TKE 4230
MULTIMEDIA

TEKNIK KOMPRESI LOSSLESS

Herman Tolle, ST., MT.
emang@brawijaya.ac.id

Kompresi

- Memampatkan / mengecilkan *raw* data
- Kompresi Multimedia: memampatan *raw* data multimedia
- Kompresi multimedia adalah mutlak mengingat ukuran *raw* data media yang sangat besar: sinyal suara, image maupun video

Tujuan Kompresi

- Memperkecil ukuran file / data
  -> penyimpanan maupun transmisi

Kompresi Berdasarkan Penerimaan

1. ***Dialoque Mode***: proses penerimaan data secara real time, mis: video conference. Kompresi data harus berada dalam batas penglihatan dan pendengaran manusia.
2. ***Retrieval Mode***: proses penerimaan data tidak real time. Dapat dilakukan *fast forward* dan *fast rewind* di *client*. Dapat dilakukan *random access* terhadap data dan dapat bersifat interaktif

Kompresi Berdasarkan Output

1. **Kompresi Lossless (Non-Lossy)**
   Hasil dekompres dari data terkompresi akan tepat sama persis dengan data sebelum dikompres
2. **Kompresi Lossy**
   Hasil dekompres dari data terkompresi tidak tepat sama persis, tetapi persepsi terhadap semantik data tetap sama

Kriteria Algoritma & Aplikasi Kompresi

- Kualitas data hasil enkoding:
  - ukuran lebih kecil,
  - data tidak rusak (untuk kompresi lossy)
- Ketepatan proses dekompresi data:
  - data hasil dekompresi tetap sama dengan data sebelum dikompres (kompresi loseless)
- Kecepatan, ratio, dan efisiensi proses kompresi & dekompresi

Klasifikasi Teknik Kompresi

- Entropy Encoding
- Source Coding
- Hybrid Coding

Entropy Encoding

- Bersifat *lossless*
- Tekniknya tidak berdasarkan media dengan spesifikasi dan karakteristik tertentu namun berdasarkan urutan data.
- Statistical encoding, tidak memperhatikan semantik data.
- Misalnya: *Run-length coding*, *Huffman coding*, *Arithmetic coding*

Source Coding

- Bersifat *lossy*
- Berkaitan dengan data semantik (arti data) dan media.
- Misalnya: *Prediction* (DPCM, DM), Transformation (FFT, DCT), *Layered Coding* (Bit position, sub-sampling, sub-band coding), *Vector quantization*

Hybrid Coding

- Gabungan antara lossy + lossless
- Misalnya: JPEG, MPEG, H.261, DVI

Basic Information Theory

Informasi dan Entropy

- Set event: $S = \{x_1, ..., x_n\}$
- S disebut alphabet jika x_i sebuah simbol (huruf) digunakan utk membangun pesan (message)
- Probabilitas kemunculan masing-masing event, $p(x_i) = p_i$
- $P = \{p_1, ..., p_n\}$, dimana $p_i \geq 0$, $\sum_{i=1}^{n} p_i = 1$
- Untuk sumber memoryless:
  - Nilai *self-information* yg berhub. dg event x_i digunakan definisi

    $$I(x_i) = -\log_k p_i$$
  - Fungsi di atas adalah ukuran informasi (*surprise* atau *unexpectedness*) dari kemunculan event x_i

- ENTROPI : Menurut Shannon, entropi dari sebuah informasi adalah minimum bit yang dibutuhkan untuk mengkodekan sebuah simbol

$$H(S) = \eta = \sum_i p_i \log_2 \frac{1}{p_i}$$

- Dimana
  - p_i = probabilitas kemunculan simbol S_i.
  - $\log_2 \frac{1}{p_i}$ = jumlah bit yang dibutuhkan untuk kode Si

- For example, in an image with uniform distribution of gray-level intensity, i.e. pi = 1/256, then the number of bits needed to code each gray level is 8 bits. The entropy of this image is 8.

