ALGORITMA: Jurnal Ilmu Komputer dan Informatika
Volume: 05, Number: 01, April 2021 ISSN 2598-6341 (online)

Analisis Parameter Kompresi Algoritma Elias Omega Code dan Fibonacci Code Pada File Digital

Ibrahim Hasan[1], Tommy[2], Nenna Irsa Syahputri[3]
Universitas Harapan Medan, Indonesi[1,2,3]
Email : ibrahimsansan17@gmail.com[1], tomshirakawa@gmail.com[2],
nenna.ziadzha@gmail.com[3]

ABSTRAK

Kebutuhan terhadap kapasitas penyimpanan yang semakin besar merupakan penyebab munculnya berbagai teknik kompresi. Dengan melakukan kompresi, data yang besar akan berkurang ukurannya sehingga dapat menghemat kapasitas penyimpanan. Dalam penelitian ini menggunakan algoritma *Elias Omega Code* dan *Fibonacci Code*, yang akan diukur kinerjanya dengan *ratio of compression*, *compression ratio* dan *space saving* pada file digital teks. Kompresi dilakukan dengan membaca *String* pada file digital teks, selanjutnya algoritma *Elias Omega Code* dan algoritma *Fibonacci Code* membuat kode *String* dan melakukan proses kompresi. Dari hasil akhir kompresi adalah File berekstensi *.EB dan *.FB berisi informasi karakter dan *String* Bit dari hasil kompresi yang dapat didekompresi. Dari hasil keluaran dekompresi adalah file digital teks yang asli yang disimpan dengan ektensi file *.txt. Pada pengujian sitem ini digunakan sampel yaitu *string* yang terdiri dari satu jenis karakter yang tersimpan pada file digital teks yang berektensi *.txt. Dari proses kompresi pada *string* karakter file digital, algortima *Fibonacci Code* lebih unggul dalam hal kompresi *ratio of compression* dengan rata-rata 1,72, *compression ratio* dengan rata-rata 57% dan *space saving* dengan rata-rata 42%.

Kata Kunci: *analisis, kompresi, teks, Elias Omega Code, Fibonacci Code*

ABSTRACT

*The need for greater storage capacity is the cause of the emergence of various compression techniques. By compressing, large data will be reduced in size so as to save storage capacity. In this research, using the Elias Omega Code and Fibonacci Code algorithms, the performance will be measured with the ratio of compression, compression ratio and space saving in digital text files. Compression is done by reading the string in a digital text file, then the Elias Omega Code algorithm and the Fibonacci Code algorithm create a string code and perform the compression process. From the final result of the compression, files with extension * .EB and * .FB contain character information and Bit String from the compression result that can be decompressed. From the output of the decompression is the original digital text file that is saved with the file extension * .txt. In testing this system, a sample is used, namely a string consisting of one type of character stored in a digital text file with the extension * .txt. From the compression process on digital file character strings, the Fibonacci Code algorithm is superior in terms of compression ratio of compression with an average of 1.72, compression ratio with an average of 57% and space saving of an average of 42%.*

Keywords*: analysis, compression, text, Elias Omega Code, Fibonacci Code*

1. PENDAHULUAN

Teknologi pemampatan data atau dikenal dengan kompresi data adalah sebuah cara untuk memadatkan data sehingga hanya memerlukan ruangan penyimpanan lebih kecil, sehingga lebih efisien dalam menyimpannya atau mempersingkat waktu pertukaran data tersebut [1]. Data adalah kenyataan dan kejadian yang berisikan fakta. hasil dari pengolahan data adalah

informasi yang dapat disajikan dalam bentuk teks, gambar, suara, maupun video. Tingginya kebutuhan informasi yang dapat diakses, terpercaya, dan cepat, membuat ukuran data sangat mempengaruhi. Semakin besar data, semakin lama pula transfer data dan mengurangi efisiensi waktu memperoleh informasi.

Salah satu data yang dapat diandalkan dalam berbagi informasi adalah data dalam bentuk berkas digital. Untuk saat ini, data berkas digital memiliki banyak jenis diantaranya adalah data digital teks. Data teks yang mengandung karakter huruf, angka dan simbol tentu juga memiliki *space* yang sedikit besar, hal ini dikarenakan banyaknya informasi yang terkandung didalamya. Akan tetapi, pengguna didalam ruang penyimpananya tidak hanya menyimpan data berupa file teks, tentu ada juga beberapa file yang dapat berukuran yang lebih besar, sehingga beberapa file tersebut harus dapat saling berbagi ruang penyimpanan tanpa harus menghapus file penting lainya.

Maka untuk membuat banyak ruang kosong dan memiliki ukuran data yang tidak besar pada media penyimpanan diperlukan teknik kompresi yang memperkecil ukuran data. Kompresi data adalah proses di mana file (teks, audio, video) dapat ditransformasikan ke file lain (terkompresi), sehingga file terkompresi dapat sepenuhnya kembali seperti file asli tanpa kehilangan informasi yang sebenarnya [2]. Teknik kompresi membutuhkan suatu langkah-langkah penyelesaian yang disebut dengan algoritma. Salah satunya adalah algoritma *Elias Omega Code* dan *Fibonacci Code*.

