Perangkat Lunak Dictionary Based Compression Memanfaatkan Transformasi Burrows-Wheeler Dan Byte Pair Encoding

Benny[1], Djonny[2]
STMIK IBBI
Jl. Sei Deli No. 18 Medan, Telp. 061-4567111 Fax. 061-4527548
Email : freebeeshen@yahoo.com, djonny_joe@gmail.com

Abstrak

File merupakan data digital yang berisi informasi-informasi. Ukuran *file* yang terlalu besar akan menjadi masalah bila *file* tersebut akan ditransfer atau dipertukarkan. Untuk itu dibutuhkan cara tertentu untuk memperkecil ukuran *file*.Salah satu caranya untuk masalah di atas adalah *file* tersebut dipadatkan melalui proses kompresi sehingga ukurannya menjadi lebih kecil dari ukuran semula dan mempersingkat waktu ketika ditransmisi. Salah satu metode yang dapat dipakai untuk mengkompresi ukuran *file* adalah dengan metode *Dictionary Based Compression* (DBC) yang bekerja dengan memanfaatkan penggantian string yang terdapat dalam kamus dimana sebelum diproses akan dilakukan transformasi Burrows-Wheeler dan teknik *encoding byte pair* dimana teknik pengkodean pada byte pair menggantikan dua pasangan karakter dengan satu karakter tunggal, sehingga dengan teknik tersebut dapat menghemat kapasitas penyimpanan data. Hasil dari tulisan ini adalah suatu perangkat lunak yang dapat melakukan kompresi dan dekompresi balik berdasarkan pada algoritma DBC pada semua jenis *file* yang belum dikompresi. Program juga dapat menampilkan besarnya rasio kompresi, rasio dekompresi, dan lama proses yang telah dikerjakan pada *file*. Hasil pengujian dengan metode kompresi DBC juga dapat menampilkan rentang rasio kompresi.

Kata kunci : *File*, *Dictionary Based Compression* (DBC), teknik *encoding byte pair*, kompresi

Abstract

A digital data file that contains the information. File size that is too large will be a problem if the files are to be transferred or exchanged. That requires a certain way to reduce the file size.
One way for the above problem is the file is compressed by the compression process so that its size becomes smaller than the original size and shorten the time when it is transmitted. One method that can be used to compress the file size is by using Dictionary Based Compression (DBC), which works by using replacement strings in the dictionary that will be processed before the Burrows-Wheeler transformation and byte pair encoding technique in which the byte pair encoding technique replaces two pair of characters with a single character, so that the technique can save the data storage capacity.The results of this paper is a software that can compress and decompress back based on DBC algorithm on all types of files are not compressed. The program can also display the amount of compression ratio, compression ratio, and a long process that has been done on the file. Compression test results with the DBC method can also show the range of compression ratios.

Keywords : data file, Dictionary Based Compression (DBC), byte pair encoding technique, compression

1. Pendahuluan

Kompresi data atau dikenal juga sebagai pemampatan data adalah suatu teknik mengubah data menjadi bentuk data lain dimana data tersebut diubah menjadi simbol yang lebih sederhana. Kompresi data mempunyai tujuan memperkecil ukuran data sehingga selain dapat menghemat media penyimpanan, juga dapat memudahkan transfer data. Sebuah berkas yang sudah dikompres, akan mengurangi *space* untuk penyimpanan dan dapat dikirim ke klien lebih cepat. Kompresi sangat berguna pada saat mengirimkan sebuah isi berkas melalui koneksi jaringan. Di dalam konteks pembahasan mengenai kompresi, rasio kompresi merujuk kepada perbandingan antara ukuran *file* hasil dengan *file* yang dikompresi.

Kompresi data merupakan suatu upaya untuk mengurangi jumlah bit yang digunakan untuk menyimpan atau mentransmisikan data. Kompresi data meliputi berbagai teknik kompresi yang diterapkan dalam bentuk perangkat lunak *(software)* maupun perangkat keras *(hardware).*[2] Bila ditinjau dari sisi penggunaannya, kompresi data bisa bersifat umum untuk segala keperluan atau bersifat khusus untuk keperluan tertentu. Keuntungan data yang terkompresi antara lain: mengurangi *bottleneck* pada proses I/O dan transmisi data, penyimpanan data lebih hemat ruang, mempersulit pembacaan data oleh

pihak yang tidak berkepentingan, dan memudahkan distribusi data dengan media *removable* seperti *flash* disk, CD, DVD, dll.

Algoritma kompresi diklasifikasikan menjadi dua jenis, yaitu algoritma kompresi *lossy* dan algoritma kompresi *lossless*.[3] Contoh dari algoritma kompresi *lossy* adalah algoritma JPEG dan algoritma MPEG, sedangkan contoh dari algoritma kompresi *lossless* adalah algoritma *Huffman*, *Shannon Fano*, RLE, LZ77, LZ78, LZW, dan LZSS. Kompresi menggunakan *lossy* mengeliminasikan beberapa data dari suatu berkas. Kompresi menggunakan *lossless* menjamin bahwa berkas yang dikompresi dapat selalu dikembalikan ke bentuk aslinya. Salah satu metode kompresi yang bersifat *lossless* yang dapat dipakai untuk melakukan kompresi terhadap teks adalah *dictionary based compression* yang bersifat statis dimana data yang akan dikompres adalah data teks. Untuk melakukan kompresi dibutuhkan sebuah kamus yang memetakan symbol pada data dengan kode yang akan digunakan. Pada kompresi berbasis kamus statik, cara pembangunan kamus, struktur kamus, dan *memory* yang digunakan, sangat mempengaruhi performansi hasil kompresi. Contoh : cara pemberian indeks pada kamus akan mempengaruhi kecepatan akses kamus, yang akan mempengaruhi kecepatan pencarian data yang dibutuhkan. Metode ini mempunyai keunggulan pada rasio kompresi jika pada susunan kamus yang digunakan mampu menggantikan suatu *string* yang panjang hanya dengan satu karakter tertentu. Tetapi untuk kasus seperti ini metode ini hanya dapat diterapkan jika data yang akan diproses hanya berapa data teks. Karena saat ini data sudah banyak disimpan dalam format biner, maka metode ini susah diterapkan. Tetapi dengan melakukan modifikasi dengan memanfaatkan transformasi *Burrows-Wheeler* dan teknik *encoding byte pair* dimana teknik pengkodean pada *byte pair* menggantikan dua pasangan karakter dengan satu karakter tunggal, sehingga dengan teknik tersebut dapat menghemat kapasitas penyimpanan data. Adapun alasan utama dipilihanya *Burrows-Wheeler* adalah kebanyakan pola string dapat disusun menjadi bentuk yang berulang sehingga perulangan tersebut dapat dimanfaatkan untuk di-*encode* dengan menggunakan *byte pair encoding* yang mengkodekan dua pasangan karakter yang sama persis. Dengan kedua teknik tersebut dapat membuat rasio kompresi pada *file* menjadi lebih besar sehingga ukuran *file* menjadi lebih kecil.

2. Literatur

a. Pengantar Kompresi Data

Kompresi Data adalah salah satu subyek di bidang teknologi informasi yang saat ini telah diterapkan secara luas. Gambar-gambar yang diperoleh di berbagai situs internet pada umumnya merupakan hasil kompresi ke dalam format GIF atau JPEG. *File video* MPEG adalah hasil proses kompresi pula. Penyimpanan data berukuran besar pada *server* pun sering dilakukan melalui kompresi.

