ISBN: 979-756-061-6

IMPLEMENTASI ALGORITMA RUN LENGTH, HALF BYTE DAN HUFFMAN UNTUK KOMPRESI FILE

Meckah Merdiyan, Wawan Indarto
Laboratorium Cisco Network Academy Program
Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia
Jalan Kaliurang Km 14 Jogjakarta 55501
Telp.(0274) 895287 ext. 122, Faks. (0274) 895007 ext. 148
E-mail: meckah_m@yahoo.com, wawan@fti.uii.ac.id

Abstrak

Masalah kompresi data merupakan salah satu aspek penting perkembangan teknologi informasi. Kompresi digunakan untuk berbagai keperluan antara lain: membackup data, transfer data dan salah satu bagian keamanan data. Terdapat banyak teknik kompresi data, tiga diantaranya adalah algoritma run length, half byte dan huffman. Masing-masing algoritma memiliki teknik kompresi yang berbeda. Algoritma run length memanfaatkan deretan karakter yang berurutan, kemudian dikompresi menggunakan format kompresi algoritma run length berjumlah 3 byte, yaitu byte penanda, jumlah karakter dan karakter yang dikompresi. Algoritma half byte memanfaatkan deretan karakter yang memiliki nibble(4 byte) kiri dari byte yang sama, kemudian dikompresi menggunakan format kompresi yang terdiri dari : byte penanda, karakter pertama yang akan dikompresi, penggabungan nibble kanan byte ke 2 dan byte ke 3, penggabungan nibble karakter selanjutnya dan ditutup dengan byte penanda. Sedangkan algoritma huffman memanfaat frekuensi karakter yang akan dikompresi, kemudian membuat node-node karakter dengan jumlah kemunculan karakter, pengurutan berdasarkan karakter ASCII dan frekuensinya, kemudian membentuk pohon huffman, pemberian bit 0 untuk cabang kiri dan bit 1 untuk cabang kanan, kemudian mencari kode untuk masing-masing karakter dan menulis hasil kompresi berdasarkan kode yang sudah didapat.

***Keywords**: Kompresi, Dekompresi, Run Length, Half Byte, Huffman.*

1. PENDAHULUAN

1.1 Latar Belakang

Di dalam dunia komputer dan internet, kompresi *file* digunakan dalam berbagai keperluan, misalnya mem*backup* data dan transfer data. Untuk mem*backup* data tidak perlu menyalin semua *file* aslinya, dengan kompresi (mengecilkan ukurannya) *file* terlebih dahulu, maka kapasitas tempat penyimpanan yang diperlukan menjadi lebih kecil. Jika data tersebut diperlukan, maka dikembalikan lagi ke *file* aslinya (dekompresi *file*).

Selain berguna pada media penyimpanan data, kompresi *file* dapat membantu memperkecil ukuran data yang ditransimisikan di dalam suatu media jaringan, seperti internet sehingga waktu yang diperlukan akan menjadi lebih pendek dan kemungkinan pekerjaan *down-load* dan *up-load* gagal akan menjadi lebih kecil.

Ada banyak teknik kompresi data yang dikategorikan menurut jenis data yang akan dikompresi, yaitu:

- Teknik kompresi untuk citra diam (*still image*) antara lain: JPEG, GIF, dan *run length*.
- Teknik kompresi untuk citra bergerak (*motion picture*) antara lain: MPEG.
- Teknik kompresi untuk data teks antara lain: *half byte*
- Teknik kompresi untuk data umum antara lain: LZW, *half byte* dan *huffman*.
- Teknik kompresi untuk data sinyal *speech* antara lain: *pulse code modulation* (PCM), *sub-band coding* (SBC), *linear predictive coding* (LPC), dan *code excited* LPC (CELP).

Di antara teknik-teknik kompresi tersebut, yang akan digunakan sebagai penelitian adalah teknik kompresi dengan algoritma *run length, half byte* dan *huffman*.

