Processamento Digital de Imagens

Prof. Bogdan Tomoyuki Nassu

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Desafio: compressão de imagens

- Técnicas de compressão de imagens são ***muito*** usadas.
  - Objetivo: reduzir o número de bytes ocupado por uma imagem.
- Considere uma imagem com 1920 x 1080 pixels, com 24bpp.
  - Quantos bytes ela ocupa?

Desafio: compressão de imagens

- Problema: reduzir o número de bytes ocupado por uma imagem.
- Considere uma imagem com 1920 x 1080 pixels, com 24bpp.
  - Quantos bytes ela ocupa?
    - 1920 x 1080 = 2.073.600 pixels.
    - 2.073.600 x 3 = 6.220.800 bytes ≈ 6 Mb por imagem.
- E se for 1 minuto de vídeo, com 30 quadros por segundo?

Desafio: compressão de imagens

- Problema: reduzir o número de bytes ocupado por uma imagem.
- Considere uma imagem com 1920 x 1080 pixels, com 24bpp.
  - Quantos bytes ela ocupa?
    - 1920 x 1080 = 2.073.600 pixels.
    - 2.073.600 x 3 = 6.220.800 bytes ≈ 6 Mb por imagem.
- E se for 1 minuto de vídeo, com 30 quadros por segundo?
  - 11.197.440.000 bytes ≈ 11 Gb!!!

Compressão de imagens

- Formatos de arquivo comuns:
  - Imagens:
    - JPEG (Joint Photographic Experts Group).
    - TIFF (Tagged Image File Format).
    - GIF (Graphics Interchange Format).
    - PNG (Portable Network Graphics).
  - Vídeo:
    - WMV (Windows Media Video).
    - RealVideo.
    - MPEG 1 e 2 (Moving Picture Experts Group).
    - H.264 / MPEG-4 AVC (Advanced Video Coding).

Não confunda formatos de compressão com formatos de arquivo "contâineres". (ex: AVI, Matroska, QuickTime)

- Nota: sempre processamos imagens descomprimidas.
  - As imagens são comprimidas para armazenamento e transmissão.
  - *Codec = coder-decoder.*

Compressão de dados

•Compressão de imagens e vídeos são casos específicos de compressão de dados.

- Teoria da informação:
  - Quantos bits são necessários para representar uma informação?
  - = calcular os limites teóricos para compressão (de qualquer dado).

Compressão *lossy* x *lossless*

- A compressão de imagens pode ou não aceitar perdas.
  - = a imagem descomprimida pode não ser idêntica à original.
- Compressão *lossy*:
  - Pode obter maiores graus de compressão, mas criando artefatos.
  - Dependendo da imagem, as diferenças são imperceptíveis.
- Compressão *lossless*:
  - Importante quando artefatos são *totalmente* indesejáveis.
    - Exemplo: aplicações médicas.
- A perda de informações causada pela compressão *lossy* pode ser medida objetivamente.
  - Medidas de comparação (ex: diferença quadrática).
  - A medida não necessariamente corresponde à qualidade percebida.
    - Características psicovisuais, estruturas, interpretação da cena.

Como comprimir?

Um exemplo ingênuo

- Vamos começar com um exemplo ingênuo, com 2 passos:
  - Reduzir o número de bits por pixel.
  - Reduzir a resolução da imagem.
- Ambos os passos são modos de *quantizar* aspectos da imagem.
- O tópico do redimensionamento foi visto anteriormente.
  - Considere um exemplo no qual a altura e a largura são divididos por 2.
    - Como a imagem é quantizada?
    - O quanto a imagem ficará comprimida?

Um exemplo ingênuo

- Reduzindo o número de bits por pixel:
  - Considere uma imagem com 8 bpp por canal:
    - Valores no intervalo [0,255].
    - Como quantizar?

Um exemplo ingênuo

- Reduzindo o número de bits por pixel:
  - Considere uma imagem com 8 bpp por canal:
    - Valores no intervalo [0,255].
  - Normalizamos os valores, dividindo por 255.
    - Valores no intervalo [0,1].
  - Multiplicamos por 2^n-1 (n = número de bits desejado).
  - Arredondamos para o inteiro mais próximo.
  - Armazenamos cada valor arredondado usando n bits.

Exemplo: 24 bpp

Esta imagem tem 44687 cores diferentes.

Exemplo: 21 bpp

Esta imagem tem 29116 cores diferentes.

Exemplo: 18 bpp

Esta imagem tem 12697 cores diferentes.

Exemplo: 15 bpp

Esta imagem tem 3236 cores diferentes.

Exemplo: 12 bpp

Esta imagem tem 610 cores diferentes.

Exemplo: 9 bpp

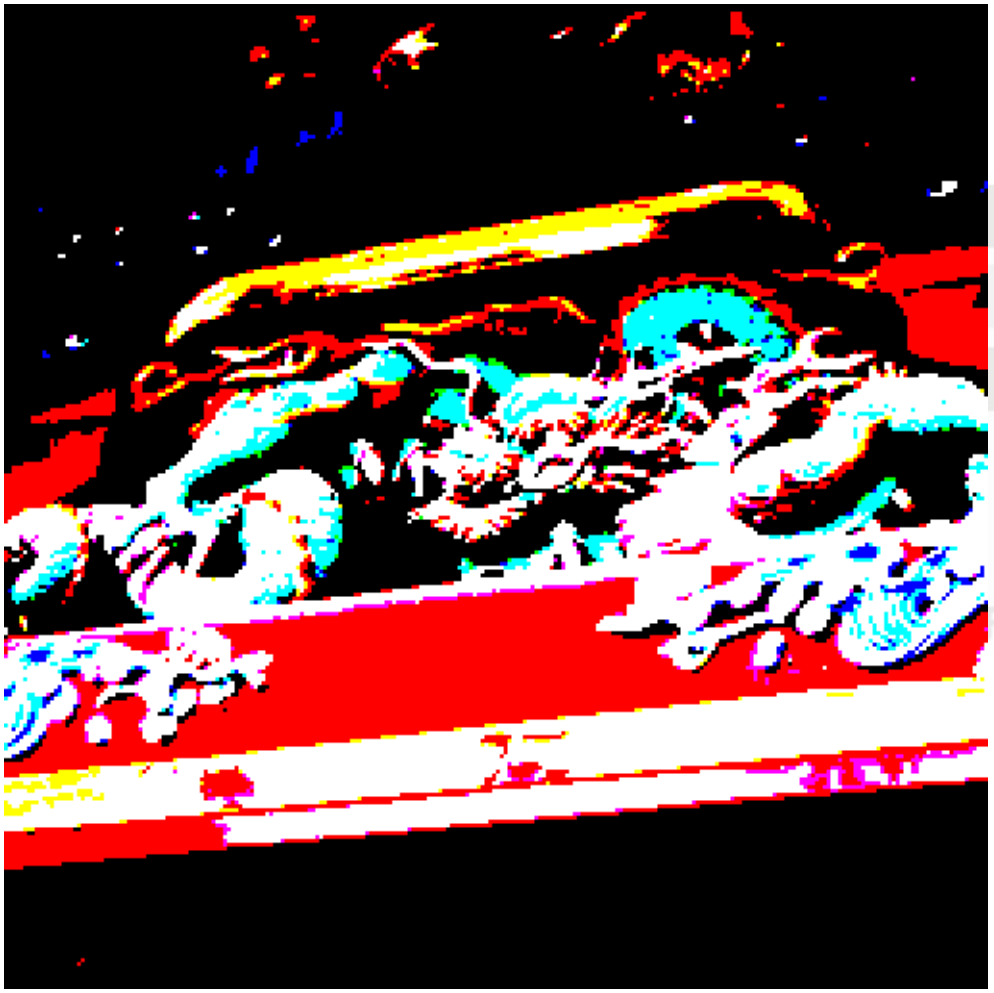
Começaram a surgir “bordas falsas”.

Esta imagem tem 122 cores diferentes.

Exemplo: 6 bpp

Esta imagem tem 29 cores diferentes.

Exemplo: 3 bpp



Esta imagem tem 8 cores diferentes.

Look-up tables (LUT)

- Outra forma de se reduzir o número de bits por pixel: *look-up tables*.
  - Cria uma tabela, com cada entrada sendo associada a uma cor.
  - Exemplo: podemos criar uma tabela com 256 valores (8 bpp), sendo que cada entrada da tabela está associada a uma cor descrita com 24 bpp.
  - A LUT pode ser padronizada ou associada à imagem comprimida.
  - LUTs de cor eram *muito* usadas em computadores antigos.
- Mapeando os valores da imagem original para a comprimida:
  - Mapeamento direto: dois pixels com a mesma cor na imagem original permanecem com a mesma cor na imagem comprimida.
    - Tende a criar regiões com cores “chapadas”.
  - *Dithering*: o valor do pixel é definido com base na sua vizinhança.
    - Usado para aproximar cores inexistentes na LUT.
  - Difusão de erro: acumula e propaga a diferença entre a cor original e a sua correspondente na LUT.

Exemplo: 8 bpp

Paleta padrão,
cor mais próxima.

Esta imagem tem 69
cores diferentes.

Exemplo: 8 bpp

Paleta padrão, *dithering*.

