Kompresi Data 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)**

Kebutuhan data (1 detik / 640x480)

Data Text

- 1 karakter = 2 bytes (termasuk karakter ASCII Extended)
- Setiap karakter ditampilkan dalam 8 x 8 pixels
- Jumlah karakter yang dapat ditampilkan per halaman
  $= \frac{640 \times 480}{8 \times 8} = 4800$ karakter
- Kebutuhan tempat penyimpanan per halaman = 4.800 × 2 byte = 9.600 byte = **9,375** Kbyte

Kebutuhan data (1 detik / 640x480)

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 log2 640 = 10
  bits
- Sumbu Vertical direpresentasikan dengan log2 480 = 9 bits
- Bits per line = 9bits + 10bits + 8bits = 27bits
- *Storage required* per screen page = 500 × 27 = 1687,5 byte =
  1,65 Kbyte

Kebutuhan data (1 detik / 640x480)

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

Jenis Kompresi Data

- **Berdasar mode penerimaan data yang diterima manusia**
- **Berdasarkan Output**

Kompresi Berdasar mode penerimaan data yang diterima manusia

- **Dialoque Mode:**
  - proses penerimaan data dimana pengirim dan penerima seakan berdialog (*real time*), seperti pada contoh *video conference*.
  - kompresi data harus berada dalam batas penglihatan dan pendengaran manusia.
- **Retrieval Model**
  - proses penerimaan data tidak dilakukan secara real time.

Kompresi 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.

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.
- Misal:
  - 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

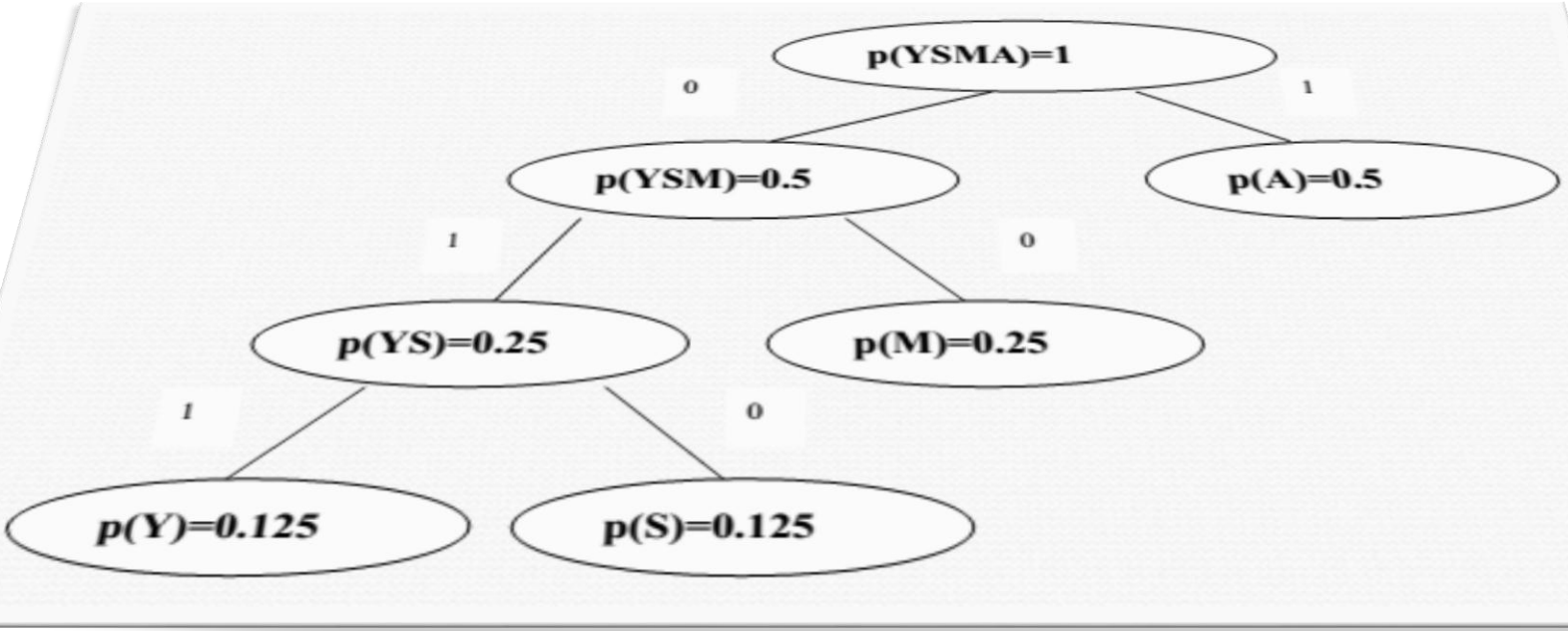
CONTOH TEKNIK KOMPRESI TEKS :

Run-Length-Encoding (RLE)

- Kompresi data teks dilakukan jika ada beberapa huruf yang sama yang ditampilkan berturut-turut:
- Mis: Data: ABCCCCCCCCCDEFGGGG = 17 karakter
- RLE tipe 1 (min. 4 huruf sama) : ABC!8DEFG!4 = 11 karakter
- RLE tipe 1 menggunakan: tanda ‘!’.
- Kelemahan: Jika ada karakter angka, tidak bisa membedakan mulai dan akhir.

