Aplikasi Kompresi Dekompresi File menggunakan Algoritma Fibonacci

Benny [1] Stephen [2]
STMIK IBBI
Jalan Sei Deli No. 18. Telp 061-4567111
Email: bennyshen77@gmail.com [1] stephen_wuz@live.com [2]

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 Fibonacci yang bekerja berdasarkan atas pengkodean dengan sistem universal code dimana nilai positif integer dari data untuk membentuk code word yang berbentuk biner dengan memanfaatkan bilangan Fibonacci. Hasil dari tulisan ini adalah suatu perangkat lunak yang dapat melakukan kompresi dan dekompresi balik berdasarkan pada algoritma Fibonacci pada semua jenis file yang belum dikompresi. Program juga dapat menampilkan besarnya rasio kompresi, rasio dekompresi, dan lama proses yang telah dikerjakan pada file serta mempunyai fasilitas proteksi dengan menginput password untuk mengamankan file hasil kompresi. Hasil pengujian dengan metode kompresi Fibonacci juga dapat menampilkan rentang rasio reduksi.

Kata kunci : kompresi file , Fibonacci.

Abstract

File represent containing digital data of informations. File size measure which too big will become the problem of /when the file will be transferred or commuted for. Is for that required by the way of certain to minimize file size measure. One of the its way to the problem of above is the file compacted to pass process of kompresi so that its size measure become smaller than size measure initialy and take a short cut time when transmission. One of the method able to wear for the compress of file size measure is with method of Fibonacci laboring by virtue of code with universal system of code where positive value of integer of data to form word code which in form of binary by exploiting number of Fibonacci. Result of from this article is a software able to do decompress and compress return pursuant to at algorithm of Fibonacci at all of file type which not yet compress. Program also can present the level of ratio of kompresi, decompression ratio, and llama process which have been done by file and also have facility of proteksi with input of password to protect file result of compress Result of examination with method of compress Fibonacci also can present to span discount ratio

Keyword : file ckompress, Fibonacci.

1. Pendahuluan

Kompresi data atau dikenal juga sebagai pemadatan 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 dan memudahkan transfer data. Selain itu pada suatu file banyak terdapat redudansi data serta untuk mempersingkat waktu transmisi sewaktu file tersebut dikirim atau di-download melalui jaringan Internet.

Kompresi data dapat dibagi menjadi dua bagian yaitu lossy dan lossless compression. Kompresi lossy menghilangkan informasi tertentu yang tidak diperlukan untuk memperkecil ukurannya sedangkan lossless tidak menghilangkan informasi pada data dan data dapat dikembalikan ke bentuknya semula. Terdapat banyak metode kompresi lossless dintaranya Huffman, LZW, LZ77, LZ78, BWT, Fibonacci, Splay Tree, dan lain-lain. Metode-metode tersebut mempunyai rasio kompresi dan kecepatan proses yang cepat.

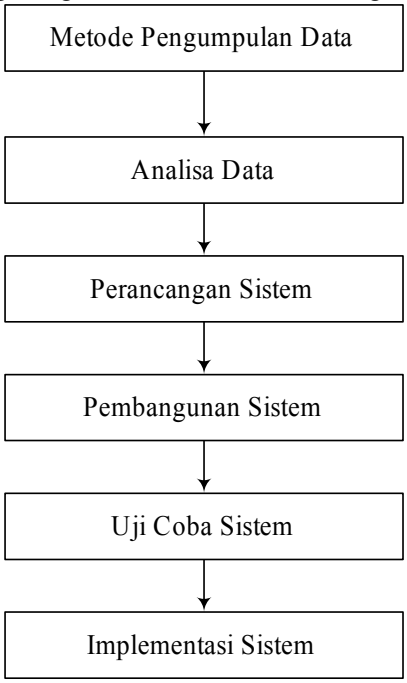
Metode kompresi Fibonacci merupakan jenis metode kompresi yang berdasarkan atas pengkodean dengan sistem universal code dimana nilai positif integer dari data untuk membentuk code word yang berbentuk biner dengan memanfaatkan bilangan Fibonacci. Untuk jenis metode kompresi dengan pengkodean universal code biasanya lebih cepat dibandingkan dengan metode dengan memanfaatkan pohon biner seperti Huffman.

