Volume VI 2011

ISSN: 1907-3526

SRITI

Seminar Nasional Riset Teknologi Informasi

Proceeding

Seminar Nasional Riset Teknologi Informasi 2011

"Implementasi Mobile Computing di Dunia Pendidikan dan Industri: Sebuah Peluang dan Tantangan"

Yogyakarta, 17 September 2011

Komputasi

Teknologi Web

Keamanan Sistem

Kecerdasan Buatan

Teknologi Basis Data

Pemodelan dan Aplikasi

Pengolahan Citra, Grafika dan Multimedia

Komunikasi Data, Jaringan Komputer dan Sistem Kendali

diselenggarakan Oleh :

YAYASAN PENDIDIKAN WIDYA BAKTI
STMIK
AKAKOM
YOGYAKARTA
Yang Pertama dan Utama

KATA PENGANTAR

Puji syukur marilah kita panjatkan ke hadirat Tuhan YME yang telah melimpahkan rahmat dan hidayahnya, sehingga dapat terselesaikannya penyusunan *Proceeding* SRITI 2011 ini. Buku ini memuat naskah hasil penelitian dari berbagai bidang kajian yang akan dipresentasikan pada Seminar Riset Teknologi Informasi (SRITI) 2011 ke-6 yang telah menjadi agenda tahunan dari Pusat Penelitian dan Pengembangan STMIK AKAKOM Yogyakarta dan sekaligus sebagai rangkaian dari peringatan 32 tahun STMIK AKAKOM.

Call for paper pada seminar SRITI 2011 naskah yang dikirimkan kepada Panitia sudah dalam bentuk *full paper*, sehingga naskah yang masuk ke panitia merupakan naskah final hasil penelitian yang siap dipublikasikan. Naskah yang masuk ke panitia selanjutnya di-*review* oleh para pakar di bidangnya. Atas kesediaan, kerjasama dan konsistensinya dalam me-*review* seluruh naskah yang dikirimkan, panitia mengucapkan banyak terima kasih.

Kegiatan SRITI 2011 mengambil tema tentang "*Implementasi Mobile Computing di Dunia Pendidikan dan Industri: Sebuah Peluang dan Tantangan*" direncanakan dapat menyidangkan secara paralel sesuai dengan kelompok kajian ilmu dalam waktu satu hari. Panitia menyadari bahwa, hingga saat ini masih banyak *paper contents* yang belum mengacu pada tema, namun mengingat lingkup bidang kajian teknologi informasi yang sangat luas, maka ke depan diharapkan masih dapat ditingkatkan kesesuaian, kedalaman, maupun spektrum kajiannya.

Meskipun kegiatan seminar dan pendokumentasian naskah dalam *proceeding* ini telah dipersiapkan dengan baik, namun kami menyadari masih terdapat banyak kekurangannya. Untuk itu, panitia mohon maaf yang sebesar-besarnya dan terima kasih atas kepercayaan serta kerjasamanya dalam kegiatan ini. Kritik dan saran perbaikan sangat diharapkan untuk penyempurnaan di masa mendatang, yang dapat dikirimkan melalui e-mail sriti@akakom.ac.id.

Kepada semua pihak yang terlibat, baik langsung maupun tidak langsung dalam penyusunan proceeding SRITI 2011, panitia mengucapkan terima kasih.

Terima kasih.

Yogyakarta, 17 September 2011

Panitia SRITI 2011
Ketua Pelaksana,

Drs. Tri Prabawa, M.Kom.

DAFTAR ISI

Perbandingan Kinerja Kompresi File Menggunakan Metode Huffman dengan Adaptive Huffman

Sudarmanto

STMIK AKAKOM Yogyakarta
Jl. Raya Janti 143. Karangjambe Yogyakarta
email: darmanto@akakom.ac.id

Abstrak

Algoritma Huffman merupakan salah satu teknik kompresi yang melibatkan distribusi frekuensi suatu simbol untuk membentuk kode yang unik. Distribusi frekuensi simbol akan mempengaruhi panjang kode Huffman-nya. Semakin sering simbol tersebut muncul dalam sebuah file maka panjang kode Huffman yang dihasilkan akan semakin pendek.
Algoritma Adaptive Huffman memiliki tujuan sama dengan algoritma Huffman, yaitu bertujuan membuat pohon kode, yang nantinya kode-kode ini digunakan untuk mewakili setiap simbol. Berbeda dengan algoritma Huffman, dimana sebelum membentuk pohon kode terlebih dahulu dihitung besar frekuensi dari setiap simbol yang akan dikodekan. Sedangkan Adaptive Huffman menggunakan suatu pohon kode yang dapat menyesuaikan bentuknya pada saat simbol dimasukan kedalam pohon, tanpa harus mengetahui besar frekuensi dari setiap simbol yang akan dikodekan.
Hasil yang diperoleh adalah bahwa tidak ada perbedaan rasio kompresi antara hasil kompresi dengan algoritma Huffman dengan Adaptive Huffman. Sementara itu waktu yang dibutuhkan metode Huffman untuk melakukan proses kompresi lebih cepat dibandingkan dengan metode Adaptive Huffman. Metode Adaptive Huffman lebih lambat karena proses pembentukan pohon Huffman dilakukan saat pembacaan karakter, sehingga bentuk pohon akan berubah pada saat proses pembacaan karakter karena terjadinya perubahan frekuensi.

