

# PENGOLAHAN CITRA DIGITAL

IMAGES COMPRESSION

GKV - IF 2020

# IMAGES COMPRESSION

- Pendahuluan
- Kriteria, Jenis dan Klasifikasi Metode Kompresi
- Huffman Code
- RLE
- JPEG

## Pemampatan citra (*image compression*).

- Pada proses ini, citra dalam representasi tidak mampat dikodekan dengan representasi yang meminimumkan kebutuhan memori.
- Citra dengan format *bitmap* pada umumnya tidak dalam bentuk mampat.
- *Citra yang sudah* dimampatkan disimpan ke dalam arsip dengan format tertentu. Kita mengenal format JPG, GIF sebagai format citra yang sudah dimampatkan.

## Penirmampatkan citra (*image decompression*).

- Pada proses ini, citra yang sudah dimampatkan harus dapat dikembalikan lagi (*decoding*) menjadi representasi yang tidak mampat.
- Proses ini diperlukan jika citra tersebut ditampilkan ke layar atau disimpan ke dalam arsip dengan format tidak mampat.
- Dengan kata lain, penirmampatan citra mengembalikan citra yang termampatkan menjadi data *bitmap*.

# Aplikasi Pemampatan Citra

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

- Aplikasi pengiriman gambar lewat *fax, video conference, pengiriman data medis, pengiriman gambar dari satelit luar angkasa, pengiriman gambar via telepon genggam, download gambar dari internet, dan sebagainya.*

Penyimpanan data (*data storing*) di dalam media sekunder (*storage*)

- Aplikasi nya antara lain aplikasi basisdata gambar, *office automation, video storage (seperti VCD, DVD)*

# Kriteria Pemampatan Citra

Waktu pemampatan dan penirmampatan (*decompression*).

- Waktu pemampatan citra dan penirmampatannya sebaiknya cepat

Kebutuhan memori.

- Memori yang dibutuhkan untuk merepresentasikan citra seharusnya berkurang secara signifikan.

Kualitas pemampatan (fidelity)

- Informasi yang hilang akibat pemampatan seharusnya seminimal mungkin sehingga kualitas hasil pemampatan tetap dipertahankan. Alat ukur PSNR.

Format keluaran

- Format citra hasil pemampatan sebaiknya cocok untuk pengiriman dan penyimpanan data.

# Jenis Pemampatan Citra

## Pendekatan statistik.

- Pemampatan citra didasarkan pada frekuensi kemunculan derajat keabuan *pixel di dalam seluruh bagian gambar*.
- Contoh metode: *Huffman Coding*.

## Pendekatan ruang

- Pemampatan citra didasarkan pada hubungan spasial antara *pixel-pixel di dalam* suatu kelompok yang memiliki derajat keabuan yang sama di dalam suatu daerah di dalam gambar.
- Contoh metode: *Run-Length Encoding*.

## Pendekatan kuantisasi

- Pemampatan citra dilakukan dengan mengurangi jumlah derajat keabuan yang tersedia.
- Contoh metode: metode pemampatan kuantisasi.

## Pendekatan fraktal

- Pemampatan citra didasarkan pada kenyataan bahwa kemiripan bagianbagian di dalam citra dapat dieksploitasi dengan suatu matriks transformasi.
- Contoh metode: *Fractal Image Compression*.

# Klasifikasi Metode Pemampatan

## Metode *lossless*

- Metode *lossless* selalu menghasilkan citra hasil penirmampatan yang tepat sama dengan citra semula, *pixel per pixel*. Tidak ada informasi yang hilang akibat pemampatan.
- Nisbah (*ratio*) pemampatan citra metode *lossless* sangat rendah.
- Contoh metode *lossless* adalah metode Huffman.

## Metode *lossy*

- Metode *lossy* menghasilkan citra hasil pemampatan yang hampir sama dengan citra semula.
- Ada informasi yang hilang akibat pemampatan, tetapi dapat ditolerir oleh persepsi mata.
- Mata tidak dapat membedakan perubahan kecil pada gambar.
- Metode pemampatan *lossy* menghasilkan nisbah pemampatan yang tinggi daripada metode *lossless*.
- Contoh metode *lossy* adalah metode JPEG dan metode fraktal.



# Metode Pemampatan Huffman

Metode pemampatan Huffman menggunakan prinsip bahwa nilai (atau derajat) keabuan yang sering muncul di dalam citra akan dikodekan dengan jumlah bit yang lebih sedikit sedangkan nilai keabuan yang frekuensi kemunculannya sedikit dikodekan dengan jumlah bit yang lebih panjang.

