

# Códigos de Huffman

**Motivação:** compressão de textos

# Códigos de Huffman

**Motivação:** compressão de textos

**Código ASCII:** todo símbolo tem um código de 8 bits.

# Códigos de Huffman

**Motivação:** compressão de textos

**Código ASCII:** todo símbolo tem um código de 8 bits.

$\Sigma$ : alfabeto finito

$f_i$ : frequência de símbolo  $i$  em  $\Sigma$

(coleção de números não-negativos cuja soma é 1)

# Códigos de Huffman

**Motivação:** compressão de textos

**Código ASCII:** todo símbolo tem um código de 8 bits.

$\Sigma$ : alfabeto finito

$f_i$ : frequência de símbolo  $i$  em  $\Sigma$

(coleção de números não-negativos cuja soma é 1)

**Objetivo:** atribuir um código binário para cada símbolo de modo que um texto seja convertido para um arquivo binário o mais compacto possível e seja fácil de decodificar.

# Códigos de Huffman

**Motivação:** compressão de textos

**Código ASCII:** todo símbolo tem um código de 8 bits.

$\Sigma$ : alfabeto finito

$f_i$ : frequência de símbolo  $i$  em  $\Sigma$

(coleção de números não-negativos cuja soma é 1)

**Objetivo:** atribuir um código binário para cada símbolo de modo que um texto seja convertido para um arquivo binário o mais compacto possível e seja fácil de decodificar.

**Códigos livres de prefixo:** o código de um símbolo não é prefixo do código de nenhum outro símbolo.

# Códigos de Huffman

**Motivação:** compressão de textos

**Código ASCII:** todo símbolo tem um código de 8 bits.

$\Sigma$ : alfabeto finito

$f_i$ : frequência de símbolo  $i$  em  $\Sigma$

(coleção de números não-negativos cuja soma é 1)

**Objetivo:** atribuir um código binário para cada símbolo de modo que um texto seja convertido para um arquivo binário o mais compacto possível e seja fácil de decodificar.

**Códigos livres de prefixo:** o código de um símbolo não é prefixo do código de nenhum outro símbolo.

Códigos livres de prefixo são fáceis de decodificar.

## Exemplo

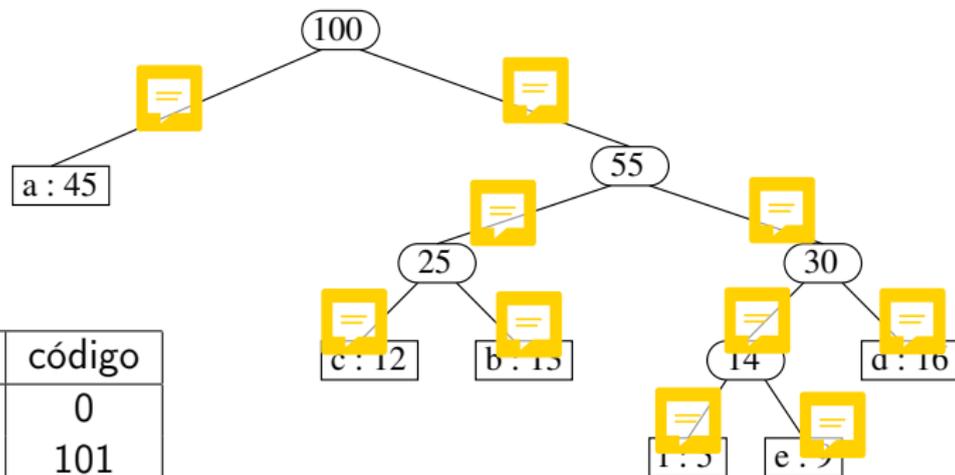
letra	freq	código
a	45	
b	13	
c	12	
d	16	
e	9	
f	5	

# Exemplo

letra	freq	código
a	45	0
b	13	101
c	12	100
d	16	111
e	9	1101
f	5	1100



# Exemplo



letra	freq	código
a	45	0
b	13	101
c	12	100
d	16	111
e	9	1101
f	5	1100



# Algoritmo de Huffman

$n$ : número de símbolos em  $\Sigma$

# Algoritmo de Huffman

$n$ : número de símbolos em  $\Sigma$

**Guloso:**

Comece com  $n$  árvores disjuntas, cada uma com um único nó, com o símbolo e sua frequência.

