ANALISIS PERBANDINGAN ALGORITMA ARITHMETIC CODING DENGAN ALGORITMA LEMPEL ZIV WELCH (LZW) DALAM KOMPRESI TEKS

[1]Roni Agus Purba, [2]Lamhot Sitorus
[1]Teknik Informatika Unika St. Thomas S.U; Jln. Setia Budi No.479-F Medan, 061-8210161
[2]Teknik Informatika Unika St. Thomas S.U; Jln. Setia Budi No.479-F Medan, 061-8210161
e-mail : purba.saputra@gmail.com., lamhot68@yahoo.com

Abstrak

Untuk melakukan kompresi teks, telah banyak algoritma yang dikembangkan dengan teknik pengompresian yang berbeda-beda, namun sayang sekali tidak ada satupun algoritma kompresi tersebut yang baik untuk mengompresi berbagai teks hal ini disebabkan karena karakteristik atau struktur setiap teks, sedangkan kebanyakan algoritma kompresi data memanfaatkan struktur teks tersebut dalam proses kompresi, akibatnya hanya pada teks tertentu saja mungkin suatu algoritma lebih efektif dari yang lainnya.

Penelitian ini penulis membahas perbandingan dua algoritma kompresi teks, yaitu Arithmetic Coding dan Lempel Ziv Welch (LZW), Algoritma-algoritma ini dipilih karena algoritma tersebut bersifat lossless dan umumnya digunakan pada teks, dimana dua algoritma pertama merupakan perwakilan dan pengembangan dari masing-masing kategori kompresi dengan teknik pengkodean yang berbeda.

Kata kunci: Kompresi, *Arithmatic Coding, Lempel Ziv Welch*

Abstract

To do text compression, there have been many algorithms developed with different compression techniques, but unfortunately there is no single compression algorithm that is good for compressing various texts because of the characteristics or structure of each text, while most data compression algorithms utilize structure the text is in the process of compression, as a result, only in certain texts may an algorithm be more effective than others.

This study the author discusses the comparison of two text compression algorithms, namely Arithmetic Coding and Ziv Welch (LZW), these algorithms are chosen because the algorithm is lossless and is commonly used in text, where the first two algorithms represent representation and development of each category compression with different coding techniques.

Keywords: Compression, Arithmatic Coding, Lempel Ziv Welch

1. PENDAHULUAN

Pertumbuhan besarnya data yang digunakan pada teknologi informasi saat ini berkembang sangat cepat yang sangat mempengaruhi media penyimpanan dan transmisi data. Hal ini disebabkan data dalam bentuk elektronis lebih disukai dari pada dalam bentuk konvensional, karena beberapa faktor tertentu seperti masalah keamanan data, kemudahan dalam pemrosesan, pendistribusian, maupun modifikasi data.

Dalam hal transmisi data, besarnya ukuran data juga turut menjadi masalah yang serius, karena memperlambat sampainya data ke tujuan semakin memakan waktu. Sehingga komputer tidak dapat digunakan untuk kebutuhan lain secara maksimal, karena harus selalu berhadapan dengan masalah besarnya ukuran data. Maka salah satu alternatif yang digunakan untuk

mengatasi masalah di atas adalah dengan melakukan kompresi teks sehingga ukurannya menjadi lebih kecil dari ukuran semula, dimana data yang dimaksudkan bisa berupa teks.

Untuk melakukan kompresi teks, telah banyak algoritma yang dikembangkan dengan teknik pengompresian yang berbeda-beda, namun sayang sekali tidak ada satupun algoritma kompresi tersebut yang baik untuk mengompresi berbagai teks hal ini disebabkan karena karakteristik atau struktur setiap *teks*, sedangkan kebanyakan algoritma kompresi data memanfaatkan struktur *teks* tersebut dalam proses kompresi, akibatnya hanya pada *teks* tertentu saja mungkin suatu algoritma lebih efektif dari yang lainnya.