# Algoritma metode Huffman

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

# Contoh Huffman Code

Misalkan terdapat citra yang berukuran 64x64 dengan 8 derajat keabuan ( $k$ ) dan jumlah seluruh pixel ( $n$ ) =  $64 \times 64 = 4096$

$k$	$n_k$	$p(k) = n_k/n$
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

# Contoh Huffman Code

1. 

7:0.02
--------

6:0.03
--------

5:0.06
--------

4:0.08
--------

3:0.16
--------

0:0.19
--------

2:0.21
--------

1:0.25
--------

2. 

7:0.02
--------

6:0.03
--------

5:0.06
--------


4:0.08
--------

3:0.16
--------

0:0.19
--------

2:0.21
--------

1:0.25
--------



```
graph TD; A(76:0.05) --- B[7:0.02]; A --- C[6:0.03]
```

A Huffman tree diagram showing a root node in an oval labeled '76:0.05'. Two lines branch down from the root to two leaf nodes in rectangles labeled '7:0.02' and '6:0.03'.

3. 

4:0.08
--------

7:0.02
--------

6:0.03
--------

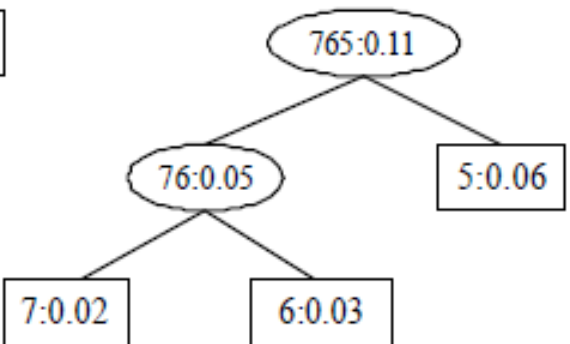
5:0.06
--------

3:0.16
--------

0:0.19
--------

2:0.21
--------

1:0.25
--------

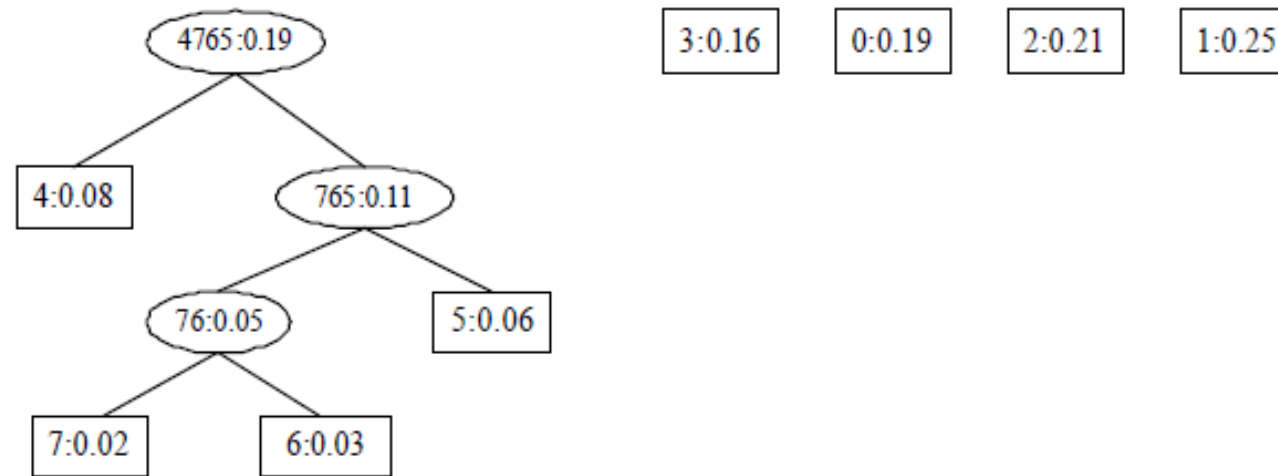


```
graph TD; A(765:0.11) --- B(76:0.05); A --- C[5:0.06]; B --- D[7:0.02]; B --- E[6:0.03]
```

