Diktat Kuliah

Pemrograman Multimedia

Yulisdin Mukhlis, ST., MT

KOMPRESI DAN TEKS

KOMPRESI DATA

- Kompresi berarti memampatkan/mengecilkan ukuran
- Kompresi data adalah proses mengkodekan informasi menggunakan bit atau information-bearing unit yang lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem enkoding tertentu.
- Contoh kompresi sederhana yang biasa kita lakukan misalnya adalah menyingkat kata-kata yang sering digunakan tapi sudah memiliki konvensi umum. Misalnya: kata “yang” dikompres menjadi kata “yg”.
- Pengiriman data hasil kompresi dapat dilakukan jika pihak pengirim/yang melakukan kompresi dan pihak penerima memiliki aturan yang sama dalam hal kompresi data.
- Pihak pengirim harus menggunakan algoritma kompresi data yang sudah baku dan pihak penerima juga menggunakan teknik dekompresi data yang sama dengan pengirim sehingga data yang diterima dapat dibaca/di-dekode kembali dengan benar.
- Kompresi data menjadi sangat penting karena memperkecil kebutuhan penyimpanan data, mempercepat pengiriman data, memperkecil kebutuhan bandwidth.

- Teknik kompresi bisa dilakukan terhadap data teks/biner, gambar (JPEG, PNG, TIFF), audio (MP3, AAC, RMA, WMA), dan video (MPEG, H261, H263).

Contoh kebutuhan data selama 1 detik pada layar resolusi 640 x 480: -

Data Teks

- 1 karakter = 2 bytes (termasuk karakter ASCII Extended)
- Setiap karakter ditampilkan dalam 8x8 pixels
- Jumlah karakter yang dapat ditampilkan per halaman =

$$\frac{640 \times 480}{8 \times 8} = 4800 \text{ karakter}$$

Kebutuhan tempat penyimpanan per halaman = 4.800×2 byte = 9.600 byte = 9.375 Kbyte

- Data Grafik Vektor
  - 1 still image membutuhkan 500 baris
  - Setiap 1 baris direpresentasikan dalam posisi horisontal, vertikal, dan field atribut sebesar 8-bit
  - sumbu Horizontal direpresentasikan dengan $\log_2 640 = 10$ bits
  - sumbu Vertical direpresentasikan dengan $\log_2 480 = 9$ bits
  - Bits per line = 9bits + 10bits + 8bits = 27bits
  - Storage required per screen page = $500 \times \frac{27}{8} = 1687{,}5$ byte = 1,65 Kbyte

- Color Display
  - Jenis : 256, 4.096, 16.384, 65.536, 16.777.216 warna
  - Masing-masing warna pixel memakan tempat 1 byte
  - Misal 640 x 480 x 256 warna x 1 byte = 307.200 byte = 300 KByte

Kebutuhan tempat penyimpanan untuk media kontinyu untuk 1 detik playback:

- Sinyal audio tidak terkompres dengan kualitas suara telepon dengan sample 8 kHz dan dikuantisasi 8 bit per sample, pada bandwidth 64 Kbits/s, membutuhkan storage:

$$\text{Required storage space/s} = \frac{64\ \text{Kbit/s}}{8\ \text{bit/byte}} \times \frac{1\,\text{s}}{1{,}024\ \text{byte/Kbyte}} = 8\ \text{Kbyte}$$

- Sinyal audio CD disample 44,1 kHz, dikuantisasi 16 bits per sample, Storage = 44,1 kHz x 16 bits = 705,6 x 10^3 bits = 88.200 bytes untuk menyimpan 1 detik playback
- Kebutuhan sistem PAL standar
    - 625 baris dan 25 frame/detik
    - 3 bytes/pixel (luminance, red chrom, blue chrom)
    - Luminance Y menggunakan sample rate 13,5 MHz
    - Chrominance (R-Y dan B-Y) menggunakan sample rate 6.75 MHz
    - Jika menggunakan 8 bit/sample, maka

$$\text{Bandwidth} = (13.5\ \text{MHz} + 6.75\ \text{MHz} + 6.75\ \text{MHz}) \times 8\,\text{bit} = 216 \times 10^6\ \text{bit/s}.$$

$$\text{Data rate} = 640 \times 480 \times 25 \times 3\ \text{byte/s} = 23{,}040{,}000\ \text{byte/s}$$

$$\text{Required storage sapce/s} = 2{,}304 \times 10^4\ \text{byte/s} \times \frac{1\,\text{s}}{1{,}024\ \text{byte/Kbyte}} = 22{,}500\text{Kbyte}$$