Untuk itu, dalam penelitian ini penulis akan membandingkan dua algoritma kompresi teks, yaitu Arithmetic Coding dan Lempel Ziv Welch (LZW), Algoritma-algoritma ini dipilih karena algoritma tersebut bersifat *lossless* dan umumnya digunakan pada teks, dimana dua algoritma pertama merupakan perwakilan dan pengembangan dari masing-masing kategori kompresi dengan teknik pengkodean yang berbeda, dimana Arithmetic Coding adalah algoritma berbasis *entropy* yang bahkan telah berhasil. Sedangkan algoritma Lempel Ziv Welch (LZW) adalah algoritma kompresi berbasis *dictionary* yang merupakan pengembangan dari algoritma *Lempel Ziv 77 (LZ77)* dan *Lempel Ziv 78* (LZ78).

2. METODOLOGI PENELITIAN

II.1. Kompresi

Kompresi data merupakan cabang ilmu komputer yang bersumber dari teori informasi. Teori informasi sendiri dipelopori oleh Claude Shannon sekitar tahun 1940-an. Teori ini mendefenisikan jumlah informasi di dalam pesan sebagai jumlah minimum *bit* yang dibutuhkan yang dikenal sebagai *entropy*. Teori ini juga memfokuskan pada berbagai metode tentang informasi termasuk tentang *redundancy* (informasi tak berguna) pada suatu pesan dimana semakin banyak *redundancy* maka semakin besar pula ukuran pesan, upaya mengurangi *redundancy* inilah yang akhirnya melahirkan subjek ilmu tentang kompresi data.

Menurut David Salomon data kompresi adalah proses pengkonversian *input* data *stream* (sumber *stream*, atau data mentah asli) ke bentuk *stream* lain (*output stream*, atau *stream* yang sudah terkompresi) yang memiliki ukuran yang lebih kecil. *Stream* adalah sebuah *file* ataupun *buffer* dalam *memory*. Sedangkan menurut Ida Mengyi Pu kompresi data adalah konteks dari ilmu komputer, ilmu (dan seni) yang menyajikan informasi ke dalam bentuk yang *compact*. Pada umumnya data kompresi terdiri dari pengambilan *stream* simbol dan mengubahnya ke dalam bentuk kode. Jika kompresi efektif, hasil kode *stream* akan lebih kecil daripada simbol asli.

Namun pada dasarnya data kompresi bertujuan untuk mengubah ukuran data menjadi lebih kecil dari ukuran semula sehingga dapat menghemat media penyimpanan dan waktu untuk transmisi data, beberapa penggunaan kompresi datadapat dilihat seperti pada gambar-gambar yang diperoleh dari internet yang biasanyadalam format JPEG (*Joint Photographic Experts Group* atau GIF (*GraphicalInterchange Format*), *modem* yang sebagian besar menggunakan kompresi, HDTV(*High Defenition Television*) akan dikompresi menggunakan MPEG-2 (*MovingPicture Experts Group 2*), dan beberapa *teks* sistem yang secara otomatis dikompresi ketika *file* tersebut disimpan.

Untuk melakukan kompresi data telah banyak algoritma yang telah dikembangkan. Namun, secara umum algoritma kompresi data dapat di bagi menjadidua kelompok besar berdasarkan dapat tidaknya rekontuksi ke data asli dilakukan.

II.2. Arithmatic Coding

Pada umumnya algoritma kompresi data melakukan teknik pengkodean dengan penggantian satu atau beberapa simbol yang sama dengan kode tertentu, berbeda dengan *Arithmetic Coding*, algoritma ini mengkodekan seluruh *inputs tream* dalam suatu *teks* dengan

sebuah angka *floating point* dengan interval [0,1) atau angka yang lebih besar atau sama dengan 0 dan lebih kecil dari 1 ($0 \leq x < 1$).

Arithmetic Coding memiliki sejarah yang penting karena pada saat itu algoritma ini berhasil menggantikan algoritma *Huffman* selama 25 tahun, *Arithmetic Coding* memiliki perfomansi yang lebih unggul dari pada *Huffman Coding* khususnya apabila diaplikasikan pada kumpulan alphabet yang ukurannya relatif kecil, awalnya *Arithmetic Coding* diperkenalkan oleh Shannon Fano dan Elias, kemudiandikembangkan secara luas oleh Pasco (1976), Rissanene (1976, 1984), dan Langdon (1984), yaitu sebagai ide alternatif yang menggantikan setiap *input simbol* dengansebuah *codeword*, yaitu dengan mengkodekan seluruh *input stream* dengan sebuah *single floating point* sebagai *output* proses kompresi. Untuk melakukan proses kompresi maupun dekompresi, *Arithmetic Coding* membutuhkan dua fase, yaitu dengan terlebih dahulu menentukan probalitas dan *range* setiap simbol lalu melakukan proses *encoding* maupuan *decoding.*

