BCM0504

Natureza da Informação

Códigos Eficientes

Prof. Alexandre Donizeti Alves

Universidade Federal do ABC

Bacharelado em Ciência e Tecnologia

Bacharelado em Ciências e Humanidades

Terceiro Quadrimestre - 2018

Motivação: Exemplo 1

$$H = \log_2(6) = 2{,}58\,bits$$

- Sobram 2 estados
- Opção: código redundante
- Podemos ser mais eficientes?

Código	Estado Do Dado
000	1
001	2
010	3
011	4
100	5
101	5
110	6
111	6

Exemplo 1

$$H = \log_2(6) = 2{,}58\,bits$$

- Para estados 5 e 6, não transmitimos o último bit
- Média de bits transmitidos:
- (4*3+2*2)/6 = 16/6= 2,67

Código	**Estado Do Dado**
000	1
001	2
010	3
011	4
10	5
10	5
11	6
11	6

Exemplo 1

$$H = \log_2(6) = 2{,}58\,bits$$

Código	Estado Do Dado
000	1
001	2
010	3
011	4
10	5
10	5
11	6
11	6

- Média de bits transmitidos:
- (4*3+2*2)/6 = 16/6= 2,67
- Muito próximo de *H*
- *H* é um ***limite inferior***

Exemplo 1

$$H = \log_2(6) = 2{,}58\,bits$$

- Supomos dado de prob. iguais
- Suponha um dado viciado com mais probabilidade para 1 e 2
- Então seria mais vantajoso codificar os estados 1 e 2 com 2 bits e os outros com 3 bits
- A entropia do dado viciado é menor

Exemplo 2

- Imaginemos que lançamos ao ar uma moeda um milhão de vezes e anotamos o resultado: 1 para cara e zero para coroa, assim:

0101101001010110101100110010100110101

- Como a probabilidade de cair cara ou coroa é a mesma, nós temos que a informação de cada jogada é um bit e a **informação total** é um milhão de bits

Exemplo 2

- E se a moeda estivesse alterada para cair cara uma vez só em cada 1000 jogadas? Assim:

  00000100............000000100000.......0001

- Em um milhão de jogadas, a moeda cairia cara umas 1000 vezes

- Seria possível utilizar uma codificação mais enxuta para representar a sequência anterior?

Uma primeira tentativa de codificação

- Para identificar um 1 no meio de 1000000 dígitos, preciso de $\log_2(1000000)= 20$ bits

- Então para codificar os mil uns que aproximadamente há na sequência de 1 milhão de dígitos precisaríamos de:

$$1000 \times 20 = 20000 \text{ bits ou}$$

$$20000/1000000 = \textbf{0,02 bits por dígito}$$

Seria possível uma codificação ainda mais enxuta para codificar a posição dos mil uns?

Uma tentativa mais enxuta de codificação

• Ao invés de codificar a posição absoluta de cada 1, poderíamos identificar cada 1 pela distância de cada 1 para o 1 anterior

1121dígitos 2346 dígitos 990 dígitos 1230 dígitos

000**1**0000....0000000**1**000000.......0000**1**00000......0**1**000000.....0

• Em média a distância de um 1 para o seguinte 1 é de uns mil dígitos

• Supondo que a distância máxima entre dois uns seja 4000 dígitos, distância que pode ser codificada com $\log_2(4000)$= 12 bits

• Neste caso, 12 bits seria o maior número de bits para codificar a distância entre uns, por exemplo 1230 = 001110011001 em binário

•Para codificar os 1000 uns precisaríamos de 1000 x 12 bits= **12000 bits** ou 12000/1000000 = **0,012 bits por dígito**

Exemplo 2

- Portanto, passamos de uma codificação de 1000000 bits para uma com 20000 bits e depois para uma com 12000 bits, conseguindo uma taxa de compressão de

$$\frac{1000000-12000}{1000000}\cdot 100 = 98{,}8\%$$

- É possível uma codificação ainda mais enxuta?

A fórmula de Shannon fornece a quantidade de informação da codificação mais enxuta possível

- Mesmo que não saibamos como comprimir mais a sequência, a fórmula de Shannon nos diz quanta informação existiria na codificação mais enxuta possível
- No nosso caso:

$$H = -\left(\frac{1}{1000}\right)\log_2\left(\frac{1}{1000}\right) - \left(\frac{999}{1000}\right)\log_2\left(\frac{999}{1000}\right) =$$

$$= 0{,}001 \cdot \log_2(1000) + 0{,}999 \cdot \log_2(1000/999) =$$

$$= 0{,}001 \cdot 3{,}32 \cdot \log_{10}(1000) + 0{,}999 \cdot 3{,}32 \cdot \log_{10}(1000/999) =$$

$$= 0{,}0114\, bits$$

Eficiência de um código

- Como a **entropia** H(X) indica o **limite mínimo** para o número médio de bits necessário para codificar um sistema de símbolos, podemos definir a **eficiência** de um código C comparando o seu comprimento médio L(C) por palavra com a entropia

Eficiência $\eta = \frac{H(X)}{L(C)} \approx 1$

esta razão é sempre menor ou igual a 1 de acordo com o Primeiro Teorema de Shannon

- Número entre 0 e 1.
  - Quanto mais próximo de 1, mais eficiente.