Jenis Kompresi Data Berdasarkan Mode Penerimaan Data oleh Manusia

- **Dialoque Mode**: yaitu proses penerimaan data dimana pengirim dan penerima seakan berdialog (real time), seperti pada contoh *video conference*.
    - Dimana kompresi data harus berada dalam batas penglihatan dan pendengaran manusia. Waktu tunda (delay) tidak boleh lebih dari 150 ms, dimana 50 ms untuk proses kompresi dan dekompresi, 100 ms mentransmisikan data dalam jaringan.
- **Retrieval Mode**: yaitu proses penerimaan data tidak dilakukan secara real time

- Dapat dilakukan *fast forward* dan *fast rewind* di client
- Dapat dilakukan random access terhadap data dan dapat bersifat interaktif

Jenis Kompresi Data Berdasarkan Output

- **Lossy Compression**
  - Teknik kompresi dimana data hasil dekompresi tidak sama dengan data sebelum kompresi namun sudah "cukup" untuk digunakan. Contoh: Mp3, streaming media, JPEG, MPEG, dan WMA.
  - Kelebihan: ukuran file lebih kecil dibanding loseless namun masih tetap memenuhi syarat untuk digunakan.
  - Biasanya teknik ini membuang bagian-bagian data yang sebenarnya tidak begitu berguna, tidak begitu dirasakan, tidak begitu dilihat oleh manusia sehingga manusia masih beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi.
  - Misal terdapat image asli berukuran 12,249 bytes, kemudian dilakukan kompresi dengan JPEG kualitas 30 dan berukuran 1,869 bytes berarti image tersebut 85% lebih kecil dan ratio kompresi 15%.
- **Loseless**
  - Teknik kompresi dimana data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi. Contoh aplikasi: ZIP, RAR, GZIP, 7-Zip
  - Teknik ini digunakan jika dibutuhkan data setelah dikompresi harus dapat diekstrak/dekompres lagi tepat sama. Contoh pada 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.

Kriteria Algoritma dan Aplikasi Kompresi Data

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

Klasifikasi Teknik Kompresi

Entropy Encoding

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

Source Coding

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

Hybrid Coding

- Gabungan antara lossy + loseless -

mis: JPEG, MPEG, H.261, DVI

Contoh-contoh Teknik Kompresi Teks

Run-Length-Encoding (RLE)

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

  Mis: Data: ABCCCCCCCCDEFGGGG = 17 karakter
  RLE tipe 1 (min. 4 huruf sama) : ABC!8DEFG!4 = 11 karakter

- RLE ada yang menggunakan suatu karakter yang tidak digunakan dalam teks tersebut seperti misalnya '!' untuk menandai.
- Kelemahan? Jika ada karakter angka, mana tanda mulai dan akhir?

  Misal data : ABCCCCCCCCDEFGGGG = 17 karakter
  RLE tipe 2: -2AB8C-3DEF4G = 12 karakter

  Misal data : AB12CCCCDEEEF = 13 karakter
  RLE tipe 2: -4AB124CD3EF = 12 karakter

- RLE ada yang menggunakan flag bilangan negatif untuk menandai batas sebanyak jumlah karakter tersebut.
- Berguna untuk data yang banyak memiliki kesamaan, misal teks ataupun grafik seperti icon atau gambar garis-garis yang banyak memiliki kesamaan pola.
- Best case: untuk RLE tipe 2 adalah ketika terdapat 127 karakter yang sama sehingga akan dikompres menjadi 2 byte saja.
- Worst case: untuk RLE tipe 2 adalah ketika terdapat 127 karakter yang berbeda semua, maka akan terdapat 1 byte tambahan sebagai tanda jumlah karakter yang tidak sama tersebut.
- Menggunakan teknik loseless
- Contoh untuk data image:

		
Original size: 10000 bytes Compressed size: 5713 bytes Ratio: 1.75	Original size: 10000 bytes Compressed size: 10100 Ratio: 0.99	Original size: 10000 bytes Compressed size: 200 Ratio: 50

Static Huffman Coding

- Frekuensi karakter dari string yang akan dikompres dianalisa terlebih dahulu. Selanjutnya dibuat pohon huffman yang merupakan pohon biner dengan **root awal** yang diberi nilai 0 (sebelah kiri) atau 1 (sebelah kanan), sedangkan selanjutnya untuk dahan **kiri** selalu diberi nilai 1(kiri) - 0 (kanan) dan di dahan **kanan** diberi nilai 0(kiri) - 1(kanan)
- A bottom-up approach = frekuensi terkecil dikerjakan terlebih dahulu dan diletakkan ke dalam leaf(daun).
- Kemudian leaf-leaf akan dikombinasikan dan dijumlahkan probabilitasnya menjadi root diatasnya.

Mis: MAMA SAYA

A = 4 -> 4/8 = 0.5

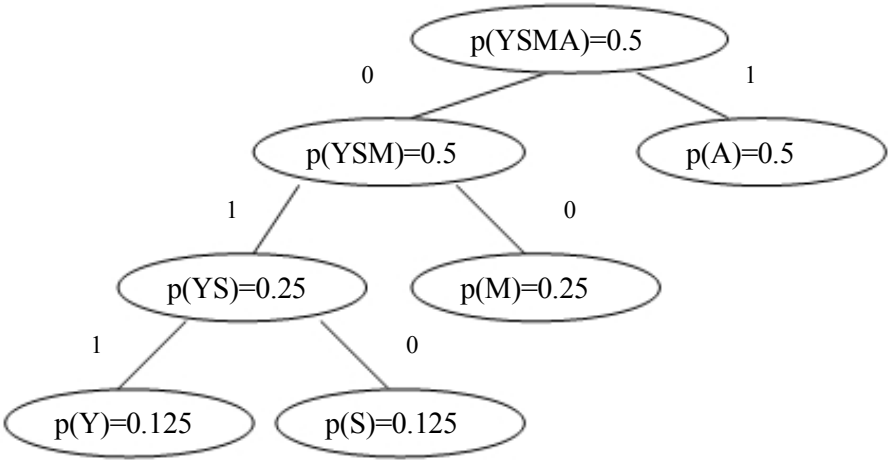
M = 2 -> 2/8 = 0.25

S = 1 -> 1/8 = 0.125

Y = 1 -> 1/8 = 0.125

Total = 8 karakter

Huffman Tree:



Sehingga w(A) = 1, w(M) = 00, w(S) = 010, dan w(Y) = 011

Contoh lain:

Jika terdapat p(A) = 0.16, p(B) = 0.51, p(C) = 0.09, p(D) = 0.13, dan p(E) = 0.11, buatlah Huffman Tree-nya dan weight masing-masing karakter!

