24 - Pemampatan Citra

(Bagian 1)

IF4073 Interpretasi dan Pengolahan Citra

Oleh: Rinaldi Munir

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2021

Pemampatan vs Penirmampatan

- *Image compression* = pemampatan citra, kompresi citra
- *Image decompression* = penirmampatan citra, dekompresi citra

- Citra dimampatkan ketika ia disimpan ke dalam *storage* atau ditransmisikan.

- Citra dinirmampatkan ketika ia ditampilkan ke layer, dicetak ke *printer*, atau disimpan ke dalam dokumen dengan format tidak mampat

Mengapa citra perlu dimampatkan?

- Representasi citra digital membutuhkan memori yang besar.
- Pemampatan citra adalah metode untuk mereduksi redundansi pada representasi citra sehingga dapat mengurangi kebutuhan memori untuk ruang penyimpanan.
- Citra dimampatkan tanpa mengurangi kualitas citra secara visual
- Tujuan:
  1. Mengurangi kebutuhan ruang penyimpanan sembari tetap mempertahankan kualitas citra secara visual. (Gonzalez, Woods and Eddins, 2017).
  2. Merepresentasikan citra dengan kualitas yang hampir sama dengan citra aslinya namun dalam bentuk yang lebih kompak.

Dapatkah anda melihat perbedaan kualitas hasil pemampatan?

Original image
(not compressed)

Compressed image

peppers.bmp, 256 x 256
(193 KB)

peppers.jpg, 256 x 256
(31 KB), JPEG Quality = 5

peppers2.jpg, 256 x 256
(24 KB), JPEG Quality = 1

- Misalkan sebuah citra berwarna berukuran 1200x1600

  Kebutuhan ruang penyimpanan:

  1200 x 1600 x 3 = 5760000 byte

  = 5,760 *Kbyte*

  = 5.76 *Mbyte*

- Misalkan sebuah film digital dengan resolusi 720x480, 30 frame/sec, selama 2 jam.

  Kebutuhan ruang penyimpanan:

$$30\frac{frame}{\sec}\times(760\times480)\frac{pixels}{frame}\times3\frac{bytes}{pixel}=31{,}104{,}000bytes/\sec$$

$$31{,}104{,}000\times\frac{bytes}{\sec}\times(60\times60)\frac{\sec}{hour}\times2hours=2.24\times10^{11}bytes$$

$$=224GByte.$$

Aplikasi pemampatan citra

1. Penyimpanan data di dalam media sekunder (*storage*)

   Citra mampat membutuhkan memori di dalam *storage* yang lebih sedikit dibandingkan dengan citra yang tidak mampat.

   Contoh: file citra berformat JPEG/JPG versus citra berformat BMP

2. Pengiriman data (*data transmission*) pada saluran komunikasi data

   Citra mampat membutuhkan waktu pengiriman yang lebih singkat dibandingkan dengan citra tidak mampat.

   Contoh: pengiriman gambar via email, *videoconference*, via satelit luar angkasa, mengunduh gambar dari internet, dan sebagainya.

Redundansi

- Redudansi pada citra adalah konten citra yang sebenarnya tidak perlu direpresentasikan dalam sejumlah bit.

- Dua macam redundansi:
  1. *Coding redundancy*: biasanya muncul sebagai hasil pengkodean yang seragam pada setiap *pixel.*
  2. *Spatial/temporal redundancy:* misalnya *pixel-pixel* bertetangga memiliki nilai intensitas yang tidak jauh berbeda.
  3. Psychovisual redundancy: persepsi visual mengakibatkan *redundancy*

Coding redundancy

Symbol r_k	Probability $p_r(r_k)$	Code 1	Length $l_1(r_k)$
$r_0 = A$	0.19	000	3
$r_1 = B$	0.25	001	3
$r_2 = C$	0.21	010	3
$r_3 = D$	0.16	011	3
$r_4 = E$	0.08	100	3
$r_5 = F$	0.06	101	3
$r_6 = G$	0.03	110	3
$r_7 = H$	0.02	111	3

- Tiap simbol, tanpa memperhatikan frekuensi kemunculannya, dikodekan dengan panjang bit yang tetap (fixed-length encoding), yaitu 3 bit.
- Rata-rata panjang bit kode untuk setiap simbol 3 bit

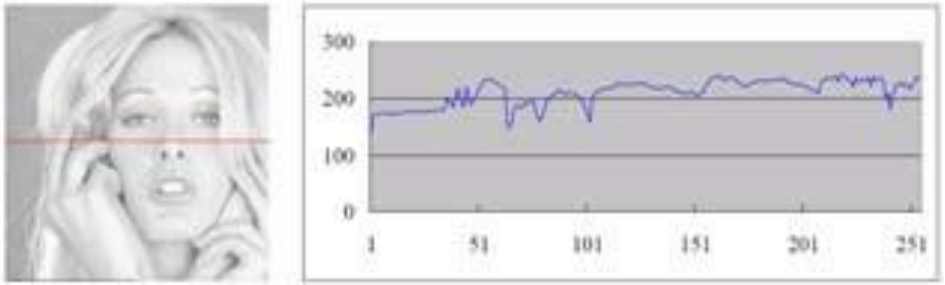
- *Coding redundancy* dapat dikurangi dengan mengkodekan simbol yang sering muncul dengan jumlah bit yang lebih sedikit

Symbol r_k	Probability $p_r(r_k)$	Code 2	Length $l_2(r_k)$
$r_0 = A$	0.19	11	2
$r_1 = B$	0.25	01	2
$r_2 = C$	0.21	10	2
$r_3 = D$	0.16	001	3
$r_4 = E$	0.08	0001	4
$r_5 = F$	0.06	00001	5
$r_6 = G$	0.03	000001	6
$r_7 = H$	0.02	000000	6

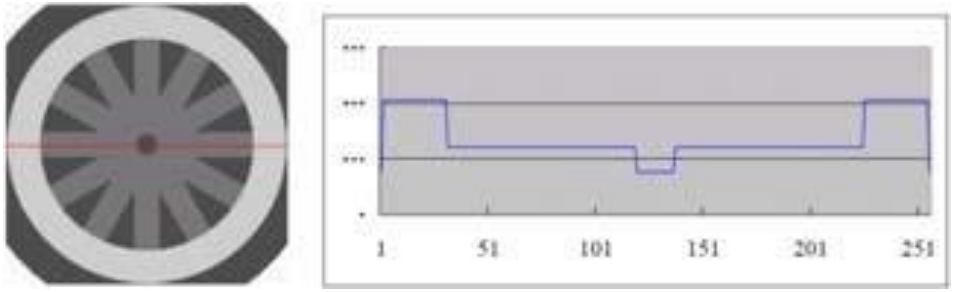
- Rata-rata panjang bit kode untuk setiap simbol = {(19 x) + (25 x 2) + (21 x 2) + … + (3 x 6) + (2 x 6)}/100 = 2.7
- Nisbah pemampatan= 3/2.7 = 1.11

Spatial redundancy

Nilai-nilai *pixel* citra Tiffany pada baris 128:

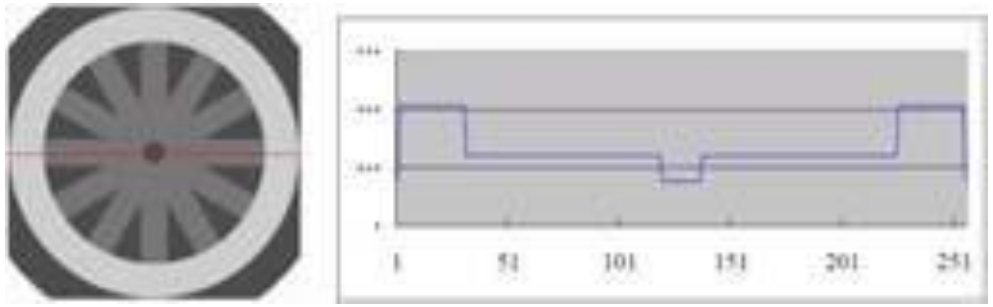


Nilai-nilai *pixel* citra roda pada baris 128:



Sumber*: Image Processing*, *Image Compression*, DR. FERDA ERNAWAN
Faculty of Computer Systems & Software Engineering, Pahang University

- Pixel-pixel baris 128 pada citra roda (256 pixel):



- Misalkan *n*1 menyatakan graylevel dan *n*2 adalah frekuensi kemunculannya

Pada baris 128, (n1, n2): (77, 1), (206, 30), (121, 88), (77,18), (121, 88), (206, 30), (77, 1)

Kodekan setiap segmen dengan 16 bit (n1 delapan bit, n2 delapan bit):
Nisbah pemampatan = (256x8) / (7x16)=18.3

Sumber*: Image Processing*, *Image Compression*, DR. FERDA ERNAWAN
Faculty of Computer Systems & Software Engineering, Pahang University

Psychovisual Redundancy

- Untuk persepsi visual manusia, informasi tertentu kurang penting. E.g ,: kuantisasi graylevel yang tepat tidak memengaruhi kualitas visualnya.

- Citra Tiffany ini jika dikodekan 5 bit/pixel tidak mempengaruhi persepsi visual manusia

8 bits/pixel

5 bits/pixel

- Metode pemampatan kuantisasi.

Sumber*: Image Processing*, *Image Compression*, DR. FERDA ERNAWAN
Faculty of Computer Systems & Software Engineering, Pahang University

Teori Informasi

- Mendefinisikan jumlah informasi di dalam pesan sebagai jumlah minimum bit yang dibutuhkan untuk mengkodekan pesan.

- Contoh:
  - 1 bit untuk mengkodekan jenis kelamin
  - 3 bit untuk mengkodekan nama hari
  - 4 bit untuk mengkodekan 0 s/d 9

- *Entropy*: ukuran yang menyatakan jumlah informasi di dalam pesan.
- Biasanya dinyatakan dalam satuan bit.

- Entropi berguna untuk memperkirakan jumlah bit rata-rata untuk mengkodekan elemen dari pesan.