Panjang Kode Rata-rata

- Panjang kode rata-rata untuk semua kode:

$$R_{rata-rata} = \sum_{k=1}^{L} p_k n_k$$

n_k = jumlah bit untuk kode ke-k

- Panjang rata-rata minimum yang dapat dicapai
  Teorema pengkodean sumber:

$$R_{\text{minimum}} \geq H(x) = -\sum_{k=1}^{L} p_k \log_2 p_k \qquad H(x) = \sum_{k=1}^{L} p_k \log_2 \frac{1}{p_k}$$

- H(x) adalah entropi sumber/pengkuantisasi (tanpa memori)

- Contoh:

R_{NBC} = 3; $R_{rata\text{-}rata}$ = 2.204; $R_{huffman}$ = 2.04

Terms

- **Enkoder / Compresor** : software (atau hardware) yang mengkodekan data orisinal menjadi data terkompres
- **Dekoder / Decompresor**: software (atau hardware) yang mendekode data terkompres menjadi data orisinal
- **Codec**:
  software (atau hardware) yang yang mengkodekan dan mendekodekan data
- **Algoritma** : teknik yang digunakan dalam proses pengkodean/kompresi

Code Word

- **Code word** : kombinasi bit yang merupakan representasi dari suatu simbol data orisinal
  - Misalnya: 1 = 001; 2 = 010; 5 = 101; 7 = 111;
- Codeword yang dibuat harus unik, tidak ambigu, relasi 1-1→ (*uniquely decodable*): unik sehingga sumber orisinal dapat dikodekan kembali secara sempurna dari deretan biner code word-nya
- **Natural Binary Code (NBC):** panjang *codeword* sama untuk semua simbol
- **Variable Length Code (VLC):** panjang cw bervariasi

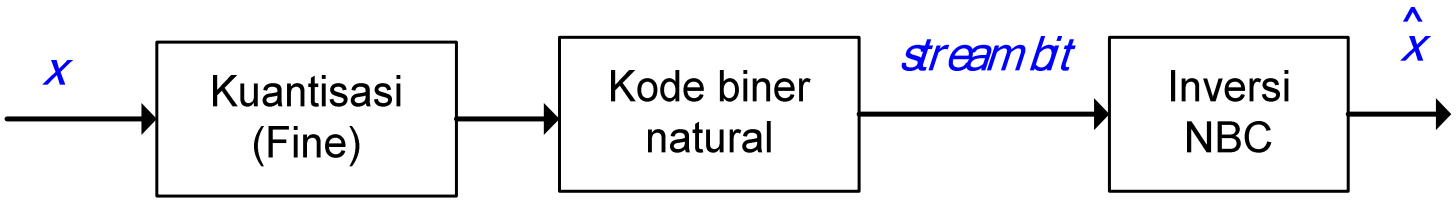
Teknik Pengkodean

- Ada 2 konsep dasar teknik pengkodean kompresi:
  - **Statis**: setiap simbol dikodekan dengan *code word* yang fixed dan selalu sama dalam kompresi. Misalnya: PCM
  - **Dinamik/Adaptif**: simbol dikodekan dengan code word fixed atau variabel dan dapat berubah (adaptif) dalam kompresi data
- Kemungkinan

Ukuran Simbol	Ukuran Code Word	
Fixed	Fixed	S
Fixed	Variabel	D
Variabel	Variabel	D
Variabel	Fixed	D

Pulse Code Modulation (PCM)

Teknik kompresi tradisional:



Tabel pengkodean:

Indeks	0	1	2	3	4	5	6	7
Kode	000	001	010	011	100	101	110	111

Panjang Tetap vs Variable

- **Natural Binary Code** (NBC) adalah pengkodean dengan panjang codeword tetap (fixed). NBC bukan merupakan representasi level kuantisasi yang efisien
- **Variable Length Coding** (VLC) – Pengkodean dengan panjang kode berbeda untuk tiap simbol. Level kuantisasi yang sering terjadi (contoh, nilai sinyal 0) diberi word kode yang pendek