a : 45

d : 16

b : 13

c : 12

e : 9

f : 5

# Algoritmo de Huffman

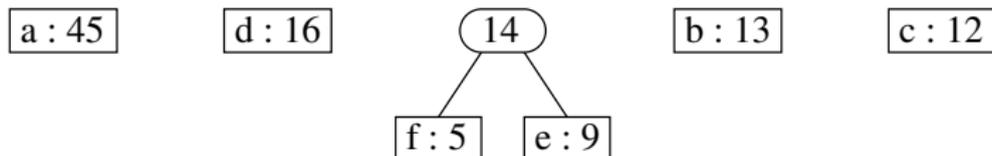
$n$ : número de símbolos em  $\Sigma$

**Guloso:**

Comece com  $n$  árvores disjuntas, cada uma com um único nó, com o símbolo e sua frequência.



A cada iteração, escolha as duas árvores de frequência menor e junte-as, com frequência somada.



# Algoritmo de Huffman

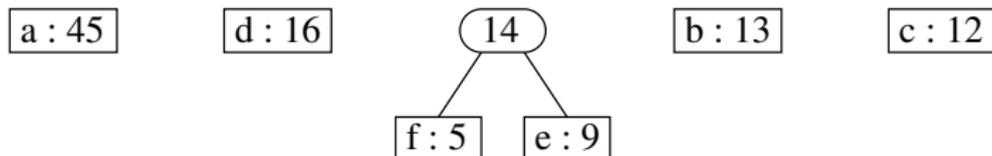
$n$ : número de símbolos em  $\Sigma$

**Guloso:**

Comece com  $n$  árvores disjuntas, cada uma com um único nó, com o símbolo e sua frequência.



A cada iteração, escolha as duas árvores de frequência menor e junte-as, com frequência somada.



Pare quando restar uma única árvore.

## Algoritmo de Huffman: exemplo

a: 45

d: 16

b: 13

c: 12

e: 9

f: 5

## Algoritmo de Huffman: exemplo

a : 45

d : 16

b : 13

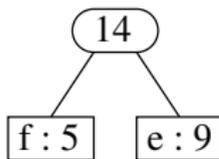
c : 12

e : 9

f : 5

a : 45

d : 16

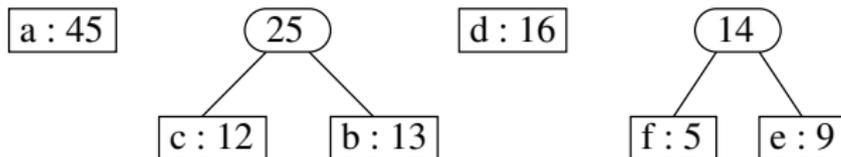
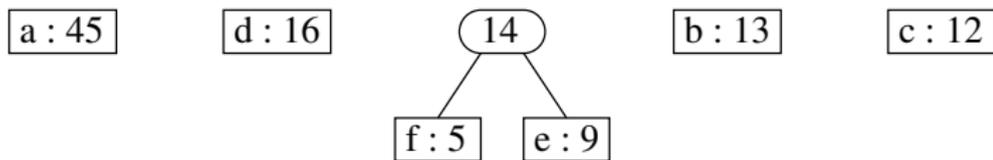


b : 13

c : 12

# Algoritmo de Huffman: exemplo

a : 45      d : 16      b : 13      c : 12      e : 9      f : 5



# Algoritmo de Huffman: exemplo

a : 45

d : 16

b : 13

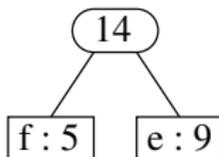
c : 12

e : 9

f : 5

a : 45

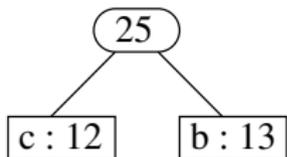
d : 16



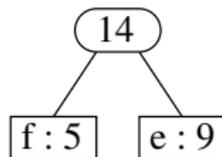
b : 13

c : 12

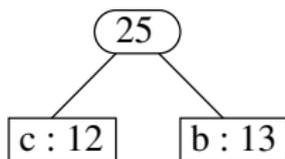
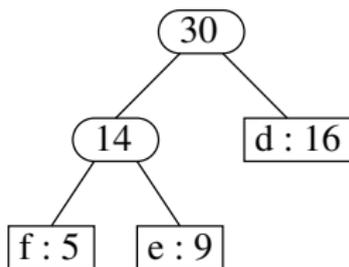
a : 45



