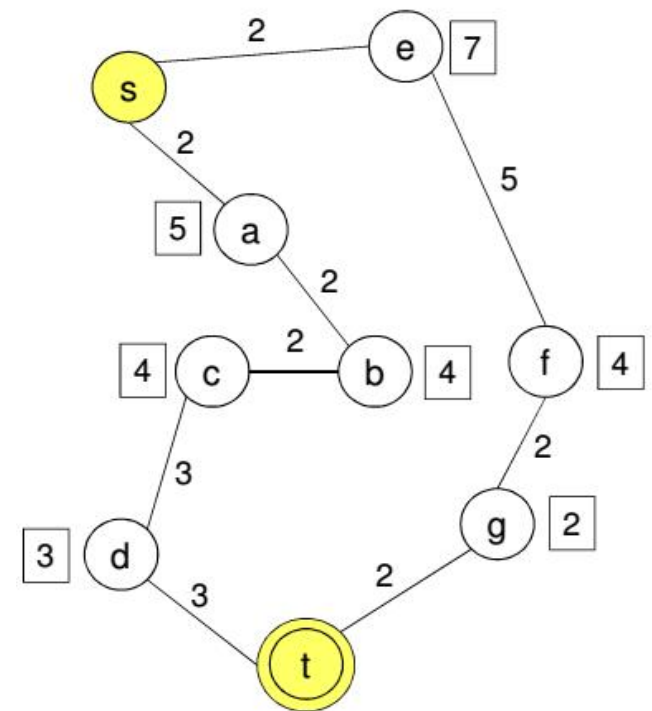
A*

- Dado um mapa, o objetivo é encontrar o caminho mais curto entre a cidade inicial **s** e a cidade destino
- Para estimar o custo do caminho restante da cidade **X** até a cidade **t** utilizaremos a distância em linha reta denotada por **dist(X,t)**
- $f(X) = g(X) + h(X) =$
 $= g(X) + \text{dist}(X,t)$



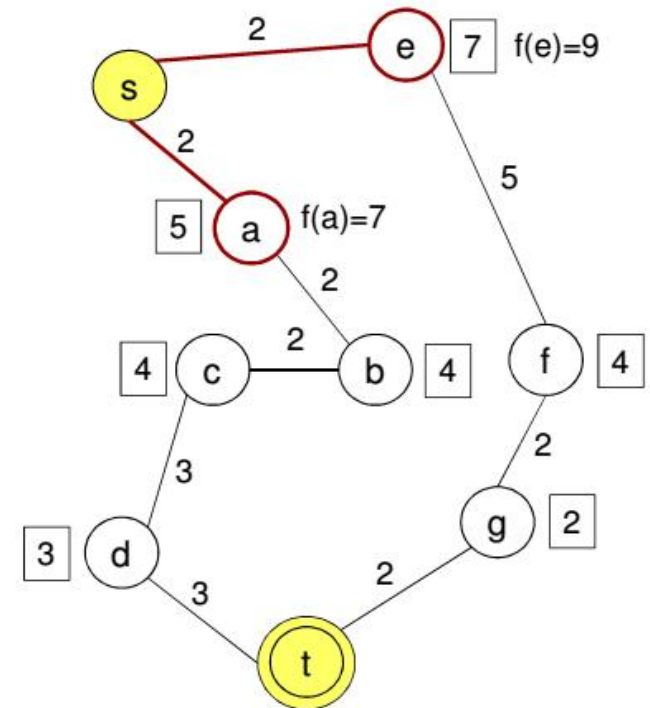
A*

- Neste exemplo, podemos imaginar a busca best-first consistindo em dois processos, cada um explorando um dos caminhos alternativos
- Processo 1 explora o caminho via **a**
- Processo 2 explora o caminho via **e**