Eficiência de um código

- Obtém dividindo a entropia de Shanon H, pelo número de bits por dígito (às vezes se fala bits por palavra)
- No caso da primeira tentativa de codificação:

$$\eta = \frac{H}{\overline{L}} = \frac{0{,}0114}{0{,}02} = 0{,}5704$$

- E no caso da segunda tentativa:

$$\eta = \frac{H}{\overline{L}} = \frac{0{,}0114}{0{,}012} = 0{,}95$$

- Portanto, a segunda tentativa de **codificação** é **mais eficiente**

Codificação de fonte eficiente

- Se fonte tem *n* símbolos possíveis: código de tamanho fixo requer $\lceil \log_2(n) \rceil$ bits por símbolo
  - Informação média por símbolo pode ser menor, se os símbolos têm diferentes probabilidades
  - Logo, é possível codificar uma cadeia de símbolos dessa fonte com menos bits na média
    - Usando código de tamanho variável
      - Menos bits para símbolos mais prováveis
      - Mais bits para símbolos menos prováveis
    - Ex.: código Morse

Codificação de fonte eficiente

- Codificação de fonte com redundância mínima
  - Ex.: considere fonte que gera símbolos que são os conceitos da UFABC
    - Valores possíveis = A, B, C, D e F
    - *Tarefa*: desenvolver sistema para transmitir uma cadeia desses conceitos em um canal de comunicação que só carrega dois valores booleanos (0 ou 1) por segundo
      - Cadeia produzida a uma taxa de um símbolo por segundo

Codificação de fonte eficiente

- Ex.: símbolos são os conceitos da UFABC
  - Para transmitir cada símbolo separadamente, cada um pode ser codificado como uma sequência de bits
    - Usar código ASCII de 7 ou 8 bits leva a perdas
      - Há só 5 símbolos e ASCII pode lidar com 128 ou 256
    - Como há 5 valores, podemos codificar por 3 bits cada
      - Entropia é no máximo $\log_2(5) = 2{,}32$ bits
      - Há mais bits que o necessário...
    - E canal também só consegue processar 2 bits por segundo...

Codificação de fonte eficiente

- Ex.: símbolos são os conceitos da UFABC
  - Usar **codificação em bloco**
    - Agrupar os símbolos em blocos, ex. 3
      - Informação em cada bloco é 3 * 2,32 = 6,97 bits
      - E o bloco pode ser transmitido com 7 bits
        - Há 125 diferentes sequências de 3 conceitos e 128 padrões de códigos disponíveis em 7 bits
        - Precisa também de maneira de indicar “fim” e de que a última transmissão tem só um conceito (ou dois)
        - Ainda são mais bits por segundo do que o canal suporta...

Codificação de fonte eficiente

- Ex.: símbolos são os conceitos da UFABC
  - Olhando a distribuição de probabilidade dos conceitos
  - Ex. curso típico, com bons alunos

A	B	C	D	F
25%	50%	12,5%	10%	2,5%

  - Qual é a informação por símbolo? E a informação média por símbolo?

Codificação de fonte eficiente

- Ex.: símbolos são os conceitos da UFABC

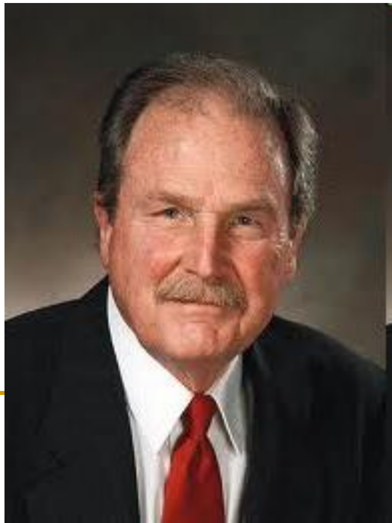
A	B	C	D	F
25%	50%	12,5%	10%	2,5%

Símbolo	Probabilidade	Informação	Contribuição à média
	p	$\log\left(\frac{1}{p}\right)$	$p\log\left(\frac{1}{p}\right)$
A	0.25	2 bits	0.5 bits
B	0.50	1 bit	0.5 bits
C	0.125	3 bits	0.375 bits
D	0.10	3.32 bits	0.332 bits
F	0.025	5.32 bits	0.133 bits
Total	1.00		1.840 bits

Informação por símbolo é 1,84 bits
$\Rightarrow$ Pode ser transmitida no canal, mas como?

