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.

Σ: alfabeto
(um conjunto finito cujos elementos chamamos de símbolos)

f_i: frequência de símbolo i em Σ
(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.

Σ: alfabeto
(um conjunto finito cujos elementos chamamos de símbolos)

f_i: frequência de símbolo i em Σ
(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.

Σ: alfabeto
(um conjunto finito cujos elementos chamamos de símbolos)

f_i: frequência de símbolo i em Σ
(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.

Σ: alfabeto
(um conjunto finito cujos elementos chamamos de símbolos)

f_i: frequência de símbolo i em Σ
(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.

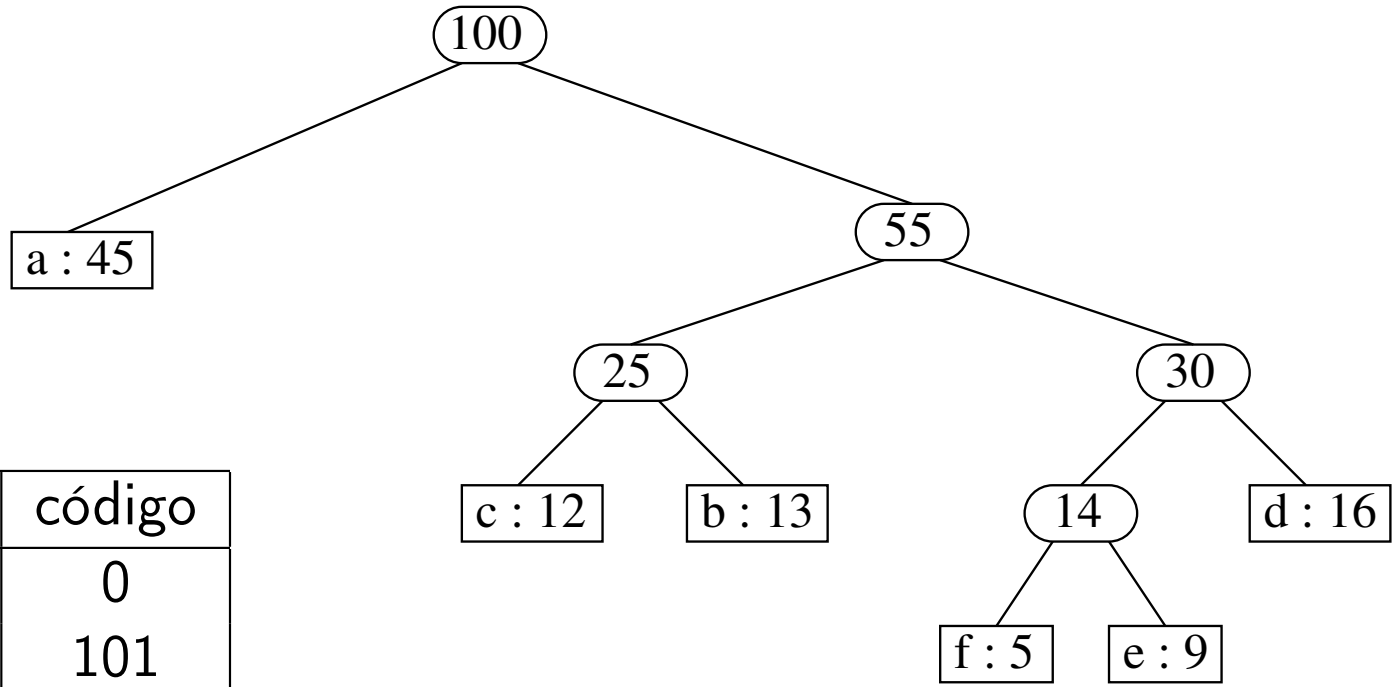
Exemplo

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

Árvore de Huffman

Como obter os códigos a partir da árvore?

Árvore de Huffman

Como obter os códigos a partir da árvore?