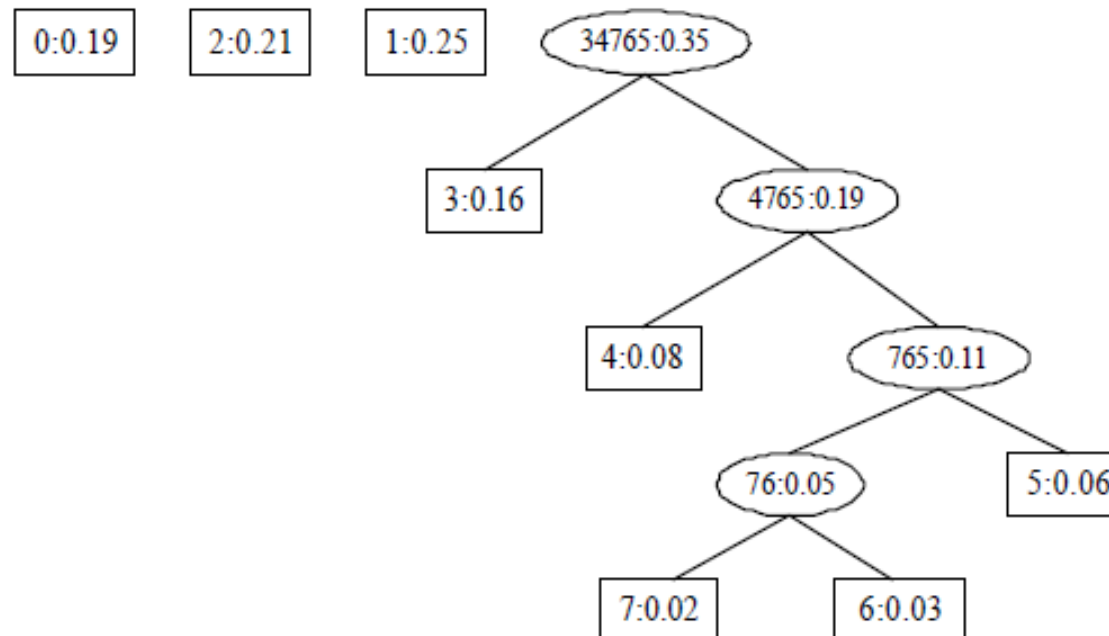
A Huffman tree diagram showing a root node in an oval labeled '765:0.11'. Two lines branch down from the root to a node in an oval labeled '76:0.05' and a leaf node in a rectangle labeled '5:0.06'. From the '76:0.05' node, two lines branch down to two leaf nodes in rectangles labeled '7:0.02' and '6:0.03'.

# Contoh Huffman Code

4.

