ISSN: 2355-536 X

PENGUJIAN KOMPRESI GABUNGAN ALGORITMA RUN IENGTH DAN HALF BYTE

DAVID
Program Studi Teknik Informatika
Sekolah Tinggi Manajemen Informatika dan Komputer Pontianak
Jln. Merdeka No. 372 Pontianak, Kalimantan Barat
David_Liauw@yahoo.com dan David_Liauw@stmikpontianak.ac.id

Abstracts

Algorithm Run Length and Half Byte has similarity in compressing data technic, which requires marker bit. Based on compressing technic from each algorithm, Run Length is more effective in picture file because it uses simultanously and similar bit character, meanwhile Half Byte algorithm utilizes character line which has left nibble (4 byte) from the same byte and that nibble seen oftenly at text files. The form of research which is used by the author in collecting secondary data is literature and observation study. The method design of software which is used by the author is Incremental model and STD (State Transition Diagram) analysis set, otherwise the testing technic is V-model testing and Ratio Compression Arithmetic. From the research result and testing, it can be seen that compression aplication with combaining Run Length Algorithm and Half Byte is more effective in *.tif, *.txt, and *.doc file because the certain file has many bytes which simultanously and orderly. Meanwhile, in other several type of file, compression result or measurement decrease data is very small than the size of original file. The suggestion from the author for the further research is by adding other several lossless compression algorithm so that obtaining better compression ratio score and independ in byte characters repetition in file.

Keywords : compression, decompression, run length, half byte, marker bit

1 PENDAHULUAN

Ukuran *file* yang semakin besar menuntut para pemakai komputer untuk mencari berbagai macam cara agar dapat menyimpan sejumlah besar *file* dalam media penyimpanan yang terbatas. Untuk itu ukuran dari *file* harus dimampatkan agar ukurannya menjadi lebih kecil. Teknik memampatkan data ini disebut dengan teknik kompresi data.

Menurut Merdiyan dan Indarto (2005:79-84) pada jurnal yang berjudul Implementasi Algoritma *Run Length*, *Half Byte* dan *Huffman* untuk Kompresi *File* menyimpulkan bahwa, algoritma *Half Byte* paling efektif pada *file-file* teks yang berisi teks-teks yang memiliki empat bit pertama yang sama, dan algoritma *Run Length* paling efektif pada *file-file* grafis yang deretan panjang karakter bit yang sama. Pada beberapa penelitian yang lain juga didapatkan kesimpulan yang sama, bahwa teknik algoritma *Run Length* lebih efektif jika digunakan pada data grafis dan teknik algoritma *Half Byte* lebih efektif jika digunakan pada data teks.

Indra Yatini B (2002:29-35) telah melakukan penelitian yang berjudul Kompresi Data *Menggunakan* Algoritma *Run length, Half Byte* dan *Huffman*. Peneliti menggunakan analisis perbandingan dari ketiga teknik algoritma tersebut. Antara ketiga metode kompresi data tersebut, penulis membandingkan kinerja dari metode tersebut, yaitu waktu yang dibutuhkan untuk mengkompresi data dan rasio hasil kompresi. Kesimpulan dari penelitian

dari segi rasio hasil kompresi yang diperoleh, maka metode *Huffman* lebih cocok untuk semua jenis *file*, kecuali *file* gambar yang lebih cocok dengan metode *Run Length*.

Samir Kumar Bandyopadhyay, Tuhin Utsab Paul, dan Avishek Raychoudhury (2011:117-121) menulis sebuah jurnal yang berjudul *Image Compression using Approximate Matching and Run Length*. Pada jurnal ini peneliti hanya membahas algoritma *Run Length* dan melihat kinerjanya dalam mengkomprei *file* grafis. Teknik algoritma ini akan lebih efektif lagi apabila digunakan untuk pencitraan medis karena karakteristik *lossless* dan gambar medis memiliki area yang besar dengan warna yang sama serta pola tata letak *pixel* yang sama, seperti pencitraan medis *X-Ray* yang memiliki area yang luas dengan warna hitam.