Kompresi Data merupakan cabang ilmu komputer yang bersumber dari Teori Informasi. Teori Informasi sendiri adalah salah satu cabang Matematika yang berkembang sekitar akhir dekade 1940-an. Tokoh utama dari Teori Informasi adalah Claude Shannon dari Bell Laboratory. Teori Informasi mengfokuskan pada berbagai metode tentang informasi termasuk penyimpanan dan pemrosesan pesan. Teori Informasi mempelajari pula tentang *redundancy* (informasi tak berguna) pada pesan. (Widhiartha, 2003: 2)

Semakin banyak *redundancy* semakin besar pula ukuran pesan, upaya mengurangi *redundancy* inilah yang akhirnya melahirkan subyek ilmu tentang Kompresi Data. Teori Informasi menggunakan terminologi *entropy* sebagai pengukur berapa banyak informasi yang dapat diambil dari sebuah pesan. Kata "*entropy*" berasal dari ilmu termodinamika. Semakin tinggi *entropy* dari sebuah pesan semakin banyak informasi yang terdapat di dalamnya. *Entropy* dari sebuah simbol didefinisikan sebagai nilai logaritma negatif dari probabilitas kemunculannya. Untuk menentukan konten informasi dari sebuah pesan dalam jumlah bit dapat digunakan rumus sebagai berikut:

$$number\ of\ bits = -log\ base\ 2\ (probability)$$

Entropy dari keseluruhan pesan adalah jumlah dari keseluruhan *entropy* dari seluruh *symbol*.

Teknik Kompresi Data dapat dibagi menjadi dua kategori besar, yaitu: (Widhiartha, 2003: 3)

1. *Lossy Compression*

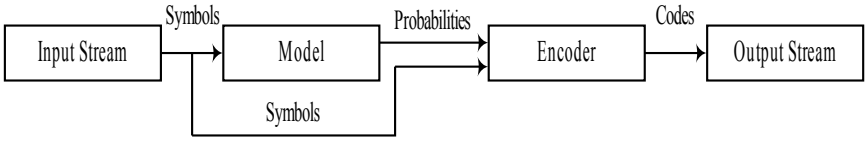
Lossy compression menyebabkan adanya perubahan data dibandingkan sebelum dilakukan proses kompresi. Sebagai gantinya *lossy compression* memberikan derajat kompresi lebih tinggi. Tipe ini cocok untuk kompresi *file* suara digital dan gambar digital. *File* suara dan gambar secara alamiah masih bisa digunakan walaupun tidak berada pada kondisi yang sama sebelum dilakukan kompresi.

2. *Lossless Compression*

Sebaliknya *Lossless Compression* memiliki derajat kompresi yang lebih rendah tetapi dengan akurasi data yang terjaga antara sebelum dan sesudah proses kompresi. Kompresi ini cocok untuk basis data, dokumen atau *spreadsheet*. Pada *lossless compression* ini tidak diijinkan ada bit yang hilang dari data pada proses kompresi.

Secara umum kompresi data terdiri dari dua kegiatan besar, yaitu *Modeling* dan *Coding*. Proses dasar dari kompresi data adalah menentukan serangkaian bagian dari data (*stream of symbols*) mengubahnya menjadi kode (*stream of codes*). Jika proses kompresi efektif maka hasil dari *stream of codes* akan lebih kecil dari segi ukuran daripada *stream of symbols*. Keputusan untuk menggantikan *symbols* tertentu dengan *codes* tertentu adalah inti dari proses modeling. Secara umum dapat diartikan bahwa sebuah model adalah kumpulan data dan aturan yang menentukan pasangan antara *symbol* sebagai *input* dan *code* sebagai *output* dari proses kompresi. Sedangkan *coding* adalah proses untuk menerapkan *modeling* tersebut menjadi sebuah proses kompresi data. (Widhiartha, 2003: 3)

Dengan menggunakan *Huffman coding* sebagai contoh, sebuah proses kompresi akan nampak seperti gambar 2.1 berikut ini:



Gambar 2.1 Diagram Statistical Model dengan *Huffman Code*

Pada Huffman *coding*, *symbols* yang sering muncul akan diubah menjadi *code* dengan jumlah bit kecil, sedangkan yang jarang muncul akan menjadi *code* dengan jumlah bit besar.

A. *Coding*

Melakukan proses *encoding* dengan menggunakan ASCII atau EBDIC yang merupakan standar dalam proses komputasi memberikan kelemahan mendasar apabila dilihat dari paradigma kompresi data. ASCII dan EBDIC menggunakan jumlah bit yang sama untuk setiap karakter, hal ini menyebabkan banyak bit yang "terbuang" untuk merepresentasikan karakter-karakter yang sebenarnya jarang muncul pada sebuah pesan.

B. *Modelling*

Jika *coding* adalah roda dari sebuah mobil maka modeling adalah mesinnya. Sebaik apapun algoritma untuk melakukan *coding* tanpa model yang baik kompresi data tidak akan pernah terwujud. Kompresi *Data Lossless* pada umumnya diimplementasikan menggunakan salah satu dari dua tipe *modeling*, yaitu *statistical* atau *dictionary-based*. Pada *Statistical-modeling*, proses kompresi menggunakan probabilitas kemunculan dari sebuah simbol sedangkan pada *dictionary-based* menggunakan kode-kode untuk menggantikan sekumpulan simbol yang telah ada dengan simbol yang lebih sederhana.

Statistical Modeling

Pada bentuk paling sederhananya, statistical-modeling menggunakan tabel statis yang berisi probabilitas kemunculan suatu karakter atau *symbol*. Tabel ini pada awalnya bersifat universal, sebagai contoh pada bahasa Inggris karakter yang paling sering muncul adalah huruf "e" maka karakter ini memiliki probabilitas tertinggi pada *file* teks yang berbahasa Inggris.

Menggunakan tabel universal pada akhirnya tidak memuaskan para ahli kompresi data karena apabila terjadi perubahan pada subyek yang dikompresi dan tidak sesuai dengan tabel universal maka akan terjadi penurunan rasio kompresi secara signifikan.

Akhirnya muncul *modeling* dengan menggunakan tabel yang adaptif, di mana tabel tidak lagi bersifat statis tetapi bisa berubah sesuai dengan kode. Pada prinsipnya dengan model ini, sistem melakukan penghitungan atau *scan* pada keseluruhan data setelah itu barulah membangun tabel probabilitas kemunculan dari tiap karakter atau *symbol*.

Model ini kemudian dikembangkan lagi menjadi *adaptive statistical modeling* di mana sistem tidak perlu melakukan *scan* ke seluruh *symbol* untuk membangun tabel statistik, tetapi secara adaptif melakukan perubahan tabel pada proses *scan* karakter per karakter.

Dictionary Based Modeling

Jika *statistical model* pada umumnya melakukan proses encode simbol satu per satu mengikuti siklus: baca karakter → hitung probabilitas → buat kodenya maka *dictionary-based modeling* menggunakan mekanisme yang berbeda. *Dictionary-based modeling* membaca *input* data dan membandingkannya dengan isi *dictionary*. Jika sekumpulan string sesuai dengan isi *dictionary* maka indeks dari *dictionary entry* lah yang dikeluarkan sebagai *output* dan bukan kodenya.

Sebagai perumpamaan dari *dictionary-based* dapat digunakan makalah ilmiah sebagai contoh. Saat kita membaca makalah ilmiah kita sering membaca nomor-nomor referensi yang bisa kita cocokkan dengan daftar pustaka di belakang. Hal ini mirip dengan proses pada *dictionary-based modeling*.

b. Dictionary Based Compression

Dictionary based compression merupakan suatu algoritma kompresi yang dilakukan berdasarkan atas pengkodean suatu *string* dengan karakter tertentu yang berukuran lebih sedikit. Dengan cara seperti ini misalkan suatu *string* karakter yang terdiri atas 10 huruf jika direpresentasikan menjadi 2 huruf maka telah menghemat ruang penyimpanan hingga 8 huruf.

