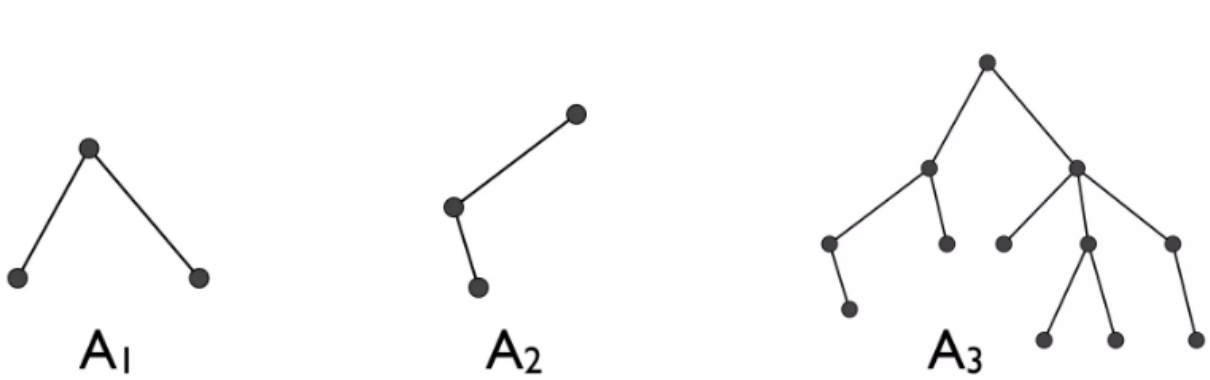
Definição

Uma **árvore (livre)** T é um grafo simples tal que para todo par de vértices v e w, existe um único caminho de v a w. Um **árvore com raiz** é uma árvore com um vértice designado como raiz.

Exemplo



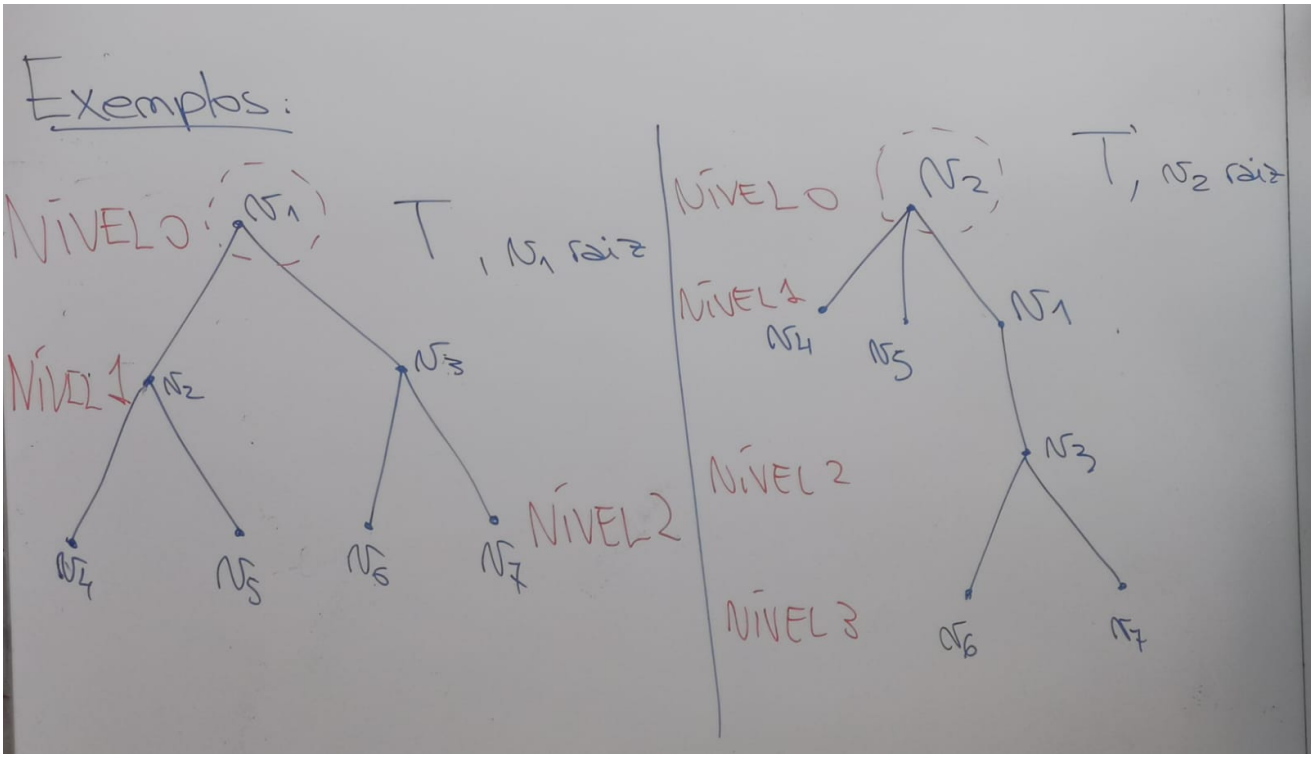
- Chaves da semifinal-final-campeã

Nível de um Vértice

> Definição: Numa árvore T com raiz v , diz-se que o **NÍVEL** de um vértice w é K, se K é a distância da raiz v a w.

