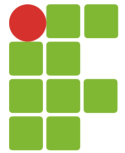
Compressão

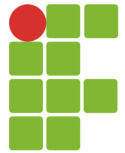
Análise de Algoritmos – Ciência da Computação

Prof. Daniel Saad Nogueira Nunes

Instituto Federal de Brasília,
Câmpus Taguatinga



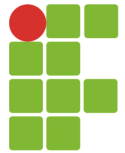
Sumário



Introdução

Compressão de Dados

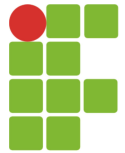
- Em Ciência da Computação, Compressão de Dados envolve o emprego de técnicas para codificar a informação utlizando menos bits que a sua representação original.
- A compressão pode ser tanto com perdas (lossy) ou sem perdas (lossless).
- Geralmente, a compressão de dados envolve um trade-off entre espaço e tempo.
  - Trocamos espaço por tempo de processamento em compressão/descompressão.
  - Trocamos espaço por qualidade...



Introdução

Compressão sem Perdas

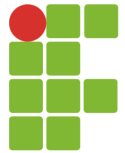
- Crucial quando não queremos perder nenhuma informação.
- Exploram a redundância da informação para comprimi-la.
- Completamente reversível.
- Exemplos:
  - Lempel-Ziv (LZ).
  - Huffman Encoding.
  - Run-Length Encoding.
  - Grammar Compression.



Introdução

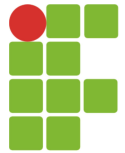
Compressão com Perdas

- A compressão com perdas é utilizada quando podemos permitir uma certa perda, que eventualmente implica em degradação de qualidade.
- Compressão alcançada é maior que a compressão sem perdas.
- Retira-se elementos não essenciais da entrada, de modo que não haja prejuízo por parte do usuário.
- Formatos de arquivos populares que envolvem compressão com perdas:
  - MPEG.
  - JPG.
  - MP3.

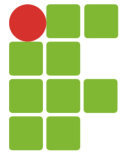


Introdução

- Nos focaremos em compressão sem perdas.
- Compressão com perdas é altamente estudada em processamento de sinais.
  - Engenharia Elétrica e de Computação.

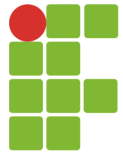


Sumário



Run-Length-Coding

- O Run-Length-Coding se foca em repetições contíguas no texto para alcançar compressão.
- Exemplo: $\mathrm{RLE}(aaaaabbacaacaaaaaaa) = 5a2b1a1c2a1c7a$



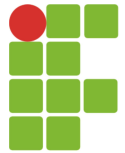
Run-Length-Coding

Algorithm 1: $\mathrm{RLE}(T)$

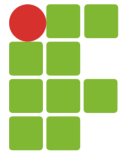
Input: T

Output: $RLE(T)$

```
1 T ← T + $
2 c ← 0
3 i ← 0
4 while i < n do
5 |   c++
6 |   if( T[i] ≠ T[i + 1] )
7 |   |   OUTPUT(c, T[i])
8 |   |   c ← 0
```

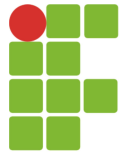


Sumário



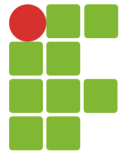
Codificação de Textos

- Suponha que tenhamos um texto com três bilhões de símbolos.
- Suponha ainda que temos apenas 4 símbolos distintos neste texto.
  - $\Sigma = \{a, c, t, g\}$.
- Quantos bits precisamos para representar cada símbolo?



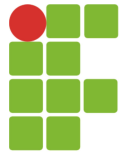
Codificação de Textos

- Suponha que tenhamos um texto com três bilhões de símbolos.
- Suponha ainda que temos apenas 4 símbolos distintos neste texto.
  - $\Sigma = \{a, c, t, g\}$.
- Quantos bits precisamos para representar cada símbolo?
- 2 bits apenas.



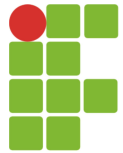
Codificação de Textos

- Suponha que tenhamos um texto com três bilhões de símbolos.
- Suponha ainda que temos apenas 4 símbolos distintos neste texto.
  - $\Sigma = \{a, c, t, g\}$.
- Quantos bits precisamos para representar cada símbolo?
- 2 bits apenas.
- $a = 00$, $c = 01$, $g = 10$ e $t = 11$.