Keluarga algoritma ini tidak melakukan *encoding* terhadap satu simbol tunggal sebagai *bit stream*, tetapi melakukan *encoding* terhadap frase dari *variable length* sebagai satu *token* tunggal (*single token*).

Sebagai contoh jika menggunakan *Random House Dictionary of the English Language*, 2nd Edition, Unabridged. Suatu kalimat dalam bahasa Inggris seperti:

'A good example of how dictionary based compression'

Dapat dikodekan menjadi:

1/1 822/3 674/4 1343/60 928/75 550/32 173/46 421/2

Pada contoh di atas huruf A dinyatakan terletak pada halaman 1 dan baris 1 sedangkan kata *good* terletak pada hal 822 pada baris 3 dan seterusnya. Dimana hasil di atas nilai pertama menjadi nomor halaman dan nilai selanjutnya merupakan entri kata dalam halaman kamus tersebut.

c. Transformasi *Burrows-Wheeler Transform* (BWT)

Burrows-Wheeler Transform atau dikenal juga sebagai *block-sorting compression* merupakan algoritma yang digunakan dalam teknik kompresi data seperti Bzip2. Metode ini ditemukan oleh Michael Burrows dan David Wheeler pada tahun 1994 saat bekerja pada *DEC Systems Research Center* di Palo Alto, California. Metode ini didasarkan atas transformasi yang tidak dipublikasikan dan ditemukan oleh Wheeler pada tahun 1983.

Michael Burrows dan David Wheeler merilis suatu laporan riset pada tahun 1994 yang membahas mengenai hasil kerja yang telah mereka lakukan pada Digital Systems Research Center di Palo Alto, California dengan judul paper "*A Block Sorting Lossless Data Compression Algorithm*" menyajikan algoritma kompresi data berdasarkan atas transformasi yang tidak dipublikasikan tersebut.

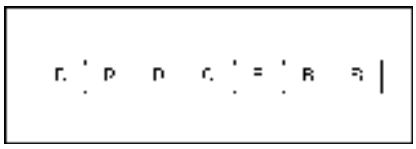
Ketika suatu karakter *string* ditransformasikan dengan BWT, tidak ada dari nilai karakter tersebut yang berubah. Transformasi hanya melakukan permutasi atas urutan-urutan karakter. Jika *string* asli mempunyai beberapa *substring* yang sering muncul, maka *string* yang ditransformasikan tersebut akan menempatkan beberapa tempat dimana suatu karakter *string* yang sering muncul tersebut atau berulang akan disusun dalam satu posisi yang berdekatan. Ini berguna dalam tahapan kompresi, karena susunan tersebut mempunyai kecenderungan lebih mudah dikompresi karena telah mempunyai data yang berulang, sebagai contoh:

Input: `SIX.MIXED.PIXIES.SIFT.SIXTY.PIXIE.DUST.BOXES`

Output: `TEXYDST.E.IXIXIXXSSMPPS.B..E.S.EUSFXDIIOIIIT`

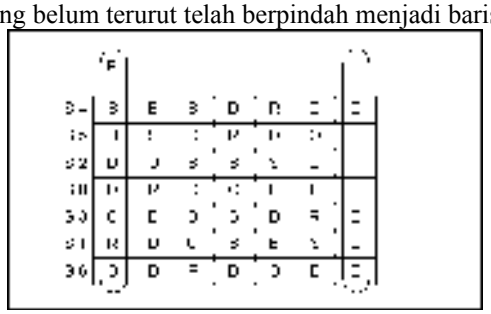
BWT merupakan algoritma yang memproses satu blok data dan kemudian mengurutkannya dengan menggunakan algoritma pengurutan. Blok hasil *output* mempunyai jumlah karakter yang sama, perbedaannya adalah uruatan atau susunan karakternya. Transformasi ini bersifat *reversible*, berarti urutan asli dari elemen data dapat ditempatkan kembali tanpa adanya kehilangan data. BWT dilakukan pada keseluruhan blok data sekaligus. Kebanyakan algoritma kompresi *lossless* beroperasi pada *streaming mode*, dimana pembacaan satu *byte* data atau beberapa *byte* pada satu saat. Karena BWT beroperasi pada data di dalam memori, maka akan terjadi kesulitan dalam memproses *file* yang berukuran besar. Dalam kasus seperti ini, *file* harus dibagi dan diproses per blok pada satu saat.

Sebagai ilustrasi atas bagaimana proses transformasi dengan BWT maka diberikan contoh data set sebagai sampel seperti terlihat pada Gambar 2.2. Seperti diketahui string tersebut terdiri atas 7 byte data. Untuk melakukan BWT, hal pertama yang harus dilakukan adalah menyatakan ukuran blok tersebut sebagai *string S* dengan panjang *N* dan *buffer* sebagai karakter terakhir yang ditempatkan dari belakang ke depan.



Gambar 2.2 Contoh *Data Set*

Merupakan hal yang penting untuk diingat bahwa pada titik ini kita tidak melakukan penyalinan rotasi N – 1 dari *string* yang di-*input*. Langkah selanjutnya adalah melakukan suatu *lexicographical sort* terhadap kumpulan karakter dalam *string* tersebut. Setelah dilakukan pengurutan, maka kumpulan dari *string* diatur seperti terlihat pada Gambar 2.3. Terdapat dua hal penting yang perlu dilihat pada gambar tersebut. Pertama, *string* telah diurutkan, dimana posisi setiap karakter telah berubah, sehingga *String* 0 yang merupakan *string* asli yang belum terurut telah berpindah menjadi baris 4 dalam *array*.



Gambar 2.3 Kumpulan Karakter dalam *String* Setelah Pengurutan

Hal kedua adalah jika kita menandai kolom pertama dan terakhir dalam matriks dengan simbol khusus F dan L, masing-masing untuk kolom pertama dan terakhir dari *array*. Kolom F terdiri atas semua karakter dalam *string* asli dalam urutan yang terurut. Sehingga *string* asli ”DRDOBBS” direpresentasikan dalam F sebagai ”BBDDORS”.

Karakter-karakter pada kolom L tidak muncul dalam urutan yang khusus, tetapi pada kenyataannya kolom ini mempunyai properti yang menarik. Setiap dari karakter dalam kolom L adalah *prefix character* pada *string* yang muncul dalam baris yang sama dalam kolom F.

Hasil akhir dari transformasi *string* pada BWT terdiri atas dua bagian: bagian pertama adalah salinan dari kolom L dan *primary index* yaitu suatu integer yang mengindikasikan dimana baris terdiri atas karakter asli pertama dari *buffer* B. Sehingga jika melakukan BWT pada contoh *string* di atas akan menghasilkan *output string* L dimana terdiri atas "OBRSDDB" dan *primary index* 5. Sedangkan bagian kedua adalah bagain hasil yang ditunjukkan oleh *string* F.
Integer 5 ini dapat ditemukan secara mudah karena karakter asli pertama dari *buffer* selalu ditemukan pada kolom L pada baris yang berisi S1. Karena S1 secara sederhana S0 dirotasi ke kiri dengan satu posisi karakter tunggal, karakter yang paling pertama dari *buffer* dirotasi ke kolom terakhir dari matriks. Oleh karena itu penempatan S1 sama halnya dengan menempatkan posisi karakter pertama dari *buffer* dalam L.

d. Byte Pair Encoding

Pada bagian ini akan dijelaskan mengenai *byte pair encoding* yang merupakan suatu algoritma *encoding* tetapi mempunyai kemampuan untuk melakukan kompresi data secara sederahana. *Byte pair encoding* merupakan suatu bentuk sederhana dari kompresi data dimana memanfaatkan pasangan karakter yang paling sering muncul dari byte data yang mana digantikan sebagai satu byte data yang berupa simbol atau karakter tertentu. Suatu tabel penggantian karakter tersebut perlu untuk dibentuk agar data asli dapat dibentuk kembali.

