

# (IIA) Prática

From WikiNote

## Contents

- 1 Ficha 6: Pesquisa no espaço de estados
  - 1.1 Exercício 1
    - 1.1.1 Pesquisa em profundidade
    - 1.1.2 Pesquisa em largura
    - 1.1.3 Heurística é admissível?
    - 1.1.4 Pesquisa Uniforme
    - 1.1.5 Pesquisa Sôfrega
    - 1.1.6 Pesquisa A\*
  - 1.2 Exercício 4
- 2 Ficha 7 - Pesquisa Local
  - 2.1 Algoritmo Trepa-Colinas
    - 2.1.1 Nota relativamente à vizinhança
  - 2.2 Algoritmo Recristalização Simulada

## Ficha 6: Pesquisa no espaço de estados

### Heurística

- É um valor numérico, baseado no custo dos operadores e basicamente diz "Para cada estado que identificamos no problema, qual é a estimativa de chegar desse estado até ao estado final". E para a heurística ser admissível, essa heurística não pode ultrapassar o custo real.

g

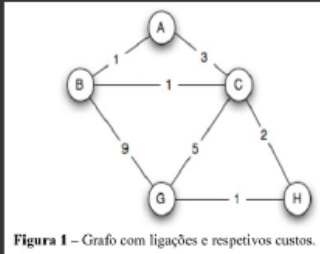
- custo real, valor que estão nas arestas

**Se nos derem 2 heurísticas e pedirem para escolhermos a melhor, temos de ver logo se ambas são admissíveis. Se não for, excluimos a que não é. Se forem ambas admissíveis, o fator de decisão deve ser aquela que mais se aproxima do custo real**

### Exercício 1

#### Pesquisa em profundidade

$h(A)=5, h(B)=4, h(C)=1$   
 $h(G)=0, h(H)=1$



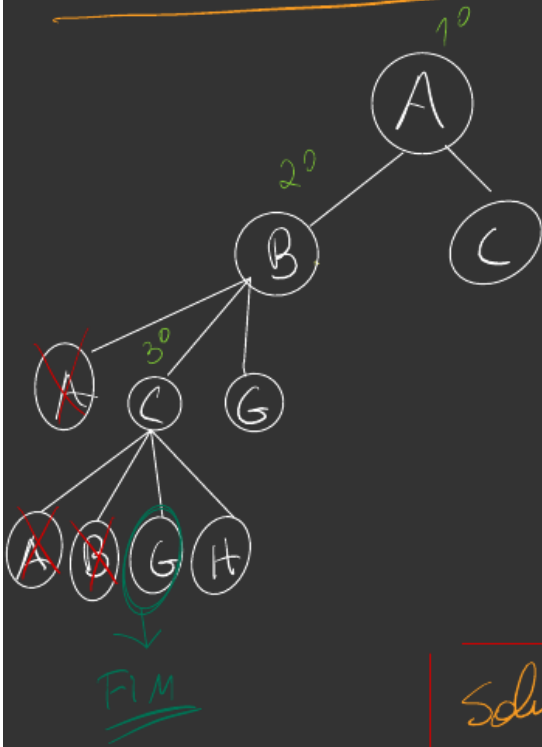
## Pesquisa em Profundidade

Próximo nó = exp?

Nesta pesquisa, é  
sempre o nó q-é este  
mais a esq. e mais  
abaixo.

Nós rejeitados?

Eliminar nós  
já expandidos



Solução: A - B - C - G

Custo: 7

Nós exp: 3 (A, B, C)

## Pesquisa em largura

$$h(A)=5, h(B)=4, h(C)=1$$

$$h(G)=0, h(H)=1$$

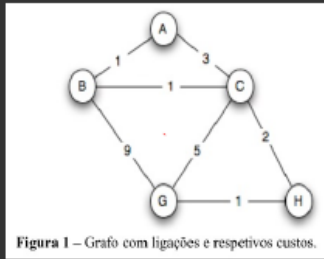
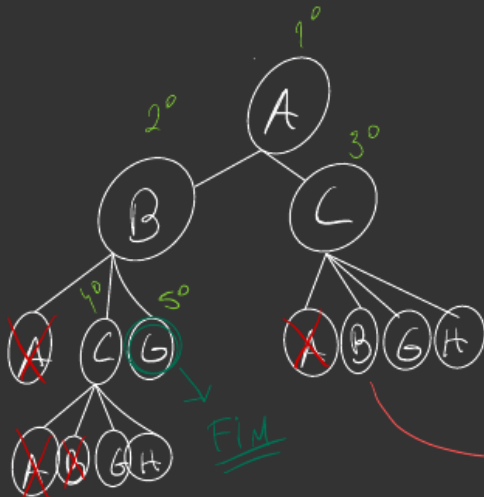


Figura 1 - Grafo com ligações e respetivos custos.

