

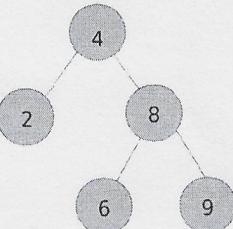
# Gabarito

Universidade Federal Rural de Pernambuco  
Departamento de Estatística e Informática  
Fundamentos de Problemas Computacionais II  
Prova: 3<sup>a</sup> V.A.  
Data: 30/03/2023  
Prof. Tiago A. E. Ferreira

**Obs.:** A prova é individual e todas as questões deverão conter seus respectivos desenvolvimentos para serem consideradas!

- 1) (2.0 Pontos) Dada a árvore ao lado, pergunta-se:

- (1.0 Ponto) Se for inserido o valor 5, haverá a necessidade de balanceamento? Se não, justifique a razão. Se sim, realize o balanceamento, demonstrando todos os passos.
- (1.0 Ponto) Por que é desejado se trabalhar com uma árvore balanceada?

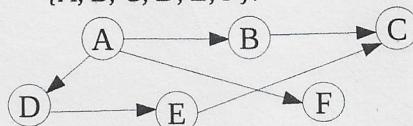


- 2) (3.0 Pontos) Seja um estacionamento E para caminhões de coleta de lixo. Deste estacionamento E é possível alcançar todas as residências de uma cidade. Sejam quatro residências (A, B, C e D). São conhecidas as distâncias do caminho de E para cada residência: 10 km para A; 14km para B; 8 km para C e 14 km para D. Também é sabido que da residência A existe um caminho para a residência B com 3 km de comprimento; de B existe um caminho para a residência D com 4km de comprimento; de C existe um caminho para a residência A com 4 km de comprimento, e existe um caminho para a residência D com 5 km de comprimento. Deseja-se definir a rota que um caminhão de coleta de lixo deve fazer para alcançar as quatro residências. Pede-se:

- (1.0 Ponto) Modele o problema com um grafo. Defina o grafo e faça um diagrama;
- (1.0 Ponto) Apresente um algoritmo computacionalmente eficiente (em pseudo-código) que calcule a rota procurada;
- (1.0 Ponto) Calcule a rota desejada utilizando o algoritmo proposto no item (b).

- 3) (2.0 Pontos) Dado grafo  $G=(V,E)$  mostrado abaixo. Observe que não são apresentados pesos nas arestas de G. Desta forma, é possível considerar que todos os pesos são iguais.

- (1.0 Ponto) É desejado calcular os caminhos mais curtos com fontes múltiplas para o grafo abaixo. Considerando apenas a matriz de pesos  $\mathbf{W}$ . Apresente um algoritmo que execute o que se deseja.
- (1.0 Ponto) Aplique o algoritmo apresentado no item (a) e mostre uma matriz  $\mathbf{D}$  com todos os pesos dos caminhos mais curtos com fonte múltiplas, de tal forma que  $d_{ij} = \delta(i,j) \forall i,j \in \{A, B, C, D, E, F\}$ .

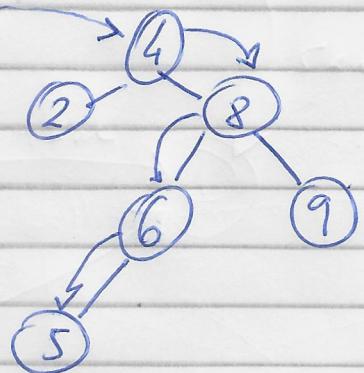


- 4) (3.0 Pontos) Seja o quebra-cabeça do Cubo Mágico. Este jogo consiste de um cubo onde há 9 peças por face, das quais 8 podem se movimentar sobre dois eixos ortogonais. Seja o estado inicial uma configuração aleatória qualquer das posições das peças e o estado final (solução buscada) a configuração onde todas as peças de uma face são da mesma cor. Pede-se:

- (1.0 Ponto) Esquematize diagramaticamente como um grafo poderia representar o espaço de estados deste problema. Descreva a modelagem do problema, ou seja, o que representa os vértices e as arestas para o dado problema.
- (1.0 Ponto) Defina um função heurística para a resolução deste problema. Verifique se a função proposta é admissível.
- (1.0 Ponto) É desejado a aplicação do algoritmo A\* para a solução deste problema. Dado que  $f(n) = g(n) + h(n)$ , onde  $h(n)$  é a função heurística e  $g(n)$  é a função de custo real executado, apresente uma possível expressão para  $f(n)$ ? Explique a expressão apresentada.