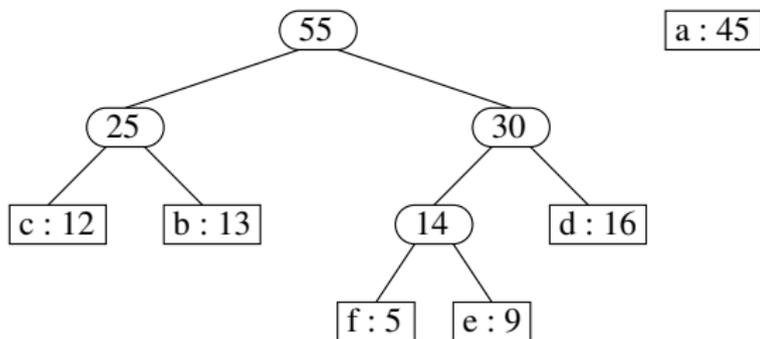
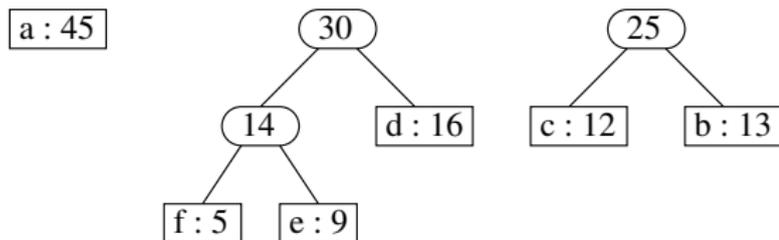
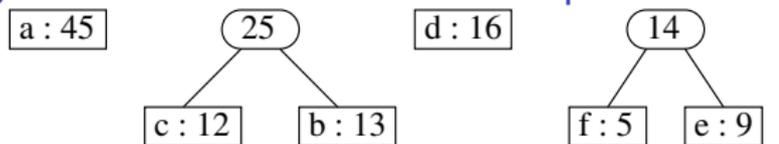
d : 16



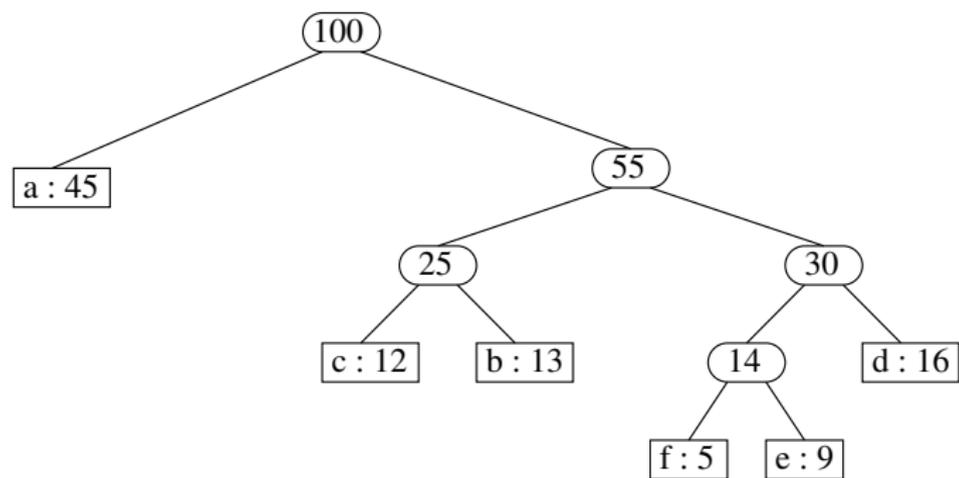
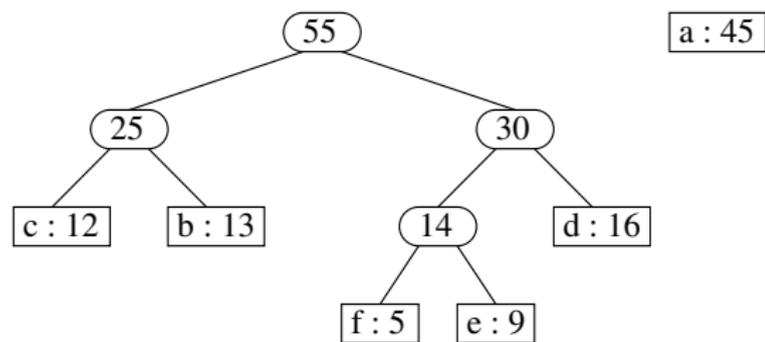
a : 45



