E-ISSN:2540-9719
Jurnal SISTEMASI, Volume 8, Nomor 1 Januari 2019 : 104 – 113 ISSN:2302-8149

KOMPRESI DATA TEKS DENGAN MENGGUNAKAN ALGORITMA SEQUITUR

[1]Yulia Darnita, [2]Khairunnisyah, [3]Husni Mubarak
[123]Prodi Teknik Informatika Fakultas TeknikUniversitas Muhammadiyah Bengkulu
Jl. Bali Kota Bengkulu, telp (0736) 22765/fax (0736) 26161
Email: yuliadarnita@umb.ac.id, khairunnisyah@umb.ac.id, Husnimubarak@gmail.com

ABSTRAK

Data merupakan salah satu hal utama yang dikaji dalam masalah teknik ilmu komputer (TIK), data bisa berhujut suatu keadaan, gambar, suara, huruf, angka, matematika, bahasa ataupun simbol-simbol lainnya fakta yang sering terjadi tentang data adalah kebutuhan akan kapasitas penyimpanan dan kebutuhan waktu transfer data, yang menjadi kasus yang harus di amati dalam kebutuhan ini disebabkan oleh data yang harus disimpan kedalam memori RAM semakin lama bertambah banyak, oleh karena itu di butuhkan media penyimpanan yang besar. Algoritma sequitur merupakan sebuah algoritma waktu linier yang menyimpulkan tata bahasa bebas konteks (*context-free gramntar*) ke dalam suatu pemampatan untuk mengurangi masukan yang berulang atau dengan kata lain melakukan pengelompokkan karakter yang sama pada isi file dan mengatasi permasalahan dalam pemampatan sebuah data, bersifat *Lossy compression*. Berdasarkan Hasil Pengujian Yang dilakukan :File hasil kompresi oleh *algoritmasequitur* memudahkan dalam menggunakan internet sehingga waktu yang diperlukan akan menjadi lebih pendek dan kemungkinan pekerjaan *Download* dan *Upload* gagal akan menjadi lebih kecil. Kemudian teransfer *file* melalui jaringan akan lebih cepat , waktu pengiriman tergantung dari *provider* yangcepat atau tidak serta ukuran *file yang akan dikirim dan* membantu dalam mengurangi ukuran dari *file* sehingga dapat mengurangi kapasitas penyimpanan suatu memori */ RAM.*

Keywords: algoritma sequitur ,data, kompresi, *tranfer*

1 PENDAHULUAN

Data merupakan salah satu hal utama yang dikaji dalam masalah teknik ilmu komputer (TIK).Data adalah sesuatu yang belum mempunyai arti bagi penerimanya dan masih memerlukan adanya suatu pengolahan.Data bisa berhujut suatu keadaan, gambar, suara, huruf, angka, matematika, bahasa ataupun simbol-simbol lainnya yang bisa kita gunakan sebagai bahan untuk melihat lingkungan, obyek, kejadian ataupun suatu konsep.

Penggunaan dan pemanfaatan data sudah mencakup banyak aspek. Data menggambarkan sebuah representasi fakta yang tersusun secara terstruktur, dengan kata lain bahwa *umumnya, data yang mewakili kodifikasi terstruktur entitas primer tunggal, serta transaksi yang melibatkan dua atau lebih entitas utama*. (Vercellis : 2009)

Berdasarkan fakta yang sering terjadi tentang data adalah kebutuhan akan kapasitas penyimpanan dan kebutuhan waktu transfer data, yang menjadi kasus yang harus di amati dalam kebutuhan ini disebabkan oleh data yang harus disimpan kedalam memori RAM semakin lama bertambah banyak, oleh karena itu di butuhkan media penyimpanan yang besar. tetapi apabila data-data tersebut bisa di kompres sehingga menjadi lebih kecil dari ukuran yang sebenarnya tanpa mengurangi isi di dalam data tersebut akan sangat membantu, Selama ini kompresi data yang di lakukan terhadap data teks bersifat *Lossy compression* atau menghilangkan karakter huruf dan angka di dalam sebuah data saat di dekompresikan, alangkah baiknya jika kompresi yang di lakukan bersifat *Lossless compression* dengan kata lain tidak menghilangkan karakter huruf atau angka yang terdapat di dalam data.(Salomon : 2007).

Menurut Nevill-Manning dan Ian Witlen (1997), sequitur merupakan sebuah algoritma waktu linier yang menyimpulkan tata bahasa bebas konteks (*context-free gramntar*) ke dalam suatu pemampatan untuk mengurangi masukan yang berulang atau dengan kata lain melakukan pengelompokkan karakter yang sama pada isi file. Maka dalam hal ini algoritma sequitur perlu di coba untuk mengatasi permasalahan dalam pemampatan sebuah data.