- Contoh: entropi untuk pesan yang menyatakan jenis kelamin = 1 bit, entropi untuk pesan yang menyatakan nama hari = 3 bit

- Secara umum, entropi pesan dihitung dengan rumus yang dikemukakan oleh Claude Shannon, 1948:

$$H(X) = -\sum_{i=i}^{n} {p_i}^2 \log(p_i)$$

p_i = peluang kemunculan simbol ke-*i* di dalam pesan *X*

- Entropi dinyatakan dalam satuan bit

- Contoh: misalkan pesan X = ‘AABBCBDB’

  n = 4 (yaitu huruf A, B, C, D)

  p(A) = 2/8, p(B) = 4/8

  p(C) = 1/8, p(D) = 1/8

$H(x) = -\{2/8\ {}^2\log(2/8) + 4/8\ {}^2\log(4/8) + 1/8\ {}^2\log(1/8) + 1/8\ {}^2\log(1/8)\ \}$

$= -\{1/4\ {}^2\log(1/4) + 1/2\ {}^2\log(1/2) + 1/8\ {}^2\log(1/8) + 1/8\ {}^2\log(1/8)\ \}$

$= -\{(1/4)\ (-\ {}^2\log(4) + (1/2)\ (-\ {}^2\log(2) + (1/8)\ (-\ {}^2\log(8) + (/8)\ (-\ {}^2\log(8)\ \}$

$= -\{(1/4)\ (-\ 2) + (1/2)\ (-1) + (1/8)\ (-3) + (1/8)(-3)\ \}$

$= -\ \{-1/2 – 1/2 – 3/8 – 3/8)$

$= -\ (-1.75)$

$= 1.75$

Entropi = 1,75 bit per simbol

Let only two symbols a, b occur in the message.

Example 1

$p(\mathsf{a}) = p(\mathsf{b}) = \frac{1}{2}$

$$H = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = \left(\frac{1}{2}\cdot 1 + \frac{1}{2}\cdot 1\right) = 1$$

Example 2

$p(\mathsf{a}) = 0,99;\ p(\mathsf{b}) = 0,01$

$$\begin{aligned} H &= -\left(0,99\log_2 0,99 + 0,01\log_2 0,01\right) \\ &= -\left(0.99\cdot(-0,0145) + 0,01\cdot(-6,6439)\right) \\ &= 0,0144 + 0,0664 = 0,0808 \end{aligned}$$

Tipe metode pemampatan citra

1. Lossy compression

- Metode *lossy* menghasilkan citra hasil pemampatan yang *hampir* sama dengan citra semula. Ada informasi yang hilang akibat pemampatan, tetapi dapat ditolerir oleh persepsi visual.
- Bertujuan untuk memperoleh nisbah pemampatan yang tinggi
- Contoh: *JPEG compression*, *fractal image compression*

2. Lossless compression

- Metode *lossless* selalu menghasilkan citra hasil penirmampatan yang tepat sama dengan citra semula, *pixel* per *pixel*. Tidak ada informasi yang hilang akibat pemampatan.
- Nisbah pemampatan rendah, namun kualitas citra mampat tetap tinggi
- Dibutuhkan untuk memampatkan citra yang tidak boleh terdegradasi akibat pemampatan, misalnya citra medis, citra x-ray
- Contoh: metode Huffman, *run-length encoding* (RLE), *quantized coding*

Lossless compression

Lossy compression

peppers.bmp, 256 x 256
(193 KB)

peppers.png, 256 x 256
(127 KB)

peppers2.jpg, 256 x 256
(24 KB), JPEG Quality = 1

Metode Pemampatan Huffman

- *Lossless compression*

- Berdasarkan algoritma *greedy*

- Prinsip kerja algoritma: *pixel* dengan nilai keabuan yang sering muncul dikodekan dengan panjang bit yang lebih sedikit, sebaliknya nilai keabuan yang jarang muncul dikodekan dengan it yang lebih panjang

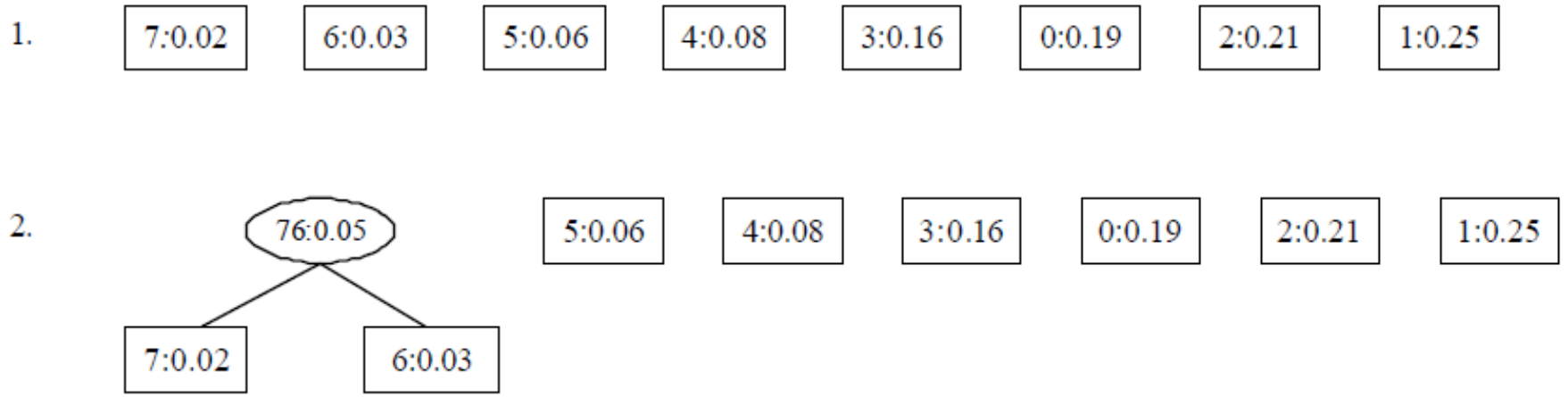
Algoritma:

1. Setiap nilai keabuan dinyatakan sebagai simpul. Setiap simpul di-*assign* dengan frekuensi kemunculan nilai keabuan tersebut.
2. Urutkan secara menaik (*ascending order*) simpul-simpul berdasarkan frekuensi kemunculannya.
3. Gabung dua buah simpul yang mempunyai frekuensi kemunculan paling kecil pada sebuah akar. Akar mempunyai frekuensi yang merupakan jumlah dari frekuensi dua buah pohon penyusunnya.
4. Ulangi langkah 2 sampai tersisa hanya satu buah pohon biner.
5. Beri label setiap sisi pada pohon biner. Sisi kiri dilabeli dengan 0 dan sisi kanan dilabeli dengan 1.
6. Telusuri pohon biner dari akar ke daun. Barisan label-label sisi dari akar ke daun menyatakan kode Huffman untuk derajat keabuan yang bersesuaian.

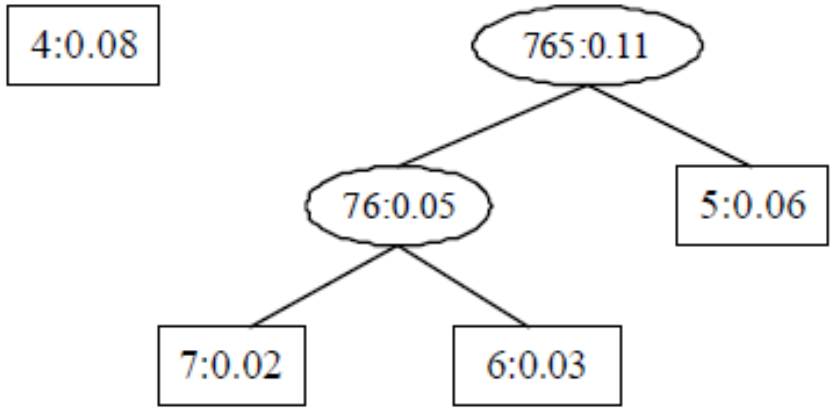
Contoh: Misalkan terdapat citra yang berukuran 64 × 64 dengan 8 derajat keabuan (*k*) dan jumlah seluruh *pixel* (*n*) = 64 × 64 = 4096.

k	n_k	$p(k) = n_k/n$
0	790	0.19
1	1023	0.25
2	850	0.21
3	656	0.16
4	329	0.08
5	245	0.06
6	122	0.03
7	81	0.02

Proses pembentuan pohon Huffman:

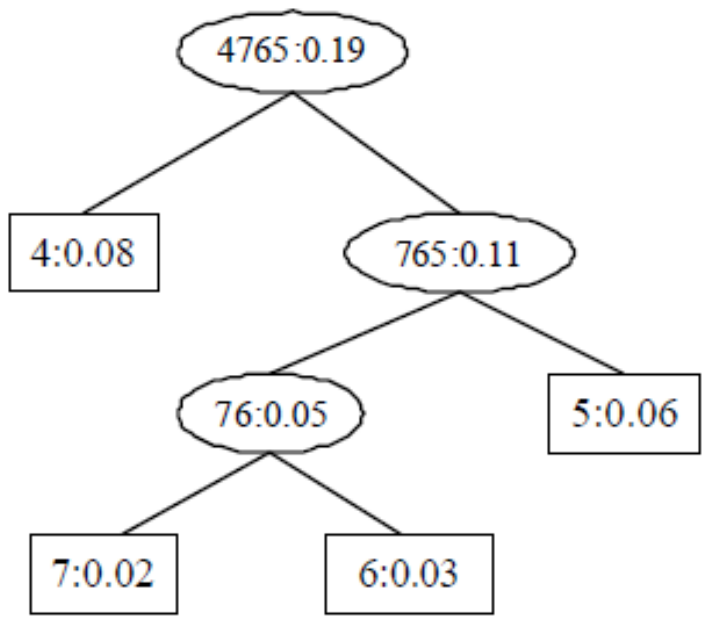


3.