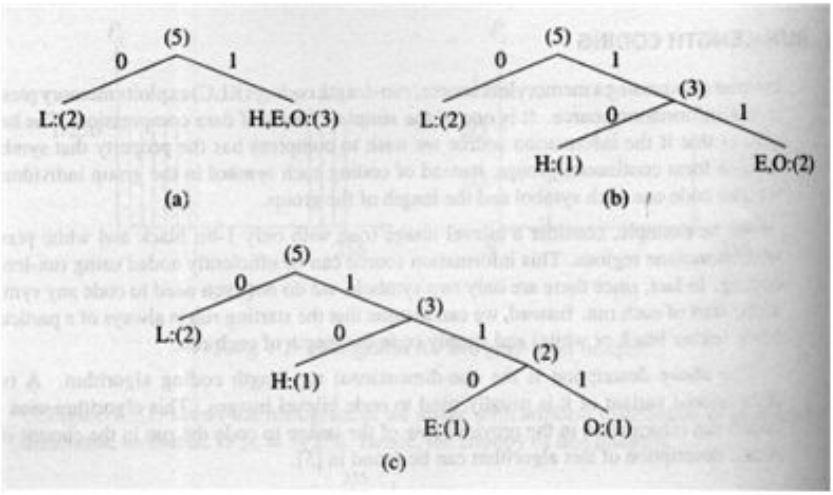
Shannon-Fano Algorithm

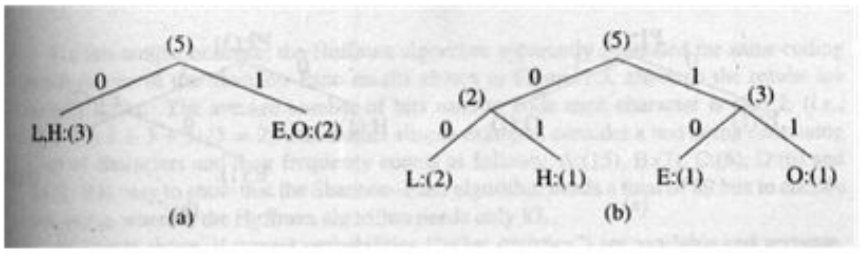
- Dikembangkan oleh Shannon (Bell Labs) dan Robert Fano (MIT)
- Contoh :

  H E L L O

Simbol	H	E	L	O
Jumlah	1	1	2	1

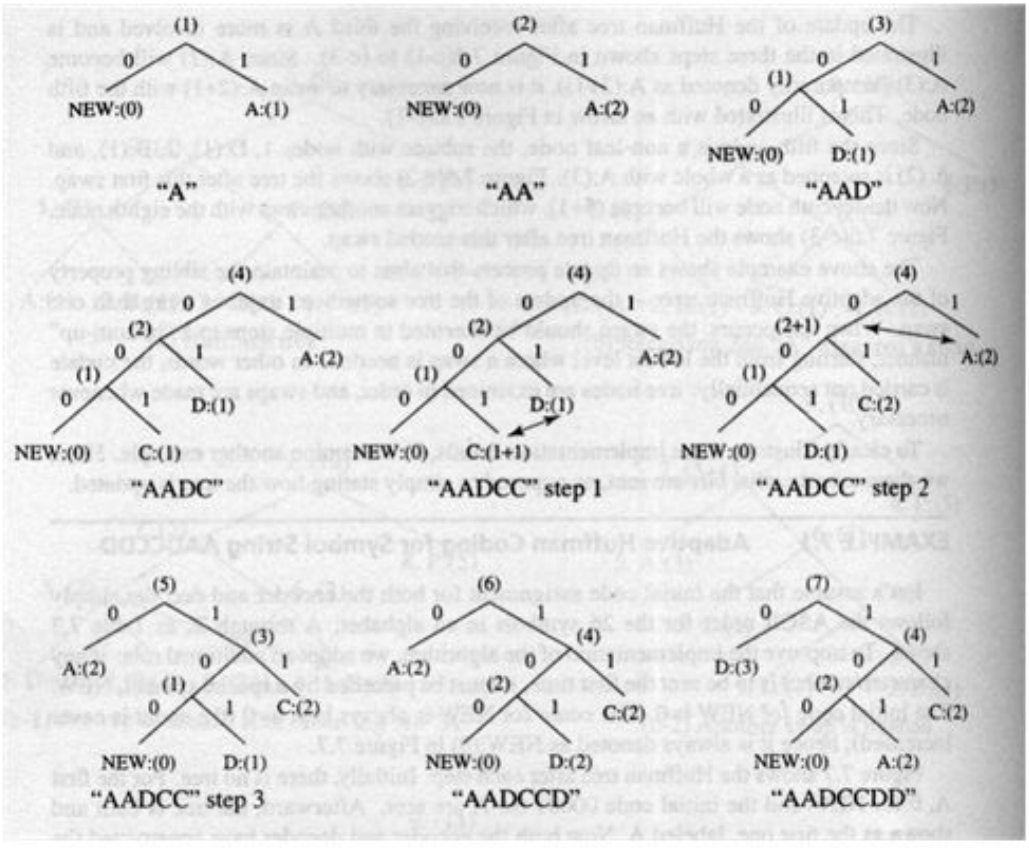
- Algoritma :
  1. Urutkan simbol berdasarkan frekuensi kemunculannya
  2. Bagi simbol menjadi 2 bagian secara rekursif, dengan jumlah yang kira-kira sama pada kedua bagian, sampai tiap bagian hanya terdiri dari 1 simbol.
- Cara yang paling tepat untuk mengimplementasikan adalah dengan membuat binary tree.