Esta imagem tem 71 cores diferentes.

Exemplo: 8 bpp

Paleta padrão,
difusão de erro.

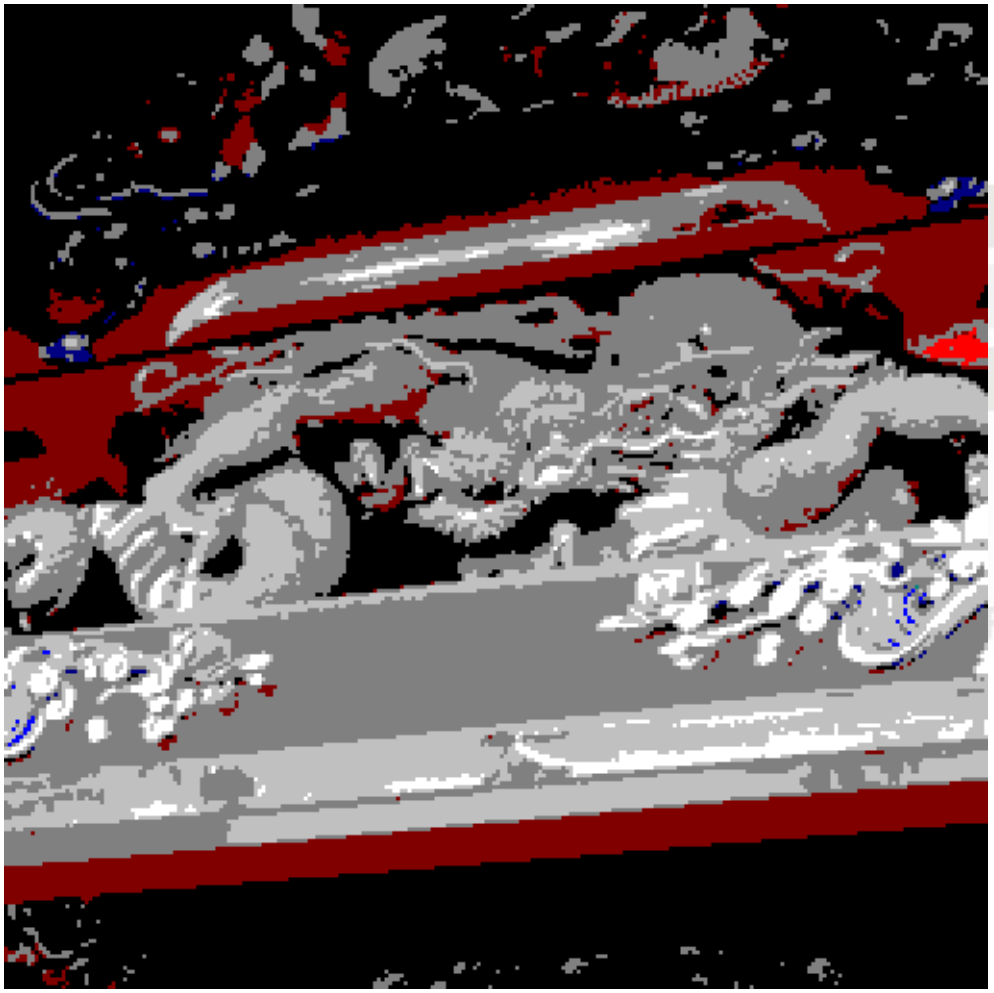
Esta imagem tem 102
cores diferentes.

Exemplo: 8 bpp

Paleta otimizada,
cor mais próxima.

Esta imagem tem 220
cores diferentes.

Exemplo: 4 bpp



Paleta padrão,
cor mais próxima.

Esta imagem tem 9
cores diferentes.

Exemplo: 4 bpp

Paleta padrão, *dithering*.

Esta imagem tem 16 cores diferentes.

Exemplo: 4 bpp

Paleta padrão,
difusão de erro.

Esta imagem tem 16
cores diferentes.

Exemplo: 4 bpp

Paleta otimizada,
cor mais próxima.

Esta imagem tem 16
cores diferentes.

Comparação: o quanto ganhamos

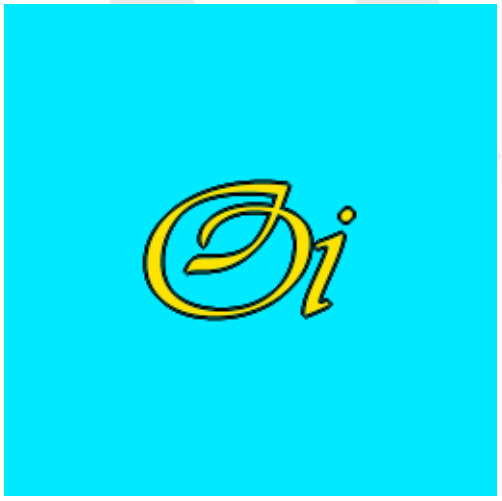
- Exemplo: 1920 x 1080 pixels, 24 bpp.

Original	6.2 Mb
Redução para 960 x 540	1.6 Mb
Redução para 21 bpp	5.4 Mb
Redução para 18 bpp	4.7 Mb
Redução para 15 bpp	3.9 Mb
Redução para 12 bpp	3.1 Mb
Redução para 9 bpp	2.3 Mb
Redução para 8 bpp	2.1 Mb
Redução para 6 bpp	1.6 Mb
Redução para 4 bpp	1 Mb

Note que a relação entre número de bytes e qualidade não é linear e direta!

Como comprimir?

- Fato: existem redundâncias.
  - Mais bits do que o necessário para representar as informações.
    - Inclui também casos onde alguns dados são mais comuns que outros.
  - Redundância espacial ou temporal.
  - Informações irrelevantes ou invisíveis.



65536 pixels.
207 cores diferentes.

Alguns valores são muito mais comuns que outros.

Precisamos mesmo de 24 bpp?

JPEG

- Para entender como a compressão funciona na prática, vamos analisar um dos formatos mais populares, o JPEG.
  - JPEG = Joint Photographic Experts Group.
- O JPEG foi criado especialmente para fotografias.
  - Compressão *lossy*.
  - Artefatos visíveis em imagens com bordas bem defindas (ex: desenhos).
- O JPEG tem diversas variações e alternativas, mas nem todas são universalmente suportadas.
  - Veremos aqui os passos de uma implementação comum.
    - **Importante**: alguns dos passos serão desconsiderados!!!

YCbCr

- Primeiro passo: conversão para o espaço de cores YCbCr.
  - Separa dados de intensidade (Y) e de cor (Cb e Cr).
- Conceito: variações de intensidade são muito mais visíveis do que variações de cor.
  - O JPEG explora isso sub-amostrando os canais Cb e Cr.
  - Normalmente, usa-se um fator de escala 1/2 ou 1/3.
  - Vejamos alguns exemplos!
- Este passo às vezes é ignorado.
  - Implementação mais simples.
  - Qualidade melhor.
  - Menos compressão.

CbCr sem sub-amostragem

CbCr com sub-amostragem (x2)

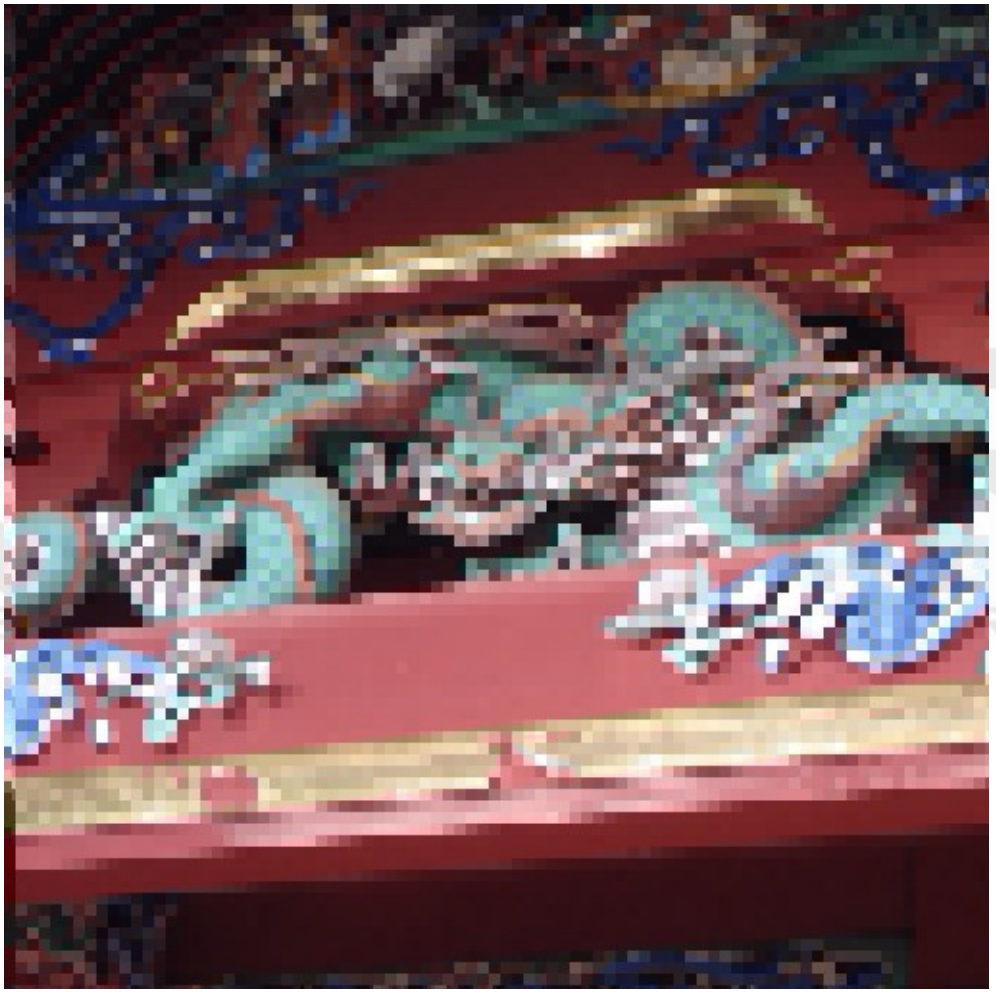
CbCr com sub-amostragem (x3)