Level Kuantisasi	Indeks k	Probabilitas P_k	NBC	Huffman
y_0	0	0.005	111	1001111
y_1	1	0.02	110	100110
y_2	2	0.14	101	101
y_3	3	0.20	100	11
y_4	4	0.51	000	0
y_5	5	0.08	001	1000
y_6	6	0.04	010	10010
y_7	7	0.005	011	1001110

VLC yang Cocok?

- Teorema pengkodean sumber tidak memberikan informasi tentang bagaimana menyusun kode VLC yang efisien
  - Panjang rata-rata kode mendekati H(x) bit/sample
  - Dapat didecode secara unik
  - Dapat didecode dengan mudah (diinginkan)
  - Sinkronisasi mandiri (fitur tambahan)
  - Dapat didecode pada dua sisi (fitur tambahan)

	y_0	y_4	y_4	y_3	y_6	y_1
Encoder	1001111	0	0	11	10010	100110
Decoder	10011110011110010100110					

Kompresi Lossless

- Data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi.
- Contoh aplikasi: ZIP, RAR, GZIP, 7-Zip
- Digunakan jika dibutuhkan data setelah dikompresi harus dapat diekstrak/dekompres lagi tepat sama.
  Contoh: data teks, data program/biner, beberapa image seperti GIF dan PNG.
- Kadangkala ada data-data yang setelah dikompresi dengan teknik ini ukurannya menjadi lebih besar atau sama

Algoritma Kompresi Loseless

- Algoritma Shannon-Fano
- Huffman Coding
- Adaptive Huffman Coding
- Arithmatic Coding
- Run Length Encoding (RLE)
- Dictionary Based Encoding

Runlength Encoding (RLE)

- Runlength Coding digunakan untuk data yang mengandung cluster yang bernilai sama

  Contoh : Nilai “0” dan “1” yang berurutan

  0000000000111111111111000111111111111000000

  10 12 3 13 6

  - Urutan yang sama mempunyai panjang yang terbatas

- Sangat efisien untuk mengkodekan keluaran kuantisasi “0” *(dead zone)*

Contoh RLE

Run-Length Encoding (RLE) Method

- This is a very simplistic approach that counts sequences of repeating symbols — storing the symbol's value and the number of repeats. Consider the following example:

- Here we have a series of blue x 6, magenta x 7, red x 3, yellow x 3 and green x 4, that is:

- which is clearly a representation using fewer *bits* (only 2/3 the original).

- This would not be a feasible method of compression if the raw data did not contain repeating symbols. In such a circumstance the compressed data would probably be larger. Consider the following:

- This would give:

- which is twice the size!

DICTIONARY-BASED CODING

- Grup simbol, kata atau phrase dinyatakan dg index/singkatan (abbreviation)
- Index/singkatan digunakan dlm transmisi data → penerima melakukan translasi ke bentuk original (menggunakan dictionary yg sama)
- Algoritma Lempel-Ziv-Welch (LZW) menggunakan teknik adaptif dan berbasiskan “kamus”
- Pendahulu LZW adalah LZ77 dan LZ78 yang dikembangkan oleh Jacob Ziv dan Abraham Lempel pada tahun 1977 dan 1978.
- Terry Welch mengembangkan teknik tersebut pada tahun 1984. LZW banyak dipergunakan pada UNIX, GIF, V.42 untuk modem.

