PERBANDINGAN ALGORITMA HUFFMAN DAN ALGORITMA SHANNON-FANO PADA PROSES KOMPRESI BERBAGAI TIPE FILE

Irwan Munandar
Balai Pendidikan dan Pelatihan
Tambang Bawah Tanah

I. Pendahuluan

Keterbatasan komputer dalam meyimpan, mengolah dan mentransfer data yang besar memerlukan ruang kapasitas penyimpanan(*storage*) yang sangat besar. Dengan perkembangan teknologi yag ada maka untuk mengurangi penyimpanan data dan meningkatkan saluran komunikasi didalam jaringan dibutuhkan aplikasi yang mendukung hal-hal tersebut yaitu aplikasi kompresi data.

Kompresi data merupakan proses pengubahan sekumpulan data menjadi suatu bentuk kode lain yang lebih efesien atau berukuran lebih kecil dari ukuran aslinya[2]. Keuntungan dari kompresi data antara lain : efesiensi penyimpanan data, mempercepat waktu transfer data, meningkatkan saluran komunikasi dalam jaringan, dan memperkecil kemungkinan data mengalami kerusakan[2].

Klasifikasi teknik kompresi dibagi 2 yaitu teknik *lossless* dan teknik *lossy[3]*. Teknik *lossless* merupakan teknik kompresi yang menjamin data input dan output (hasil kompresi) adalah sama dari segi keakuratan yang dikandungnya sehingga tidak boleh adanya kerusakan satu bit saja yang mengakibatkan hasil kompresi tidak bermanfaat[3] contoh : file zip, RAR. Teknik *lossy* merupakan teknik kompresi yang mengakibatkan hilangnya data-data tertentu untuk mencapai rasio yang lebih baik[3], contoh File JPEG, MPEG.

Pada artikel ini penulis melakukan analisa algoritma kompresi yaitu algoritma yang akan dibandingkan adalah algoritma shannon Fano dan Algoritma Huffman, keduanya termasuk dalam klasifikasi teknik yang sama (*lossless*). Tujuan penulis dalam artikel ini yaitu mengetahui kelebihan dan kekurangan masing-masing algoritma kompresi dengan mengukur rasio file hasil kompresi terhadap file asli dan kecepatan kompresinya , Dengan mengembangkan program aplikasi untuk melakukan proses kompresi dan dekompresi.

II. Metode dan Landasan Teori

Dalam artikel ini penulis hanya menggunakan 2 algoritma kompresi yaitu algoritma Shannon Fano dan Algoritma Huffman. Metodologi yang digunakan dalam artikel ini yaitu dengam melakukan kajian teori, merancang pengembangan program, membangun aplikasi kompresi dan dekompresi , serta menganalisis hasil kompresi dan dekompresi dengan menggunkanan data yang sudah digunakan dalam penelitian-penelitian sebelumnya[3][4] yaitu 12 (dua belas) golongan tipe file.

