(IIA) Prática

From WikiNote

Contents

Ficha 6: Pesquisa no espaço de estados

Heuristica

- É um valor númerico, 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 heuristica ser admissivel, essa heuristica nao pode ultrapassar o custo real.

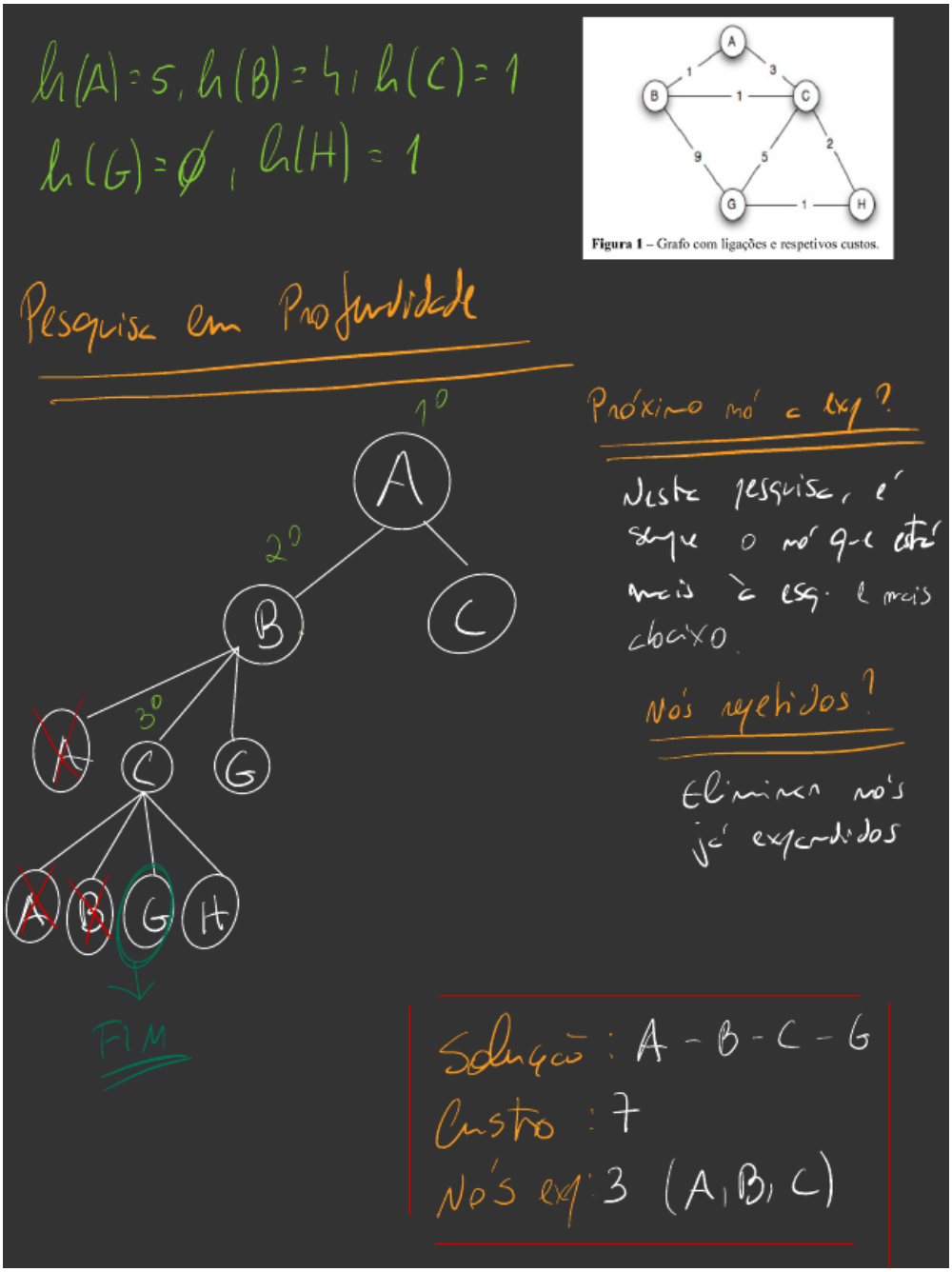
g