2. Metode Penelitian

Aplikasi ini dirancang dengan menggunakan metodogi *Rapid Application Development* (RAD). Perancangan sistem menggunakan UML (*Unified Modelling Language*) untuk menggambarkan proses kerja beserta dengan algoritma program sedangkan perancangan *interface* dilakukan untuk merancang *user interface* untuk fungsi-fungsi seperti yang disebutkan di atas. Rancangan *interface* akan dilakukan pada IDE (*Integrated Development Environment*) dari *Visual Basic 6.0*. Digunakan untuk melakukan pengujian atas program yang dirancang dan bila terdapat kesalahan akan dilakukan koreksi. Adapun cara pengujian program dilakukan dengan menguji sejumlah *input*. Apabila ditemukan kesalahan maka program akan di-*debug* per baris untuk menentukan kesalahan dalam hal implementasi.

2.1 Kerangka Kerja

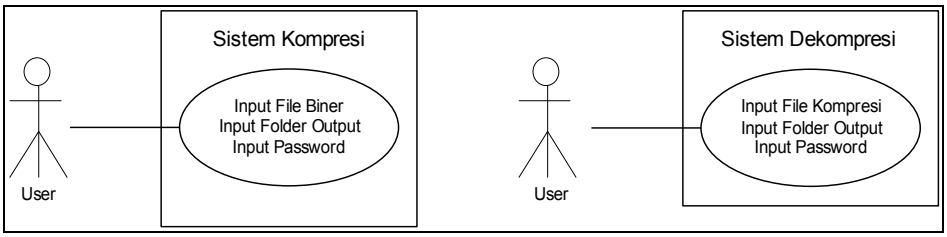
Aplikasi ini dirancang dengan menggunakan metodogi *Rapid Application Development* (RAD). Adapun bentuk langkah perancangan digambarkan dalam bentuk diagram seperti pada Gambar 1



Gambar 1 Langkah dalam Perancangan dengan *Rapid Application Development*

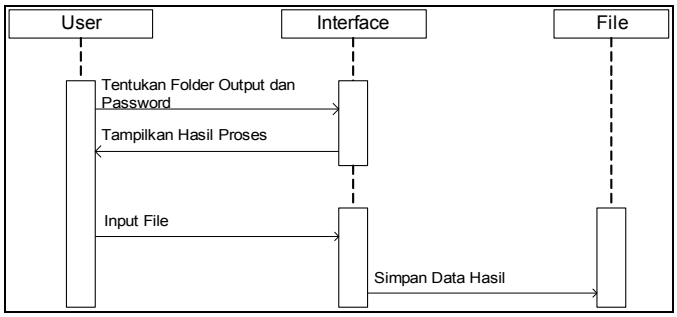
2.2 Analisis Proses

Pada bagian analisis proses ini menggunakan UML untuk menggambarkan proses kompresi dan cara kerja program. Adapun rancangan UML ini diperlihatkan pada Gambar 2.



Gambar 2 Diagram UML

Pada *use case* di atas terlihat terdapat aktor yaitu user yang akan menggunakan sistem ini. Aktor user terhadap sistem kompresi dan dekompresi mempunyai hak akses terhadap *file* biner menentukan *folder output* dan mengisikan *password*. *File input* untuk proses kompresi adalah *file* biner sedangkan *file input* untuk proses dekompresi adalah *file* hasil kompresi.
Selanjutnya dari masing-masing aktivitas pada *use case* tersebut dapat dikembangkan lagi bentuk *Sequence Diagram* sebagai berikut:



Gambar 3 *Sequence Diagram*

2.3 Pembahasan Cara Kerja

Pada bagian pembahasan ini akan dilakukan pembahasan mengenai cara pemanfaatan *Fibonacci coding* yang merupakan *universal code* untuk melakukan kompresi terdapat data. Sebelum penjelasan mengenai cara kerja *Fibonacci coding* tersebut maka perlu diperhatikan bahwa semua metode *universal code* memperlakukan data yang akan diproses dalam suatu bentuk bilangan apakah bilangan berjenis *integer* ataupun *long*. Bentuk bilangan tersebut diubah menjadi kode biner kemudian diproses dan disimpan kembali. Sebagai contoh misalkan suatu string yang terdiri atas 4 huruf (4 *byte* = 32 *bit*) yang dapat dikategorikan sebagai jenis data *long* sebelum diproses dengan teknik *universal code* mempunyai jumlah *bit* 4 *byte* x 8 *bit* = 32 *bit*. Setelah dilakukan pemrosesan dengan *universal code* maka jumlah *bit* akan menjadi lebih kecil dari 32 sehingga dapat dilakukan penghematan ruang penyimpanan.

