BAB IV

HASIL DAN UJI COBA

IV.1. Jalannya Uji Coba

Uji coba dilakukan terhadap beberapa *file* dengan ektensi dan ukuran berbeda untuk melihat hasil kompresi dari aplikasi yang telah selesai dirancang. *File* yang di uji berektensi **.html, *.txt, *.rtf, *.doc, *.bmp, *.jpg,* dan **.pdf*. Dalam menjalankan program ada beberapa langkah sebelum menggunakannya, yaitu :

1. Proses Kompresi

   Aplikasi diuji untuk melakukan kompresi *file,* apakah telah sesuai dengan perancangan, adapun cara melakukan kompresi melalui aplikasi ini adalah sebagai berikut.

   a. Menjalankan aplikasi, pada tampilan awal akan dijumpai *form splash* sebagai penanda bahwa aplikasi baru saja terbuka.
   b. Mencari *input file*, pada menu utama dilakukan penentuan lokasi *input file* asli dan lokasi *output file* kompresi.
   c. Memilih proses, untuk proses kompresi pada *combo* ”Aksi” dipilih proses yang akan dilakukan yaitu ”Kompresi”.
   d. Proses *file,* setelah semua *field* diisi, proses dapat dilakukan dengan klik tombol ”Proses”. Jika ada kesalahan *input* maka proses dibatalkan dan pesan kesalahan akan tampil.

e. Info proses, dari proses yang telah dilakukan aplikasi akan memberikan informasi dari perbandingan *input file* dan *output file*. Dan hasil *file* terkompresi dapat ditemukan pada *directory* yang telah di*inputkan*

2. Proses Dekompresi

Aplikasi diuji untuk mengetahui apakah telah sesuai perancangan untuk dapat mengembalikan data pada *file* yang sebelumnya telah terkompresi, cara melakukan kompresi melalui aplikasi ini adalah sebagai berikut.

a. Menjalankan aplikasi, pada tampilan awal akan dijumpai *form splash* sebagai penanda bahwa aplikasi baru saja terbuka.
b. Mencari *input file,* yaitu mencari *file* yang telah terkompresi dengan aplikasi, pada menu utama dilakukan penentuan lokasi *input file* terkompresi dan lokasi *output file* dekompresi.
c. Memilih proses, untuk proses kompresi pada *combo* ”Aksi” dipilih proses yang akan dilakukan yaitu ”Dekompresi”.
d. Proses *file,* setelah semua *field* diisi proses dapat dilakukan dengan klik tombol ”Proses”. Jika ada kesalahan *input* maka proses dibatalkan dan pesan kesalahan akan tampil.
e. Info proses, dari proses yang telah dilakukan aplikasi akan memberikan informasi dari perbandingan *input file* dan *output file*. Dan hasil *file* derkompresi dapat ditemukan pada *directory* yang telah di*inputkan*

IV.2. Tampilan Layar

Dari hasil rancangan tampilan layar terdiri dari *form-form* pendukung, seperti *form splash*, *form* utama, *form about* dan *form help*. Berikut tampilan layar dari hasil rancangan yang telah dibangun.

