25 - Pemampatan Citra

(Bagian 1)

IF4073 Pemrosesan Citra Digital

Oleh: Rinaldi Munir

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

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

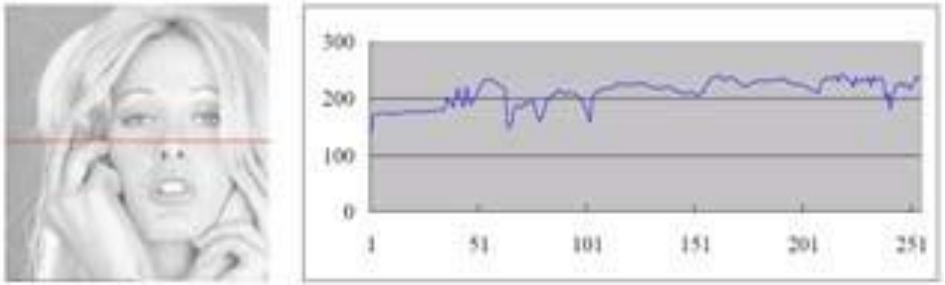
Symbol r_k	Probability $p_r(r_k)$	Code 2	Length $l_2(r_k)$
$r_0 = A$	0.19	11	2
$r_1 = B$	0.25	01	2
$r_2 = C$	0.21	10	2
$r_3 = D$	0.16	001	3
$r_4 = E$	0.08	0001	4
$r_5 = F$	0.06	00001	5
$r_6 = G$	0.03	000001	6
$r_7 = H$	0.02	000000	6

- 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

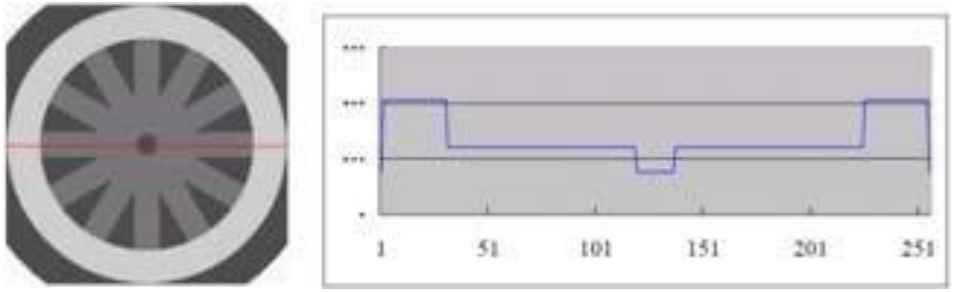
- Nisbah pemampatan= 2.7/3 = 0.9 atau 90%
- Artinya menjadi 90% dari rata-rata ukuran bit semula

Spatial redundancy

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

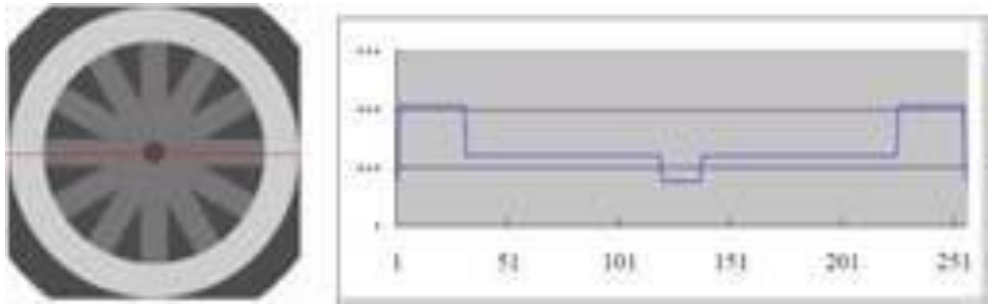


Nilai-nilai *pixel* citra roda pada baris 128:



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 = (7x16) / (256 x 8) = 0.055 atau 5,5%
Artinya menjadi 5,5% dariukuran semula, atau 94,5% telah dimampatkan

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:

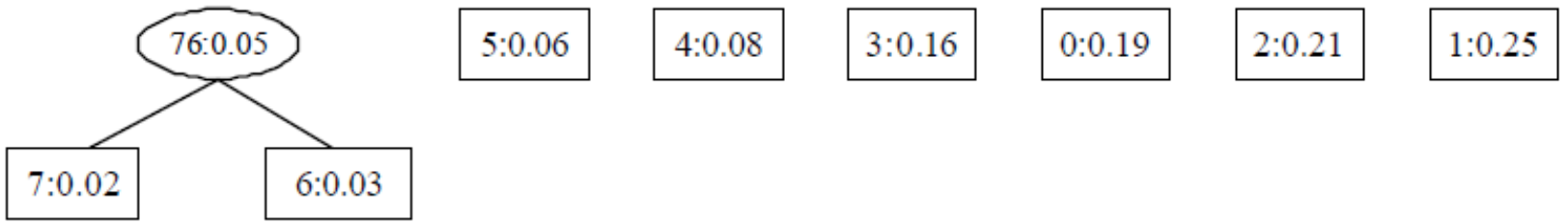
1. Setiap nilai keabuan dinyatakan sebagai simpul. Setiap simpul di-*assign* dengan frekuensi kemunculan nilai keabuan tersebut.
2. Urutkan secara menaik (*ascending order*) simpul-simpul berdasarkan frekuensi kemunculannya.
3. Gabung dua buah simpul yang mempunyai frekuensi kemunculan paling kecil pada 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.

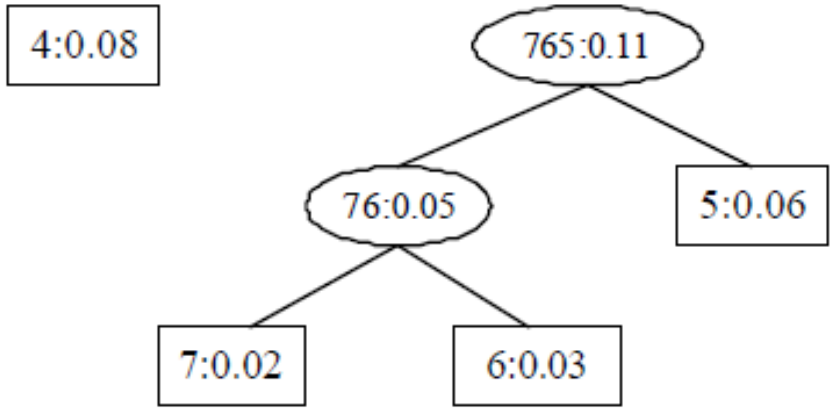
k	n_k	$p(k) = n_k/n$
0	790	0.19
1	1023	0.25
2	850	0.21
3	656	0.16
4	329	0.08
5	245	0.06
6	122	0.03
7	81	0.02

Proses pembentuan pohon Huffman:

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

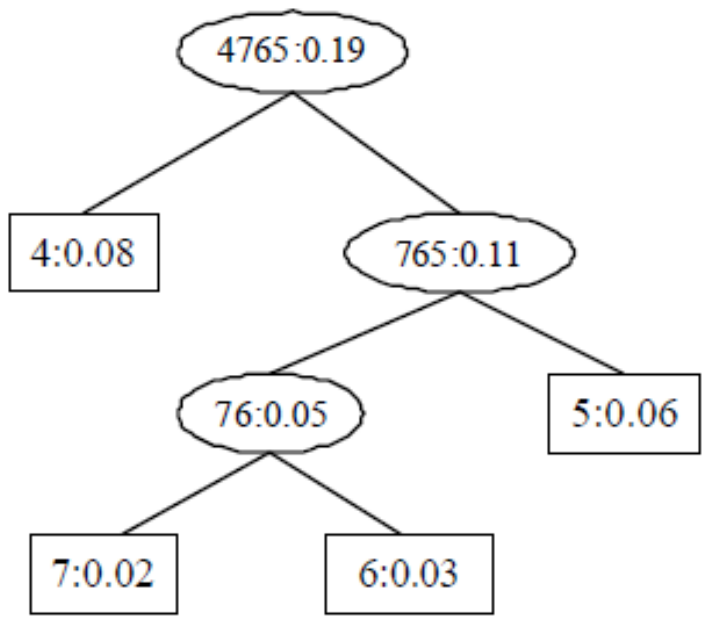
2. 