Fibonacci coding merupakan universal *code* yang dapat mengkodekan integer positif menjadi kode biner dalam bentuk *code word*. Semua *token* akan diakhiri dengan angka "11" dan tidak akan menggandung angka "11" sebelum akhir *code word*-nya.

Rumus yang digunakan untuk menghasilkan kode Fibonacci adalah:

1. $N = \sum_{i=0}^{k} d(i)F(i)$
2. $d(i) = 1 \rightarrow d(i+1) = 0$

Kode Fibonacci hampir sama dengan hubungannya dengan representasi Fibonacci, yang merupakan sistem numerik dengan posisi yang kadang kala digunakan oleh para ahli matematika. Kode Fibonacci dikhususkan untuk mengkodekan bilangan bulat sehingga kadang kala disebut juga dengan representasi integer Fibonacci, kecuali dengan *order* digit yang dibalik dan penambahan "1" pada nilai hasil akhirnya.

Untuk melakukan pengkodeaan suatu bilangan bulat X dapat dilakukan dengan tahapan sebagai berikut:

1. Carilah suatu bilangan Fibonacci terbesar yang sama ataupun kurang dari X; Kurangkan bilangan X dengan bilangan tersebut, dan pertahankan hasil sisa pengurangannya.
2. Jika nilai yang dikurangkan sama dengan posisi ke-n dari bilangan unik Fibonacci, tempatkan angka "1" pada digit ke N sebagai *output*.
3. Ulangi langkah 1 dan 2 di atas, dengan menggatikan sisa pengurangan sekarang menjadi X, hingga mendapatkan nilai *remainder* 0.
4. Tempatkan angka "1" setelah hasil akhir pada hasil yang didapat dan angka 0 untuk setiap posisi yang tidak ditempat.

Sedangkan proses *decoding* balik dari *code word* hasil Fibonacci *coding* adalah:

1. Hilangkan nilai angka "1"

2. Gantilah nilai *bit* dengan nilai 1,2,3,5,8,13,...dan seterusnya (bilangan Fibonacci) dan kemudian hasilnya ditambah dengan 1.

3. Hasil dan Analisis

3.1 Proses Kompresi

Seperti yang telah diketahui sebelumnya secara teori proses kompresi terhadap suatu *file* adalah merepresentasikan string ataupun isi dari *file* dengan *code word* ataupun *string* lain yang lebih pendek sehingga didapatkan keuntungan sewaktu *file* disimpan kembali.

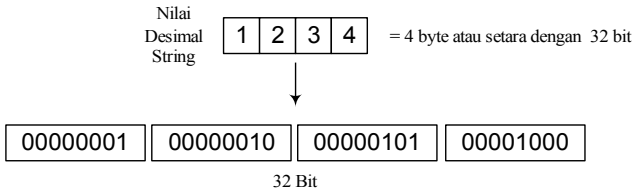
Proses dari kompresi menggunakan Fibonacci *coding* dilakukan dengan cara pembacaan isi *file* secara per *byte* kemudian mengubah nilai byte tersebut menjadi suatu bilangan bulat dilanjutkan dengan mencari *code word* atau Fibonacci *code* yang sesuai dengan nilai tersebut. Dengan cara seperti ini maka nilai *code word* yang memberikan keuntungan adalah nilai *code word* dengan jumlah representasi sebanyak 6 *bit* saja. Untuk itu bilangan bulat yang dapat direpresentasikan haruslah yang lebih kecil dari nilai 15.

Seperti diketahui bahwa jika pembacaan dilakukan secara per satu *byte* (8 bit = 2^8) maka nilai kisaran yang dihasilkan adalah 0 – 255 atau sebanyak 256 kemungkinan. Untuk itu pada proses ini akan dilakukan pada bilangan di bawah nilai 15 dikarenakan representasi di atas ini sudah menghabiskan ruang 8 *bit* juga.

Dengan perkataan lain semakin kecil nilai *byte* yang dibaca maka semakin pendek pula *code word* yang diperlukan untuk merepresentasikan nilai ini. Bila suatu isi *file* banyak mengandung nilai yang kecil ini dalam jumlah yang banyak maka rasio kompresi akan menjadi kecil sehingga rasio reduksi akan menjadi besar. Sebagai contoh misalnya suatu *file* mempunyai rasio kompresi sebesar 80% berarti rasio reduksinya adalah 20% yang berarti ukuran *file* dikurangi sebesar 20%.