***Kata Kunci:** Adaptive Huffman, Distribusi frekuensi, Huffman, Kompresi, Pohon kode*

1. Pendahuluan

1.1. Latar Belakang

Dalam dunia komputer dikenal istilah pemampatan/kompresi data, kompresi merupakan teknik pemampatan data sehingga diperoleh data baru yang memiliki ukuran lebih kecil dari data aslinya, dengan kata lain media penyimpanan yang diperlukan untuk menyimpan data akan menjadi lebih kecil.

Ada dua jenis metode kompresi yaitu *lossless* dan *lossy*. Kompresi yang bersifat *lossless* tidak menghilangkan informasi-informasi dari file asli sehingga cocok untuk diterapkan untuk file dokumen. Sebaliknya metode kompresi *lossy* akan menghilangkan informasi yang dianggap tidak signifikan, sehingga cocok terapkan untuk file multimedia berukuran besar.

Teknik pemampatan data dapat dikategorikan menurut jenis data yang akan dimampatkan, yaitu :

- Teknik kompresi untuk citra diam (*still image*) antara lain : JPEG (*Joint Photographic Experts Group*), GIF (*Graphics Interchange Format*), dan RLC (*Run Length Coding*).
- Teknik kompresi untuk citra bergerak (*motion picture*) contohnya : MPEG (*Motion Picture Expert Group*).
- Teknik kompresi untuk data umum antara lain : LZW (*Lempel-Ziv Coding*) dan Huffman.
- Teknik kompresi untuk data sinyal *speech* antara lain : PCM (*Pulse Code Modulation*), SBC (*Sub-Band Coding*).

Teknik kompresi umum dapat mengkompresi semua jenis data tetapi tidak seefisien teknik kompresi khusus. Contoh perangkat lunak yang digunakan untuk mengkompresi data menggunakan teknik kompresi umum adalah Winzip, Winrar dan lain sebagainya. Teknik kompresi umum ini menggunakan metode kompresi bersifat *lossless*.

1.2. Tujuan Penelitian

Dasar dari kompresi adalah meminimalkan jumlah representasi data tanpa menghilangkan informasi penting yang terkandung didalamnya. Pokok masalah dari penelitian ini adalah mengimplementasikan algoritma Huffman dan Adaptive Huffman yang digunakan untuk proses kompresi file. Hasil berbentuk program enkoder untuk mengkompresi sebuah file. Enkoder hanya bisa melakukan kompresi satu file dalam sekali proses. Hasil pengolahan file dengan enkoder berupa file hasil kompresi dengan ekstensi *.huf. Dibuat juga program dekoder yang akan mengembalikan file hasil kompresi menjadi seperti aslinya.

Selanjutnya, dengan dua buah algoritma tersebut, file hasil akan dibandingkan, baik dari sisi rasio kompresi maupun waktu kompresi.

2. Tinjauan Pustaka

2.1 Mengenal Kompresi

Kompresi bekerja dengan mencari pola-pola perulangan pada data dan menggantinya dengan sebuah penanda tertentu. Sebagai contoh sebuah karakter ASCII yang diwakili oleh 8 bit, ini diganti diwakili dengan 2 bit, sehingga bila karakter ini digunakan berulang sebanyak 10 kali maka ukurannya menjadi 10 x 8 bit = 80 bit, tetapi bila karakter ini di wakili dengan 2 bit maka ukurannya menjadi 10 x 2 = 20 bit.

Ada dua jenis metode kompresi yang digunakan yaitu *non-lossy* (sering disebut *lossless*) dan *lossy*.

Kompresi *lossless* merupakan metode kompresi yang mempertahankan keutuhan informasi yang dikandung oleh data, sehingga bila hasil kompresi dilakukan proses dekompresi akan dihasilkan data yang sama seperti sebelum dikompresi. Karena sifat kompresi *lossless* tidak menghilangkan informasi-informasi dalam file asli sehingga cocok diterapkan untuk file dokumen, database dan spreadsheet. Beberapa teknik algoritma yang digunakan untuk kompresi *lossless* antara lain : Huffman, Adaptive Huffman dan LZW.

Kompresi *lossy* merupakan metode kompresi yang menghilangkan informasi yang dikandung untuk data yang dianggap tidak signifikan. Biasanya metode ini diterapkan pada file multimedia berukuran besar. Contohnya adalah file MP3 (Audio), MPEG (video), dan JPEG (citra).