# Gabarito → 3º V.A. Fundamentos de Problemas Computacionais II

1) A) inserindo (5)

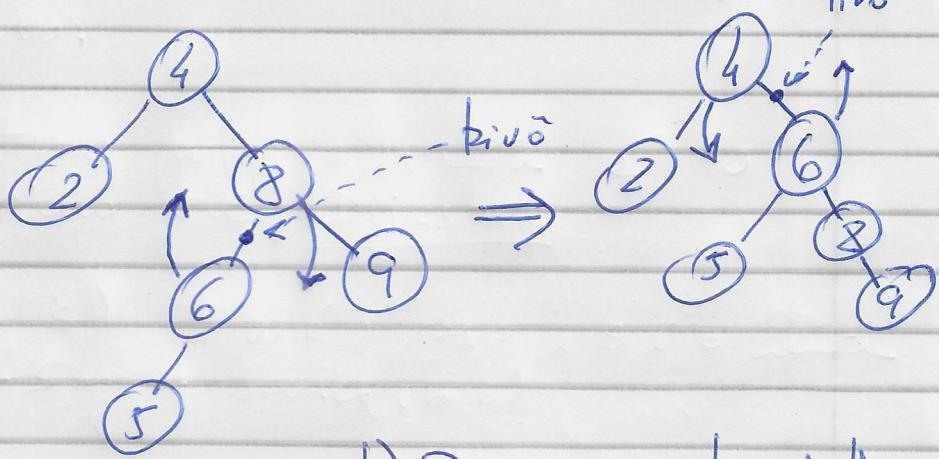


Com a inserção, calcula-se o fator de balanceamento.

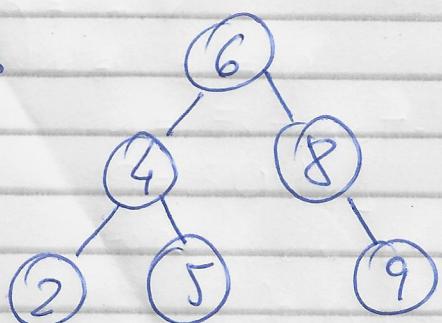
→ O nodo raiz (4) tem fator de balanceamento igual a 2 (ou -2), logo a árvore está desbalanceada.

Há a necessidade de uma rotação dupla à esquerda:

a)



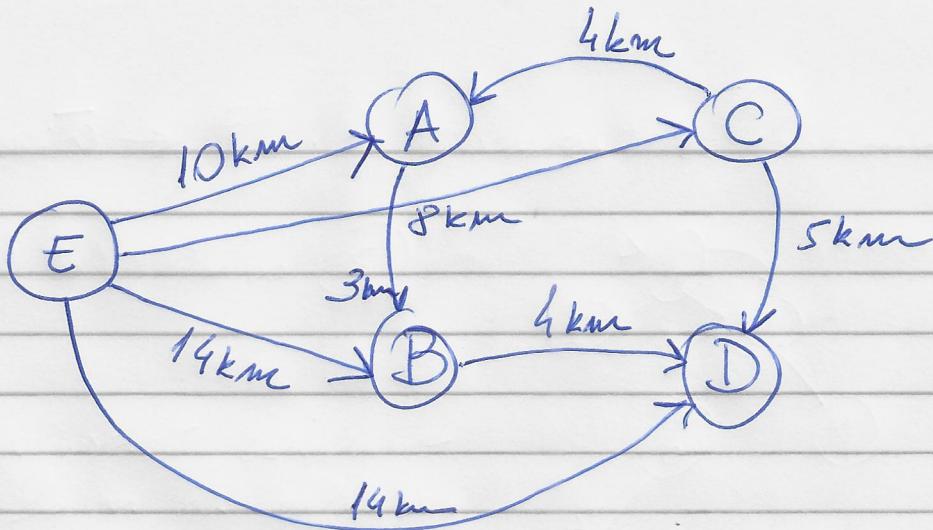
→



ÁRVORE平衡ada

b) Deseja-se trabalhar com uma árvore balanceada para se garantir uma profundidade sempre da  $O(\lg n)$ , onde  $n$  é o nº de nodos da árvore!