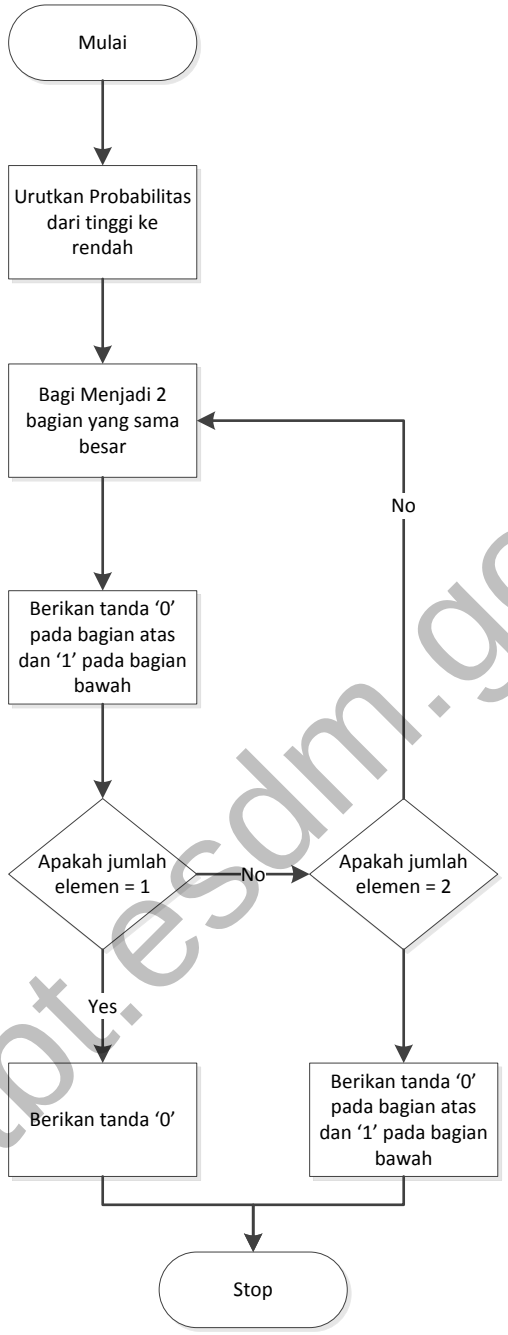
2.1 Algoritma Shannon Fano

Teknik coding Shanon-Fano merupakan salah satu algoritma pertama yang tujuannya adalah membuat *codeword* dengan redudansi minimum. Ide dasar untuk dari membuat *codeword* dengan *variable-code length*, seperti kode huffman yang ditemukan beberapa tahun kemudian. Seperti yang disebutkan sebelumnya algoritma shanon-Fano didasarkan pada *variable length-word*, yang berarti beberapa simbol pada pesan (yang akan dikodekan) direpresentasikan dengan *codeword* yang lebih pendek dari simbol yang ada dipesan, maka *codeword* semakin pendek. Dalam memperkirakan panjang setiap *codeword* maka dapat ditentukan dari probabilitas setiap simbol yang dipresentasikan oleh *codeword* tersebut[6].

Shannon-Fano *coding* menghasilkan *codeword* yang tidak sama panjang, sehingga kode tersebut bersifat unik dan dapat dikodekan. Secara keseluruhan penerapan algoritma shannon-fano pada proses kompresi dapat dilihat pada prosedur berikut ini[6] :

1. Menyusun probabilitas simbol dari sumber yang paling tinggi ke yang paling rendah.
2. Membagi menjadi 2(dua) bagian yang sama besar, dan memeberikan nilai ‘0’ untuk bagian atas dan ‘1’ untuk bagian bawah
3. Ulangi langkah ke 2(dua), setiap pembagian dengan probabilitas yang sama sampai dengan tidak mungkin dibagi lagi.
4. *Encode* setiap simbol asli dari sumber menjadi urutan biner yang dibangkitkan oleh setiap proses pembagian tersebut.

Dari langkah-langkah yang telah dipaparkan pada algortima shanon-fano maka langkah-langkah tersebut dapat dilihat pada gambar 2.1



Gambar 2.1 Alur Program Algoritma Shannon-Fano[6]

2.2 Algoritma Huffman

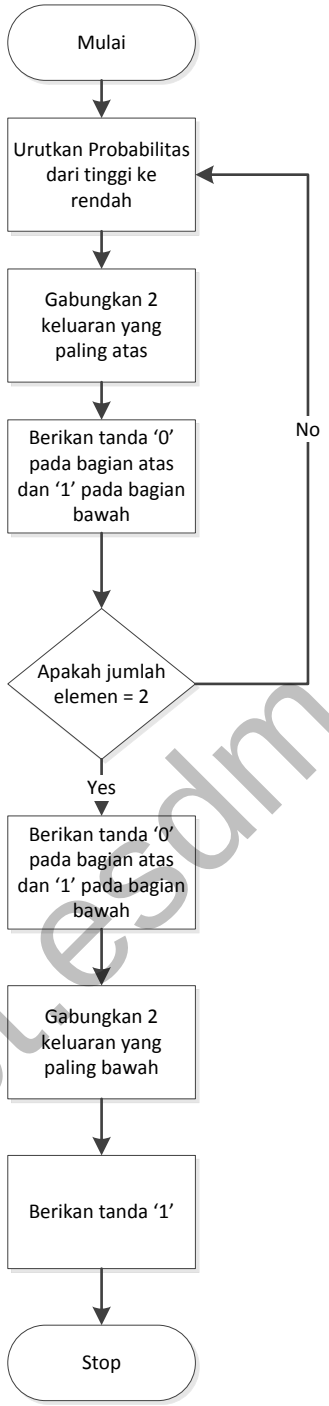
Algoritma huffman dibuat oleh seorang mahasiswa MIT bernama David Huffman pada tahun 1952, merupakan salah satu metode paling lama dan paling terkenal dalam kompresi teks. Algortima huffman menggunakan prinsip pengkodean yang mirip dengan kode morse, yaitu tiap karakter (simbol) dikodekan hanya dengan rangkaian bit, dimana karakter yang sering muncul dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang muncul dikodekan dengan rangkaian bit yang lebih panjang[6].