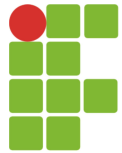
Codificação de Textos

- Suponha que tenhamos um texto com três bilhões de símbolos.
- Suponha ainda que temos apenas 4 símbolos distintos neste texto.
  - $\Sigma = \{a, c, t, g\}$.
- Quantos bits precisamos para representar cada símbolo?
- 2 bits apenas.
- $a = 00$, $c = 01$, $g = 10$ e $t = 11$.
- O texto codificado $01010001000111 0010$ corresponde à qual texto original?



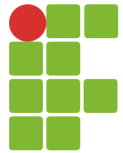
Codificação de Textos

- Usando apenas 2 bits por símbolo, qual o espaço necessário para armazenar o arquivo?
- Se o símbolo a tivesse 3 vezes mais ocorrência que os outros símbolos, teríamos uma outra representação mais vantajosa?



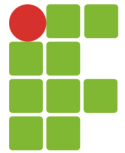
Codificação de Textos

- Usando apenas 2 bits por símbolo, qual o espaço necessário para armazenar o arquivo?
- Se o símbolo a tivesse 3 vezes mais ocorrência que os outros símbolos, teríamos uma outra representação mais vantajosa?
- Sim! $a = 0$, $b = 1$, $c = 10$ e $d = 11$.



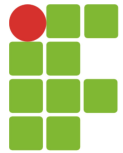
Codificação de Textos

- Usando apenas 2 bits por símbolo, qual o espaço necessário para armazenar o arquivo?
- Se o símbolo a tivesse 3 vezes mais ocorrência que os outros símbolos, teríamos uma outra representação mais vantajosa?
- Sim! $a = 0$, $b = 1$, $c = 10$ e $d = 11$.
- Códigos de comprimento variável (VLC) tem representação mais eficiente dependendo da frequências dos símbolos.



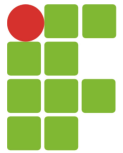
Codificação de Textos

- Usando apenas 2 bits por símbolo, qual o espaço necessário para armazenar o arquivo?
- Se o símbolo a tivesse 3 vezes mais ocorrência que os outros símbolos, teríamos uma outra representação mais vantajosa?
- Sim! $a = 0$, $b = 1$, $c = 10$ e $d = 11$.
- Códigos de comprimento variável (VLC) tem representação mais eficiente dependendo da frequências dos símbolos.
- Qual o problema dessa representação?



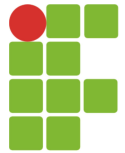
Codificação de Textos

- Usando apenas 2 bits por símbolo, qual o espaço necessário para armazenar o arquivo?
- Se o símbolo a tivesse 3 vezes mais ocorrência que os outros símbolos, teríamos uma outra representação mais vantajosa?
- Sim! $a = 0$, $b = 1$, $c = 10$ e $d = 11$.
- Códigos de comprimento variável (VLC) tem representação mais eficiente dependendo da frequências dos símbolos.
- Qual o problema dessa representação?
- O texto codificado $01010001000111 0010$ corresponde à qual texto original?



Códigos de Comprimento Variável

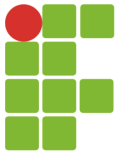
- Alguns Códigos de Comprimento variável possuem a propriedade indesejável de ambiguidade.
- No entanto, uma classe específica de códigos de comprimento variável se vê livre dessa indesejável propriedade.
- Códigos de prefixo.
  - Nenhum símbolo codificado é prefixo de outro símbolo codificado.
  - A decodificação é feita sem ambiguidade.
- Exemplo: $a = 0$, $c = 10$, $g = 110$ e $t = 111$.



Códigos de Comprimento Variável

Código de Huffman

- Huffman em 1952 elaborou o famoso algoritmo que iria ser batizado com seu nome.
- Este algoritmo construía uma estrutura de dados no formato de árvore que gerasse um código de prefixo ótimo!
- Se baseia na propriedade de atribuir códigos menores à símbolos mais frequentes do texto.
- Conhece alguma coisa que faz isso?