2) a)



$G = (V, E)$ ,  $V$  representa os locais  
 $E$  representa as coligações entre os locais, onde os pesos não são necessariamente distâncias.

b) Observe que o grafo  $G_1$  é dirigido e acíclico, não havendo arestas com pesos negativos.

↳ Algoritmo mais eficiente é o Algoritmo de Dijkstra

Dijkstra ( $G, w, s$ ) //  $w$  é o peso e  $s$  é a fonte  
 Initialize-Single-Source ( $G, s$ )

$$S = \emptyset$$

$$Q = G.V$$

while  $Q \neq \emptyset$

$$u = \text{Extract-Min}(Q)$$

$$S = S \cup \{u\}$$

for each  $v \in G.\text{Adj}[u]$

$$\text{Relax}(u, v, w)$$

and

## Initialize - Single-Source ( $G, s$ )

for each  $v \in G.V$

:  $v.d = \infty$

:  $v.\pi = \text{NIL}$

$s.d = 0$

2.

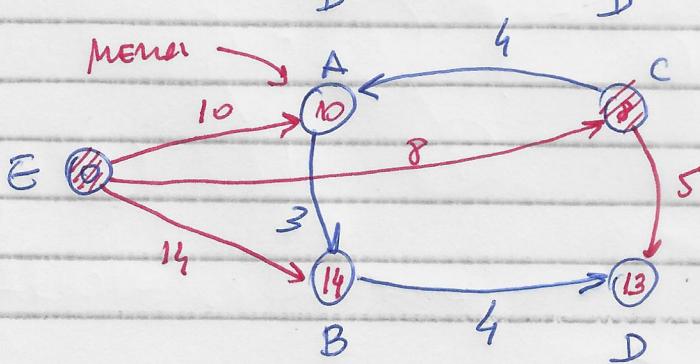
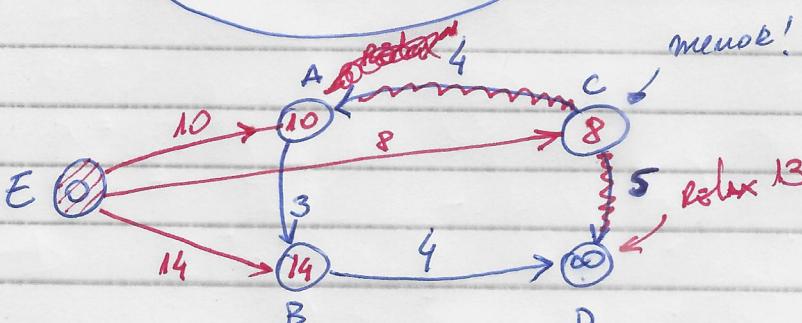
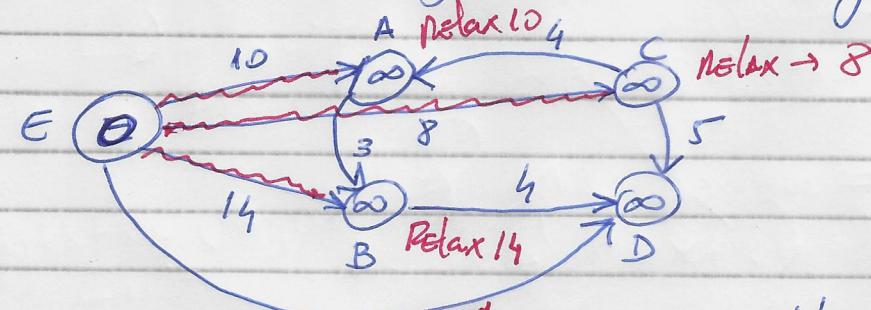
Relax ( $u, v, w$ )