Dalam pengolahan kompresi data teks kita bisa menggunakan algoritma yang ada sebagai alat bantu kompresi data, berdasarkan karakter algoritma tentu memiliki cara kerja yang berbeda dari

algoritma lainnya, tergantung pada fungsi kegunaan, serta pemilihan algoritma yang benar dan tepat, seperti Algoritma Sequitur dalam bidang ilmu komputer di gunakan sebagai algoritma kompresi pada data teks yang bersifat *Lossless compression.(* Cormen : 2001*).*

2 TINJAUAN PUSTAKA

2.1 Kompresi Data

Pendapat para ahli mengatakan bahwa data kompresi adalah proses mengkonversikan sebuah input data *stream* (sumber, atau data mentah asli) menjadi data *stream* lainnya (bit *stream* hasil, atau *stream* yang telah terkompresi) yang berukuran lebih kecil. Kompresi data adalah proses yang dapat mengubah sebuah aliran data masukan (sumber atau data asli) ke dalam aliran data yang lain (keluaran atau data yang dimampatkan) yang memiliki ukuran yang lebih kecil*Data adalah* catatan atas kumpulan fakta. Data merupakan bentuk jamak dari datum, berasal dari bahasa Latin yang berarti "sesuatu yang diberikan".Data bisa berujut suatu keadaan, Teks, gambar, suara, ataupun simbol-simbol lainnya yang bisa kita gunakan sebagai bahan untuk melihat lingkungan, obyek, kejadian ataupun suatu konsep(David : 2007*).*

Kompresi data adalah ilmu atau seni merepresentasikan informasi dalam bentuk yang lebih compact,. David Salomon mengatakan bahwa data kompresi adalah proses mengkonversikan sebuah input data stream (stream sumber, atau data mentah asli) menjadi data stream lainnya (bit stream hasil, atau stream yang telah terkompresi) yang berukuran lebih kecil.[5] Pemampatan merupakan salah satu dari bidang teori informasi yang bertujuan untuk menghilangkan redundansi dari sumber. Pemampatan bermanfaat dalam membantu mengurangi konsumsi sumber daya yang mahal, seperti ruang hard disk atau perpindahan data melalui internet(Yudistira Yoga Aji,Eko Darwiyanto,Gia Septiana, 2016).

Tujuan dari kompresi data adalah untuk merepresentasikan suatu data digital dengan sesedikit mungkin bit, tetapi tetap mempertahankan kebutuhan minimum untuk membentuk kembali data aslinya. Data digital ini dapat berupa text, gambar, suara, dan kombinasi dari ketiganya, seperti video.Untuk membuatsuatu data menjadi lebih kecil ukurannya daripada data asli, diperlukan tahapan-tahapan (algoritma) untuk mengolah data tersebut. Menurut Thomas H. Cormen algoritma adalah suatu prosedur komputasi yang didefinisikan secara baik, membutuhkan sebuah atau sekumpulan nilai sebagai input, dan menghasilkan sebuah atau sekumpulan nilai sebagai output.

Dalam algoritma kompresi data, tidak ada algoritma yang cocok untuk semua jenis data. Hal ini disebabkan karena data yang akan dimampatkan harus dianalisis terlebih dahulu, dan berharap menemukan pola tertentu yang dapat digunakan untuk memperoleh data dalam ukuran yang lebih kecil. Karena itu, muncul banyak algoritma-algoritma kompresi data.

a. Kompresi Lossy Kompresi data yang menghasilkan file data hasil kompresi yang tidak dapat dikembalikan menjadi file data sebelum dikompresi secara utuh. Ketika data hasil kompresi di-decode kembali, data hasil decoding tersebut tidak dapat dikembalikan menjadi sama dengan data asli tetapi ada bagian data yang hilang.
b. Kompresi Lossless Kompresi data yang menghasilkan file data hasil kompresi yang dapat dikembalikan menjadi file data asli sebelum dikompresi secara utuh tanpa perubahan apapun. Kompresi jenis ini ideal untuk kompresi teks.Algoritma yang termasuk dalam metode kompresi lossless diantaranya adalah dictionary coding dan huffman coding.