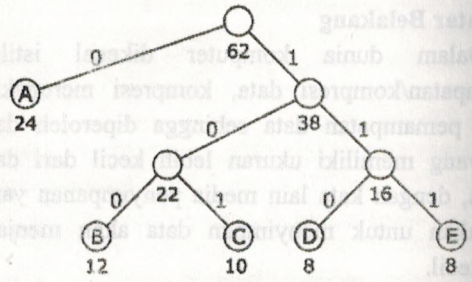
Rasio kompresi merupakan bilangan yang menunjukan perbandingan antara file terkompres dengan besar file asli. Untuk menentukan rasio kompresi digunakan rumus:

$$R = \left(\frac{BesarFileTerkompresi}{BesarFileAsli} \right) x100\%$$

Nilai rasio ini dipengaruhi oleh isi data dari file yang akan dikompres. Bila pada file terdapat banyak data yang kembar maka rasio kompresi akan semakin tinggi, sebaliknya bila sedikit data yang kembar maka rasionya akan semakin kecil.

2.2 Algoritma Huffman

Algoritma Huffman merupakan salah satu teknik kompresi yang melibatkan distribusi frekuensi suatu simbol untuk membentuk kode yang unik. Distribusi frekuensi simbol akan mempengaruhi panjang kode Huffman-nya, semakin sering simbol tersebut muncul dalam sebuah file maka panjang kode Huffman yang dihasilkan akan semakin pendek, contohnya dapat dilihat pada Gambar 1. Pemetaan simbol-simbol ke kode Huffman akan menghasilkan kode-kode biner yang dapat dikonversikan menjadi suatu file baru dengan ukuran file dan jumlah karakter yang lebih kecil dari file awal. Proses inilah yang dikenal sebagai proses kompresi.



Gambar 1 Pohon Huffman

Dasar pemikiran algoritma Huffman adalah 8 bit, sehingga dimisalkan didalam sebuah file berisi deretan simbol ASCII "ABRACADABRA" maka panjang deretan bit dalam file adalah 11 x 8 bit = 88 bit = 11 byte. Jika setiap simbol tersebut diwakili oleh bit kode misalnya A=0, B=01, C=0010, D=0011 dan R=000 maka jumlah bitnya adalah 23 bits (10100010010100111010001) .

Kode Huffman mempunyai beberapa sifat seperti berikut.

1. Setiap kode merupakan prefix kode dan merupakan kode biner. Maksudnya adalah kode biner yang mewakili setiap simbol harus unik, dengan kata lain suatu kode tidak dapat dibentuk dari kode lain.
   Contoh simbol A = 0, B = 11, C = 011.
   Simbol C = 011 kode ini dibentuk bukan dari gabungan antara kode A = 0 dan B = 11 tetapi kode C = 011 adalah kode prefix (kode awal).
2. Simbol yang paling sering muncul, jumlah bit yang mewakilinya lebih kecil dibandingkan dengan karakter lainnya yang jarang muncul.

Seperti yang dijelaskan diatas algoritma Huffman digunakan untuk menetukan kode-kode dengan kriteria bahwa kode harus unik dan karakter yang sering muncul dibuat kecil jumlah bitnya, adapun algortima pembuatan kode Huffman adalah sebagai berikut.

1. Hitung frekuensi kemunculan dari masing-masing simbol
2. Susun simbol berdasarkan jumlah frekuensi, diurutkan dari yang berfrekuensi besar ke kecil.
3. Cari dua simbol dengan frekuensi kecil.
4. Buat akar pohon dari dua simbol berfrekuensi kecil dan tentukan frekuensi akar pohon dengan menjumlahkan frekuensi dua simbol tersebut.
5. Letakkan simbol yang memiliki frekuensi lebih besar di kiri akar pohon dan yang berfrekuensi kecil di kanan akar pohon.
6. Kerjakan langkah 2 sampai dengan langkah 5 sampai semua simbol masuk kedalam pohon Huffman.
7. Buat daftar kode biner dari pohon Huffman yang mewakili simbol.
8. Kodekan setiap simbol berdasarkan dengan kode binernya.

2.3 Algoritma Adaptive Huffman

Algoritma Adaptive Huffman) memiliki tujuan sama dengan algoritma Huffman, yaitu bertujuan membuat pohon kode, yang nantinya kode-kode ini digunakan untuk mewakili setiap simbol yang membentuk pohon. Adaptive Huffman menggunakan suatu pohon kode yang dapat menyesuaikan bentuknya pada saat simbol dimasukan kedalam pohon, tanpa harus mengetahui besar frekuensi dari setiap simbol yang akan dikodekan. Hal ini berbeda dengan algoritma Huffman, yaitu sebelum membentuk pohon kode terlebih dahulu dihitung besar frekuensi dari setiap simbol yang akan dikodekan.