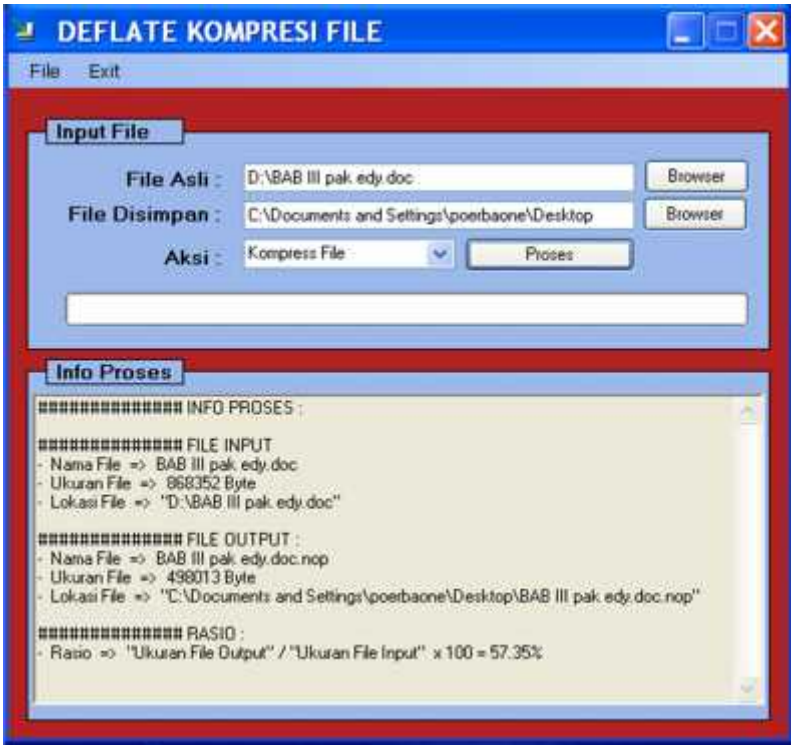
IV.2.1. Layar *Form Splash*

Form splash sebagai pengenal mengenai tujuan perancangan aplikasi, seperti judul dan logo STMIK Potensi Utama. Dalam beberapa detik akan tampil *form* utama yang berfungsi untuk melakukan kompresi, seperti gambar halaman bantuan IV.1 berikut.

Gambar IV.1. Layar *Form Splash*

IV.2.2. Layar *Form* Utama

Form utama adalah *form* yang berfungsi sebagai tempat bagi pengguna untuk melakukan *input* dan proses kompresi maupun dekompresi *file*, terdapat beberapa menu, yaitu menu *about* untuk menampilkan *form about*, menu *help* untuk menampilkan *form help*, dan menu *exit* untuk keluar dari aplikasi. Dan beberapa tombol yang berfungsi sebagai prosedur proses. Seperti gambar IV.2 berikut:



Gambar IV.2. Layar *Form* Utama

Dalam menggunakan aplikasi ini ada beberapa tahapan yang harus dilakukan pengguna diantaranya :

1. Pada *form* utama ini pengguna dapat menentukan *file* yang ingin diproses dengan klik tombol *browser* pada *input*an *file* asli.
2. Setelah itu menentukan disimpan dimana hasil proses yang akan dilakukan dengan klik tombol *browser* pada *input*an *file* disimpan.
3. Lalu memilih aksi yang akan dilakukan kompresi atau dekompresi.
4. Setelah semua ditentukan pengguna dapat melakukan proses dengan klik tombol proses. Dari hasil dapat diketahui info mengenai hasil proses serta rasio kompresi yang dilakukan.
5. Jika salah satu *file* inputan tidak diisi, saat kita klik tombol proses maka akan muncul informasi kesalahan. Seperti gambar VI.3

Gambar IV.3. Layar *Form* kesalahan

IV.2.3. Layar *Form About*

Pada *form* ini tidak ada *input* atau *output. Form* ini hanya menampilkan informasi dari tujuan perancangan aplikasi. dapat dilihat pada gambar IV.3 berikut.

Gambar IV.4. Layar *Form About*

IV.2.4. Layar *Form Help*

Pada *form* ini tidak ada *input* atau *output. Form* ini hanya menampilkan informasi dari cara penggunaan aplikasi. seperti pada gambar IV.4 berikut.

Gambar IV.5. Layar *Form Help*

IV.3. Hardware dan Software yang dibutuhkan

Untuk menjalankan aplikasi ini dibutuhkan beberapa perangkat, baik perangkat keras (*hardware*) maupun perangkat lunak (*software*).

1. Spesifikasi Perangkat Keras (*Hardware*) yang dibutuhkan diantaranya. Komputer setingkat dengan *Pentium* 4 atau diatasnya, *Keyboard, Monitor,* dan *Mouse.*
2. Spesifikasi Perangkat Lunak (*Software*) yang dibutuhkan agar aplikasi dapat berjalan yaitu, Sistem Operasi *Windows XP* dan *.Net Framework Version 2.0.*

IV. Analisa Hasil

Analisa hasil adalah analisis terhadap *output* yang dihasilkan oleh aplikasi, adapun analisis yang penulis jelaskan antara lain anlisis hasil kompresi dan analisis hasil dekompresi, yang dapat dijelaskan sebagai berikut.