Codificação de fonte eficiente

- Há um procedimento para construir códigos de tamanho variável bastante eficientes
  - Requerem na média ou menos que $H + 1$ bits por símbolo
    - Mesmo se H for muito menor que $\log_2(n)$
  - São os **códigos de Huffman**



Código de Huffman

- Maneira de codificar símbolos com diferentes probabilidades, com mínima redundância e sem símbolos especiais $\Rightarrow$ mais compacto
  - **Objetivo**: obter um livro de códigos (**dicionário**) tal que o tamanho médio dos códigos é minimizado
    - Símbolos pouco frequentes com códigos longos e símbolos comuns com códigos pequenos

Código de Huffman

- **Algoritmo:**

1. **Início**:
   - O código parcial de cada símbolo é inicialmente uma string binária vazia
   - Defina um conjunto para cada símbolo
     - Probabilidade do conjunto = probabilidade do símbolo
   - Ex.: conceitos
     - Início: (A = “” p = 0,25) (B = “” p = 0,5) (C = “” p = 0,125)

       (D = “” p = 0,1) (F = “” p = 0,025)

Código de Huffman

- **Algoritmo:**

2. **Teste de iteração (*loop*):**
    - Se há exatamente um conjunto (sua probabilidade será 1), a iteração termina
    - Senão, o dicionário consiste de códigos associados a cada um dos símbolos no conjunto

Código de Huffman

- **Algoritmo:**

3. **Ação de iteração (*loop*)**:
    - Se há dois ou mais conjuntos, pega os dois de menor probabilidade
        - Em empate, escolhe um deles
    - Preceder os códigos para os símbolos em um conjunto com 0 e o outro com 1
    - Define novo conjunto com união dos dois conjuntos processados e probabilidade igual à soma das duas probabilidades
    - Substituir os dois subconjuntos pelo novo
    - Repetir o loop (voltar ao passo 2)

Código de Huffman

- Ex.: conceitos
  - ***Início***:
    - (A = “” p = 0,25) (B = “” p = 0,5) (C = “” p = 0,125) (D = “” p = 0,1) (F = “” p = 0,025)
  - *Iteração 1*:
    - (A = “” p = 0,25) (B = “” p = 0,5) (C = “” p = 0,125) (D = “1” F = “0” p = 0,125)
  - *Iteração 2*:
    - (A = “” p = 0,25) (B = “” p = 0,5) (C = “1” D = “01” F = “00” p = 0,25)
  - *Iteração 3*:
    - (B = “” p = 0,5) (A = “1” C = “01” D = “001” F = “000” p = 0,5)
  - ***Fim***:
    - (B = “1” A = “01” C = “001” D = “0001” F = “0000” p = 1,0)

Código de Huffman

- Ex.: conceitos
  - Dicionário:

Símbolo	Código
A	0 1
B	1
C	0 0 1
D	0 0 0 1
F	0 0 0 0

Código de Huffman

- Desenvolvido por David A. Huffman em 1952
- Atribuição de códigos menores a símbolos mais frequentes (assim como Shannon-Fano)
- Constrói-se uma árvore binária (árvore de Huffmann) recursivamente a partir da junção dos símbolos de menor probabilidade
- Cada aresta da árvore possui um bit (0 ou 1) (assim como Shannon-Fano)

Código de Huffman

- Supõe-se que cada estado gerado é independente dos anteriores, como no Shannon-Fano
  - a distribuição das probabilidades é sempre a mesma
- Eficiência=1 se possível (L(C)=H(X))
  - Igualmente ou mais eficiente do que Shannon-Fano

- Código de Huffman é **ótimo**

Exemplo

- Suponha um alfabeto de 7 letras (A, B, C, D, E, F, G), e uma “linguagem” na qual essas letras possuem as seguintes probabilidades (frequências):
  - A: 0,30
  - E: 0,22
  - C: 0,17
  - D: 0,12
  - B: 0,09
  - F: 0,06
  - G: 0,04