Sebagai contoh pada bagian ini diberikan contoh bagaimana *byte pair encoding* bekerja. Misalkan terdapat data yang akan dikodekan sebagai berikut:

aaabaaabac

Pasangan byte "aa" paling sering muncul, sehingga akan digantikan dengan satu byte yang tidak digunakan dalam data misalnya dengan karakter "Z". Sehingga setelah dilakukan penggantian dengan suatu tabel: Z ← aa didapatkan data menjadi:

ZabZabac

Dalam hal ini byte data "Za" juga paling sering muncul, sehingga dapat digantikan dengan suatu byte data yang juga tidak dipakai, misalnya "Y" (dalam kasus seperti ini "Za" tidak dapat digantikan dengan "Z", karena setiap kemunculan dari "Z" akan diganti dengan "aa"). Penggantian ini dengan menggunakan tabel: Z ← aa dan Y ← Za. Sehingga diperoleh hasil:

YbYbac

Sekali lagi jika dilakukan penggantian atas pasangan byte yang paling sering muncul dengan menggunakan tabel: Z ← aa Y ← Za X ← Yb maka diperoleh hasil:

XXac

Data terakhir ini tidak dapat dikompresi lagi dengan byte *pair encoding* karena tidak terdapat pasangan byte yang paling sering muncul. Untuk melakukan proses dekompresi data, maka secara sederhana melakukan penggantian dengan urutan terbalik seperti berikut:
XXac menjadi YbYbac
YbYbac menjadi ZabZabac
ZabZabac menjadi aaabaaabac

e. Visual Basic

Microsoft Visual Basic adalah bahasa pemrograman yang bekerja dalam lingkup *Microsoft Windows*. *Microsoft Visual Basic* bekerja dengan menggunakan objek-objek sebagai komponen pemrogramannya. Setiap objek digambarkan pada layar dan melakukan pengaturan *property* terhadap objek yang digambarkan. Beberapa kemampuan atau manfaat dari *Visual Basic* di antaranya seperti:

1) Untuk membuat program aplikasi berbasis *Windows*.
2) Untuk membuat objek-objek pembantu program seperti *Kontrol ActiveX*, *File Help*, aplikasi internet, dan pembuatan sistem pakar.
3) Menguji program (*debugging*) dan menghasilkan program akhir berakhiran EXE yang bersifat *executable*, atau dapat digunakan. (http://dspace. widyatama. ac. Id / bitstream/ handle/ 10364/ 938/bab2.pdf?sequence=4)

Berikut ini merupakan langkah-langkah pembuatan program dengan *Microsoft Visual Basic For Windows* adalah:

1) Membuat tampilan antarmuka program. Pada tahap awal, tampilan tidak perlu dibuat sempurna, karena tampilan ini dapat diubah-ubah pada tahap akhir pembuatan program.
2) Membuat kode program. Kode *Visual Basic* akan diletakkan pada kontrol atau *form* yang akan menggunakan kode tersebut. Kode menjadi milik sebuah kontrol atau *form* akan dijalankan jika terdapat kejadian terhadap kontrol atau *form* tersebut.
3) Mengkompilasikan program. Tahap akhir dari pembuatan program adalah mengkompilasikan program sehingga menjadi program yang berdiri sendiri dan dapat dijalankan dalam lingkungan *Windows*.

3. Pemodelan Sistem

Terdapat banyak bentuk model yang dapat digunakan dalam perancangan sistem antara lain model narasi, model *prototype*, model grafis dan lain-lain. Perangkat yang digunakan untuk memodelkan suatu sistem diantaranya adalah: (Ladjamudin, 2006: 165)

a. Context Diagram
b. Data Flow Diagram
c. Kamus Data
d. Spesifikasi Proses

3.1 Langkah Perancangan Program Kompresi

Terdapat empat tahapan yang dipergunakan oleh peneliti berdasarkan atas *analisis* di atas untuk mengembangkan program kompresi ini yaitu:

a) *Planning* dimana pada tahapan ini peneliti melakukan perencanan untuk menentukan algoritma yang dipakai yaitu algoritma *Burrows Wheeler* dan *Byte Pair Encoding*. Proses *input* mencakup semua jenis *file* dimana hasil *output* mempunyai format dan ekstensi sendiri sehingga dapat dibedakan dengan format *file* aplikasi yang lain.

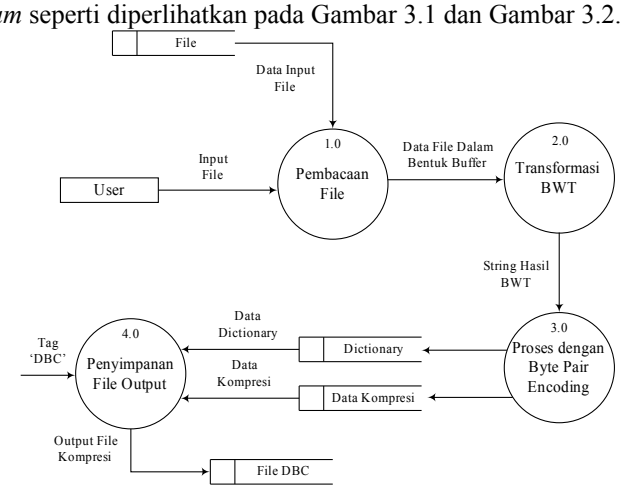
b) *Prototyping* dimana peneliti melakukan penyusunan dan pengelompokkan komponen-komponen *visual* berupa objek dari bahasa pemrograman *Visual Basic 6.0* yang digunakan untuk mengembangkan program ini.

c) *Analysis* dimana peneliti melakukan analisis berupa tata letak dan dan penentuan setiap fungsi yang setiap objek yang terdapat pada rancangan program.

d) *Design* dimana hasil rancangan akhir akan dilakukan penelitian kode program sehingga aplikasi dan *user interface* yang dirancang dapat digunakan sebagai program kompresi alternatif.

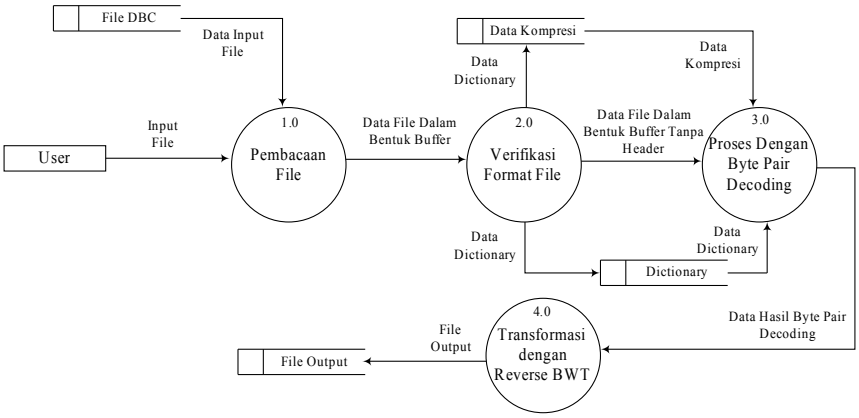
3.2 Analisis Proses

Untuk keperluan analisis proses dari sistem yang dirancang, maka dapat digambarkan dalam bentuk *data flow diagram* seperti diperlihatkan pada Gambar 3.1 dan Gambar 3.2.