II.3. Lempel Ziv Welch

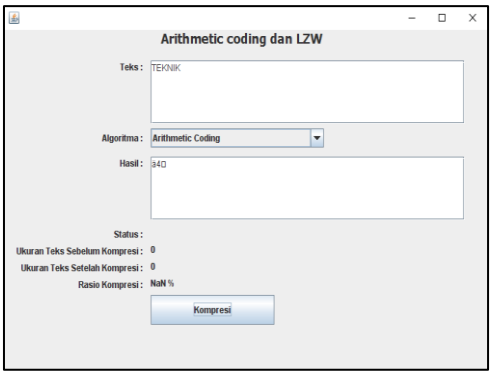
Algoritma LZW merupakan algoritma kompresi data *lossless* yang ditemukan olehTerry Welch, dimana algoritma ini merupakan peningkatan dari versi sebelumnyayaitu algoritma *Lempel Ziv 77* (LZ77) dan *Lempel Ziv 78* (LZ78) yang dikembangkan oleh Abraham Lempel dan Jacob Ziv pada tahun 1977 dan 1978. Kemudian algoritma LZW dipublikasikan oleh Terry Welch pada tahun 1984.

Algoritma LZW adalah algoritma yang bersifat adaptif dan berbasis *dictionary*, dimana selama proses kompresi atau dekompresi berlangsung algoritma inimenggunakan sebuah *dictionary* yang dibentuk selama pembacaan *input stream*.*Dictionary* dibentuk dengan tujuan untuk menyimpan karakter atau pola *string*tertentu yang berguna untuk mengkodekan simbol atau *string* pada input stream yangmerujuk pada *index* dalam *dictionary*. Sebelum proses kompresi atau dekompresidimulai, sebuah *dictionary* akan diinisialisasi dengan simbol atau karakter-karakterdasar penyusun input *stream*, sehingga nilai default sebuah *dictionary* akan berisi 256 karakter ASCII dengan *index* 0-255. Sehingga pada awal pembacaan proses *encoding* maupun *decoding*, karakter atau kode pertama akan selalu ditemukan pada *dictionary.*

3. HASIL DAN PEMBAHASAN

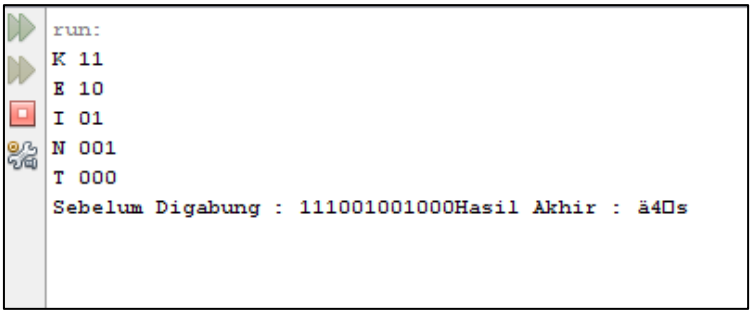
III.1. Hasil

Implementasi merupakan proses pembangunan komponen-komponen pokok sebuah sistem berdasarkan desain yang sudah dibuat. Implementasi sistem juga merupakan sebuah proses pembuatan dan penerapan sistem secara utuh baik dari sisi perangkat keras maupun perangkat lunaknya. Implementasi yang akan dijelaskan di sini meliputi lingkungan perangkat lunak.