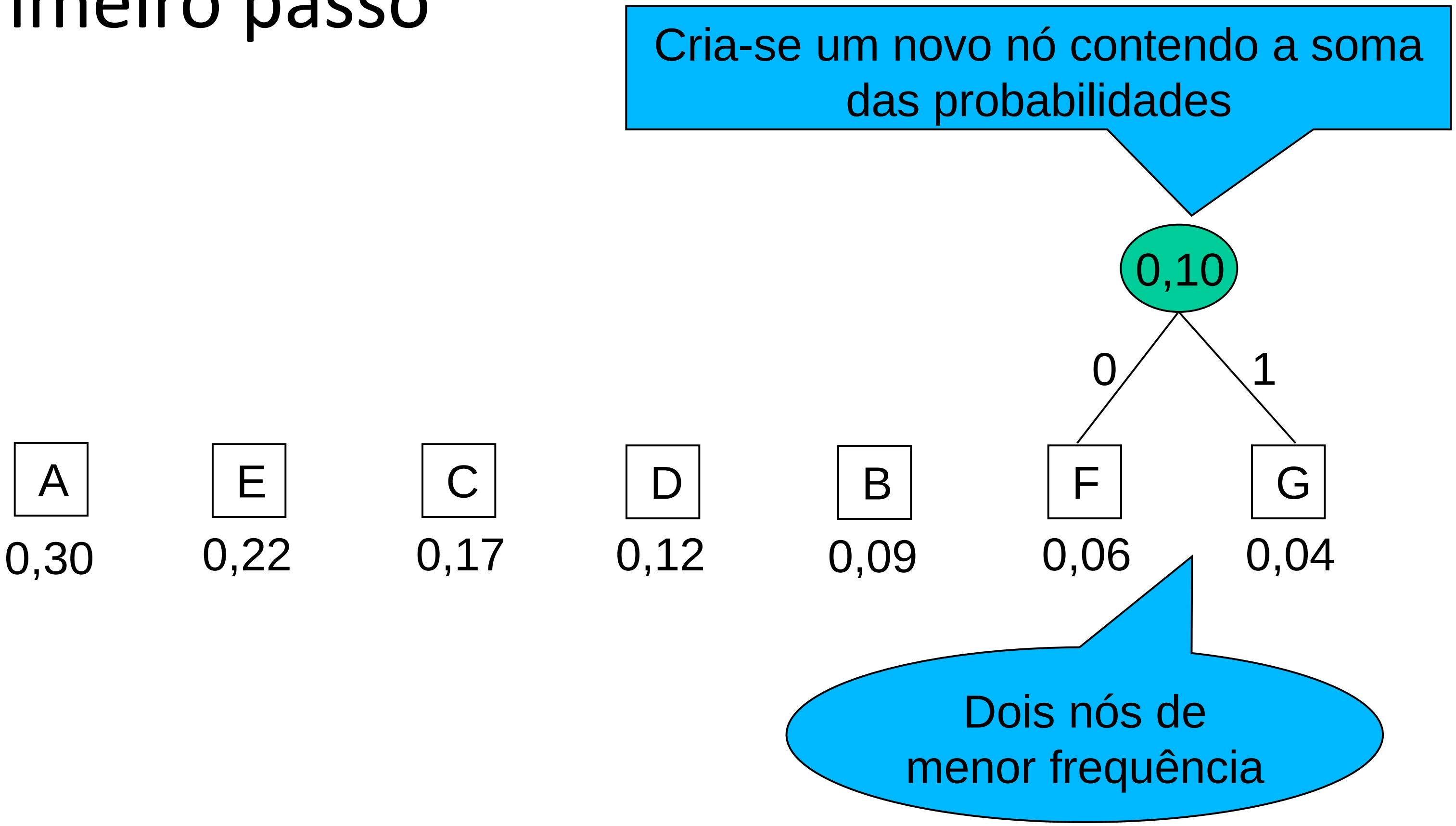
Exemplo

- Início do algoritmo de construção do código
  - Ordena-se os símbolos pelas suas frequências (ordem decrescente)