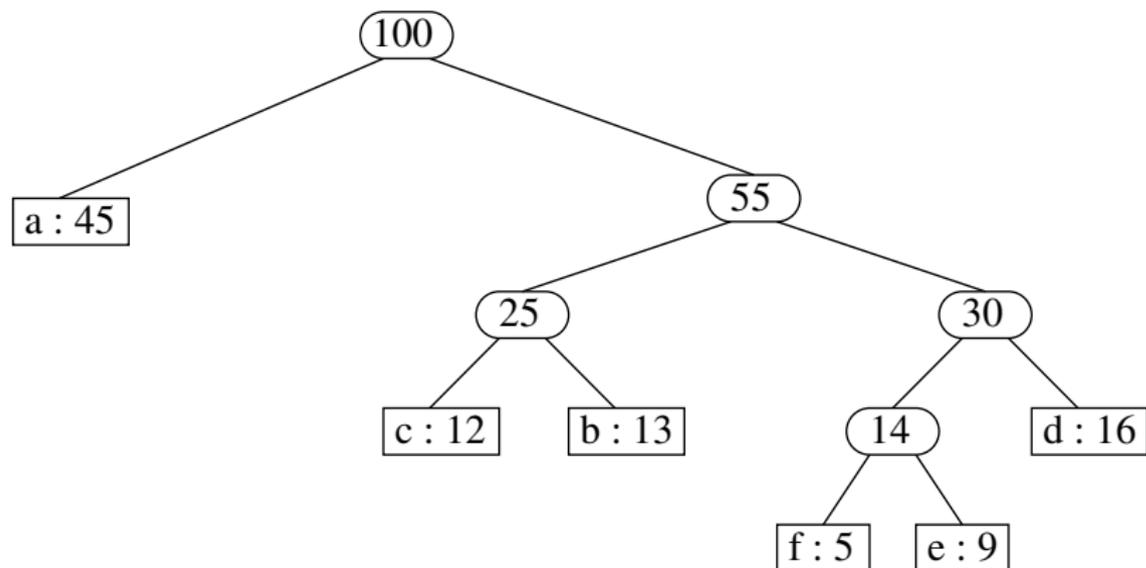
# Algoritmo de Huffman: exemplo



## Algoritmo de Huffman: exemplo



# Árvore de Huffman



# Árvore de Huffman

Como obter os códigos a partir da árvore?

# Árvore de Huffman

Como obter os códigos a partir da árvore?

Associe a cada símbolo um número binário assim:

# Árvore de Huffman

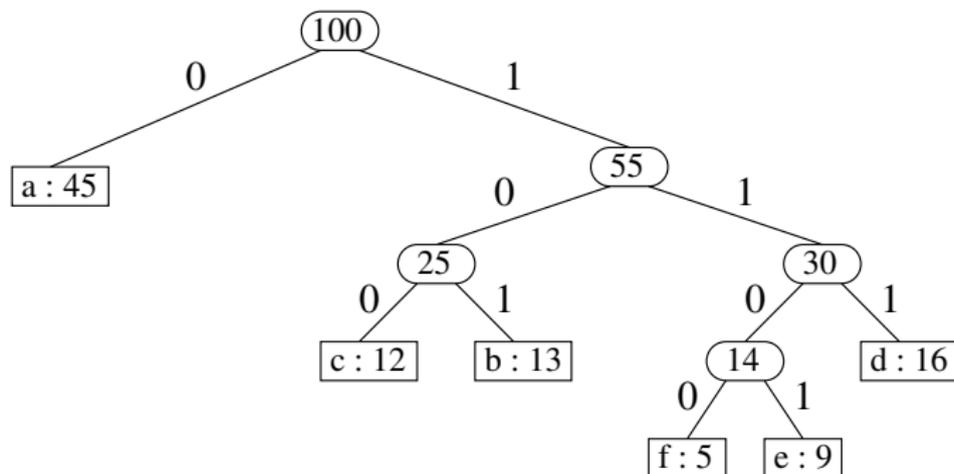
Como obter os códigos a partir da árvore?

