TEKNIK KOMPRESI LOSSLESS TEXT

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

Algoritma Kompresi Loseless

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

RLE – Run Length Encoding

- Kompresi data teks dilakukan jika ada beberapa huruf yang sama yang ditampilkan berturut-turut:

  contoh

  data : ABCCCCCCCCCDEFGGGG = 17 karakter

  RLE tipe 1 (min 4 huruf sama)

  ABC!8DEFG!4 = 11 karakter

RLE #2

- RLE ada yang menggunakan suatu karakter yang tdk digunakan dalam teks, seperti “!” untuk menandai.

RLE #3

- Kelemahan, jika ada karakter angka, mana tanda mulai dan akhirnya?

  data: ABCCCCCCCCCDEFGGGG = 17 karakter

  RLE tipe 2 : -2AB8C-3DEF4G = 12 karakter

  data: AB12CCCCDEEEF = 13 karakter

  RLE tipe 2 : -4AB124CD3EF = 12 karakter

RLE #4

- RLE ada yang menggunakan flag bilangan negatif untuk menandai batas sebanyak jumlah karakter tersebut

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

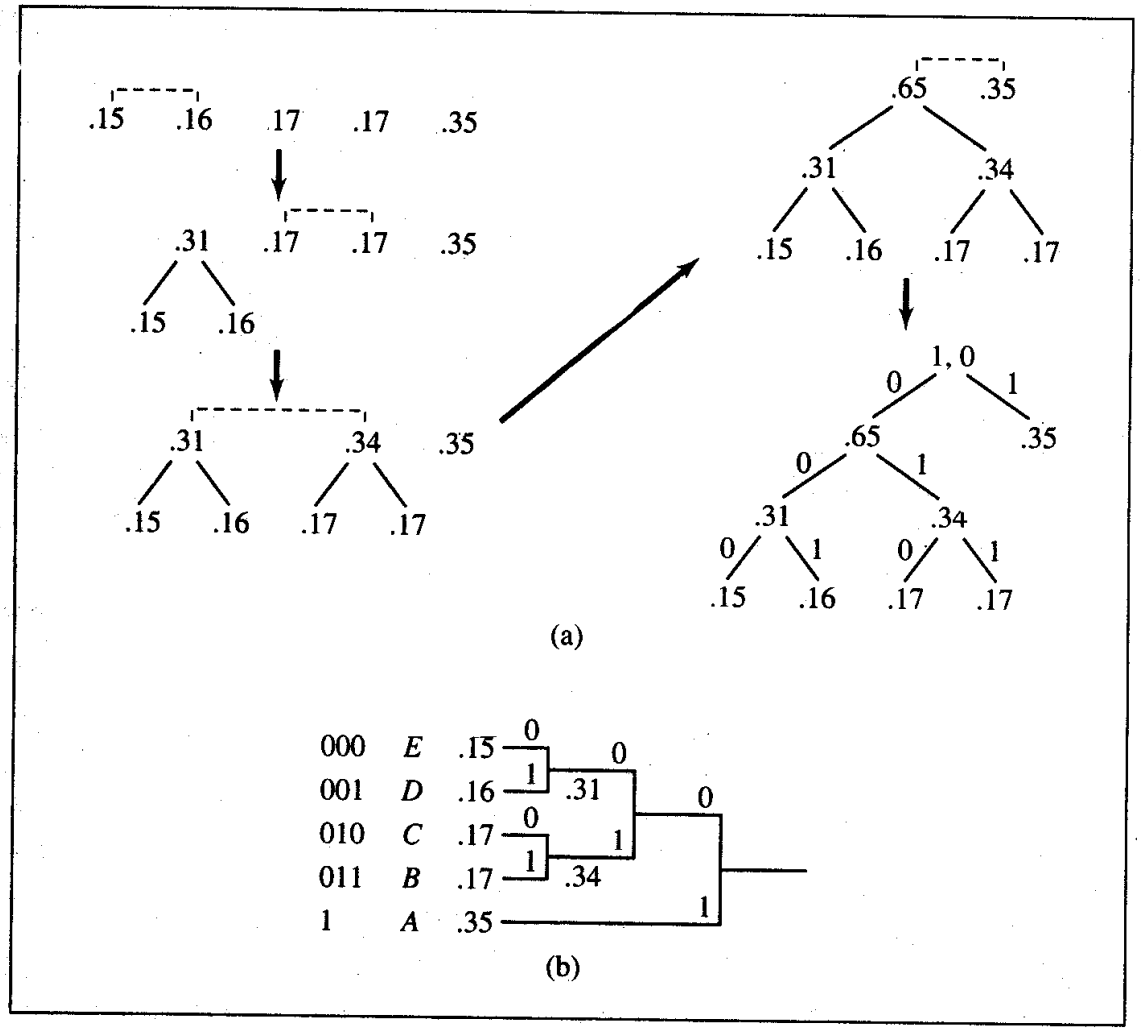
- Pengkodean Huffman bertujuan untuk mengkodekan setiap simbol dengan panjang kode berbeda, simbol yg paling sering muncul akan memiliki kode lebih pendek