Adaptive Huffman Coding

- Metode SHC mengharuskan kita mengetahui terlebih dahulu frekuensi masing-masing karakter sebelum dilakukan proses pengkodean. Metode AHC merupakan pengembangan dari SHC dimana proses penghitungan frekuensi karakter dan pembuatan pohon Huffman dibuat secara dinamis pada saat membaca data.
- Algoritma Huffman tepat bila dipergunakan pada informasi yang bersifat statis. Sedangkan untuk multimedia application, dimana data yang akan datang belum dapat dipastikan kedatangannya (audio dan video streaming), algoritma Adaptive Huffman dapat dipergunakan.
- Metode SHC maupun AHC merupakan kompresi yang bersifat loseless.
- Dibuat oleh David A. Huffman dari MIT tahun 1952
- Huffman banyak dijadikan "back-end" pada algoritma lain, seperti Arithmetic Coding, aplikasi PKZIP, JPEG, dan MP3.



DICTIONARY-BASED CODING

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.

Algoritma Kompresi:

```
BEGIN
     S = next input character;
     While not EOF
          {
          C = next input character;
          If s + c exists in the diactionary
               S = s + c
          Else
               {
               Output the code for s;
               Add string s + c to the dictionary with a new code S
               = c;
               }
          }
END
```

Let's start with a very simple dictionary (also referred to as a *string table*), initially containing only three characters, with codes as follows:

code	string
1	A
2	B
3	C

Now if the input string is ABABBABCABABBA, the LZW compression algorithm works as follows:

s	c	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

Algoritma Dekompresi:

```
BEGIN
      S = NULL;
      while not EOF{
            K = NEXT INPUT CODE;
            Entry = dictionary entry for K;
            Ouput entry;
            if(s != NULL)
                  add string s + entry[0] to dictionary with new code S
            = Entry;
      }
END
```

Contoh Dekompresi

Input : 1 2 4 5 2 3 4 6 1

S	K	Entry/output	Code	String
			1	A
			2	B
			3	C
NULL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	4	AB	9	CA
AB	6	ABB	10	ABA
ABB	1	A	11	ABBA
A	EOF			

Hasil Dekode: **ABABBABCABABBA**

Aplikasi Kompresi

- 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.
    - Beberapa method Zip:
        ③ Shrinking : merupakan metode variasi dari LZW

- ③ Reducing : merupakan metode yang mengkombinasikan metode same byte sequence based dan probability based encoding.
- ③ Imploding : menggunakan metode byte sequence based dan Shannon-Fano encoding.
- ③ Deflate : menggunakan LZW
- ③ Bzip2, dan lain-lain

o Aplikasi: WinZip oleh Nico-Mak Computing -

RAR File

o Ditemukan oleh Eugene Roshal, sehingga RAR merupakan singkatan dari **R**oshal **Ar**chive pada 10 Maret 1972 di Rusia.

o Berekstensi .rar dan MIME application/x-rar-compressed

o Proses kompresi lebih lambat dari ZIP tapi ukuran file hasil kompresi lebih kecil.

o Aplikasi: WinRAR yang mampu menangani RAR dan ZIP, mendukung volume split, enkripsi AES.