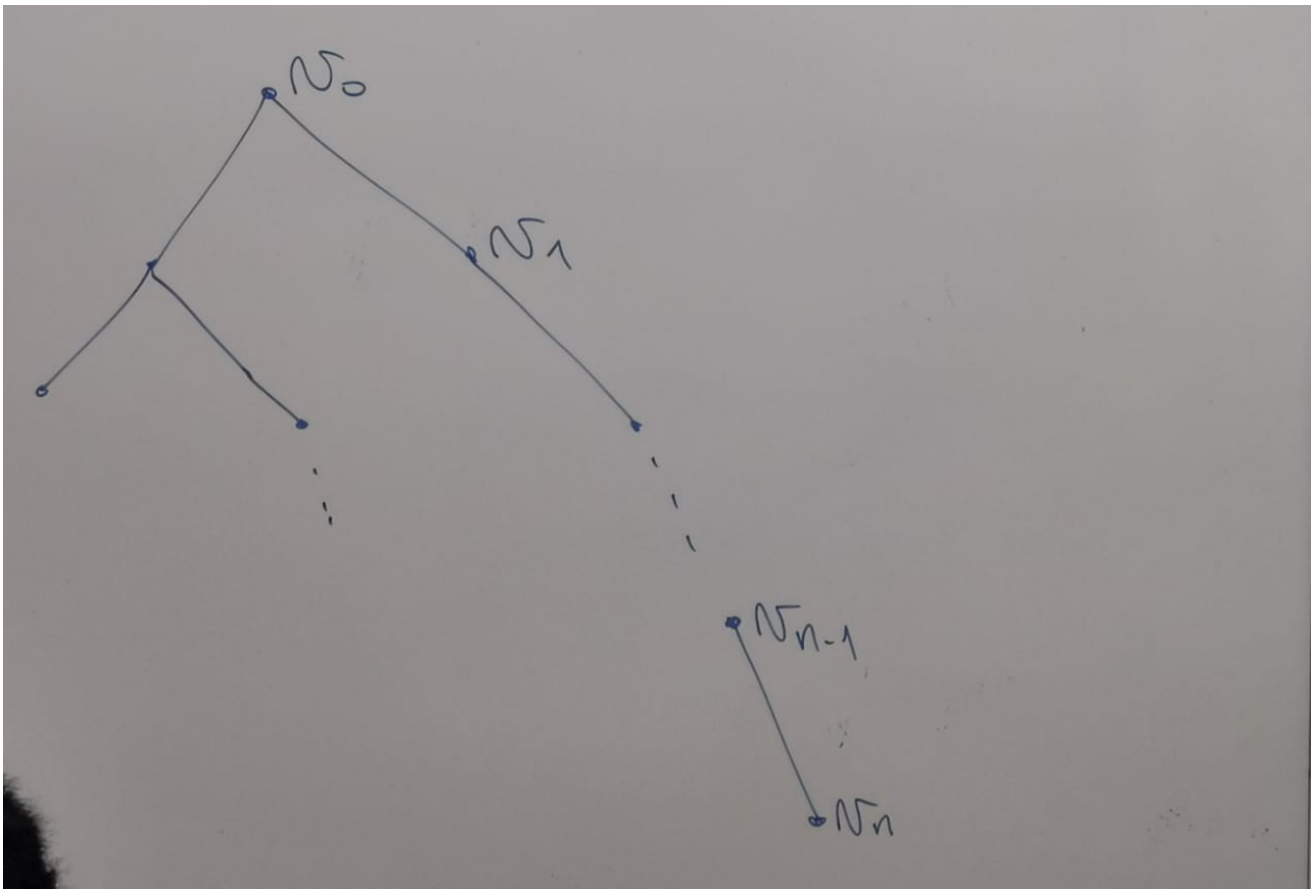
Altura de uma Árvore

> Definição: A altura de T é o valor do máximo nível



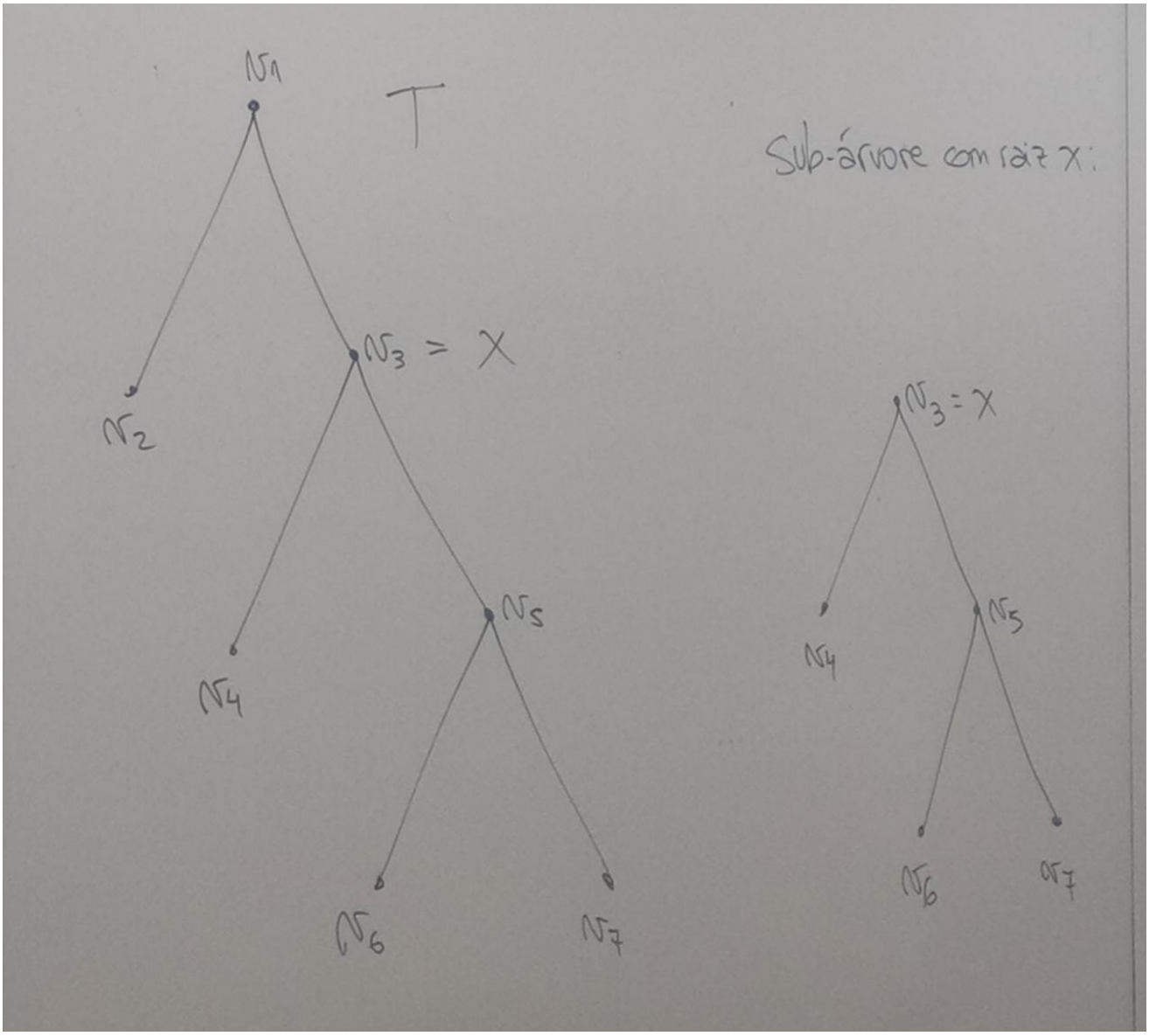
Propriedades

> Definicão: Seja T uma árvore com raiz v_0. Sejam x, y, z vértices de T e $(v_0, v_1, \ldots, v_n)$ um caminho simples em T.



Então:
(a) v_{n-1} é o **pai** de v_n.
(b) $v_0, \ldots, v_{n-1}$ são **antecessores** de v_n.
(c) v_n é **filho** de v_{n-1}.
(d) se x é antecessor de y, então y é **descendente** de x.
(e) se x e y são filhos de z, então x e y são irmãos.
(f) se x não tem filhos, x é um **vértice terminal** ou **folha**.
(g) se x não é termninal, ele é **vértice interno** ou **vértice tronco**.
(h) a **sub-árvore** de T com raiz x é o grafo com conjunto de vértices V e conjunto de arestas E, onde V. contém x e todos seus descendentes e

$$E := \{\text{e/ e é aresta em algum caminho simples de x a algum vértice de V}\}$$