1.2 Hipotesis

Dugaan-dugaan yang akan dibuktikan yaitu algoritma *run length* paling efektif pada *file-file* grafis yang berisi deretan panjang karakter yang sama. Algoritma *half byte* paling efektif pada *file-file* text yang berisi text-text yang memiliki empat bit (*nible*) pertama yang sama. Sedangkan algoritma *huffman* cukup efektif untuk berbagai macam jenis *file*. Tidak ada algoritma yang paling efektif untuk setiap *file* karena hasil kompresi setiap algoritma tergantung dari isi *file* yang akan dikompresi. Ketiga algoritma tidak optimal jika digunakan untuk kompresi *file* yang sudah dikompresi menggunakan metode lain, misalnya untuk mengkompres *file* dengan format *mpeg*, *jpeg* dan format kompresi *file* yang lain.

ISBN: 979-756-061-6

2. LANDASAN TEORI

2.1 Pengertian Kompresi

Istilah kompresi berasal dari kata bahasa inggris yaitu *compression* yang berarti pamampatan. Secara teknis, kompresi berarti memampatkan segala sesuatu yang berukuran besar sehingga menjadi lebih kecil. Jadi kompresi *file* berarti proses untuk memampatkan *file* agar ukurannya menjadi lebih kecil.

2.2 Perbandingan Kompresi

Ada banyak cara untuk mengukur perbandingan (*ratio*) kompresi dari *file* yang terkompresi terhadap *file* asal. Dalam tulisan ini digunakan perbandingan yang didefinisikan sebagai:

*(1-(ukuran_kompresi / ukuran_asal)) * 100*

2.3 Jenis-jenis Kompresi

- Lossy
  Kompresi data lossy (*destruktif*) dilakukan dengan menghilangkan bagian informasi yang dianggap tidak penting, sehingga dihasilkan rasio kompresi yang sangat tinggi
- Lossles
  Kompresi data lossless (*non-destruktif*) memampatkan data secara *reversible*, artinya jika sebuah data dikompres, kemudian didekompres lagi, maka data tersebut sama dengan data aslinya atau pengkompresian dilakukan tanpa menghilangkan bagian dari informasi.

2.4 Teknik-teknik Kompresi

Beberapa teknik kompresi data yang akan digunakan untuk penelitian ini merupakan jenis kompresi *lossless*, antara lain:

2.4.1 *Run length*

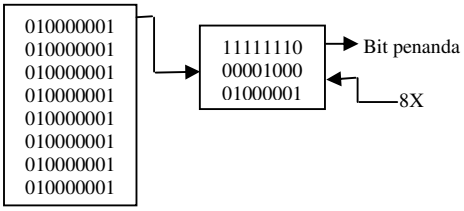
Teknik *run length* bekerja berdasarkan sederetan karakter yang berurutan. Data masukan akan dibaca dan sederetan karakter yang sesuai dengan deretan karakter yang sudah ditentukan sebelumnya disubstitusi dengan kode tertentu. Kode khusus ini biasanya terdiri dari tiga buah karakter.

Sc	Cc	X

Gambar 1. Tiga buah karakter

Keterangan dari gambar tersebut adalah sebagai berikut:
- Sc adalah karakter khusus yang dipakai sebagai tanda kompresi
- Cc adalah banyaknya karakter yang dekompresi
- X adalah karakter berurutan yang akan dikompresi

Sebagai contoh apabila karakter khusus (Sc) yang digunakan adalah # dan Cc dalam bilangan desimal, maka jika digunakan untuk kompresi potongan *string* "Jarrrrrringan", maka akan diperoleh hasil "Ja#6ringan".



Gambar 2. Contoh kompresi menggunakan algoritma *run length*

Algoritma kompresi adalah:
a. Cari deretan karakter yang sama secara berurutan, jika ada lakukan kompresi data.
b. Tulis *byte* penanda pada *file* kompresi, *byte* penanda berupa 8 deretan *bit* yang boleh dipilih sembarang asalkan digunakan secara konsisten pada seluruh *byte* penanda kompresi. *Byte* penanda ini berfungsi untuk menandai bahwa karakter selanjutnya adalah karakter kompresi sehingga tidak membingungkan pada saat mengembalikan *file* yang sudah dikompresi ke *file* aslinya.
c. Tulis deretan *bit* untuk menyatakan jumlah karakter yang sama berurutan.
d. Tulis deretan *bit* yang menyatakan karakter yang berulang.
e. Ulangi langkah 1-4 sampai karakter terakhir.