Gambar 3.1 Analisis Proses Kompresi

Pada analisis proses kompresi tersebut dimulai dari *user* menentukan *file* yang akan diproses dimana pada proses pertama dilakukan pembacaan isi *file* dimana hasilnya dinyatakan dalam bentuk *array buffer*. Data dalam bentuk array *buffer* tersebut kemudian ditransformasikan dengan menggunakan BWT (*Burrows-Wheeler Transform*) dimana hasil dari transformasi BWT (*Burrows-Wheeler Transform*)tersebut akan diproses lagi dengan *byte pair encoding*. Pada tahapan ini data *output* terdiri atas dua bagian yaitu bagian data kompresi dan bagian data yang berupa *dictionary* (kamus) yang berisikan pasangan byte yang di-*encoding*. Setelah tahapan ini selesai maka dilakukan proses penyimpan *file output* dimana digabungkan *output* antara Tag 'DBC' (*Dictionary Based Compression*) dengan data *dictionary* dan data hasil kompresi.



Gambar 3.2 Analisis Proses Dekompresi

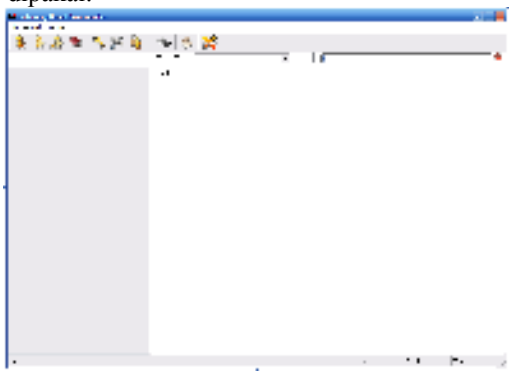
Kebalikan dari proses kompresi merupakan proses dekompresi dimana pada langkah pertama *user* menentukan *file* yang akan diproses dimana dalam hal ini yang mempunyai ekstensi DBC. Setelah itu dilakukan pembacaan *file* dan hasilnya disimpan dalam bentuk *array buffer*. Selanjutnya data dalam bentuk *array buffer* ini dipisahkan dalam bentuk data *dictionary* dan data kompresi yang akan digunakan dalam proses *decoding* dengan *byte pair*. Proses *decoding* dilakukan pada data kompresi berdasarkan pasangan data *dictionary* yang dibentuk pada proses kompresi. Hasil dari proses *decoding* kemudian diproses dengan transformasi *reverse* BWT hasilnya berupa *file output*.

4. Perancangan *Prototype*

Antarmuka merupakan suatu media interaksi (interaktif) antara komputer dengan pemakai (*user*). Pada sistem operasi yang berbasis grafis (*graphic user interface* atau GUI) seperti *Windows*, antarmuka dari suatu perangkat lunak biasanya berupa jendela (*window*). Melalui jendela ini pemakai dapat berinteraksi dengan perangkat lunak yang digunakannya.

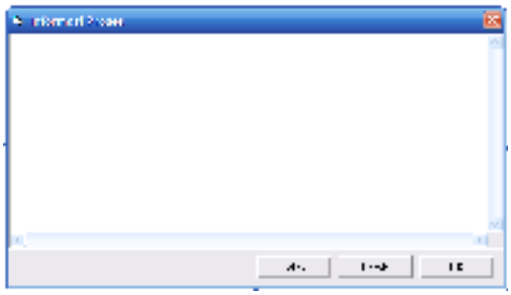
Perancangan program ini meliputi penempatan dan penyusunan objek-objek yang terdapat dalam *form Visual Basic*. *Form* yang dirancang terlebih dahulu dibuat rancangan dasarnya agar memudahkan sewaktu membuat rancangan di dalam *Visual Basic*. Pada bagian perancangan ini akan dijelaskan tentang perancangan *form* untuk program kompresi dan dekompresi dengan algoritma transformasi *Burrows Wheeler* dan *Byte Pair Encoding*. *Form* yang dirancang hanya terdiri atas dua *form* saja yaitu *form* utama yang merupakan bagian tampilan utama dari program dan *form about* yang berisi keterangan program dan keterangan serta nama peneliti.

Gambar 4.1 berikut ini merupakan perancangan dari *form* utama program dan *form about* beserta dengan komponen *Visual Basic* yang dipakai.

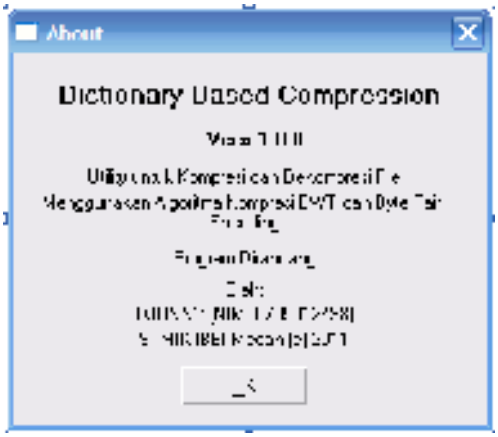


Gambar 4.1 Rancangan Tampilan Utama Program

Untuk menampilkan informasi mengenai proses yang berlangsung, maka dibentuk suatu *window* yang dapat menampilkan *file* logistik (.LOG) berupa informasi untuk setiap *file* yang diproses. Rancangan ini menggunakan *window* dengan objek *rich text box* seperti terlihat pada Gambar 4.2 dan Gambar 4.3.

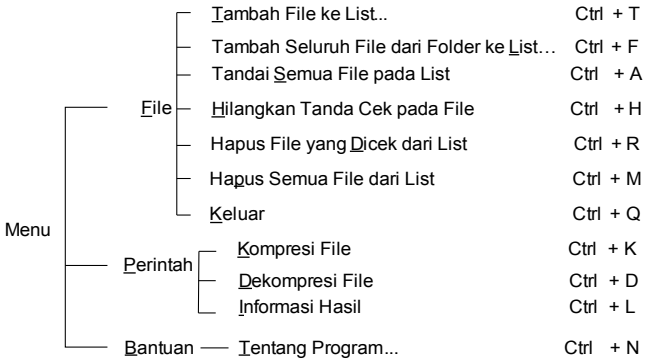


Gambar 4.2 Rancangan Tampilan Informasi Hasil Proses



Gambar 4.3 Rancangan Tampilan *Form About*

Pada *form* utama terdapat *menu* yang mempunyai fungsi-fungsi yang sama dengan *command button*. Adapun struktur menu dari program ini ditunjukkan pada Gambar 4.4.



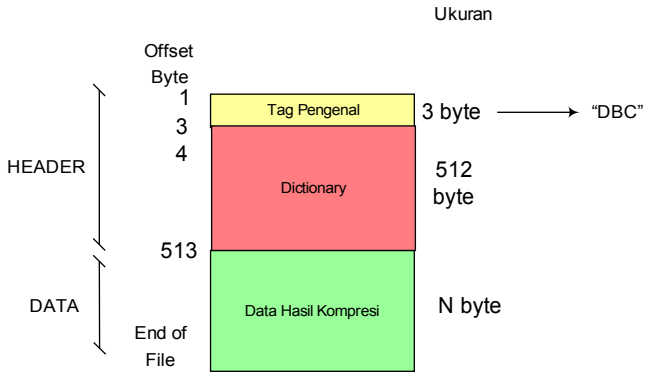
Gambar 4.4 Struktur Menu Program

4.1 Ekstensi *File* Hasil Kompresi

Untuk membedakan *file* hasil kompresi dengan *file* asli maka program akan menambahkan ekstensi tambahan yaitu ".DBC". Bila *file* tersebut didekompresi kembali maka ekstensi tambahan ini akan otomatis dibuang.