A	E	C	D	B	F	G
0,30	0,22	0,17	0,12	0,09	0,06	0,04

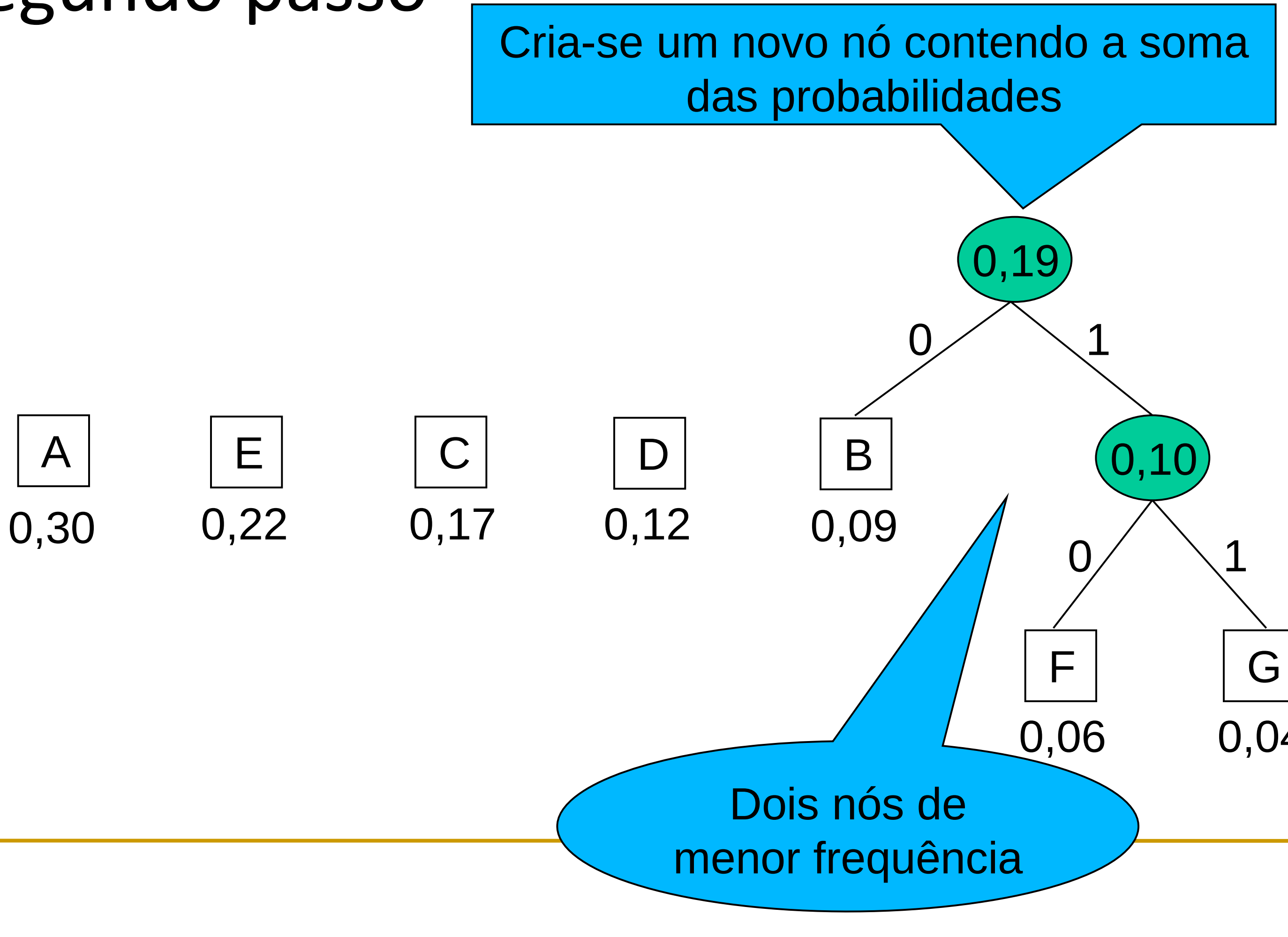
Exemplo

- Primeiro passo



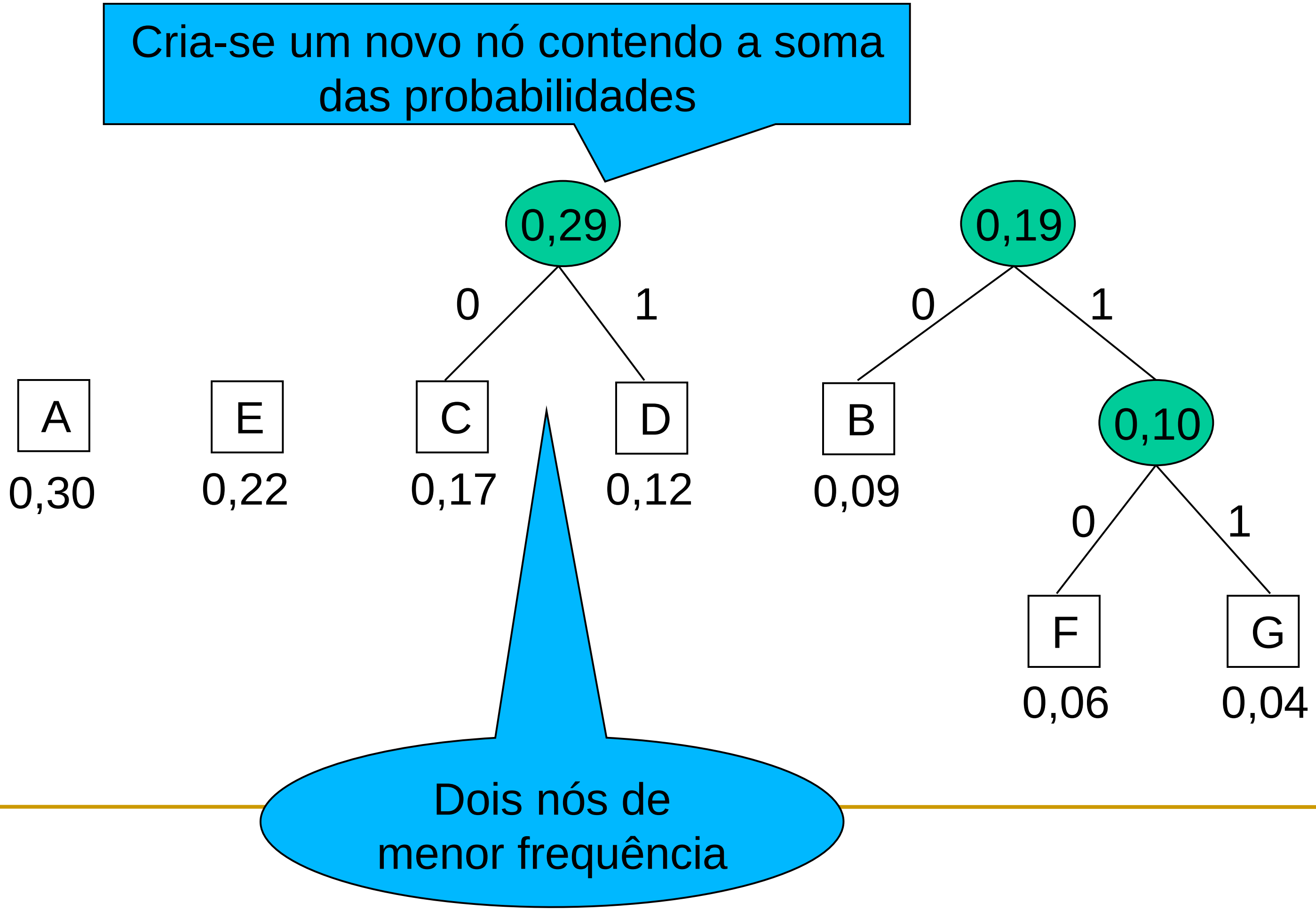
Exemplo

- Segundo passo



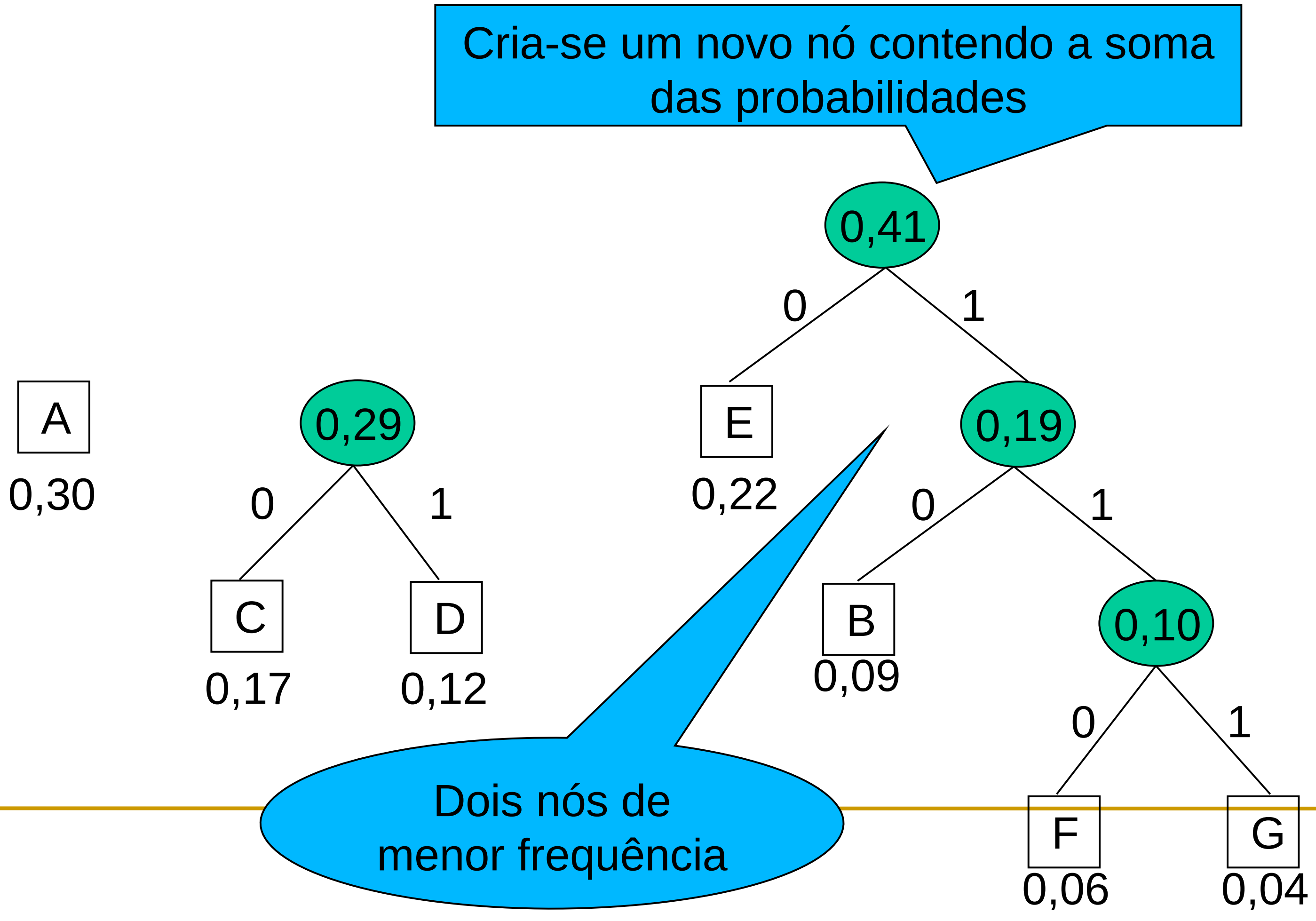