Berdasarkan tipe kode yang digunakan untuk mengubah pesan awal (isi data yang diinputkan) menjadi sekumpulan *codeword*, algoritma huffman termasuk kedalam kelas algoritma yang menggunakan metode statik. Metode statik adalah metode yang selalu menggunakan peta kode yang sama, metode ini membutuhkan 2 fase (*two phase*): fase pertama untuk menghitung probabilitas kemunculan tiap simbol dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan di transmisikan[6].

Sedangkan berdasarkan teknik pengkodean simbol yang digunakan, algoritma huffman menggunakan metode *symbolwise*. Metode *symbolwise* adalah metode yang menghitung peluang kemunculan dari setiap simbol dalam satu waktu, dimana simbol sering muncul diberikode lebih pendek dibandingkan simbol yang jarang muncul. Algoritma dari huffman *encoding* (cara menyusun string biner dari teks yang ada) adalah sebagai berikut ini[6] :

1. Pengurutan keluaran sumber mulai dari probabilitas paling rendah ke paling tinggi.
2. Membandingkan 2 keluaran yang sama dekat kedalam satu keluaran yang probabilitasnya merupakan jumlah dari probabilitas sebelumnya.
3. Apabila setelah dibagi masih terdapat 2 keluaran, maka lanjut kelangkah berikutnya, namun apabila masih terdapat lebih dari dua, kembali kelangkah 1
4. Memberikan nilai ‘0’ dan ‘1’ untuk kedua keluaran.
5. Apabila sebuah keluaran merupakan hasil dari penggabungan 2 keluaran dari langkah sebelumnya, maka berikan tanda ‘0’ dan ‘1’ untuk *codeword* nya, ulangi sampai keluaran merupakan satu keluaran berdiri sendiri.

Dari langkah-langkah yang telah dipaparkan pada algortima huffman maka langkah-langkah tersebut dapat dilihat pada gambar 2.2

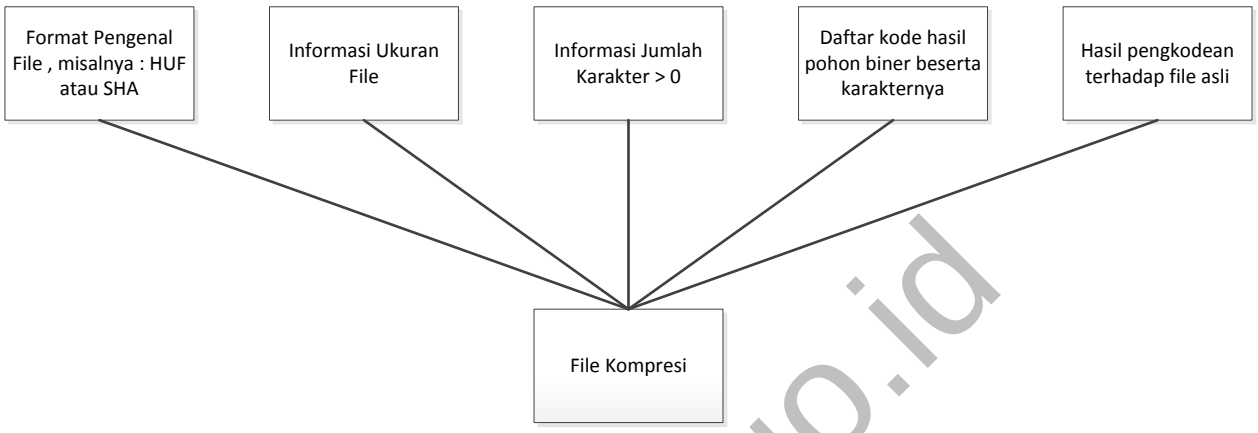


Gambar 2.2 Alur program Algoritma Huffman[6]