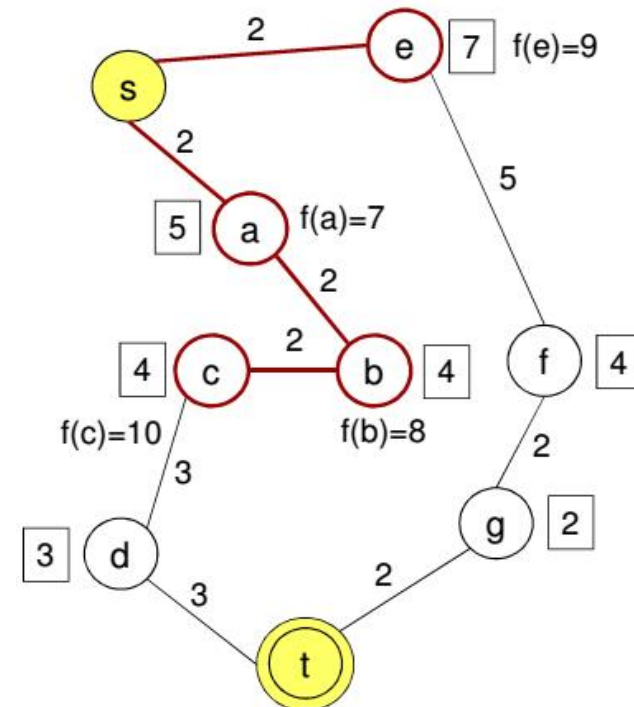
A*

- $f(a) = g(a) + \text{dist}(a, t) = 2 + 5 = 7$
- $f(e) = g(e) + \text{dist}(e, t) = 2 + 7 = 9$
- Como o valor-f de **a** é menor do que de **e**, o processo 1 (busca via **a**) permanece ativo enquanto o processo 2 (busca via **e**) fica em estado de espera



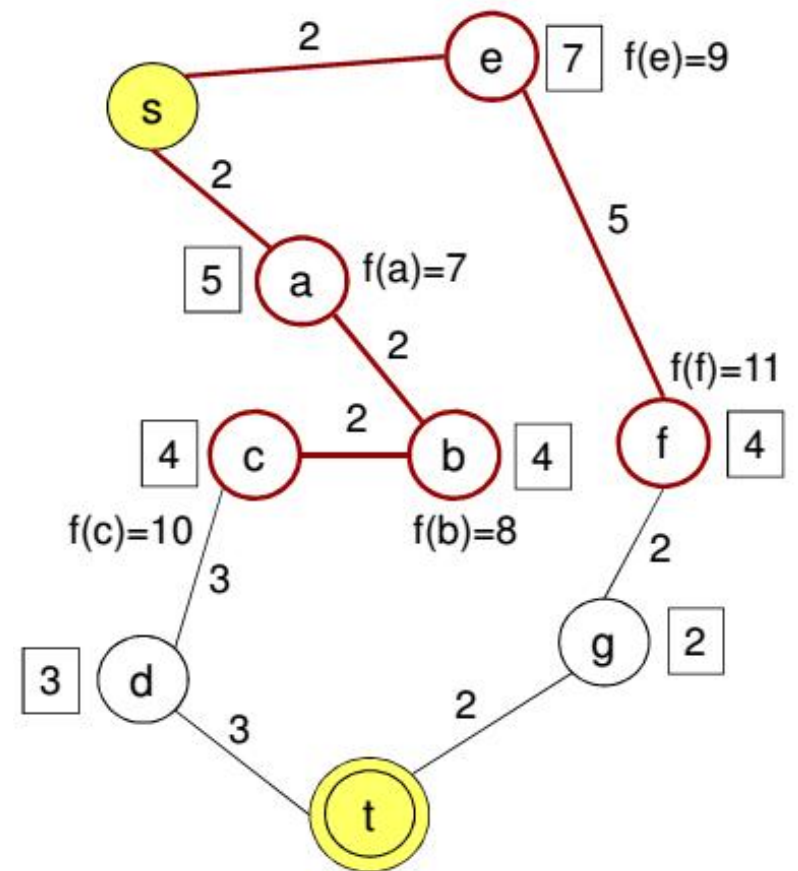
A*

- $f(a) = g(a) + \text{dist}(a, t) = 2 + 5 = 7$
- $f(e) = g(e) + \text{dist}(e, t) = 2 + 7 = 9$
- Como o valor- f de **a** é menor do que de **e**, o processo 1 (busca via **a**) permanece ativo enquanto o processo 2 (busca via **e**) fica em estado de espera
- $f(b) = g(b) + \text{dist}(b, t) = 4 + 4 = 8$
- $f(c) = g(c) + \text{dist}(c, t) = 6 + 4 = 10$
- Como $f(e) < f(c)$
 - O processo 2 prossegue para a cidade **f**