## Pesquisa em Largura



Solução: A - B - G

Custo: 10

Nós exp: 4 (A, B, C, G)

Próximo nó a exp?

É o nó que está mais à esquerda no mesmo nível e que ainda não foi expandido

Expandimos o B, anulamos repetições e depois expandimos o 1º C porque tem de se expandir sempre ao mesmo nível

→ Não se elimina o B porque o B foi expandido no ramo da esq. e não é direto

Heurística é admissível?

$f = \text{menor } f \Rightarrow \text{próximo no}$

Uniforme  $\rightarrow f = g$  (acumulado)  $\rightarrow$  custo real

Sôfrega  $\rightarrow f = h \rightarrow$  heurística

$A^*$   $\rightarrow f = g$  (acumulado) +  $h$

$\hookrightarrow$  Garante solução ótima se a heurística for admissível

Heurística é Admissível ?  $h \leq \text{custo real}$

$h(A) = 5 \leq 5$  custo real  $A \rightarrow G$  ✓

$h(B) = 4 \leq 4$  custo real  $B \rightarrow G$  ✓

$h(C) = 1 \leq 3$  custo real  $C \rightarrow G$  ✓

$h(H) = 1 \leq 1$  custo real  $H \rightarrow G$  ✓

$h(G) = 0$  ✓  $\rightarrow$  A heurística do estado final tem de ser sempre 0

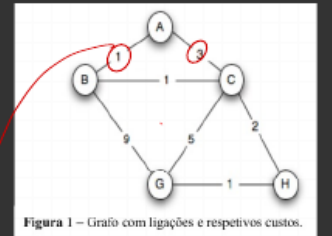
Como é admissível em todos os estados, é uma heurística admissível

## Pesquisa Uniforme

Pesquisa Uniforme  $f = \epsilon_g$

$$h(A)=5, h(B)=4, h(C)=1$$

$$h(G)=\emptyset, h(H)=1$$

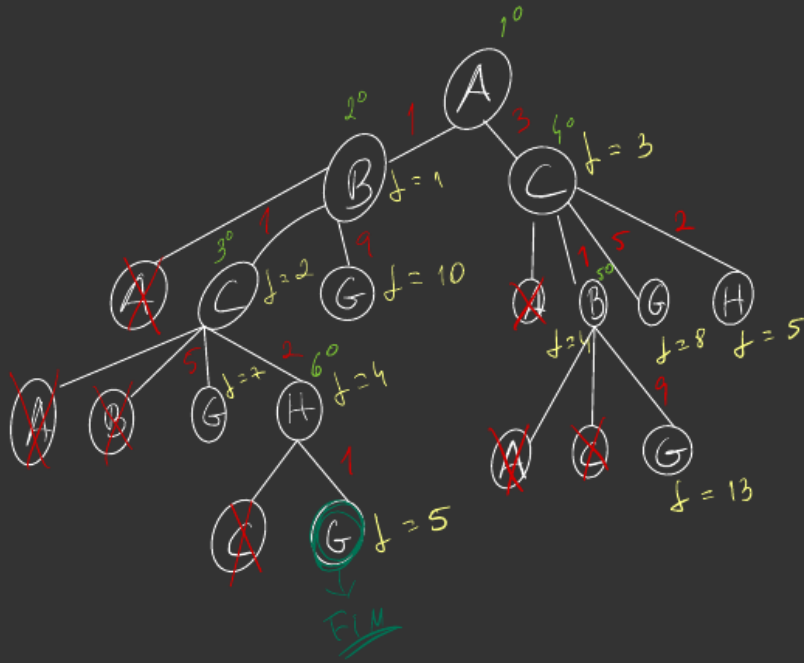


Custo Real (g)

Próximo nó = exp?