Penelitian yang dilakukan Annirudha tentang kompresi algoritma *Elias Omega Code* dapat diambil kesimpulan bahwa proses kompresi algoritma hanya mengecilkan ukuran data 5 sampai dengan 8 persen saja [3]. Sedangkan Penelitian yang dilakukan Budiman, menyatakan bahwa algoritma *Fibonacci Codes* lebih efesien untuk kompresi *string homogen* dan *string heterogen* [4]. Sehingga dibutuhkanya sebuah analisa kedua algoritma yang memiliki tingkat kompresi file yang lebih efesien dari segi parameter hasil kompresi.

2. LANDASAN TEORI

2.1 Kompresi

Kompresi data adalah ilmu atau seni yang merepresentasikan informasi dalam bentuk yang lebih *compact* [5]. Istilah kompresi tersebut diterjemahkan dari kata bahasa Inggris "*compression*" yang berarti pemampatan. Dalam bidang teknik, kompresi berarti proses memampatkan sesuatu yang berukuran besar sehingga menjadi kecil. Dengan demikian, kompresi data berarti teknik untuk mengurangi ukuran data agar penyimpanannya jauh lebih padat dan juga untuk mengurangi waktu pengiriman data tersebut [6]. Kompresi data bertujuan untuk mengurangi jumlah bit yang digunakan untuk menyimpan atau mengirimkan informasi.

Tujuan daripada kompresi data tidak lain adalah untuk mengurangi data yang berlebihan (*Redundancy Data)* sehingga ukuran data menjadi lebih kecil dan lebih ringan sehingga mengurangi biaya untuk penyimpanan [7].

A. Pengelompokan Kompresi

Secara garis besar terdapat 2 buah penggolongan algoritma kompresi data yaitu kompresi *lossy* dan kompresi *lossless* [7].

1. Kompresi *Lossless* merupakan metoda kompresi data yang memungkinkan data asli dapat disusun kembali dari data hasil kompresi maka rasio kompresi pun tidak dapat terlalu besar untuk memastikan semua data dapat dikembalikan ke bentuk semula. Contoh metode ini adalah *Levenstein Code, Fibonacci Codes, Elias Gamma Code, Shannon-Fano Coding, Huffman Coding, Arithmetic Coding, Run Length Encoding* dan lain-lain.
2. Kompresi *Lossy* adalah suatu metode untuk mengkompresi dan mendekompresi data yang

dimana data setelah didekompresi mungkin berbeda dari data aslinya, tetapi perbedaan itu cukup dekat . Metode ini paling sering digunakan untuk kompres data multimedia (*Audio file* dan gambar). Format kompresi *Lossy* mengalami *generation loss* yaitu jika mengalami proses kompresi- dekompresi berulang kali maka akan menyebabkan kehilangan kualitas secara progresif. Contoh metode ini adalah *Transform Coding*, *Wavelet*, dan lain- lain.

B. Ukuran Kompresi

Ketika mengukur performa dari suatu algoritma kompresi, biasanya akan fokus pada efisiensi dari ruang penyimpanan dan efisiensi waktu termasuk dalam faktor lainnya [8]. Performa suatu algoritma kompresi juga bergantung pada tipe dan struktur dari sumber masukan, yang mana juga akan tergantung pada apakah termasuk dalam kategori kompresi *lossless* atau *lossy*. Untuk menilai dan mengetahui keefektifan suatu algoritma kompresi ataupun untuk mengetahui perbandingan *file* sebelum dan setlah di kompresi, maka diperlukan beberapa parameter kinerja kompresi yang harus diperhatikan sebagai berikut [8]. Ada beberapa faktor pembanding yang digunakan dalam penelitian ini, yaitu *Ratio of Compression* (RC), *Compression Ratio* (CR), *Space Saving*(SS), [9] :

1. *Ratio of Compression* (Rc)
   Ratio of Compression (Rc) adalah perbandingan antara ukuran data sebelum dikompresi dengan ukuran data setelah dikompresi .

$$Rc = \frac{\text{Ukuran Data Sebelum Kompresi}}{\text{Ukuran Data Setelah Kompresi.}} \quad (1)$$

2. *Compression Ratio* (Cr)
   Compression Ratio (Cr) adalah pers.entasi besar data yang telah dikompresi yang didapat dari hasil perbandingan antara ukuran data setelah dikompresi dengan ukuran data sebelum dikompresi.

$$Cr = \frac{\text{Ukuran Data Setelah Kompresi}}{\text{Ukuran Data Sebelum Kompresi}} \times 100\% \quad (2)$$

3. *Saving Percentages / Space Saving* (SS)
   Saving Percentages menghitung persentasi penyusutan dari data sebelum dilakukan kompresi.

$$Cr = \frac{\text{Ukr Data Sebelum-Ukr Data Setelah}}{\text{Ukuran Data Sebelum}} \times 100\% \quad (3)$$

2.3 Algoritma Elias Omega Code

Algoritma *Elias Omega Code* merupakan algoritma kompresi yang diperkenalkan oleh Peter Elias pada tahun 1975. Gagasan utama kode ini adalah untuk awalan integer dikodekan dengan representasi kode dari besarnya order dan inisialisasi kode secara rekursif. Algoritma *Elias Omega Code* mengurutkan karakter yang paling banyak muncul ke bit terkecil dan karakter yang langka ke bit terbesar. Dengan begitu, ukuran *file* dapat diminimalisir dari ukuran aslinya [10].