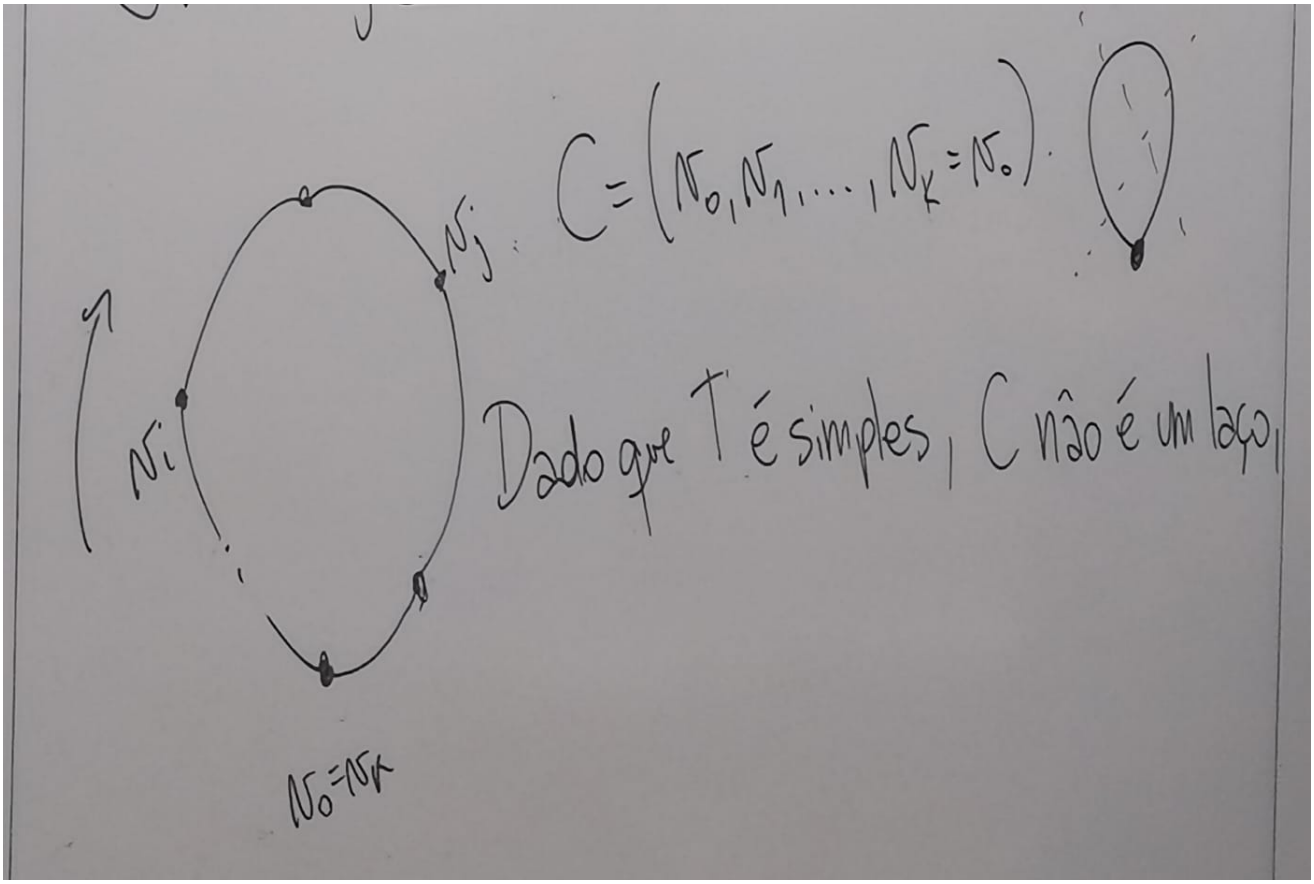
Afirmações

Teorema: Seja T um grafo com n vértices.
Então as seguintes afirmações são equivalentes.

(a) T é uma árvore.
(b) T é conexo e acíclico (não contém ciclos).
(c) T é conexo e tem (exatamente) $n-1$ arestas.
(d) T é acíclico e tem (exatamente) $n-1$ arestas.

Prova $(a) \Rightarrow (b)$

Seja T uma árvore, então T é conexo porque todos os vértices estão conectados entre si.
Vamos provar que T não contém ciclos.
Supomos pelo contrário que T contém um ciclo C'. Seja C um ciclo simples contido em C'.



Logo C visita pelo menos dois vértices diferentes v_i e v_j, com $i < j$. Então:

$$(v_i, v_{i+1}, \ldots, v_j) \quad \text{e} \quad (v_i, v_{i-1}, \ldots, v_0 = v_k, \ldots, v_j)$$

São dois caminhos diferentes de v_i a v_j. Isto contradiz a definição de árvore. Concluímos que T é acíclico.