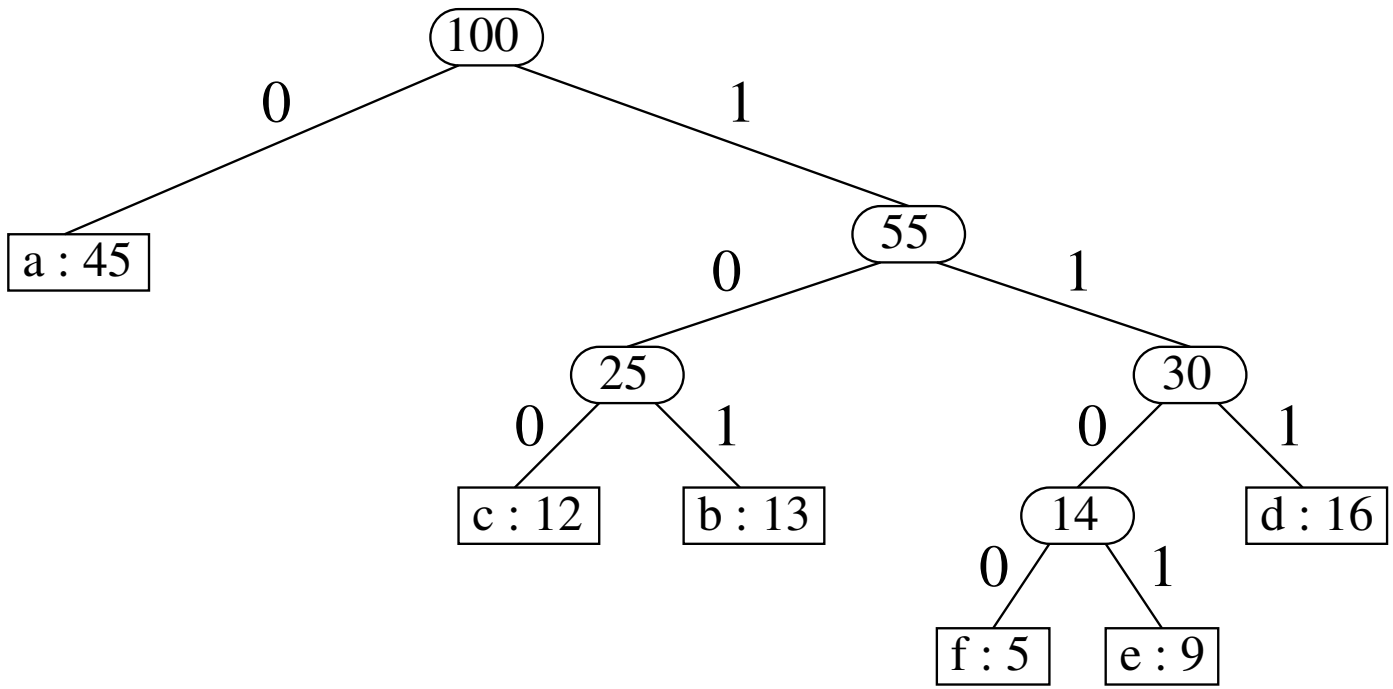
Associe a cada símbolo um número binário assim:

Árvore de Huffman

Como obter os códigos a partir da árvore?

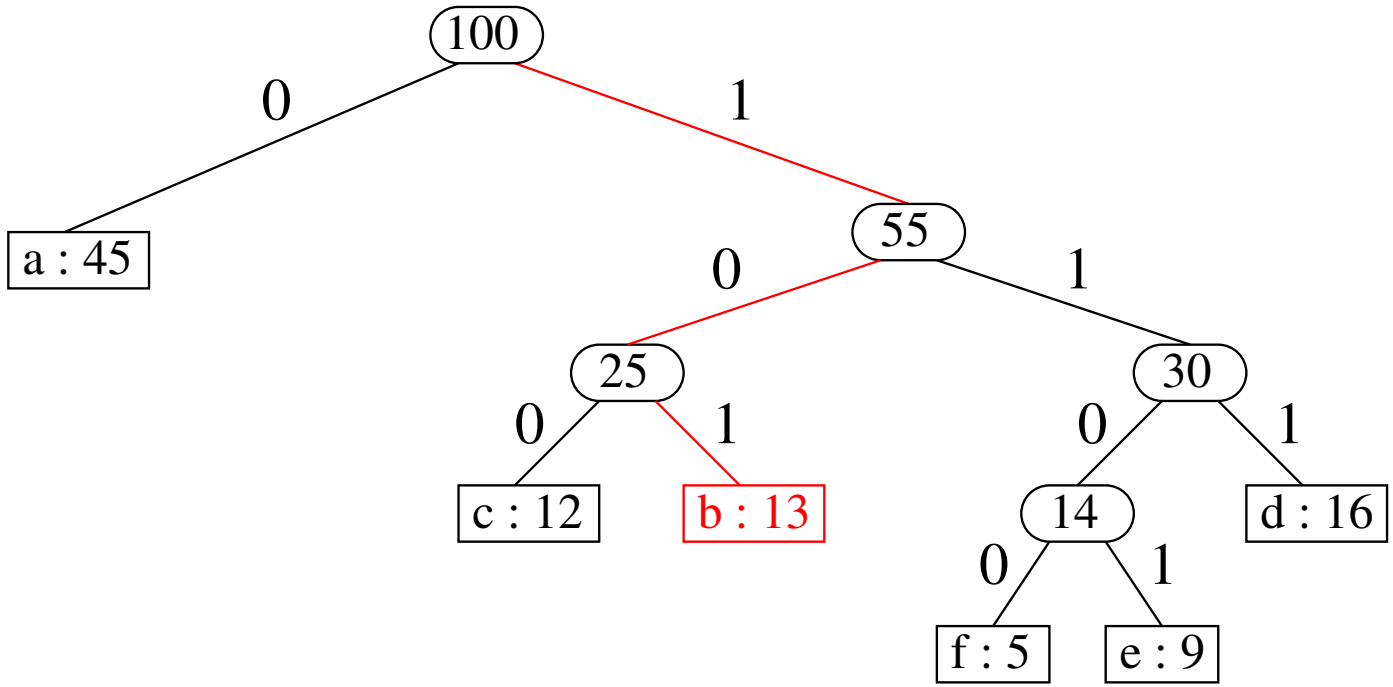
Associe a cada símbolo um número binário assim:

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 Σ

Algoritmo de Huffman

n: número de símbolos em Σ

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

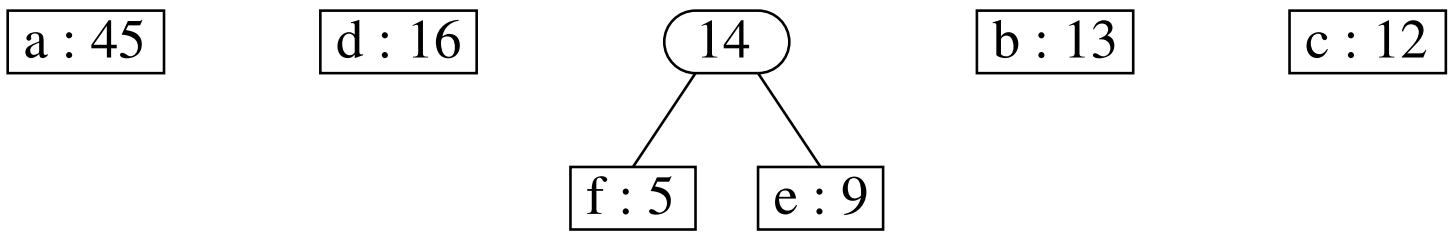
n: número de símbolos em Σ

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

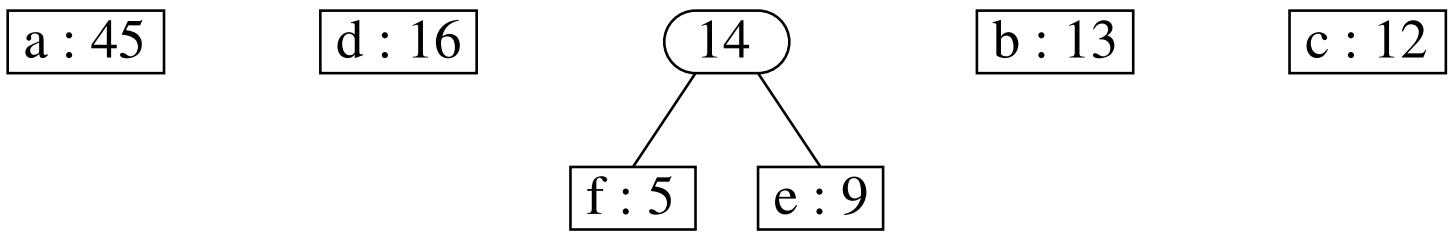
n: número de símbolos em Σ

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.

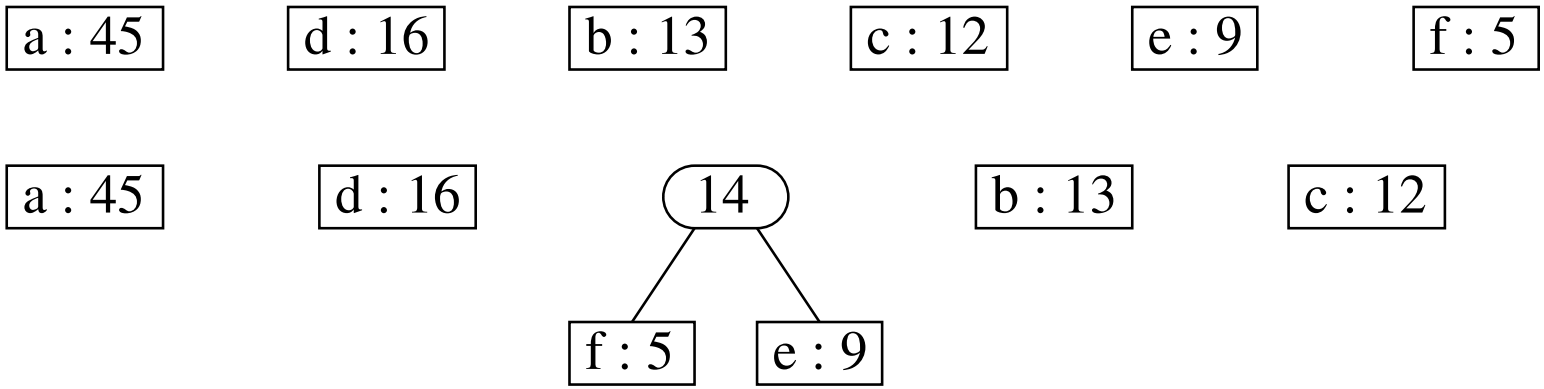


Pare quando restar uma única árvore.