1. Analisa Hasil Kompresi

Implementasi algoritma *deflate* dapat mengetahui seberapa jauh kemampuan untuk memampatkan data. Pada penjelasan sebelumnya, pengujian dilakukan kepada sebuah data teks. Pengujian berikutnya akan melibatkan beberapa jenis data, antara lain:

a. Data teks dengan ekstensi **.html*, **.txt*, **.doc*, **.rtf*, **.pdf*.
b. Data gambar dengan ekstensi **.bmp* dan **.jpg*.

Analisa hasil dapat dilihat pada tabel IV.1 dibawah ini, untuk ukuran tiap *file* ditampilkan dalam ukuran *byte*. Dan data yang ditampilkan adalah nama *file*, ukuran asli sebelum dikompresi, ukuran hasil terkompresi dan rasio hasil perbandingan kedua *file.*

Tabel IV.1. Analisa Hasil Kompresi

No	Nama File	Ukuran Asli (Byte)	Ukuran Terkompresi (Byte)	Rasio (%)
1	Test HTML.html	420.936	75.938	18,04
2	Test TXT.txt	1.933.302	134.691	6,96
3	Test DOC.doc	3.845.120	355.244	9,23
4	Test RTF.rtf	3.061.760	285323	9,31
5	Test PDF.pdf	40.145	28.030	69.82
6	Test BMP.bmp	1.752.714	16466	0.93
7	Test JPG.jpg	125.770	111.915	88.98

Hasil yang didapat dari uji coba yang dijelaskan pada tabel IV.1 diatas adalah bervariasi, hal ini diakibatkan tingkat redundansi data yang dimiliki tiap *file* berbeda-beda. Data teks berektensi **.html*, **.txt*, **.doc*, **.rtf* memiliki rasio kompresi yang lebih kecil. Ini dikarenakan data teks umumnya memiliki tingkat redundansi cukup tinggi.

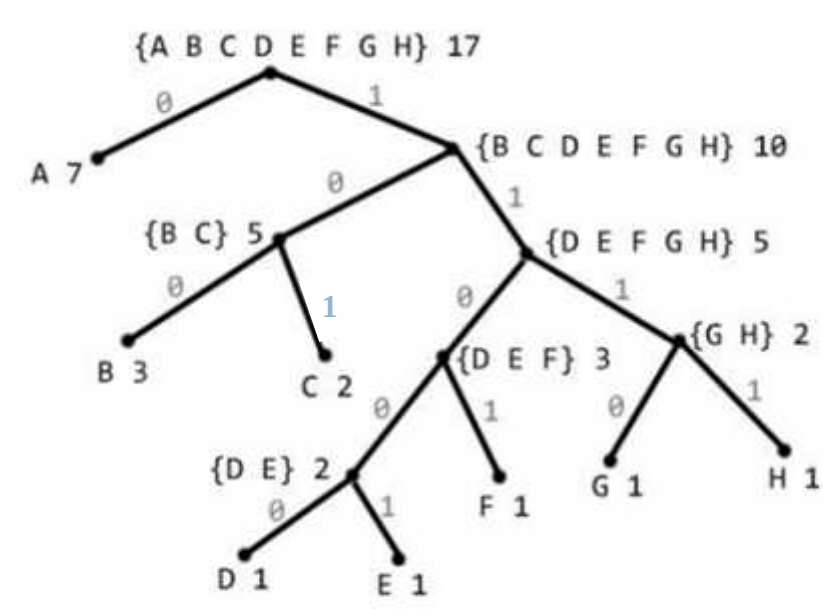
Proses Kompresi dari algoritma *deflate* dilakukan pada dua tahap. Tahap pertama adalah melakukan proses pembuatan blok-blok atau penyingkatan karakter dengan menggunakan algoritma LZ77. Tahap kedua adalah mengambil hasil penyingkatan karakter dari algoritma LZ77 dan melakukan proses kompresi dengan menggunakan *Huffman Tree* terhadap karakter tersebut. Hasil dari proses *encoding* algoritma *deflate* ini adalah berupa karakter yang telah disingkat yang merupakan hasil dari *encoding* algoritma LZ77 dan memiliki kode biner yang lebih pendek yang merupakan hasil dari *encoding* algoritma *Huffman.* Misal kita memiliki *string* berikut: ABFBAAABCHEAADACG