Códificação de Huffman

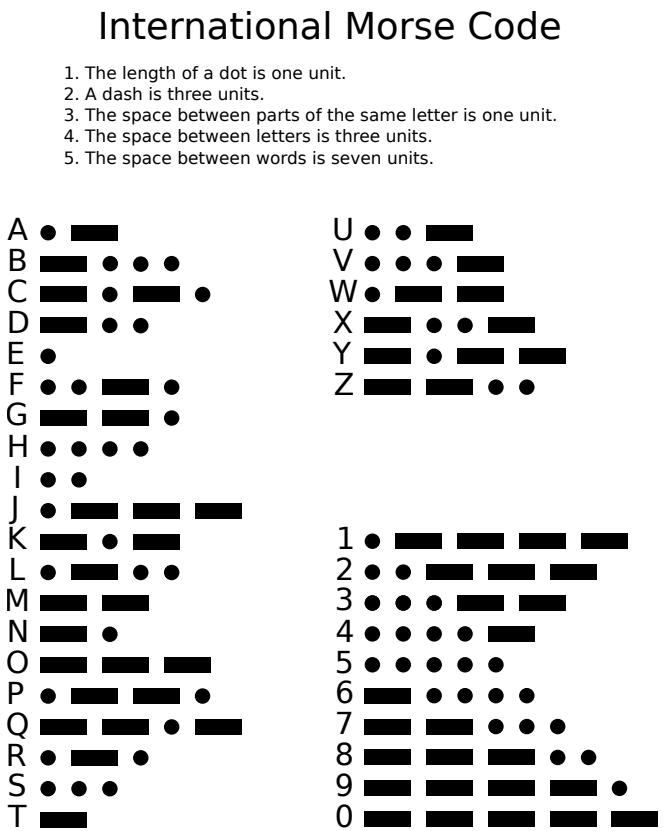
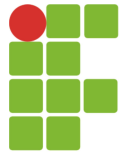
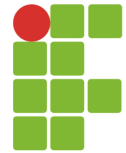


Figura: Código Morse (não é um código de prefixo!).



Código de Huffman



Construção da Árvore de Huffman

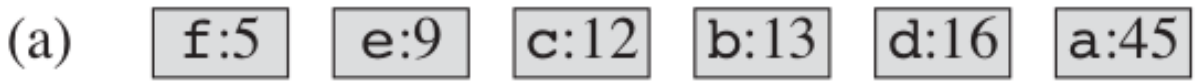


Figura: Construção da Árvore de Huffman.

Construção da Árvore de Huffman

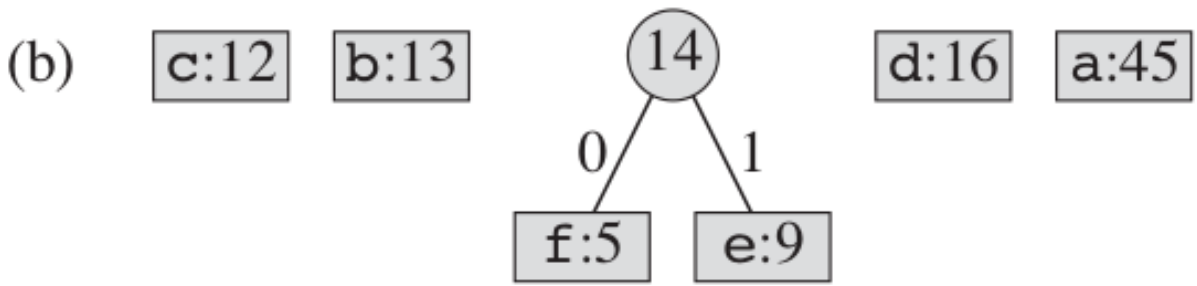


Figura: Construção da Árvore de Huffman.