CbCr com sub-amostragem (x4)

CbCr com sub-amostragem (x8)

Y com sub-amostragem (x2)

Y com sub-amostragem (x3)



CbCr sem sub-amostragem

CbCr com sub-amostragem (x2)

CbCr com sub-amostragem (x3)

CbCr com sub-amostragem (x4)

CbCr com sub-amostragem (x8)

Y com sub-amostragem (x2)

Y com sub-amostragem (x3)

DCT

- Segundo passo: a imagem é “fatiada” em blocos de 8x8 pixels.
  - Se o número de blocos não for inteiro, o último bloco é “preenchido”.
    - Lembre-se das técnicas para tratamento de margens.
- Cada bloco 8x8 é transformado em outro bloco 8x8 aplicando-se a transformada discreta do cosseno (DCT).
  - Assim como a DFT, é uma transformada para o domínio da frequência.

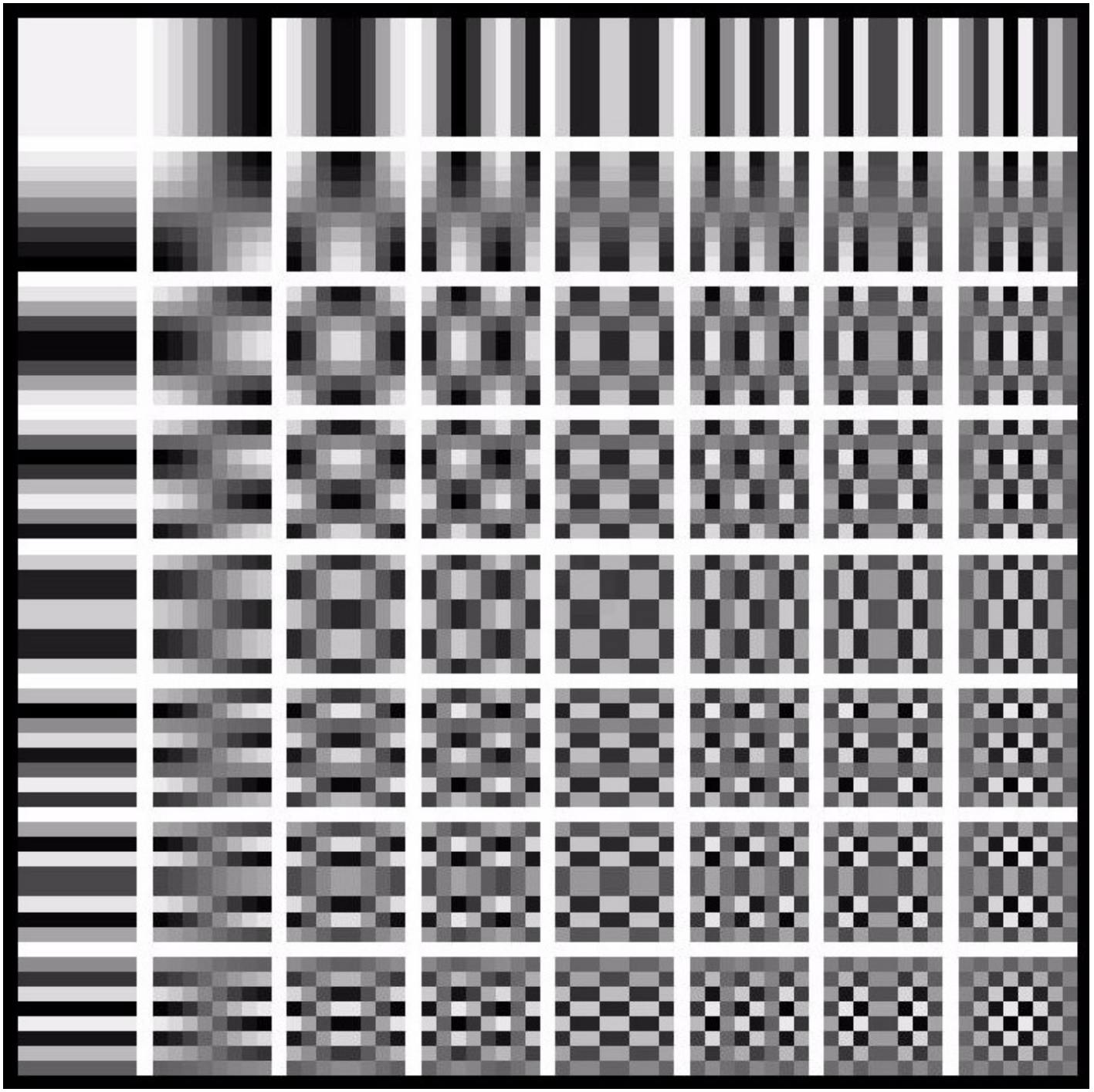
$$G(k,l)=\frac{\alpha(k)\alpha(l)}{4}\sum_{y=0}^{7}\sum_{x=0}^{7}[f(x,y)-128]\cos\left[\frac{(2x+1)\pi k}{16}\right]\cos\left[\frac{(2y+1)\pi l}{16}\right]$$

$$\alpha(k)=\frac{1}{\sqrt{2}}\ se\ k=0\,;1\ do\ contrário$$

DCT

•Aplicar a DCT é equivalente a calcular, para cada bloco 8x8, a resposta a 64 filtros.

- Os kernels dos 64 filtros podem ser pré computados.
- O kernel de cada filtro tem exatamente 8x8 pixels.
  - = cada filtro produz somente 1 resposta para cada bloco 8x8.
- As respostas também são organizadas em blocos 8x8.
  - = a entrada da DCT é um bloco com 8x8 pixels, a saída é um bloco com 8x8 respostas de filtros.



Exemplo: bloco de 8x8 pixels

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

8bpp.

Exemplo: DCT

-415	-29	-62	25	55	-20	-1	3
7	-21	-62	9	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	1	-4	1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	-1	0	-1

Arredondando.

Quantizando o espectro

- Para cada bloco 8x8, quantizamos as 64 saídas da DCT.
  - Ideia: dividir cada saída por um número, arredondar para inteiro.
  - O JPEG só introduz perdas neste passo e na subamostragem das cores.
  - Nota: compressão com perdas normalmente envolve quantização.
    - = mapear *n* valores para *m* valores, com $m < n$.
    - = aumenta as redundâncias!
- As frequências mais altas são divididas por números maiores.
  - Frequências mais altas = mais à direita e abaixo.
  - Diferenças pequenas são mais perceptíveis em regiões mais “limpas”.
  - Efeito: frequências mais altas ficarão com valores menores.
    - Frequentemente, zero!
- Existem tabelas padronizadas para os divisores de cada frequência.
  - Existem também várias tabelas usadas por aplicações e fabricantes.
  - As tabelas podem ser adicionadas ao cabeçalho do arquivo final.

Exemplo: tabela padronizada

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Exemplo: bloco 8x8 quantizado

-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Comprimindo o bloco

- Após a quantização, os valores do bloco são ordenados seguindo um padrão zigue-zague, gerando um vetor de 64 valores.
  - O vetor é comprimido usando *run length encoding* e códigos de Huffman.
    - Ambos são esquemas para compressão sem perdas.
- Nota: o JPEG inclui outros passos, mas vamos nos focar apenas nesses últimos dois.
  - Além disso, vamos estudá-los de forma geral.
    - Foco: *conceitos,* não particularidades introduzidas pelo JPEG.

Run length encoding (RLE)

- Técnica simples para reduzir redundâncias espaciais.
  - Uso clássico: fax.
- Ideia: substituir blocos de valores repetidos por 2 valores:
  - Número de repetições.
  - Valor a ser repetido.
- Exemplo:
  - Sequência: AAAAABCCCABBCCCCCCCCCCCCCAA
  - Codificada: 5A 1B 3C 1A 2B 12C 2A
- Se existem poucas repetições, a RLE pode *expandir* os dados!
  - Funciona bem aplicada diretamente a imagens binárias e desenhos.
  - O JPEG usa RLE para comprimir blocos de 0s, incluindo uma sequência especial 00 (0 repetições de 0) para o bloco final.