Agar lebih dipahami bagaimana cara kerja kompresi dengan Fibonacci *coding* maka penulis akan menjelaskan dalam bentuk contoh:

Misalkan dalam suatu *file* tertentu terdapat 4 buah karakter *string* dimana masing-masing karakter ini mempunyai nilai 1{SOH}, 2{STX}, 3 {ETX}, dan 4 {EOT} (nilai dalam desimal ASCII) seperti terlihat pada Gambar 4.



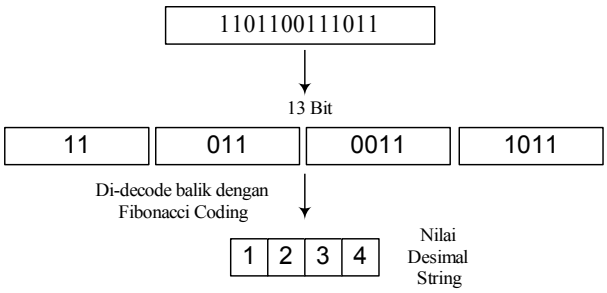
Gambar 4 Contoh *Input*

3.2 Proses Dekompresi

Proses dekompresi yang merupakan proses kebalikan dari proses di atas. Masalah yang muncul adalah apabila 13 *bit* di atas telah digabungkan maka bagaimana kita mengetahui bahwa nilai dari *byte* yang satu dengan yang lainnya. Karena keistimewaan dari Fibonacci *coding* yang selalu membentuk kode unik ”11” di setiap akhir hasilnya maka gabungan dari bit seperti pada contoh di atas dapat ditelusuri. Sebagai contoh *string* di bawah ini merupakan gabungan dari 13 *bit* di atas.

1101100111011

Pada proses penelusuran akan dilakukan pembacaan dari bit secara satu per satu bit. Jika ditemukan pembacaan bit dengan pasangan yang berturut-turut ”11” maka pembacaan akan diinterupsi dan hasil bit pertama akan diambil hingga menemukan pasangan bit tersebut. Sebagai contoh di atas pembacaan posisi bit pertama mendapat nilai ”1” kemudian dilakukan pembacaan pada bit kedua dan ditemukan nilai ”1” lagi, disebabkan gabungan dari kedua bit tersebut menghasilkan pasangan bit ”11” maka kedua nilai tersebut akan diambil. Pembacaan kemudian dilanjutkan dengan bit ketiga yang menghasilkan bit ”0” dan dilanjutkan dengan bit keempat menghasilkan bit ”1” karena ”01” bukanlah pasangan ”11” maka akan dilanjutkan pada bit kelima yang merupakan bit ”1” dan karena bit sebelumnya juga ”1” maka telah terbentuk pasangan ”11” dan nilai ”011” akan diambil. Begitulah seterusnya hingga bit terakhir dimana jumlah pengambilan berarti sama dengan jumlah representasi karakter.



Gambar 5. Proses Dekompresi

3.3 Algoritma

Algoritma merupakan langkah terstruktur dalam menyelesaikan suatu masalah. Algoritma dalam program ini secara garis besar dapat diuraikan menjadi:

1. Algoritma Membentuk Kode Fibonacci

```
Untuk Nilai Y dari 1 hingga 257
    Set Nilai Value Menjadi Y
    Set Nilai Fibonacci(Y).LeadingZero = 0
    Set Nilai Fibonacci(Y).Value = 1
    Set Nilai bitcount = 0
    Untuk Nilai X = 11 Hingga 0 Berkurang -1
        Jika Nilai - BitNumVal(X) < 0 Maka
            Jika NIlai Fibonacci(Y).Value > 1 Maka
                Set Nilai Fibonacci(Y).LeadingZero =  Fibonacci(Y).LeadingZero + 1
            Akhir If
            Jika Tidak
                Set Nilai bitcount = bitcount + 1
                Set Nilai Fibonacci(Y).Value =  Fibonacci(Y).Value + 2 ^ bitcount
                Set Nilai Fibonacci(Y).LeadingZero = -1 * (X  > 0)
                Set Nilai Value = Value - BitNumVal(X)
                Set X = X - 1
            Akhir If
        Jika bitcount > 0 Maka
            Set Nilai bitcount = bitcount + 1
        Akhir If
        Loop X
        Loop Y
```