Mempertimbangkan hal-hal tersebut, penulis akan menguji dan merancang sebuah aplikasi kompresi yang berdasarkan pada penggabungan algoritma *Run Length* dan *Half Byte* karena kedua teknik algoritma kompresi ini memiliki kemiripan pada tekniknya, tetapi keunggulannya dalam pemadatan jenis *file* yang berbeda. Teknik algoritma *Run Length* bekerja berdasarkan sederetan karakter berulang yang berurutan dan sama, setelah itu akan diberikan *bit* penanda pada data yang dikompres. Seperti halnya pada *Run Length*, dalam teknik algoritma *Half Byte* juga memerlukan *bit* penanda sebagai penanda dari sederetan data yang sudah dikompres, tetapi algoritma ini hanya memanfaatkan deretan 4 *bit* kiri yang sama dan berderetan.

Tujuan utama dari *penelitian* ini adalah menghasilkan aplikasi kompresi yang lebih efektif dengan menggabungkan algoritma *Run Length* dan *Half Byte* dan menggunakan aplikasi tersebut untuk proses kompresi dan dekompresi beberapa jenis data, serta dapat mengetahui hasil pengujiannya. Dalam penggabungan perangkat lunak, penulis akan menggunakan pendekatan heuristik untuk menggabungkan kedua algoritma tersebut. Penulis juga akan menguji perangkat lunak tersebut pada beberapa jenis *file*, yaitu data teks, data citra, data suara, dan data video.

2 METODOLOGI PENELITIAN

Dalam penelitian ini bentuk penelitian yang akan digunakan oleh penulis adalah studi kasus, di mana penulis akan mempelajari teknik-teknik kompresi dari buku dan *website* yang berkaitan dengan penelitian. Penulis akan melakukan pengumpulan data sekunder. Data sekunder adalah data yang diperoleh peneliti dari sumber yang sudah ada. Data sekunder juga dapat digunakan sebagai sarana pendukung untuk memahami masalah yang akan diteliti. Teknik pengumpulan data yang digunakan adalah observasi dan studi literatur. Observasi data secara langsung akan didapatkan dari hasil pengukuran dengan menggunakan aplikasi yang dibuat berdasarkan teknik kompresi dengan algoritma *Run Length* dan *Half Byte*. Sedangkan studi literatur, dilakukan dengan mempelajari buku, literatur, referensi, jurnal dan artikel-artikel lainnya yang relevan dengan permasalahan yang dihadapi. Metode perancangan perangkat lunak yang digunakan adalah model Incremental, serta teknik pengujian yang digunakan adalah pengujian V-model dan perhitungan rasio kompresi. Semakin kecil persentase rasio kompresi yang didapatkan maka ukuran data yang berhasil dikompres semakin besar.

$$Rasio\ Kompresi = \frac{(ukuran\ hasil\ kompresi \times 100)}{ukuran\ data\ asli} = \cdots \%$$

ISSN: 2355-536X

3 HASIL PENELITIAN DAN PEMBAHASAN

Proses kompresi yang akan digunakan dalam *software* yang dirancang adalah gabungan antara algoritma *Run Length* dan *Half Byte*. Oleh karena itu, agar proses kompresi dan dekompresi dapat berjalan dengan baik maka teknik kompresi yang digunakan akan diubah sedemikian rupa. Contoh proses kompresi algoritma *Run Length* pada *file* yang berukuran 19 *byte* yang akan ditulis dalam bilangan heksadesimal : a) *File* : "73 75 75 75 75 75 75 75 75 75 75 75 96 96 96 96 96 96 96"; b) Terlihat pada *file* tersebut terjadi pengulangan karakter "75" dan "96" maka *file* dapat dilakukan proses kompresi; c) Hasil kompresi : "73 00 75 **0B** 00 96 **07**", "**0B**" berarti 11 kali pengulangan dan "**07**" berarti 7 kali pengulangan dan "00" adalah *bit* penanda yang akan digunakan untuk menentukan karakter yang telah dikompres.

Bila diperhatikan pada contoh di atas, algoritma *Run Length* dapat bekerja dengan baik. Akan tetapi, apabila digabungkan dengan algoritma *Half Byte* akan ada kemungkinan terjadi kesalahan pembacaan *bit* penanda atau proses kompresi yang terjadi dua kali sehingga pada saat proses dekompresi akan sulit dan kemungkinan besar gagal. Oleh karena itu, perancang akan membuat sedikit perubahan penggunaan *bit* penanda pada metode kompresi algoritma *Run Length* agar pada saat proses dekompresi *file* tidak terjadi kesalahan. Misalnya, *file* di atas dengan *byte* "73 00 75 **0B** 00 96 **07**" menjadi "73 00 75 **0B** 00 00 96 **07** 00" ("00" sebagai *bit* penanda awal dan akhir). Sehingga program dapat mengenal mana *byte* yang telah dikompres dengan algoritma *Run Length* dan mempermudah program untuk melakukan proses dekompresi.