Códigos de Huffman

- Técnica para codificação com tamanho variável.
  - = número de bits diferente para cada símbolo.
  - Serve para dados genéricos.
  - Cada sequência de N bits é substituída por outra sequência.
    - Mapeamento feito com LUT.
- Ideia: símbolos que aparecem mais vezes devem usar menos bits.
  - Por outro lado, símbolos raros podem ficar com mais bits do que tinham originalmente!

Códigos de Huffman

- Passo 1: coloca os símbolos em ordem de probabilidade.
- Exemplo:
  - Original: BAFDBCAFAADFAAFDAEBAFDF
  - Ordenando:

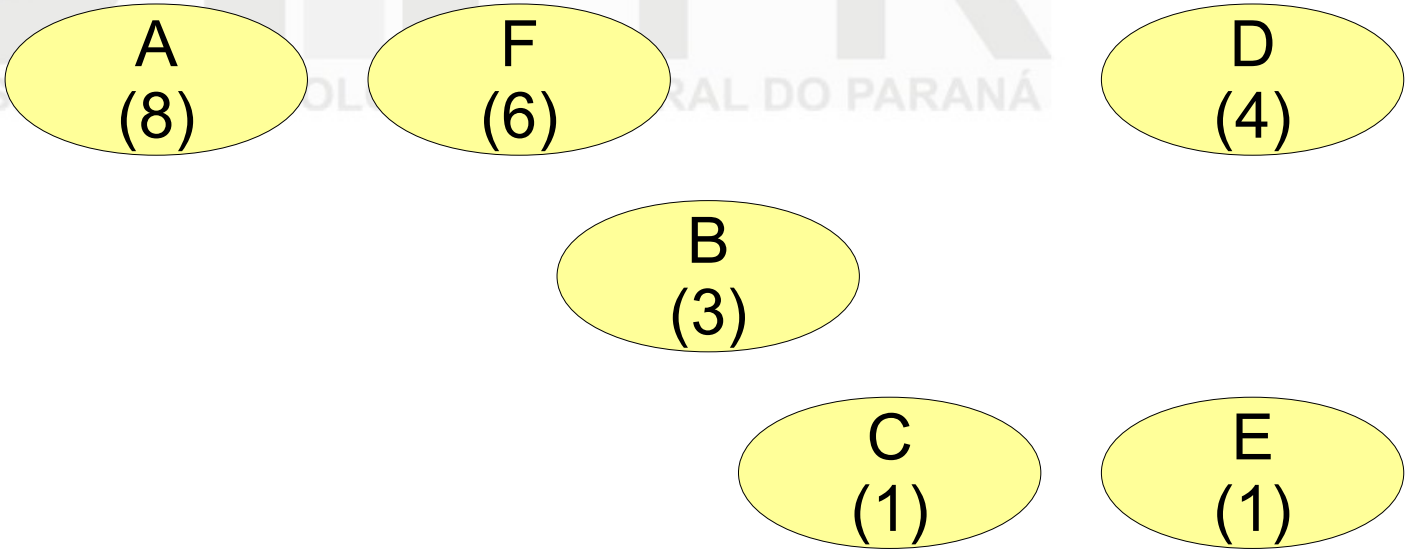
Símbolo	Ocorrências
A	8
F	6
D	4
B	3
C	1
E	1

Códigos de Huffman

- Passo 2: monta uma árvore binária, agrupando os nodos 2 a 2.
  - Árvore desbalanceada!
  - As folhas são os símbolos originais.
  - O número de ocorrências de um nodo é igual à soma dos números de ocorrências dos seus filhos.

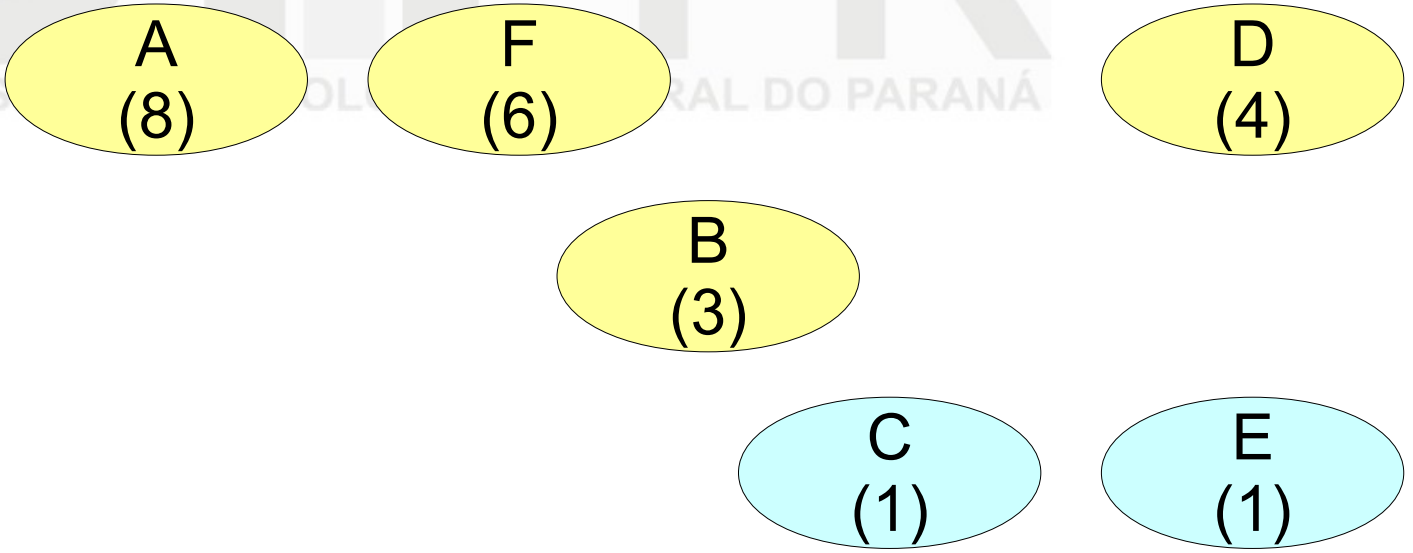
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



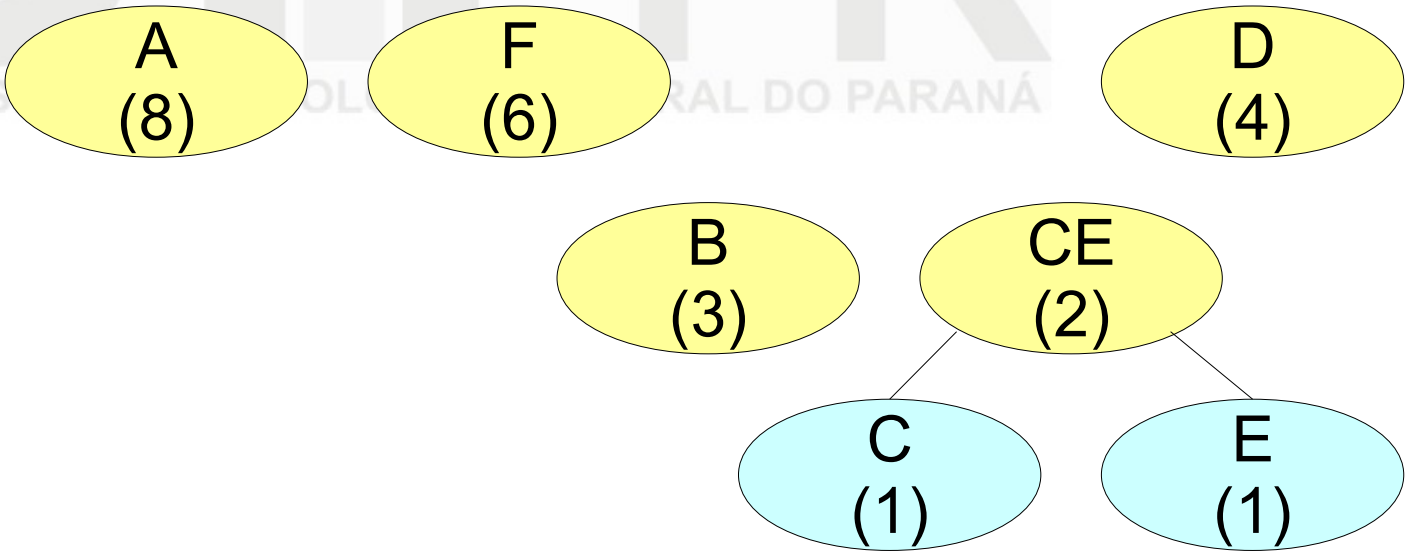
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