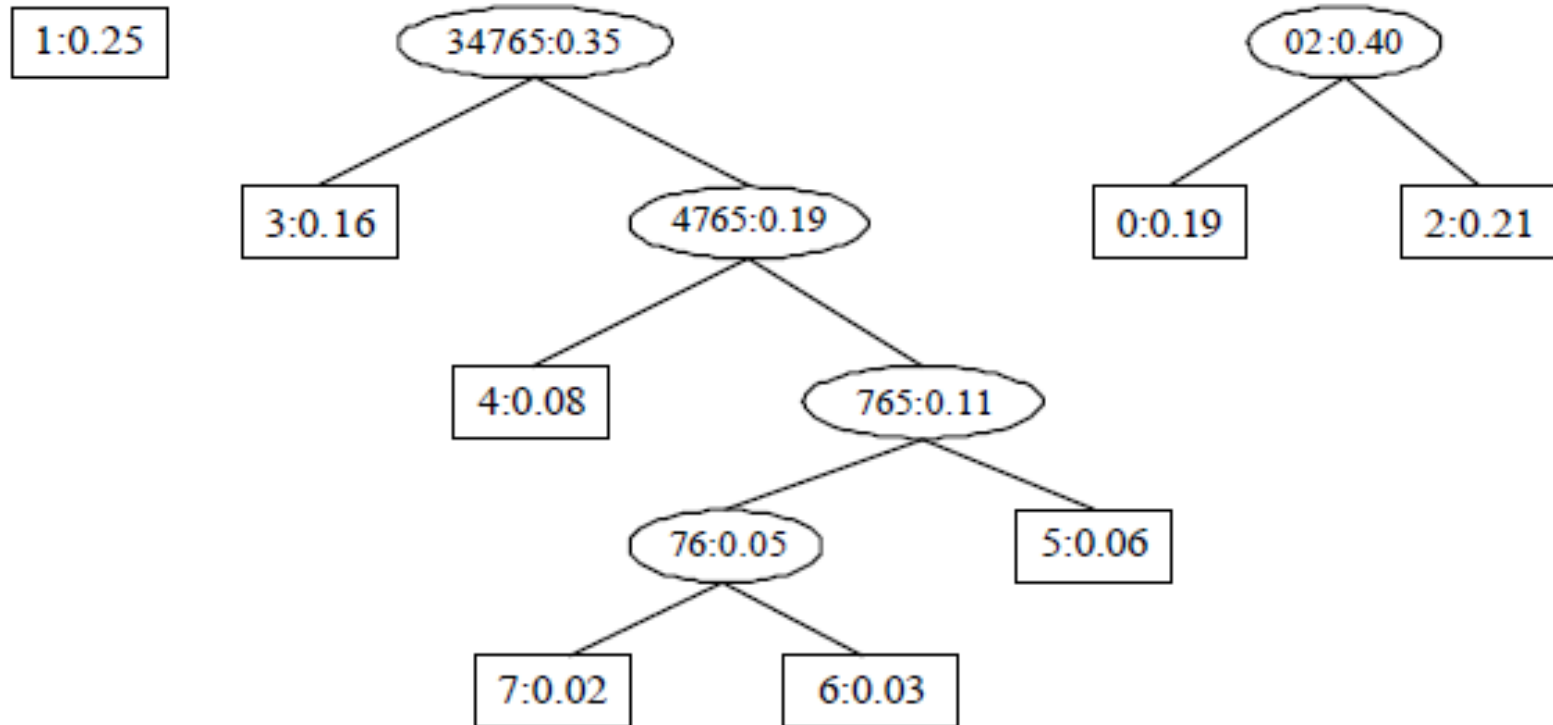


5.



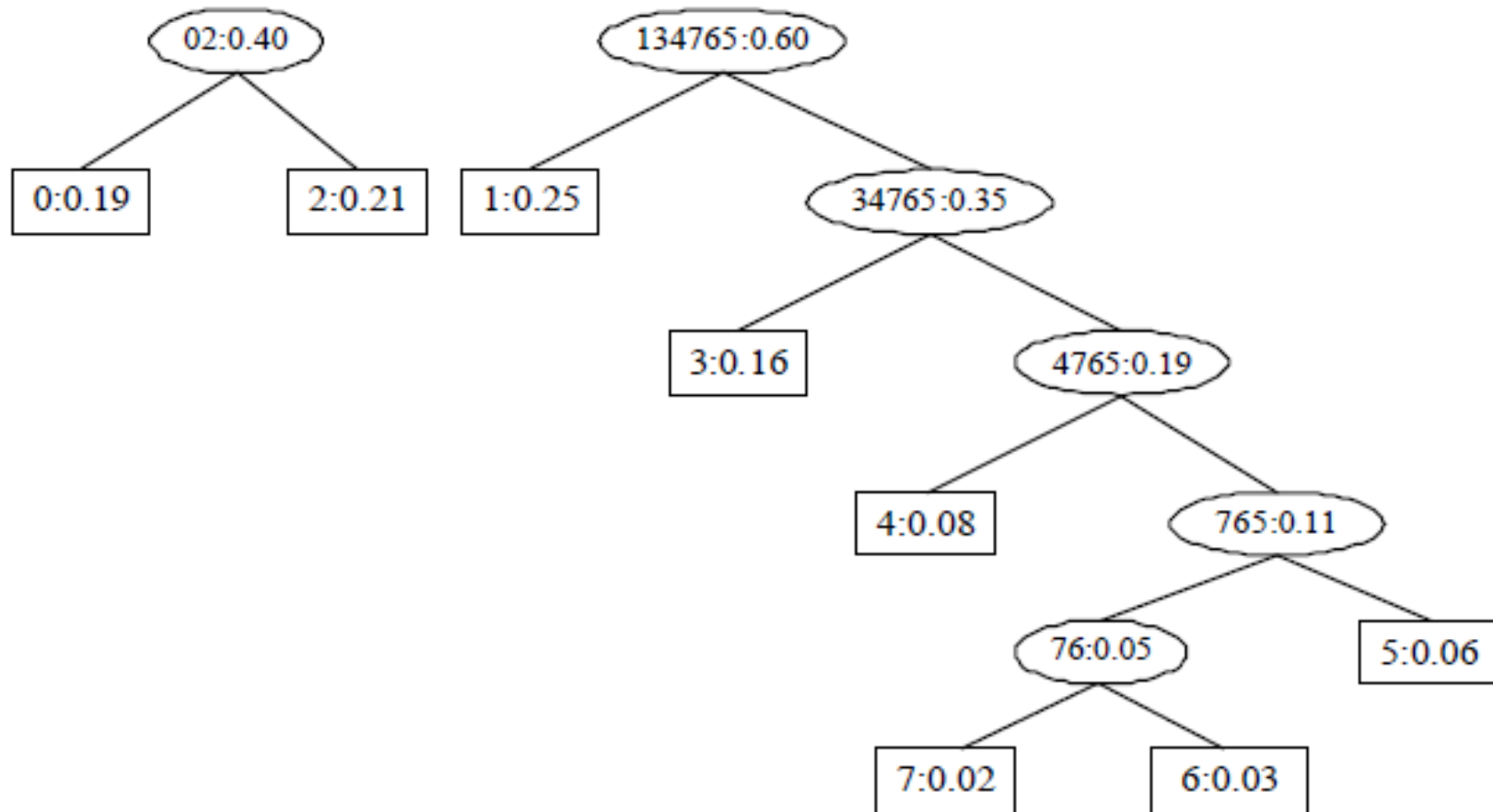
# Contoh Huffman Code

6.