Proses kompresi data didasarkan pada kenyataan bahwa pada hampir semua jenis data selalu terdapat pengulangan pada komponen data yang dimilikinya, misalnya didalam suatu text kalimat akan terdapat pengulangan penggunaan huruf alphabet dari huruf a sampai dengan huruf z. Kompresi data melalui proses encoding berusaha untuk menghilangkan unsur pengulangan ini dengan mengubahnya sedemikian rupa sehingga ukuran data menjadi lebih kecil, baik untuk berbagai jenis data seperti data teks, gambar, video, suara, dan lain-lain. Proses pengurangan unsur pengulangan ini dapat dilakukan dengan memakai beberapa teknik kompresi. Misalnya jika suatu komponen muncul berulang kali dalam suatu data, maka komponen tersebut tidak harus dikodekan berulang kali pula tapi dapat dikodekan dengan menulis frekuensi munculnya komponen dan di mana komponen tersebut muncul.Teknik kompresi data lainnya, berusaha untuk mencari suatu bentuk kode yang lebih pendek untuk suatu komponen yang sering muncul.

Keberhasilan pengkompresian data tergantung dari besarnya data itu sendiri dan tipe data yang memungkinkan untuk dikompresi Biasanya beberapa komponen-komponen di dalam data yang sifatnya

lebih umum dari yang lainnya banyak dipakai pada algoritma kompresi data yang memanfaatkan sifat ini.Hal ini dinamakan redudancy.Makin besar redudancy di dalam data makin tinggi pula tingkat keberhasilan kompresi data. Dalam proses kompresi data, terdapat konsep umum probabilitas yang menunjukkan suatu ukuran berapa banyak informasi yang terdapat dalam suatu rangkaian data atau yang disebut dengan entropy yang dapat direpresentasikan secara matematis(Ferrianto Gozali & Mervyn, 2004).

2.2 Algoritma

Dalam matematika dan komputasi algoritma merupakan kumpulan perintah yang saling berkaitan untuk menyelesaikan suatu masalah.Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir.Dalam penyusunannya diperlukan urutan serta logika agar algoritma yang dihasilkan sesuai dengan yang diharapkan.Algoritma merupakan bagian terpenting yang tidak dapat dipisahkan dari pemrograman. Meskipun sintaksis dan semantik yang dibuat benar adanya, dengan algoritma yang keliru, permasalahan yang ingin dipecahkan dengan teknik pemrograman tidak akan berhasil. Oleh karena itu, sebelum membuat program aplikasi, hal pertama yang harus kita pahami algoritma atau prosedur pemecahannya. Hal ini bertujuan agar program yang telah dibuat dapat sesuai dengan yang diharapkan (Ramadhani, Cipta. 2015)

2.2.1 Algoritma Kompresi Sejenis

A. Algoritma LZW (Lempel-Ziv-Welch)

Algortima *LZW* sering digunakan dalam pengkompresi data image *Jpeg*, algoritma ini menggunakan teknik dictionary dalam kompresinya. Dimana string karakter digantikan oleh kode table yang dibuat setiap ada string yang masuk. Tabel dibuat untuk referensi masukan string selanjutnya. Ukuran tabel dictionary pada algoritma *LZW* asli adalah *4096* sampel atau *12 bit*, dimana *256* sampel pertama digunakan untuk table karakter single (*Extended ASCII),* dan sisanya digunakan untuk pasangan karakter atau string dalam data input. Algoritma *LZW* melakukan kompresi dengan mengunakan kode table *256* hingga *4095* untuk mengkodekan pasangan byte atau string. Dengan metode ini banyak string yang dapat dikodekan dengan mengacu pada string yang telah muncul sebelumnya dalam teks.(Abraham dan Jacob : 1977 *)*

B. Algoritma Huffman

Algoritma *Huffman* adalah salah satu algoritma kompresi tertua.Algoritma tersebut digunakan untuk membuat kompresi jenis *lossycompression,* yaitu pemampatan data dimana tidak satu byte pun hilang sehingga data tersebut utuh dan disimpan sesuai dengan aslinya. Algoritma *Huffman* menggunakan prinsip pengkodean yang mirip dengan kode Morse, yaitu tiap karakter (simbol) dikodekan hanya dengan rangkaian beberapa bit, dimana karakter yang sering muncul dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang muncul dikodekan dengan rangkaian bit yang lebih panjang.*(Huffman : 1952*).