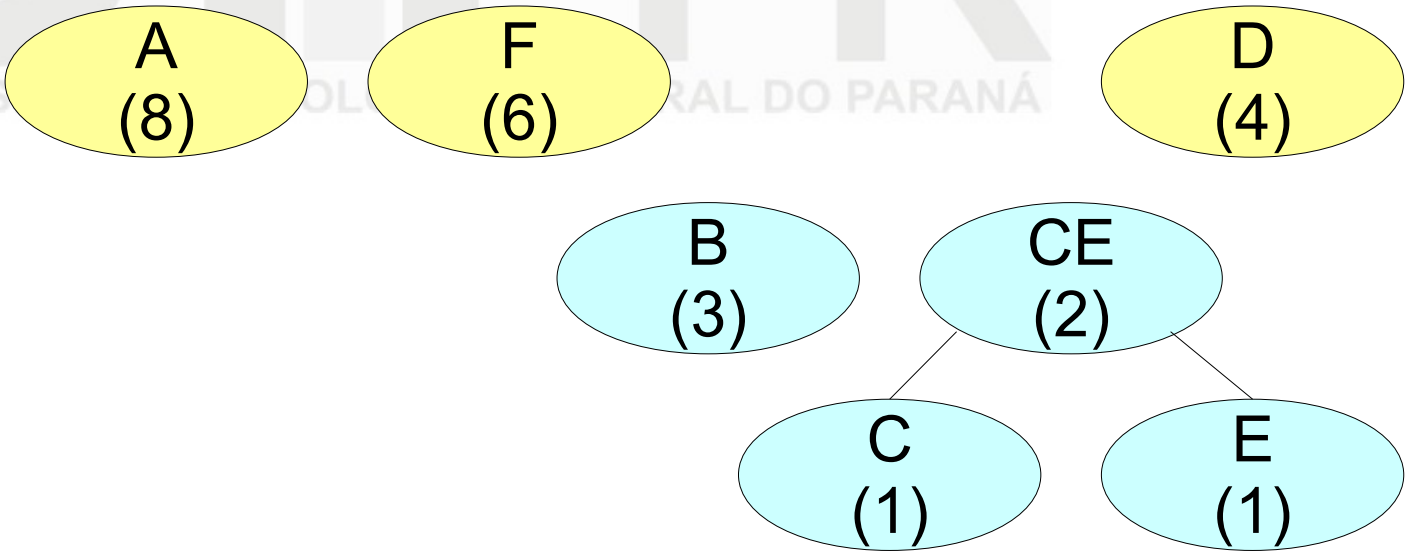
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



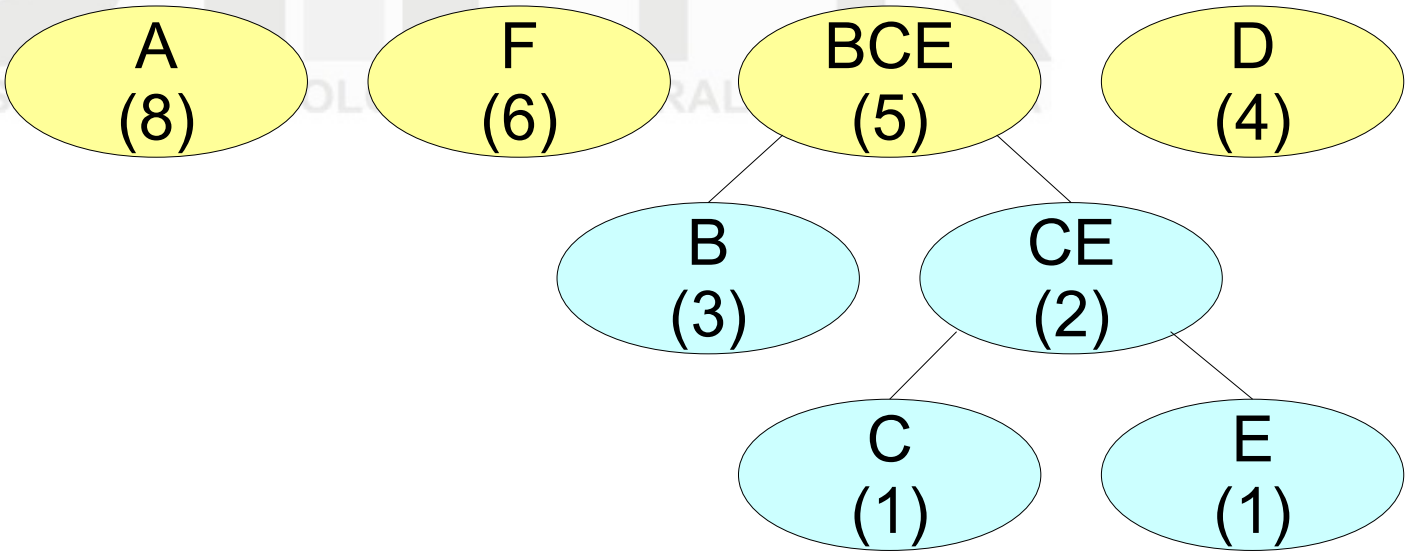
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



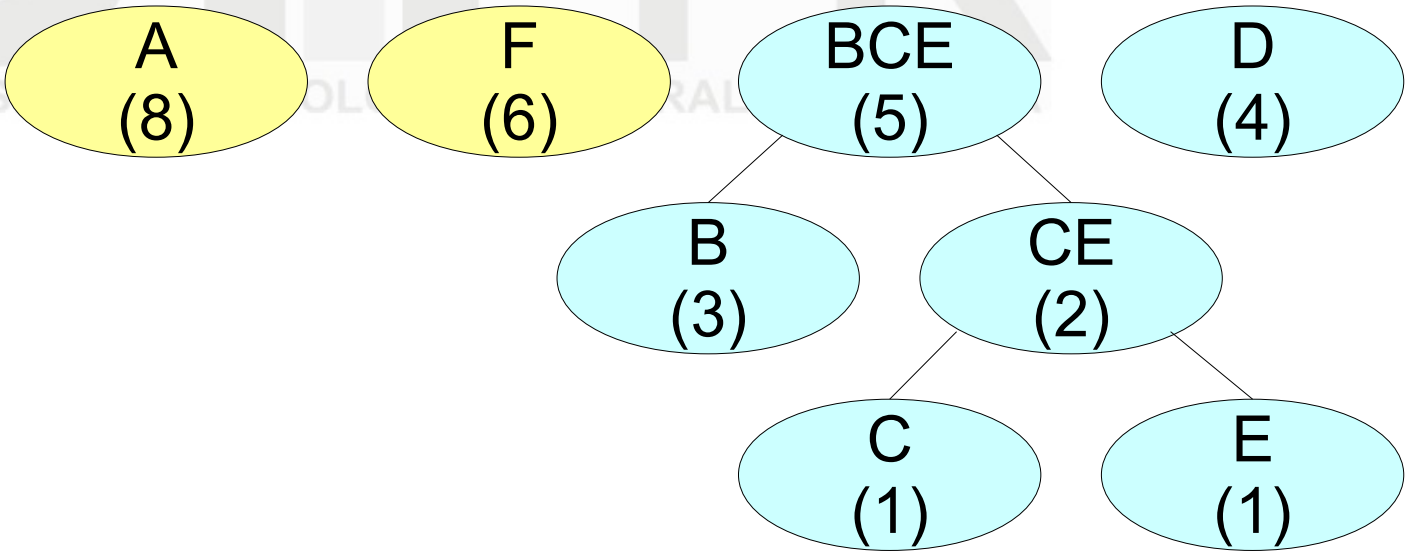
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



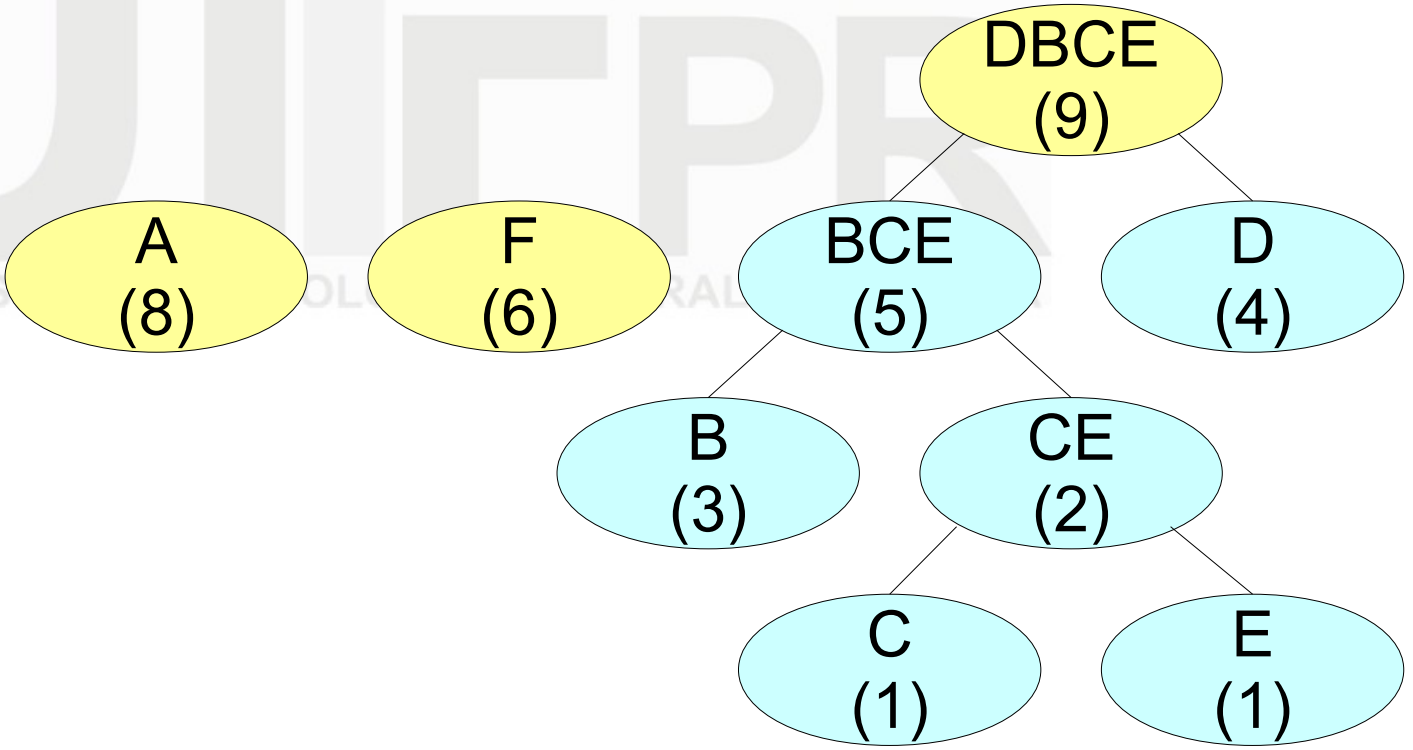
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