Static Dictionary

- sama sepanjang proses encoding
- Akronim: Unibraw, CEO, UN, ASCII
- Abbreviation: dr, Mlg
- Coined abbreviation (M—male, #$—load instruction)
- List pola simbol yg paling sering → dan codeword-nya
  - n-gram → pola n deretan simbol yg sering digunakan (mis. Tri-gram dlm text Inggris: the, que, neu, ome, …)

  → tdk efisien utk semua situasi (language spesific, domain spesific)

Dynamic Directory

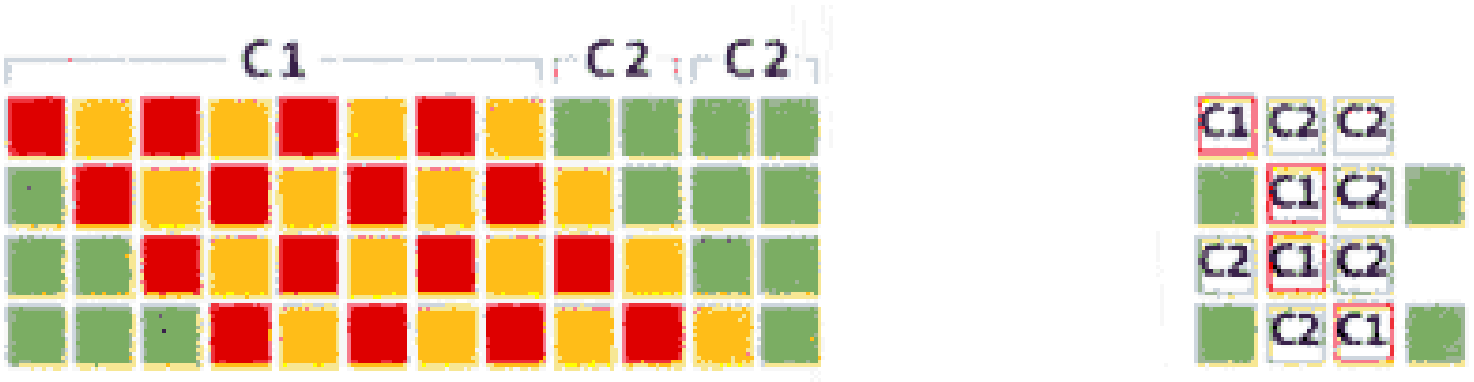
- Dibangun dan dimodifikasi secara dinamis selama proses encoding dan decoding
- Dibangun dari pola umum yang ditemukan di input
- Pola dikodekan dg index, offset atau address pd dictionary
- Kecepatan ditentukan oleh ukuran dictionary dan teknik pencarian
- Kebanyakan teknik didasarkan pd teori dari Jacob Ziv dan Abraham Lempel.

Aplikasi Teknik Dictionary

- Aplikasi utk Kompresi:
  - LZ77:
    - ARJ^{T}, LHarc^{T}, PKZip^{T} UC2^{T}
  - LZ78/LZW:
    - ARC^{T}, PAK^{T}, UNIX^{T}, $ Compress, .gif

Kompresi LZW (Lempel, Ziv dan Welch) pada GIF

- proses *encoding* yang mencari rangkaian *pixel* yang sama pada gambar. Pola yang lebih sering muncul mendapatkan sebuah kode yang mewakili rangkaian tersebut dalam file terkompresi.



Shannon-Fano Coding

Suboptimal code

- Shannon code
- Shannon-Fano code

Optimal code

- Huffman code
- Arithmetic coding

Efisiensi macam-macam code diukur dengan:

$$effisiensi = \frac{H(S)}{L_{avg}}.100\%$$

Shannon-Fano Coding

order the source letters into a sequence s *according to the probability of occurrence;*
ShannonFano (*sequence* s)
    if s *has two letters*
        attach 0 *to the codeword of one letter and* 1 *to the codeword of another;*
    else if s *has more than one letter*
        divide s *into two subsequences* s_1 *and* s_2, *with the minimal difference between probabilities of each subsequence;*
        extend the codeword for each letter in s_1 *by attaching* 0, *and attaching* 1 *to each codeword for letters in* s_2;
        ShannonFano(s_1);
        ShannonFano(s_2);