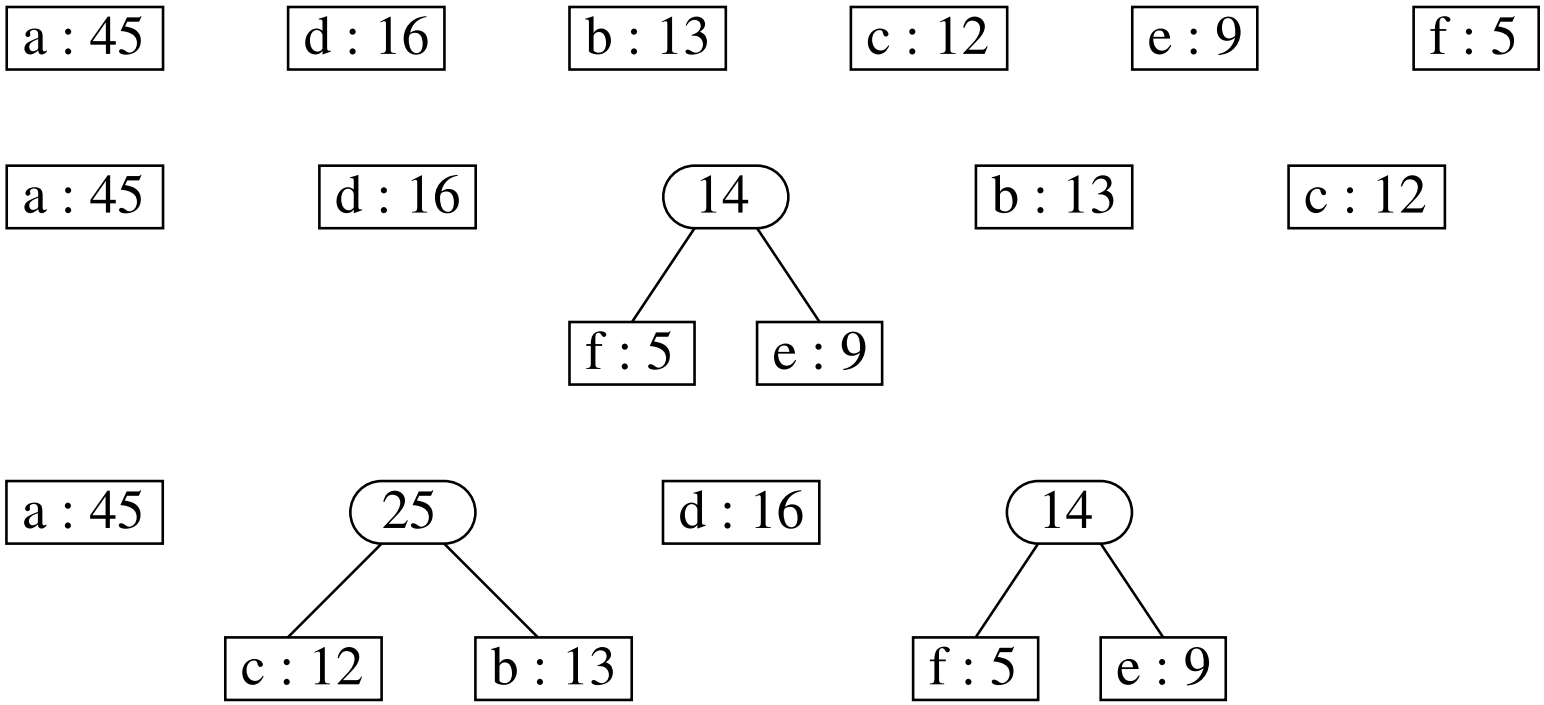
Algoritmo de Huffman: exemplo



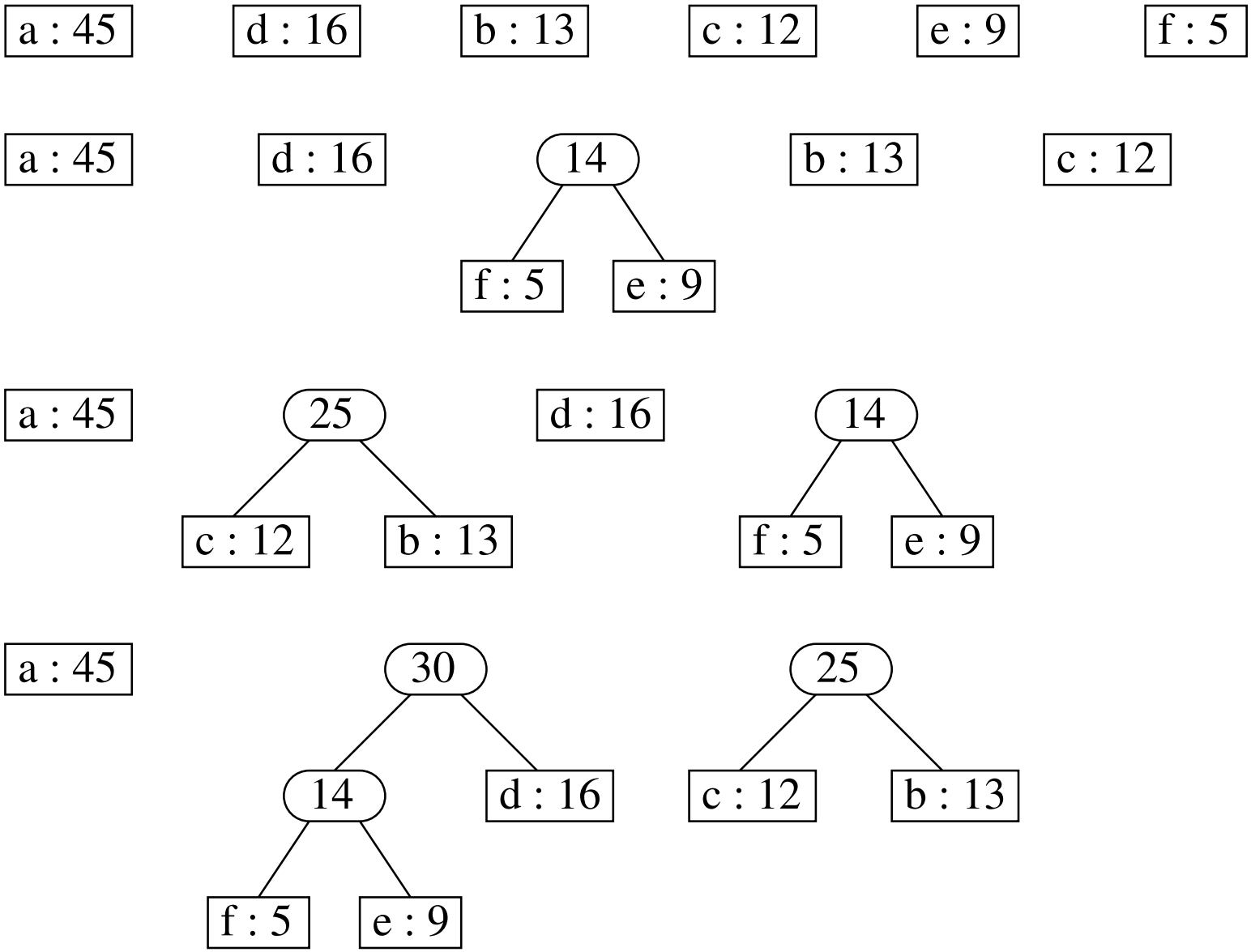
Algoritmo de Huffman: exemplo



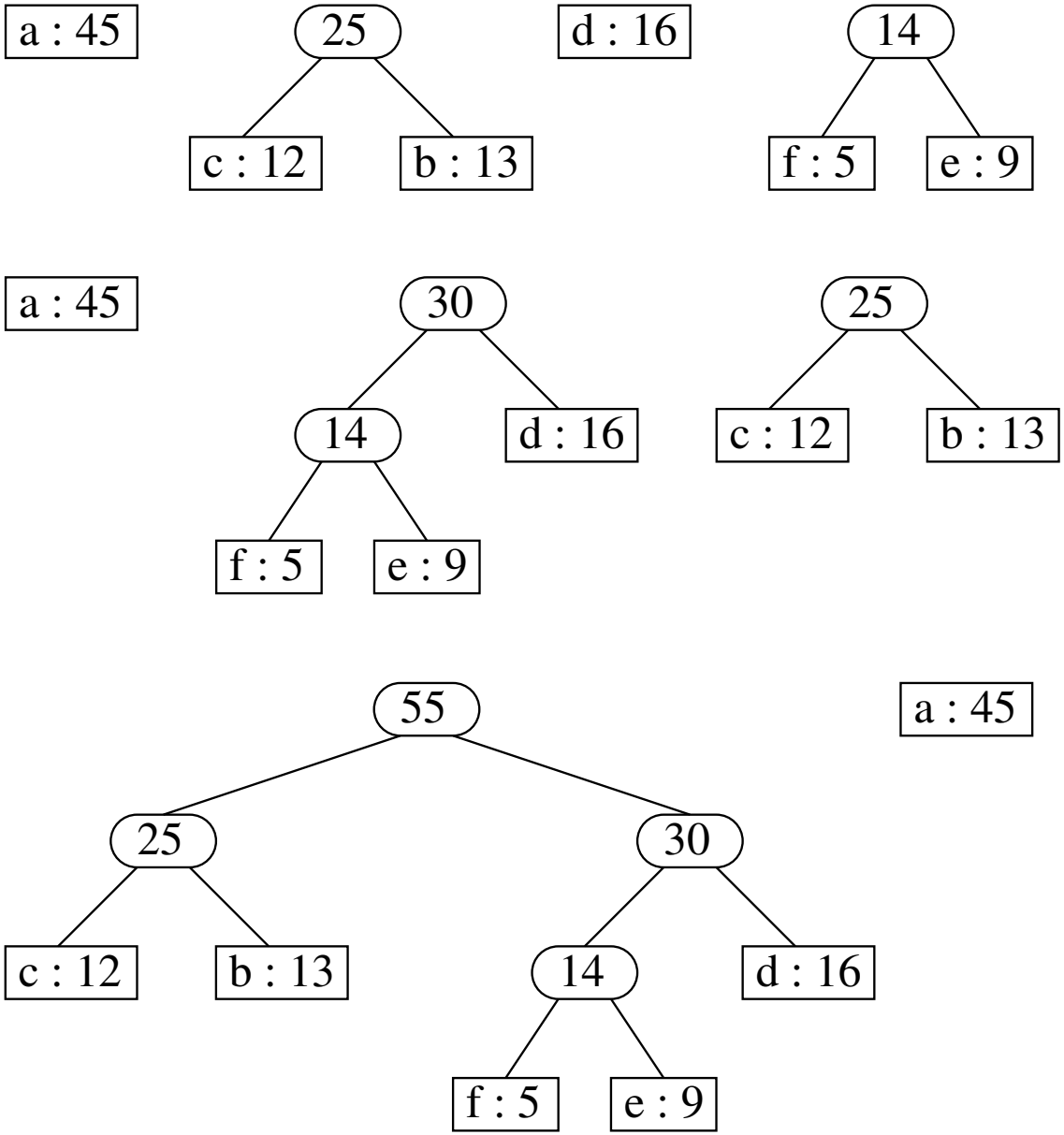
Algoritmo de Huffman: exemplo



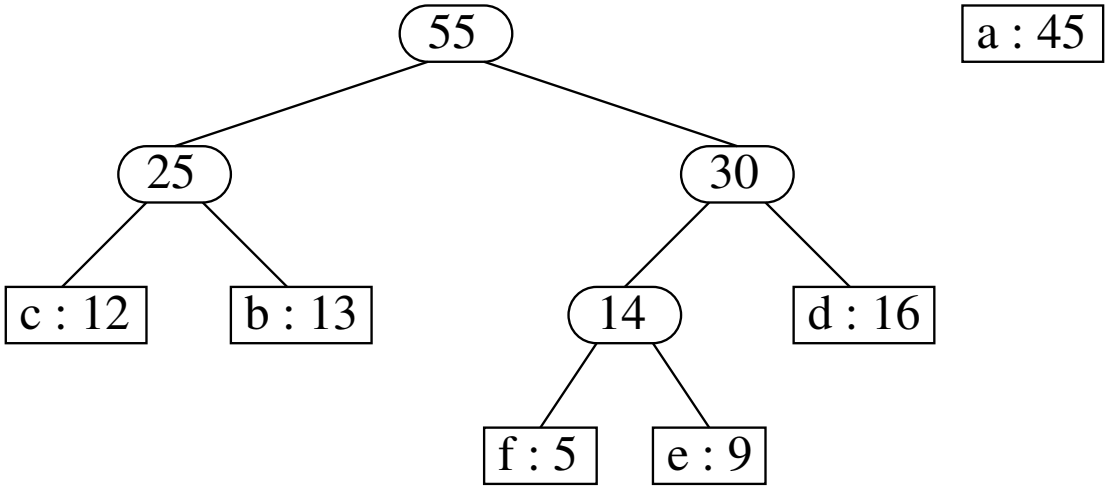
Algoritmo de Huffman: exemplo

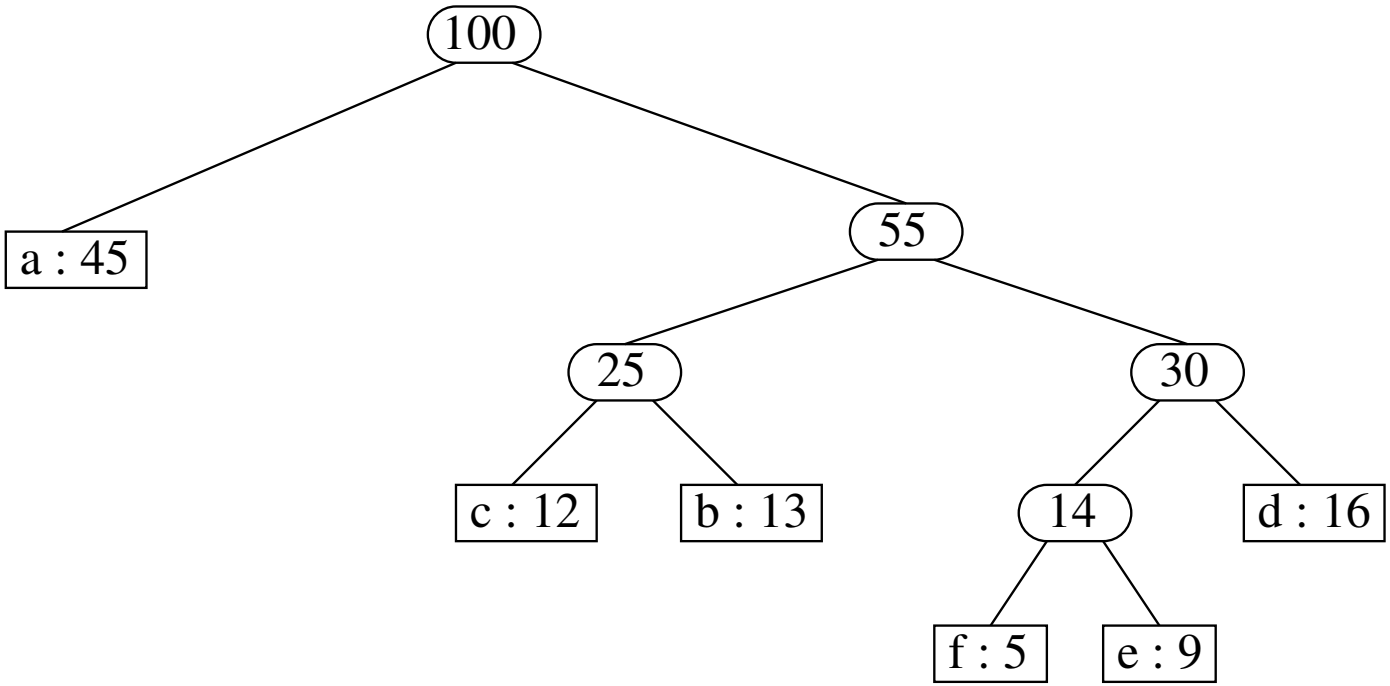


Algoritmo de Huffman: exemplo

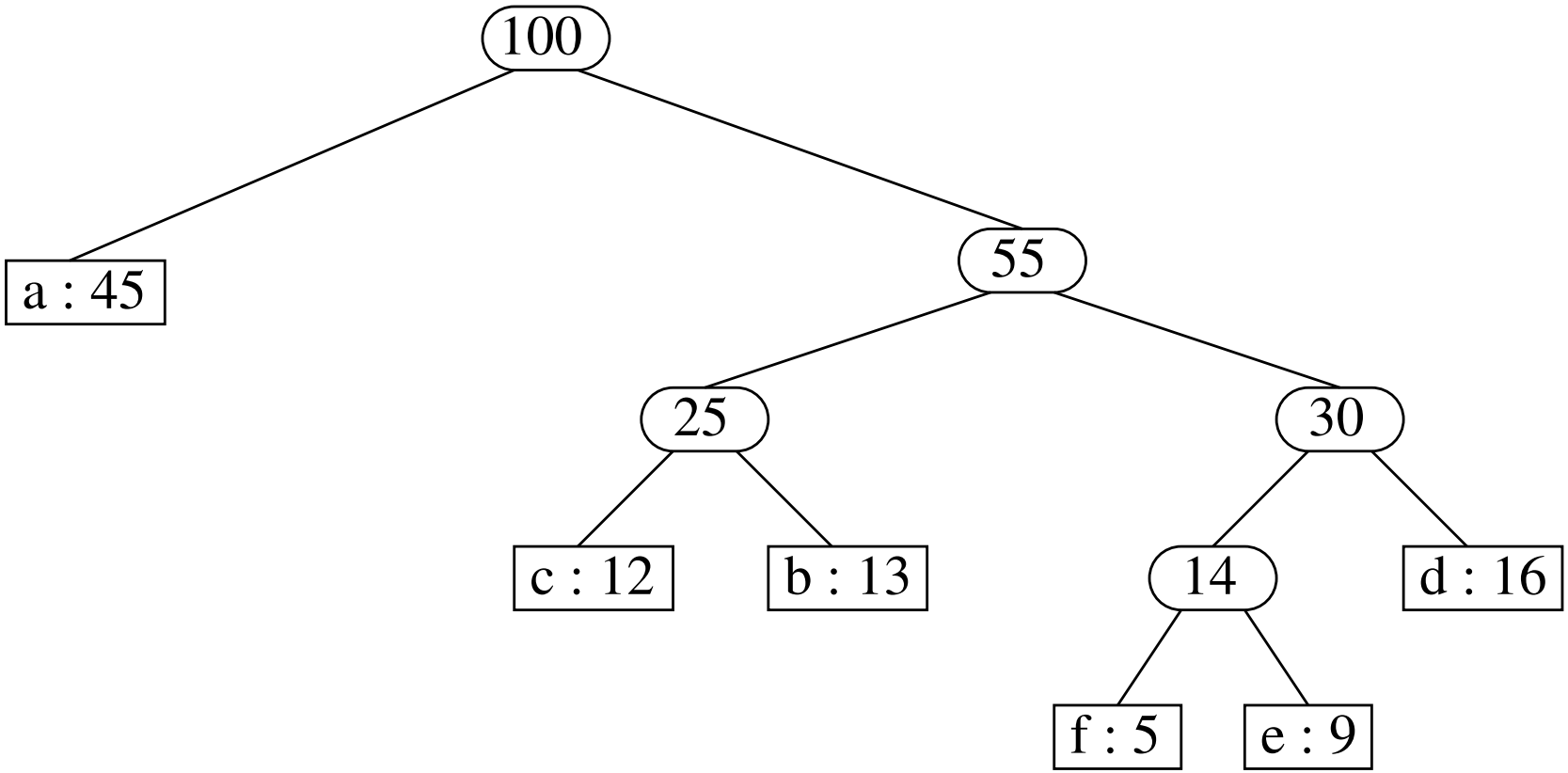


Algoritmo de Huffman: exemplo





Árvore de Huffman



Algoritmo de Huffman

n: número de símbolos em Σ

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 Σ

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 (Σ, f, n)