3.



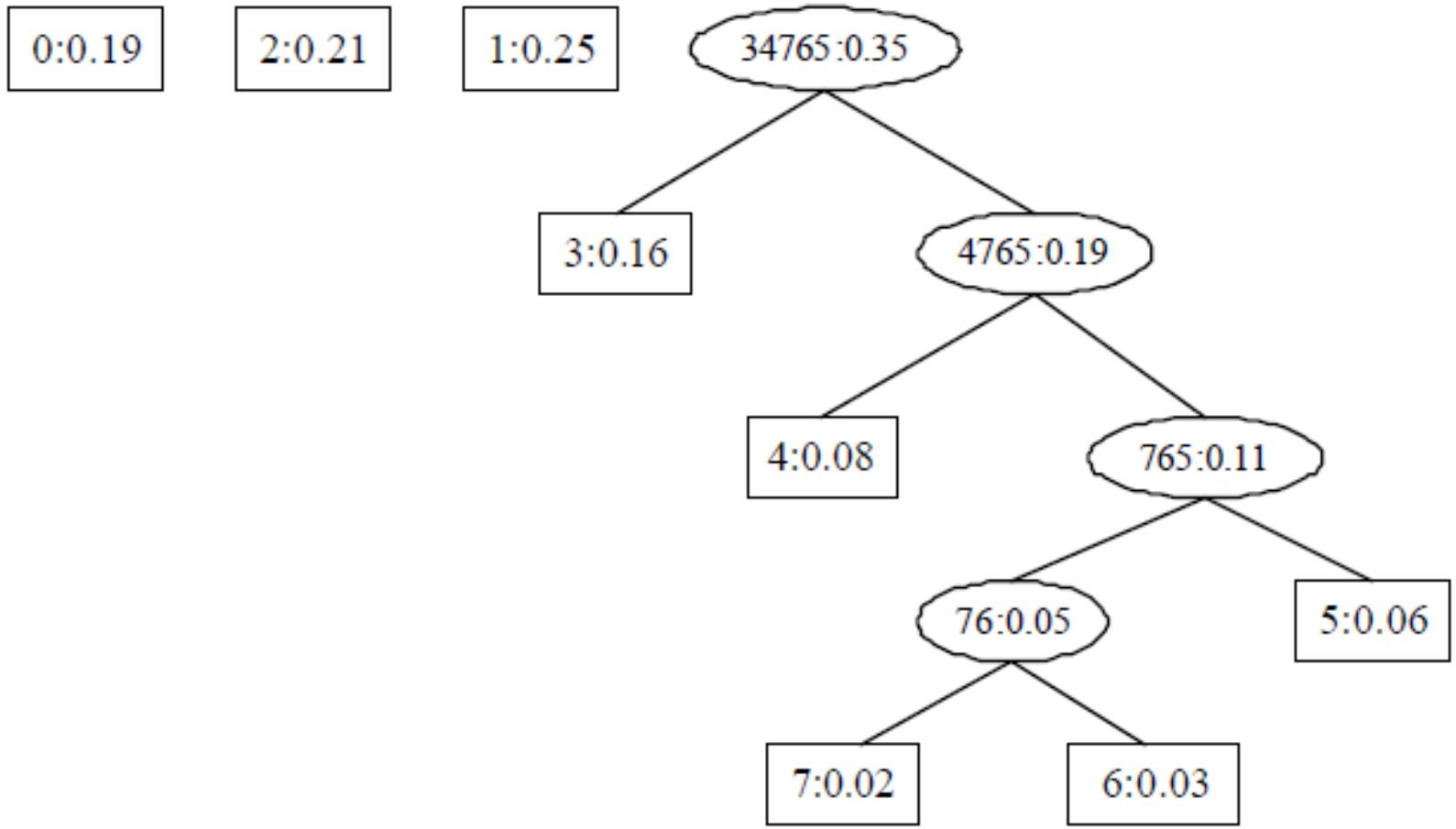
3:0.16 0:0.19 2:0.21 1:0.25

4.