Encode dalam bilangan bulat positif *n* dilakukan dalam langkah- langkah berikut [8]:

1. Inisialisasikan *code_so_far* = 0.
2. Jika *n = 1*, maka *stop* (proses berhenti). Jika tidak, maka tambahkan representasi *binary* dari *n* ke *code_so_far*. Asumsikan jumlah *binary* bit sebagai L (panjang digit bit).

3. Ulangi langkah *b* hingga representasi *binary* dari L = 1 untuk menggunakan *n.*
4. Dimisalkan jika *n* = 17 dan *code_so_far* = 0. Ubahlah *n* ke dalam bentuk *binary* menjadi *n* = 17_{10} = 10001_2 dengan L = 5.

Tabel 1. Elias Omega Code

1	0	10	11 1010 0
2	10 0	11	11 1011 0
3	11 0	12	11 1100 0
4	10 100 0	13	11 1101 0
5	10 101 0	14	11 1110 0
6	10 110 0	15	11 1111 0
7	10 111 0	16	10 100 10000 0
8	11 1000 0	17	10 100 10001 0
9	11 1001 0	18	10 100 10010 0

Sumber : Salomon dan Giovanni, 2010

Kemudian untuk *Decode* maka dilakukan dalam langkah-langkah berikut [10] :

1. Inisialisasikan nilai *n* adalah 1.
2. Baca bit selanjutnya. Jika bernilai 0, maka *stop* (proses berhenti). Jika tidak, baca bit selanjutnya, dam kelompokan pada*n+1* bit untuk *n*, dan ulangi langkah ini
3. Pembacaan bit dilakukan berturut-turut, dapat dimulai hanya 0 atau diawali dengan 1 yang diikuti dengan *n* digit bit. Jika grup adalah 0, maka nilai integernya berupa *n;* yaitu 1. Dan jika grup dimulai dari 1, maka *n* menjadi nilai grup yang dipresentasikan ke dalam *binary.*

2.4 Algoritma Fibonacci Code

Leonardo Pisano Fibonacci adalah seorang ahli matematika Italia, Dia dianggap sebagai Matematikawan terbesar dari abad Pertengahan, Dia berperan penting dalam menghidupkan kembali matematika kuno. Dalam bukunya yang berjudul Liber Abaci, memperkenalkan Eropa dengan notasi Hindu-Arab untuk bilangan [11].

Leonardo Pisano Fibonacci lahir sekitar tahun 1170 dan meninggal sekitar tahun 1250 di Pisa, Italia. Dia menulis teks matematika, antara lain memperkenalkan Eropa dengan notasi Hindu- Arab untuk bilangan. Meskipun buku-bukunya harus ditulis dengan tangan, tetapi teks tentang natematika beredar luas.

Sekarang deret Fibonacci adalah salah satu objek matematika terkenal. Bilangan Fibonacci didefinisikan sebagai barisan bilangan yang suku-sukunya merupakan penjumlahan 2 suku sebelumnya. Bilangan Fibonacci dapat ditunjukkan sebagai barisan bilangan:
0,1,1,2,3,5,8,13,21,34,55,89,144,233,…
Langkah-langkah dalam pembuatan sebuah kode Fibonacci adalah sebagai berikut (wikipedia, 2020):

1. Tentukan sebuah bilangan bulat positif n yang lebih besar atau sama dengan 1
2. Temukan bilangan Fibonacci f terbesar yang lebih kecil atau sama dengan n, kurangkan nilai n dengan f dan catat sisa pengurangan nilai n dengan f.
3. Jika bilangan yang dikurangkan adalah bilangan yang terdapat dalam deret Fibonacci F(i), tambahkan angka “1” pada i-2 dalam kode Fibonacci yang akan dibentuk.
4. Ulangi langkah 2, tukar nilai n dengan sisa pengurangan nilai n dengan f sampai sisa pengurangan nilai n dengan f adalah 0.
5. Tambahkan angka “1” pada posisi paling kanan kode Fibonacci yang akan dibentuk

Adapun tabel nilai *fibonacci codes* dapat dilihat pada tabel 2 :

Tabel 2. Kode Nilai Fibonacci

Simbol	Kode Fibonacci
1	11
2	011
3	0011
4	1011
5	00011
6	10011
7	01011
8	000011
9	100011
10	010011
11	001011
12	101011
13	0000011
14	1000011

Sumber : Wikipedia, 2019

3. METODE PENELITIAN

3.1 Implementasi Algoritma Elias Omega Code

Adapun proses algoritma *Elias Omega Code* terdiri dari kompresi, perhitungan parameter kompresi dan proses dekompresi.