4.1.1 Perancangan Struktur *File Output*

Hasil kompresi pada *file* akan mempunyai ekstensi ".DBC" dan dibentuk dengan struktur yang sederhana. Terdiri atas dua bagian yaitu bagian "*Header*" dan bagian "Data". Bagian "*Header*" merupakan bagian awal data yang berisi in*form*asi mengenai kode pengenal. Bagian ini mempunyai ukuran maksimum 4 *byte* sisanya 512 *byte* merupakan bagian yang dicadangkan untuk *dictionary*. "Header" merupakan bagian yang penting untuk proses dekompresi Sedangkan bagian "Data" merupakan data hasil kompresi atas *file* berdasarkan algoritma *Burrows-Wheeler* dan *Byte Pair Encoding* seperti ditunjukkan pada Gambar 4.5.



Gambar 4.5 Struktur *File Output*

Pada bagian DATA terdiri data dari suatu *file* yang dikompresi. Implementasi program kompresi dan dekompresi ini nantinya menggunakan panjang *dictionary* sebesar 512 *byte*.

4.2 Implementasi Sistem

Implementasi sistem program ini mencakup spesifikasi kebutuhan perangkat keras (*hardware*) dan spesifikasi perangkat lunak (*software*).

4.2.1 Spesifikasi Perangkat Keras dan Perangkat Lunak

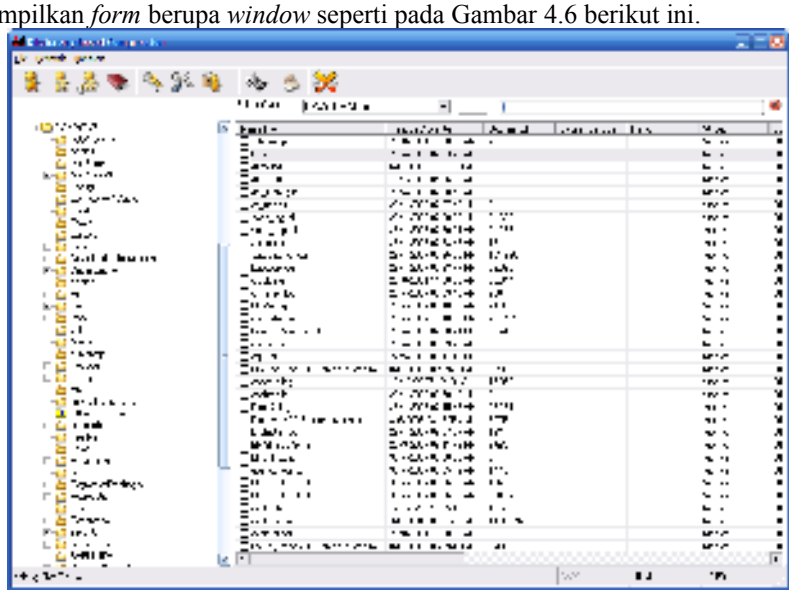
Program ini dijalankan dengan menggunakan perangkat keras (*hardware*) yang mempunyai spesifikasi minimal adalah sebagai berikut :

1. Prosesor Intel Pentium IV 2.0 GHz.
2. Memory 256 MB.
3. Harddisk 120 GB.
4. PCI VGA *card* 64 MB.
5. Monitor dengan resolusi 1024 × 768 *pixel*.
6. *Mouse*
7. *Keyboard*

Adapun perangkat lunak (*software*) yang digunakan untuk menjalankan aplikasi ini adalah lingkungan sistem operasi MS-Windows 98 atau MS-Windows NT/2000/XP.

4.3 Cara Menjalankan Program

Proses untuk menjalankan program kompresi dan dekompresi dengan algoritma DBC ini dengan mengklik pada *file executable* dengan *file* yang mempunyai *icon*  . Setelah dijalankan maka pertama sekali akan ditampilkan *form* berupa *window* seperti pada Gambar 4.6 berikut ini.



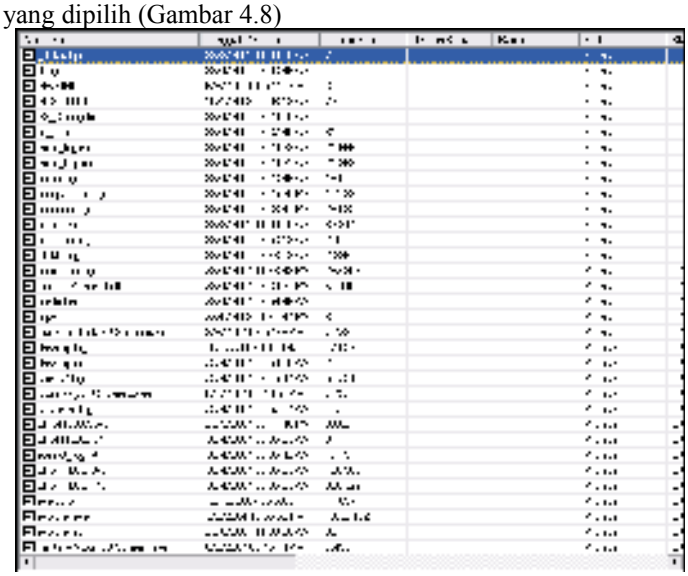
Gambar 4.6 Tampilan Antarmuka Program

Antarmuka program ini cukup sederhana dimana pada bagian sisi kiri merupakan *tree view* untuk memilih *folder* yang nantinya semua *file* pada *folder* ini akan ditampilkan *file list* di sebelah kanan atas. Selanjutnya adalah memilih satu *file* dengan mengklik pada *file* tersebut. Program secara otomatis akan menampilkan *file* yang dipilih tersebut pada *text box file* yang dipilih.

Pada bagian tampilan *folder* ini *user* hanya perlu memilih *folder* dengan mengklik langsung pada bagian *folder* ini dimana program akan secara otomatis menampilkan setiap *file-file* yang terdapat pada *folder* tersebut. Hasil tampilan dari *file* tersebut akan ditampilkan pada *list view* seperti terlihat pada Gambar 4.7.

Gambar 4.7 Tampilan Memilih *Folder*

Untuk pilihan yang sama, user dapat juga memilih *folder* atau *file* melalui menu. Pilihan menu File → Tambah File ke List ... maka program akan menambahkan satu *file* tunggal ke dalam *listview*. Pilihan File → Tambah Seluruh File dari Folder ke List ... maka program akan menambahkan seluruh *file* yang terdapat pada *folder* yang dipilih (Gambar 4.8)



Gambar 4.8 Tampilan Memilih *File* yang Akan Diproses

Untuk menentukan *file-file* mana yang akan diproses, maka terlebih dahulu perlu ditandai dengan mengklik pada tombol *check box* yang berada di samping kiri dari *file*. Jika *file* yang akan diproses terlalu banyak, *user* tidak perlu menandainya satu per satu tetapi dapat melalui pilihan menu File → Tandai Semua File pada List dan untuk proses yang berkebalikan *user* dapat memilih File → Hilangkan Tanda Cek pada File.