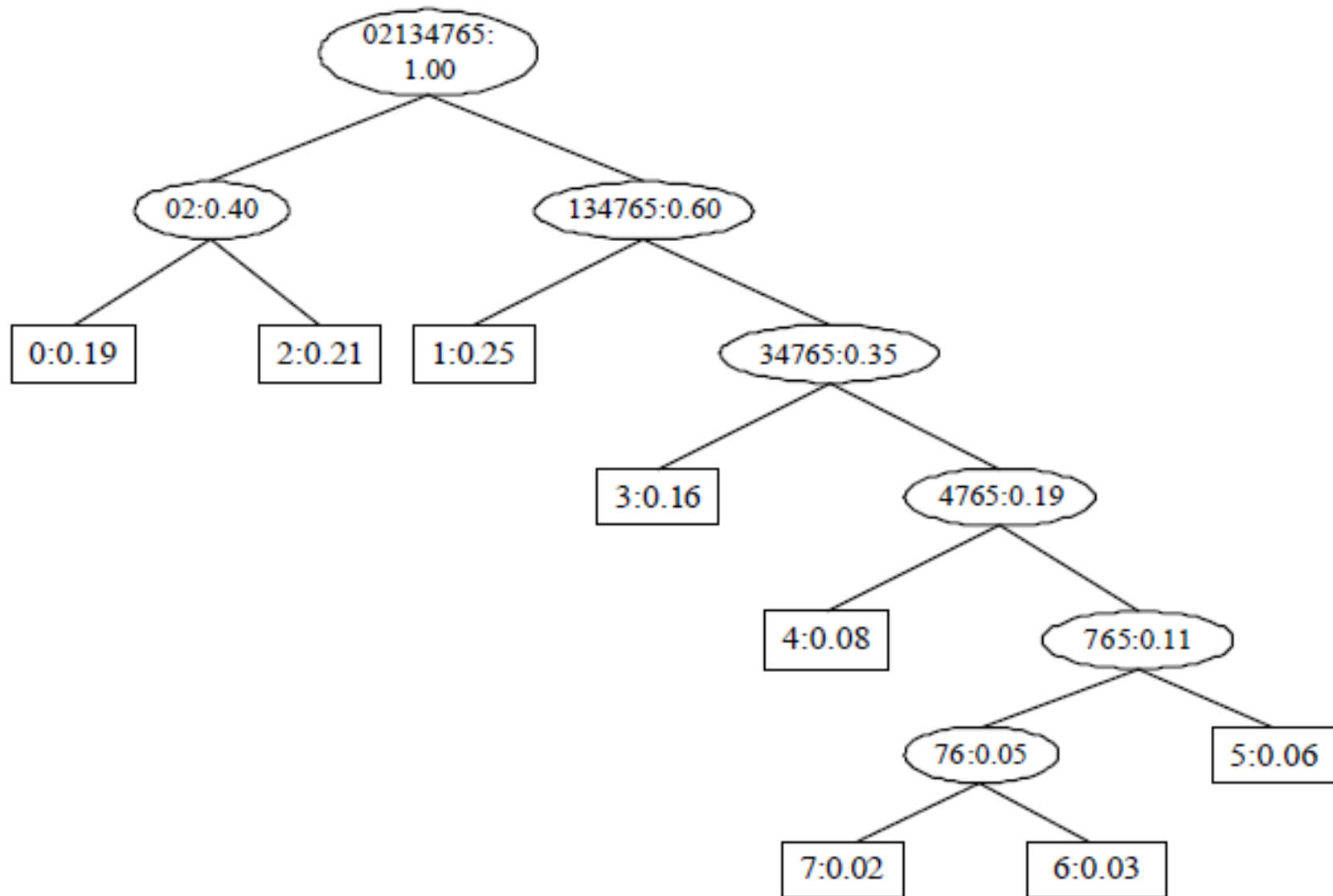
# Contoh Huffman Code

7.



# Contoh Huffman Code

8.





# Contoh Huffman Code

Kode Huffman yang dihasilkan untuk derajat keabuan 0-7

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

Ukuran citra asli =  $4096 \times 3 \text{ bit} = 12288 \text{ bit}$

Setelah dikompresi =  $(790 \times 2 \text{ bit}) + (1023 \times 2 \text{ bit}) + (850 \times 2 \text{ bit}) +$   
 $(656 \times 3 \text{ bit}) + (329 \times 4 \text{ bit}) + (245 \times 5 \text{ bit}) +$   
 $(122 \times 6 \text{ bit}) + (81 \times 6 \text{ bit}) = 11053 \text{ bit}$

Nisbah pemampatan =  $(100\% - \frac{11053}{12288} \times 100\%) = 10\%$  , yang artinya 10% dari citra semula telah dimampatkan. ■

# Tentukan Huffman Code

<b>Derajat Keabuan</b>	<b>probabilitas</b>
0	0,3
1	0,1
2	0,2
3	0,06
4	0,09
5	0,07
6	0,03
7	0,15

# Metode Pemampatan

## *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, sedangkan nilai kedua  $(q)$  menyatakan jumlah *pixel* berurutan yang memiliki derajat keabuan tersebut (dinamakan *run length*).