III. Hasil dan Pembahasan

Proses Kompresi algoritma Shanon fano dan Algoritma Huffman pada dasarnya sama terdiri dari 3 tahapan yaitu : Pengurutan sumber pesan (*Source Message*), pembuatan pohon dan daftar kode (*List Code*)[6]. pada file hasil kompresi harus ditandai pada awal datanya sehingga sewaktu pengambilan ke file asli dapat dikenali, apakah file tersebut benar

merupakan hasil kompresi dengan algoritma yang di maksud, untuk lebih jelasnya dapat dilihat pada gambar 3.1.



Gambar 3.1 Unsur-unsur File Hasil Kompresi[6]

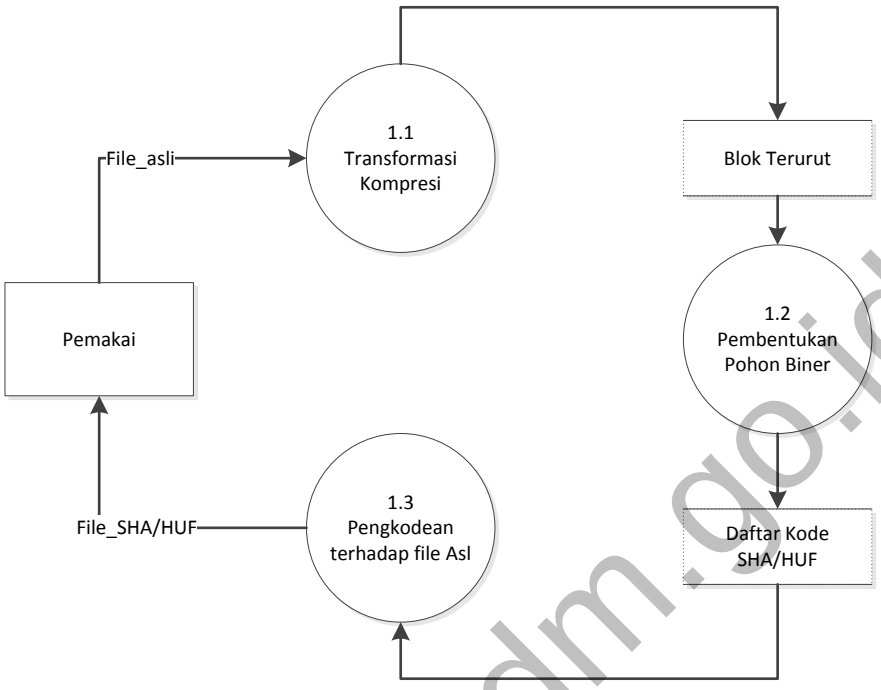
Proses dekompresi merupakan proses untuk mengambalikan file hasil kompresi menjadi file aslinya tanpa menghilangkan satu bit unsur informasinya. Dekompresi berarti menyusun kembali data dari string biner menjadi sebuah karakter kembali. Pada intinya proses dekompresi file hasil kompresi algoritma Shannon-Fano dan algoritma Huffman menggunakan metode yang sama. Langkah-langkah proses dekompresi terdiri dari 3 (tiga) tahap yaitu : pengkodean balik daftar kode (list kode) dari hasil pohon biner, pengkodean Balik karakter hasil kompresi menjadi rangkaian bit, transformasi balik (invers transform) menjadi sebuah file asal.[6]