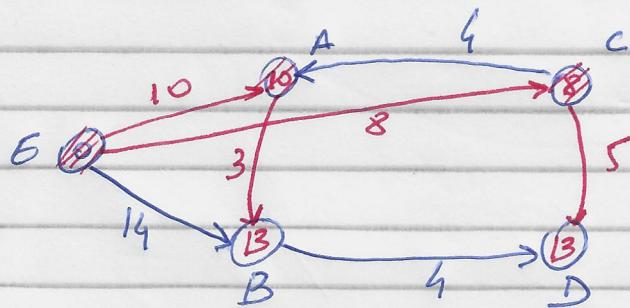
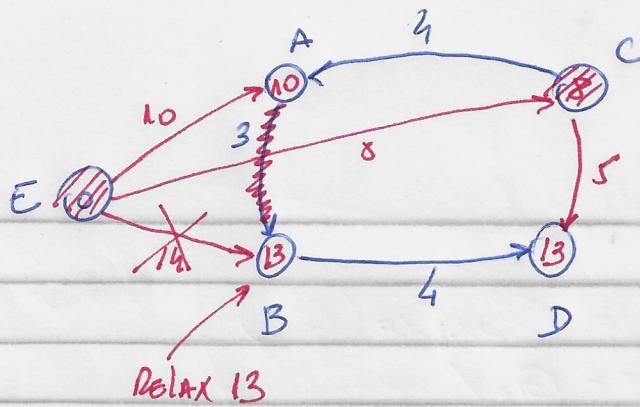
if  $v.d > u.d + w(u, v)$

$v.d = u.d + w(u, v)$

$v.\pi = u$

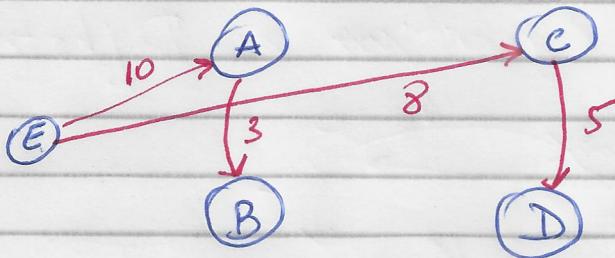
c) usando o algoritmo de Dijkstra:





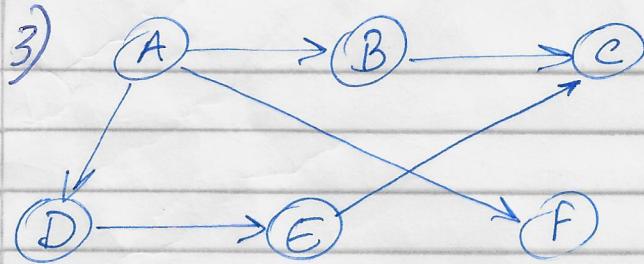
Tanto  $\textcircled{B}$  com  $\textcircled{D}$  são  
mínimos!  
Mas não é possível  
relaxar min's menor  
outro vértice.

Assim, a solução de Dijkstra é



Mas o caminho deve fazer duas travesias,  
e é necessário que o caminho faça  
apenas uma rota!

Assim, se for suposto que as rotas existem  
nos dois sentidos, uma vez que feiam  
dadas apenas as distâncias, um caminho  
deve ir  $(\textcircled{A} \leftarrow \textcircled{B})$  e depois  $(\textcircled{C} \leftarrow \textcircled{D})$



a) Se  $W$  a matriz de peso, dada por:

	A	B	C	D	E	F
A	0	1	$\infty$	1	$\infty$	1
B	$\infty$	0	1	$\infty$	$\infty$	$\infty$
C	$\infty$	$\infty$	0	$\infty$	$\infty$	$\infty$
D	$\infty$	$\infty$	$\infty$	0	1	$\infty$
E	$\infty$	$\infty$	1	$\infty$	0	$\infty$
F	$\infty$	$\infty$	2	$\infty$	$\infty$	0

Faster - All - Pairs - Shortest - Paths ( $W$ )

$n = W.$  rows

$L^{(n)} = W$

$m = 1$

while  $m < n-1$

let  $L^{(2m)}$  be a new  $n \times n$  Matrix

$L^{(2m)} = \text{Extend - Shortest - Paths } (L^{(m)}, L^{(m)})$

$m = 2m$

return  $L^{(m)}$

onde :

# Extend - shortest - Paths (L, w)

$n = L$ , rows

Let  $L' = (l'_{ij})$  be a new  $n \times n$  matrix

For  $i = 1$  to  $n$

For  $j = 1$  to  $n$

$$l'_{ij} = \infty$$

For  $k = 1$  to  $n$

$$l'_{ij} = \min(l'_{ij}, l_{ij} + w_{kj})$$

Return  $L'$

b) Aplicando o Algoritmo proposto.