Construção da Árvore de Huffman

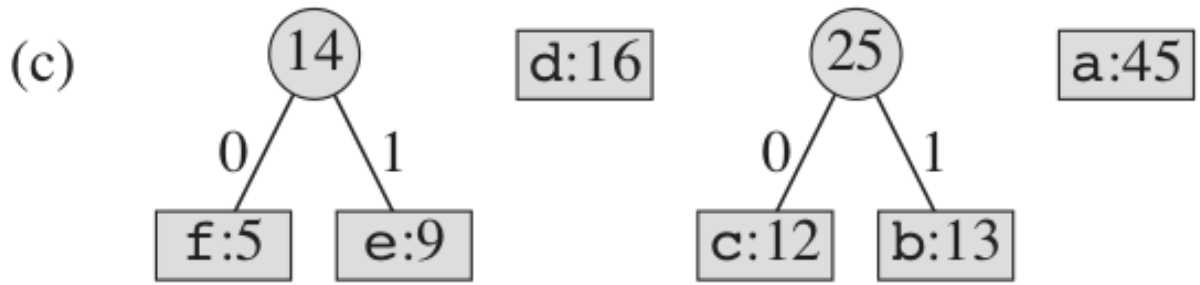
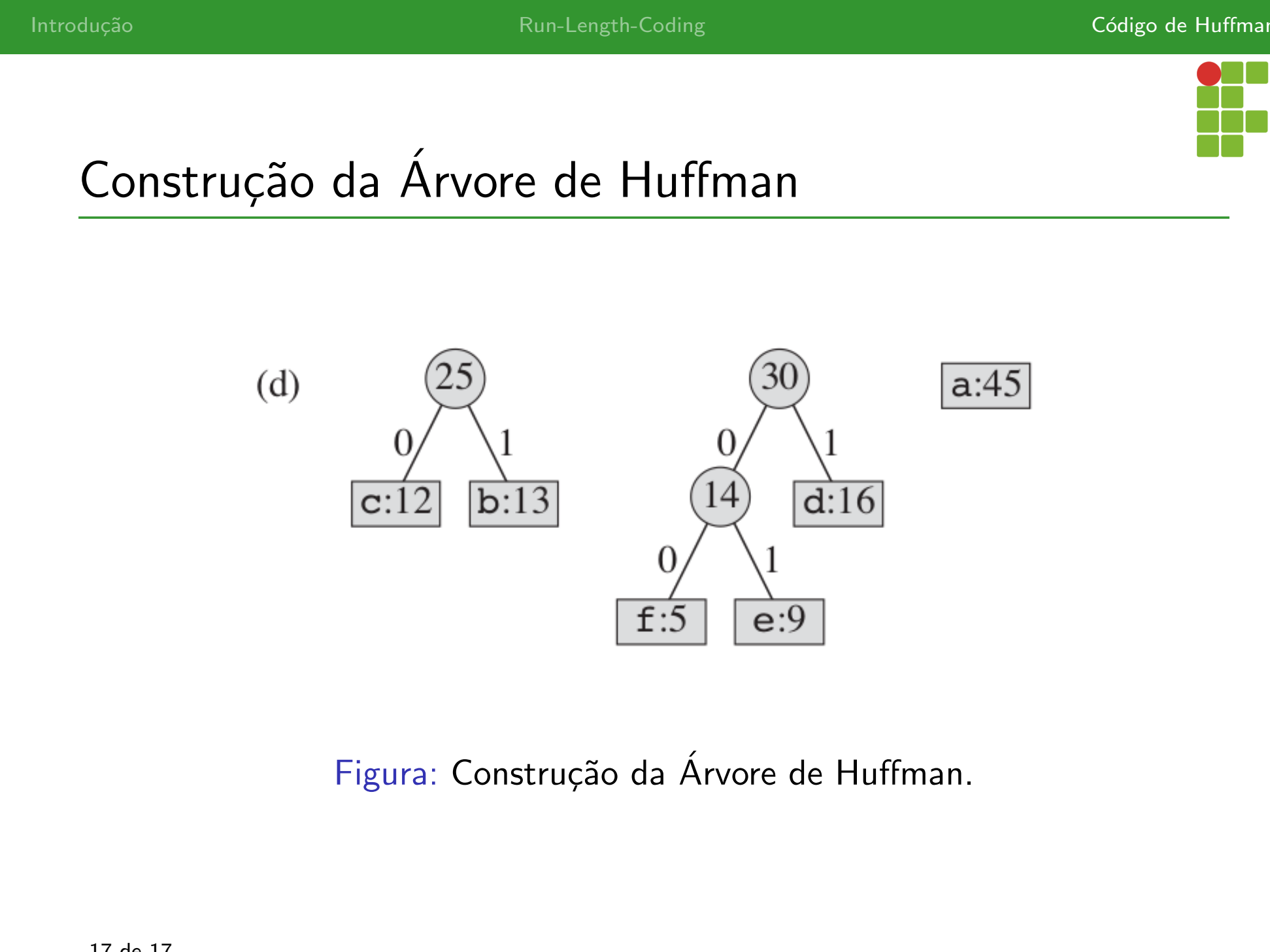


Figura: Construção da Árvore de Huffman.



Construção da Árvore de Huffman

Figura: Construção da Árvore de Huffman.