 1  Q ← BUILD-PQ(Σ, f, n)
 2  para i ← 1 até n − 1 faça
 3      x ← EXTRACT-MIN(Q)
 4      y ← EXTRACT-MIN(Q)
 5      z ← NOVA-CEL()
 6      esq[z] ← x
 7      dir[z] ← y
 8      f[z] ← f[x] + f[y]
 9      INSERT(Q, z, f[z])
10  devolva EXTRACT-MIN(Q)
```

Algoritmo guloso

```
HUFFMAN (Σ, f, n)
 1  Q ← BUILD-PQ(Σ, f, n)
 2  para i ← 1 até n − 1 faça
 3      x ← EXTRACT-MIN(Q)
 4      y ← EXTRACT-MIN(Q)
 5      z ← NOVA-CEL()
 6      esq[z] ← x
 7      dir[z] ← y
 8      f[z] ← f[x] + f[y]
 9      INSERT(Q, z, f[z])
10  devolva EXTRACT-MIN(Q)
```

Consumo de tempo: $\mathrm{O}(n \lg n)$.

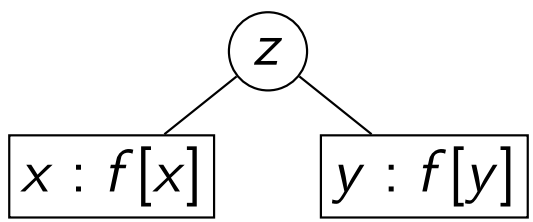
Escolha gulosa de Huffman

Árvore de código para Σ ($|\Sigma| = n \geq 2$): árvore binária cujas folhas são os símbolos de Σ e em que todo nó interno tem 2 filhos.

$d_T(c)$: profundidade de c em T = comprimento do código de c.

Custo: $B(T) = \sum_{c \in \Sigma} f[c]\, d_T(c)$. T é ótima se $B(T)$ é mínimo.

Lema. Sejam x e y símbolos tais que
$f[x] \leq f[y] \leq f[c]$ para todo $c \in \Sigma \setminus \{x, y\}$.
Então existe uma árvore ótima em que x e y são folhas irmãs, filhas de um mesmo nó z:



Escolha gulosa de Huffman: prova

Prova: Seja T uma árvore ótima e seja D a altura de T.