Algoritma dekompresi adalah:
a. Lihat karakter pada hasil kompresi satu-persatu dari awal sampai akhir, jika ditemukan *byte* penanda, lakukan proses pengembalian.
b. Lihat karakter setelah *byte* penanda, konversikan ke bilangan desimal untuk menentukan jumlah karakter yang berurutan.
c. Lihat karakter berikutnya, kemudian lakukan penulisan karakter tersebut sebanyak bilangan yang telah diperoleh pada karakter sebelumnya (*point* 2).
d. Ulangi langkah 1, 2 dan 3 sampai karakter terakhir.

2.4.2 *Half byte*

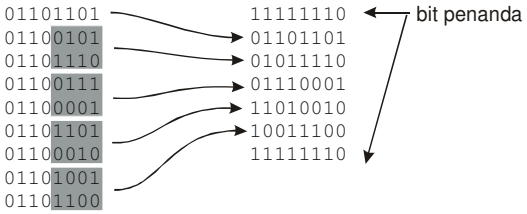
Seperti halnya pada run length, dalam teknik *half byte* juga memerlukan karakter khusus sebagai tanda dari sederetan data yang sudah dikompresi.

 ISBN: 979-756-061-6

Sc	N1	N2	N4	…	N_x	Sc
		N3	N5	…	N_x	

Gambar 3. Format teknik *half byte*

Keterangan dari gambar (2.3) adalah sebagai berikut:
- Sc adalah karakter khusus sebagai tanda kompresi.
- N adalah karakter yang sudah dikompresi.



Gambar 4. Contoh kompresi menggunakan algoritma *half byte*

Deretan data pada gambar (4), sebelah kiri merupakan deretan data pada *file* asli, sedangkan deretan data sebelah kanan merupakan deretan data hasil kompresi dengan algoritma *half byte*.

Algoritma kompresi adalah:
1. Cari deretan karakter yang 4 *bit* pertamanya sama secara berurutan, lakukan kompresi.
2. Tulis *byte* penanda pada *file* kompresi, berupa 8 deretan *bit* (1 *byte*) yang boleh dipilih sembarang asalkan digunakan secara konsisten pada seluruh *byte* penanda kompresi. *Byte* penanda ini berfungsi untuk menandai bahwa karakter selanjutnya adalah karakter kompresi, sehingga tidak membingungkan pada saat mengembalikan *file* yang sudah dikompresi ke *file* aslinya.
3. Tulis karakter pertama dari 4 *bit* kiri berurutan dari *file* asli.
4. Gabungkan 4 *bit* kanan karakter kedua dan ketiga kemudian tulis ke *file* kompresi. Lakukan hal ini sampai akhir deretan karakter dengan 4 *bit* pertama yang sama.
5. Tutup dengan menulis *byte* penanda pada *file* kompresi.
6. Ulangi langkah 1-5 sampai karakter terakhir.

Algoritma dekompresi:
1. Lihat karakter pada hasil kompresi satu-persatu dari awal sampai akhir, jika ditemukan *byte* penanda, lakukan proses pengembalian.
2. Lihat karakter setelah *byte* penanda, tulis karakter tersebut pada *file* pengembalian.
3. Lihat karakter berikutnya, ambil 4 *bit* kiri dan 4 *bit* kanannya, lalu masing-masing 4 *bit* kiri dan 4 *bit* kanan digabungkan dengan 4 *bit* kiri karakter setelah *byte* penanda. Hasil gabungan tersebut ditulis pada *file* pengembalian. Lakukan sampai ditemukan *bit* penanda.
4. Ulangi langkah 1-3 sampai karakter terakhir.