Pada *string* secara keseluruhan, terdapat huruf A sampai H, yaitu 8 huruf. Secara general, untuk merepresentasikan n karakter dengan simbol-simbol yang berbeda, kita membutuhkan log n bit per simbol. Berarti, pada kasus ini kita membutuhkan 2 log 8 atau 3 bit untuk setiap simbol. Setelah pengkodean, tiap karakter memiliki simbol unik sebagai berikut: A 000, B 001, C 010, D 011, E 100, F 101, G 110, H 111

Berarti, *string* di atas dapat direpresentasikan dengan simbol menjadi sebagai berikut : 000 001 101 001 000 000 000 001 010 111 100 000 000 011 000 010 110 . Sama dengan 17 * 3 = 51 bit. Hasil penyingkatan / pengelompokan

karakter menggunakan algoritma LZ77 dari *string* diatas menjadi ABCDEFGH. Disimpan didalam sebuah *Dictionary*. Tahap kedua adalah melakukan proses kompresi dengan menggunakan *Huffman Tree* terhadap karakter tersebut. Langkah-langkah untuk men-encoding suatu *string* biner adalah sebagai berikut :

1. Tentukan karakter yang akan di-encoding
2. Mulai dari akar, baca setiap bit yang ada pada cabang yang bersesuaian sampai ketemu daun dimana karakter itu berada.
3. Ulangi langkah 2 sampai seluruh karakter di-encoding



Dari hasil pembacaan pohon diatas, didapatkan simbol untuk masing-masing karakter sebagai berikut:

Karakter	Kode Huffman
A	0
B	100
C	101
D	11000
E	11001
F	111
G	1110
H	1111

Jadi hasil yang didapat dari pohon tersebut adalah 0 100 101 11000 11001 111 1110 1111 menjadi 28 bit.

Jadi, Hasil yang didapat dari proses kompresi adalah

Ukuran sebelum di kompresi = 51 bit

Ukuran terkompresi = 28 bit

Rasio Kompresi $= \frac{\text{Ukuran terkompresi}}{\text{ukuran Asli}} = \frac{28}{51} x\ 100\% = 54{,}9\ \%$

2. Analisa Hasil Dekompresi

Sama seperti analisa hasil kompresi diatas, beberapa tipe *file* didekompresi mengambil contoh dari *output* kompresi sebelumnya agar dapat diketahui hasil *output* dekompresi menghasilkan *file* aslinya. Adapun perbandingan *file* yang didekompresi dapat dilihat pada table IV.2.

Tabel IV.II. Analisa Hasil Dekompresi

No	Nama File	Ukuran Terkompresi (Byte)	Ukuran Asli (Byte)	Rasio (%)
1	Test HTML.html.nop	75.938	420.936	554,3
2	Test TXT.txt.nop	134.691	1.933.302	1.435
3	Test DOC.doc.nop	355.244	3.845.120	1.082
4	Test RTF.rtf.nop	285323	3.061.760	1.073
5	Test PDF.pdf.nop	28.030	40.145	143,2
6	Test BMP.bmp.nop	16466	1.752.714	10.644
7	Test JPG.jpg.nop	111.915	125.770	112,3

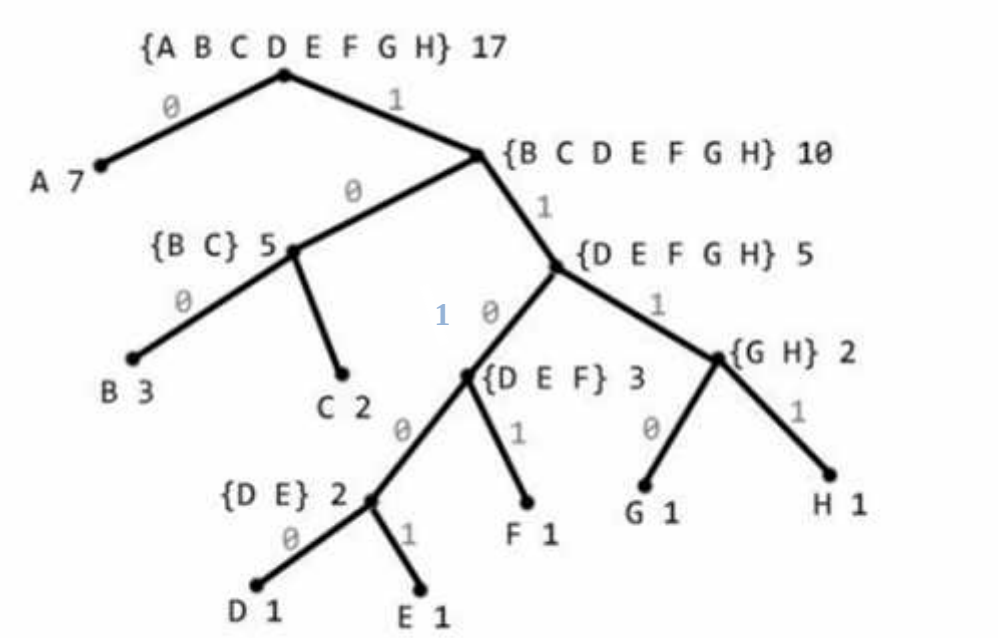
Hasil yang didapat dari uji coba yang dijelaskan pada tabel IV.2 diatas menunjukan rasio hasil *output file* menjadi lebih besar, hal ini dikarenakan *file* yang telah didekompresi kembali ke ukuran semula sebagaimana *file* aslinya.