Shannon-Fano Coding

- Contoh

  $S = \{A, B, C, D, E\}$

  $P = \{0.35, 0.17, 0.17, 0.16, 0.15\}$

- Pengkodean Shannon-Fano:
  - Bagi S kedalam s_1 dan s_2 (pilih yang memberikan perbedaan $p(s_1)$ dan $p(s_2)$ terkecil
  - s_1 = (A,B) → $p(s_1) = p(A) + p(B) = 0{,}52$
  - s_2 = (C,D,E) → $p(s_2) = p(C) + p(D) + p(E) = 0{,}48$
  - Panggil ShannonFano()

Shannon-Fano Coding

x_i	p_i	codeword
A	.35	00
B	.17	01
C	.17	10
D	.16	110
E	.15	111

- Panjang code rata-rata:

  L_{sh} = 0,35*2 + 0,17*2 + 0,17*2 + 0,16*3+0,15*3 = 2,31
- Efisiensi = (2,23284/2,31)*100 = 96,66 %

Shannon-Fano Algorithm

- Dikembangkan oleh Shannon (Bell Labs) dan Robert Fano (MIT).

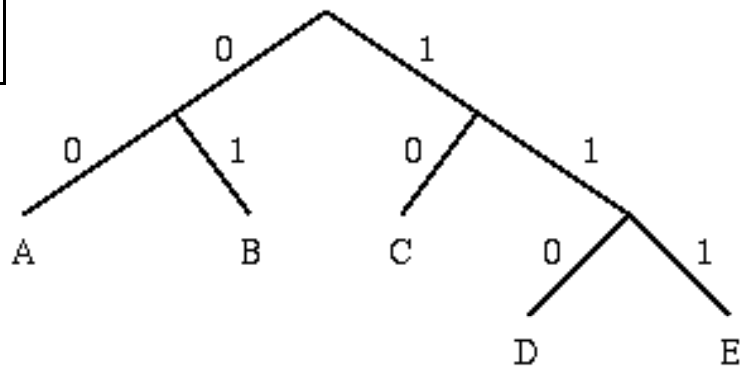
Algoritma :

1. Urutkan simbol berdasarkan frekuensi kemunculannya
2. Bagi simbol menjadi 2 bagian secara rekursif, dengan jumlah yang kira-kira sama pada kedua bagian, sampai tiap bagian hanya terdiri dari 1 simbol.

- Cara yang paling tepat untuk mengimplementasikan adalah dengan membuat binary tree.

Contoh Shannon-Fano

Symbol	A	B	C	D	E
Count	15	7	6	6	5



Symbol	Count	$\log_2(1/p_i)$	Code	Subtotal (# of bits)
A	15	1.38	00	30
B	7	2.48	01	14
C	6	2.70	10	12
D	6	2.70	110	18
E	5	2.96	111	15

Huffman Coding

- Optimal code pertama dikembangkan oleh David Huffman

- Utk sumber $S = \{x_1, \ldots, x_n\}$; Probabilitas $P = \{p_1, \ldots\ldots, p_n\}$; Codewords $\{c_1, \ldots\ldots, c_n\}$; dan Panjang $\{l_1, \ldots\ldots, l_n\}$. Terdapat optimal binary prefix code dengan karakteristik:

  Teorema:

  (a) Jika $p_j > p_i$, *maka* $l_j \leq l_i$

  (b) Dua codeword dari dua simbol dg probabilitas terendah mempunyai panjang yg sama

  (c) Dua codeword terpanjang identik kecuali pada digit terakhir

Pengkodean Huffman

- Pengkodean Huffman bertujuan untuk mengkodekan setiap simbol dengan panjang kode berbeda, simbol yg paling sering muncul akan memiliki kode lebih pendek