- custo real, valor que estao nas arestas

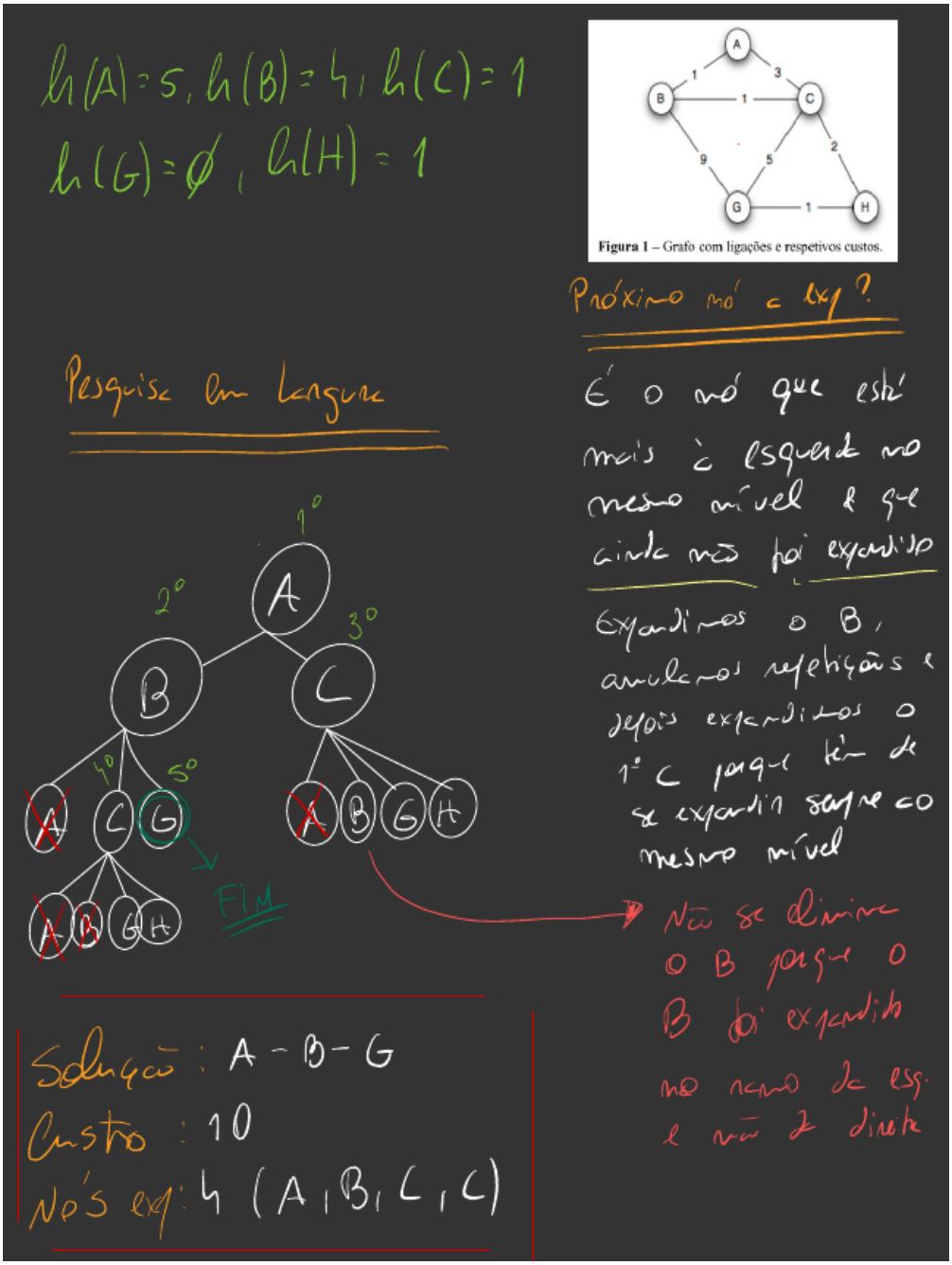
Se nos derem 2 heuristicas e pedirem para escolhermos a melhor, temos de ver logo se ambas sao admissiveis Se nao for excluimos a que nao é. Se forem ambas admissiveis, o fator de decisao deve ser aquela que mais se aproxima do custo real