1. Kompresi Berdasarkan Algoritma *Elias Omega Code*
   Sebelum melakukan kompresi menggunakan algoritma *Elias Omega Code*, terlebih dahulu disusun berdasarkan frekuensi terbesar dan ditentukan banyaknya jumlah bit pada sampel karakter tersebut. Adapun prosesnya dapat dilihat pada tabel 3:

Tabel 3. Urutan Karakter Sesuai Banyaknya Frekuensi

No	*Karakter*	*BinnerKarakter*	*Bit*	*Frek*	*Bit * Frek*
1	A	01000001	8	4	32
2	N	01001110	8	2	16
3	I	01001001	8	2	16
4	R	01010010	8	2	16
5	S	01010011	8	2	16
6	U	01010101	8	1	8
7	V	01010110	8	1	8

Tabel 3. Urutan Karakter Sesuai Banyaknya Frekuensi (Lanjutan)

No	*Karakter*	*BinnerKarakter*	*Bit*	*Frek*	*Bit * Frek*
8	E	01000101	8	1	8
9	T	01010100	8	1	8
10	Spasi	00100000	8	1	8
11	H	01001000	8	1	8
12	P	01010000	8	1	8
Total					**152 Bit**

Seperti yang terlihat pada tabel 3, satu karakter terdiri dari 8 bit. Sehingga dengan jumlah karakter sebanyak 19 karakter dikalikan dengan 8 bit, maka total keseluruhan ukuran karakter adalah 152 bit. Setelah nilai bit didapatkan dari semua karakter, selanjutnya adalah melakukan kompresi menggunakan algoritma *Elias Omega Code*. kompresi dilakukan dengan mengalikan nilai frekuensi dari karakter yang muncul dengan nilai frekuensi bit dari algoritma *Elias Omega Codes*. Adapun untuk mendapat nilai bit algoritma *Elias Omega Codes* adalah sebagai berikut :

5. Inisialisasikan *code_so_far* = 0.
6. Jika *n = 1*, maka *stop* (proses berhenti). Jika tidak, maka tambahkan representasi *binary* dari *n* ke *code_so_far*. Asumsikan jumlah *binary* bit sebagai L (panjang digit bit).
7. Ulangi langkah *b* hingga representasi *binary* dari L = 1 untuk menggunakan *n.*
8. Dimisalkan jika *n* = 17 dan *code_so_far* = 0. Ubahlah *n* ke dalam bentuk *binary* menjadi $n = 17_{10} = 10001_2$ dengan L = 5.

Sehingga didapatkan kode nilai *Elias Omega Code* sesuai dengan jumlah karakter yang akan dikompresi seperti yang terlihat pada tabel 4:

Tabel 4. Nilai Algoritma *Elias Omega Code*

N	*Kode Elias*
1	0
2	10 0
3	11 0
4	10 100 0
5	10 101 0
6	10 110 0
7	10 111 0
8	11 1000 0
9	11 1001 0
10	11 1010 0
11	11 1011 0
12	11 1100 0

Selanjutnya dilakukan kompresi file teks yang sudah disusun seperti yang terlihat pada tabel 4 dengan mengalikan banyaknya frekuensi karakter yang muncul dengan banyaknya nilai bit dari kode nilai algoritma *Elias Omega Code*. Adapun proses keseluruhanya dapat dilihat pada tabel 5:

Tabel 5. Kompresi Algoritma *Elias Omega Code*

N	*Karakter*	*Elias Omega Codes*	*Jumlah Bit Omega*	*Frek Karakter*	*Bit * Frek*
1	A	0	1	4	4
2	N	10 0	3	2	6
3	I	11 0	3	2	6
4	R	10 100 0	6	2	12
5	S	10 101 0	6	2	12
6	U	10 110 0	6	1	6
7	V	10 111 0	6	1	6

Tabel 5. Kompresi Algoritma *Elias Omega Code* (Lanjutan)

N	*Karakter*	*Elias Omega Codes*	*Jumlah Bit Omega*	*Frek Karakter*	*Bit * Frek*
8	E	11 1000 0	7	1	7
9	T	11 1001 0	7	1	7
10	Spasi	11 1010 0	7	1	7
11	H	11 1011 0	7	1	7
12	P	11 1100 0	7	1	7
Total					**87 bit**

Seperti yang terlihat pada tabel 5, maka untuk mendapakan karakter baru hasil kompresi, digabungkan nilai kode *Elias Omega Code* yang berurutan dengan karakter sampel awal yaitu "UNIVERSITAS HARAPAN". Contohnya karakter "U" mendapatkan nilai kode *Elias Omega Code* pada tabel 5 yaitu "101100". Proses selanjutnya dilakukan dengan cara yang dapat dilihat seperti tabel 6:

Tabel 6. Bit Baru Kompresi Elias Omega Code

No	*Karakter*	*Kode Elias Omega Code*	*Jumlag Bit*
1	U	101100	6
2	N	100	3
3	I	110	3
4	V	101110	6
5	E	1110000	7
6	R	101000	6
7	S	101010	6
8	I	110	3
9	T	1110010	7
10	A	0	1
11	S	101010	6
12	Spasi	1110100	7
13	H	1110110	7
14	A	0	1
15	R	101000	6
16	A	0	1
17	P	1111000	7
18	A	0	1
19	N	100	3
Total			**87**