É o que tem o menor valor do  $f$

Se exploramos nós na esquerda, não podemos analisar na direita, temos na mesma de os explorados



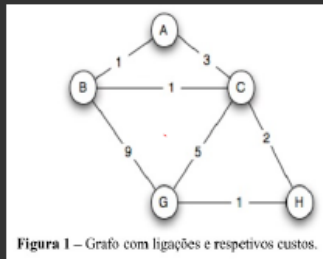
Solução: A-B-C-H-G

Custo: 5

Nós exp: 6 (A, B, C, C, B, H)

**Pesquisa Sôfrega**

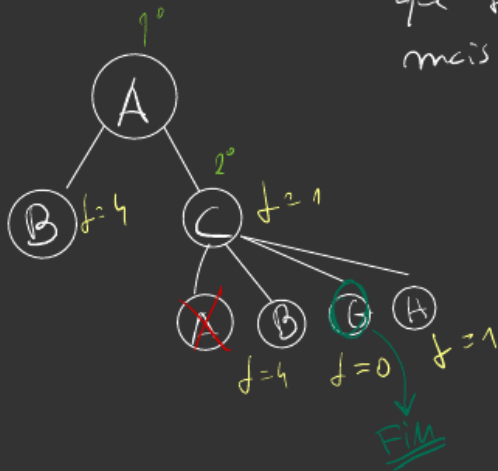
$h(A)=5, h(B)=4, h(C)=1$   
 $h(G)=0, h(H)=1$



Pesquisa sôfrega  $f = h$

Próximo nó é exp?

Escolher o nó  
que tem a heurística  
mais baixa



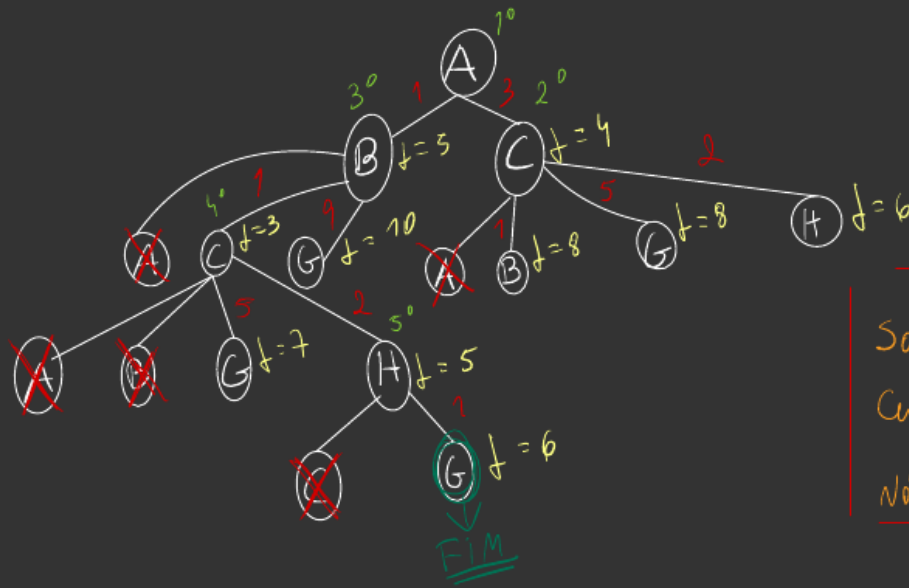
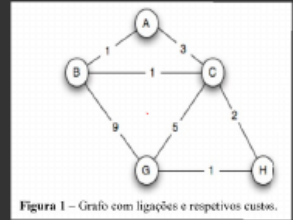
A sôfrega é válida no  
instanto não lige ao  
custo e quando aparece  
o objetivo no caminho  
ele vai logo para lá

Solução: A - C - G  
 Custo: 8  
 Nós exp: 2 (A, C)

**Pesquisa A\***

Pesquisa A\*  $f = E_g + h$

$h(A)=5, h(B)=4, h(C)=1$   
 $h(G)=\emptyset, h(H)=1$



Solução: A - B - C - H - G

Custo: 5

Nós Ex: 5 (A, C, B, C, H)

Esta pesquisa dá-nos a melhor solução possível desde que a heurística seja admissível

## Exercicio 4

$A^* = f = g + h$

Para o exemplo do A e do C

$f = g + h \Rightarrow 5 = g + 4 \Rightarrow g = 1$

$f = g + h \Rightarrow 5 = g + 1 \Rightarrow g = 4$

Figura 4 – Grafo com ligações e respectivas heurísticas.