Pada saat melakukan proses kompresi, ada kemungkinan akan terjadi pembengkakan ukuran *file*. Misalnya, *file* yang berisikan *byte* seperti "73 75 75 96 96 96" dan setelah dikompres dengan algoritma *Run Length* menjadi "73 00 75 **02** 00 00 96 **03** 00", maka akan terjadi pembengkakan ukuran *file* yang sebelumnya berukuran 6 *byte* menjadi 9 *byte* setelah proses kompresi. Oleh karena itu, pada program yang dirancang akan ditambahkan sebuah sebuah variabel dan rumus agar proses kompresi tidak akan mengakibatkan pembengkakan ukuran *file*, serta *bit* penanda yang khusus supaya saat proses dekompresi tidak terjadi kesalahan dan analisis ini juga akan diterapkan pada algoritma *Half Byte*.

Pada proses implementasi aplikasi kompresi untuk mengetahui hasil pengujian, pengujian dilakukan dengan menggunakan spesifikasi komputer sebagai berikut : a) Sistem operasi: Windows 7 Ultimate 32-bit; b) Framework : .NET Framework 4; c) Prosesor : Intel Core i5-2430M CPU 2.40 GHz; d) Memori : 2048 MB RAM DDR 3; e) VGA (*Video Graphics Array*) *Card* : Intel HD Graphics Family 748 MB; dan f) HDD (*Hard Disk Drive*) : ATA Hitachi HT554756 5400rpm 640 GB.

Berikut ini adalah hasil pengujian komponen dengan V-model dalam mencari kekurangan dan memverifikasi fungsi dari komponen *software* (misalnya modul, program, objek, kelas, dll) yang diuji secara terpisah :

Tabel 1. Hasil Pengujian Komponen

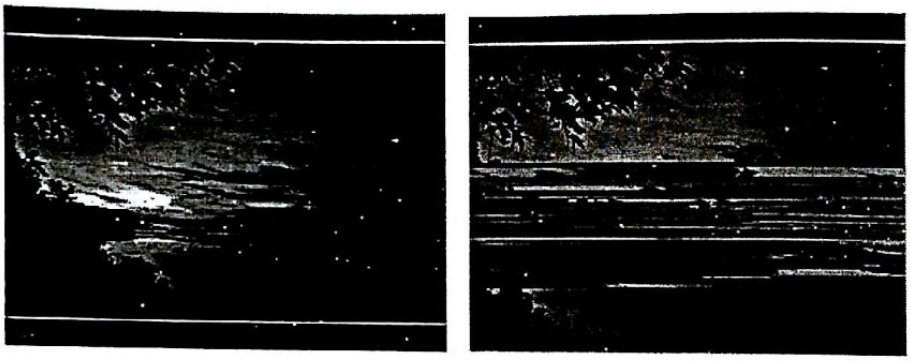
No.	Pengujian Komponen	Keterangan	Hasil
1	*Function* ReadAllByteToList	*Function* yang berfungsi untuk membaca *file* ke dalam bentuk *byte*	Berhasil