3.1 Diagram Aliran Data

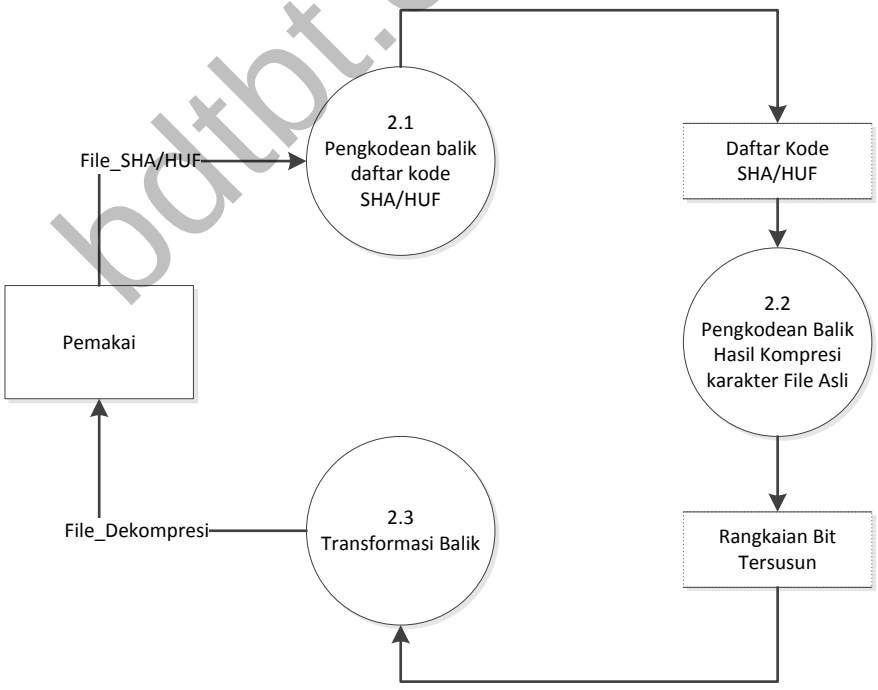
Perancangan sistem adalah pendefinisian dan kebutuhan –kebutuhan fungsional dan persiapan untuk rancang bangun implementasi ; menggambarkan bagaimana suatu sistem dibentuk. Diagram Aliran Data (DAD) adalah representasi dari sebuah sistem, DAD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data diantara komponen-komponen ssbuah sistem, dan penyimpanan dari data tersebut[1].

Dalam sistem kompresi data menggunakan algoritma Shannon-Fano dan Algoritma Huffman ini aliran data dimulai dari inputan berupa file asli atau file terkompresi, yang selanjutnya akan melalui beberapa proses baik nanti akan di kompresi atau di dekompresi. Berdasarkan analisa terdapat 1 entitas yaitu pemakai yang berinteraksi langsung terhadap proses kompresi dan dekompresi sebagai sumber data dan tujuan data, hasil dari kajian

penulis maka DAD yang di dapat sampai level 2 di paparkan pada gambar 3.2 dan gambar 3.3.



Gambar 3.2 DAD Level 2 Proses Kompresi



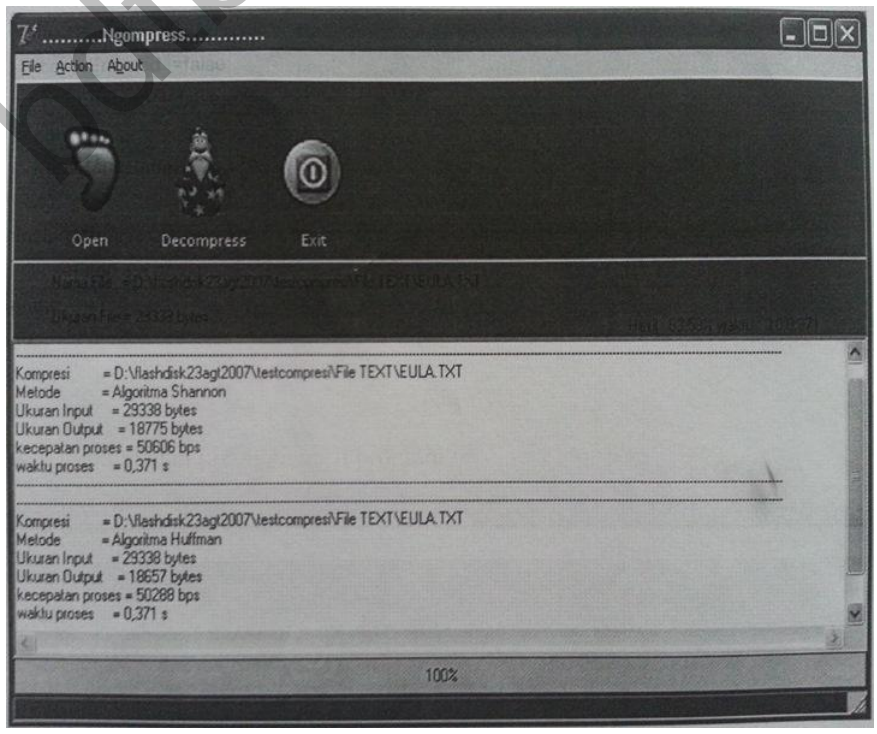
Gambar 3.3 DAD Level 2 Proses dekompresi