2.2.2 Algoritma Sequitur

Menurut *Nevill-Manning dan Ian Witlen* (*1997*), *Sequitur*merupakan sebuah algoritma waktu linier yang menyimpulkan tata bahasa bebas konteks (*context-free grammaar*) ke dalam suatu pemampatan untuk mengurangi masukan yang berulang. Operasi *sequitur* terdiri dari pemastian dua sifat yang berlaku. Ketika menjelaskan algoritma, sifat-sifatnya bertindak sebagai batasan. Algoritma beroperasi menjalankan batasan didalam sebuah tata bahasa yang ketika digram keunikan (diagram uniqueness) dilanggar, sebuah aturan baru dibentuk, dan ketika batasan aturan kegunaan (*Rule utility*) dilanggar, aturan yang sia-sia dihapus . (*Ervin, Kompresi Data Teks Menggunakan Pendekatan Grammar Compression Dengan Algoritma Sequitur, 2011*). Berikut ini menguraikan bagaimana dua batasan ini terjadi:

1. Diagram Keunikan (*Diagram Uniqueness*). Diagram uniqueness mempunyai arti bahwa tidak ada pasangan dari simbol atau diagram muncul lebih dari sekali dalam sebuah tata bahasa. Jika hal ini terjadi maka akan melanggar aturan batasan pertama (*diagram uniqueness*) sehingga akan membentuk aturan baru (*simbol non-terminal*) yang akan menggantikan simbol atau diagram yang muncul lebih dari sekali.

2. Aturan Kegunaan (*Rule Utility*). Rule utility mempunyai arti bahwa setiap aturan produksi digunakan lebih dari sekali. Dan jika ada aturan yang hanya digunakan sekali maka akan terjadi pelanggaran pada batasan kedua (*rule utility*) sehingga aturan yang hanya dipakai sekali akan dihapus atau dihilangkan (Ervin, 2011).

Sequitur merupakan sebuah algoritma waktu linier yang menyimpulkan tata bahasa bebas konteks (*context-free grammar*) ke dalam suatu pemampatan untuk mengurangi masukan yang berulang, ketika menjelaskan algoritma sifat-sifatnya bertindak sebagai batasan. Algoritma ini sendiri beroperasi menjalankan batasan didalam sebuah tata bahasa yang ketika digram keunikan (*digram uniqueness*) dilanggar, sebuah aturan baru dibentuk, dan ketika batasan aturan kegunaan *Rule utility* di langgar aturan yang sia-sia dihapus.

Sedangkan algoritma Sequitur beroperasi dengan menjalankan batasan diagram keunikan (*Diagram uniqueness*) dan aturan kegunaan (*Rule utility*). Beberapa batasan pelanggaran adalah sangat penting untuk dideteksi secara efisien.

Di bawah ini merupakan algoritma *s* :

Masukkan karakter pertama s untuk membuat hasil dari $S \rightarrow abcababcd$;

Mengulang

Mencocokkan pada non-terminal yang ada:

$S \rightarrow AcAab \quad \rightarrow \quad A \rightarrow ab$

$S \rightarrow AcAa$

Membuat sebuah non-terminal baru :

$S \rightarrow AcAAc \quad A \rightarrow ab$

$\rightarrow \quad B \rightarrow Ac$

Memindahkan non-terminal :

$S \rightarrow AcAAc \quad A \rightarrow ab$

$S \rightarrow BAB \quad \rightarrow \quad B \rightarrow Ac$

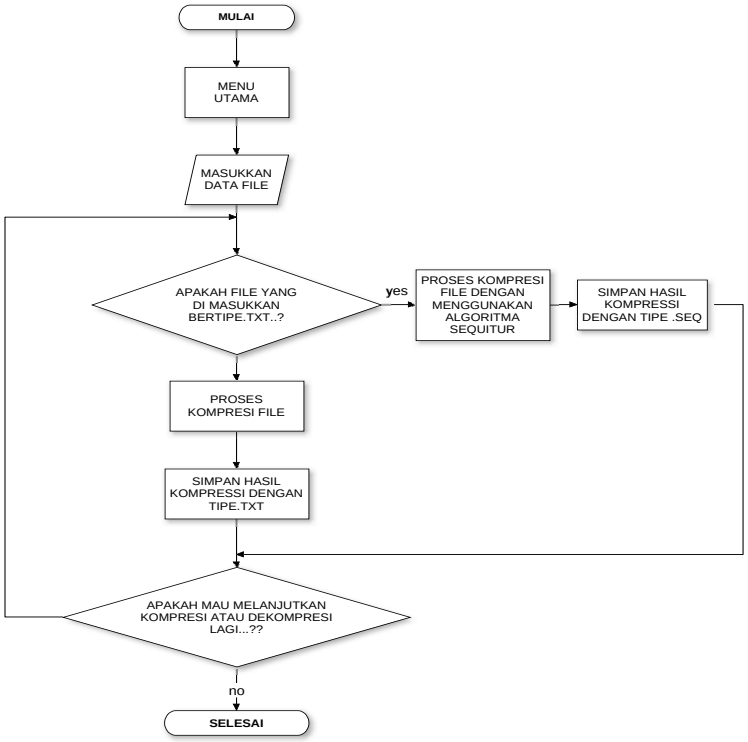
Memasukkan karakter baru :