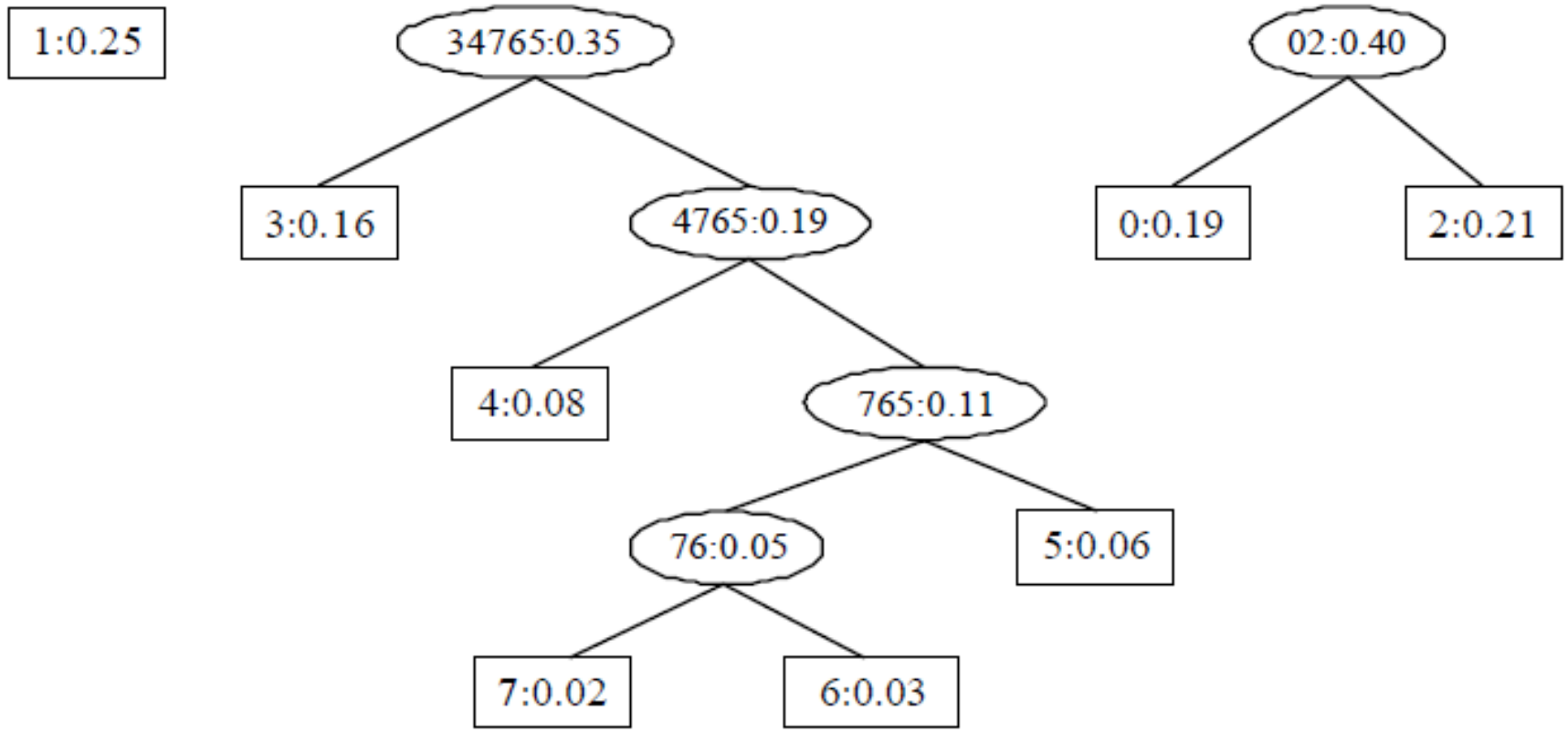


3:0.16 0:0.19 