2.4.3 *Huffman*

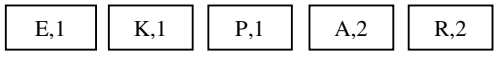
Algoritma *huffman* dengan menggunakan pohon biner adalah sebagai berikut:
1. Siapkan daftar peluang untuk tiap simbol yang muncul
2. Cari dua simbol yang memiliki peluang terkecil
3. Simbol pertama diberikan *bit* 0 dan yang lainnya berikan *bit* 1
4. Gabungkan probabilitas kedua simbol tersebut menjadi satu kombinasi
5. Ulangi langkah 2,3 dan 4 sampai semua simbol telah digabungkan menjadi 1 kombinasi saja.
6. Tulis karakter berdasarkan kode masing-masing karakter.

Sebagai contoh, sebuah *file* yang akan dikompresi berisi karakter-karakter "PERKARA". Dalam kode ASCII masing-masing karakter dikodekan sebagai berikut :

P = 01010000
E = 01000101
R = 01010010
K = 01001011
A = 01000001

Langkah-langkah selanjutnya adalah sebagai berikut:
1. Hal yang pertama dilakukan adalah menghitung frekuensi kemunculan masing-masing karakter, jika dihitung ternyata P muncul sebanyak 1 kali, E sebanyak 1 kali, R sebanyak 2 kali, K sebanyak 1 kali dan A sebanyak 2 kali. Untuk karakter yang memiliki frekuensi kemunculan sama seperti E, K dan P disusun menurut kode ASCII-nya, begitu pula untuk A dan R.
2. Frekuensi tersebut jika diurutkan makin naik sebagai berikut:
   E = 1
   K = 1
   P = 1
   A = 2
   R = 2
3. Pohon *huffman* dibuat mengikuti langkah-langkah sebagai berikut:

   3.1 Buatlah 5 node untuk masing-masing pasangan (karakter, frekuensi).



Gambar 5. Lima buah node lengkap dengan frekuensinya

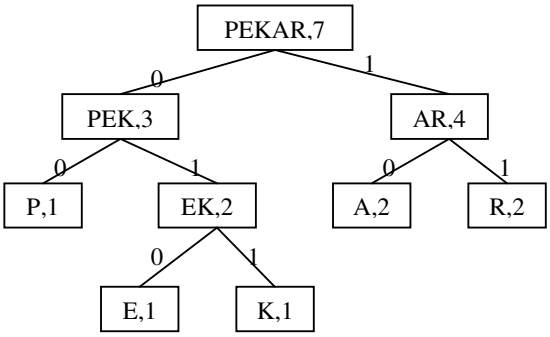
   3.2 Pilihlah dua node dengan frekuensi terkecil (dua node paling kiri), dan buatlah node baru yang isinya merupakan gabungan dua

 ISBN: 979-756-061-6

node terkecil tersebut. Dua node baru tersebut menjadi bapak dari dua node tersebut. Node (EK,2) menjadi bapak dari node (E,1) dan (K,1). Node bapak tersebut diurutkan bersama node-node yang lain (node (P,1), node (A,2) dan node (R,2)), sedangkan node (E,1) dan node (K,1) tidak ikut diurutkan lagi karena merupakan node anak

3.3 Lakukan seperti langkah 2 sehingga seluruh karakter yang ingin dikodekan terbentuk node akar atau sebuah pohon (*tree*). Gambar berikut bentuk pohon *huffman* yang sudah terbentuk.

3.4 Langkah selanjutnya, mulai dari akar (*root*) dari pohon ini, lintasi atau lewatilah pada karakter yang ingin dikodekan. Berilah *bit* 0 pada node yang merupakan anak kiri dan *bit* 1 pada node yang merupakan anak kanan. Maka akan diperoleh *tree* berikut ini:



Gambar 6. Pohon huffman lengkap dengan kode urutan *bit*.