Exercicio 1

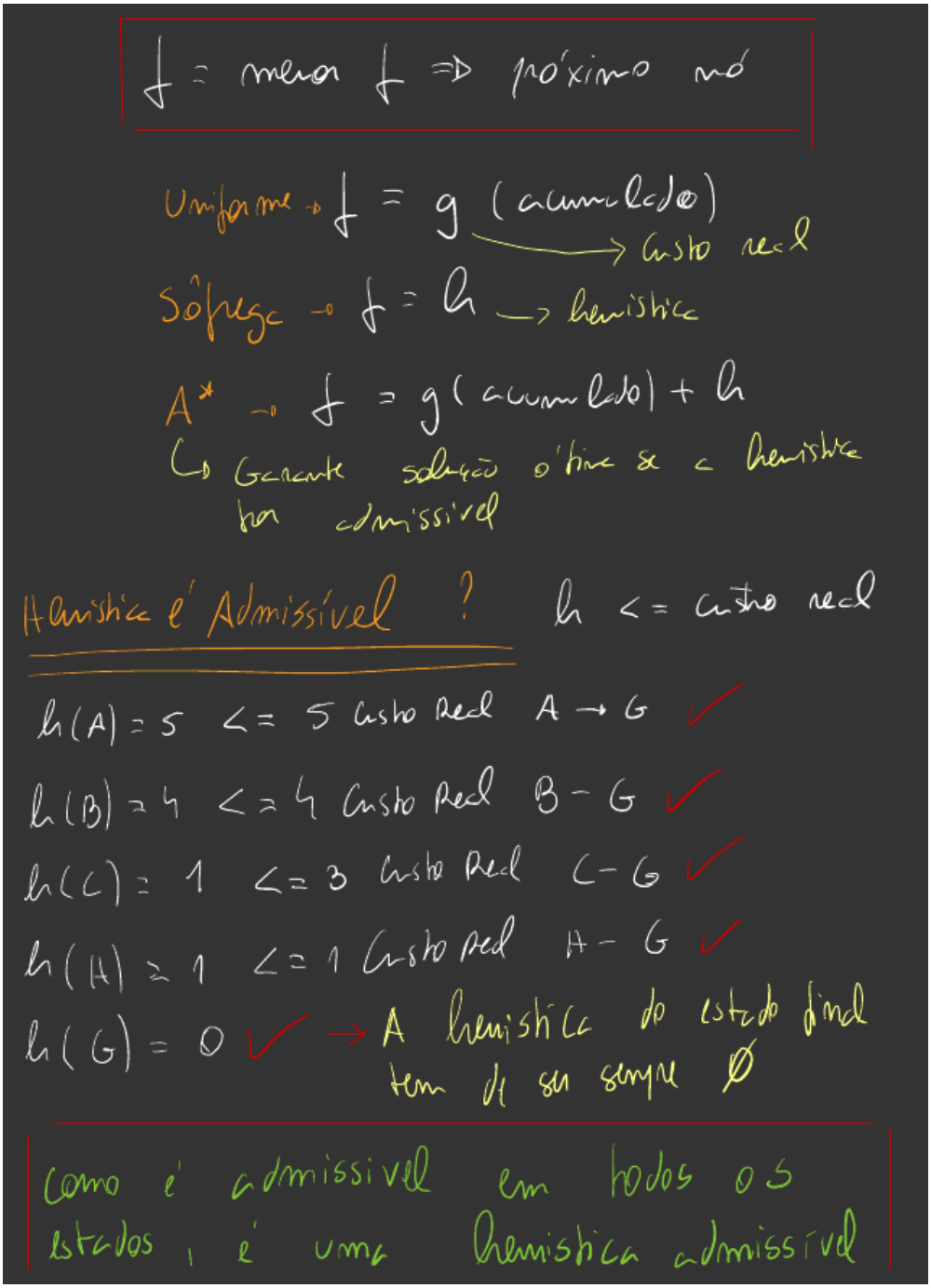
Pesquisa em profundidade