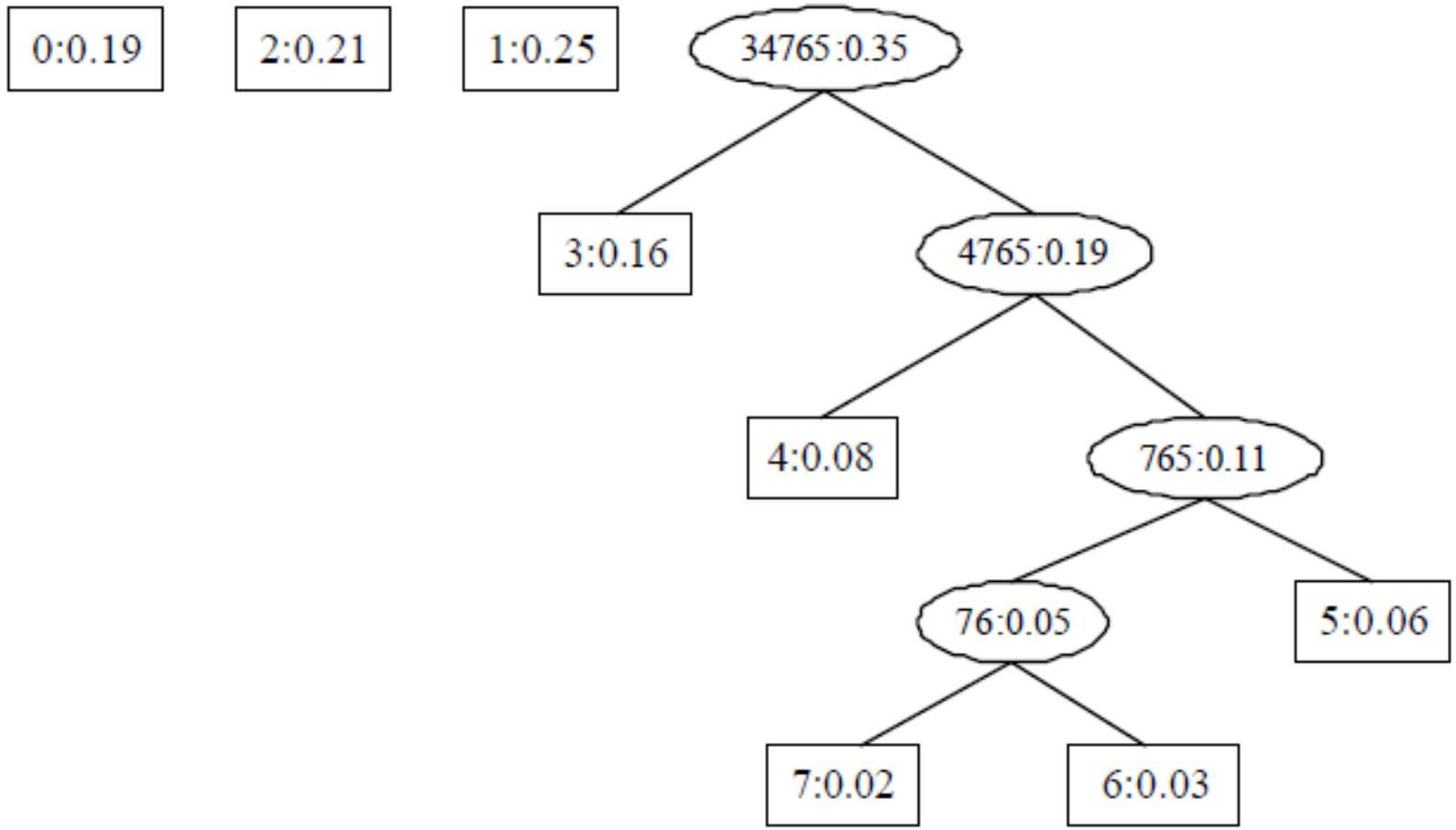
3:0.16 0:0.19 2:0.21 1:0.25

4.

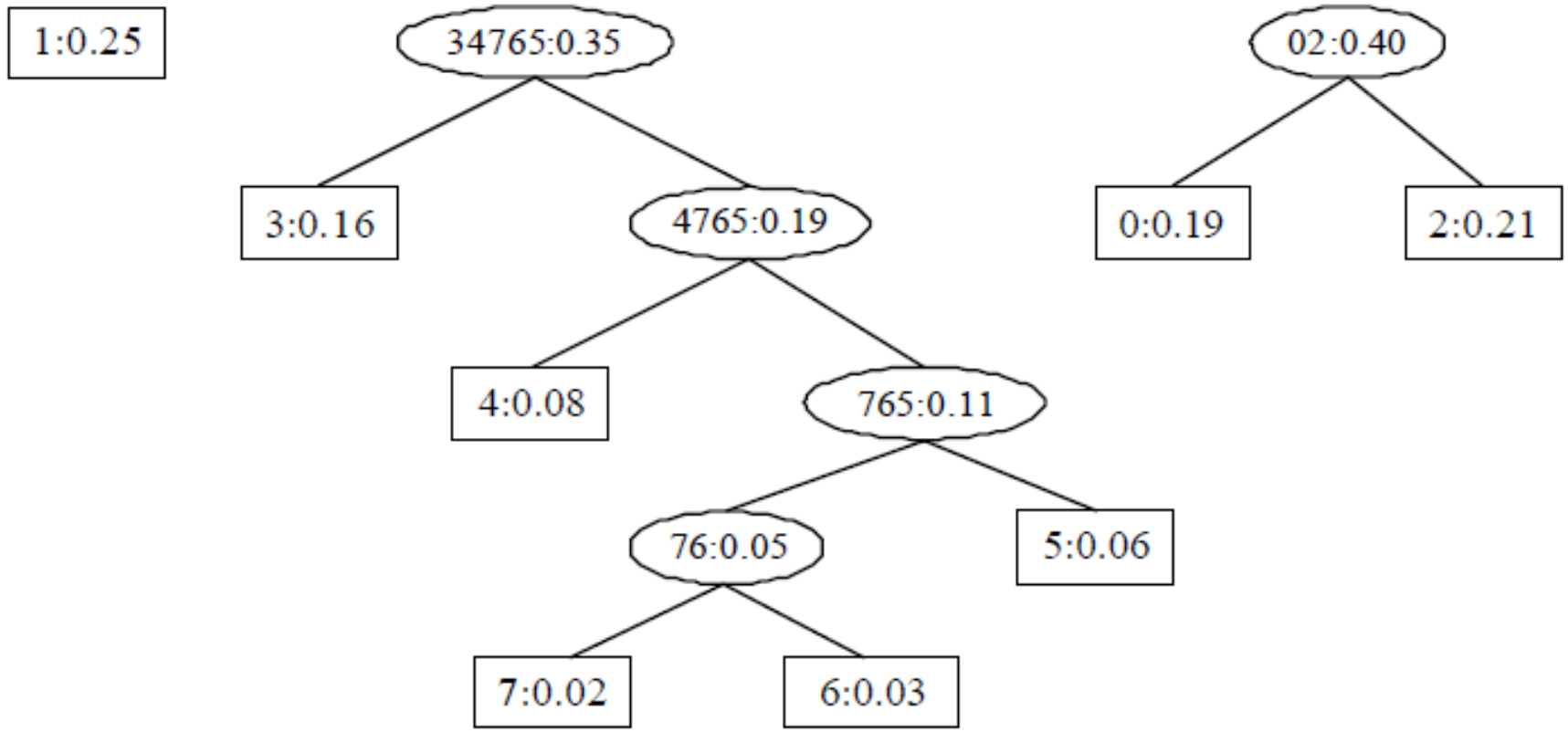


3:0.16 0:0.19 2:0.21 1:0.25

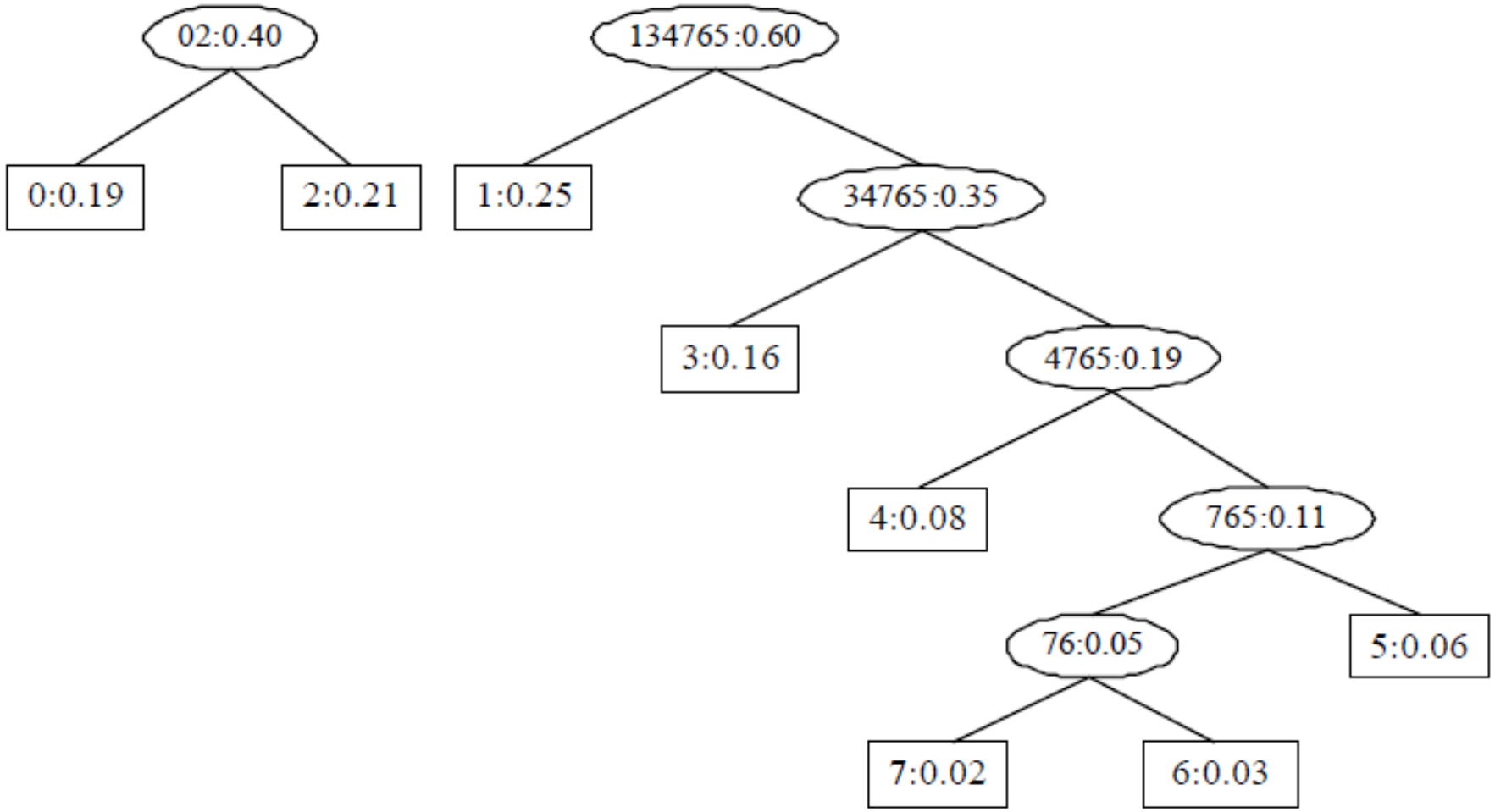
5.