2. Algoritma Membaca Kode Fibonacci

```
Set BitVal = BacaBitdariArray(FromArray, Posisi, 1)
    Jika BitVal = 1 Maka
```

```
            Jika LastCode = True Maka
                Keluar dari Looping
            Jika Tidak
                Set Nilai LastCode = True
        Akhir If
        Set Nilai Temp = Temp + BitNumVal(bitcount)
        Jika Tidak
            Set Nilai LastCode = False
        Akhir If
        Set Nilai bitcount = bitcount + 1
        Loop
    Set Nilai ReadFibonacciCode = Temp – 1
```

3. Algoritma Menambah Bit ke *Array*

```
Untuk Setiap X = Numbits - 1 Hingga 0 Berkurang -1
    Set Nilai OutByteBuf = OutByteBuf * 2 + (-1 *   ((Number And 2 ^ X) > 0))
    Set Nilai OutBitCount = OutBitCount + 1
    Jika OutBitCount = 8 Maka
        Set Nilai Toarray(OutPos) = OutByteBuf
        Set Nilai OutBitCount = 0
        Set Nilai OutByteBuf = 0
```

```
            Set Nilai OutPos = OutPos + 1
            Jika OutPos > Panjangdari(Toarray) Maka
                Set Toarray(OutPos + 500)
            Akhir If
        Akhir If
    Loop dari X
```

4. Algoritma Membaca Bit dari Array

```
    Untuk Setiap X = 1 Hingga Numbits
        Set  Nilai Temp = Temp * 2 + (-1 *  ((FromArray(FromPos) And 2 ^ (7 - ReadBitPos)) > 0))
        Set Nilai ReadBitPos = ReadBitPos + 1
        Jika ReadBitPos = 8 Maka
            Jika Posisi + 1 > NilaiTerbesar(FromArray) Maka
                Lakukan Looping X < Numbits
                Set Temp = Temp * 2
                Set X = X + 1
                Loop
                    Set FromPos = FromPos + 1
                Keluar dari Loop
            Akhir If
            Set Nilai FromPos = FromPos + 1
```

```
            Set Nilai ReadBitPos = 0
        End If
        Next
        ReadBitsFromArray = Temp
```

5. Algoritma Reducer

Lakukan pembacaan stream pada *file*
Konversi ke dalam bentuk bit
Lakukan split atas kelompok 4 bit
Simpan sebagai byte stream baru

6. Algoritma Kompresi Fibonacci

Lakukan Pembentukan Kode Fibonacci

```
Set OutStream =(PanjangDari(ByteArray))
Untuk setiap X = 0 hingga NilaiTerbesar(ByteArray)
Panggil Prosedur AddFibonacciToArray(OutStream,
    ByteArray(X))
Loop X
Panggil Prosedur AddFibonacciToArray(OutStream, 256)
    Jika OutBitCount > 0 Maka
        Panggil Prosedur AddBitsToArray(OutStream, 0, 8 – OutBitCount)
    Akhir If
Cetak Hasil ByteArray(OutPos)
```

7. Algoritma Dekompresi Fibonacci

Panggil Prosedur untuk Membentuk Kode Fibonacci

```
Set OutStream = (Len(ByteArray))
Set Char = ReadFibonacciCode(ByteArray)
Lakukan Looping While Char <> 256
Panggil AddCharToArray(OutStream, Char)
    Set Char = ReadFibonacciCode(ByteArray)
        Loop
    Set OutPos = OutPos - 1
Cetak Hasil ByteArray(OutPos)
```

3.4 Pengujian

Untuk mengetahui hasil pengujian algoritma Fibonacci yang telah diimplementasikan dapat dilihat pada Tabel 3.3. Pengujian dilakukan dengan menggunakan spesifikasi komputer yang berbeda. Pada pengujian pertama dilakukan dengan menggunakan spesifikasi sebagai berikut:

1. Prosesor Intel Core i3 2.0 GHz
2. *Memory* SDRAM 1024 MB
3. *Harddisk* 40GB
4. Sistem Operasi *Windows XP Profesional*

Pada pengujian kedua dilakukan dengan menggunakan spesifikasi sebagai berikut:

1. Prosesor Intel PIV 1.7 GHz
2. *Memory* SDRAM 512 MB
3. *Harddisk* 40GB
4. Sistem Operasi *Windows XP*

Hasil pengujian terhadap waktu kompresi dapat dilihat pada Tabel 3.3 berikut.