Kemudian gabungan setiap nilai bit kode *Elias Omega Code* dari atas ke bawah menjadi "10110010011010111011100001010001010101110011001010101110100111011001010000111100000100"

Jumlah bit yang dihasilkan adalah 87 bit. Proses selanjutnya adalah melakukan penambahan *padding* dan *flag bits*. Hal ini dilakukan karena jumlah bit hasil kompresi tidak habis dibagi 8 atau memiliki sisa. Sedangkan *flags b*it adalah nilai binner dari nilai angka *padding*. Karena jumlah bit hasil kompresi adalah 81 bit tidak habis dibagi 8 dan memiliki sisa 7 maka ditambahkan *padding* sebanyak kurangnya sisa yaitu 1 bit nilai binner "**0**". Dengan demikian, *flags bit*nya adalah nilai binner dari angka 1 (*padding*) yaitu "**00000001**". Sehingga terbentuk *string* baru hasil kompresi keseluruhanya adalah :
"10110010011010111011100001010001010101101110010010101011101001110110010100
00111100001000**00000001**"
Adapun total keseluruhan bit setelah ditambahkan *padding* dan *flag bits* adalah 87 bit + 1 bit (*padding*) + 8 bit (*flag bits*) = 96 bit
Berdasarkan hasil kompresi maka, didapatkan hasil kompresi algoritma *Elias Omega Code* berupa pengecilan ukuran file teks dari 152 bit menjadi 96 bit. Adapun hasil akhirnya setiap bit dipecah menjadi 8 bit kemudian dirubah kedalam bentuk karakter ASCII yang dapat dilihat pada tabel 7:

Tabel 7. Hasil Akhir Kompresi

No	*Binner*	*Nilai Desimal*	*Karakter*
1	10110010	178	²
2	01101011	107	K
3	10111000	184	¸
4	01010001	81	Q
5	01010110	86	V
6	11100100	228	Ä
7	10101011	171	«
8	10100111	167	§
9	01100101	101	E
10	00001111	15	SI (tidak Nampak)
11	0000100**0**	8	BS (tidak Nampak)
12	**00000001**	1	SOH (tidak Nampak)

2. Menghitung Parameter Hasil Kompresi Elias Omega Code
   Berdasarkan hasil kompresi dengan *Elias Omega Code* pada proses sebelumnya, adapun dapat dihitung proses kinerja kompresinya berupa parameter yaitu :
   a. *Ratio of Compression* (R_C)
      $Rc = \frac{Ukuran\ data\ sebelum\ dikompresi}{Ukuran\ data\ setelah\ dikompresi}$
      $Rc = \frac{152}{96}$
      Rc = 1,58
   b. *Compression Ratio* (C_R)
      $Cr = \frac{Ukuran\ data\ setelah\ dikompresi}{Ukuran\ data\ sebelum\ dikompresi}$x100%
      $Cr = \frac{96}{152}$x100
      Cr = 63,1 %
   c. *Space Saving* (S_s)

$$Ss = \frac{Ukuran\ data\ sebelum\ dikompresi - setelah\ kompresi}{Ukuran\ data\ sebelum\ dikompresi} x100\%$$

$$Ss = \frac{152-96}{152} x100$$

Ss = 36,84 %

3.2 Implementasi Algoritma Fibonacci Code

Adapun proses algoritma *Fibonacci Code* terdiri dari kompresi, perhitungan parameter kompresi dan proses dekompresi.

1. Kompresi Berdasarkan Algoritma *Fibonacci Code*
   Sebelum melakukan kompresi menggunakan algoritma *Fibonacci Code*, karakter yang sudah disusun pada tabel 3, terlebih dahulu disusun berdasarkan frekuensi terbesar dan ditentukan banyaknya jumlah bit pada sampel karakter tersebut. Adapun prosesnya dapat dilihat pada tabel 8:

Tabel 8. Urutan Karakter Sesuai Banyaknya Frekuensi

No	*Karakter*	*BinnerKarakter*	*Bit*	*Frek*	*Bit * Frek*
1	A	01000001	8	4	32
2	N	01001110	8	2	16
3	I	01001001	8	2	16
4	R	01010010	8	2	16
5	S	01010011	8	2	16
6	U	01010101	8	1	8
7	V	01010110	8	1	8
8	E	01000101	8	1	8
9	T	01010100	8	1	8
10	Spasi	00100000	8	1	8

Tabel 8. Urutan Karakter Sesuai Banyaknya Frekuensi (Lanjutan)

No	*Karakter*	*BinnerKarakter*	*Bit*	*Frek*	*Bit * Frek*
11	H	01001000	8	1	8
12	P	01010000	8	1	8
Total					**152 Bit**