0:0.19
2:0.21
1:0.25
34765:0.35
3:0.16
4765:0.19
4:0.08
765:0.11
76:0.05
5:0.06
7:0.02
6:0.03

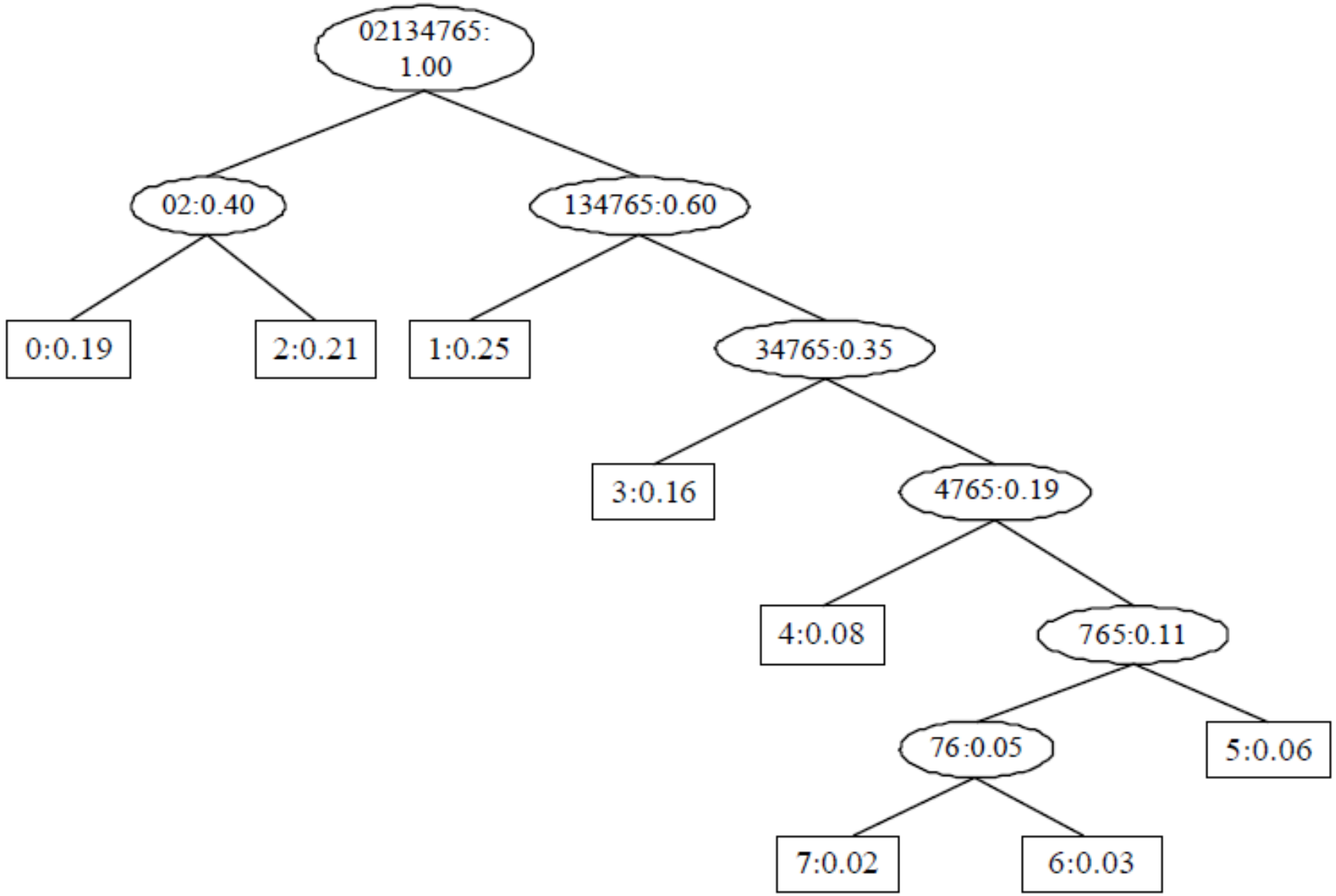
6.


1:0.25
34765:0.35
02:0.40
3:0.16
4765:0.19
0:0.19
2:0.21
4:0.08
765:0.11
76:0.05
5:0.06
7:0.02
6:0.03

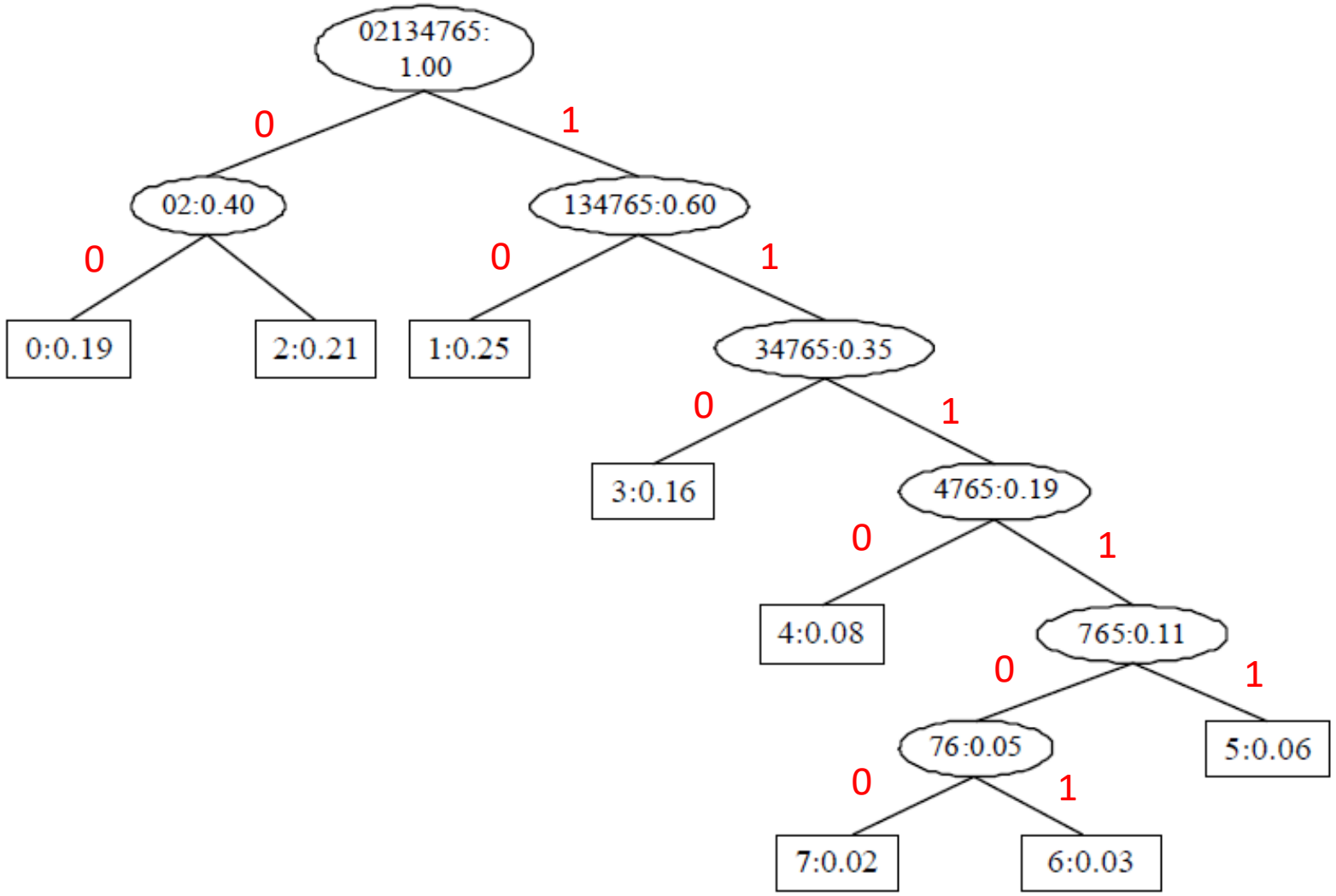
7.


02:0.40
134765:0.60
0:0.19
2:0.21
1:0.25
34765:0.35
3:0.16
4765:0.19
4:0.08
765:0.11
76:0.05
5:0.06
7:0.02
6:0.03

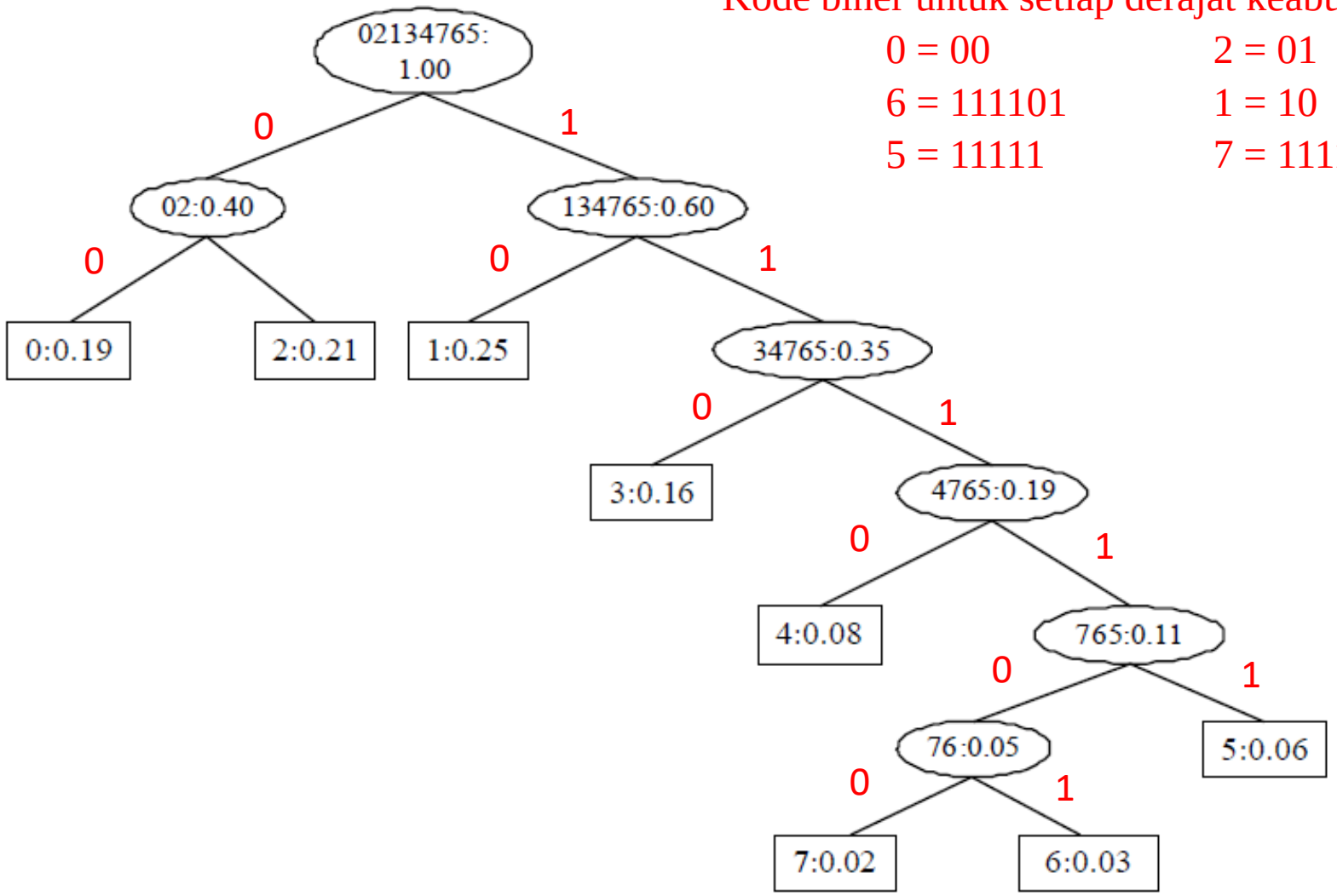
8.