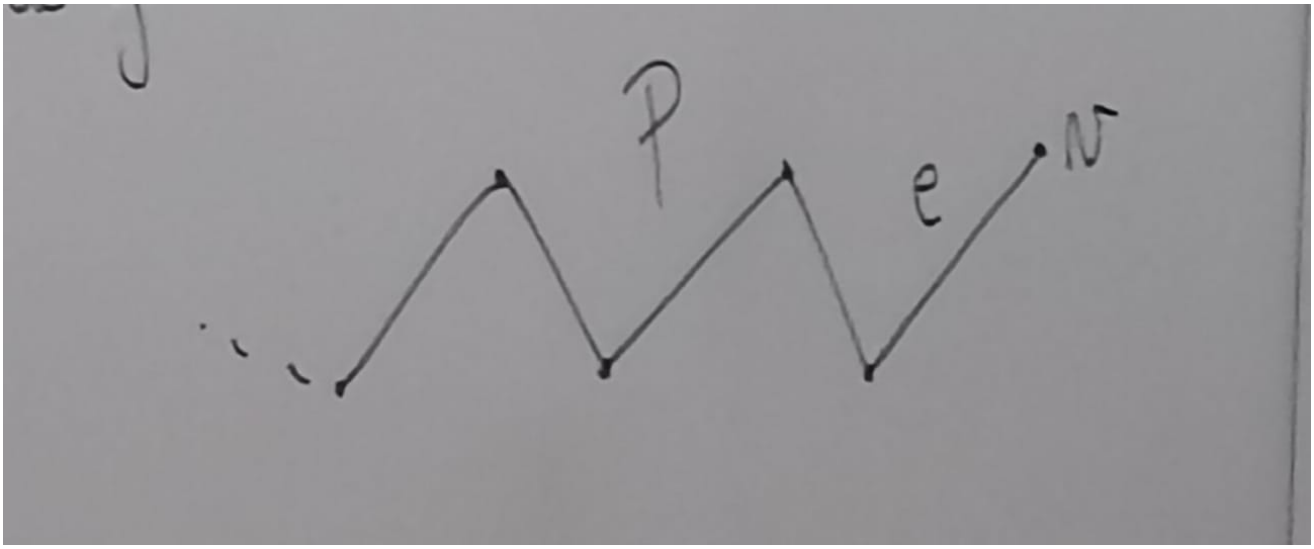
Prova $(b) \Rightarrow (c)$

Seja T conexo e acíclico (com n vértices). Vamos provar por indução em n.

Para $n = 1$:
· (sim, é um ponto)

Tem e = 0 = n - 1 arestas

Supomos que a tese vale para $n = k$, e seja T conexo e acíclico com $k + 1$ vértices. Seja P um caminho em T de cumprimeto máximo, sem arestas repetidas. Dado que T é acíclico, P acaba em um vértice v de grau 1.



Chamamos de e a última aresta visitada.
Consideramos o subgrafo T^* obtido de T removendo v e e. Logo T^* tem k vértice, é conexo e acíclico. Então por hipótese indutiva T^* tem $k-1$ arestas. Concluímos que tem k arestas.