Seja a uma folha de profundidade D e seja b o irmão de a.
b é folha (senão haveria folha com profundidade $> D$).

Troca. Se $d_T(u) = D$ e $f[v] \leq f[u]$, trocar u e v de lugar
altera o custo em $(f[v] - f[u])\,(D - d_T(v)) \leq 0$.

Trocando os nomes de a e b se preciso, podemos supor $x \neq b$.

Passo 1. Se $x \neq a$, troque x e a; vale $f[x] \leq f[a]$.
Agora x e b são folhas irmãs de profundidade D.

Passo 2. Se $b \neq y$, troque y e b; vale $f[y] \leq f[b]$, pois $b \neq x$.

A árvore T^* obtida tem $B(T^*) \leq B(T)$, logo é ótima,
e nela x e y são folhas irmãs, filhas de um mesmo nó z. □

Trocar x e y por z está correto

$\Sigma' := (\Sigma \setminus \{x, y\}) \cup \{z\}$, com $f[z] := f[x] + f[y]$.

Dada árvore T' para Σ', seja T a árvore para Σ obtida trocando a folha z por um nó z com filhos x e y.

Como $d_T(x) = d_T(y) = d_{T'}(z) + 1$, temos
$B(T) = B(T') + f[x] + f[y]$.

Toda árvore para Σ em que x e y são irmãos é obtida assim.

Logo, se T' é ótima para Σ', então T é ótima entre as árvores com x e y irmãos e, pelo Lema, T é ótima para Σ.

Conclusão. Depois de juntar x e y em z, HUFFMAN continua exatamente como HUFFMAN aplicado a Σ', com $n - 1$ símbolos. Por indução em n, HUFFMAN devolve uma árvore ótima.