02134765:
1.00
02:0.40
134765:0.60
0:0.19
2:0.21
1:0.25
34765:0.35
3:0.16
4765:0.19
4:0.08
765:0.11
76:0.05
5:0.06
7:0.02
6:0.03

Beri label setiap sisi pada pohon biner. Sisi kiri dilabeli dengan 0 dan sisi kanan dilabeli dengan 1



Telusuri pohon biner dari akar ke daun. Barisan label-label sisi dari akar ke daun menyatakan kode Huffman untuk derajat keabuan yang bersesuaian.



Kode biner untuk setiap derajat keabuan sebagai berikut:

0 = 00	2 = 01	4 = 1110
6 = 111101	1 = 10	3 = 110
5 = 11111	7 = 111100	

- Ukuran citra sebelum pemampatan (1 derajat keabuan = 3 bit) adalah 4096×3 bit = 12288 bit

- Ukuran citra setelah pemampatan:

  $(790 \times 2 \text{ bit}) + (1023 \times 2 \text{ bit}) + (850 \times 2 \text{ bit}) +$

  $(656 \times 3 \text{ bit}) + (329 \times 4 \text{ bit}) + (245 \times 5 \text{ bit}) +$

  $(122 \times 6 \text{ bit}) + (81 \times 6 \text{ bit}) = 11053$ bit

- Nisbah (ratio) pemampatan = $\frac{11053}{12288} \times 100\% = 89.95\%$

Artinya: - citra berhasil dimampatkan menjadi 89.95% dari citra semula
- atau 10.05 % dari citra semula telah dimampatkan

Metode *Run-Length Encoding* (RLE)

- Metode *RLE* cocok digunakan untuk memampatkan citra yang memiliki kelompok-kelompok *pixel* berderajat keabuan sama.

- Pemampatan citra dengan metode *RLE* dilakukan dengan membuat rangkaian pasangan nilai (*p*, *q*) untuk setiap baris *pixel*.

- Nilai pertama (*p*) menyatakan derajat keabuan (*graylevel*)

- Nilai kedua (*q*) menyatakan jumlah *pixel* berurutan yang memiliki derajat keabuan tersebut (dinamakan *run length*).

Contoh: Tinjau citra 10 × 10 *pixel* dengan 8 derajat keabuan yang dinyatakan sebagai matriks derajat keabuan sebagai berikut (100 buah nilai):

0	0	0	0	0	2	2	2	2	2
0	0	0	1	1	1	1	2	2	2
1	1	1	1	1	1	1	1	1	1
4	4	4	4	3	3	3	3	2	2
3	3	3	5	5	7	7	7	7	6
2	2	6	0	0	0	0	1	1	0
3	3	4	4	3	2	2	2	1	1
0	0	0	0	0	0	0	0	1	1
1	1	1	1	0	0	0	2	2	2
3	3	3	2	2	2	1	1	1	1

Pasangan nilai untuk setiap baris *run:*

(0, 5), (2, 5)
(0, 3), (1, 4), (2, 3)
(1, 10)
(4, 4), (3, 4), (2 2)
(3, 3), (5, 2), (7, 4), (6, 1)
(2, 2), (6, 1), (0, 4), (1, 2), (0, 1)
(3, 2), (4, 2), (3, 1), (2, 2), (1, 2)
(0, 8), (1, 2)
(1, 4), (0, 3), (2, 3)
(3, 3), (2, 3), (1, 4)

Semuanya ada 31 pasangan nilai, 31 × 2 = 62 nilai.

- Ukuran citra sebelum pemampatan (1 derajat keabuan = 3 bit) adalah 100 x 3 bit = 300 bit.

- Ukuran citra setelah pemampatan (derajat keabuan = 3 bit, run length = 4 bit):

  (31 x 3) + (31 x 4) bit = 217 bit

- Nisbah pemampatan = $\frac{217}{300} \times 100\% = 72.33\%$

Artinya: - citra berhasil dimampatkan menjadi 72.33% dari citra semula
- atau 27.67 % dari citra semula telah dimampatkan

- Metode *RLE* dapat dikombinasikan dengan metode Huffman untuk mengkodekan nilai-nilai hasil pemampatan *RLE*

- Tujuannya untuk meningkatkan nisbah pemampatan.

- Mula-mula lakukan pemampatan RLE, lalu hasilnya dimampatkan lagi dengan metode Huffman.

Metode Pemampatan Kuantisasi (*Quantizing Compression*)

- Metode ini mengurangi jumlah derajat keabuan, misalnya dari 256 menjadi 16, yang tentu saja mengurangi jumlah bit yang dibutuhkan untuk merepresentasikan citra.

- Misalkan *P* adalah jumlah pixel di dalam citra semula, akan dimampatkan menjadi *n* derajat keabuan.

- Algoritma metode kuantisasi:

1. Buat histogram citra semula (citra yang akan dimampatkan).
2. Identifikasi *n* buah kelompok di dalam histogram sedemikian sehingga setiap kelompok mempunyai kira-kira *P*/*n* buah *pixel*.
3. Nyatakan setiap kelompok dengan derajat keabuan 0 sampai *n* –1. Setiap *pixel* di dalam kelompok dikodekan kembali dengan nilai derajat keabuan yang baru.

Contoh: Tinjau citra yang berukuran 5×13 *pixel*:

2	9	6	4	8	2	6	3	8	5	9	3	7
3	8	5	4	7	6	3	8	2	8	4	7	3
3	8	4	7	4	9	2	3	8	2	7	4	9
3	9	4	7	2	7	6	2	1	6	5	3	0
2	0	4	3	8	9	5	4	7	1	2	8	3

yang akan dimampatkan menjadi citra dengan 4 derajat keabuan (0 s/d 3), jadi setiap derajat keabuan direpresentasikan dengan 2 bit

Histogram citra semula:

```
0 **
1 **
2 *********
3 **********
4 *********
5 ****
6 *****
7 ********
8 *********
9 ******
```

Ada 65 *pixel*, dikelompokkan menjadi 4 kelompok derajat keabuan. Tiap kelompok ada sebanyak rata-rata 65/4 = 16.25 *pixel* per kelompok:

```
-------------------------------------------------
          0 **
13        1 **                                  0
          2 *********
-------------------------------------------------
20        3 **********
          4 *********                           1
-------------------------------------------------
          5 ****
17        6 *****                               2
          7 ********
-------------------------------------------------
15        8 *********                           3
          9 ******
-------------------------------------------------
```

Citra setelah dimampatkan menjadi:

0	3	2	1	3	0	2	1	3	2	3	1	2
1	3	2	1	2	2	1	3	0	3	1	2	1
1	3	1	2	1	3	0	1	3	0	2	1	3
1	3	1	2	0	2	2	0	0	2	2	1	0
0	0	1	1	3	3	2	1	2	0	0	3	0

Ukuran citra sebelum pemampatan (1 derajat keabuan = 4 bit):

65×4 bit = 260 bit

Ukuran citra setelah pemampatan (1 derajat keabuan = 2 bit):