2:0.21 1:0.25

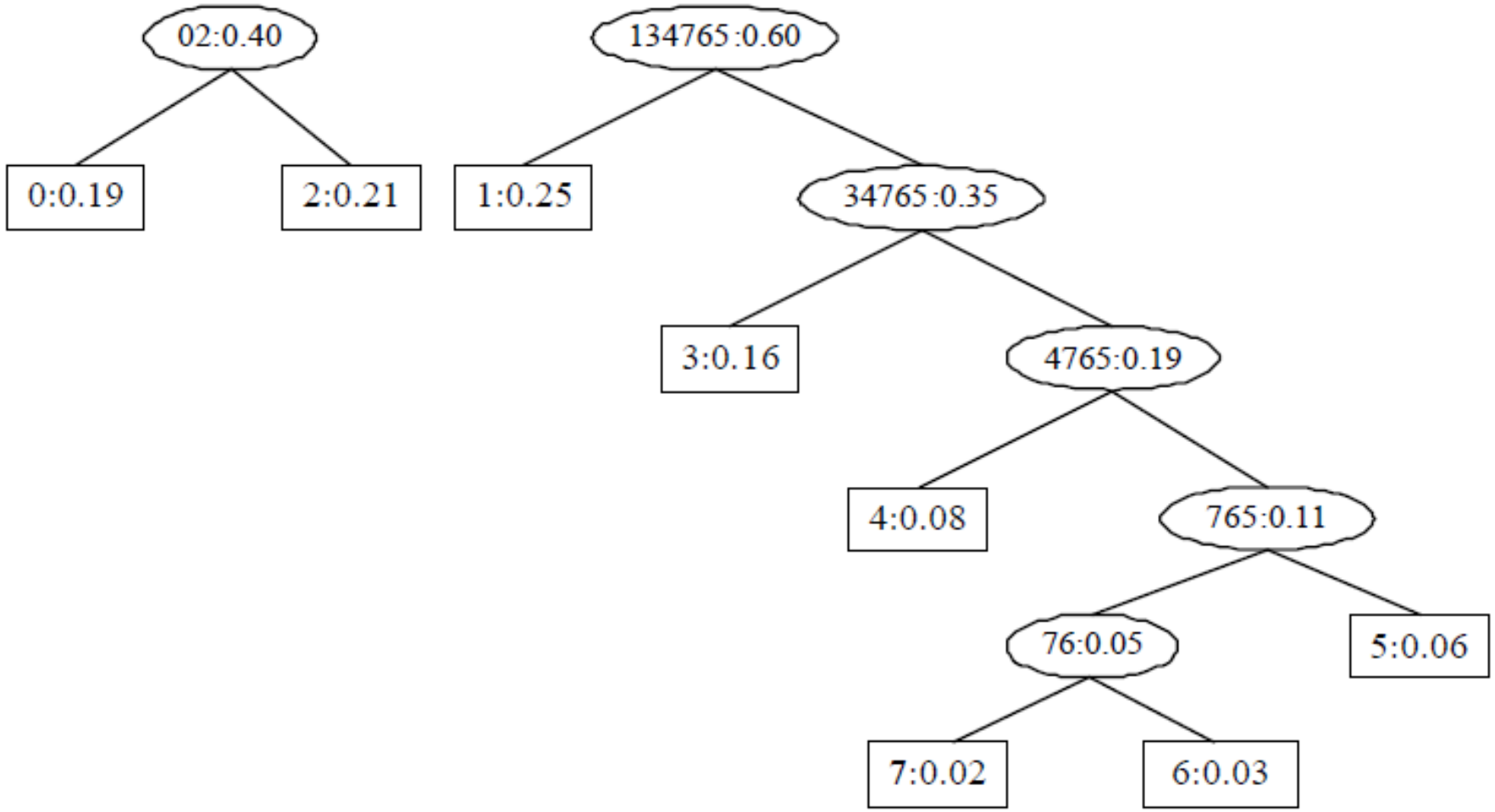
5.