- Algoritma Enkoding Huffman
  1. Simbol diurutkan berdasarkan probabliti kemunculan. Dua simbol terbawah diberi assign 0 dan 1. -> Splitting stage
  2. Dua simbol terbawah tadi dijumlahkan dan menjadi kode sumber baru dan probabilitasnya dijumlahkan. Diurutkan menjadi stage 2
  3. Proses tersebut diurutkan sehingga urutannya hanya tinggal 2 baris dengan assign 0 dan 1.
  4. Kode word untuk simbol tersebut adalah kombinasi biner yg terjadi, dilihat dari belakang

Huffman Coding

HuffmanAlgorithm()

for *each letter create a tree with a single root node and order all trees according to the probability of letter occurrence;*

while *more than one tree is left*

take the two trees t_1, t_2 *with the lowest probabilities* p_1, p_2 and *create a tree with probability in its root equal* to $p_1 + p_2$ *and with* t_1 *and* t_2 *as its subtrees;*

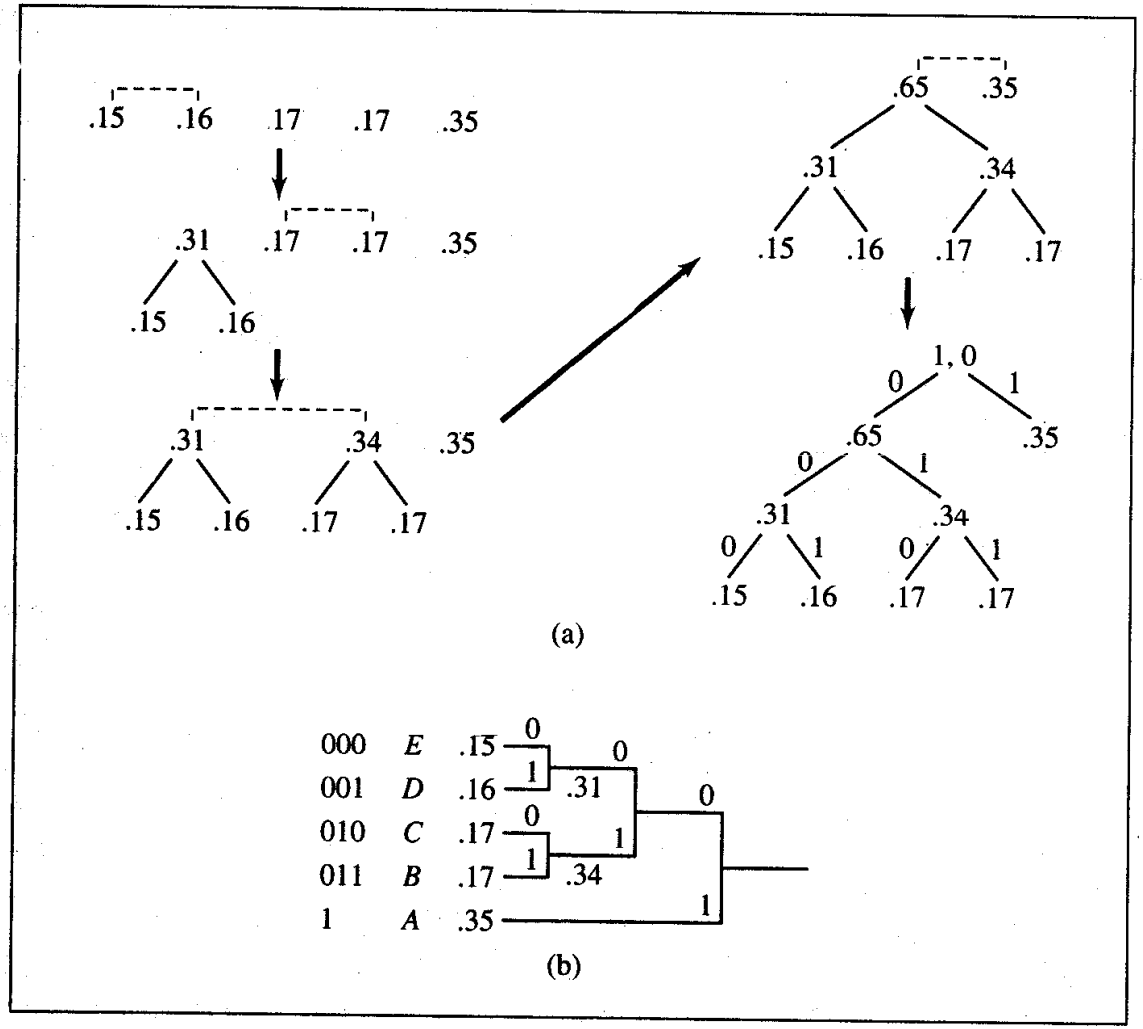
associate 0 *with each left branch and* 1 *with each right branch;*