$$D^{(1)} = W = \begin{pmatrix} & j \rightarrow \\ i \downarrow & 0 & 1 & \infty & 1 & \infty & 1 \\ 0 & \infty & 0 & 1 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 1 & \infty & \infty \\ \infty & \infty & 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 0 \end{pmatrix}$$

Caminhos com  
até 1 aresta

$$D^{(2)} = \begin{pmatrix} 0 & 1 & 2 & 1 & 2 & 1 \\ \infty & 0 & 1 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 1 & \infty \\ \infty & \infty & 1 & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

Caminhos com  
até 2 arestas.

$$D^{(3)} = \begin{pmatrix} 0 & 1 & 2 & 1 & 2 & 1 \\ \infty & 0 & 1 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 1 & \infty \\ \infty & \infty & 1 & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

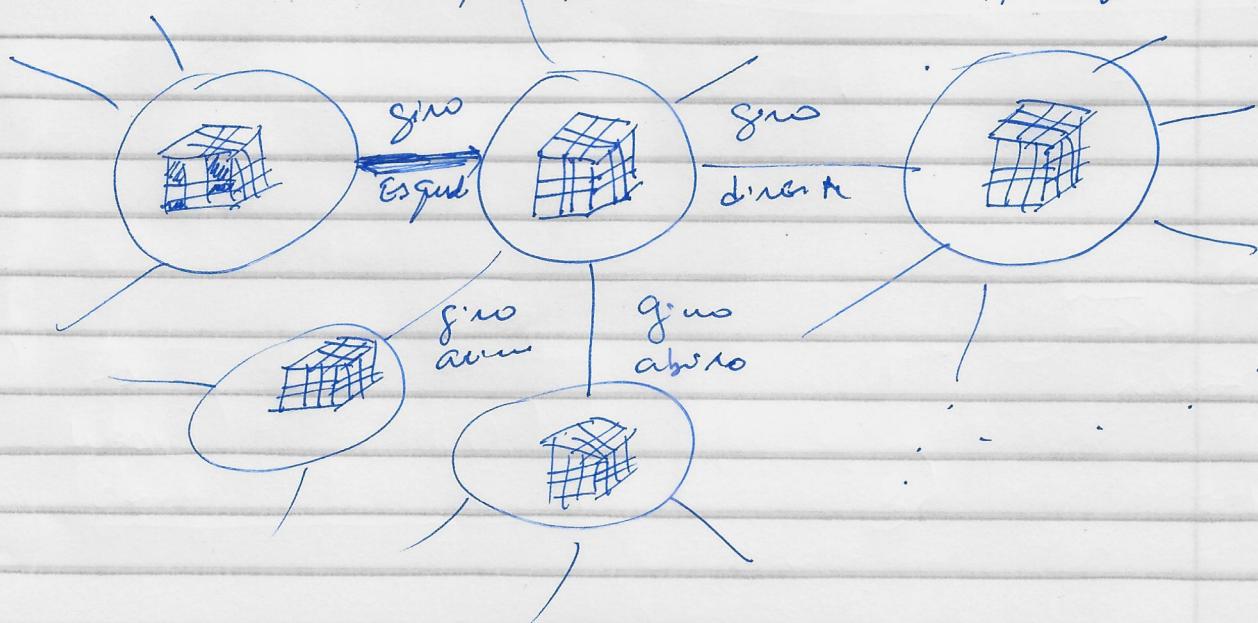
Caminhos com  
até 3 arestas

e por fim, Capinhos com até 8 arestas, que é equivalente a capinhos com até 5 arestas, visto que  $|G.V| = 6$ .

$$D^{(5)} = D^{(8)} \left( \begin{array}{cccccc} 0 & 1 & 2 & 1 & 2 & 1 \\ \infty & 0 & 1 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 1 & \infty \\ \infty & \infty & 1 & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 \end{array} \right) //$$