Seperti yang terlihat pada tabel 8, satu karakter terdiri dari 8 bit. Sehingga dengan jumlah karakter sebanyak 19 karakter dikalikan dengan 8 bit, maka total keseluruhan ukuran karakter adalah 152 bit. Setelah nilai bit didapatkan dari semua karakter, selanjutnya adalah melakukan kompresi menggunakan algoritma *Fibonacci Code*. kompresi dilakukan dengan mengalikan nilai frekuensi dari karakter yang muncul dengan nilai frekuensi bit dari algoritma *Fibonacci Code*. Adapun untuk mendapat nilai bit algoritma *Fibonacci Code* adalah sebagai berikut :

a. Tentukan sebuah bilangan bulat positif n yang lebih besar atau sama dengan 1
b. Temukan bilangan *Fibonacci* f terbesar yang lebih kecil atau sama dengan n, kurangkan nilai n dengan f dan catat sisa pengurangan nilai n dengan f.
c. Jika bilangan yang dikurangkan adalah bilangan yang terdapat dalam deret *Fibonacci* F(i), tambahkan angka "1" pada i-2 dalam kode *Fibonacci* yang akan dibentuk.
d. Ulangi langkah 2, tukar nilai n dengan sisa pengurangan nilai n dengan f sampai sisa pengurangan nilai n dengan f adalah 0.
e. Tambahkan angka "1" pada posisi paling kanan kode *Fibonacci* yang akan dibentuk

Adapun tabel nilai *Fibonacci code* yang didapatkan dari sejumlah banyaknya *string* file digital sampel yang akan dikompresi seperti yang terlihat pada tabel 9:

Tabel 9. Kode Nilai *Fibonacci*

N	*Kode Fibonacci*
1	11
2	011
3	0011
4	1011
5	00011
6	10011
7	01011
8	000011
9	100011
10	010011
11	001011
12	101011

Selanjutnya dilakukan kompresi file teks yang sudah disusun seperti yang terlihat pada tabel 9 dengan mengalikan banyaknya frekuensi karakter yang muncul dengan banyaknya nilai bit dari kode nilai algoritma *Fibonacci Code*. Adapun proses keseluruhanya dapat dilihat pada tabel 10:

Tabel 10. Kompresi Berdasarkan Algoritma *Fibonacci Code*

N	*Karakter*	*Fibonacci Code*	*Jumlah Bit Fibonacii*	*Frekuensi Karakter*	*Bit * Frek*
1	A	11	2	4	8
2	N	011	3	2	6
3	I	0011	4	2	8
4	R	1011	4	2	8
5	S	00011	5	2	10
6	U	10011	5	1	5
7	V	01011	5	1	5
8	E	000011	6	1	6
9	T	100011	6	1	6
10	Spasi	010011	6	1	6
11	H	001011	6	1	6
12	P	101011	6	1	6
Total					**80 bit**

Seperti yang terlihat pada proses kompresi menggunakan algoritma *Fibonacci Code* tabel 10, maka untuk mendapakan karakter baru hasil kompresi, digabungkan nilai kode *Fibonacci Code* yang berurutan dengan karakter sampel awal yaitu "UNIVERSITAS HARAPAN". Contohnya karakter "U" mendapatkan nilai kode *Fibonacci Code* pada tabel 10 yaitu "10011". Proses selanjutnya dilakukan dengan cara yang dapat dilihat seperti pada tabel 11:

Tabel 11. Susunan Bit Baru Kompresi Fibonacci Code

No	*Karakter*	*Kode Fibonacci Code*	*Jumlag Bit*

1	U	10011	5
2	N	011	3
3	I	0011	4
4	V	01011	5
5	E	000011	6
6	R	1011	4
7	S	00011	5
8	I	0011	4
9	T	100011	6
10	A	11	2
11	S	00011	5
12	Spasi	010011	6
13	H	001011	6
14	A	11	2
15	R	1011	4
16	A	11	2
17	P	101011	6
18	A	11	2
19	N	011	3
Total			**80**

Kemudian gabungan setiap nilai bit kode *Fibonacci Code* dari atas ke bawah menjadi :
"1001101100110101100001110110001100111000111100011010011001011111011111010 1111011"
Jumlah bit yang dihasilkan adalah 80 bit. Proses selanjutnya adalah melakukan penambahan *padding* dan *flag bits*. Karena jumlah bit hasil kompresi *fibonacci* 80 habis dibagi 8 dan tidak memiliki maka tidak perlu ditambahkan *padding* tetapi tetap ditambahkan *flags bit* yang berfungsi untuk membaca untuk proses dekompresi. Berhubungan padding tidak ada, maka flag bitnya adalah menambahkan 8 bit diakhir dengan nilai 0**.** Sehingga *string* bit yang terbentuk keseluruhanya adalah :
"1001101100110101100001110110001100111000111100011010011001011111 0111110101111011**00000000**"
Berdasarkan hasil kompresi maka, didapatkan hasil kompresi algoritma *Fibonacci Code* berupa pengecilan ukuran file teks dari 152 bit menjadi:
80 + 8 bit = 88 bit Adapun hasil akhirnya setiap bit dipecah menjadi 8 bit kemudian dirubah kedalam bentuk karakter ASCII yang dapat dilihat pada tabel 12:

Tabel 12. Hasil Kompresi Fibonacci

No	*Binner*	*Nilai Desimal*	*Karakter*
1	10011011	155	>
2	00110101	53	5
3	10000111	135	‡
4	01100011	99	C
5	00111000	56	8
6	11110001	241	Ñ
7	10100110	166	¦