Prova $(c) \Rightarrow (d)$

- Exercício 😔

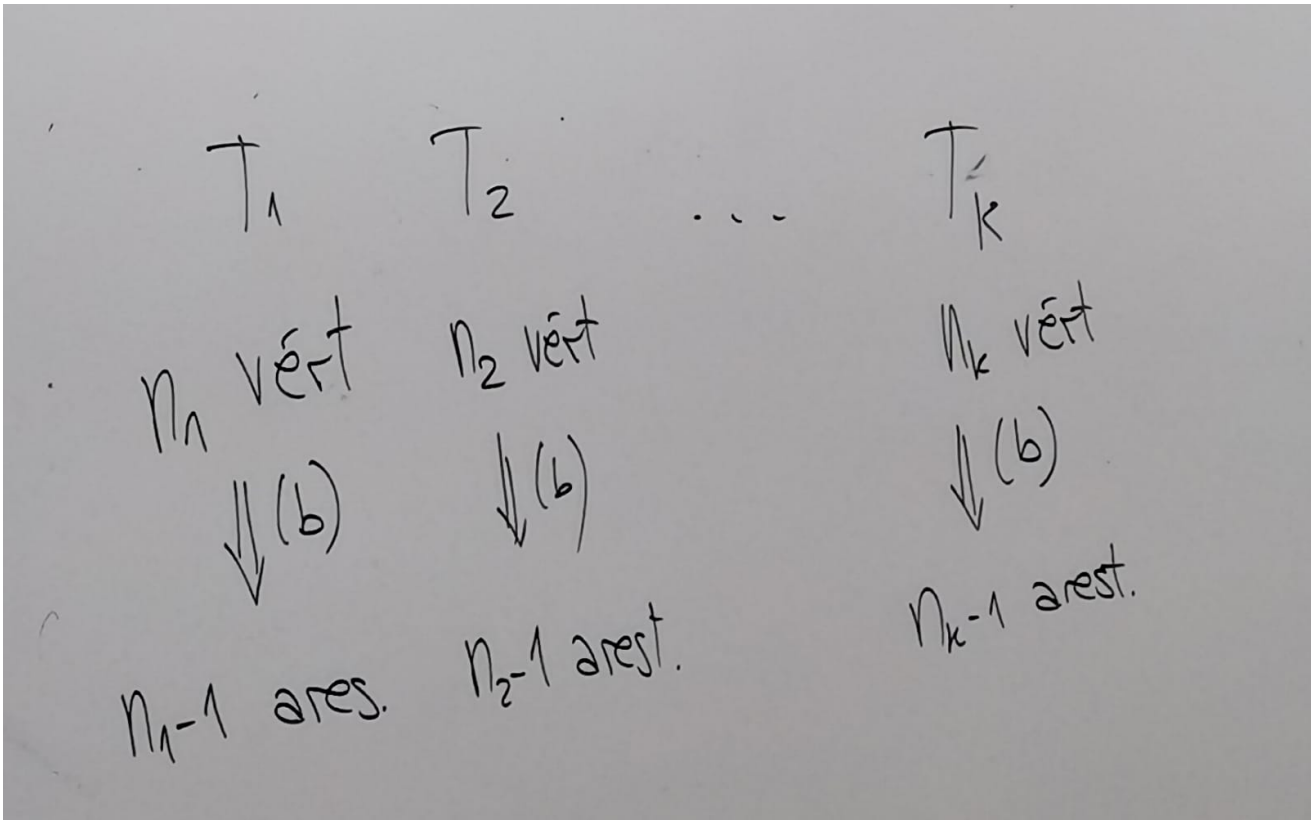
Prova $(d) \Rightarrow (a)$

Seja T acíclico com (n vértices e) $n-1$ arestas.

T acíclico $\Rightarrow$ T simples
T acíclico $\Rightarrow$ não existem dois caminhos entre nenhum par de vértices.

Só falta provar que T é conexo.

Sejam $T_1, \ldots, T_k$ componentes conexas de T.



$n_1 + n_2 + \ldots + n_k = n$ vért.

Arestas: $n_1 - 1 + \ldots + n_k - 1 = n - k$ arestas
Dado que T tem $n - 1$ arestas, necessariamente $k = 1$.

Aplicações

- Modelagem de estruturar hierárquicas
- Sistemas de arquivos no computador (directory tree)
  Exemplo: Home $\rightarrow$ Program Files $\rightarrow$
  $\rightarrow$ Users $\rightarrow$
  $\rightarrow$ Windows $\rightarrow$
- Códigos de Huffman: representações com caract. com cadeias de bits com comprimentos variáveis.
  Exemplo: ASCII $\rightarrow$ 7-bits representações

Descrição:
Input(dados):
$A = (a_1, \ldots, a_n)$ alfabeto (a ser representado)
$W = (w_1, \ldots, w_n)$ peso (frequência)
$w_i = \text{weight}(a_i)$, $i = 1, \ldots, n$