Associe a cada símbolo um número binário assim:

Rotule com 0 as arestas da árvore

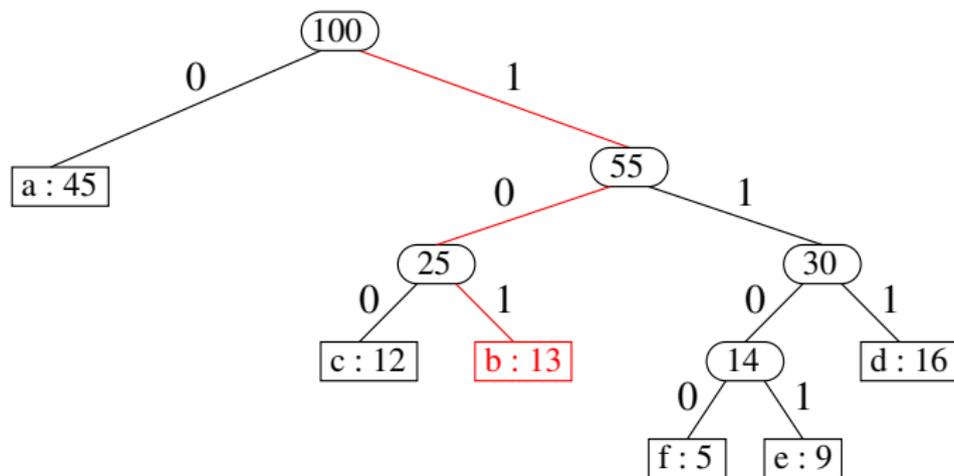
que ligam um nó com seu filho esquerdo e

com 1 as arestas que ligam um nó com seu filho direito.



# Árvore de Huffman

Como obter os códigos a partir da árvore?



O código correspondente a cada símbolo é a concatenação dos *bits* associados às arestas do caminho da raiz até a folha correspondente ao símbolo.

**Exemplo:** O código de *b* é 101.

# Algoritmo de Huffman

$n$ : número de símbolos em  $\Sigma$

**Guloso:**

Comece com  $n$  árvores disjuntas, cada uma com um único nó, com o símbolo e sua frequência.

A cada iteração, escolha as duas árvores de frequência menor e junte-as, com frequência somada.

Pare quando restar uma única árvore.

# Algoritmo de Huffman

$n$ : número de símbolos em  $\Sigma$

## Guloso:

Comece com  $n$  árvores disjuntas, cada uma com um único nó, com o símbolo e sua frequência.

A cada iteração, escolha as duas árvores de frequência menor e junte-as, com frequência somada.

Pare quando restar uma única árvore.

## Perguntas:

- ▶ Este algoritmo produz um código ótimo? 
- ▶ Como implementá-lo do modo mais eficiente possível?

# Algoritmo guloso

**HUFFMAN** ( $A, f, n$ )

```
1   $Q \leftarrow$  BUILD-MIN-HEAP( $A, f, n$ )
2  para  $i \leftarrow 1$  até  $n - 1$  faça
3     $x \leftarrow$  EXTRACT-MIN( $Q$ )
4     $y \leftarrow$  EXTRACT-MIN( $Q$ )
5     $z \leftarrow$  NOVA-CEL()
6     $esq[z] \leftarrow x$ 
7     $dir[z] \leftarrow y$ 
8     $f[z] \leftarrow f[x] + f[y]$ 
9    INSEREHEAP( $Q, z, f[z]$ )
10 devolva EXTRACT-MIN( $Q$ )
```

## Algoritmo guloso

**HUFFMAN** ( $A, f, n$ )

```
1   $Q \leftarrow \text{BUILD-MIN-HEAP}(A, f, n)$ 
2  para  $i \leftarrow 1$  até  $n - 1$  faça
3       $x \leftarrow \text{EXTRACT-MIN}(Q)$ 
4       $y \leftarrow \text{EXTRACT-MIN}(Q)$ 
5       $z \leftarrow \text{NOVA-CEL}()$ 
6       $\text{esq}[z] \leftarrow x$ 
7       $\text{esq}[z] \leftarrow y$ 
8       $f[z] \leftarrow f[x] + f[y]$ 
9       $\text{INSEREHEAP}(Q, z, f[z])$ 
10 devolva  $\text{EXTRACT-MIN}(Q)$ 
```

**Consumo de tempo:**  $O(n \lg n)$ .