ANALISA KOMPRESI DATA TEXT DENGAN MENGGUNAKAN METODE *HUFFMAN* DAN *HALF-BYTE*

Oleh:

FAISAL MUIS [1] , KHUSNUL NOVIANINGSIH [2] , YEFFRY HANDOKO PUTRA [3]

[1)] JURUSAN TEKNIK INFORMATIKA (10199085)

[2)] DOSEN JURUSAN TEKNIK INFORMATIKA

[3)] DOSEN TEKNIK KOMPUTER

FAKULTAS TEKNIK DAN ILMU KOMPUTER

UNIVERSITAS KOMPUTER INDONESIA

BANDUNG

1. PENDAHULUAN

Data komputer pada dasarnya terdiri atas 3 macam yakni suara, text dan gambar. Dan ada pula hasil dari kombinasi diantara ketiganya. Dari ketiga data di atas data textlah yang dominan dalam sistem komputer yang perlu di kompresi karena ukurannya rata-rata relatif besar dan selalu mengalami perubahan dan penambahan dalam selang waktu tertentu. Sedangkan data yang lainnya biasanya telah mengalami proses kompresi seperti MP3 (Suara), dan Jpg (Gambar).

Kompresi tidak hanya bertujuan untuk menghemat kapasitas memori saja tetapi juga dapat dimanfaatkan dalam proses transmisi dan enkripsi data dalam jaringan. Secara umum kompresi data bekerja dengan mengambil lambang masukan atau simbol masukan, memproses dan menuliskan kode ke file yang telah dimampatkan. Ada dua teknik pemampatan data text yakni menggunakan Dictionary Based atau Statistical Modeling. Dictionary Based adalah mengubah simbol secara per-byte data menjadi kode dengan panjang yang tetap. Sedangkan Statistical Modeling adalah mengubah simbol per-byte berdasarkan probability masing-masing karakter menjadi kode dengan panjang tidak tetap.

Tujuan penelitian ini adalah untuk mencari dan memilih teknik yang tepat dalam kompresi data text dengan studi kasus terhadap algoritma *Huffman* dan *Half-Byte*.

2. BATASAN MASALAH

Pembahasan masalah dibatasi hanya pada pengukuran nilai estimasi penyusutan terhadap file text dengan ukuran minimal 250 byte dengan menggunakan bantuan program.

3. HASIL dan PEMBAHASAN

Kompresi adalah suatu cara yang dipakai dalam sistem komputer dengan tujuan untuk menyusutkan ukuran suatu data atau file menjadi lebih kecil dari ukuran sebenarnya.

Secara umum kompresi data bekerja dengan cara mengambil lambang masukan, memproses dan menuliskan kode baru ke file yang telah dimampatkan

Algoritma yang dipakai dalam proses pemampatan data dapat bermacam-macam dan beraneka ragam. ini tercermin dari banyaknya teknik, metode dan program kompresi yang berkembang dan yang ada sekarang. Contohnya adalah winzip, winrar, dan winace.

Akan tetapi kesemuanya dapat dikelompokkan ke dalam dua teknik utama yakni *lossy compression* dan *lossles compression*

lossy compression adalah teknik pemampatan data dengan cara menghilangkan atau membuang bit-bit yang kurang dominan atau dianggap kurang perlu sehingga akan menghasilkan jumlah bit yang kecil atau sedikit. Teknik ini sangat cocok digunakan untuk mengkompresi data gambar karena pada file gambar tidak mengutamakan ketelitian akan tetapi visualisasi objek yang dilihat mata manusia

lossles compression adalah membuat duplikasi data imputan dengan panjang bitnya lebih pendek dari aslinya. Teknik ini sangat cocok digunakan untuk mengkompresi data alfanumerik baik berupa text ataupun numeric seperti data yang ada dalam database maupun spreadsheets.

Untuk membuat suatu duplikasi data menggunakan teknik lossles dapat memakai *Dictionary Based* atau *Statistikal Modeling*

Dictionary Based adalah teknik pembuatan duplikasi data dengan cara mengubah simbol per-byte dari tiap karakter menjadi kode baru yang panjang tiap symbol atau karakternya seragam, sesuai panjang yang ingin dipakai perancang program kompresi yang akan dibangun

Statistikal Modeling dalam proses pembuatan duplikasi data melakukan pendekatan yang berbeda yang mana, simbol baru yang terbentuk panjangnya tidak harus seragam akan tetapi harus mengikuti probabilitas kemunculannya. Dalam artian jika karakter sering muncul maka jumlah bits-nya dibikin lebih pendek dari yang jarang muncul.