Adaptive Huffman memiliki karakteristik yang sama dengan Huffman yaitu :

1. Setiap kode tidak dibentuk dari kode-kode yang lainnya.
2. Simbol yang paling sering muncul, lebih kecil jumlah bitnya dibandingkan dengan simbol lainnya.

Yang berbeda hanyalah proses pembentukan pohon kodenya saja. Pada Adaptive Huffman proses pembentukan pohon dimulai dengan pohon kosong, sedangkan pada Huffman dimulai dengan dua buah simbol yang memiliki frekuensi terkecil.

Dalam proses pembentukan pohon kode algoritma Adaptive Huffman menggunakan sebuah karakter kontrol (control character atau Escape Code). Karakter kontrol ini digunakan untuk mengindentifikasi simbol baru yang tidak terdapat dalam pohon kode pada saat *run-time*, karakter kontrol ini disebut NYA (*Not Yet Available*) atau NYT (*Not Yet Transmitted*).

Adapun algoritma Adaptive Huffman adalah sebagai berikut.

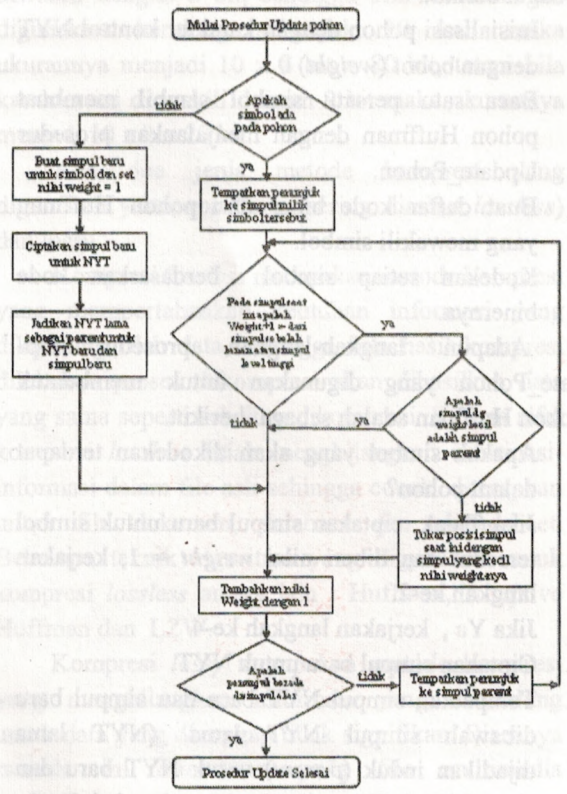
1. Inisialisasi pohon dengan karakter kontrol NYT dengan bobot (*weight*) 0.
2. Baca satu persatu simbol sambil membuat pohon Huffman dengan menjalankan prosedur Update_Pohon.
3. Buat daftar kode biner dari pohon Huffman yang mewakili simbol.
4. Kodekan setiap simbol berdasarkan kode binernya.

Adapun langkah-langkah prosedur Update_Pohon yang digunakan untuk membentuk pohon Huffman adalah sebagai berikut.

1. Apakah simbol yang akan dikodekan terdapat dalam pohon?
   Jika Tidak, ciptakan simpul baru untuk simbol tersebut dan diberi nilai *weight* = 1, kerjakan langkah ke-2.
   Jika Ya , kerjakan langkah ke-4
2. Ciptakan simpul baru untuk NYT.
3. Tempatkan simpul NYT baru dan simpul baru dibawah simpul NYT lama. (NYT lama dijadikan induk (*parent*) untuk NYT baru dan simpul baru). Kemudian kerjakan langkah ke-6

4. Tempatkan pointer ke simpul dari simbol tersebut
5. Apakah nilai *weight*+1 simpul saat ini, lebih besar dari *weight* simpul lainnya yang berada diposisi kanan dalam 1 level atau lebih besar dari *weight* simpul yang berada pada level tertinggi.
   Jika Tidak, kerjakan langkah ke-6.
   Jika Ya, kerjakan langkah ke-8
6. Tambahkan nilai weight simpul ini dengan 1.
7. Apakah simpul yang dipilih saat ini merupakan simpul akar.
   Jika Ya, prosedur Update_Pohon telah selesai.
   Jika Tidak, tempatkan penunjuk ke simpul induk, dan kerjakan langkah ke-5
8. Apakah simpul yang memiliki *weight* kecil adalah simpul induk,
   Jika Ya, kerjakan langkah ke-6.
   Jika Tidak kerjakan langkah ke-9.
9. Tukar posisi simpul saat ini dengan simpul yang memiliki nilai weight lebih kecil. Kemudian kerjakan langkah ke-6.

Dari penjelasan algoritma prosedur Update_Pohon diatas, dapat digambarkan flowchart seperti Gambar 2.