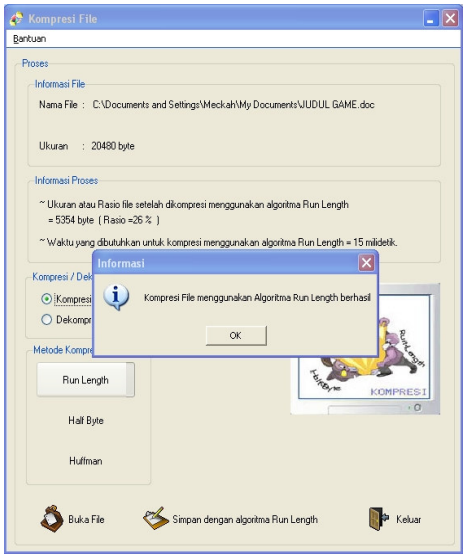
Untuk mendapatkan kode masing-masing karakter, dilakukan *traversal* (lintasan) dari node paling bawah melewati semua *ancestor* (jalur) sampai ke *ancestor* (jalur) paling atas sebelum node akar. Maka jika diterapkan pada node (K,1) dihasilkan 110 dibalik menjadi 011, yang merupakan kode dari K. Dengan cara yang sama dihasilkan kode dari E adalah 010, kode dari P adalah 00, kode dari A adalah 10 dan kode dari R adalah 11.

Algortima dekompresi:

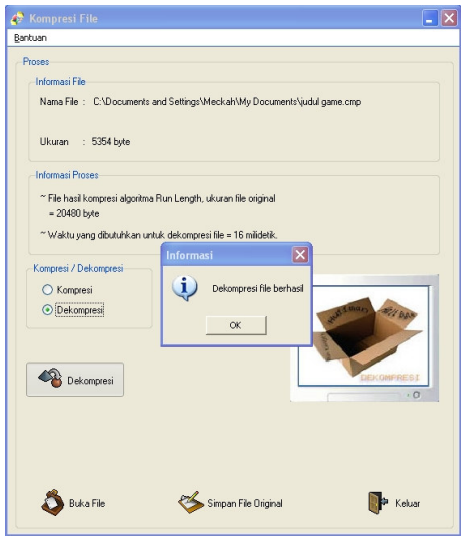
a. Ambil satu-persatu *bit* hasil kompresi mulai dari kiri, jika *bit* tersebut termasuk dalam daftar kode, lakukan pengembalian, jika tidak ambil kembali *bit* selanjutnya dan gabungkan *bit* tersebut.
b. Demikian selanjutnya dikerjakan sampai *bit* terakhir sehingga akan didapatkan hasil dekompresi.

3. IMPLEMENTASI

Program implementasi algoritma *run length, half byte* dan *huffman* untuk kompresi file mempunyai dua antarmuka, yaitu antarmuka kompresi dan dekompresi.



Gambar 7. Antarmuka kompresi *file*



Gambar 8. Antarmuka dekompresi *file*

4. ANALISIS

Analisis perbandingan digunakan untuk melihat kinerja dari ketiga algoritma. Perbandingan berupa analisis hasil kompresi *file*/rasio terhadap ukuran *file* asli dan waktu proses kompresi dan dekompresi terhadap ukuran *file*.

4.1 Analisis waktu proses terhadap ukuran *file*

Waktu proses kompresi/dekompresi *file* tergantung pada ukuran *file* yang akan dikompresi/dekompresi dan spesifikasi komputer(*hardware* dan *software*), terutama perangkat keras yang akan memproses kompresi/dekompresi. Pada penelitian ini, pengujian

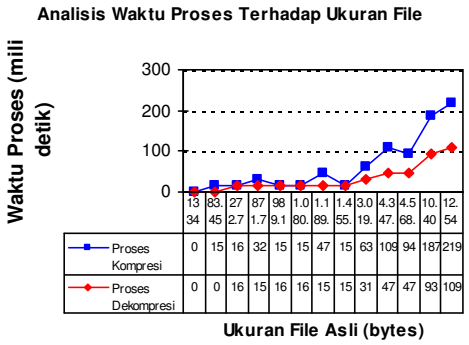
ISBN: 979-756-061-6

proses kompresi dan dekompresi *file* menggunakan spesifikasi komputer sebagai berikut:

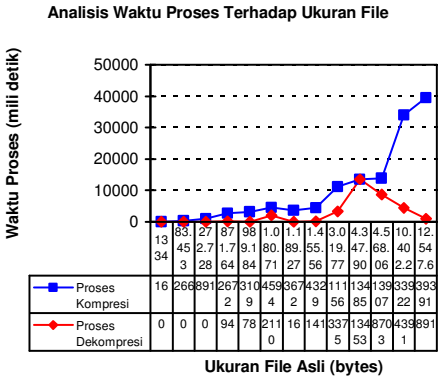
- Processor P4 1,6 Ghz
- DDR 256 MB/PC 2100
- Sistem Operasi Windows XP SP 1

Tabel 1. Analisis waktu proses kompresi/dekompresi terhadap ukuran *file*

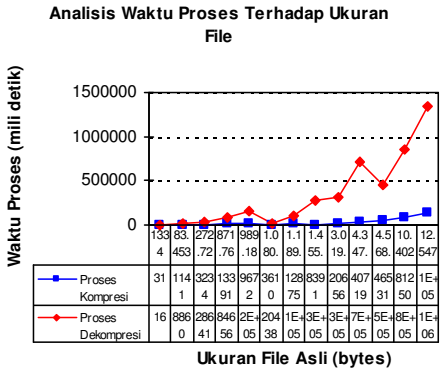
Nama file	*Ukuran File Asli (byte)*	*Waktu proses kompresi(milidetik)*			*Waktu proses kompresi(milidetik)*		
		Run length	*Half byte*	*Huffman*	*Run length*	*Half byte*	*Huffman*
Tteropong.bmp	1334	0	16	31	0	0	16
Christmas.gif	83453	15	266	1141	0	0	8860
Amikom.jpg	272718	16	891	3234	16	0	28641
Kartu lebaran.cdr	871764	32	2672	13391	15	94	84656
Sidang.ppt	989184	15	3109	9672	16	78	156453
STIE.tif	1080712	15	4594	3610	16	2110	20438
Research.pdf	1189271	47	3672	12875	15	16	110531
istilah-ti.txt	1455569	15	4329	8391	15	141	279172
Teknologi.doc	3019776	63	11156	20656	31	3375	307813
TEST-HDR.avi	4347904	109	13485	40719	47	13453	712250
O La La.mp3	4568065	94	13907	46531	47	8703	450125
3.mpg	10402224	187	33922	81250	93	4391	848082
Layang.wav	12547628	219	39391	136187	109	891	1348625



Gambar 9. Grafik waktu proses terhadap ukuran *file* (algoritma *run length*)



Gambar 10. Grafik waktu proses terhadap ukuran *file* (algoritma *half byte*)



Gambar 11. Grafik waktu proses terhadap ukuran *file* (algoritma *huffman*)

Dari data pada tabel (1) menunjukkkan bahwa algoritma *run length* lebih cepat dalam menyelesaikan proses kompresi, misalnya pada file *teropong.bmp*(0 detik), *cristmas.gif*(15 detik), *amikom.jpg*(16 detik), sedangkan untuk dekompresi misalnya pada file *teropong.bmp*(0 detik), *cristmas.gif*(0detik), *amikom.jpg*(16 detik). Sedangkan algoritma *huffman* membutuhkan waktu paling lama untuk proses kompresi misalnya pada file *teropong.bmp*(31 detik), *cristmas.gif*(1141 detik), *amikom.jpg*(3234 detik), terlebih pada proses dekompresi misalnya pada file *teropong.bmp*(16 detik), *cristmas.gif*(8860 detik), *amikom.jpg*(28641 detik), sebab pada proses dekompresi algoritma *huffman* mencocokkan deretan bit(bit per bit) pada *file* hasil kompresi dengan daftar kode pada masing-masing karakter.

Pada uji coba proses kompresi dengan menggunakan beberapa ekstensi *file* dengan ukuran yang berbeda–beda dapat disimpulkan bahwa semakin besar ukuran *file*, maka waktu proses kompresi/dekompresi *file* semakin lama.