3.2 Hasil perancangan program

Aplikasi yang akan dibangun berupa perangkat lunak untuk kompresi dan dekompresi yang menggunakan algoritma pengkodean Shannon-Fano dan Algoritma pengkodean Huffman. Tujuan dari pengembangan perangkat lunak ini adalah untuk menghasilkan sebuah aplikasi kompresi dan dekompresi dengan performansi tinggi. Pada perancangan program aplikasi ini terdiri dari beberapa menu yang didesain dalam proses kompresi dan dekompresi data, yaitu :

1. Menu *Open file* adalah menu untuk membuka file dan memilih file yang akan dikompresi maupun di dekompresi
2. Menu *Save File* adalah untuk menyimpan file hasil kompresi atau dekompresi di dalam storage
3. Menu *Compress* adalah menu untuk mengkompresi data menjadi ukuran lebih kecil dari data aslinya dengan menggunakan algoritma shanon-Fano dan Algoritma Huffman
4. Menu *Decompress* adalah menu untuk mengembalikan data yang telah dikompres ke dalam bentuk asli tanpa adanya kehilangan(loss) dari data aslinya
5. Menu *Quit* adalah menu keluar aplikasi

Dari penjabaran menu yang di desain sebelumnya untuk lebih jelasnya dapat dilihat pada hasil programnya pada gambar 3.4.



Gambar 3.4 hasil tampilan program aplikasi kompresi dan dekompresi

3.3 Hasil implementasi Algoritma Shanon Fano

Percobaan kompresi yang dilakukan terhadap beberapa golongan tipe file dengan algoritma shanon-fano menunjukan hasil seperti pada tabel 3.1.

Tabel 3.1 Hasil Kompresi Shannon-Fano

Tipe File	Total Ukuran File (bytes)	Shannon –Fano		
		Total Ukuran File (bytes)	Rasio (%)	Rata-rata kecepatan (bps)
Calgary Corpus	3251493	1827370	56,20%	55.051
Canterbury Corpus	11159482	4958901	44,44%	75.386
Aplikasi	2092814	1731814	82,75%	58.605
Hasil Kompresi	1474771	1477182	100,16%	58.831
Object	1106530	812294	73,41%	37.258
Database	1178502	779781	66,17%	73.683
Executable	4938680	4065056	82,31%	58.478
Gambar	1565130	1531321	97,84%	67.070
Multimedia	4877883	4818614	98,78%	59.802
Source Code	146284	92621	63,32%	22.164
Teks	506015	353179	69,80%	52.787
UNIX	73645	59201	80,39%	25.484

Dari perhitungan diatas yang merujuk pada tabel 4.1 dapat dilihat bahwa golongan tipe file Catenbury Corpus mengalami pengurangan yang cukup baik dibanding dengan golongan tipe-tipe file yang lain yaitu 44,44% dari total ukuran asli menghemat sekitar 6.200.581 bytes dan jika dilihat pada golongan tipe file yang telah terkompresi malah mengakibatkan penambahan dari total ukuran aslinya yaitu 1.477.182 atau 100,16% jadi kelebihan 0,16%, perubahan-perubahan ukuran hasil kompresi tersebut dapat disebabkan jumlah karakter yang sama dan begantung pada probabilitas yang dibangun dan tidak tergantung pada urutan aliran datanya.

Sedangkan kecepatan kompresi menggunakan algoritma Shannon-Fano yang bisa dilihat pada tabel 3.1 bahwa untuk golongan tipe file Catenbury Corpus menunjukan proses kompresi yang cukup baik yaitu rata-rata 75.386 bps atau memerlukan waktu rata-rata sekitar 21,9 detik dan rata-rata ukuran file hasil kompresi shannon-fano tipe golongan file catenbury corpus sebesar 1.652.967 bytes. Perubahan-perubahan kecepatan kompresi dapat disebabkan jumlah karakter yang sama dan kecepatan komputer yang digunakan.