$S \rightarrow BABd$

Sampai tidak ada karakter yang tersisa (*Ervin, 2011*)

3 METODOLOGI PENELITIAN

3.1 Flowchart Algoritma Sequiture



Gambar 3.1 *Flowchat* Aplikasi Kompressi data teks dengan menggunakan algoritma Sequiture

Berikut adalah Proses kompresi :

1. Jika file sumber telah ditentukan dan file tujuan telah ditetapkan maka proses yang pertama yaitu mengkompress file sumber dengan algoritma sequitur tersebut.
2. Kemudian file sumber tersebut huruf yang sama sebanyak dua buah kodekan dengan *A* .
3. Lalu dilanjutkan dengan membuat pengkodean baru (misal *"B"*) untuk dua buah karakter yang sama dan dilanjutkan seperti langkah awal sebelumnya
4. Kemudian dilanjutkan dengan proses pengkodean isi file yang akan dikompressi dengan mencari kesamaan 2 buah karakter dan mengkodekannya berturut setelah pengkodean terakhir. Hal ini terus-menerus dilakukan secara teratur sampai seluruh isi file habis dibaca dan apabila ditemukan "Pengkodean" maka disimpan untuk nantinya dikonversikan pada proses Dekompressi file sumber.

Untuk mengkompresi sebuah data teks yang mengandung kata *abcccdccd*, langkah yang dilakukan oleh algoritma Sequituradalah sebagai berikut :

1. Masukkan karakter $S \rightarrow abcccdccd$
2. Kodekan $A \rightarrow cc$
3. Mencocokkan pengkodean dengan data :
   $S \rightarrow abcccdcc$
   $A \rightarrow cc$
4. Mengkodekan A ke dalam terminal :
   $S \rightarrow abAdA$
5. Memasukkan karakter baru :
   $S \rightarrow abAdAd$
6. Kodekan
   $B \rightarrow Ad$
7. Mencocokkan pengkodean dengan data :
   $S \rightarrow abAdAd$
   $B \rightarrow$ Ad
8. Mengkodekan *B* ke dalam terminal :
   $S \rightarrow abBB$

Maka, diperolehlah hasil output *abBB*, yang mana akan digunakan oleh metode ini untuk mengkompresi ataupun medekompresi data teks tersebut. Berikut adalah Dekompressi :

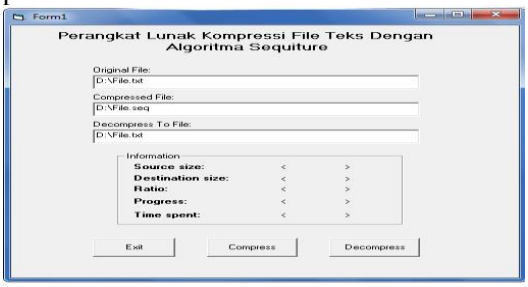
1. Pada Dekompressi algoritma sequitur ini file sumber akan dibaca dan dilakukan proses pengembalian ke struktur asal file dengan cara membaca kode-kode yang ada dan menterjemahkannya.
2. File atau teks tersebut dibaca dan langsung diproses dengan menelusuri setiap karakter yang ada dan mencocokkan dengan kode yang telah ada.
3. Proses ini terus berlangsung sampai semua file terlewati dan karakter yang sesuai dengan kode-kode yang tersedia akan berubah atau terkonversi kedalam bentuk file asli atau belum mengalami proses kompressi.

4 HASIL DAN PEMBAHASAN

4.1 Hasil

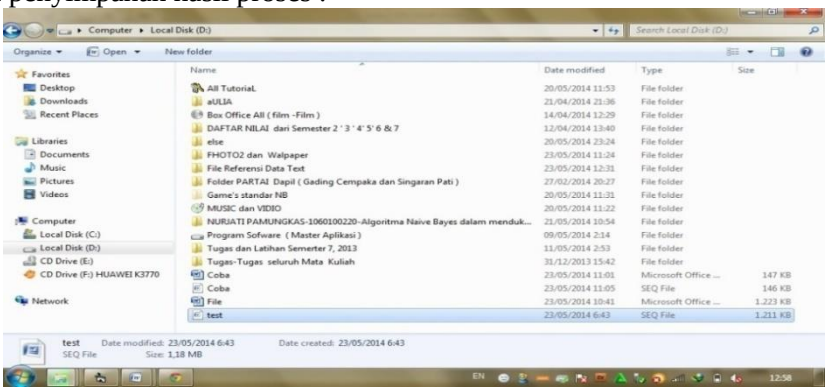
A. Menu Utama

Halaman utama merupakan tampilan tatap muka yang dapat diakses pengguna dari aplikasi kompresi teks ini dengan menggunakan algoritma sequitur guna mengurangi ukuran file yang akan dikompresikan dan di dekompresikan .