Algoritma *Huffman*

Dasar pemikiran algoritma ini adalah bahwa setiap karakter ASCII biasanya diwakili oleh 8 bits. Jadi misalnya suatu file berisi deretan karakter “ABACAD” maka ukuran file tersebut adalah 6 x 8 bits = 48 bit atau 6 bytes. Jika setiap karakter tersebut di beri kode lain misalnya A=1, B=00, C=010, dan D=011, berarti kita hanya perlu file dengan ukuran 11 bits (10010101011), yang perlu diperhatikan ialah bahwa kode-kode tersebut harus unik atau dengan kata lain suatu kode tidak dapat dibentuk dari kode-kode yang lain. Pada contoh diatas jika kode D kita ganti dengan 001, maka kode tersebut dapat dibentuk dari kode B ditambah dengan kode A yaitu 00 dan 1, tapi kode 011 tidak dapat dibentuk dari kode-kode yang lain. Selain itu karakter yang paling sering muncul, kodenya diusahakan lebih kecil jumlah bitnya dibandingkan dengan karakter yang jarang muncul. Pada contoh di atas karakter A lebih sering muncul (3 kali), jadi kodenya dibuat lebih kecil jumlah bitnya dibanding karakter lain.

Untuk menentukan kode-kode dengan kriteria bahwa kode harus unik dan karakter yang sering muncul dibuat kecil jumlah bitnya, kita dapat menggunakan konsep pohon biner.

Algoritma *Half Byte*

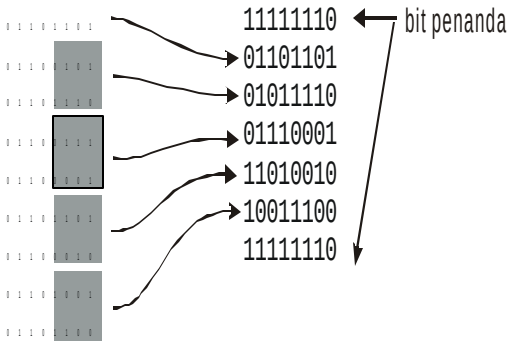
Algoritma *Half-Byte* memanfaatkan empat bit sebelah kiri yang sering sama secara berurutan terutama pada file-file text. Misalnya pada suatu file text berisi tulisan “mengambil”, dalam heksadesimal dan biner karakter-karakter tersebut diterjemahkan sebagai :

Tabel 3.1 Kode ASCII karakter

Karakter	Heksadesimal	Biner
m	6D	01101101
e	65	01100101
n	6E	01101110
g	67	01100111
a	61	01100001
m	6D	01101101
b	62	01100010
i	69	01101001
l	6C	01101100

Jika anda perhatikan karakter-karakter tersebut memiliki empat bit sebelah kiri yang sama yaitu 0110. Gejala seperti inilah yang dimanfaatkan oleh Algoritma *Half-Byte*.

Saat karakter yang empat bit pertamanya sama diterima secara berderet tujuh kali atau lebih, algoritma ini mengkompres data tersebut dengan bit penanda kemudian karakter pertama dari deretan empat bit yang sama diikuti dengan pasangan empat bit terakhir deretan berikutnya dan ditutup dengan bit penutup. Algoritma ini paling efektif pada file-file text dimana biasanya berisi text-text yang memiliki empat bit pertama yang sama. Agar lebih jelas algoritma Half-Byte dapat digambarkan sebagai berikut :



Gambar 3.1. Ilustrasi pemampatan dengan algoritma *Half-Byte*

4. KESIMPULAN

1. *Half-Byte* lebih cocok untuk mengkompresi data text yang ukurannya relatif kecil (lebih kecil dari 300 Byte) serta jenis datanya adaptif. Sedangkan untuk data yang ukuran diatasnya *Huffman* lebih tepat.
2. Dilihat dari kemampuan dan tingkat penyusutannya untuk mengkompresi data berbentuk text lebih baik menggunakan *Huffman* karena tidak memiliki keterbatasan kerja walaupun karakteristik datanya beranekaragam.
3. Teknik *Statistikal Modeling* harus menyertakan seluruh informasi kodenya ke file hasil kompresi sedangkan *Dictionary Based* bisa tidak menyertakan informasi kode dan bila diperlukan paling hanya menyertakan bit penanda saja.

DAFTAR PUSTAKA

Salomon D., *Data Compression 2nd Edition : The Complete Reference*, Springer- Vertag New York, 2000.

Nelson, Mark.,*The Data Compression Book : Second Edition*, Jean-Loup Gailly New York, 1996.

Kristanto Andri., *Keamanan Data Pada Jaringan Komputer*, Gava Media Yogyakarta, 2003.

Santosa Insap., *Struktur Data: Menggunakan Turbo Pascal 6.0*, Andi Offset Yogyakarta,1995.

Kurniawan Andi., *Pemrograman Borland Delphi 7.0*, Wahana Komputer Semarang, 2003.

http://www.data-compression.com/lossless.shtml

http://www.data-compression.com/huffman.shtml