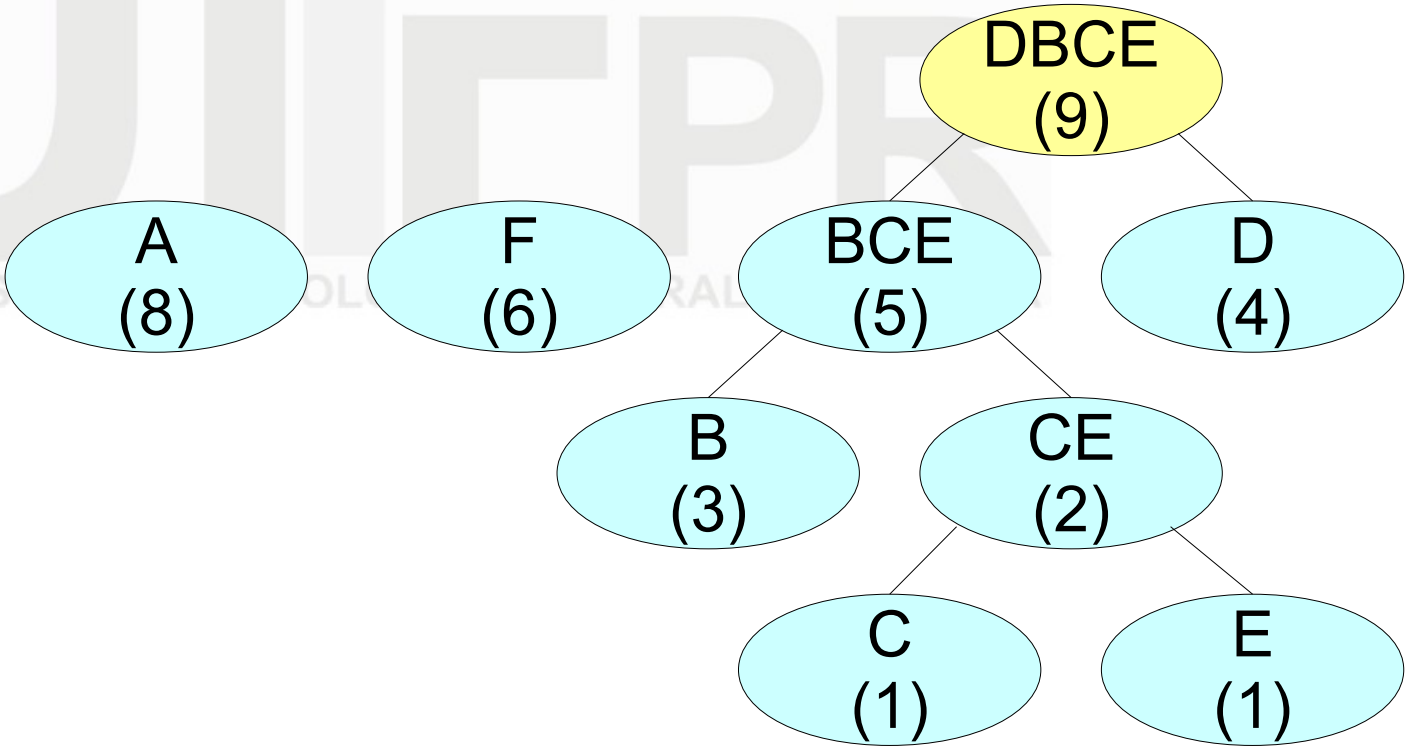
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



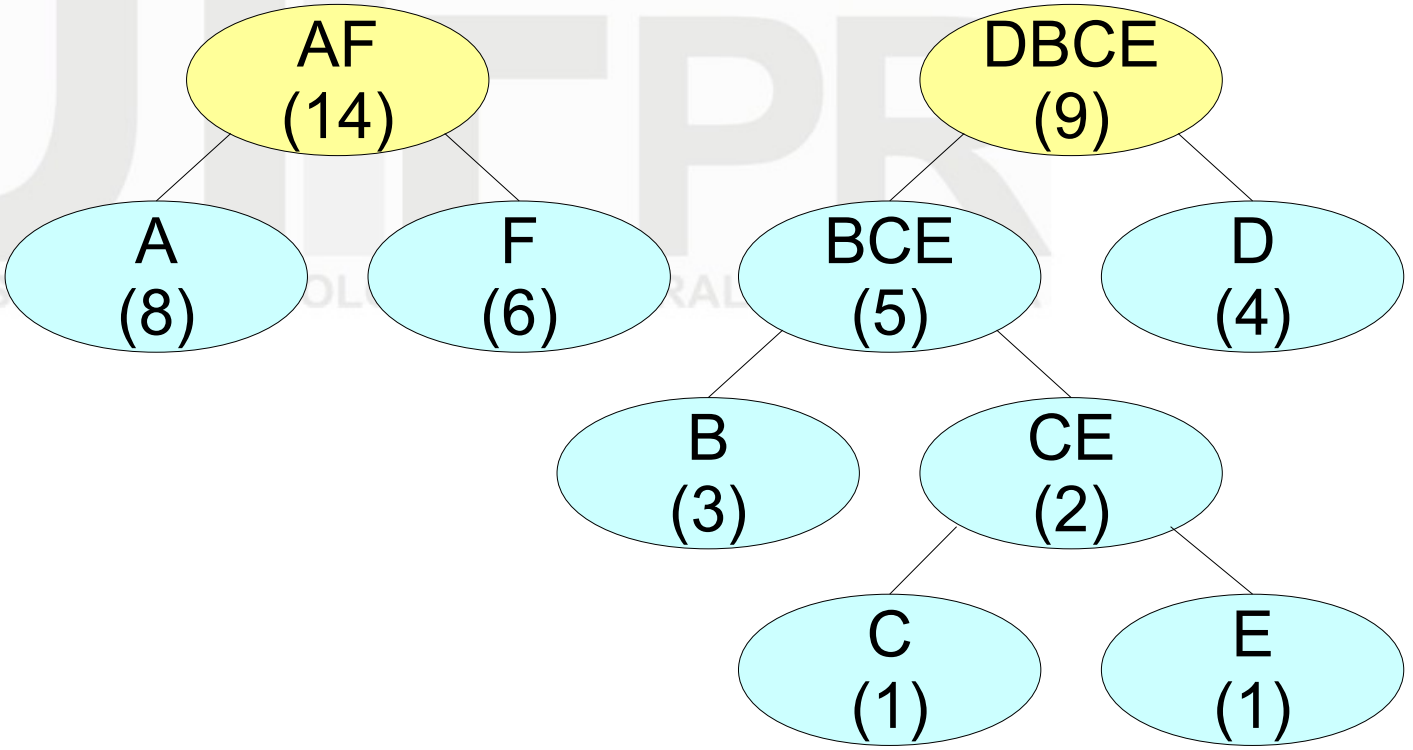
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