Utilizou-se o A\* para encontrar o caminho mais curto entre S e G. Foi adotada a heurística admissível h. Na figura 4 pode confirmar os valores atribuídos por h. Em cada iteração de execução do algoritmo surgiu a lista ordenada dos caminhos parciais que estão a ser expandidos. A ordenação foi feita pelo valor de f do último nó do caminho parcial. Os passos de execução do algoritmo podem ser consultados na lista seguinte:

- $\{(S), f(S)=7\}$ ;
- $\{(S,A), f(A)=5\}, \{(S,C), f(C)=5\}$ ;
- $\{(S,A,B), f(B)=2\}, \{(S,C), f(C)=5\}$ ;
- $\{(S,A,B,C), f(C)=4\}, \{(S,C), f(C)=5\}, \{(S,A,B,G), f(G)=10\}$ ;
- $\{(S,C), f(C)=5\}, \{(S,A,B,C,G), f(G)=7\}, \{(S,A,B,G), f(G)=10\}$ ;
- $\{(S,A,B,C,G), f(G)=7\}, \{(S,C,G), f(G)=8\}, \{(S,C,A), f(A)=9\}, \{(S,A,B,G), f(G)=10\}$ ;
- Pesquisa termina.

Determine o custo de cada ligação entre dois nós do grafo, considerando que:

- Existe um mecanismo de deteção de ciclos nos caminhos gerados;
- Em caso de empate, os nós são expandidos por ordem alfabética.

Atenção que é preciso somar sempre o "g" acumulado

$f = g + h$   
 $10 = g + 0$   
 $10 = g + 2 \rightarrow$  acumulado desde cima  
 $g = 8$

## Ficha 7 - Pesquisa Local

A filosofia destes algoritmos muda uma vez que o espaço de procura é muito grande.

Então estes algoritmos trabalham com o "Espaço das soluções", ou seja, não fazem uma pesquisa exaustiva do espaço de procura mas tentam ir melhorando uma solução para o problema.

- **Representação:**
  - Como representar uma solução para o problema?
- **Função de avaliação:**
  - Atribui a qualidade à solução
  - Identificar se o problema é de minimização ou maximização
  - Há a possibilidade de surgirem soluções inválidas?
    - Penalizar
    - Reparar
- **Vizinhança:**
  - Cria uma nova solução a partir da solução atual

Se a for um problema de minimização o objetivo é encontrar valores sempre mais baixos, se for um problema de maximização é ir encontrando valores mais altos.

## Algoritmo Trepacolinhas



```

Trepacolinhas(solucao, k)
    custo = avalia(solucao)
    Repete k iteracoes
        nova_sol = gera_vizinho (solucao)
        novo_custo = avalia(nova_sol)
        se novo_custo > custo (maximização) então
            sol = nova_sol
            custo = novo_custo
        fim_se
    Fim_repete

```

Este algoritmo tem uma limitação, por exemplo quando existe um ótimo local.

O ponto de partida deste algoritmo é crucial para o resultado final que o mesmo tem.

### Nota relativamente à vizinhança

Normalmente com uma vizinhança maior iremos ter um resultado pior a uma vizinhança menor. Uma vizinhança que faz mais alterações normalmente produz piores resultados.

## Algoritmo Recristalização Simulada

- Ideia: fugir aos ótimos locais permitindo a aceitação de soluções piores de forma "controlada"

```

Recristalização_simulada(solucao, k)
    t = TMAX
    itera = 0
    custo = avalia(solucao)
    ENQUANTO t > TMIN
        Repete k vezes
            nova_sol = gera_vizinho (solucao)
            novo_custo = avalia(nova_sol)
            SE novo_custo > custo (maximização) ENTÃO
                sol = nova_sol
                custo = novo_custo
            SENÃO //aceita solução pior
                SE rand_01() <= exp((novo_custo-custo)/t) ENTÃO
                    sol = nova_sol
                    custo = novo_custo
            FIM_SE
        FIM_SE
    Fim_Repete
    itera = itera + 1
    t = t * fator_arrefecimento
    FIM_ENQUANTO

```

Para maximização!  
Se for minimização substituir por :  
(custo - novo\_custo)

Este algoritmo não tem um número de iterações fixas como o Trepacolinhas tem.

Existe é o TMAX (Temperatura Máxima), o arrefecimento e o TMIN (Temperatura Mínima).

O fator de descida é o arrefecimento que vai do TMAX para o TMIN.

Quanto mais próxima a TMIN estiver de 0 melhor, uma vez que assim o algoritmo aceita menos soluções más.

Retrieved from "http://zebisnaga.pt/wiki/index.php?title=(IIA)\_Prática&oldid=1124"