- 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

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

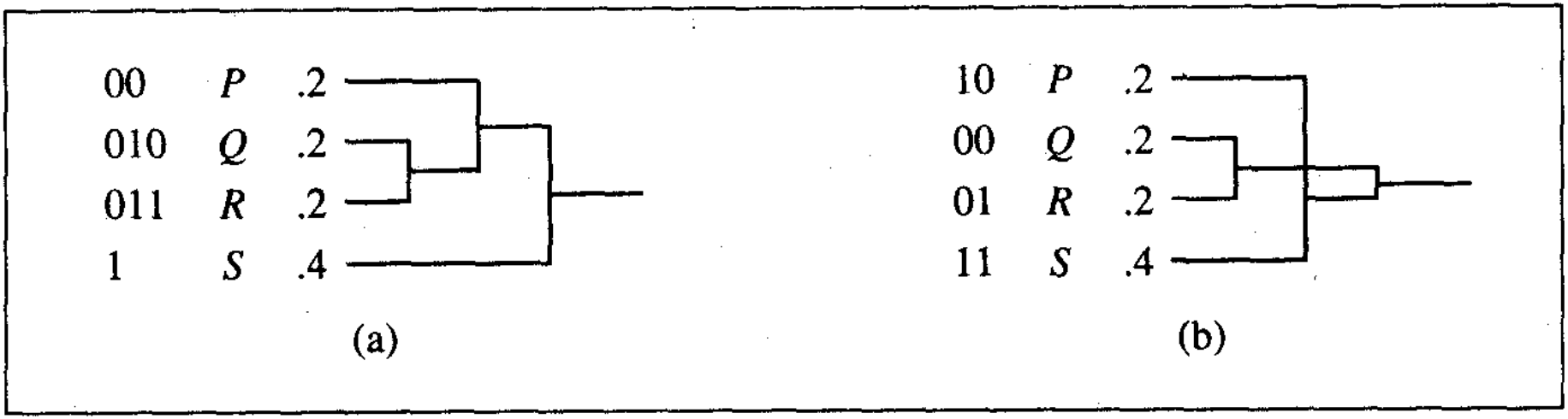
H(S) = 2,23284

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

Huffman Coding

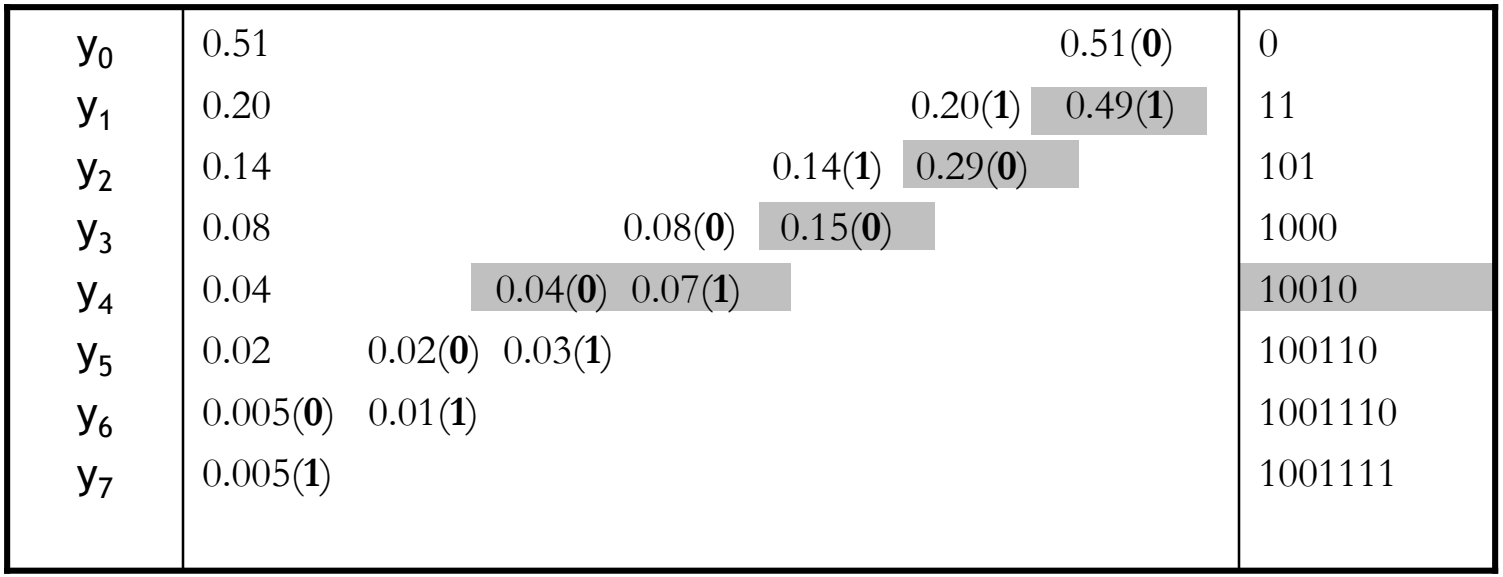
- Tergantung pada bagaimana memilih probabilitas terendah saat membangun Huffman tree

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



Konstruksi Kode Huffman

- Aturan penyusunan kode Huffman:



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)

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

- 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).

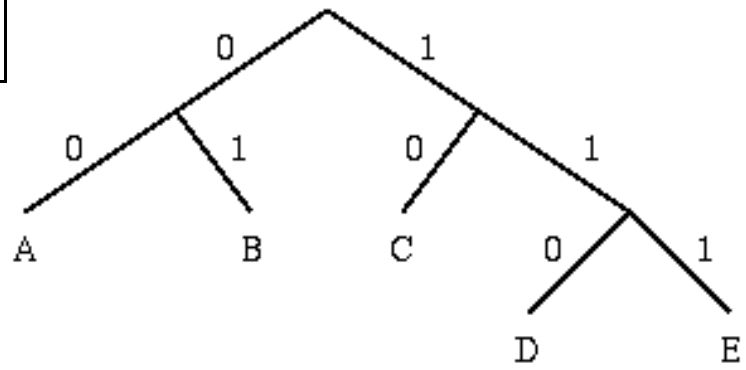
Algoritma :

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

Aplikasi Kompresi Loseless

- ZIP File Format
- RAR File

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.