Gambar 2 Diagram Alir prosedur Update_Pohon

3. Model Penelitian

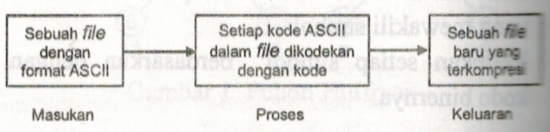
3.1 Kebutuhan Sistem

Penelitian ini memberi hasil berupa perangkat lunak untuk kompresi file dengan metode Huffman dan Adaptive Huffman. Pengembangan sistem ini akan membutuhkan perangkat pendukung yaitu perangkat keras (hardware) dan perangkat lunak (software). Perangkat keras yang digunakan untuk membuat program ini adalah komputer standar dengan prosesor Pentium 4. Sedangkan perangkat lunak yang digunakan untuk membuat sistem adalah Sistem Operasi Windows (98/Me/NT4/2000/XP), dan pengembang aplikasi Borland Delphi 6 standar tanpa tambahan komponen luar.

3.2 Sistem Enkoder

Pada perancangan sistem enkoder, masukan yang akan diproses adalah file-file yang merupakan file yang berisi dari kode-kode ASCII. Sehingga dimungkinkan semua file dengan ekstensi apapun yang ada didalam sistem operasi Windows dapat dikompresi dengan enkoder ini. File yang akan diproses oleh enkoder dalam satu kali proses kompresi adalah sebuah file, sehingga enkoder tidak dapat menangani kompresi banyak file dalam satu kali proses kompresi.

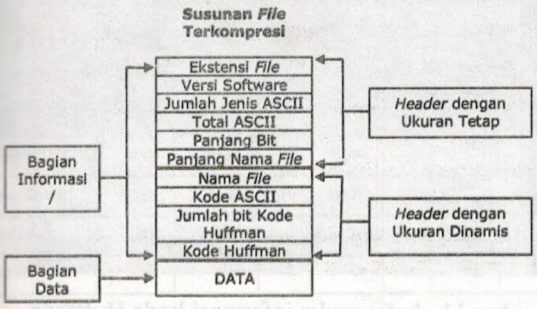
Pada proses enkoder, enkoder akan membaca setiap kode didalam file masukan kemudian memproses kode-kode ASCII tersebut dengan metode Huffman atau Adaptive Huffman. Akhir dari proses ini adalah terbentuknya kode Huffman yang akan digunakan untuk proses kompresi, proses kompresi dilakukan dengan merubah setiap kode ASCII dalam file masukan kedalam kode Huffman.



Gambar 3 Aliran proses dalam enkoder

File keluaran dari proses enkoder ini terdiri dari data dan informasi. Data berisi urutan dari (setiap kode ASCII) pada file masukan yang sudah dikodekan dengan kode Huffman. Informasi, yang sering disebut *header*, berisi informasi-informasi yang dibutuhkan proses dekoder untuk mengembalikan data pada file terkompresi menjadi file

aslinya. Adapun susunan informasi header file dalam file terkompresi adalah sebagai berikut.

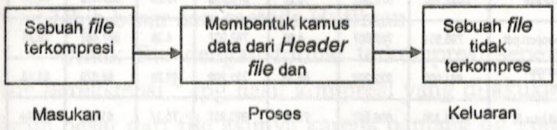


Gambar 4 Rancangan Header File Terkompresi

3.3 Sistem Dekoder.

Pada proses dekompresi, dekoder akan membaca informasi dan data yang terdapat dalam file terkompresi. Setelah menerima masukan dekoder akan mengambil informasi yang dibutuhkan untuk menterjemahkan data, apabila informasi yang dibutuhkan valid, maka dekoder akan membentuk kamus data yang berisi kode ASCII dan kode Huffman yang mewakilinya. Selanjutnya dekoder mengambil data yang terbentuk dari kode Huffman dan menterjemahkan setiap data menjadi kode ASCII.

Keluaran sistem dekoder adalah sebuah file tidak terkompres, dengan isi sama dengan file aslinya sebelum dilakukan proses kompresi.



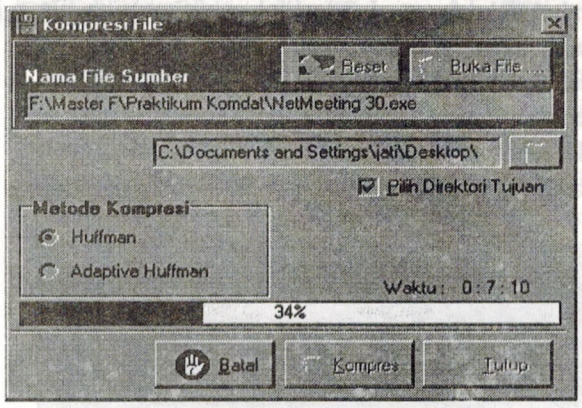
Gambar 5 Rancangan proses dekompresi

3. Hasil

3.1 Antar Muka Proses Kompresi

Seperti telah disebutkan, bahwa penelitian ini memberikan hasil berupa program aplikasi kompresi dekompresi dengan metode Huffman dan Adaptive Huffman.