create a unique codeword for each letter by traversing the tree from the root to the leaf containing the probability corresponding to this letter and putting all encountered 0*s and* 1*s together;*

Huffman Coding

Contoh:

x_i	p_i
A	0,35
B	0,17
C	0,17
D	0,16
E	0,15



Huffman Coding

- Dari Huffman tree dapat dibuat tabel codeword:

A	1
B	011
C	010
D	001
E	000

L_{Huff} = 0,35*1 + 0,17*3 + 0,17*3 + 0,16*3 + 0,15*3 = 2,3

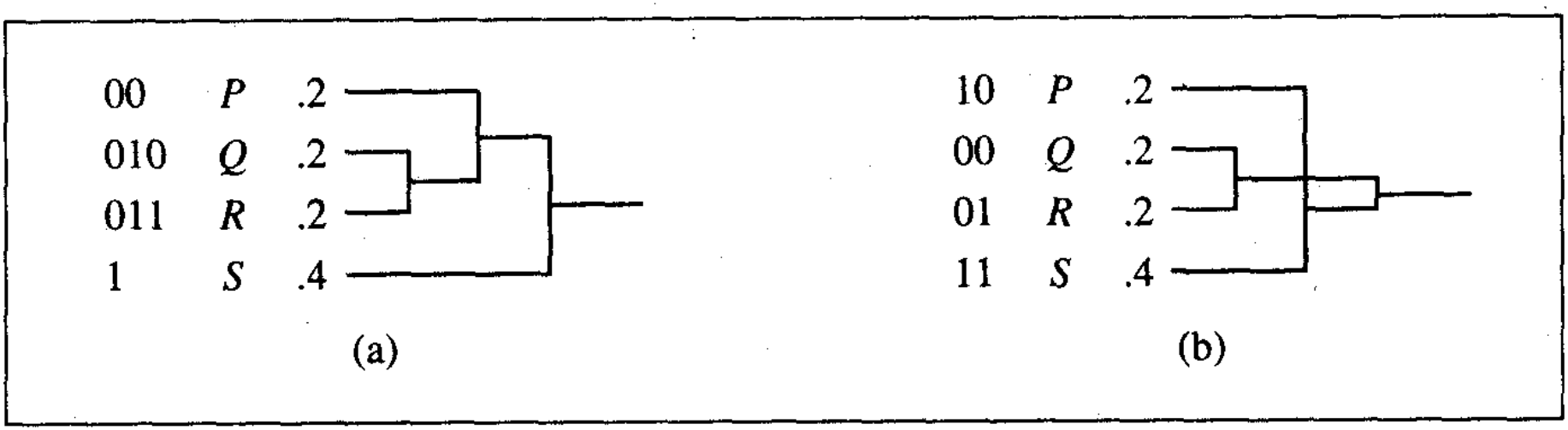
H(S) = 2,23284

Efisiensi = (2,23284/2,3) x 100 % = 97,08%

Huffman Coding

- Tergantung pada bagaimana memilih probabilitas terendah saat membangun Huffman tree

  → Huffman tree tidak unik
- Namun, panjang rata-rata codeword selalu sama utk tree yang berbeda