Proses (dekompresi) *decoding* pada algoritma *deflate* merupakan kebalikan dari proses *encoding*nya (kompresi). Langkah pertama adalah melakukan proses *decoding* dengan menggunakan algoritma *Huffman*. Kemudian langkah kedua adalah mengambil hasil dari proses *decoding* dengan menggunakan algoritma *Huffman* dan melakukan proses *decoding* kembali dengan menggunakan algoritma LZ77.

Langkah-langkah men-*decoding* suatu *string* biner dengan menggunakan pohon Huffman adalah sebagai berikut :

1. Baca sebuah bit dari *string* biner.
2. Mulai dari akar
3. Untuk setiap bit pada langkah 1, lakukan traversal pada cabang yang bersesuaian.
4. Ulangi langkah 1 , 2 dan 3 sampai bertemu daun. Kodekan rangkaian bi yang telah dibaca dengan karakter di daun.

5. Ulangi dari langkah 1 sampai semua bit di dalam *string* habis.



Sebagai contoh kita akan men-decoding *string* biner yang bernilai ”0”Setelah kita telusuri dari akar, maka kita akan menemukan bahwa *string* yang mempunyai kode Huffman “0 ” adalah karakter A. kemudian *string* biner 100 dapat Karakter B dan selanjutnya sampai membentuk Karakter ABCDEFGH. Kemudian langkah kedua adalah mengambil hasil dari proses *decoding* dengan menggunakan algoritma *Huffman* dan melakukan proses *decoding* kembali dengan menggunakan algoritma LZ77. Urutan *index* menjadi acuan untuk dapat menghasilkan *file* asli, *String* diambil dengan cara mencari posisi karakter pada *dictionary* dan menggabungkan karakter – karakter yang telah diambil dari *dictionary* sehingga ditemukan bit yang cocok untuk ditulis kembali menjadi ABFBAAABCHEAADACG .

IV.4. Kelebihan dan Kekurangan

Pandangan penilaian kelebihan dan kekurangan dari hasil perancangan hingga pengembangan penulis simpulkan dengan beberapa keterangan sebagai berikut.

1. Kelebihan dari aplikasi yang dirancang :
    a. Aplikasi memproses *file* dan memberikan info proses berupa informasi *input* dan *output file* sehingga pengguna dapat menyimpulkan sejauh mana aplikasi ini berhasil memapatkan *file* yang diproses.
    b. Aplikasi ini mampu memampatkan *file* dengan ektensi bertipe **.html*, **.txt*, **.rtf*, **.doc*, **.bmp*, **.jpg*, dan **.pdf*, dengan berbagai ukuran dalam mengkompresi *file*.
    c. Aplikasi ini juga dapat mendekompresi *file* yang telah dikompresi kembali ke *file* asli tanpa merusak data dari *file* sebelumnya.
2. Kekurangan dari aplikasi yang dirancang :
    a. Dalam proses kompresi, aplikasi ini belum mampu mengatasi untuk memproses *file* dengan ukuran yang sangat kecil. Sehingga perlu pengembangan yang lebih terhadap algoritma yang diterapkan
    b. Dengan semakin banyaknya pengembang aplikasi kompresi *file*. Kompresi *file* dengan algoritma *deflate* ini masih belum maksimal dalam memampatkan berbagai jenis *file.*