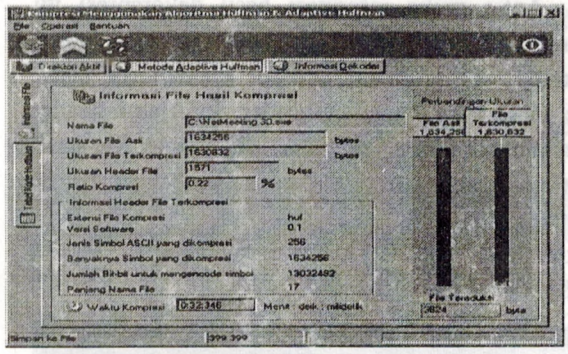
Pada antar muka untuk proses kompresi ini terdapat dua tombol utama yaitu tombol Buka File yang digunakan untuk memilih file yang akan dikompresi dan tombol Kompresi yang digunakan untuk memulai proses kompresi. Selain itu ada pilihan metode yang akan digunakan untuk proses kompresi yaitu metode Huffman dan Adaptive Huffman. Gambar 6 adalah antar muka aplikasi untuk proses kompresi.



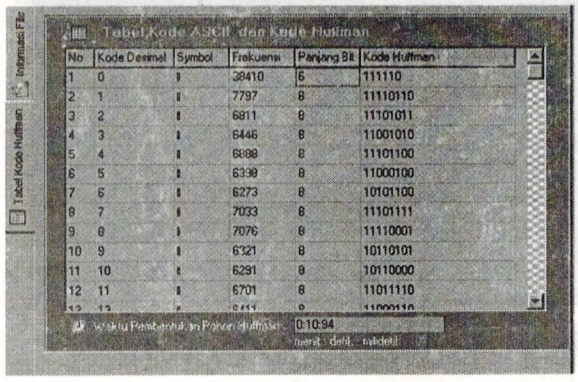
Gambar 6 Antar muka proses kompresi

3.2 Antar Muka Hasil Proses Kompresi.

Antar muka ini menampilkan informasi file yang berhasil dikompresi. Informasi utama yang ditampilkan antara lain nama file yang dikompresi, ratio kompresi, ukuran file terkompresi, ukuran header file, informasi header file. Informasi ini seperti terlihat pada Gambar 7. Selain itu, informasi kode Huffman dari setiap simbol yang dikompresi ditampilkan, contohnya seperti Gambar 8.



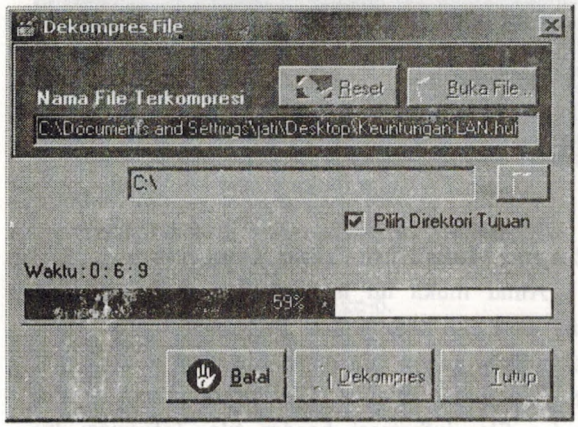
Gambar 7 Antar muka informasi hasil kompresi



Gambar 8 Antar muka informasi kode Huffman

3.3 Antar Muka Proses Dekompresi

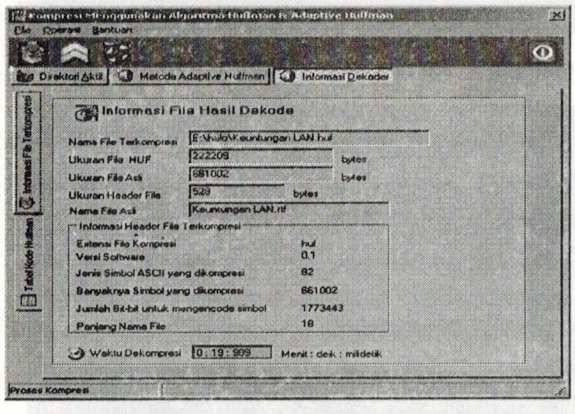
Antar muka untuk dekompresi ini terdapat tombol dekompres dan tombol untuk memilih file yang akan didekompresi. Proses dekompresi dilakukan dengan memilih file terkompresi dan menekan tombol Dekompres pada form Dekompres File. Gambar 9 adalah antar muka aplikasi untuk proses dekompresi.