Tabel 1. Hasil Pengujian Waktu Kompresi

No.	Nama File	Ukuran File (byte)	Ukuran File Output (byte)	Rasio Kompresi	Lama Proses Komputer I (ms)	Lama Proses Komputer II (ms)
1.	Autopatcher.Log	151.990	112.283	73,88%	1.562	**2.435**
2.	Bab1.Doc	37.376	23.110	61,83%	282	**575**
3.	Fibonacci.Exe	696.320	526.732	75,65%	7.140	**11.812**
4.	Banana.ani	11.904	6.850	57,54%	94	**212**
5.	Book1.Xls	13.824	7.633	55,22%	156	**352**
6.	Presentasi.Ppt	108.032	81.411	75,36%	1.172	**1.780**
7.	Test.Ico	32.038	19.404	60,57%	281	**356**
8.	**Coba.Mdb**	**385.024**	**214.243**	**55,64%**	**2.219**	**4.210**

Tabel 2. Hasil Pengujian Waktu Dekompresi

No.	Nama File	Ukuran File (byte)	Ukuran File Output (byte)	Rasio Dekompresi	Lama Proses Komputer I (ms)	Lama Proses Komputer II (ms)
1.	Autopatcher.Txt.Fbc	112.283	151.990	135,36%	704	**2048**
2.	Bab1.Doc.Fbc	23.110	37.376	161,73%	157	**541**
3.	Fibonacci.Exe.Fbc	526.732	696.320	132,20%	3.406	**9.257**
4.	Banana.Ani.Fbc	6.850	11.904	173,78%	47	**189**
5.	Book1.Xls.Fbc	7.633	13.824	181,11%	49	**254**
6.	Presentasi.Ppt.Fbc	81.411	108.032	132,70%	609	**1.840**
7.	Test.Ico.Fbc	19.404	32.038	165,11%	171	**564**
8.	**Coba.Mdb.Fbc**	**214.243**	**385.024**	**179,71%**	**1.344**	**4.678**

Dari hasil analisis di atas bahwa rata-rata kompresi pada *file* dokumen *seperti word, excel dan powerpoint* paling baik sedangkan hasil kompresi terburuk didapat pada *file executable* Hal ini dapat dijelaskan karena pada *file text* umumnya lebih sering dijumpai karakter yang berulang dan sama sedangkan pada *file executable* yang mengandung kode biner lebih sulit untuk menentukan karakter yang berulang karena isi *file* pada biner lebih acak.

4. Kesimpulan

Berdasarkan uji coba dari beberapa file, maka dapat diambil beberapa kesimpulan antara lain bahwa proses dekompresi lebih cepat dibandingkan dengan proses kompresi karena pada proses dekompresi proses pengubahan code word dapat langsung dilakukan pengecekan terhadap setiap bit dari isi file input. Untuk semua jenis file yang diujicoba dapat disimpulkan bahwa algoritma kompresi Fibonacci mempunyai rentang rasio reduksi rata-rata 20% - 40%. Ini berarti algoritma ini cukup baik mengingat cara kerja algoritma ini sederhana dan mudah diimplementasikan. Rata-rata kompresi pada file dokumen seperti word, excel dan powerpoint paling baik sedangkan hasil kompresi terburuk didapat pada file executable Hal ini dapat dijelaskan karena pada file text umumnya lebih sering dijumpai karakter yang berulang dan sama sedangkan pada file executable yang mengandung kode biner lebih sulit untuk menentukan karakter yang berulang karena isi file pada biner lebih acak. Ukuran file yang dapat diproses minimun sebesar 1 Kbyte dan maksimal sebesar 40 Megabyte.

Daftar Pustaka

[1.] Baca, R, 2002, The Fast Fibonacci Decompression Algorithm, Department of Computer Science, VSB - Technical University of Ostrava, Czech Republic.

[2.] Data Compression, http://www.data-compression.com/index.shtml, tanggal akses 18 April 2013.

[3.] Shannon, C. E., 2008, A Mathematical Theory of Communication, The Bell System Technical Journal, Vol. 27, pp. 379 − 423, 623 − 656, July, October.

[4.] Data Compression, http://ciips.ee uwa.edu.au/ ~morris/Year2/ PLDS210/index.html, tanggal akses 18 April 2013.

[5.] Wirth, N. 2006, Algoritma + Struktur Data = Program, Prentice-Hall, Inc., Diterjemahkan oleh P. Insap Santosa, Penerbit ANDI Yogyakarta