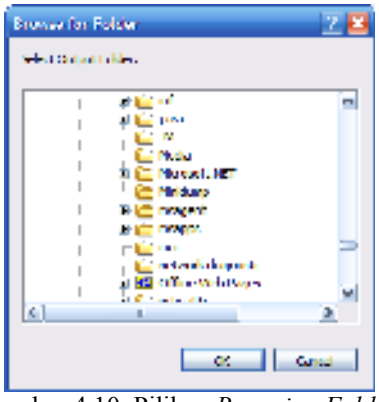
Untuk menghapus atau menghilangkan *file* terpilih dari listview, *user* dapat mengakses melalui menu File → Hapus File yang Dicek dari List. Sedangkan untuk menghilangkan seluruh *file* maka dapat dilakukan melalui menu File → Hapus Semua File dari List.

Sebagai catatan semua fungsi yang disebutkan di atas juga dapat dilakukan melalui klik *mouse* kanan pada bagian *listview*.

Berikutnya adalah menentukan *folder output* dengan cara mengisi pada *combo box* 'Folder Output' atau mengklik pada tombol di ujungnya (tombol bertanda ....). Kemudian program akan memunculkan sebuah kotak dialog untuk memilih *folder*. Setelah itu nama *file output* akan secara otomatis diberikan dan ditambahkan ekstensi *.DBC. Bagian ini akan menyimpan daftar *folder* yang pernah dipilih oleh *user* (Gambar 4.9).

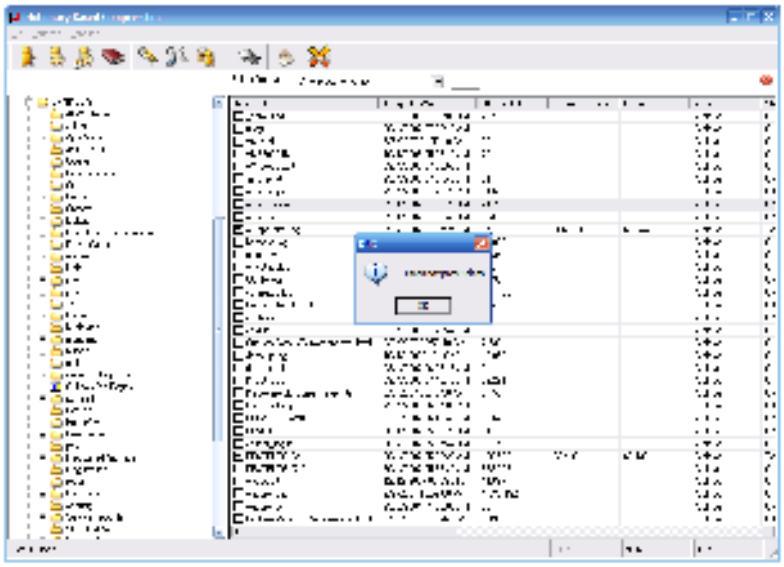
Gambar 4.9 Pilihan *Folder Output*

Tampilan untuk memilih *folder output* akan menampilkan sebuah kotak dialog untuk memilih *folder* seperti diperlihatkan pada Gambar 4.10 berikut ini.



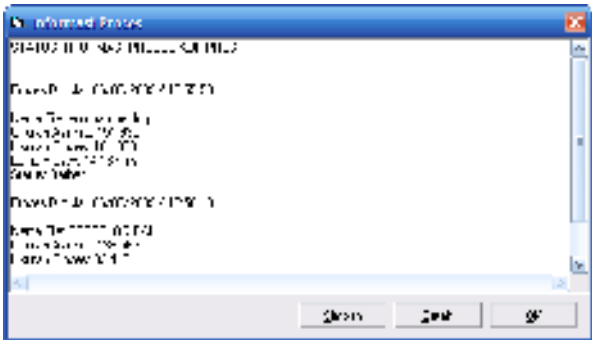
Gambar 4.10 Pilihan *Browsing Folder*

Setelah semua parameter yang diperlukan diisikan, maka untuk melakukan proses kompresi dapat diakses melalui menu Perintah → Kompresi File (Ctrl + K). Setelah itu untuk proses kebalikannya dapat diakses melalui menu Perintah → Dekompresi File (Ctrl + D) (Gambar 4.11)



Gambar 4.11 Tampilan Proses Kompresi & Dekompresi

Untuk menampilkan keterangan berupa informasi hasil proses dapat dilakukan melalui menu Perintah >> Informasi Hasil (Ctrl + L) dimana akan ditampilkan sebuah window seperti ditunjukkan pada Gambar 4.12. *User* dapat menyimpan teks log tersebut dengan menekan tombol Simpan ataupun mencetak langsung ke *printer* melalui tombol Cetak. Untuk menutup *window* informasi proses ini dapat dilakukan dengan menekan tombol OK.



Gambar 4.12 Tampilan Informasi Proses

4.4 Pengujian

Untuk mengetahui hasil pengujian algoritma DBC yang telah diimplementasikan dapat dilihat pada Tabel 4.1. Pengujian dilakukan dengan menggunakan spesifikasi komputer yang berbeda. Pada pengujian pertama dilakukan dengan menggunakan spesifikasi sebagai berikut:

1. Prosesor Intel Pentium Core 2 Duo 1.86 GHz
2. Memori DDR 2700 1 GB
3. *Motherboard* dengan FSB 800 MHz
4. *Hard disk* dengan ukuran 120 GB
5. VGA Card 128 MB
6. *Monitor* LCD
7. *Printer* untuk kebutuhan mencetak laporan
8. *Keyboard* dan *Mouse*

Pengujian ini dilakukan dengan membandingkan hasil kompresi antara program yang dirancang oleh peneliti dengan 3 (tiga) program kompresi yang banyak digunakan yaitu Winzip 9.0, WinRAR 3.0 dan 7-Zip 4.65. Pada pengujian ini peneliti menggunakan tiga jenis *file* yaitu *file bmp*, *file doc* dan *file txt*. Hasil pengujian dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil Pengujian

Nama File	Ukuran Asli	Ukuran Hasil	DBC	Winzip 9.0	WinRAR 3.0	7-ZIP 4.65
Uji1.bmp	196.664 *byte*	170.053 *byte*	86,47%	60,23%	44,02%	53,35%
Uji2.bmp	12.406 *byte*	8.773byte	70,72%	18,76%	17,56%	17,37%
Uji1.doc	31.232 *byte*	21.886 *byte*	70,08%	11,93%	11,05%	10,16%
Uji2.doc	41.984 *byte*	30.264 *byte*	72,08%	21,20%	20,46%	18,69%
Uji1.txt	1.330 *byte*	960 *byte*	72,18%	16,61%	14,36%	16,46%
Uji2.txt	1.150 *byte*	974 *byte*	84,70%	41,04%	39,21%	41,73%
Uji1.log	151.990 *byte*	104.320 *byte*	68,64%	89,77%	93,62%	93,58%
Uji1.log	29.082 *byte*	20.509 *byte*	70,52%	86,81%	88,80%	88,85%
Uji1.ico	122.187 *byte*	95.824 *byte*	78,42%	76,57%	77,62%	80,44%
Uji1.ico	118.714 *byte*	94.108 *byte*	79,27%	76,92%	78,67%	81,67%

Rata-Rata	–	–	75,31%	49,98%	48,54%	50,23%

Dari hasil pengujian dan perhitungan rata-rata rasio kompresi didapat hasil aplikasi WinRAR 3.0 mempunyai rasio kompresi yang paling kecil kemudian disusul oleh Winzip 9.0 dan 7-Zip 4.65, serta terakhir adalah algoritma DBC yang dibuat oleh peneliti. Dengan rasio kompresi rata-rata 48,54% berarti aplikasi WinRAR 3.0 mempunyai rasio reduksi rata-rata 51,46%, hasil rasio kompresi dari 7-Zip 4.65 mempunyai rasio kompresi rata-rata 50,23% berarti menghasilkan rasio reduksi rata-rata 49,77%. Untuk *Winzip 9.0* mempunyai rasio kompresi rata-rata 49,98% berarti menghasilkan rasio reduksi sebesar 50,02%. Hasil dari aplikasi DBC yang dibuat oleh peneliti mempunyai rasio kompresi rata-rata 75,31% dengan rasio reduksi 24,69%. Maka dapat disimpulkan bahwa hasil rasio kompresi dari aplikasi DBC masih jauh dibandingkan dengan hasil yang diperoleh dari ketiga aplikasi kompresi yang diuji. Pada bagian ini akan dilakukan pengujian terhadap beberapa jenis *file* yang mempunyai ukuran *file* yang besar yaitu di atas 1 MB.