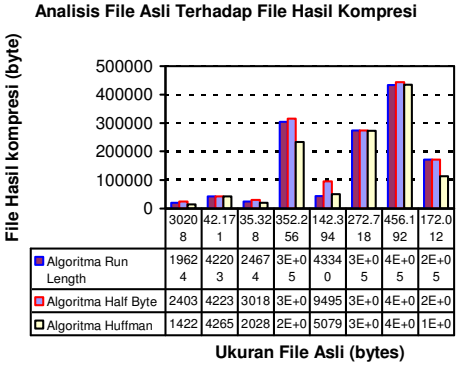
4.2 Analisis ukuran file asli terhadap file hasil kompresi

Analisis dengan membandingkan ukuran *file* sebelum dikompresi dan setelah *file* dikompresi.

Tabel 2. Analisis ukuran file asli terhadap file hasil kompresi

Nama file	*Ukuran File Asli(byte)*	*Ukuran file hasil kompresi(byte)*		
		Run length	*Half byte*	*Huffman*
bisnis plan warnet.xls	30208	19624 (64%)	24037 (79%)	14224 (47%)
Bart.gif	42171	42203 (100%)	42232 (100%)	42659 (101%)
Distance.rtf	35328	24674 (69%)	30182 (85%)	20285 (57%)
16.fla	352256	303853 (86%)	315278 (89%)	233728 (66%)
ABAYO.tif	142394	43340 (30%)	94952 (66%)	50799 (35%)
Amikom.jpg	272718	273035 (100%)	273121 (100%)	272403 (99%)
aplikasi-sistem.doc	456192	433381 (94%)	443933 (97%)	435085 (95%)
Data.txt	172012	171061 (99%)	171124 (99%)	113056 (65%)

ISBN: 979-756-061-6



	30208	42171	35328	352256	142394	272718	456192	172012
Algoritma Run Length	19624	42203	24674	3E+05	43340	3E+05	4E+05	2E+05
Algoritma Half Byte	24037	42232	30182	3E+05	94950	3E+05	4E+05	2E+05
Algoritma Huffman	14224	42659	20285	2E+05	50790	3E+05	4E+05	1E+05

Gambar 12. Grafik *file* asli terhadap *file* hasil kompresi

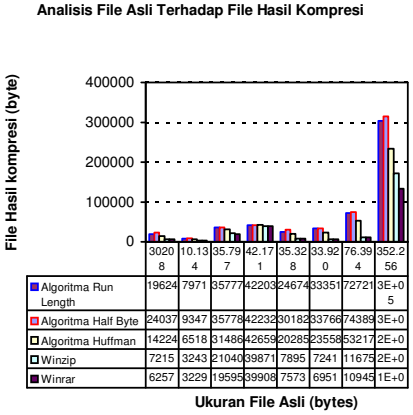
Dari data pada tabel (2) menunjukkan bahwa algoritma *Huffman* cukup efektif untuk berbagai macam jenis *file*, yaitu pada file *bart.gif*(47 %), *distance.rtf*(57 %), *16.fl*a(66 %), *amikom.jpg*(99 %), *data.txt*(65 %) , meskipun ada beberapa *file* yang hasil kompresinya lebih besar dari yang dihasilkan oleh algoritma *run-length*, yaitu pada file *abayo.tif*(30 %), *aplikasi-sistem.doc*(94 %). Namun tidak ada algoritma yang paling efektif untuk setiap *file* karena hasil kompresi setiap algoritma tergantung dari isi *file* yang akan dikompresi.

4.3 Analisis algoritma *run length*, *half byte*, *huffman*, *winzip* dan *winrar*

Analisis dengan membandingkan algoritma *run lengt*, *half byte*, *huffman, winzip* dan *winrar*.

Tabel 3. Perbandingan algoritma *run lengt*, *half byte*, *huffman, winzip* dan *winrar*.