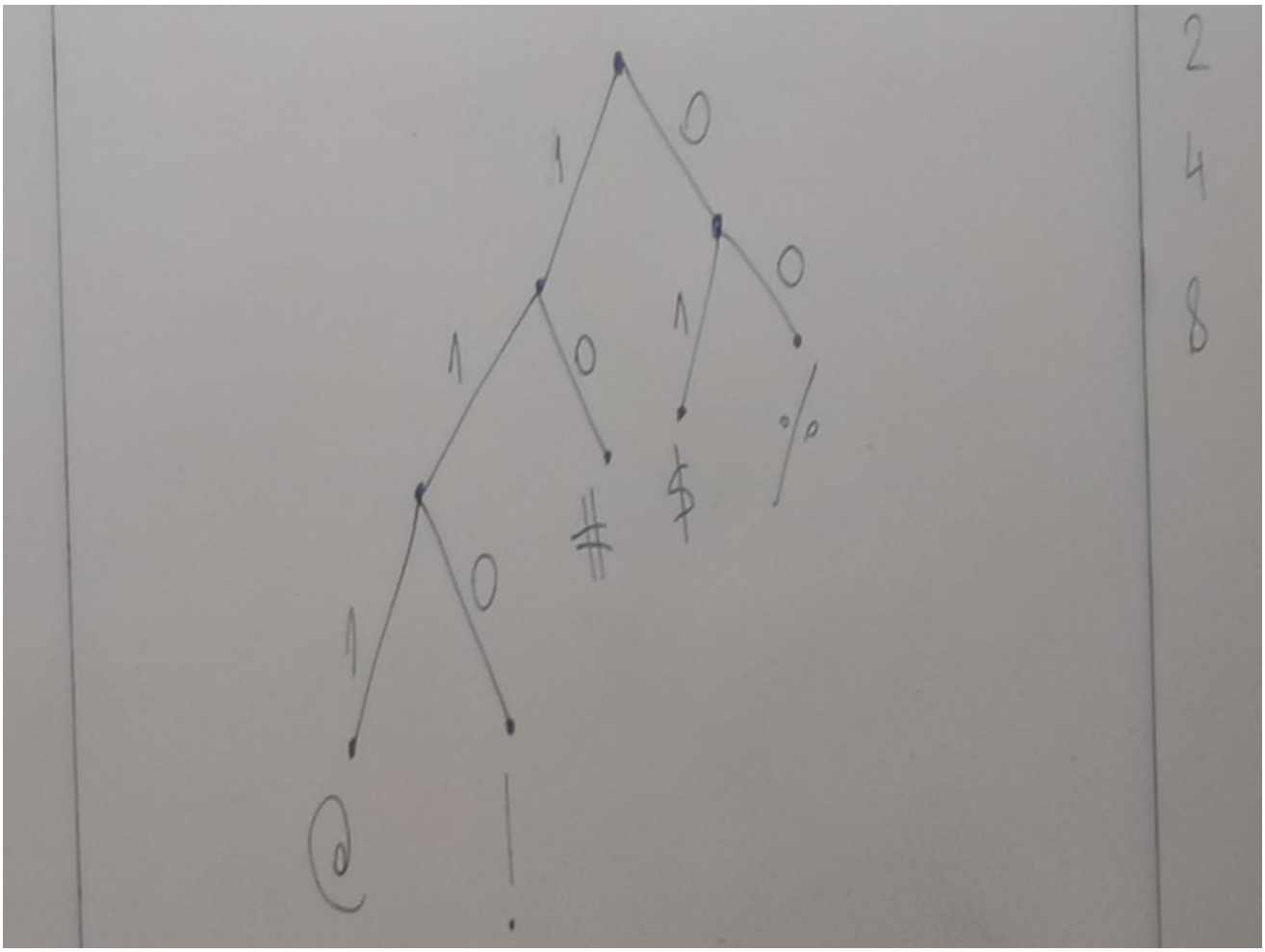
Objetivo: achar um código
$C(W) = (c_1, \ldots, c_n) \rightarrow \text{output}$
oned cada c_i é uma cadeia de 0 e 1 tal que:

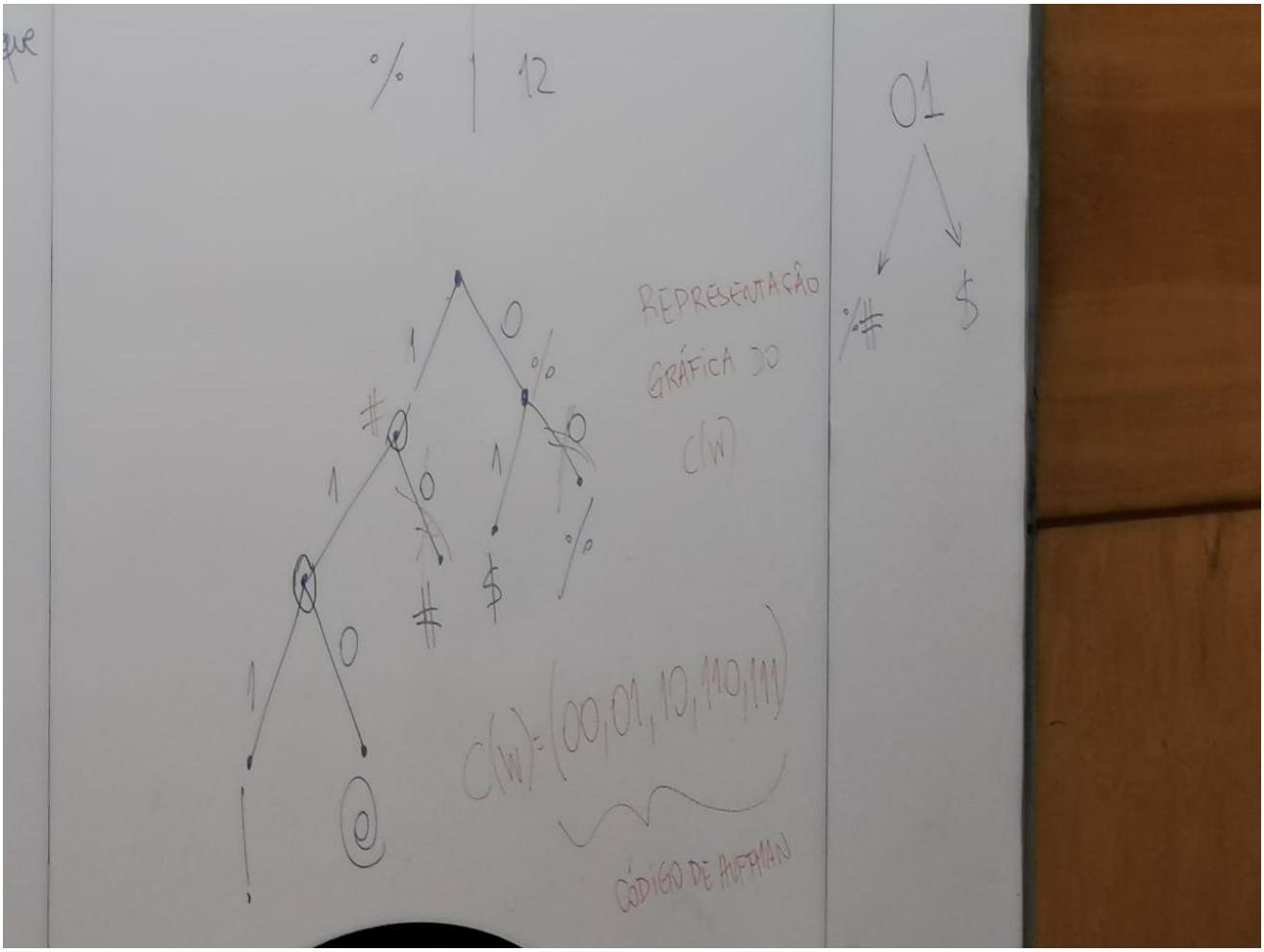
$$L(C(w)) = \sum_{i=1}^{n} w_i \, \text{length}(c_i)$$

seja minimizado.

caracteres	frequências
!	2
@	3
#	7
$	8
%	12

$$W = (12, 8, 7, 3, 2)$$





Construção de um código Huffman

We show how Algorithm 9.1.9 constructs an optimal Huffman code using Table 9.1.2.

TABLE 9.1.2 ■ Input for Example 9.1.10.

Character	*Frequency*
!	2
@	3
#	7
$	8
%	12

The algorithm begins by repeatedly replacing the smallest two frequencies with the sum until a two-element sequence is obtained:

$$2, 3, 7, 8, 12 \rightarrow 2 + 3, 7, 8, 12$$
$$5, 7, 8, 12 \rightarrow 5 + 7, 8, 12$$
$$8, 12, 12 \rightarrow 8 + 12, 12$$
$$12, 20$$

The algorithm then constructs trees working backward beginning with the two-element sequence 12, 20 as shown in Figure 9.1.13. For example, the second tree is obtained from the first by replacing the vertex labeled 20 by the tree of Figure 9.1.14 since 20 arose as the sum of 8 and 12. Finally, to obtain the optimal Huffman coding tree, we replace each frequency by a character having that frequency (see Figure 9.1.15).

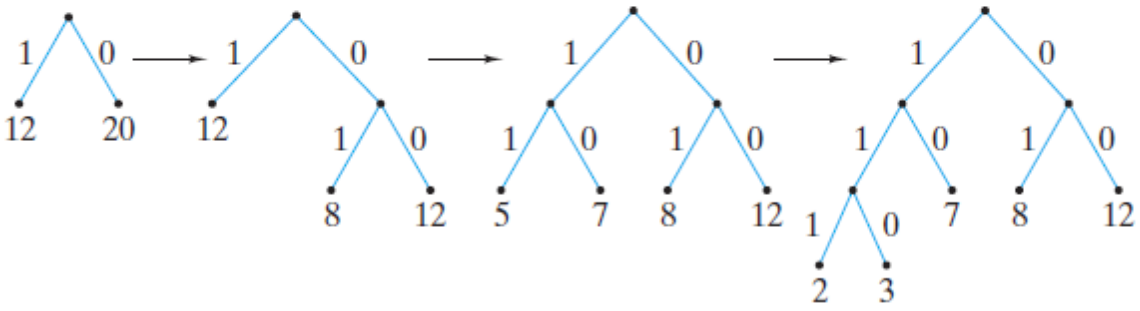


Figure 9.1.13 Constructing an optimal Huffman code.

Por último, basta substituir os caracteres pelas frequências na árvore Huffman.