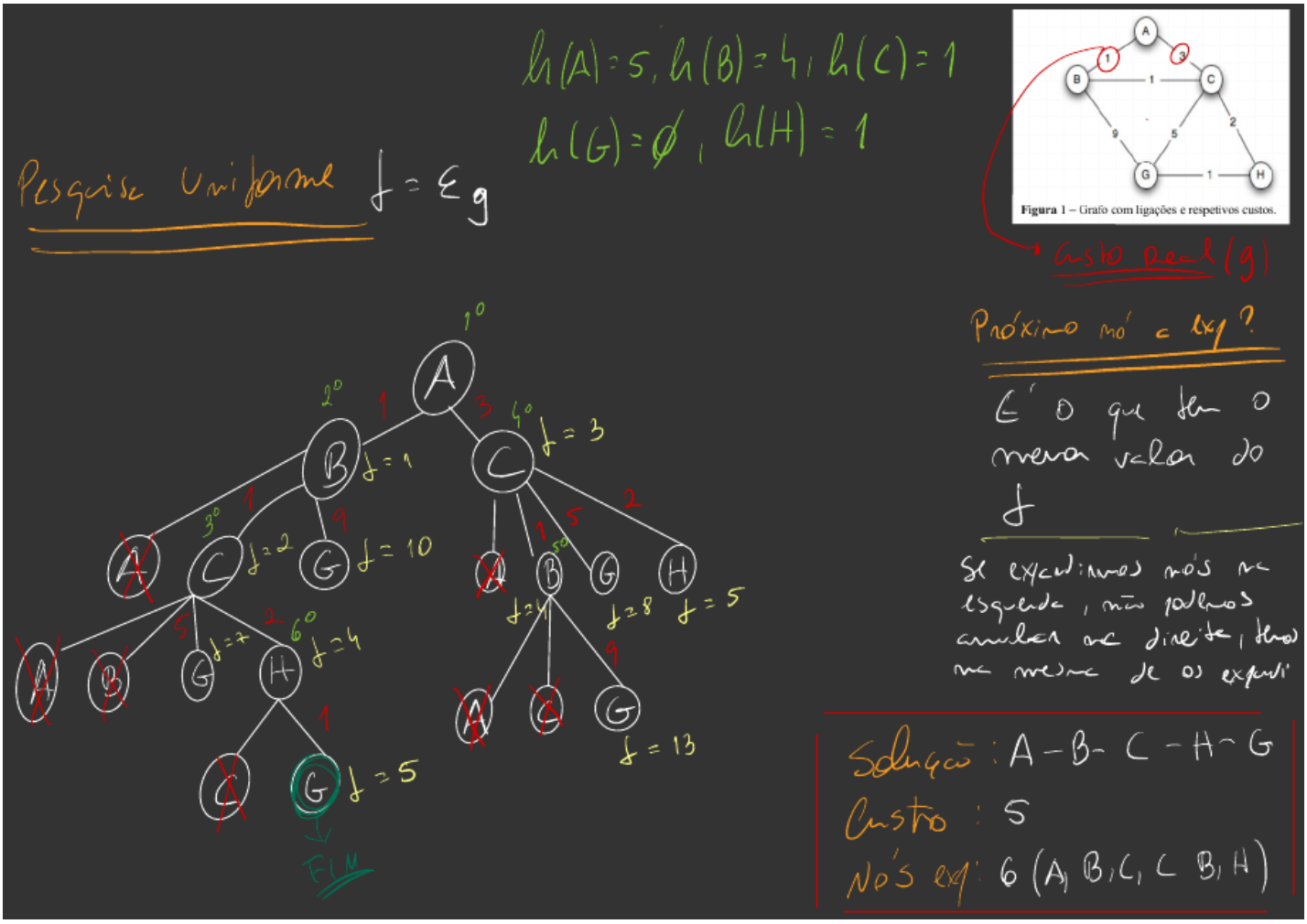
Pesquisa em largura