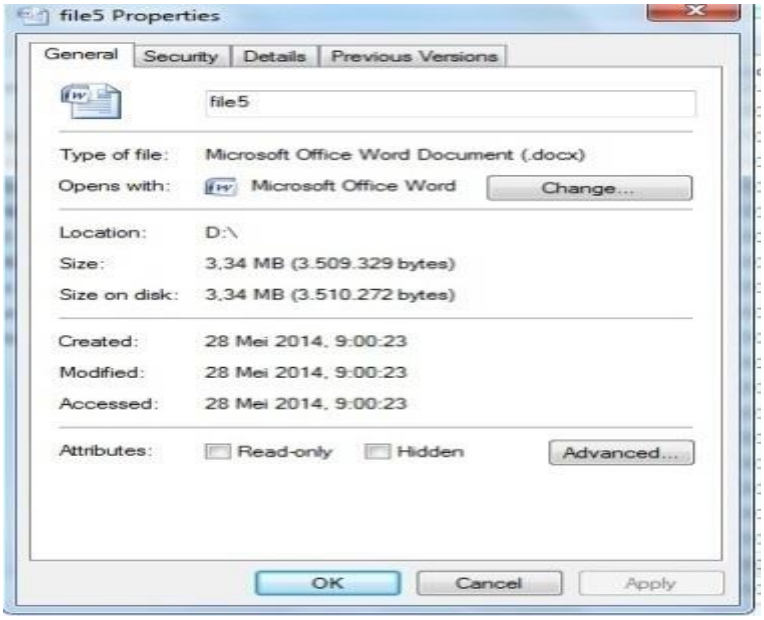
Gambar 4.1. Menu Utama Aplikasi Kompresi

Kemudian menentukan alamat *directory* dan nama *file* yang akan diproses dan menentukan *directory* tujuan penyimpanan hasil proses .



Gambar 4.2. *Directory Lokasi Penyimpanan File*

Lalu melihat kapasitas ukuran awal *file* pada Properties guna untuk memantau ukuran semula sebelum di kompresikan.



Gambar 4.3. Properties Awal file5.docx

B. Menu Kompresi

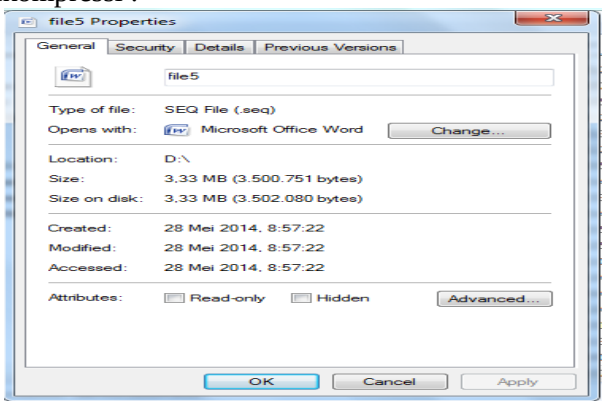
Kemudian mulai melakukan proses kompresian pada *file* dengan menuliskan alamat *directory* dan nama *file* yang akan diproses.

Gambar 4.4. Form Halaman Proses Kompresi

Kompressi berlangsung dengan intruksi / pengkodean algoritma sequitur yang di panggil tombol Compress *Command button1*. Kemudian menunggu proses Kompresi berlangsung sampai pemberitahuan muncul.

Gambar 4.5. Pemberitahuan Proses Kompresi Selesai

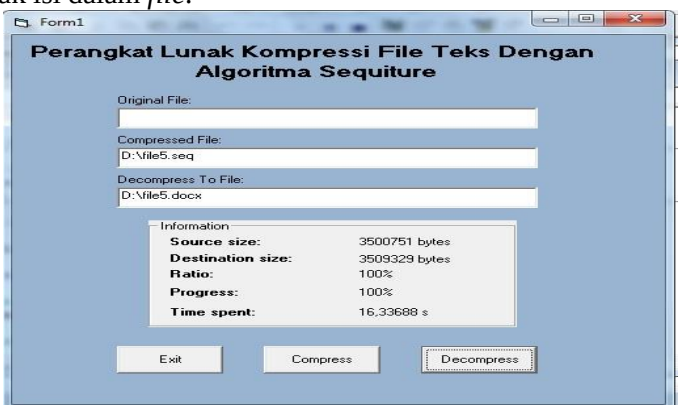
Kemudian melihat hasil kompressi dengan cara melihat ukuran pada *properties*. Hasilnya ukuran awal berkurang setelah dikompressi .