65×2 bit = 130 bit

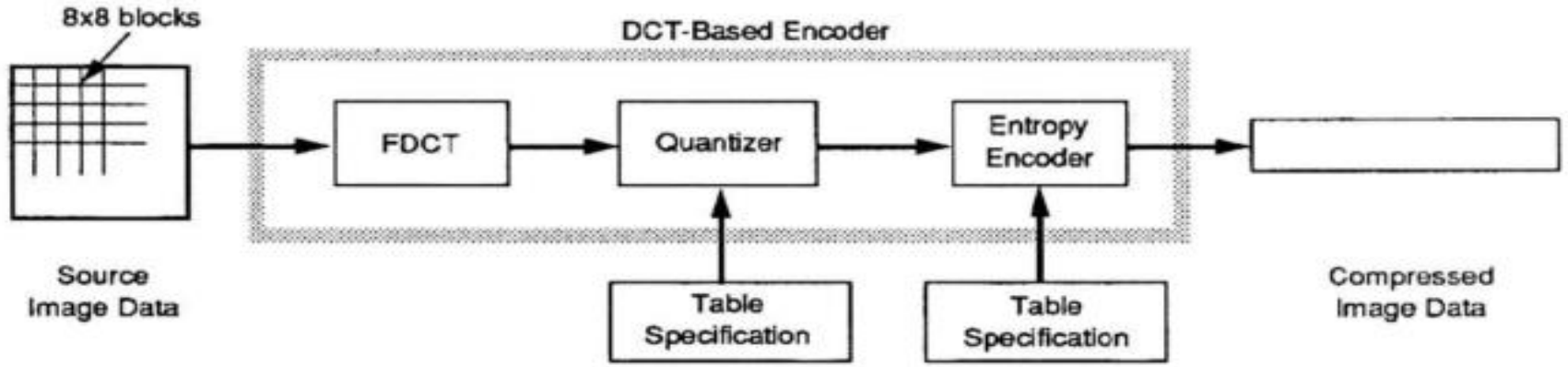
Nisbah pemampatan = 13/260 x 100% = 50%

Metode Pemampatan JPEG

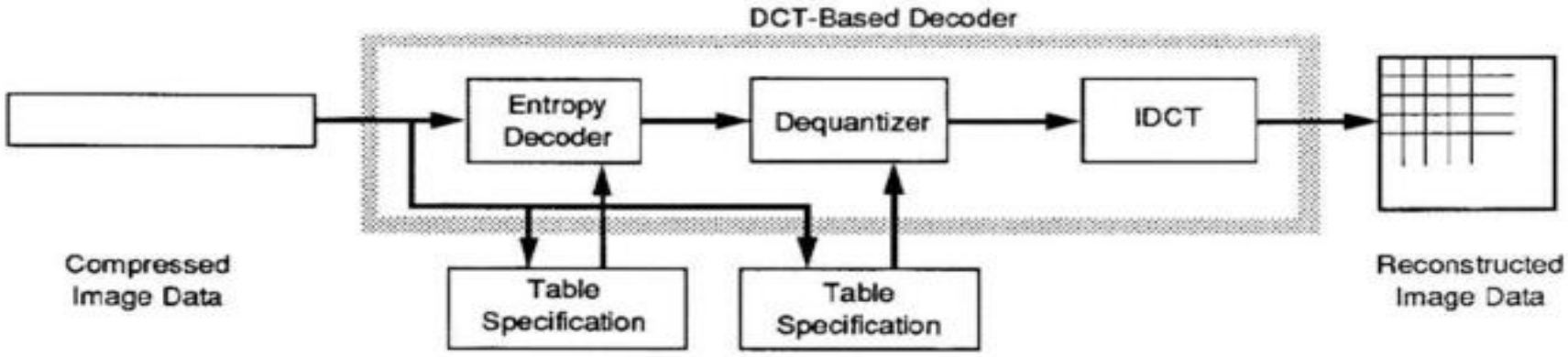
- JPEG = *Joint Photographic Experts Groups*.
- Merupakan standard kompresi citra sejak tahun 1992.
- Termasuk *lossy compression*, yang berarti beberapa kualitas visual ada yang hilang selama proses kompresi. Hasil dekompresi tidak kembali sama dengan citra semula.
- Metode JPEG dapat diterapkan pada citra *grayscale* dan citra berwarna RGB.
- Jika citra berwarna, maka RGB dikonversi terlebih dahulu ke ruang warna *YCbCr*, yang dalam hal ini *Y* adalah komponen *luminance*, dan *Cb* dan *Cr* adalah komponen *chrominance* dari citra.
- Terhadap komponen *chrominance* dilakukan *subsampling* untuk mengurangi ukuran file. *Subsampling* dikerjakan dengan mengambil 4 *pixel* bertetangga dan menghitung rata-ratanya menjadi satu nilai.

Diagram umum JPEG Compression

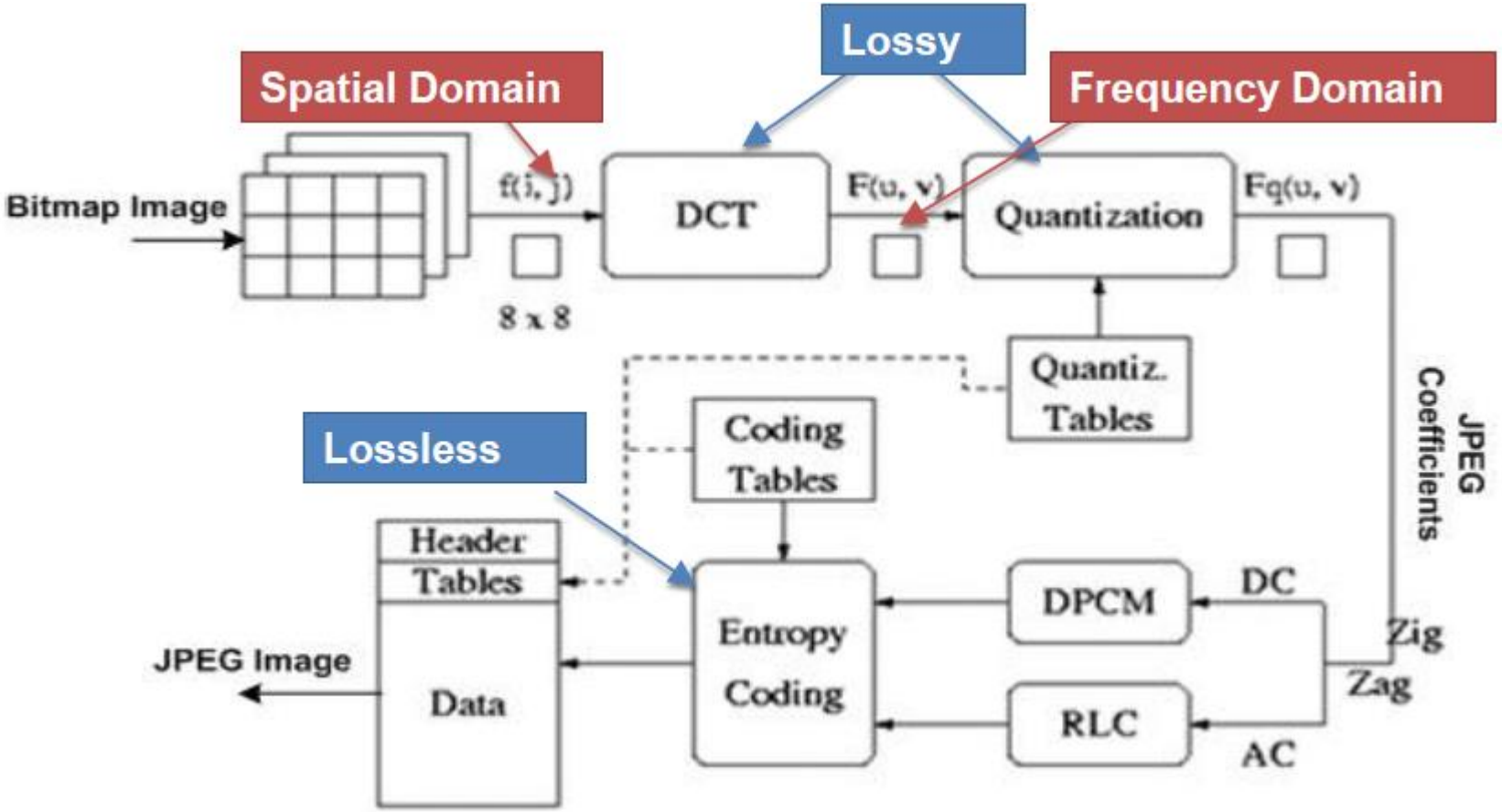
Encoder:



Decoder:

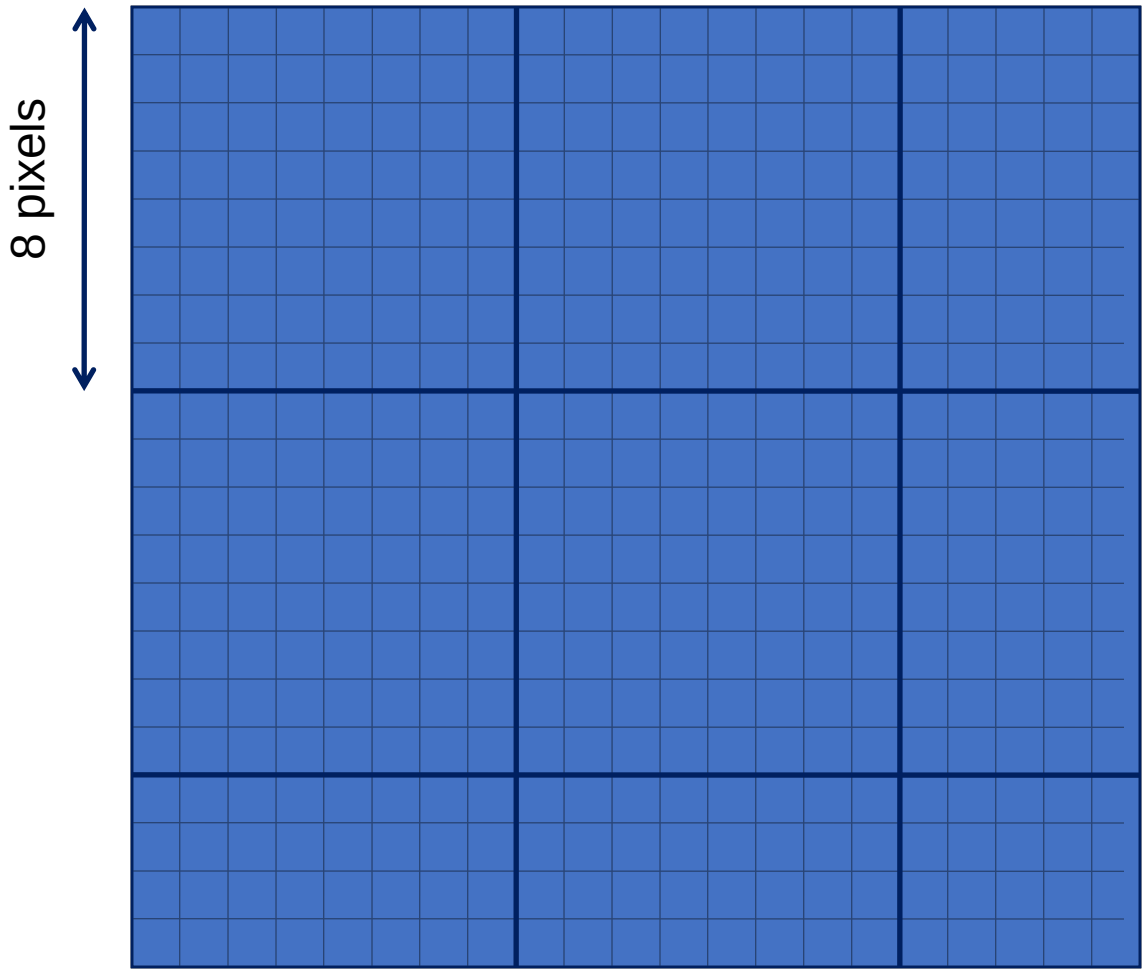


- Diagram rinci JPEG encoder):



Sumber: Mahendra Kumar, *Steganography and Steganalysis of JPEG Images*.