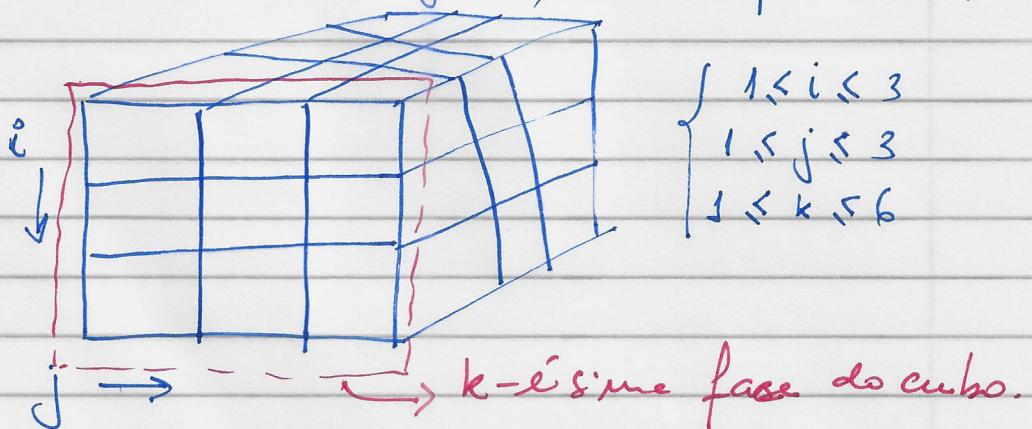
4) a) Vértices  $\Rightarrow$  representam uma possível configuração das peças do cubo mágico.

Arestas  $\Rightarrow$  possíveis transições com uma jogada (ou movimentação) no cubo mágico.



b) Uma possível função heurística para a distância é uma "linha reta" entre a posição atual do peão e a posição desejada.

Como o processo é 3-dimensional, cada peça tem 3 índices de localização, como por exemplo  $i, j, k$



Seja  $i_p^*, j_p^*$  e  $k_p^*$  as posições na solução da cadeia  $p$ . entao

$\xrightarrow{\text{função heurística}}$  
$$h_p(n) = \sqrt{(i_p - i_p^*)^2 + (j_p - j_p^*)^2 + (k_p - k_p^*)^2}$$

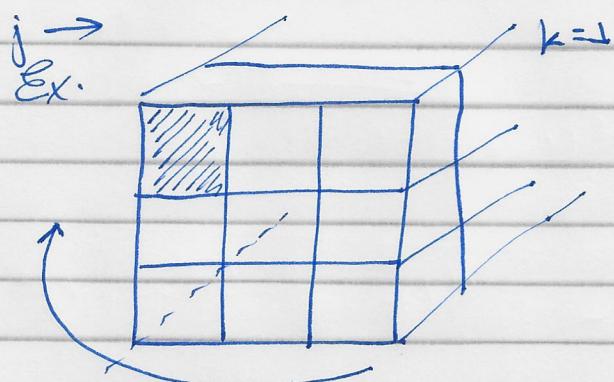
para uma  $n$ -ésima  
peça  $p$  nodo de  $G$

$$h(n) = \sum_p h_p(n)$$

Como o custo real para deslocar as peças sempre permanecerá um distorção sobre a superfície do cubo, o custo real sempre será maior que  $h(n)$ .

Logo  $h(n)$  é admissível!

c) para o caso real é possível definir a quantidade de deslocamento da peça p do estado inicial até o estado n



Suponha um movimento onde uma face é girada em  $90^\circ$ , o movimento mais simples

A peça ~~achunca~~ deixa de sentir as forças  $(1, 1, 1)$  para a posição  $(1, 3, 1)$ . Portanto a peça "anda" 2 casas, ou

$$((1, 1, 1) - (1, 3, 1))^2$$

$$\sqrt{(1-1)^2 + (1-3)^2 + (1-1)^2} = \underline{\underline{2}}$$

Seja  $\gamma$  o percurso executado pelas grades, iniciado no estado inicial até o estado n

Siga a posição da peça p no estado n dada por

$$(i_p(n), j_p(n), k_p(n))$$

e defina  $(i_p(-1), j_p(-1), k_p(-1)) = (i_p(0), j_p(0), k_p(0))$

E assim

$$g(n) = \sum_{r=0}^n \sqrt{(i_p(r) - i_p(r-1))^2 + (j_p(r) - j_p(r-1))^2 + (k_p(r) - k_p(r-1))^2}$$

$$\text{logo } f(n) = g(n) + h(n)$$

$$f(n) = \left\{ \sum_r \sqrt{(i_p(r) - i_p(r-1))^2 + (j_p(r) - j_p(r-1))^2} + \right.$$
$$\left. + \sqrt{(k_p(r) - k_p(r-1))^2} \right\} + \sum_p \sqrt{(i_p - i_p^*)^2 + (j_p - j_p^*)^2 + (k_p - k_p^*)^2}$$