Gambar 4.6.Properties File5.Seq Hasil Kompresi

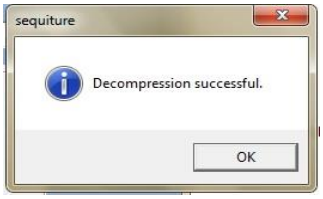
C. Menu Dekompresi

Selanjutnya pengujian *Dekompresi* / mengembalikan *file* ke ukuran semula tampa merusak isi dalam *file*tanpa merusak isi dalam *file*.



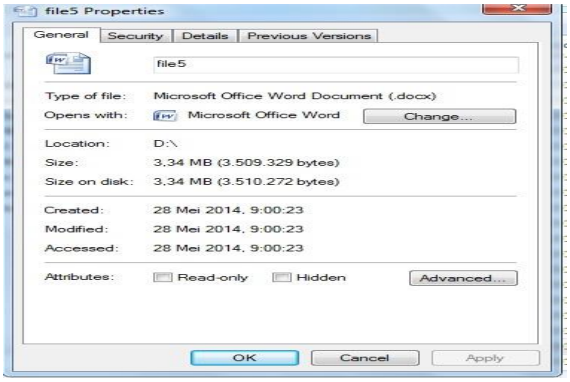
Gambar 4.7. Form Halaman Proses Dekompresi

Dekompressi berlangsung dengan intruksi / pengkodean algoritma sequitur yang di panggil tombol Decompress *Command button2*. Selanjutnya menunggu proses *Dekompres*i berlangsung sampai pemberitahuan muncul .



Gambar 4.8. Pemberitahuan proses Dekompresi Selesai

Setelah proses Dekompresi maka *file* sudah kembali ke bentuk dan ukuran awal tampa merusak isi file tersebut (*Lossless Compression*).

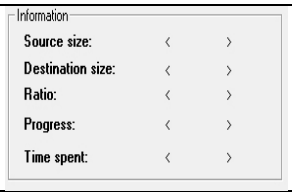


Gambar 4.9. Properties file5.docx Hasil Dekompresi

4.2 Pembahasan

4.2.1 Black Box Testing

Tabel 4.1. Tabel Uji Item

No	Item uji	Pengujian	Jenis uji	Hasil
1.		Proses Kompresi,.	Black Box	Berjalan sesuai fungsinya seperti yang di harapkan
2.		Proses Dekompresi	Black Box	Berjalan sesuai fungsinya seperti yang di harapkan
3.		Menanmpilakan Rincian dari proses	Black Box	Berjalan sesuai fungsinya seperti yang di harapkan
4.		Kerja Tombol keluar	Black Box	Berjalan sesuai fungsinya seperti yang di harapkan
5.		Label Penentuan File Asal Kompresi	Black Box	Berjalan sesuai fungsinya seperti yang di harapkan
6.		Label Penentuan File .SeQ	Black Box	Berjalan sesuai fungsinya seperti yang di harapkan
7.		Label Penentuan File hasil Decompresi	Black Box	Berjalan sesuai fungsinya seperti yang di harapkan

4.2.2 Hasil Pengujian

Dari hasil pengujian program aplikasi yang di lakukan terhadap beberapa file yang mempunyai ukuran berbeda, maka dalam hal ini dapatkan di simpulkan algoritma sequitur bekerja baik dalam mengkompresikan file teks, terutama pada file yang mempunyai ukuran yang lebih besar seperti pada tabel 4.1 berikut:

Table 4.2 Tabel Hasil pengujian *Kompresi file*

No	Nama file	Jenis File	Ukuran *file*	Kompresi	Pengurangan kompresi
1.	*File1*	*.docx*	*11,5 KB*	*11,1 Kb*	*0,4 Kb*
2.	*File2*	*.docx*	*147 KB*	*146 Kb*	*1 Kb*
3.	*File3*	*.docx*	*509 KB*	*503 Kb*	*5 Kb*
4.	*File4*	*.docx*	*1.513 KB*	*1.505 Kb*	*8 Kb*
5.	*File5*	*.docx*	*3.428 KB*	*3.419 Kb*	*9 Kb*

Tabel.4.3 Tabel Hasil *Dekompresi file .Seq*

No	Nama file	Jenis File	Ukuran File	DekompresI	Pengembalian kompresi
1.	*File1*	*.Seq*	*11,1 Kb*	*11,5 KB*	*0,4 Kb*
2.	*File2*	*.Seq*	*146 Kb*	*147 KB*	*1 Kb*
3.	*File3*	*.Seq*	*503 Kb*	*509 KB*	*5 Kb*
4.	*File4*	*.Seq*	*1.505 Kb*	*1.513 KB*	*8 Kb*
5.	*File5*	*.Seq*	*3.419 Kb*	*3.428 KB*	*9 Kb*