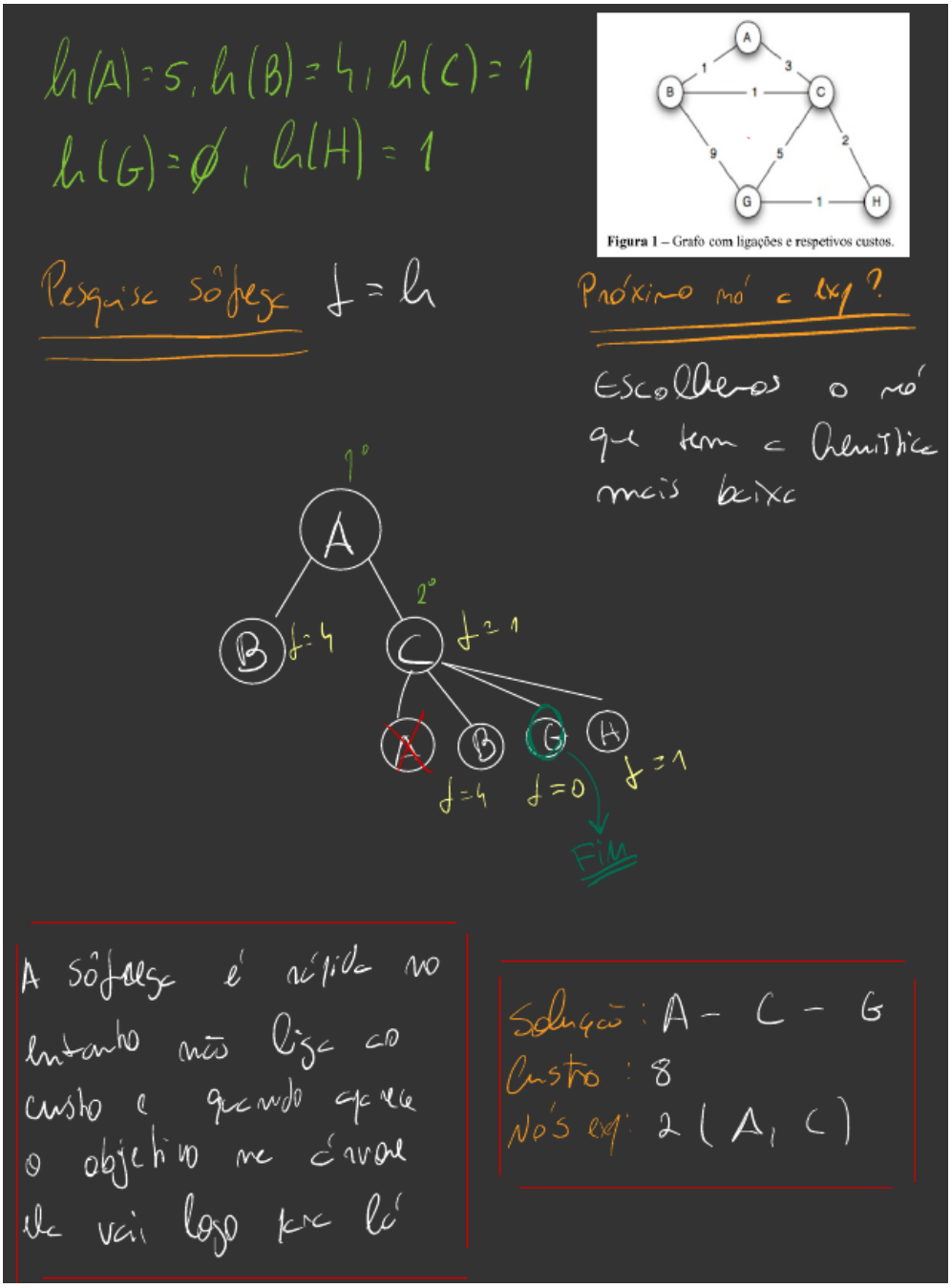
Heuristica é admissivel?