# *Run-Length Encoding (RLE)*

Tinjau citra 10 x 10 *pixel dengan 8 derajat keabuan yang dinyatakan sebagai matriks derajat keabuan sebagai berikut*

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

# *Run-Length Encoding (RLE)*

Pasangan nilai untuk setiap baris *run* yang dihasilkan dengan metode pemampatan RLE:

(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 atau  $31 \times 2 = 62$  nilai.

# *Run-Length Encoding (RLE)*

- Ukuran citra sebelum pemampatan (1 derajat keabuan = 3 bit) adalah  $100 \times 3 \text{ bit} = 300 \text{ bit}$
- Sedangkan ukuran citra setelah pemampatan (derajat keabuan = 3 bit, *run length* = 4 bit) :

$$(31 \times 3) + (31 \times 4) \text{ bit} = 217 \text{ bit}$$

- Nisbah pemampatan =  $(100\% - (217/300) \times 100\%)$   
 $= 27.67\%$

(27.67% dari citra semula telah dimampatkan)

# Kompresi JPEG

- JPEG (***Joint Photographics Expert Group***) merupakan kompresi citra pertama yang memiliki standar internasional.
- Teknik kompresi ini merupakan hasil kerja sama antara ITU (International Telecommunication Union), ISO (International Organization for Standardization) dan IEC (International Electrotechnical Commission)
- JPEG memanfaatkan keterbatasan mata manusia dalam melihat warna.
- Mata manusia tidak sensitif terhadap perubahan warna yang kecil, dibandingkan dengan perubahan kecerahan (*brightness*) yang kecil.
- Kompresi ini sangat cocok sekali diaplikasikan pada foto-foto digital (*Digital Images*).

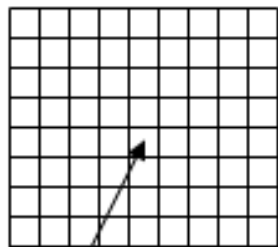
## Algoritma – JPEG (Encoding)

- Sebagai inputan dalam *encoding*, *pixel-pixel* dalam matriks citra dikelompokkan menjadi blok matriks 8x8
- Untuk setiap blok ini, tiap-tiap pixelnya dikurangi 128, kemudian ditransformasi menggunakan DCT, sehingga menghasilkan 64 koefisien DCT
- Proses kuantisasi, dengan membagi tiap elemen dalam matriks yang telah dikonversi dengan tabel kuantisasi dan bulatkan ke integer terdekat.
- Membentuk vektor dengan men-*scan AC Coefficient* matriks hasil kuantisasi secara zig-zag
- RLE atau Huffman Code untuk kompresi