8	01011111	95	_
9	01111101	125	}
10	01111011	123	{
11	00000000	0	

2. Menghitung Parameter Hasil Kompresi Fibonacci Code
   Berdasarkan hasil kompresi dengan *Fibonacci Code* pada proses sebelumnya, adapun dapat dihitung proses kinerja kompresinya berupa parameter yaitu :
   a. *Ratio of Compression* (R_C)
      $Rc = \frac{Ukuran\ data\ sebelum\ dikompresi}{Ukuran\ data\ setelah\ dikompresi}$
      $Rc = \frac{152}{88}$
      Rc = 1,72
   b. *Compression Ratio* (C_R)
      $Cr = \frac{Ukuran\ data\ setelah\ dikompresi}{Ukuran\ data\ sebelum\ dikompresi}$x100%
      $Cr = \frac{88}{152}$x100
      Cr = 57,8 %
   c. *Space Saving* (S_s)
      $Ss = \frac{Ukuran\ data\ sebelum\ dikompresi - setelah\ kompresi}{Ukuran\ data\ sebelum\ dikompresi}$x100%
      $Ss = \frac{152-80}{152}$x100
      Ss = 42,1 %

4. HASIL DAN PEMBAHASAN

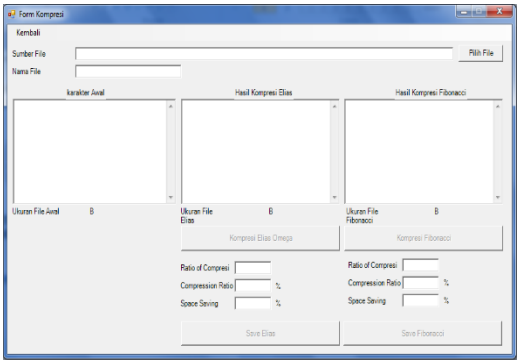
4.1 Implementasi Sistem

Tujuan dari pengujian ini adalah untuk mengetahui apakah aplikasi yang telah dibuat sesuai dengan perancanganya. Selain itu juga, untuk mengetahui detail dari jalanya aplikasi serta kesalahan yang ada untuk jadikan pengembangan dan perbaikan lanjut. Proses pengujian ini dibutuhkan beberapa peralatan-peralatan berupa perangkat keras dan perangkat lunak.

4.2 Pengujian Sistem

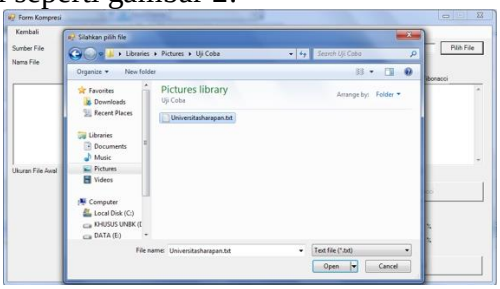
Adapun proses *user* melakukan analisis parameter kompresi *file* digital menggunakan algoritma *Elias Omega Code* dan *Fibonacci Code* terdiri dari *user* melakukan pencarian *file* digital, kemudian melakukan kompresi menggunakan algoritma *Elias Omega Code* dan didapatkan hasil parameter, kemudian user melakukan kompresi kembali dengan algoritma *Fibonacci Code* sehingga didapatkan parameter kedua. Dari hasil uji kompresi kedua algoritma didapatkan perbandingan parameter. Adapun proses tersebut dapat dilihat pada gambar di bawah ini :

1. Memilih *File* Digital
   Proses pertama yang dilakukan user didalam *form* kompresi adalah melakukan pemilihan *file* digital dengan menekan *button* Pilih *file* seperti gambar 1:



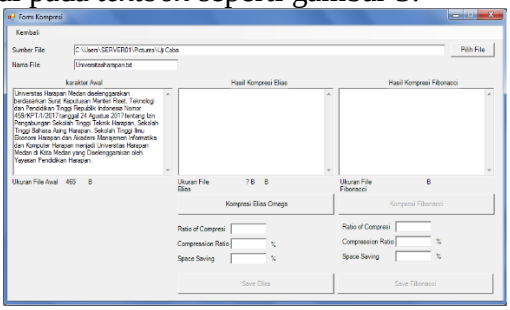
Gambar 1. Proses Memilih *File* Digital

Seperti yang terlihat pada gambar 1, klik *button* pilih *file* untuk menampilkan *form* pencarian *file* digital pada didirektori seperti gambar 2:



Gambar 2. Tampilan *Form* Pencarian *File* digital

User hanya memilih *file* digital yang akan dikompresi dan klik *button open*, sehingga akan tampil informasi *file* digital pada *textbox* seperti gambar 3:

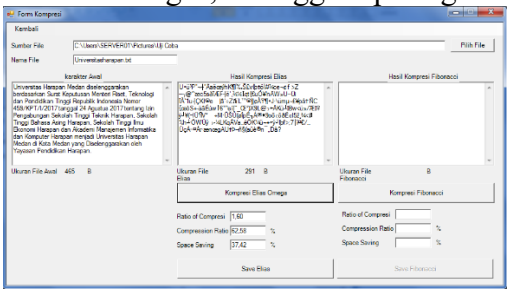


Gambar 3. Informasi *File* Digital

Seperti yang terlihat pada gambar 3, didapatkan *string* awal *file* digital dengan ukuran 455 *Byte*.