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

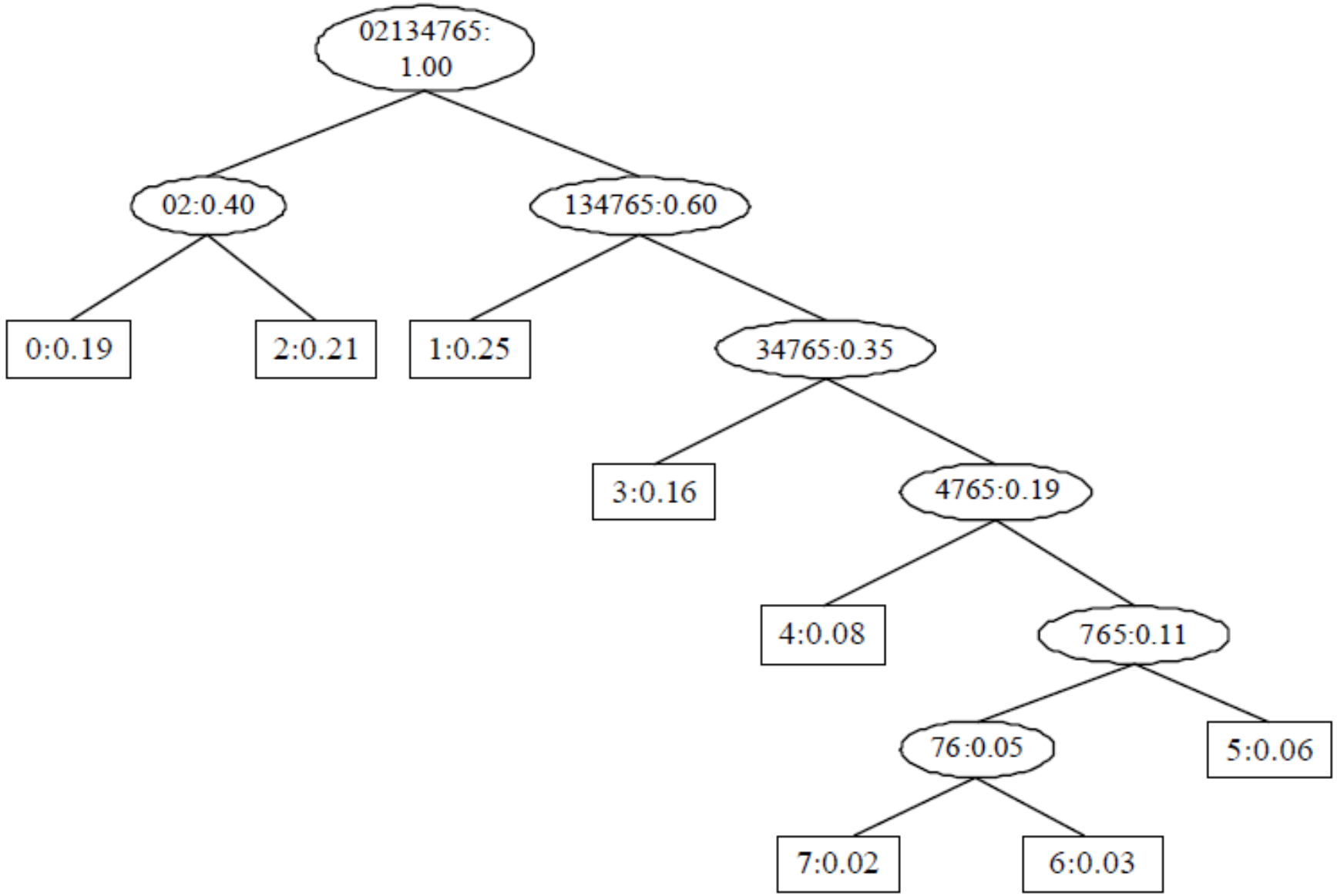
6.