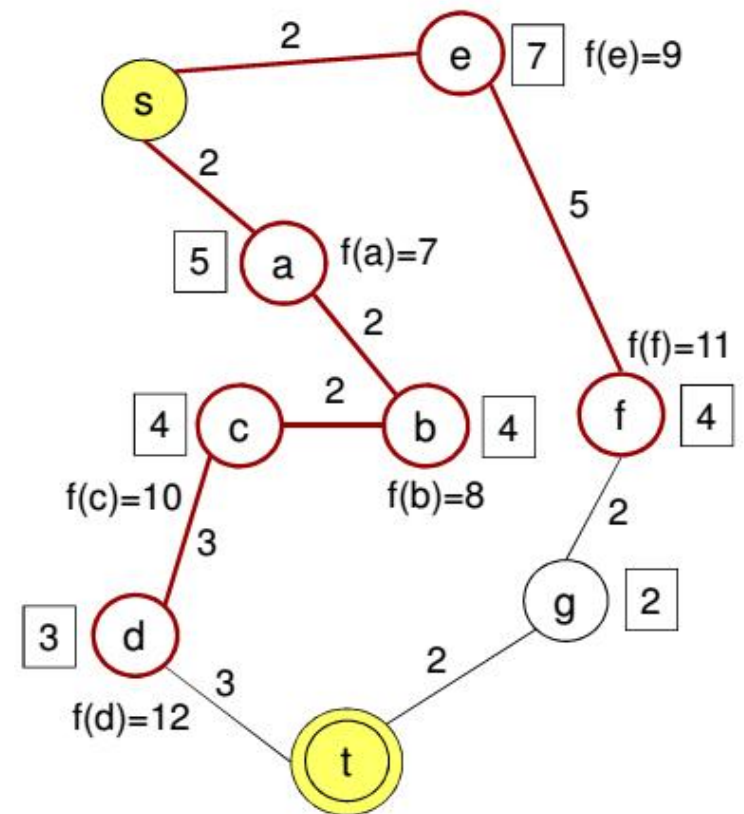
A*

- $f(f) = g(f) + \text{dist}(f, t) = 7 + 4 = 11$
- Como $f(f) > f(c)$ agora o processo 2 espera e o processo 1 prossegue



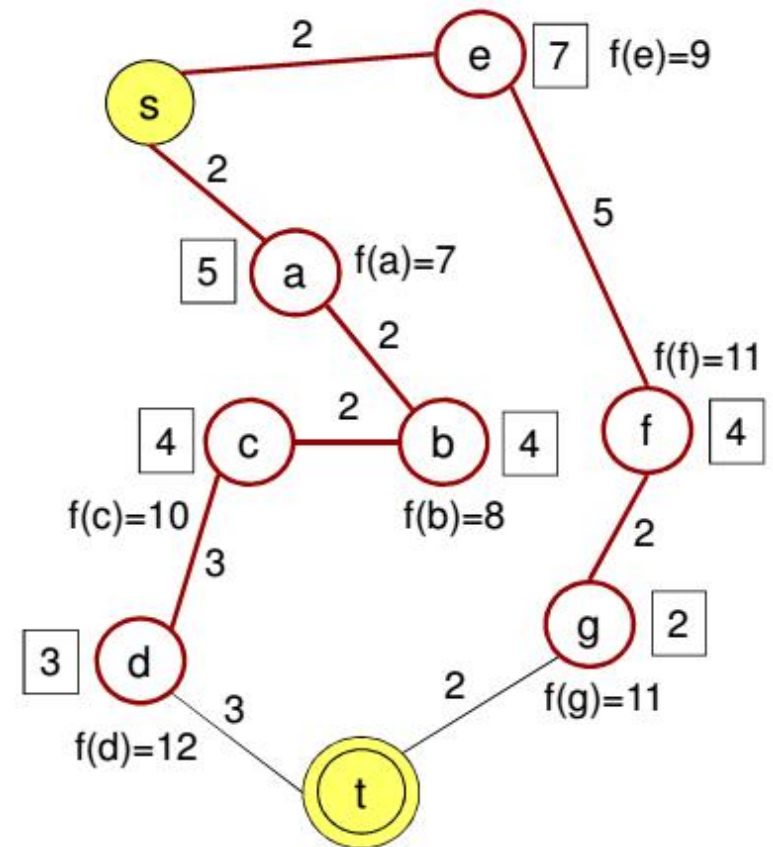
A*

- $f(f) = g(f) + \text{dist}(f, t) = 7 + 4 = 11$
- Como $f(f) > f(c)$ agora o processo 2 espera e o processo 1 prossegue
- $f(d) = g(d) + \text{dist}(d, t) = 9 + 3 = 12$
- Como $f(d) > f(f)$ o processo 2 reinicia



A*

- $f(f) = g(f) + \text{dist}(f, t) = 7 + 4 = 11$
- Como $f(f) > f(c)$ agora o processo 2 espera e o processo 1 prossegue
- $f(d) = g(d) + \text{dist}(d, t) = 9 + 3 = 12$
- Como $f(d) > f(f)$ o processo 2 reinicia chegando até o destino t
- $f(g) = g(g) + \text{dist}(g, t) = 9 + 2 = 11$



A*

- $f(f) = g(f) + \text{dist}(f, t) = 7 + 4 = 11$
- Como $f(f) > f(c)$ agora o processo 2 espera e o processo 1 prossegue
- $f(d) = g(d) + \text{dist}(d, t) = 9 + 3 = 12$
- Como $f(d) > f(f)$ o processo 2 reinicia chegando até o destino t
- $f(g) = g(g) + \text{dist}(g, t) = 9 + 2 = 11$
- $f(t) = g(t) + \text{dist}(t, t) = 11 + 0 = 11$