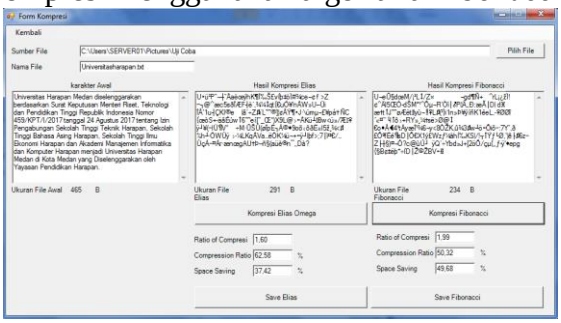
2. Proses Kompresi *File* Digital

Seperti yang terlihat pada gamber 3, untuk membandingan parameter hasil kompresi, pertama *user* melakukan kompresi menggunakan algoritma *Elias Omega Code* dengan menekan *button* "Kompresi Elias Omega", sehingga seperti gambar 4:



Gambar 4. Proses Kompresi *Elias*

Seperti yang terlihat pada gambar 4 proses kompresi pertama dilakukan dengan algoritma *Elias Omega Code*, sehingga menghasilkan *string* baru dengan ukuran yang lebih kecil dari ukuran semula yaitu 291 *Byte* dengan parameter sebagai berikut:
RC = 1.60
CR = 62.58%
SS = 37.42%.
Kemudian dilanjutkan kompresi menggunakan algoritma *Fibonacci Code* seperti gambar 5:



Gambar 5. Proses Kompresi *Fibonacci*

Seperti yang terlihat pada gambar 5 proses kompresi kedua dilakukan dengan algoritma *Fibonacci Code*, sehingga menghasilkan *string* baru dengan ukuran yang lebih kecil dari ukuran semula yaitu 234 *Byte* dengan parameter sebegai berikut:
RC = 1.99
CR = 50.32%
SS = 49.68%.

5.1 Kesimpulan

Berdasarkan hasil dari implementasi sistem yang telah dilakukan pada bab sebelumnya, dapat diambil kesimpulan bahwa:

1. Kompresi dan dekompresi menggunakan algoritma *Elias Omega Code* dan *Fibonacci Codes* berhasil dilakukan terhadap *file* digital dengan membaca setiap *string file* digital.
2. Berdasarkan analisis parameter kompresi pada penelitian, didapatkan bahwa algoritma *Fibonacci* lebih baik dalam mengkompress *file* digital dari pada algoritma *Elias Omega Code* dengan presentasi perbandingan *space saving* yang lebih besar dengan perbandingan 36,84% untuk *Elias* dan 42,1% untuk *Fibonacci*.
3. Perancangan aplikasi kompresi berhasil menerapkan algoritma *Elias Omega Code* dan *Fibonacci Code* untuk kompresi *file* digital dengan mendapatkan ukuran hasil kompresi *file* digital rata-rata sebesar < 50 %

DAFTAR PUSTAKA

[1] Tetti. P.S, dkk,"Penerapan Algoritma Levenstein Pada Aplikasi Kompresi File MP3",KOMIK, vol.2, No.1, pp.334-342, 2018
[2] Michael. S,"Perbandingan Algoritma Elias Delta Code dan Unary Coding Dalam Kompresi Citra Forensik"djuornal, Vol.1, No.1, pp.18-26, 2020
[3] Marlina. L, dkk, "*Data Compression Using Elias Delta Code*" *International Journal of Recent Trends in Engineering & Research (IJRTER)*, Vol. 3, pp.210-217, 2017
[4] Subada. M.A," Analisis Perbandingan Algoritma Even-Rodeh Code dan Algoritma Fibonacci Code untuk Kompresi File Teks", Skripsi, 2018
[5] Shanmusgasundaram, S dan Lourdusamy. R,"*A Comparative Study Of Text Compression Algorithms. International Journal of Wisdom Based Computing*", Vol. 1 (3), pp.68-76, 2011.

[6] Budiman. M.A dan Rachmawati. D, "*On Using Goldbach GO Codes and EvenRodeh Codes for Text Compression*". *Material Science and Engineering* 180, 2017

[7] Rani. M dan Singh. V, "*An Enhanced Text Compression System Based on ASCII Values and Huffman Coding*". *International Journal of Computer Science Trends and Technology* (IJCST), Vol.4, Issue 3 , 2016.

[8] Kodituwakku. S.R,"*Comparison Of Lossless Data Compression Algorithms For Text Data*". *Indian Journal of Computer Science and Engineering,* Vol 1, pp.416-426, 2011

[9] Salomon. D. 2007. *Variable-Length Codes for Data Compression*. Springer: USA.

[10]Wikipedia, (2019, Jun28), Fibonacci Coding [online] Available: https://en.wikipedia.org/wiki/Fibonacci_coding.