Citra dibagi menjadi blok-blok berukuran 8 x 8 *pixel*:



Setiap blok (dalam ranah spasial) ditranformasi ke ranah frekeuensi dengan *Discrete Cosine Transform* (DCT):

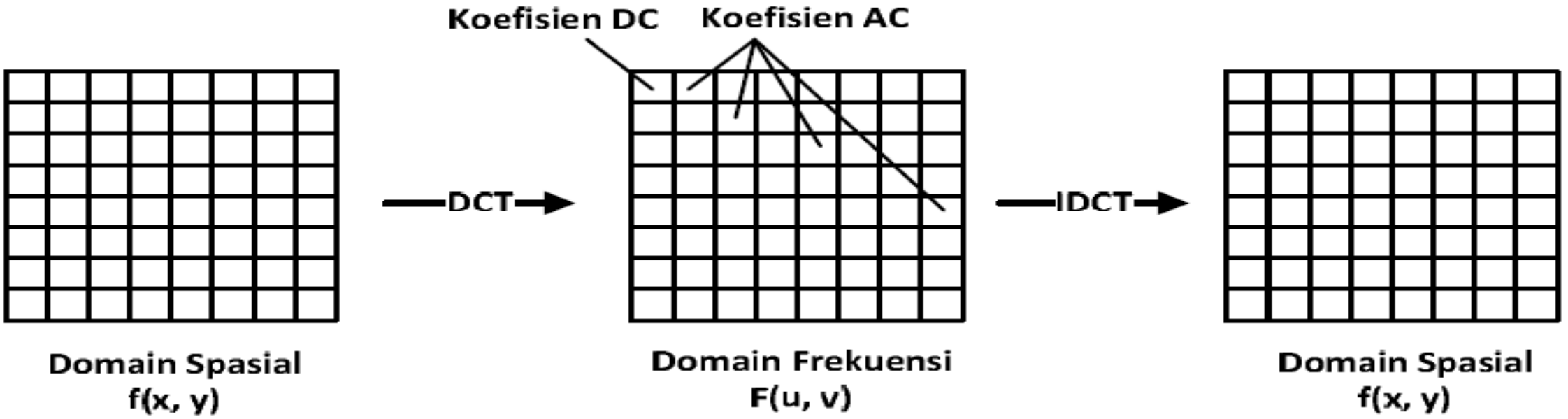
$$F(u,v) = \frac{1}{4} C(u)C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos \frac{\pi(2x+1)u}{16} \cos \frac{\pi(2y+1)v}{16}$$

$$C(u), C(v) = \frac{1}{\sqrt{2}}, \text{untuk } u, v = 0$$

$$C(u), C(v) = 1, \text{untuk } u, v \text{ lainnya}$$

Sedangkan transformasi balikannya adalah Inverse *Discrete Cosine Transform* (IDCT):

$$f(x,y) = C(u)C(v) \sum_{u=0}^{7} \sum_{v=0}^{7} F(u,v) \cos \frac{\pi(2x+1)u}{16} \cos \frac{\pi(2y+1)v}{16}$$



Elemen pojok kiri atas (1,1) disebut koefisien DC, sedangkan 63 elemen lainnya disebut koefisien AC.

- Hasilnya adalah blok-blok berukuran 8 x 8 dengan nilai *floating point*.
- Contoh:

$$F(u,v) = \begin{bmatrix} 515 & 65 & -12 & 4 & 1 & 2 & -8 & 5 \\ -16 & 3 & 2 & 0 & 0 & -11 & -2 & 3 \\ -12 & 6 & 11 & -1 & 3 & 0 & 1 & -2 \\ -8 & 3 & -4 & 2 & -2 & -3 & -5 & -2 \\ 0 & -2 & 7 & -5 & 4 & 0 & -1 & -4 \\ 0 & -3 & -1 & 0 & 4 & 1 & -1 & 0 \\ 3 & -2 & -3 & 3 & 3 & -1 & -1 & 3 \\ -2 & 5 & -2 & 4 & -2 & 2 & -3 & 0 \end{bmatrix}$$

- Selanjutnya, setiap nilai di dalam blok dikuantisasi menjadi *integer* dengan cara membaginya dengan elemen matriks kuantisasi Q berukuran 8 x 8 dan membulatkannya ke integer terdekat.

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

$$\hat{F}(u,v) = round\left(\frac{F(u,v)}{Q(u,v)}\right)$$

- Contoh hasil proses kuantisasi:

$$\begin{matrix} 32 & 6 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$\hat{F}(u, v)$

- Kuantisasi ke *integer* membuat komponen frekuensi tinggi dibulatkan menjadi nol, sedangkan komponen frekuensi sisanya menjadi bilangan positif kecil dan bilangan negatif kecil.

- Proses kuantisasi inilah yang dinamakan *lossy*, karena ada informasi yang hilang akibat pembulatan.

- Pada proses penirmampatan (di dalam *decoder*), hampiran nilai F(u,v) diperoleh kembali dengan mengalikan $\hat{F}(u,v)$ dengan Q(u,v).

$$\tilde{F}(u,v) = \hat{F}(u,v) \times Q(u,v)$$

Contoh:

- An 8 × 8 block from the Y image of ‘Lena’

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

Original Image Block $f(i, j)$

515	65	-12	4	1	2	-8	5
-16	3	2	0	0	-11	-2	3
-12	6	11	-1	3	0	1	-2
-8	3	-4	2	-2	-3	-5	-2
0	-2	7	-5	4	0	-1	-4
0	-3	-1	0	4	1	-1	0
3	-2	-3	3	3	-1	-1	3
-2	5	-2	4	-2	2	-3	0

$F(u, v)$ DCT Coefficients

- Fig. 9.2: JPEG compression for a smooth image block.

Quantized DCT Coefficients

$$\begin{bmatrix} 32 & 6 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$\hat{F}(u, v)$

De-Quantized DCT Coefficients

$$\begin{bmatrix} 512 & 66 & -10 & 0 & 0 & 0 & 0 & 0 \\ -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$\tilde{F}(u, v)$

Reconstructed Image Block

$$\begin{bmatrix} 199 & 196 & 191 & 186 & 182 & 178 & 177 & 176 \\ 201 & 199 & 196 & 192 & 188 & 183 & 180 & 178 \\ 203 & 203 & 202 & 200 & 195 & 189 & 183 & 180 \\ 202 & 203 & 204 & 203 & 198 & 191 & 183 & 179 \\ 200 & 201 & 202 & 201 & 196 & 189 & 182 & 177 \\ 200 & 200 & 199 & 197 & 192 & 186 & 181 & 177 \\ 204 & 202 & 199 & 195 & 190 & 186 & 183 & 181 \\ 207 & 204 & 200 & 194 & 190 & 187 & 185 & 184 \end{bmatrix}$$

$\tilde{f}(i, j)$

Error

$$\begin{bmatrix} 1 & 6 & -2 & 2 & 7 & -3 & -2 & -1 \\ -1 & 4 & 2 & -4 & 1 & -1 & -2 & -3 \\ 0 & -3 & -2 & -5 & 5 & -2 & 2 & -5 \\ -2 & -3 & -4 & -3 & -1 & -4 & 4 & 8 \\ 0 & 4 & -2 & -1 & -1 & -1 & 5 & -2 \\ 0 & 0 & 1 & 3 & 8 & 4 & 6 & -2 \\ 1 & -2 & 0 & 5 & 1 & 1 & 4 & -6 \\ 3 & -4 & 0 & 6 & -2 & -2 & 2 & 2 \end{bmatrix}$$

$(i, j) = f(i, j) - \tilde{f}(i, j)$

Fig. 9.2 (cont'd): JPEG compression for a smooth image block.

- Another 8 × 8 block from the Y image of 'Lena'

70	70	100	70	87	87	150	187
85	100	96	79	87	154	87	113
100	85	116	79	70	87	86	196
136	69	87	200	79	71	117	96
161	70	87	200	103	71	96	113
161	123	147	133	113	113	85	161
146	147	175	100	103	103	163	187
156	146	189	70	113	161	163	197

f(i, j)

-80	-40	89	-73	44	32	53	-3
-135	-59	-26	6	14	-3	-13	-28
47	-76	66	-3	-108	-78	33	59
-2	10	-18	0	33	11	-21	1
-1	-9	-22	8	32	65	-36	-1
5	-20	28	-46	3	24	-30	24
6	-20	37	-28	12	-35	33	17
-5	-23	33	-30	17	-5	-4	20

F(u, v)

Fig. 9.3: JPEG compression for a textured image block.

$$\begin{bmatrix} -5 & -4 & 9 & -5 & 2 & 1 & 1 & 0 \\ -11 & -5 & -2 & 0 & 1 & 0 & 0 & -1 \\ 3 & -6 & 4 & 0 & -3 & -1 & 0 & 1 \\ 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$\hat{F}(u, v)$

$$\begin{bmatrix} -80 & -44 & 90 & -80 & 48 & 40 & 51 & 0 \\ -132 & -60 & -28 & 0 & 26 & 0 & 0 & -55 \\ 42 & -78 & 64 & 0 & -120 & -57 & 0 & 56 \\ 0 & 17 & -22 & 0 & 51 & 0 & 0 & 0 \\ 0 & 0 & -37 & 0 & 0 & 109 & 0 & 0 \\ 0 & -35 & 55 & -64 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$\tilde{F}(u, v)$

$$\begin{bmatrix} 70 & 60 & 106 & 94 & 62 & 103 & 146 & 176 \\ 85 & 101 & 85 & 75 & 102 & 127 & 93 & 144 \\ 98 & 99 & 92 & 102 & 74 & 98 & 89 & 167 \\ 132 & 53 & 111 & 180 & 55 & 70 & 106 & 145 \\ 173 & 57 & 114 & 207 & 111 & 89 & 84 & 90 \\ 164 & 123 & 131 & 135 & 133 & 92 & 85 & 162 \\ 141 & 159 & 169 & 73 & 106 & 101 & 149 & 224 \\ 150 & 141 & 195 & 79 & 107 & 147 & 210 & 153 \end{bmatrix}$$

$\tilde{f}(i, j)$

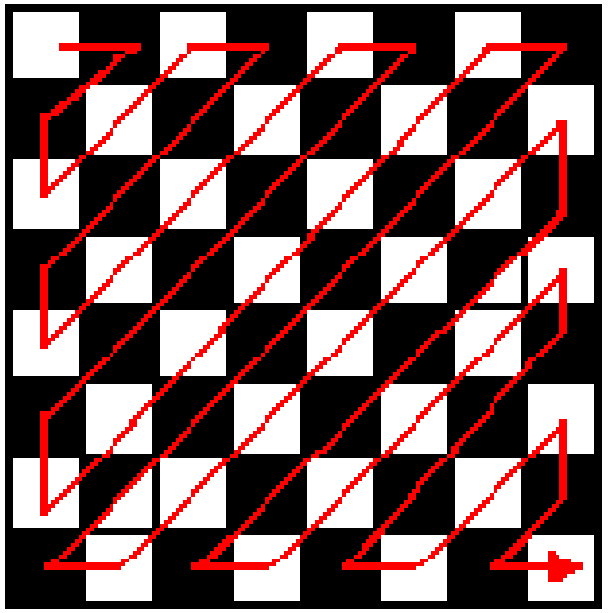
$$\begin{bmatrix} 0 & 10 & -6 & -24 & 25 & -16 & 4 & 11 \\ 0 & -1 & 11 & 4 & -15 & 27 & -6 & -31 \\ 2 & -14 & 24 & -23 & -4 & -11 & -3 & 29 \\ 4 & 16 & -24 & 20 & 24 & 1 & 11 & -49 \\ -12 & 13 & -27 & -7 & -8 & -18 & 12 & 23 \\ -3 & 0 & 16 & -2 & -20 & 21 & 0 & -1 \\ 5 & -12 & 6 & 27 & -3 & -2 & 14 & -37 \\ 6 & 5 & -6 & -9 & 6 & 14 & -47 & 44 \end{bmatrix}$$

$(i, j) = f(i, j) - \tilde{f}(i, j)$

Fig. 9.3 (cont'd): JPEG compression for a textured image block.