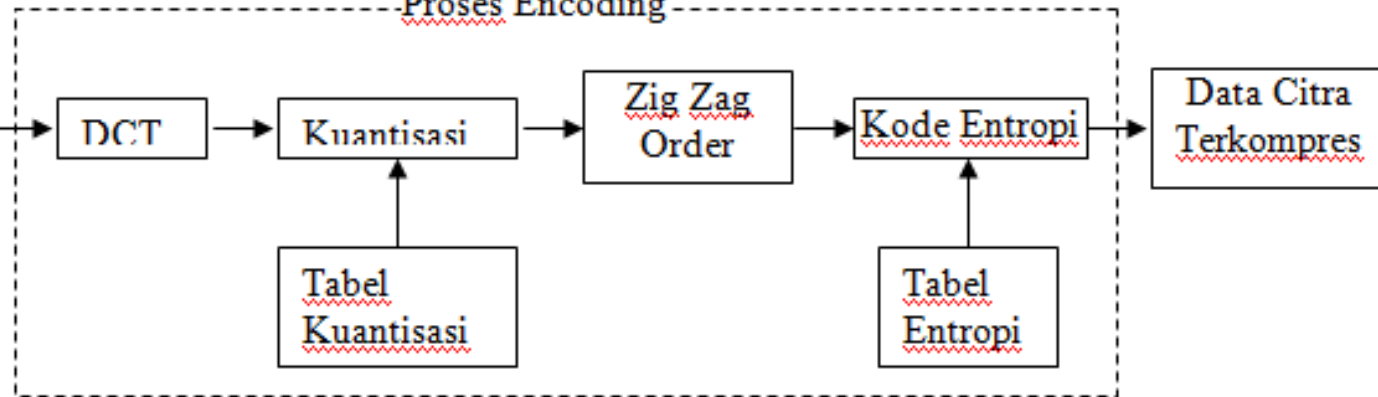
# JPEG - Encoding

Data Citra Input



Blok 8 x 8

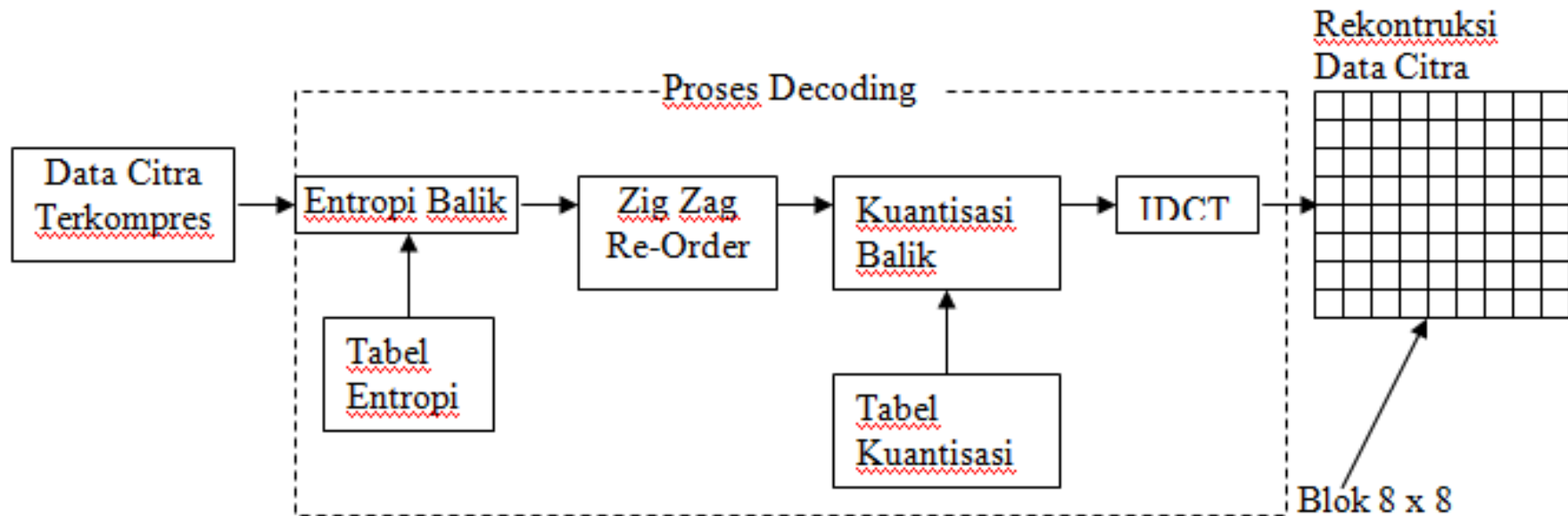
Proses Encoding



## Algoritma – JPEG (Decoding)

- Lakukan *Huffman Decoding* yaitu proses membalik dari kode bit-bit proses pengkodean Huffman menjadi nilai semula
- *Inverse* zig-zag
- Dekuantisasi dengan mengkalikan matriks dengan tabel kuantisasi *Inverse* DCT yaitu melakukan proses transformasi invers DCT untuk mendapatkan nilai – nilai piksel kembali.
- Tambahkan masing-masing kelompok 8 x 8 piksel dengan 128
- Gabung kembali kelompok-kelompok piksel menjadi sebuah satu kesatuan matriks awal.

# JPEG - Decoding



# JPEG-1



110	110	118	118	121	126	131	131
108	111	125	122	120	125	134	135
106	119	129	127	125	127	138	144
110	126	130	133	133	131	141	148
115	116	119	120	122	125	137	139
115	106	99	110	107	116	130	127
110	91	82	101	99	104	120	118
103	76	70	95	92	91	107	106

-18	-18	-10	-10	-7	-2	3	3
-20	-17	-3	-6	-8	-3	6	7
-22	-9	1	-1	-3	-1	10	16
-18	-2	2	5	5	3	13	20
-13	-12	-9	-8	-6	-3	9	11
-13	-22	-29	-18	-21	-12	2	-1
-18	-37	-46	-27	29	-24	-8	-10
-25	-52	-58	-33	-36	-37	-21	-22

Matriks 8x8 (kiri) dan hasilnya setelah dikurang 128 (kanan)