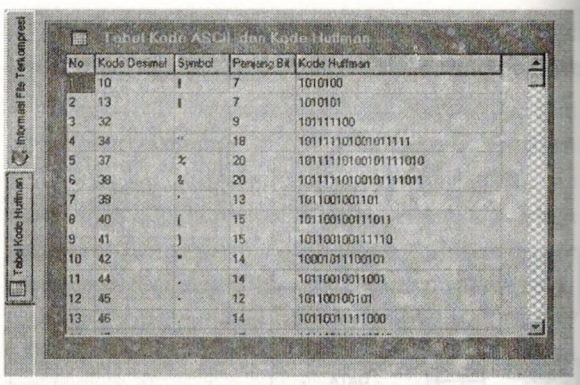
Gambar 9 Antar muka proses dekompresi

3.4 Antar Muka Hasil Proses Dekompresi

Antar muka informasi hasil dekompresi ini menampilkan informasi utama hasil proses dekompresi, informasi tersebut diantaranya nama file terkompresi, ukuran header, informasi header file, ukuran file terkompresi dan ukuran file hasil dekompresi, seperti terlihat pada Gambar 10. Selain itu juga menampilkan daftar karakter ASCII yang dikompresi beserta kode Huffman-nya, seperti terlihat pada Gambar 11.



Gambar 10 Antar muka informasi hasil dekompresi



Gambar 11 Antar muka informasi kode Huffman

3.5 Perbandingan Hasil Kompresi

Perbandingan hasil kompresi digunakan untuk melihat perbandingan ukuran file masukan dengan ukuran file hasil (file terkompresi). Tabel 1 menunjukkan perbandingan hasil kompresi menggunakan metode Huffman dan Adaptive Huffman. serta file kompresi yang dihasilkan aplikasi Winrar.

Tabel 1 Perbandingan file hasil kompresi dengan ukuran file terkompresi lebih kecil dari file asli

Nama File	Ukuran Asli (byte)	Huffman (byte)	% Rasio	Adaptive Huffman	% Rasio	Winrar (byte)	% Rasio
Explorer.exe	1.032.192	875.282	15.20	875.282	15.20	367.935	64,36
irianto-report.pdf	796.931	762.007	4.38	762.007	4.38	587.487	26,29
Keuntungan LAN.rtf	661.002	222.209	66.38	222.209	66.38	58.679	91.13
ServerLinux.rtf	2.815.420	698.957	75.17	698.957	75.17	61.826	99.99
Temp.bmp	304.182	291.881	4.04	291.881	4.04	103.908	66.05
README.txt	8.314	5.263	36.70	5.263	36.70	3.001	64.73

Dari hasil diatas dapat diambil kesimpulan bahwa file dengan format text, semakin besar ukuran file maka rasio kompresinya semakin besar, hal ini disebabkan adanya kode ASCII yang memiliki redudansi tinggi pada file tersebut sehingga susunan bit kode Huffman-nya semakin pendek.

Pada file berekstensi *.exe dan *.pdf diatas, hasil kompresi memiliki rasio kompresi kecil karena semua jenis kode ASCII dikompresi dan rata-rata panjang kode Huffman yang mewakili setiap karakter ASCII memiliki panjang 8 bit.

Tabel 2 Perbandingan file hasil kompresi dengan ukuran file terkompresi lebih besar dari file asli

Nama File	Ukuran Asli (byte)	Huffman (byte)	% Rasio	Adaptive Huffman	% Rasio	Winrar (byte)	% Rasio
Wall09.jpg	114.000	115.395	0	115.395	0	114.069	0
Default_blue.jpg	133.902	135.393	0	135.393	0	133.469	0.03
Tanya.txt	656	663	0	663	0	447	12.25
Abra.txt	12	59	0	60	0	79	0

Pada file Tanya.txt ukuran file kompresinya semakin besar disebabkan sedikitnya redundansi dari kode ASCII tertentu, sehingga susunan bit-bit kode huffman yang mewakilinya tidak terlalu pendek. Kemudian pada file Abra.txt ukuran file terkompresnya semakin besar hal ini disebabkan ukuran file terlalu kecil yaitu 12 byte, sehingga hasil kompresi kurang optimal karena ukuran header-nya jauh lebih besar dari ukuran data hasil kompresinya.

Pada file abra.txt ukuran file hasil kompresi Huffman dan Adaptive Huffman berbeda 1 byte, karena pada metode Adaptive Huffman, kode Huffman yang mewakili kode ASCII berfrekuensi kecil dimungkinkan memiliki susunan bit kode Huffman lebih panjang 1 bit, sehingga file hasil kompresi antara metode Adaptive Huffman dimungkinkan memiliki perbedaan ukuran file 1 byte lebih besar dari metode Huffman.

Untuk file dengan format terkompresi seperti file berekstensi *.jpg hasil kompresi yang dilakukan lebih besar dari file aslinya karena panjang bit kode huffman yang mewakili setiap kode ASCII rata-rata memiliki panjang 8 bit (sama dengan ukuran kode ASCII) selain itu ukuran header file yang dihasilkan besar, karena semua jenis kode ASCII dikodekan.