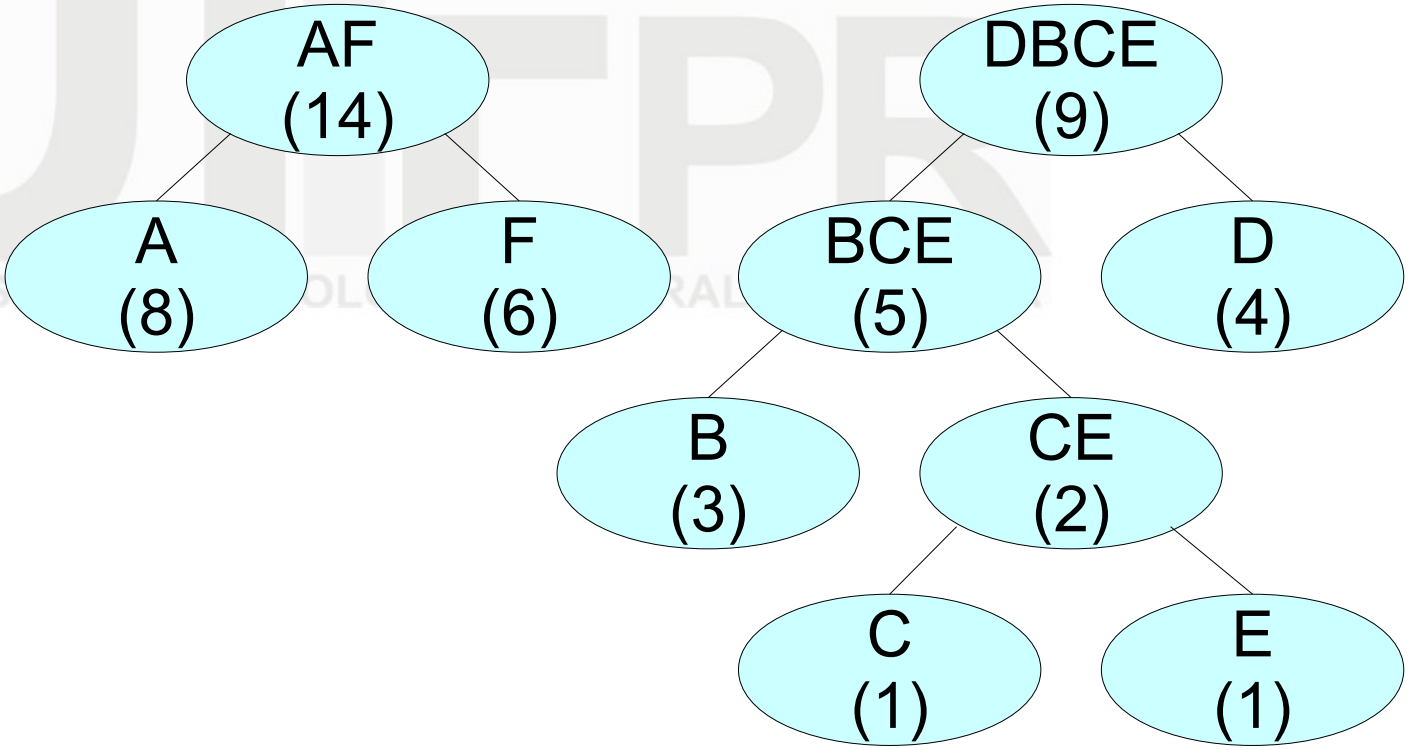
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



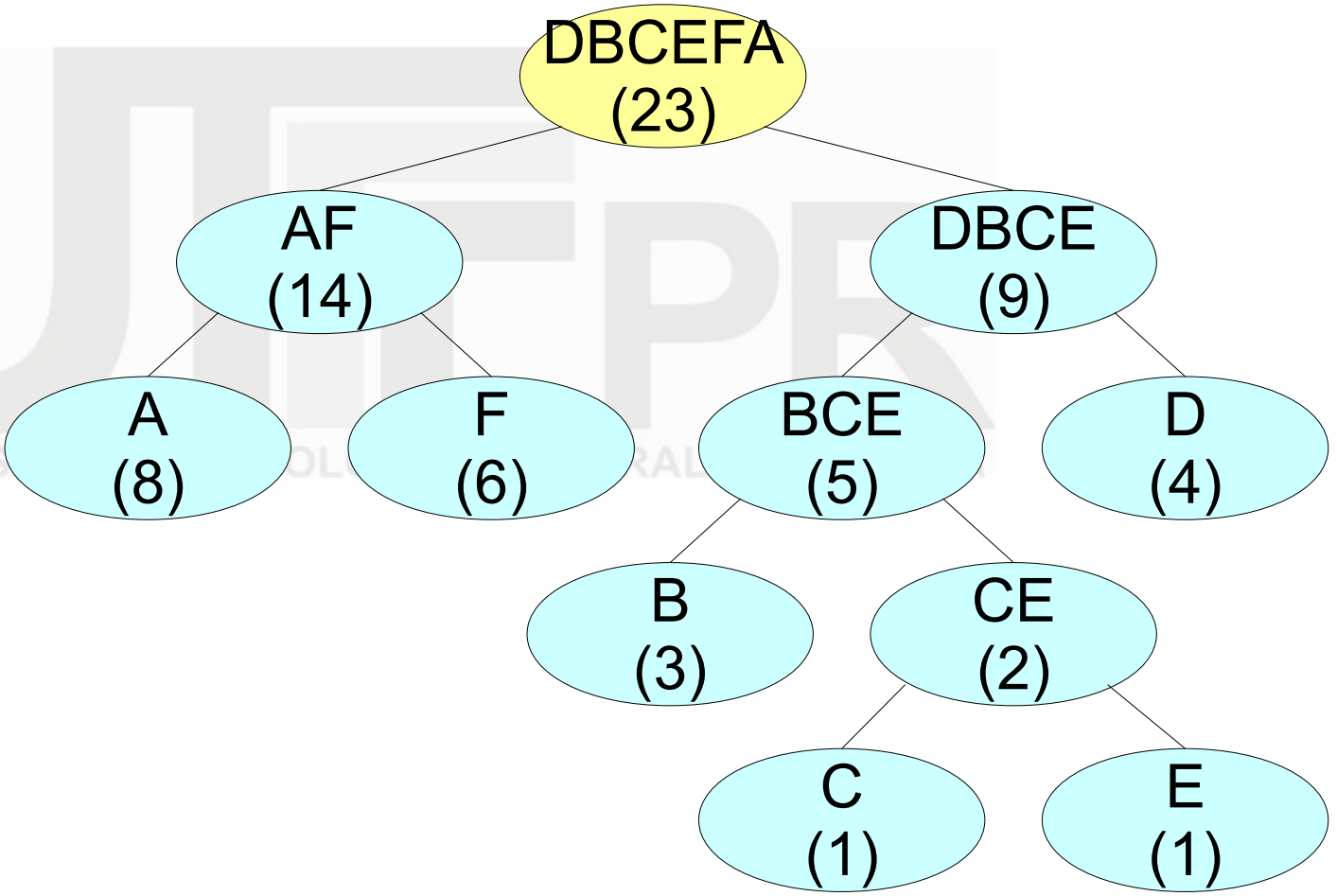
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



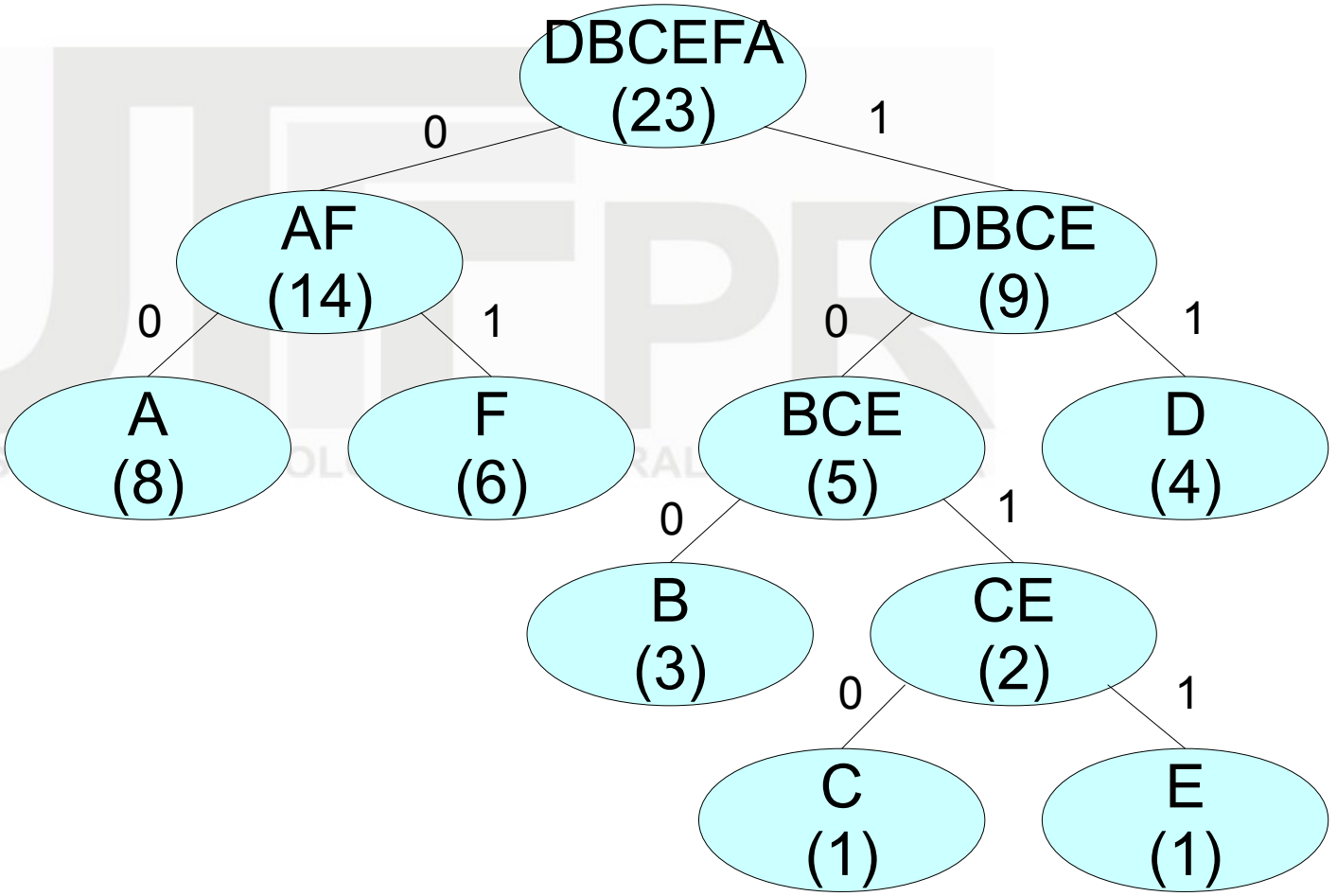
Códigos de Huffman

•Passo 3: percorre a árvore até chegar às folhas. Para cada nodo, atribui o bit 0 para o filho com mais ocorrências e o bit 1 para o outro.