Exemplo

- Terceiro passo



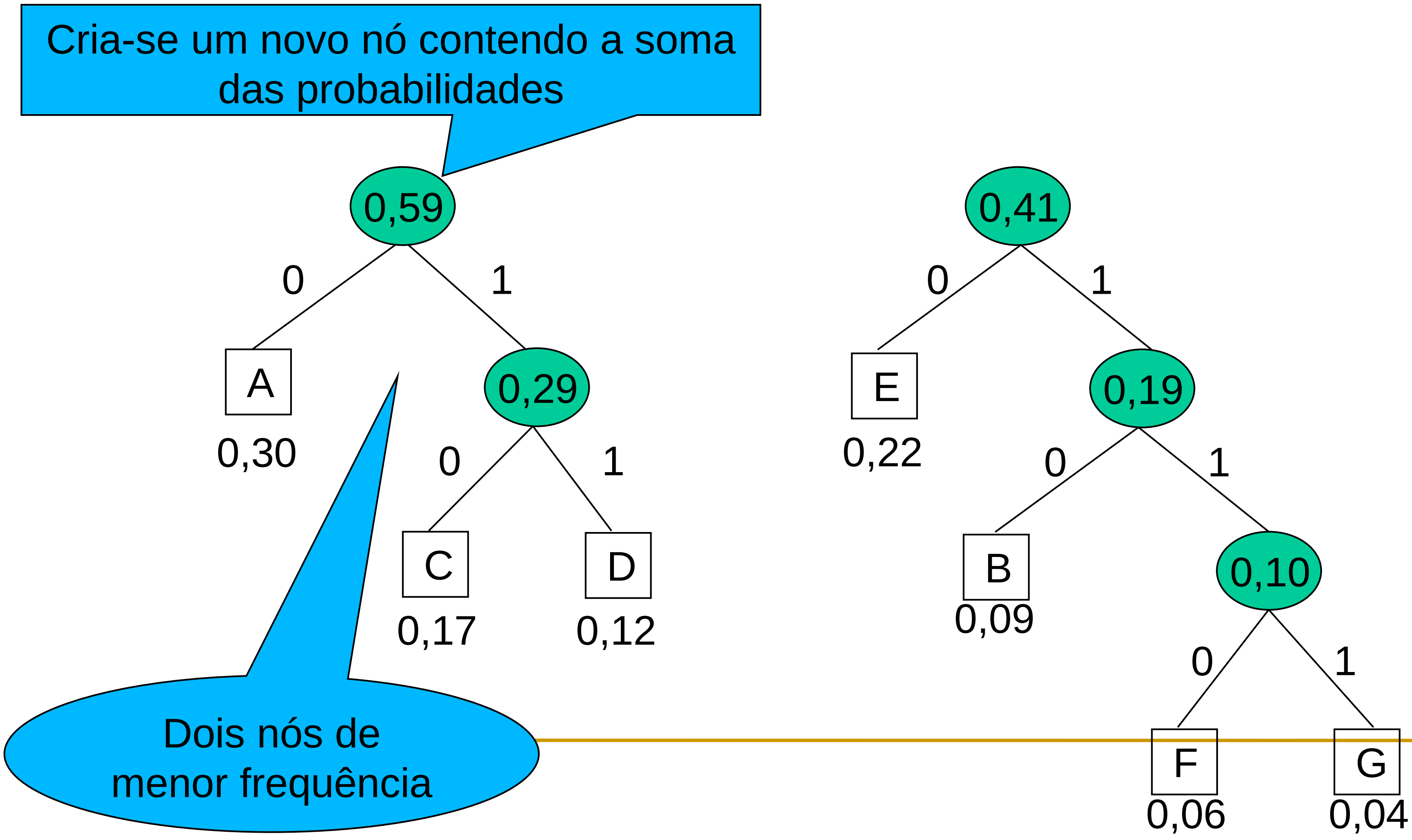
Exemplo

- Quarto passo



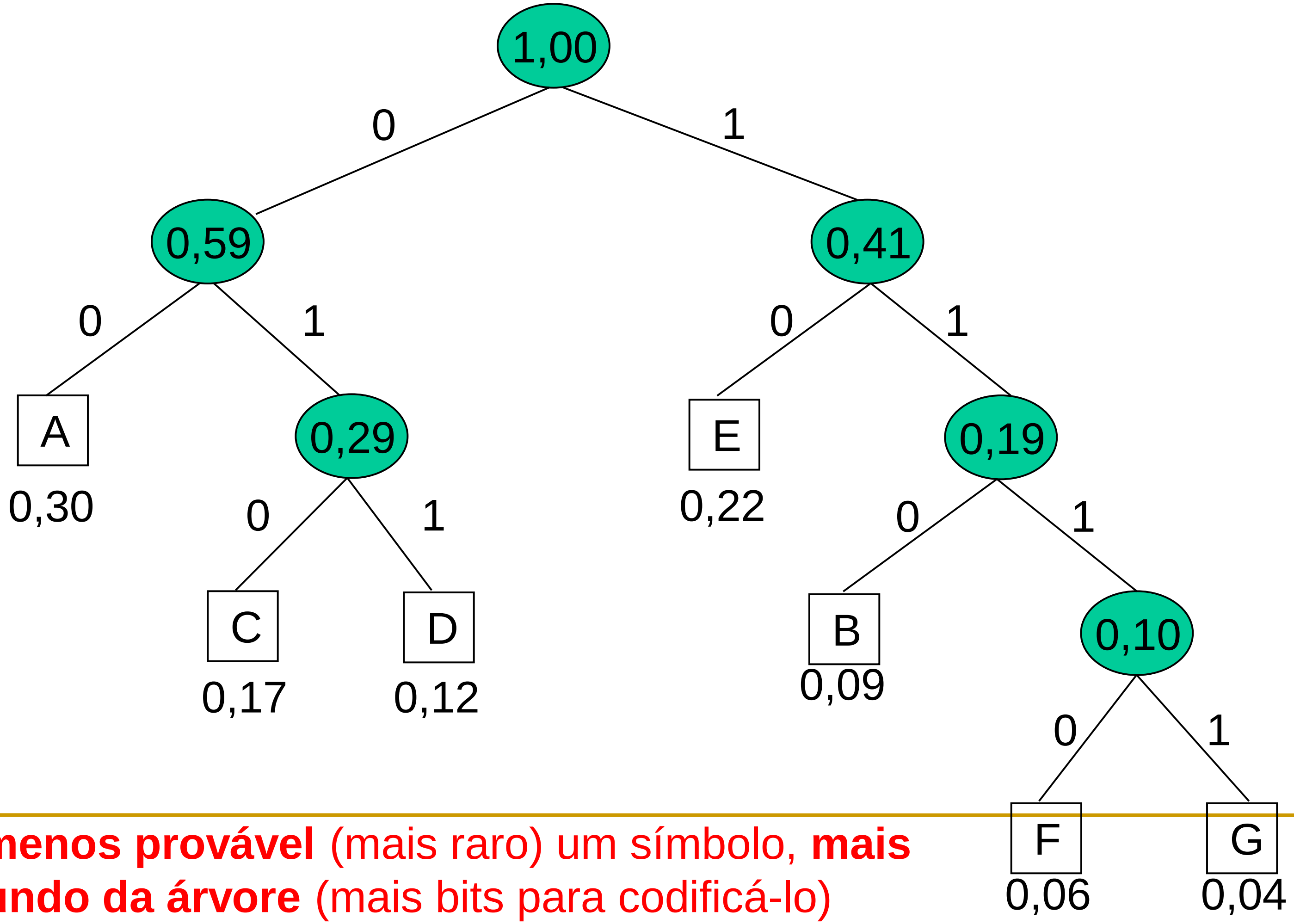
Exemplo

- Quinto passo



Exemplo

- Passo final



Quanto **menos provável** (mais raro) um símbolo, **mais para o fundo da árvore** (mais bits para codificá-lo)

Exemplo

- Para obter o código a partir da árvore, basta percorrer todos os seus caminhos possíveis
  - A: 00
  - B: 110
  - C: 010
  - D: 011
  - E: 10
  - F: 1110
  - G: 1111

Propriedade:
Um símbolo nunca é prefixo de outro

↓

Não há ambiguidade

↓

Não há necessidade de um código especial para separar um símbolo do outro como no código Morse