# JPEG-2

-18	-18	-10	-10	-7	-2	3	3
-20	-17	-3	-6	-8	-3	6	7
-22	-9	1	-1	-3	-1	10	16
-18	-2	2	5	5	3	13	20
-13	-12	-9	-8	-6	-3	9	11
-13	-22	-29	-18	-21	-12	2	-1
-18	-37	-46	-27	29	-24	-8	-10
-25	-52	-58	-33	-36	-37	-21	-22

DCT



-89.00	-63.47	18.21	-6.85	7.50	13.45	-7.00	0.13
74.14	-2.90	-19.93	-21.04	-17.88	-10.81	8.29	5.26
-63.65	3.10	5.08	14.82	10.12	9.33	1.31	-0.62
3.73	2.85	6.67	8.99	-3.38	1.54	1.04	-0.62
2.50	0.57	-4.46	0.52	3.00	-2.89	-0.32	1.33
7.52	-1.80	-0.63	-0.10	0.41	-3.21	-2.74	-2.07
-3.40	0.43	0.81	0.28	-0.40	-0.19	-0.58	-1.09
-2.26	-0.88	1.73	0.23	-0.21	-0.12	1.23	1.61

# JPEG-3

- *DCT*:  $C(p, q) = \alpha_p \alpha_q \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} I(m, n) \cos \frac{\pi(2m+1)p}{2N} \cos \frac{\pi(2n+1)q}{2N}$
- *IDCT*:  $I(m, n) = \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q C(p, q) \cos \frac{\pi(2m+1)p}{2N} \cos \frac{\pi(2n+1)q}{2N}$
- Keterangan: Citra berukuran  $M \times N$

$$0 \leq p \leq M - 1 \quad 0 \leq q \leq N - 1$$


$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}} & , p = 0 \\ \sqrt{\frac{2}{M}} & , 1 \leq p \leq M - 1 \end{cases} \quad \alpha_q = \begin{cases} \frac{1}{\sqrt{N}} & , q = 0 \\ \sqrt{\frac{2}{N}} & , 1 \leq q \leq N - 1 \end{cases}$$

# JPEG-4

## Piksel 8x8

[illegible]

DCT



## Koefisien DCT

Figure 1 shows a 7x7 grid representing a 2D lattice. The grid is divided into three regions:  $F_L$  (top-left, white),  $F_M$  (middle, gray), and  $F_H$  (bottom-right, white). The gray region  $F_M$  is a 3x3 square in the center. The white region  $F_L$  is the top-left 4x4 square. The white region  $F_H$  is the bottom-right 3x3 square.

# JPEG-5

-89.00	-63.47	18.21	-6.85	7.50	13.45	-7.00	0.13
74.14	-2.90	-19.93	-21.04	-17.88	-10.81	8.29	5.26
-63.65	3.10	5.08	14.82	10.12	9.33	1.31	-0.62
3.73	2.85	6.67	8.99	-3.38	1.54	1.04	-0.62
2.50	0.57	-4.46	0.52	3.00	-2.89	-0.32	1.33
7.52	-1.80	-0.63	-0.10	0.41	-3.21	-2.74	-2.07
-3.40	0.43	0.81	0.28	-0.40	-0.19	-0.58	-1.09
-2.26	-0.88	1.73	0.23	-0.21	-0.12	1.23	1.61

-6	-6	2	0	0	0	0	0
6	0	-1	-1	-1	0	0	0
-5	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	0	0	0	0	0



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

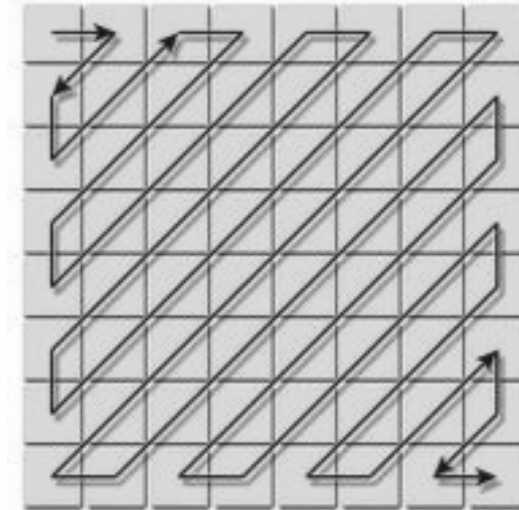
Matriks Kuantisasi JPEG



Bulatkan ke integer terdekat



# JPEG-6

[illegible]

Sehingga dari vektor hasil dari *zig-zag scan* untuk *AC coefficient* yaitu :

-6 -6 6 -5 0 2 0 -1 0 0 0 0 0 -1 0 0 -1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0

## RLE atau Huffman Code dari vektor di atas

# JPEG-7