- Os bits atribuídos no caminho até um dos símbolos originais são usados para representar aquele símbolo.

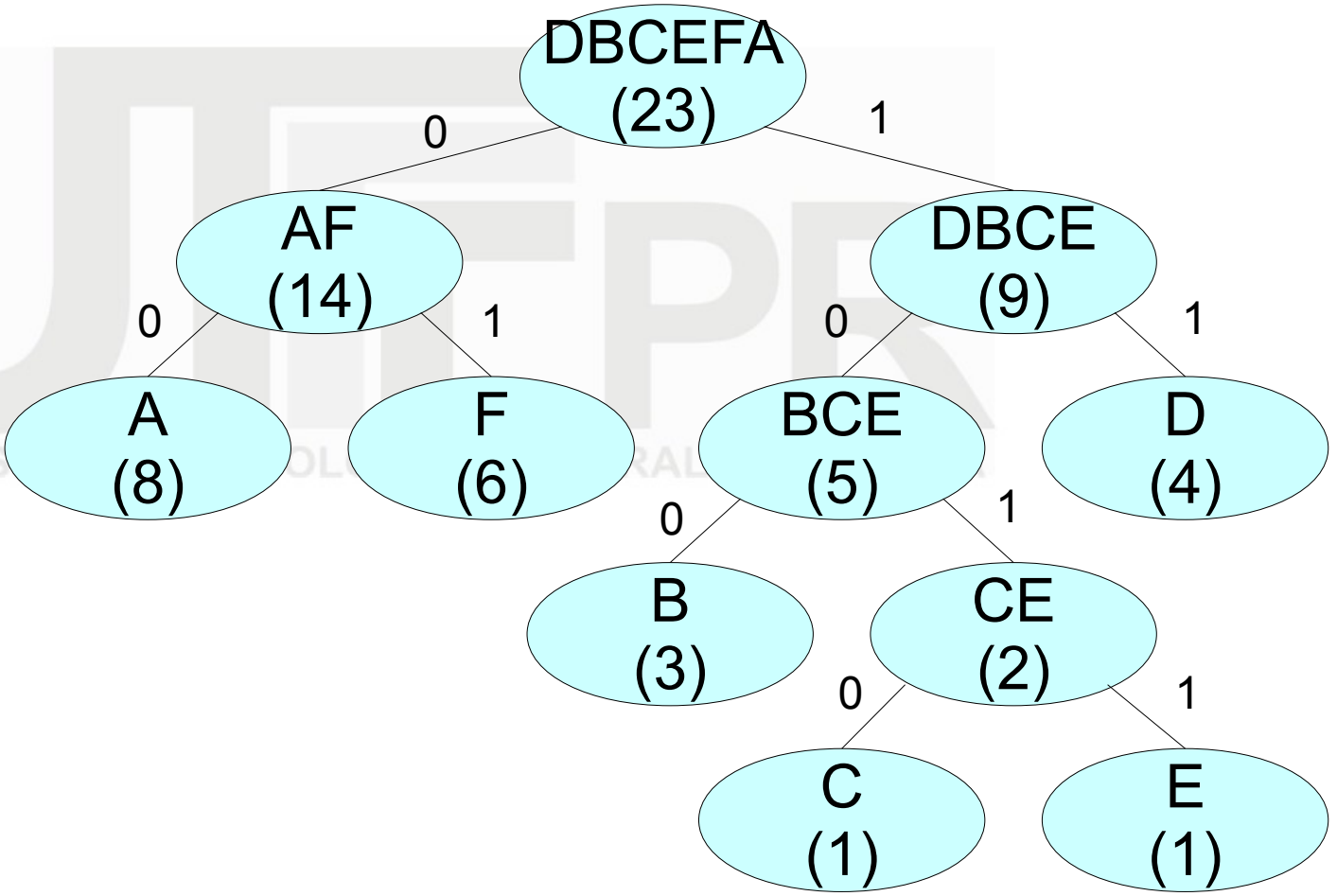
Códigos de Huffman: exemplo

Símbolo	Ocorr.
A	8
F	6
D	4
B	3
C	1
E	1



Códigos de Huffman: exemplo

Símbolo	Código
A	00
F	01
D	11
B	100
C	1010
E	1011



Códigos de Huffman

Símbolo	Código
A	00
F	01
D	11
B	100
C	1010
E	1011

- Passo 4: para codificar / decodificar uma sequência, basta usar a LUT.
- Exemplo:
  - Original: BAFDBCAFAADFAAFDAEBAFDF
    - 6 símbolos → com tamanho fixo, exige pelo menos 3 bits por símbolo.
      - = 23*3 = 69 bits.
  - Codificada: 10000011110010100001000011010000011100101110000011101
    - = 53 bits.

Códigos de Huffman: notas

- Códigos de Huffman e variações são **muito** usados!
  - Exemplos: MP3, ZIP, rede...
- Para decodificar uma sequência de bits, precisamos "resolver" os símbolos um a um, na ordem em que eles aparecem.
  - = a decodificação de um código de Huffman não é paralelizável.
- A LUT precisa ser conhecida, ou incorporada aos dados.
  - No caso do JPEG, usam-se LUTs padronizadas.
- Associar diretamente cores a códigos de Huffman funciona bem apenas se o número de cores é pequeno.
  - Lembre-se que o JPEG **não** faz isso!

JPEG: notas

- O formato JPEG inclui várias opções, como:
  - Fazer ou não a conversão para YCbCr?
  - O quanto subamostrar os canais Cb e Cr?
  - Qual tabela usar para quantizar as respostas da DCT?
  - Qual tabela usar para a codificação de Hufmann?
  - (Existem outras variações e opções!)
- Softwares que usam o formato JPEG costumam ter *presets* de qualidade, que variam aspectos como os citados acima.
  - Vejamos alguns exemplos...

JPEG: exemplo (compressão 0)

JPEG: exemplo (compressão 1)

JPEG: exemplo (compressão 15)

JPEG: exemplo (compressão 30)

JPEG: exemplo (compressão 45)

JPEG: exemplo (compressão 60)

JPEG: exemplo (compressão 75)

JPEG: exemplo (compressão 90)

JPEG: exemplo (compressão 99)

JPEG: exemplo (compressão 0)

JPEG: exemplo (compressão 1)

JPEG: exemplo (compressão 15)

JPEG: exemplo (compressão 30)

JPEG: exemplo (compressão 45)

JPEG: exemplo (compressão 60)

JPEG: exemplo (compressão 75)

JPEG: exemplo (compressão 90)

JPEG: exemplo (compressão 99)

Tamanhos dos arquivos...

	Exemplo 1 (256 x 256)	Exemplo 2 (240 x 160)
Dados (em Bytes)	196.608	115.200
BMP	196.662	115.254
JPEG 1%	62.086 (31.57%)	29.353 (25.47%)
JPEG 15%	21.083 (10.72%)	10.978 (9.53%)
JPEG 30%	14.701 (7.48%)	7.651 (6.64%)
JPEG 45%	11.774 (5.99%)	6.383 (5.54%)
JPEG 60%	9.893 (5.03%)	5.325 (4.62%)
JPEG 75%	7.615 (3.87%)	4.130 (3.58%)
JPEG 90%	4.477 (2.28%)	2.658 (2.31%)
JPEG 99%	2.257 (1.15%)	2.023 (1.76%)

Importante: os tamanhos dependem do conteúdo das imagens!

Outras alternativas...

- Codificação aritmética:
  - JPEG-2000, H.264 / MPEG-4 AVC.
  - Generalização do conceito dos códigos de Huffman.
  - Permite mais compressão que os códigos de Huffman, mas é mais complicado de implementar e tem restrições ligadas a patentes.
- Separação em planos de bits.
  - JPEG-2000.
  - Gera imagens binárias, uma para cada bit da cor.
    - Comprime cada uma separadamente.
  - Ideia: existem menos variações nos bits mais significativos.
- Codificação preditiva.
  - Formatos de vídeo em geral.
  - Redundância temporal: valores repetidos ou parecidos costumam aparecer nas mesmas posições, ou se deslocar de forma previsível.
  - Codifica as diferenças para certos quadros anteriores (keyframes).