2	*Read Only* *Bit* penanda pada *Run Length* dan *Half Byte*	Memberikan *bit* penanda pada untuk masing-masing algoritma	Berhasil
3	*Function* UniCompress (Kompresi)	Jika menggunakan kombinasi 1 *bit* penanda pada setiap algoritma (ukuran *file* =< 250 MB)	Berhasil
		Jika menggunakan kombinasi 2 *bit* penanda pada setiap algoritma (ukuran *file* =< 250 MB)	Berhasil
		Jika menggunakan kombinasi 3 *bit* penanda pada setiap algoritma (ukuran *file* =< 250 MB)	Berhasil
		Jika menggunakan kombinasi 4 *bit* penanda pada setiap algoritma (ukuran *file* =< 250 MB)	Berhasil
		Jika menggunakan kombinasi 5 - 9 *bit* penanda pada setiap algoritma (ukuran *file* =< 250 MB)	Berhasil
		Jika menggunakan kombinasi 10 *bit* penanda pada setiap algoritma (ukuran *file* =< 250 MB)	Berhasil
4	*Function* UniDecompress (Dekompresi)	Jika menggunakan kombinasi 1 *bit* penanda, pada saat melakukan proses dekompresi *file office*, **.pdf*, **.mp3*, **.mp4*, **.mkv*, **.tif*, **.jpg*, dan **.bmp*. Terjadi pembengkakan ukuran *file* dan *System Out Of Memory*	Gagal
		Jika menggunakan kombinasi 2 *bit* penanda, pada saat melakukan proses dekompresi *file office*, **.pdf*, **.mp3*, **.mp4*, **.mkv*, **.tif*, **.jpg*, dan **.bmp*. Terjadi pembengkakan ukuran *file* dan *System Out Of Memory*	Gagal
		Jika menggunakan kombinasi 3 *bit* penanda, pada saat melakukan proses dekompresi *file* **.docx*, **pptx*, **.mp4*, **.mkv*, dan **.pdf* . Terjadi pembengkakan ukuran *file* dan *System Out Of Memory*	Gagal
		Jika menggunakan kombinasi 4 *bit* penanda pada setiap algoritma (ukuran *file* =< 250 MB)	Berhasil
		Jika menggunakan kombinasi 5 - 9 *bit*	Berhasi

ISSN: 2355-536X

		penanda pada setiap algoritma (ukuran *file* =< 250 MB)	1
		Jika menggunakan kombinasi 10 *bit* penanda pada setiap algoritma (ukuran *file* =< 250 MB)	Gagal
5	*Function* CheckStateInData Grid (Tabel Data)	Mengecek tipe *file* yang terdapat pada Tabel Data, jika tipe *file* adalah **.lhc* maka tombol dekompresi yang berfungsi, dan sebaliknya.	Berhasil

Jika program menggunakan kombinasi 4 sampai dengan 9 *bit* penanda pada saat melakukan proses dekompresi pada *file* yang berukuran lebih dari 250 MB, program akan gagal mengembalikan data tersebut kembali seperti semula dan muncul sebuah *window* seperti pada gambar 4. Akan tetapi, kegagalan proses dekompresi tidak hanya terjadi karena ukuran *file* yang lebih dari 250 MB, tetapi ada juga kesalahan karena program keliru dalam membaca *bit* penanda.



(a) (b)

Gambar 1. (a) Data asli (b) Kegagalan proses dekompresi

Berdasarkan dari tabel pengujian komponen, penguji akan menggunakan aplikasi kompresi dengan kombinasi 4 *bit* penanda karena proses kompresi dan dekompresi lebih stabil dibandingkan dengan kombinasi *bit* yang lainnya, dan juga semakin sedikit panjang *bit* penanda yang digunakan maka semakin besar kemungkinan data yang akan dapat dikompres.

Berikut ini adalah hasil pengujian integrasi terhadap *hardware* dengan metode V-model :

Tabel 2. Hasil Pengujian *Hardware*

No.	Sistem Operasi	*Hardware*	Ukuran File	Hasil
1	Windows 7 Ultimate 64-bit	RAM 4GB DDR3	=< 250MB	Berhasil
		RAM 4GB DDR3	> 250MB	Gagal
2	Windows 7 Ultimate	RAM 2GB	=<	Berhasi

	32-bit	DDR3	250MB	1
		RAM DDR3	2GB	> 250MB	Gagal
3	Windows 7 Ultimate 32-bit	RAM DDR3	1GB	=< 250MB	Berhasil
		RAM DDR3	1GB	> 250MB	Gagal

Jika diperhatikan pada tabel 2 dapat diambil sebuah kesimpulan bahwa kegagalan atau keberhasilan aplikasi dalam melakukan proses kompresi dan dekompresi bukan dari masalah sistem *hardware* pada komputer. Akan tetapi, kegagalan atau keberhasilan suatu proses kompresi dan dekompresi terletak pada struktur *coding* atau panjangnya *bit* penanda yang digunakan pada algoritma tersebut.