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

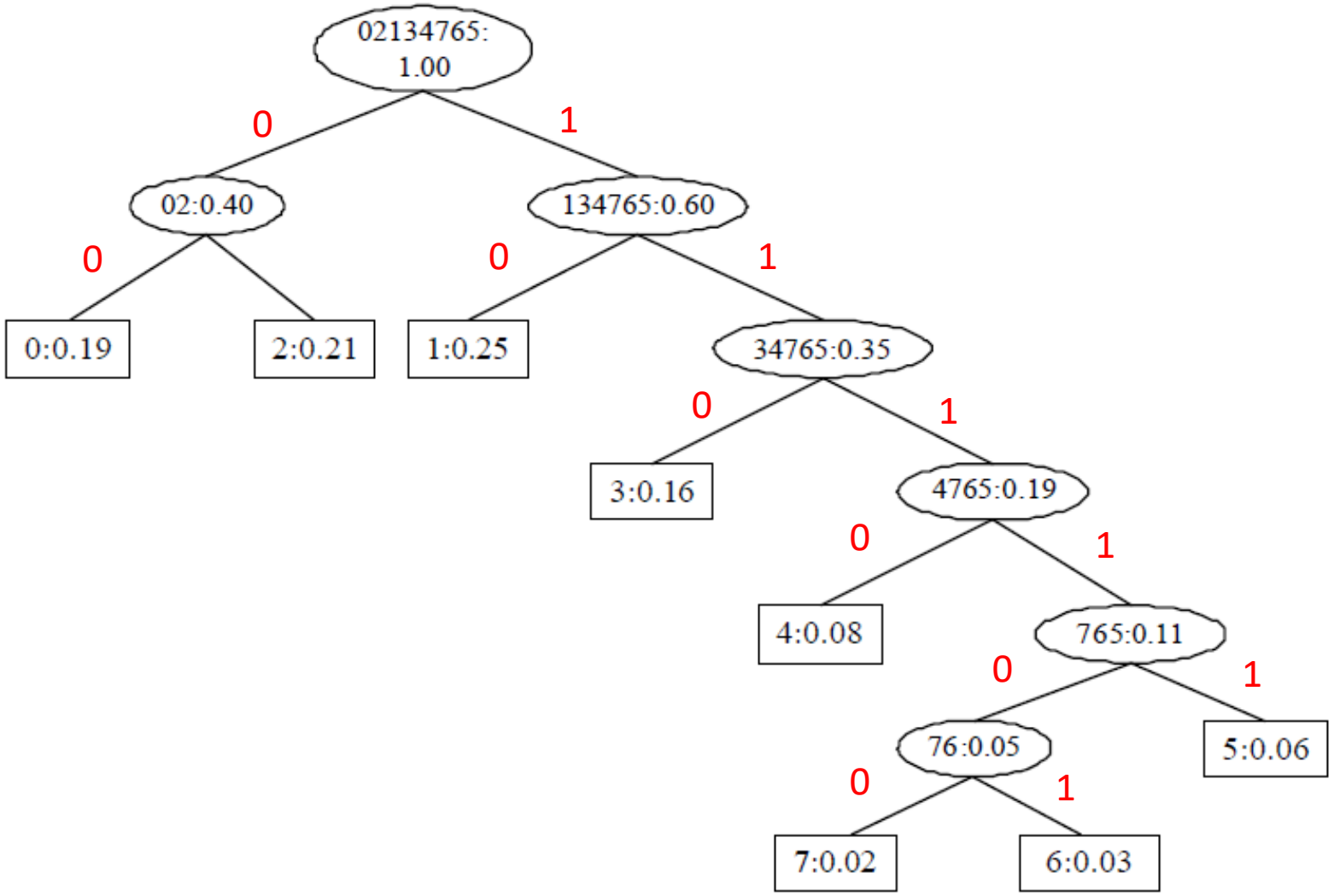
7.