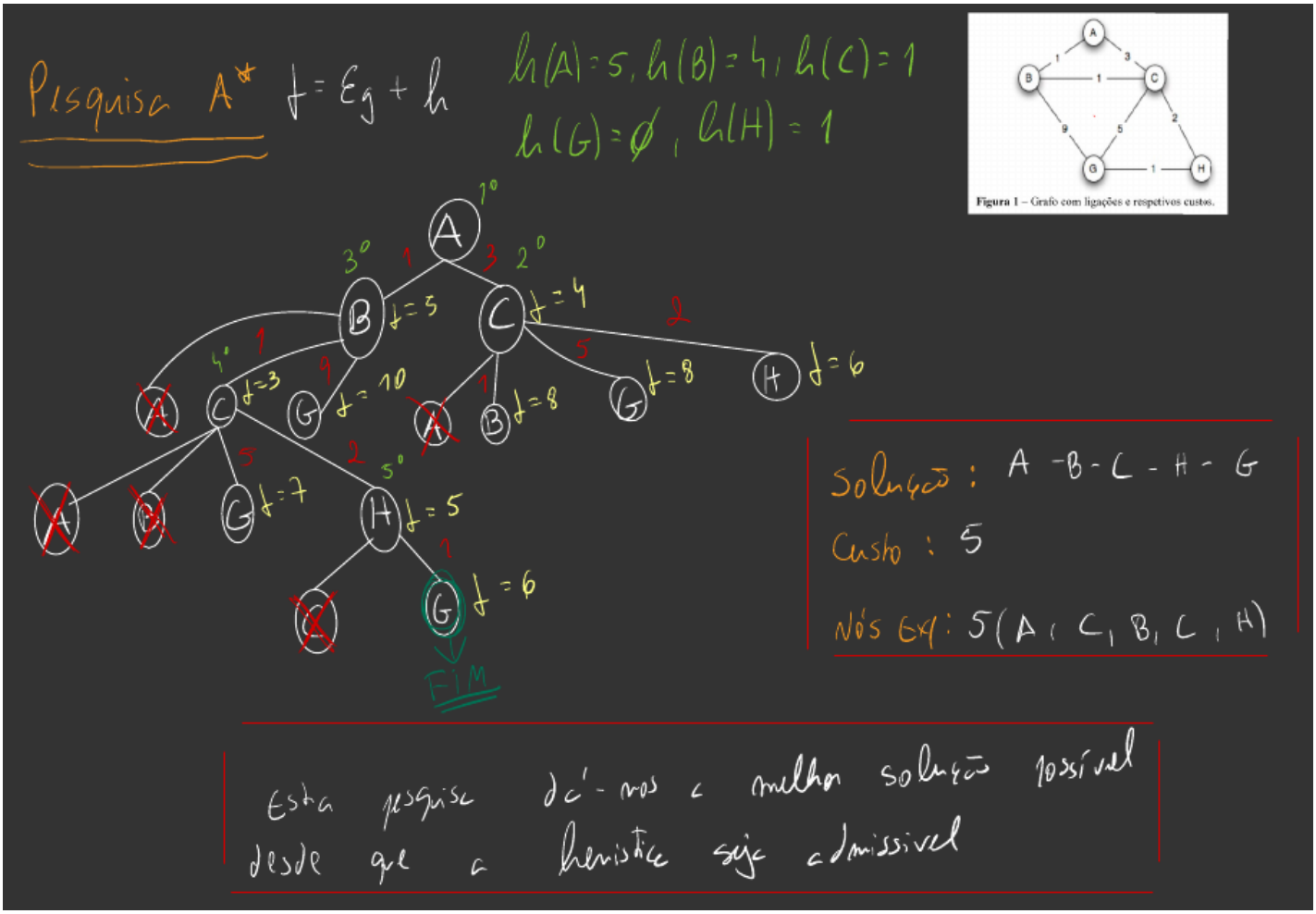
Pesquisa Uniforme



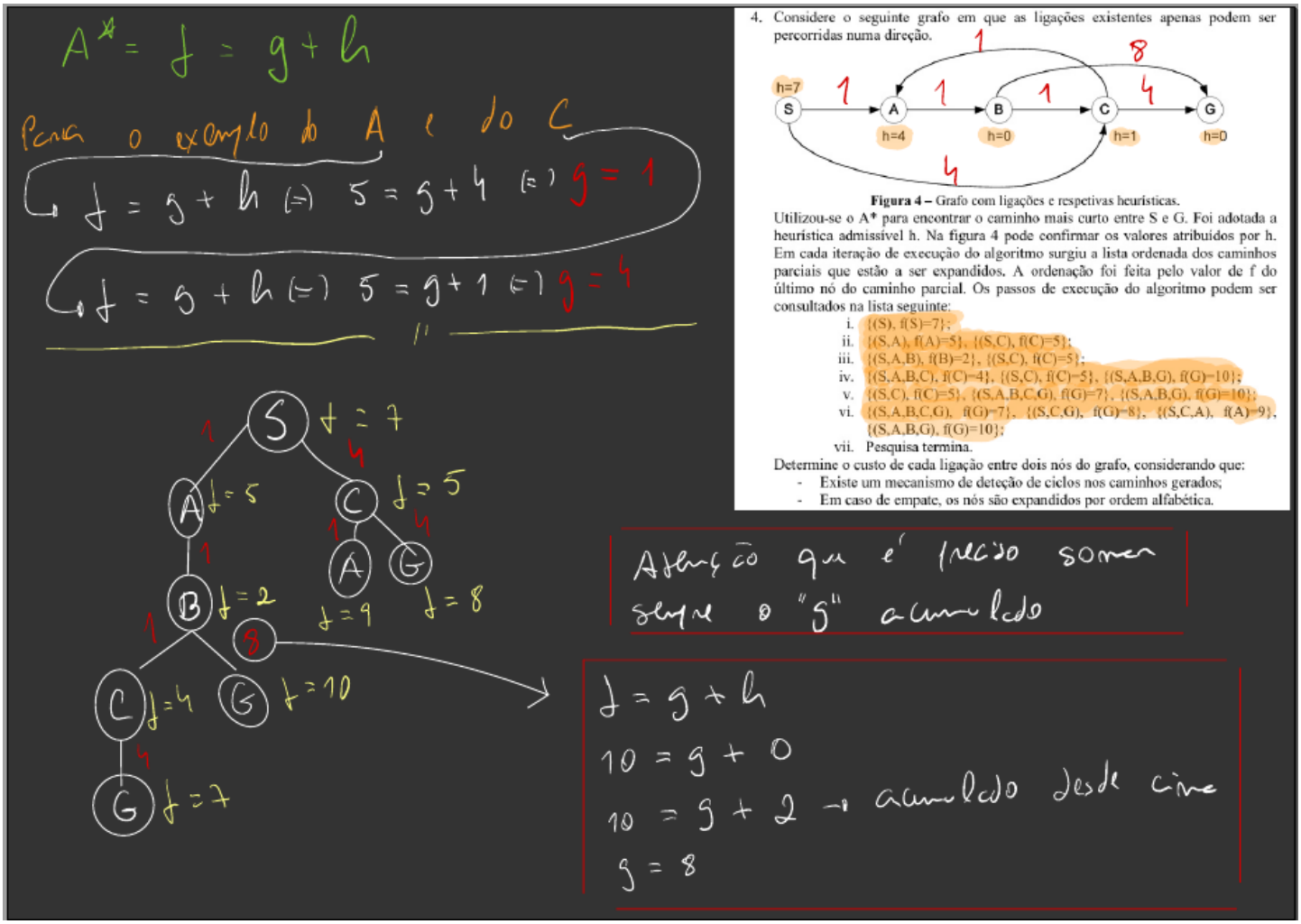
Pesquisa Sôfrega


h(A) = 5, h(B) = 4, h(C) = 1
h(G) = 0, h(H) = 1
Pesquisa sôfrega f = h
Próximo nó a ex?
Escolhemos o nó que tem a heurística mais baixa
A
B
C
G
H
f=4
f=1
f=4
f=0
f=1
Fim
A sôfrega é nítida no entanto não liga ao custo e quando apanha o objectivo ne é nada de vai logo lá
Solução: A - C - G
Custo: 8
Nós exp: 2 (A, C)

Pesquisa A*


Pesquisa A*
f = Eg + h
h(A) = 5, h(B) = 4, h(C) = 1
h(G) = ∅, h(H) = 1
Figura 1 – Grafo com ligações e respetivos custos.
Solução : A - B - C - H - G
Custo : 5
Nós Exp: 5(A, C, B, C, H)
FIM
Esta pesquisa dá-nos a melhor solução possível
desde que a heurística seja admissível

Exercicio 4



4. Considere o seguinte grafo em que as ligações existentes apenas podem ser percorridas numa direção.

Figura 4 – Grafo com ligações e respetivas 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:

i. {(S), f(S)=7};
ii. {(S,A), f(A)=5}, {(S,C), f(C)=5};
iii. {(S,A,B), f(B)=2}, {(S,C), f(C)=5};
iv. {(S,A,B,C), f(C)=4}, {(S,C), f(C)=5}, {(S,A,B,G), f(G)=10};
v. {(S,C), f(C)=5}, {(S,A,B,C,G), f(G)=7}, {(S,A,B,G), f(G)=10};
vi. {(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};
vii. 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.

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 Trepa-Colinas

```
Trepa-Colinas(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 vizinhaça 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 Trepa Colinas tem.

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

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.