Berikut ini adalah hasil pengujian aplikasi kompresi yang telah dirancang terhadap beberapa jenis data teks :

Tabel 3. Hasil Pengujian Terhadap Data Teks

No.	Nama File	Ukuran Data (KB)	Waktu (milidetik)		Ukuran Data Kompresi (KB)	Rasio	Hasil De-kompresi
			Kompresi	Dekompresi			
1	Data TXT	230,599	66	13	162,254	70%	Berhasil
2	Data DOC	1789	427	85	1467,841	82%	Berhasil
3	Data PPT	4071,5	884	229	3528,544	86%	Berhasil
4	Data XLS	1917	426	98	1891,85	98%	Berhasil
5	Data PDF	9054,296	2125	341	9041,895	100%	Berhasil

Dari tabel 3 dapat dilihat bahwa rasio kompresi paling rendah adalah 70% pada *file* Data TXT, sedangkan pada Data PDF dan XLS rasio kompresi yang didapatkan sangat besar.

Berikut ini adalah hasil pengujian aplikasi kompresi yang telah dirancang terhadap beberapa jenis data citra :

Tabel 4. Keterangan Data Citra

No.	Nama File	Resolusi Gambar	Kedalaman *Bit*	Ukuran *File* (KB)
1	Data GIF	160x120	1-bit	985,499
2	Data BMP	4497x3264	24-bit	43005,805
3	Data IFF	4928x3864	24-bit	32300,172

ISSN: 2355-536X

4	Data JPG	4928x3864	24-bit	1829,885
5	Data PNG	3450x2285	24-bit	13582,421
6	Data TGA	4497x3264	32-bit	57336,793
7	Data TIF	4928x3924	24-bit	56685,012

Tabel 5. Hasil Pengujian Terhadap Data Citra

No.	Nama File	Waktu (milidetik)		Ukuran Data Kompresi (KB)	Rasio	Hasil Dekompresi
		Kompresi	Dekompresi			
1	Data GIF	292	68	985,499	100%	Berhasil
2	Data BMP	9017	2093	40475,815	94%	Berhasil
3	Data IFF	7215	1237	32300,172	100%	Berhasil
4	Data JPG	471	84	1827,539	100%	Berhasil
5	Data PNG	2899	489	13582,405	100%	Berhasil
6	Data TGA	12575	2474	56683,525	98%	Berhasil
7	Data TIF	8526	2028	37672,318	66%	Berhasil

Jika diperhatikan dari tabel 4 dan 5, terdapat empat data citra yang sama sekali tidak mengalami perubahan setelah diproses oleh aplikasi dan data TGA hanya mengalami sedikit pengurangan ukuran data. Sedangkan pada data TIF didapatkan rasio kompresi 66% yang menunjukkan bahwa terjadi perubahan ukuran data yang besar setelah melalui proses kompresi.

Berikut ini adalah hasil pengujian aplikasi kompresi yang telah dirancang terhadap beberapa jenis data suara :

Tabel 6. Keterangan Data Suara

No.	Nama File	Durasi Musik (menit:detik)	*Bit Rate*	Ukuran *File* (KB)
1	Data FLAC	03:32	751 kbps	19495,063
2	Data MP3	03:27	320 kbps	8126,809
3	Data OGG	03:10	131 kbps	3038,353
4	Data WAV	03:47	1411 kbps	39123,043
5	Data WMA	03:28	128 kbps	3267,585

Tabel 7. Hasil Pengujian Terhadap Data Suara

ISSN: 2355-536X

No.	Nama File	Waktu (milidetik)		Ukuran Data Kompresi (KB)	Rasio	Hasil Dekompresi
		Kompresi	Dekompresi			
1	Data FLAC	4322	734	19491,082	100%	Berhasil
2	Data MP3	1799	299	8022,376	98%	Berhasil
3	Data OGG	756	127	3037,553	100%	Berhasil
4	Data WAV	8971	84	39039,985	100%	Gagal
5	Data WMA	722	198	3259,887	100%	Berhasil

Jika diperhatikan dari tabel 7, dapat disimpulkan bahwa teknik kompresi yang digunakan pada program tidak efektif pada data suara dan hanya *file* MP3 yang mengalami sedikit pemampatan. Pada proses pengembalian data (dekompresi) *file* WAV mengalami kegagalan.