5 PENUTUP

5.1 Kesimpulan

1. Pada Proses Kompresi Pada : File1 ukuran awal 11,5 Kb menjadi 11,1 Kb berkurang 0,4 Kb , File2 ukuran awal 147 Kb menjadi 146 Kb berkurang 1 Kb, File3 ukuran awal 509 Kb menjadi 503 Kb berkurang 5 Kb, File4 ukuran awal 1.513 Kb menjadi 1.505 Kb berkurang 8 Kb, File5dengan ukuran awal 3.428 Kb menjadi 3.419 Kb berkurang 9 Kb
2. Pada Proses Dekompresi di lakukan pada File1 ukuran awal 11,1 Kb menjadi 11,5 KB pemgembalian ukuran 0,4 kb, File2 ukuran awal 146 Kb menjadi 147 KB pemgembalian ukuran 1kb, File3 ukuran awal 503 Kb menjadi 509 KB pemgembalian ukuran 5kb, File4 ukuran awal 1.505 Kb menjadi 1.513 KB pemgembalian ukuran 8kb, File5 ukuran awal 3.419 Kb menjadi 3.428 KB pemgembalian ukuran 9kb. Dekompresi yang dilakukan dengan algoritma sequitur tanpa merusak isi di dalam file tersebut (Lossless Compression).
3. File hasil kompresi oleh algoritmasequitur memudahkan dalam melakukan kegiatan menggunakan internet sehingga waktu yang diperlukan akan menjadi lebih pendek dan kemungkinan pekerjaan Download dan Upload gagal akan menjadi lebih kecil. Kemudian teransfer file melalui jaringan akan lebih cepat , waktu pengiriman tergantung dari provider yangcepat atau tidak serta ukuran file yang akan dikirim.
4. Algoritma sequitur membantu dalam mengurangi ukuran dari file sehingga dapat mengurangi kapasitas penyimpanan suatu memori / RAM. Tingkat keamanan file kompressi cukup terjaga,file tidak mengalami kerusakan setelah proses dekompresi dilakukan (Lossless Compression) . Algoritma sequitur sebaiknya di gunakan pada file berukuran besar saja, karna algoritma ini kurang berfungsi pada file yang berukuran keci, hal itu dikarnakan proses membaca karakter yang sama dalam suatu file.

5.2 Saran

Kompresi sequitur ini dapat dikembangkan lagi dalam hal jenis data teks dan membuat perbandingan yang lebih luas pada algoritma pengkompresian *teks* teks laninya.Penulis juga menyarankan untuk pengembangan selanjut nya agar program yang dibuat lebih *variatif* dan *inovatif* dari program aplikasi yang telah ada sebelumnya.

REFERENSI

Abraham Lempel & jacob Zib,1977, Lempel –Ziv-Welch Algorithm (LZW)

Aji Yoga Yudistira, Darwiyanto Eko, ST.,MT, Septia Gia, S.Si., M.Sc 2016,Analisis Perbandingan Kompresi dan Dekompresi Menggunakan Algoritma ShannonFano 2 Gram Dan Lempel Ziv

Welch Pada Terjemahan Hadits Shahih Muslim, e-Proceeding of Engineering, Vol.3, No.3, ISSN : 2355-9365

David Huffman. 1952 . The Huffman Algorithm .

Ervin E. 2011. Kompresi Data Teks Menggunakan Pendekatan Grammar Compression Dengan Algoritma Sequitur,ti.ukdw.ac.id/ojs/index.php/ informatika/article/view/41.

Gozali Ferrianto & Mervyn 2004, Analisis Perbandingan Kompresi Data Dengan Teknik Arithmetic Coding Dan Run Length Encoding, JETri, Universitas Trisakti, Vol. 4, No. 1, ISSN 1412-0372.

Halverson. M. 2000. Microsoft Visual Basic 6.0 Profesional, Cetakan I, Penerbit PT. Elex Media Komputindo : Jakarta.

Jogiyanto H.M. 2002. Pengenalan Komputer, Penerbit Andi Offset :Yogyakarta.

Naraprtaama. 2006 .Teknik Kompresi File, Penerbit Andi, Yogyakarta.

Niklaus Wirth. 2007. Algoritma + Struktur Data = Program

Rimincha Amnu. 2006 .Context Free Grammar. http:// www.laynetworks.com//Context%2oFree%20Gtammar.htm

Salomon . 2007. The Compression Algorithm ,Data Compression reference Center.