- Selanjutnya, koefisien DCT hasil kuantisasi dibaca secara zig-zag untuk membantu tahap *entropy coding*.

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63



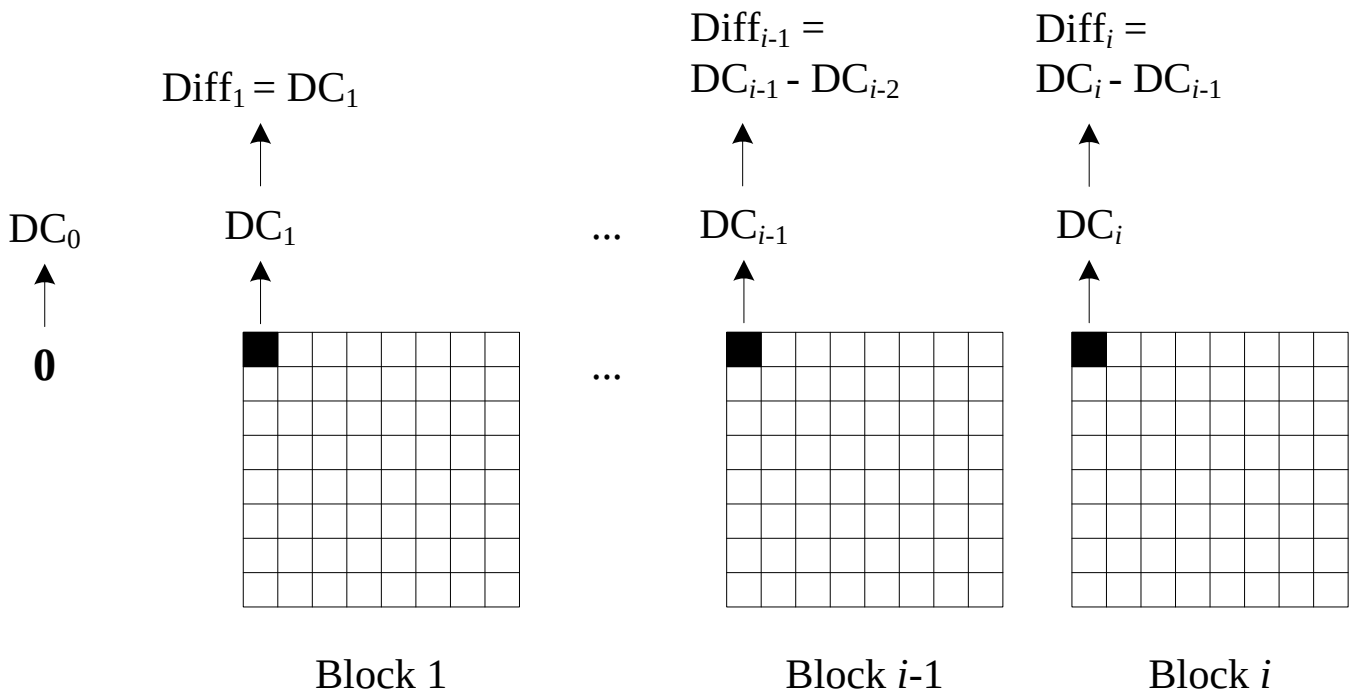
- Tahap *entropy coding* adalah *arithmetic coding* (RLE dan DPCM)dan *Huffman coding*, keduanya adalah metode *lossless compression*.

- *Huffman coding* mengkompresi hasil *arithmetic coding*. Pohon Huffman kemudian disimpan sebagai *header* file JPEG.

DPCM pada Koefisien DC

- Koefisien DC pada setiap blok dikodekan dengan DPCM sebagai berikut:

$$Diff_i = DC_{i+1} - DC_i \quad \text{dan} \quad Diff_0 = DC_0.$$

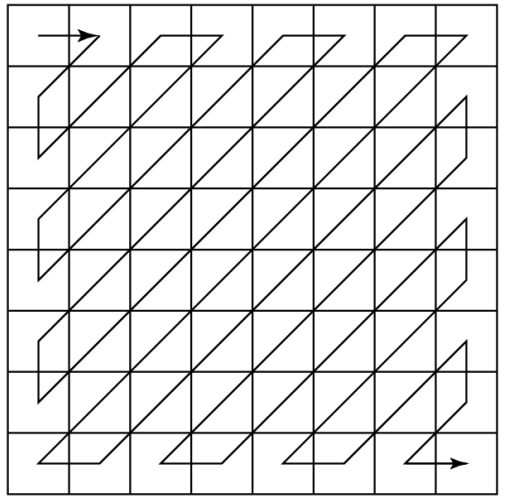


- Jika koefisien DC untuk 5 blok pertama adalah 150, 155, 149, 152, 144, maka DPCM akan menghasilkan 150, 5, -6, 3, -8.
- Selanjutnya, semua *Diff* tersebut dikodekan dengan Huffman Coding bersama-sama dengan koefisien AC

Run-length Encoding (RLE) pada Koefisien AC

- RLE bertujuan untuk menghasilkan himpunan *{#-zeros-to-skip, next non-zero value}*.

- Untuk membuatnya paling mungkin mencapai *run* nol yang panjang, maka dlakukan pemindaian zig-zag untuk mengubah matriks 8 × 8 menjadi vektor dengan 64 elemen.



$$\hat{F}(u, v)$$

$$\begin{matrix}
32 & 6 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}$$

Original Image

Compressed Image

- JPEG image compression menghasilkan artefak-artefak di dalam citra hasil pemampatan, namun masih dapat ditolerir secara visual

- *Less computational complexity*

Sumber*: Image Processing*, *Image Compression*, DR. FERDA ERNAWAN
Faculty of Computer Systems & Software Engineering, Pahang University

Metrik Pengukuran Pemampatan Citra

1. Nisbah (*ratio*) pemampatan

   Bermacam-macam rumus menghitung nisbah pemampatan

$$Nisbah1 = \frac{ukuran\ citra\ sesudah\ dimampatkan}{ukuran\ citra\ sebelum\ dimampatkan}\ \text{x}\ 100\%$$

artinya ukuran citra sekarang menjadi Nisbah1 (dalam persen) kali ukuran citra semula.

$$Nisbah2 = 100\% - \frac{ukuran\ citra\ sesudah\ dimampatkan}{ukuran\ citra\ sebelum\ dimampatkan}\ \text{x}\ 100\%$$

artinya citra sebanyak Nisbah2 (dalam persen) telah dimampatkan.

- Contoh: Citra semula berukuran 256×256 pixels, 8-bit per pixel, grayscale.

  Ukuran citra adalah 65536 byte (64 kb).

  Setelah dimampatkan, ukuran citra menjadi 40280 byte

  Nisbah = (40280/65536) x 100% = 61,5 %

  (artinya ukuran citra menjadi 61,5% dari ukuran semula)

  Nisbah = 100% - 61,5% = 38,5%

  (artinya 38,5% citra sudah dimampatkan)

2. Ukuran *fidelity*

- Kualitas sebuah citra bersifat subyektif dan relatif, bergantung pada pengamatan orang yang menilainya
- Kualitas hasil pemampatan dapat diukur secara kuantitatif dengan menggunakan besaran *PSNR* (*peak signal-to-noise ratio*).
- *PSNR* dihitung untuk mengukur perbedaan antara citra semula dengan citra hasil pemampatan

$$PSNR = 20 \times \log_{10}\left(\frac{b}{rms}\right)$$

b adalah nilai sinyal terbesar (pada citra dengan 256 derajat keabuan, $b = 255$)
rms (*root mean square*) adalah akar pangkat dua dari selisih antara citra semula dengan citra hasil pemampatan

$$rms = \sqrt{\frac{1}{\text{Lebar} \times \text{Tinggi}} \sum_{i=1}^{N} \sum_{j=1}^{M} (f_{ij} - f'_{ij})^2}$$

f dan f’ masing-masing menyatakan nilai pixel citra semula dan nilai pixel citra hasil pemampatan

- *PSNR* memiliki satuan *decibel* (dB).

- *PSNR* berbanding terbalik dengan *rms*. Nilai *rms* yang rendah yang menyiratkan bahwa citra hasil pemampatan tidak jauh berbeda dengan citra semula, sehingga menghasilkan *PSNR* yang tinggi, yang berarti kualitas pemampatannya bagus.

- Nilai rms yang tinggi menyatakan galat yang besar akibat pemampatan, sehinga menghasilkan PSNR yang rendah.

- Semakin besar nilai *PSNR*, semakin bagus kualitas pemampatannya.

- Dalam prakteknya, nilai PSNR $\geq$ 30 menyatakan kualitas citra yang sudah dianggap baik. Di bawah 30 itu dikatakan citra mengalami degradasi yang semakin besar dengan menurunnya PSNR.

peppers.bmp, 256 x 256
(193 KB)

peppers.jpg, 256 x 256
(52.2 KB), JPEG Quality = 5

```
>> ref = imread('peppers512.bmp');
>> A = imread('peppers512-med.jpg');
>> psnr = psnr(A, ref)

psnr =

   35.08
```

peppers2.jpg, 256 x 256
(24 KB), JPEG Quality = 1

```
>> ref = imread('peppers512.bmp');
>> A = imread('peppers512-low.jpg');
>> psnr = psnr(A, ref)

psnr =

   30.5051
```