Static Huffman Coding (SHC)

- Misal: 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
  - Frekuensi karakter dari string yang akan dikompres dianalisa terlebih dahulu dengan huffman coding.
  - 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)



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

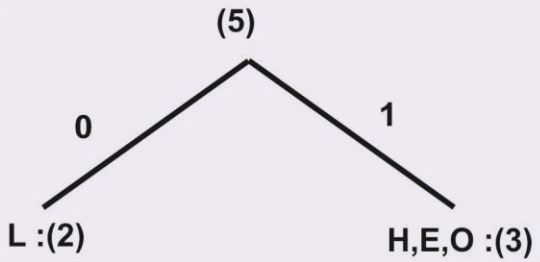
Shannon-Fano Algorithm

- Dikembangkan oleh Shannon (Bell Labs) dan Robert Fano (MIT)
- Contoh :
  - H E L L O

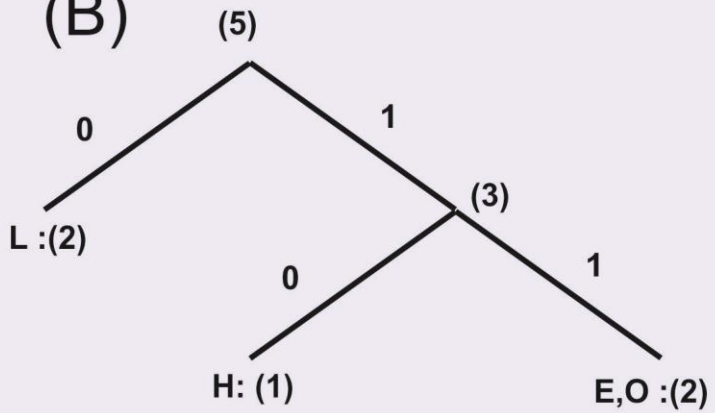
Simbol	H	E	L	O
Jumlah	1	1	2	1

- Algoritma :
  - Urutkan simbol berdasarkan frekuensi kemunculannya
  - 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.

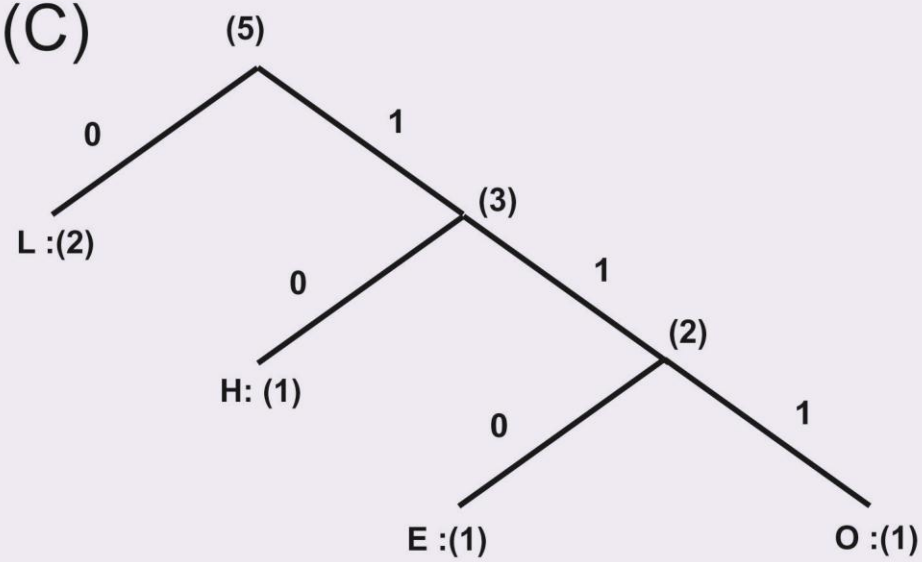
(A)



(B)



(C)



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

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:
  - 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
  - Bzip2, dan lain-lain
- Aplikasi: WinZip oleh Nico-Mak Computing.

RAR File

- Ditemukan oleh Eugene Roshal,
- Sehingga RAR merupakan singkatan dari Roshal Archive pada 10 Maret 1972 diRusia
- 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

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

SEKIAN