Berikut ini adalah hasil pengujian aplikasi kompresi yang telah dirancang terhadap beberapa jenis data video :

Tabel 8. Keterangan Data Video

No.	Nama File	Durasi Video (menit:detik)	Resolusi	Frame Rate	Ukuran *File* (KB)
1	Data 3GP	05:31	320x180	23 fps	9123,873
2	Data AVI	24:15	848x480	23 fps	185759,264
3	Data FLV	05:58	426x240	23 fps	15365,499
4	Data MKV	65:45	800x450	29 fps	202435,645
5	Data MP4	24:18	1280x720	23 fps	88065,052

Tabel 9. Hasil Pengujian Terhadap Data Video

No.	Nama File	Waktu (milidetik)		Ukuran Data Kompresi (KB)	Rasio	Hasil Dekompresi
		Kompresi	Dekompresi			
1	Data 3GP	2217	385	9115,841	100%	Berhasil
2	Data AVI	42744	9068	185730,854	100%	Berhasil
3	Data FLV	3464	581	15316,837	100%	Berhasil
4	Data MKV	45230	7274	202430,549	100%	Berhasil

ISSN: 2355-536X

5	Data MP4	20033	3114	88064,279	100%	Berhasil

Dari rasio kompresi dapat disimpulkan bahwa teknik kompresi pada aplikasi sangat tidak efektif untuk memampatkan data-data video karena rasio kompresi yang didapatkan adalah 100%.

4 KESIMPULAN

Berdasarkan hasil penelitian, dan pembahasan yang telah dilakukan penulis, dapat disimpulkan bahwa terbukti jika proses kompresi dilakukan pada karakter berulang yang berurutan dan sama, maka rasio kompresinya akan sangat kecil yang berarti bahwa data yang berhasil dikompres semakin besar (*file* *.*tif* 66% dan *.*txt* 70%). Sedangkan jika *file* memiliki *byte* yang variatif, maka peluang untuk mengurangi ukuran data dengan teknik kompresi yang digunakan akan semakin kecil. Aplikasi yang dirancang dengan menggabungkan teknik kompresi algoritma *Run Length* dan *Half Byte* tidak efektif pada *file* video dan suara, dan kurang efektif pada beberapa jenis *file* gambar dan teks. Kedua algoritma sangat bergantung pada karakter *bit* yang berulang ada pada *file*.

DAFTAR PUSTAKA

[1] Salomon, David., 2008, *A Concise Introduction to Data Compression*, Springer, London.

[2] Salomon, David. 2007. *Data Compression The Complete Reference,* 4th Ed, Springer, London.

[3] Pu, Ida M., 2006, *Fundamental Data Compression*, Butterworth-Heinemann, London.

[4] Pressman, Roger S., 2001, *Software engineering: a practitioner's approach,* 5th edition, McGraw-Hill, New York.

[5] Sujaini, Herry dan Yesi Mulyani, 2000, *Algoritma RUN-LENGTH HALF-BYTE dan HUFFMAN untuk Pemampatan File*, http://kambing.ui.ac.id/bebas/v01/OnnoWPurbo/contrib/network/buku-algoritma-run-length-half-byte-untuk-pemampatan-file-06.doc, diakses tanggal 2 Juni 2013.

[6] Meckah Merdiyan, dan Wawan Indarto, 2005, *Implementasi Algoritma Run Length, Half Byte dan Huffman untuk Kompresi File*, *Jurnal Fakultas Teknologi Industri Universitas Islam Indonesia*, http://journal.uii.ac.id/index.php/Snati/article/view/1391 diakses tanggal 8 Maret 2013.

[7] Yatini B, Indra, 2002, *Kompresi Data Menggunakan Algoritma Run-length, Half-byte Dan Huffman*, *Jurnal STMIK AKAKOM*, http://pustaka2.ristek.go.id/katalog/index.php/search katalog/byId/274550 diakses pada tanggal 13 Maret 2013.

[8] Samir Kumar Bandyopadhyay, Tuhin Utsab Paul, dan Avishek Raychoudhury, 2011, *University Of Calcutta Paper, Image Compression using Approximate Matching and Run Length,* http://ebookbrowse.com/paper-17-image-compression-using-approximate matching -and-run-length-pdf-d348634984 diakses tanggal 31 Mei 2013.