Construção da Árvore de Huffman

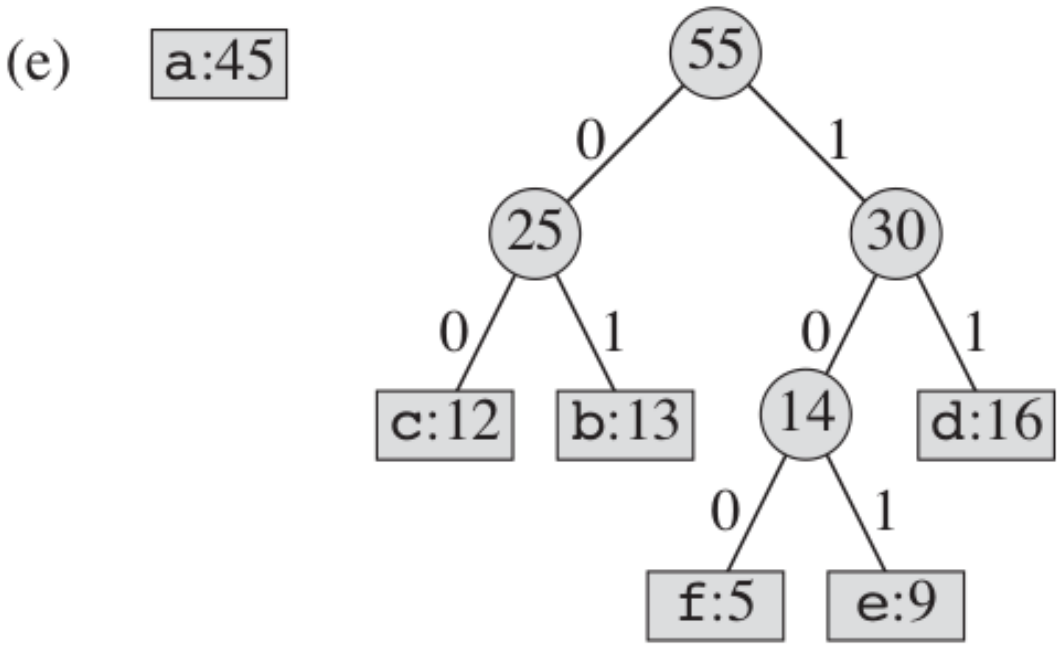


Figura: Construção da Árvore de Huffman.

Construção da Árvore de Huffman

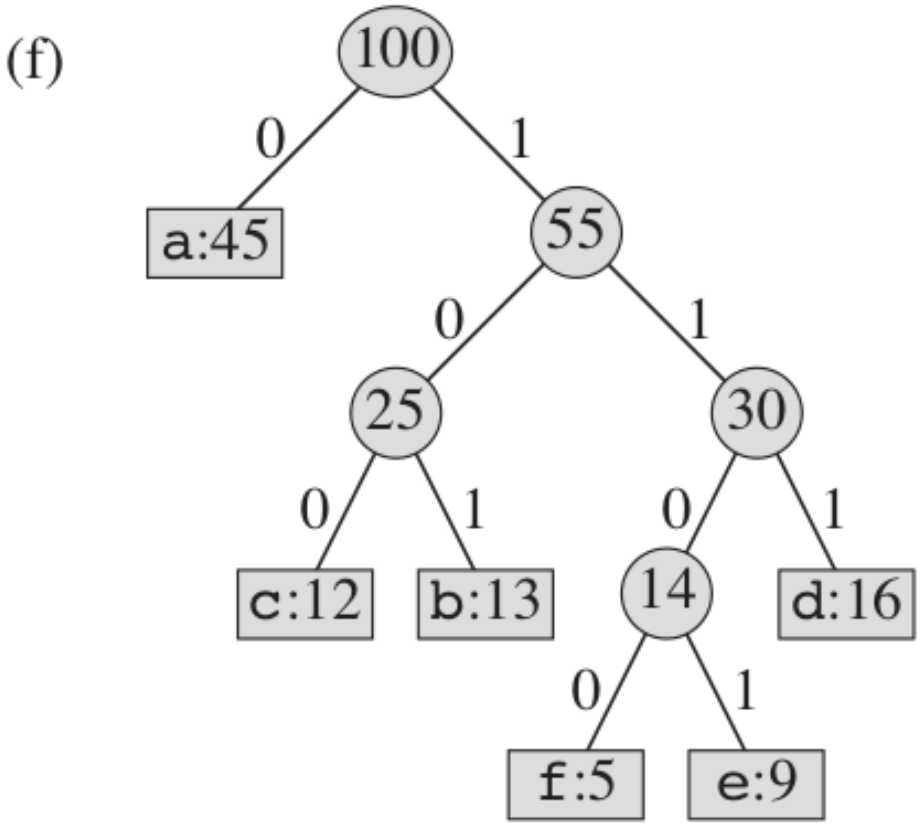


Figura: Construção da Árvore de Huffman.