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

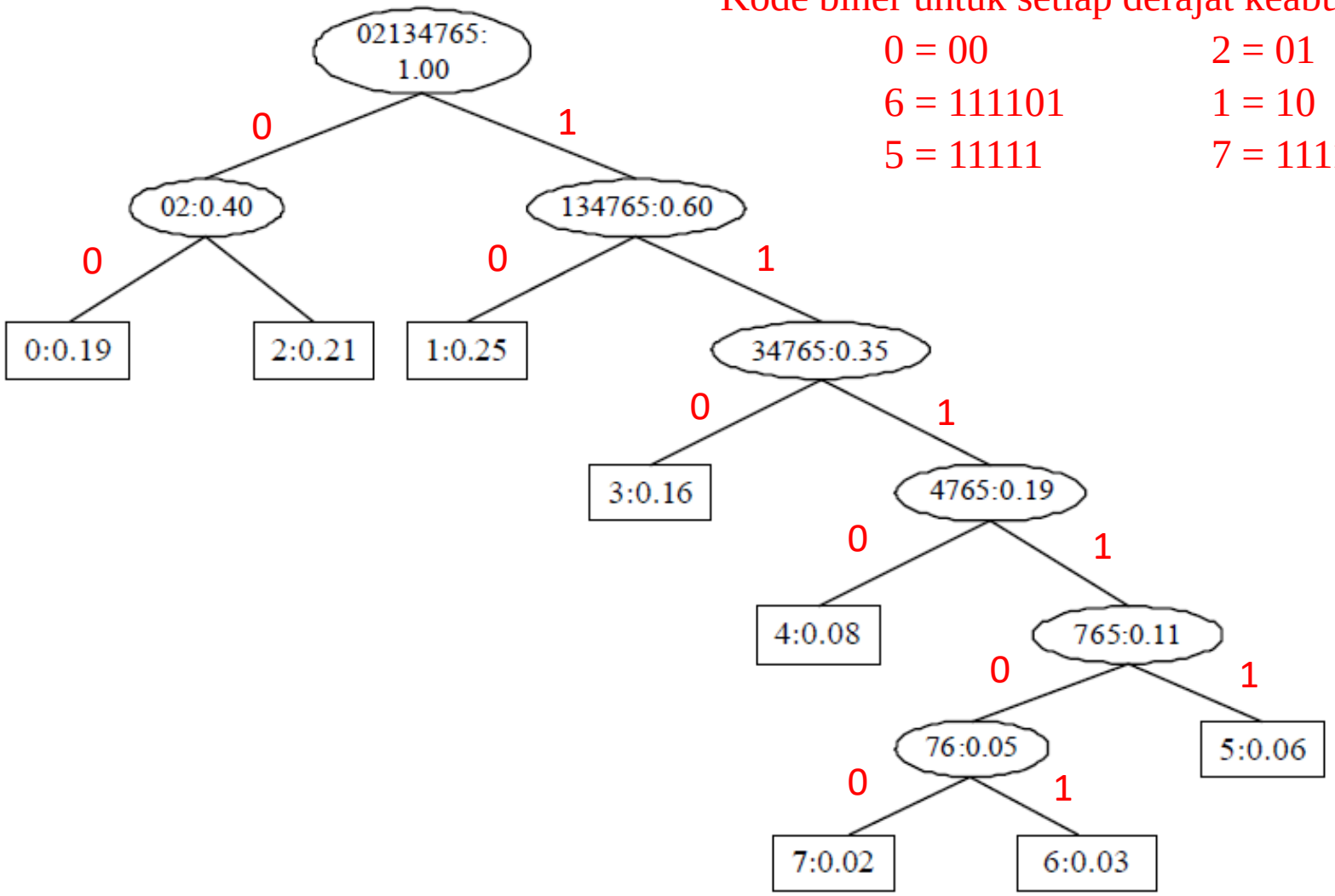
8.


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%

Bersambung ke Bagian 2