Tabel 4.2 Pengujian Terhadap Ukuran *File* Besar

Nama File	Ukuran Asli	Ukuran Hasil	DBC	Winzip 9.0	WinRAR 3.0	7-ZIP 4.65
Komsal.mdb	2.211.840 bytes	2.152.850 byte	97,33%	6,02%	3,26%	2,80%
Superhugebg.bmp	5.534.102 bytes	5.201.106 byte	93,98%	7,85%	5,46%	3,82%
Word.doc	5.504.512 bytes	5.490.232 byte	99,74%	13,53%	12,51%	11,67%

Rata-Rata	-	-	97,01%	9,13%	10,61%	6,09%

Dari pengujian ini dapat disimpulkan juga proses efektif yang paling baik pada algoritma DBC adalah kompresi pada *file* berjenis teks seperti *txt* dan *log* seperti ditunjukkan pada Tabel 4.1 hingga Tabel 4.2. Bagian merupakan pengujian untuk lama proses kompresi untuk algoritma DBC seperti ditunjukkan Tabel 4.3 dan 4.4.

Tabel 4.3 Lama Proses Kompresi DBC

Nama File	Ukuran Asli	Lama Proses DBC
Uji1.bmp	196.664 *byte*	36.690 ms
Uji2.bmp	12.406 *byte*	828 ms
Uji1.doc	31.232 *byte*	61.328 ms
Uji2.doc	41.984 *byte*	201.718 ms
Uji1.txt	1.330 *byte*	2.281 ms
Uji2.txt	1.150 *byte*	1.521 ms
Uji1.log	151.990 *byte*	15.062 ms
Uji2.log	29.082 *byte*	984 ms
Uji1.ico	122.187 *byte*	65.281 ms
Uji2.ico	118.714 *byte*	59.609 ms

Rata-Rata Lama Waktu	44.530,20 ms

Tabel 4.4 Lama Proses Dekompresi DBC

Nama File	Ukuran Asli	Lama Proses DBC
Uji1.bmp.dbc	170.053 *byte*	2.031 ms
Uji2.bmp.dbc	8.773byte	344 ms
Uji1.doc.dbc	21.886 *byte*	219 ms
Uji2.doc.dbc	30.264 *byte*	320 ms
Uji1.txt.dbc	960 *byte*	94 ms
Uji2.txt.dbc	974 *byte*	16 ms
Uji1.log	104.320 *byte*	1.109 ms
Uji2.log	20.509 *byte*	266 ms
Uji1.ico	95.824 *byte*	1.407 ms
Uji2.ico	94.108 *byte*	1.063 ms
Rata-Rata Lama Waktu		686,90 ms

Dari hasil analisis kecepatan proses dapat disimpulkan bahwa rata-rata lama waktu proses untuk proses dekompresi lebih cepat dibandingkan dengan lama proses kompresi. Hal ini dikarenakan pada proses dekompresi tidak perlu lagi dilakukan proses pembentukan *dictionary* seperti pada proses kompresi melainkan hanya dilakukan pembacaan dan pencocokan dari data *dictionary* yang terdapat pada *file ouput* sehingga proses dekompresi dapat berjalan lebih cepat.

5. Kesimpulan

Berdasarkan pembahasan dari bab-bab sebelumnya yang telah dilakukan maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Dari hasil pengujian dan perhitungan rata-rata rasio kompresi didapat hasil aplikasi WinRAR 3.0 mempunyai rasio kompresi yang paling kecil kemudian disusul oleh Winzip 9.0 dan 7-Zip 4.65, serta terakhir adalah algoritma DBC yang dibuat oleh peneliti, tetapi untuk *file* berukuran besar maka rasio kompresi terbaik diperoleh dari 7-Zip.
2. Dari hasil pengujian ini dapat disimpulkan bahwa algoritma DBC tidak ada perbedaan mencolok antara hasil kompresi *file* berjenis bmp, doc, txt, log, dan ico dengan rasio kompresi berkisar antara 70-80% dengan catatan bahwa besar rasio kompresi teks paling rendah yaitu 74,01%. Tetapi pada ukuran *file* besar perbedaan sangat mencolok antara algoritma DBC dengan yang lainnya.
3. Proses dekompresi lebih cepat dibandingkan dengan proses kompresi dikarenakan pada proses dekompresi tidak perlu lagi dilakukan proses pembentukan *dictionary* seperti pada proses kompresi melainkan hanya dilakukan pembacaan dan pencocokan dari data *dictionary* yang terdapat pada *file ouput* sehingga proses dekompresi dapat berjalan lebih cepat.
4. Adapun keunggulan dari program ini adalah dapat memproses semua jenis *file* biner kecuali yang telah terkompresi dengan metode *dictionary based compression* yang merupakan metode kompresi *lossless* sehingga data dapat dikembalikan ke dalam bentuk aslinya.

Saran

Untuk pengembangan lebih lanjut program kompresi ini, maka dapat diberikan beberapa saran sebagai berikut:

1. Metode ini lebih disarankan untuk digunakan pada jenis *file* teks dan *bitmap* dengan variasi warna yang seragam.
2. Untuk memperkecil rasio kompresi dapat dilakukan dengan menggabungkan beberapa algoritma kompresi seperti *Huffman*, *Shannon Fano*, LBE dan lain sebagainya. Misalnya dengan melakukan elaborasi kode Huffman untuk menghasilkan *dictionary* yang lebih kompak.
3. Metode ini dapat di rekomendasikan untuk peralatan *gadget* seperti *handphone*, *smart phone* dan PDA mengingat algoritma yang cukup sederhana.

Referensi

[1] Nelson, M. and Gailly, J. L., **The Data Compression Book**, Second Edition, M & T Books, 2002.
[2] Kandaga, Tjatur, Analisis Penerapan Kompresi dan Dekompresi Data dengan Menggunakan Metode Statistik dan Kamus, Jurnal Informatika, Vol. 2, No.2, pp.81 - 91, Desember 2006.
[3] Salomon, David, Data Compression: The Complete Reference, Springer, 2004.
[4] Schild, Herbert, C The Complete Reference, 4th ed., osborne / McGraw-Hill,2000.

[5] Shannon, C. E., **A Mathematical Theory of Communication**, The Bell System Technical Journal, Vol. 27, pp.379 − 423, 623 − 656, July, October, 1948.
[6] Widhiartha, P., 2003, **Pengantar Kompresi Data**, www.ilmukomputer.com, tanggal akses 11 Maret 2011.
[7] Yourdan, Edward, 2003, **Modern Structured Analysis**, PDF Document.
[8] Anonim, **Byte Pair Encoding**, http://en.wikipedia.org/byte_pair_encoding, tanggal akses 15 Maret 2011.
[9] Anonim, **Burrows-Wheeler Transform**, http://en.wikipedia.org/burrows _wheeler_transform, tanggal akses 15 Maret 2011.
[10] Anonim, **Data Compression**, http://www.data-compression.com/index.shtml, tanggal akses 12 Maret 2011.