Nama file	*Ukuran File (byte)*	*Ukuran file hasil kompresi(byte)*				
		Run length	*Half byte*	*Huffman*	*Winzip*	*Winrar*
bisnis plan warnet.xls	30208	19624 (64%)	24037 (79%)	14224 (47%)	7215 (24%)	6257 (20%)
Cloud Logo.ico	10134	7971 (78%)	9347 (92%)	6518 (64%)	3243 (32%)	3229 (31%)
16.swf	35797	35777 (99%)	35778 (99%)	31486 (87%)	21040 (59%)	19595 (54%)
Bart.gif	42171	42203 (100%)	42232 (100%)	42659 (101%)	39871 (95%)	39908 (94%)
Distance.rtf	35328	24674 (69%)	30182 (85%)	20285 (57%)	7895 (22%)	7573 (21%)
Login.php	33920	33351 (98%)	33766 (99%)	23558 (69%)	7241 (21%)	6951 (20%)
Phoenix.sql	76394	72721 (95%)	74389 (97%)	53217 (69%)	11675 (15%)	10945 (14%)
16.fla	352256	303853 (86%)	315278 (89%)	233728 (66%)	171373 (49%)	132955 (37%)



	30208	10134	35797	42171	35328	33920	76394	352256
Algoritma Run Length	19624	7971	35777	42203	24674	33351	72721	3E+05
Algoritma Half Byte	24037	9347	35778	42232	30182	33766	74389	3E+05
Algoritma Huffman	14224	6518	31486	42659	20285	23558	53217	2E+05
Winzip	7215	3243	21040	39871	7895	7241	11675	2E+05
Winrar	6257	3229	19595	39908	7573	6951	10945	1E+05

Gambar 13. Grafik perbandingan dengan produk komersial

Dari data pengujian pada tabel (3) menunjukkan bahwa hasil kompresi menggunakan *winrar* lebih kecil dibandingkan dengan empat teknik yang lain, dengan hasil pada masing-masing file cloud bisnis planwarnet.xls(20 %)), cloud logo.ico(31 %), 16.swf(54 %), bart.gif(94 %), distance.rtf(21 %), login.php(20 %), phoenix.sql(14 %), 16.fla(37 %). Kemudian *winzip* pada urutan kedua dengan hasil pada masing-masing file cloud bisnis planwarnet.xls(24 %), cloud logo.ico(31 %), 16.swf(59 %), bart.gif(95 %), distance.rtf(22 %), login.php(21 %), phoenix.sql(15 %), 16.fla(49 %), sedangkan pada urutan ketiga yaitu algoritma *huffman* dengan hasil pada masing-masing file cloud bisnis planwarnet.xls(47 %), cloud logo.ico(64 %), 16.swf(87 %), distance.rtf(57 %), login.php(69 %), phoenix.sql(69 %), 16.fla(66 %).

5. KESIMPULAN

Dengan demikian dari uraian tersebut, maka hipotesis diterima, sebab tidak ada algoritma yang paling efektif untuk setiap *file* karena hasil kompresi setiap algoritma tergantung dari isi *file* yang akan dikompresi dan ketiga algoritma tidak efektif/ optimal menghasilkan *file* hasil kompresi yang sudah dikompresi menggunakan metode lain, misalnya untuk mengkompresi *file* dengan format JPEG.

PUSTAKA

[1] Adisantoso, J, Danny D. S dan Bib P. S. (2004). *Kompresi Data Menggunakan Algoritma Huffman*, Yogyakarta: Seminar Nasional Aplikasi Teknologi Informasi.

[2] Gillbert dan Thomas R. M. (1996). *Data and Compression: Tool and Techniques*, England: Jhon Wiley & Sons Ltd.

[3] Kristanto, A. (2003). *Keamanan Data Pada Jaringan Komputer*, Yogyakarta: Gava Media.

[4] Suryanto, T. (1995). *Pemampatan File dengan Algoritma Huffman*, Jakarta: Dinastindo.

[5] Wirahadi. (2000). *Kompresi Data Menggunakan Metode Huffman dan Implementasi Dengan Turbo C*, Yogyakarta: Tugas Akhir Jurusan Teknik Informatika Universitas Islam Indonesia.