Konstruksi Kode Huffman

- Aturan penyusunan kode Huffman:

y_0	0.51					0.51(**0**)	0
y_1	0.20				0.20(**1**)	0.49(**1**)	11
y_2	0.14			0.14(**1**)	0.29(**0**)		101
y_3	0.08		0.08(**0**)	0.15(**0**)			1000
y_4	0.04	0.04(**0**)	0.07(**1**)				10010
y_5	0.02	0.02(**0**)	0.03(**1**)				100110
y_6	0.005(**0**)	0.01(**1**)					1001110
y_7	0.005(**1**)						1001111

Keterbatasan Pengkodean Huffman

- Rate selalu lebih besar dari 1.0 bit/sample
- Predesain kode
- Tabel kode tetap
- Jika probabilitas berbeda dengan yang digunakan dalam desain, ekspansi data dapat terjadi
- Versi praktek:

  - Implementasi *two-pass*
  - Blok adaptif (tabel kode per blok data)
  - Huffman rekursif (perubahan tabel kode secara kontinyu)

Latihan

- Rancang kode word untuk 5 simbol (S0, S1, S2, S3, S4) dengan probabiliti sbb: S0=0,4; S1=0,25; S2=0,2; S3=0,1; S4=0,05.
- Hitung Panjang kode rata-rata dan Entropi dari pengkodean Huffman untuk simbol-simbol tersebut

Jawaban:

Simbol	P	Code Word
S0	0.4	00
S1	0.2	10
S2	0.2	11
S3	0.1	010
S4	0.1	011

- L = 0,4(2) + 0,2(2) + 0,2(2) + 0,1(3) + 0,1(3) = 2,2
- H(x) = 0,52877 + 0,46439 + 0,464439 + 0,33219 + 0,33219 = 2,12193

Sinkronisasi dalam VLC

- Dekoder perlu menerjemahkan bit-bit sebelum dapat didekodekan
- Kesalahan bit dapat menyebabkan hilangnya sinkronisasi antara encoder dan decoder VLC
- Contoh:
  - Level kuantisasi : 4 4 3 0 5 3 4
  - Bit NBC : 000 000 100 111 001 100 000
  - Bit Huffman : 0 0 11 1001111 1000 11 0
  - Diasumsikan bit ke-4 salah
  - NBC decoded : 4 3 3 0 5 3 4
  - Huffman decoded : 4 4 2 4 4 3 3 5 3 4

Hilang sinkronisasi

Aplikasi Kompresi Loseless

- Format File:
  - ZIP
  - RAR
  - GIF

ZIP File Format

- Ditemukan oleh Phil Katz untuk program PKZIP kemudian dikembangkan untuk WinZip, WinRAR, 7-Zip.
- Berekstensi *.zip dan MIME application/zip
- Dapat menggabungkan dan mengkompresi beberapa file sekaligus menggunakan bermacam-macam algoritma, namun paling umum menggunakan Katz's Deflate Algorithm.

Metode ZIP

Beberapa method Zip:

- *Shrinking* : merupakan metode variasi dari LZW
- *Reducing* : merupakan metode yang mengkombinasikan metode *same byte sequence based* dan *probability based encoding*.
- *Imploding* : menggunakan metode *byte sequence based* dan *Shannon-Fano encoding*.
- *Deflate* : menggunakan LZW
- Aplikasi: WinZip oleh Nico-Mak Computing

RAR File

- Ditemukan oleh Eugene Roshal, sehingga RAR merupakan singkatan dari Roshal Archive pada 10 Maret 1972 di Rusia.
- Berekstensi .rar dan MIME application/x-rar-compressed.
- Proses kompresi lebih lambat dari ZIP tapi ukuran file hasil kompresi lebih kecil.
- Aplikasi: WinRAR yang mampu menangani RAR dan ZIP, mendukung volume split, enkripsi AES.