Tabel 3 Perbandingan waktu kompresi

		Waktu (menit : detik : milidetik)			
		Huffman		Adaptive Huffman	
Nama File	Ukuran	Pembentukan pohon	Kompresi	Pembentukan pohon	Kompresi
Explorer.exe	1.032.192	0:0:10	0:1:675	0:4:86	0:14:652
irianto-report.pdf	796.931	0:0:50	0:9:63	0:4:507	0:13:540
Keuntungan LAN.rtf	661.002	0:0:50	0:3:495	0:0:230	0:3:655
ServerLinux.rtf	2.815.420	0:0:40	0:12:358	0:0:601	0:13:89
Temp.bmp	304.182	0:0:10	0:3:415	0:1:562	0:4:927
Tanya.txt	656	0:0:0	0:0:20	0:0:0	0:0:20
README.txt	8.314	0	0:0:70	0:0:20	0:0:90
Wall09.jpg	114.000	0:0:10	0:1:362	0:0:681	0:2:13
Default_blue.jpg	133.902	0:0:20	0:1:562	0:0:852	0:2:384
Abra.txt	12	0	0:0:10	0	0:0:10

Dari hasil diatas, waktu yang dibutuhkan metode Huffman untuk melakukan proses kompresi lebih cepat dibandingkan dengan metode Adaptive Huffman. Pada metode Huffman proses kompresi lebih cepat karena bentuk pohon Huffman tidak mengalami perubahan akibat perubahan frekuensi dari karakter yang masuk, karena frekuensi dari setiap karakter sudah diketahui. Metode Adaptive Huffman lebih lambat karena pada metode Adaptive Huffman proses pembentukan pohon Huffman dilakukan saat pembacaan karakter, sehingga bentuk pohon akan berubah pada saat proses pembacaan karakter karena terjadinya perubahan frekuensi. Proses perubahan pohon inilah yang membuat metode Adaptive Huffman lebih lambat prosesnya.

3.6 Perbandingan Metode Huffman dan Adaptive Huffman

Dari aplikasi yang telah dibuat dapat disimpulkan bahwa file hasil perbandingan dari kedua metode ini yang sangat signifikan adalah waktu kompresi. Berikut ini tabel Perbandingan metode Huffman dan Adaptive Huffman.

Tabel 4 Perbandingan Kompresi Metode Huffman dan Adaptive Huffman

Perbandingan	Huffman	Adaptive Huffman
File Hasil Kompresi	Besar file sama atau lebih kecil 1 byte dari Adaptive Huffman	Besar file sama atau lebih besar 1 byte dari Huffman
Waktu Kompresi	Cepat, karena waktu proses pembentukan pohon Huffman-nya lebih cepat..	Lambat, karena waktu proses pembentukan pohon Huffman-nya lebih lambat.

4. Kesimpulan

Berdasarkan hasil penelitian di atas, maka kesimpulan yang dapat ditarik adalah:

1. Sistem dapat mengkompresi semua file dengan ekstensi apapun.
2. Header file mampu menampung kode Huffman yang dihasilkan proses kompresi sampai 32 bit.
3. File hasil kompresi metode Huffman mempunyai ukuran sama atau lebih kecil 1 byte dari metode Adaptive Huffman

4. Waktu kompresi dengan metode Huffman lebih cepat dibanding metode Adaptive Huffman, karena waktu proses pembentukan pohon Huffman-nya lebih cepat..

Daftar Pustaka

[1] David Salomon, 2000, Data Compression: the complete reference, Springer-Verlag New York

[2] Heri Sujaini dan Yessi Mulyani, 2000, Algoritma Run-Length, Half-Byte dan Huffman untuk Pemampatan File

[3] http://data.uta.edu/~ramesh/book/MultimediaSystems/pdfs/M2L2.pdf, Lossless Compression

[4] http://www.binaryessence.com/dct, Binary Essence, Adaptive Huffman Code

[5] http://www.cs.cf.ac.uk/Dave/Multimedia/node212.html, Adaptive Huffman Coding

[6] http://www.cs.sfu.ca/CC/365/li/squeeze/AdaptiveHuff.html, Adaptive Huffman Compression

[7] http://www.dcl.kcl.ac.uk/, Dracopoulos., Dimitris C., Compression Method for Multimedia lecture 2 Huffman Coding,

[8] http://www.xcf.berkeley.edu/~ali/K0D/Algorithms/huff/, Adaptive Huffman Encoding

[CV Penulis]

Sudarmanto, menyelesaikan studi S1 pada tahun 1989 jurusan Teknik Elektro di Universitas Gadjah Mada. Melanjutkan program S2 jurusan Teknik Elektro di Universitas Gadjah Mada selesai pada tahun 1998. Saat ini bekerja sebagai staf pengajar di STMIK AKAKOM Yogyakarta.