3.4 Hasil Implementasi Algoritma Huffman

Setelah memahami langkah-langkah atau cara kerja kompresi menggunakan algoritma huffman. Dengan demikian sehingga percobaan kompresi yang dilakukan terhadap beberapa golongan tipe file dengan algoritma huffman ditunjukan pada tabel 3.2.

Tabel 3.2 Hasil Kompresi Huffman

Tipe File	Total Ukuran File (bytes)	Huffman		
		Total Ukuran File (bytes)	Rasio (%)	Rata-rata kecepatan (bps)
Calgary Corpus	3251493	1819270	55,95%	58.788
Canterbury Corpus	11159482	4937544	44,25%	72.988
Aplikasi	2092814	1722660	82,31%	71.994
Hasil Kompresi	1474771	1474021	99,95%	88.787
Object	1106530	806133	72,85%	51.272
Database	1178502	777866	66,00%	70.213
Executable	4938680	4031255	81,63%	77.146
Gambar	1565130	1525130	97,44%	73.801
Multimedia	4877883	4795868	98,32%	69.926
Source Code	146284	91855	62,79%	21.699
Teks	506015	346819	68,54%	49.699
UNIX	73645	58795	79,84%	33.927

Pada tabel 3.2 dapat dilihat bahwa hasil kompresi pada beberapa tipe file tersebut mengalami perubahan/penurunan ukuran file yang tidak drastis, golongan tipe file catenbury corpus mengalami pengurangan yang cukup baik dibanding dengan golongan tipe-tipe file yang lain yaitu 44,25% dari total ukuran asli menghemat sekitar 6.221.938 bytes dan jika golongan tipe file yang telah terkompresi hanya mengalami sedikit penurunan sekitar 0,05% dari total ukuran aslinya yaitu 1.474.021 bytes atau 99,95%.

Kecepatan kompresi menggunakan algoritma huffman yang bisa dilihat pada tabel 3.2 bahwa golongan tipe file hasil kompresi menunjukan kecepatan proses kompresi yang cukup baik yaitu rata-rata 88,787 bps atau memerlukan waktu rata-rata sekitar 8,3 detik dari rata-rata ukuran file hasil kompresi huffman tipe golongan file hasil kompresi sebesar 737.011 bytes. Perubahan-perubahan kecepatan kompresi dapat disebabkan jumlah karakter yang sama dan kecepatan komputer yang digunakan serta perubahan ukuran file bergantung pada probabilitas yang dibangun dan tidak tergantung pada urutan aliran datanya.

IV. Kesimpulan

Berdasarkan hasil analisa yang dilakukan bahwa secara rata-rata algoritma Huffman lebih unggul dari algoritma Shannon baik dari segi rasio maupun dari sisi kecepatan proses kompresi. Secara rata-rata algoritma Huffman menghasilkan rasio file hasil kompresi yang lebih baik dibanding dengan algoritma Shannon-fano yaitu sebesar 75,8%. Algoritma Huffman membutuhkan waktu yang lebih baik dengan rata-rata yaitu 61,687 byte/sec. Untuk kategori file multimedia , file hasil kompresi dan file gambar menghasilkan kompresi yang kurang baik karena pada umumnya karakteristik file-file tersebut sedikit ragam simbol.

DAFTAR PUSTAKA

[1]. Hartono, Jogiyanto. *Pengenalan komputer*. Yogyakarta : penerbit ANDI, . 1999.

[2]. *Defenition Data Compression*. http://www.dogma.net/DataCompression , diakses tahun 2007

[3]. *Data Compression.* http://www.geocities.com/m99datacompression/papers/rle2.htm , diakses tahun 2007

[4]. *Data Compression.* http://www.archives.org/web/20020214085725/http://rasip.fer.hr/research , diakses tahun 2007

[5]. *Data Compression*, http://www.cs.mcgill.ca/-nferns/cs251 diakses tahun 2007

[6]. Nelson, Mark. Gailly, Jean Loup. *The data Compression Book, Seceond Edition.* New York : M&T Book, 1996

[7]. Tokheim, Roger.L. *Prinsip-prinsip digital.* Jakarta : Penerbit Erlangga. 1994