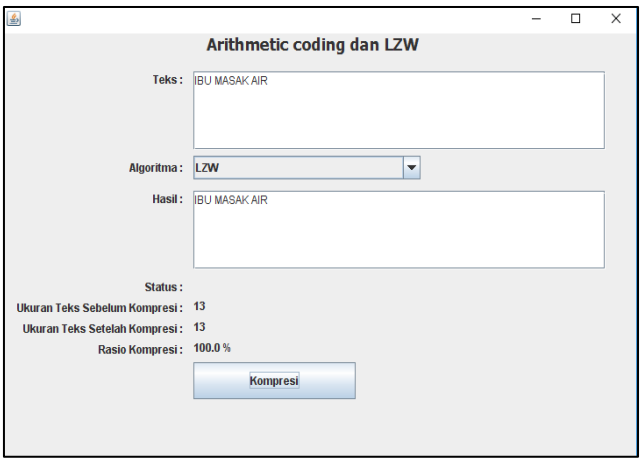
Gambar 1. Tampilan Form Kompresi teks Arithmetic Coding

Dimana dalam form ini, kalau diinput teks maka hasil kompresi ke Arithmetic Coding. Tidak menapilkan ukuran teks sebelum kompresi dan ukuran teks setelah dikompresi, dan hanya menampilkan resiko kompresi saja.



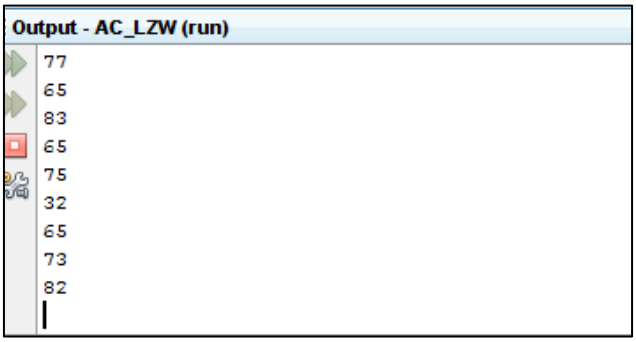
Gambar 2. Tampilan Form Kompresi teks Lempel Ziv Welch (LZW)

Pada output Kompresi Arithmetic Coding ini, setiap-tiap karakter akan tampil nilai kompresinya. Dan di hitung dalam bilangan biner.



Gambar 3. Tampilan Form Kompresi teks Lempel Ziv Welch (LZW) dan Arithmatic Coding

Pada form ini jika meninput teks ke Lempel Ziv Welch (LZW), maka hasil kompresi Lempel Ziv Welch (LZW) akan tampil ukuran teks sebelum dan ukuran teks setelah kompresi. Kemudian tampil resiko kompresinya, sedangkan kalau mengkompresi ke Arithmetic Conding tidak tampil ukuran teks sebelum kompresi dan ukuran teks setelah kompresi.



Dimana output kompresi teks Lempel Ziv Welch (LZW), dihitung setiap-tiap karakter mempunyai nilai tiap karakter.

III.2. Pembahasan

Misalkan kata yang di ambil adalah "TEKNIK" maka dilakukan pendataan karakter dari "TEKNIK" yang di jabarkan pada Tabel 1

Tabel 1. Perihitungan Probabilitas Frekuensi Kemunculan Karakter

Karakter	Frekuensi
T	1
E	1
K	1
N	1
I	1
K	1
Jumlah	6

Tabel 2. Probabilitas Frekuensi Kemunculan Karakter Setiap Karakter

Karakter	Frekuensi	Probabilitas
T	1	1/6 = 0,6
E	1	1/6 = 0,6
K	1	1/6 = 0,6
N	1	1/6 = 0,6
I	1	1/6 = 0,6
K	1	1/6 = 0,6
Jumlah	6	0,96

Dihitung jangkauan setiap karakter, dengan titik terendah (*low*) adalah 0,0 dan titik tertinggi (*high*) adalah 1,0, sehingga jumlah dari probabilitas seluruh simbol sama dengan titik tertinggi dari jangkauan tersebut.

Tabel 3. Jangkauan Setiap Karakter

Karakter	Frekuensi	Probabilitas	Jangkauan	
			Low_range	High_range
T	1	1/6 = 0,6	0,84	1,0
E	1	1/6 = 0,6	0,68	0,84
K	1	1/6 = 0,6	0,52	0,68
N	1	1/6 = 0,6	0,36	0,52
I	1	1/6 = 0,6	0,2	0,36
K	1	1/6 = 0,6	0,16	0,20
Jumlah	6	0,96	-	

Diinisialisasi nilai titik terendah (*low*) sebagai 0 dan titik tertinggi (*high*) sebagai 1,0. Kemudian dilakukan perhitungan *low* dan *high* sesuai data setiap karakter dengan rumus kompresi Arithmetic Coding sebagai berikut:

Range = high-low
Low = low + range x high_range
High = low + range x low_range

Perhitungan *low* dan *high* pada proses kompresi Arithmetic Coding dilakukan per 2 karakter dan dijabarkan pada **Tabel 4.**

Tabel 4. Kompresi Arithmetic Coding

Karakter	**Low/High**	
TE	Low	0,0 + (1,0 – 0,0) * 0,84 = 0,84
	High	0,0 + (1,0 – 0,0) * 1,0 = 1,0
	Low	0,84+ (1,0 – 0,84) * 0,68 = 0,95
	High	0,84 + (1,0 – 0,84) * 0,84 =0,97
KN	Low	0,0 + (1,0 – 0,0) * 0,52 = 0,53
	High	0,0 + (1,0 – 0,0) * 0,68 = 0,68
	Low	0,52 + (0,68 – 0,52) * 0,36 = 0,58
	High	0,52 + (0,68 – 0,52) * 0,52 = 0,60
IK	Low	0,0 + (1,0 – 0,0) * 0,2 = 0,2
	High	0,0 + (1,0 – 0,0) * 0,36 = 0,36
	Low	0,2 + (0,36 – 0,2) * 0,4 = 0,26
	High	0,2 + (0,36 – 0,2) * 0,2 = 0,2

Representasi dari kata **"TEKNIK"** yang telah dimampatkan diambil dari nilai biner yang berada di antara(0,95 – 0,97),(0,68 – 0,60) dan (0,4 – 0,2).

K 11
E 10
I 01
N 001
T 000

Sebelum Digabung :111001001000

Aplikasi kompresi dengan algoritman LZW di sini akan di jelaskan pengkompresian terhadap beberapa karakter "KAKAK MASAK AIR" akan di kompresi dengan LZW. Isi *dictionary* pada awal proses dise karakter dasar yaitu : **"KAAKK_MASAK_AIR"** . *Bit dictionar*y yang dipakai adalah *bit*. tahapan proses kompresi ditunjukkan Diketahui : Input String : KAKAK_MASAK_AIR

Table 5. Tahapan Proses Kompresi

No	Input	Code	Char	Output
1	K	_	_	none
2	A	256	KA	K
3	K	257	AK	A
4	A	257	KAK	none
5	K	_	_	KA
6	_	259	K_	K
7	M	260	_M	_
8	A	261	MA	M
9	S	262	AS	A
10	A	263	SA	S
11	K	_	_	none
12	_	264	AK_	AK
13	A	265	A_	_
14	I	266	AI	A
15	R	267	IR	I
				R

Hasil kompresi dari string yaitu : **"**KAAKK_MASAK_AIR**"**
Decimal : 75,65,256,75,95,77,65,83,257,95,65,73,82.
Heksa : 4B,41,100,4B,5F,4D,41,53,101,5F,41,49,52.
Biner : 1001011-1000001-100000000-1001011-1011111-1001101 1000001-1010011-100000001-1011111-1000001-1001001-1010010.
(KAAKK_MASAK_AIR)

4. KESIMPULAN

Kesimpulan yang di peroleh dari hasil pengerjaan Proyek Perangkat Lunak ini tentan analisa perbandingan Algoritma Arithmetic Coding dan Algoritma LZW dalam mengkompresi teks yaitu :

a. Penerapan metode *dictionary* dengan menggunakan Algoritma Lempel Ziv Welch (LZW) dalam apliksai kompresi teks memiliki waktu cepat dalam melakukan proses kompresi teks.
b. Uji coba yang dilakukan, dengan melakukan pengkompresi terhadap beberapa teks menggunakan Algoritma Lempel Ziv Welch (LZW) sangat bagus untuk mengkompresi teks.
c. Proses pengkompresian terhadap sebuah teks yang berukuran besar akan menghasilkan teks dengan ukuran yang lebih besar dari aslinya. Hal ini disebabkan karna munculnya karakter yang berbeda-beda dan membuat tambah banyak karakter yang tersimpan *dictionary,* sehingga menghasilkan teks yang besar dari aslinya.
d. Algoritma Arithmetic Coding ini cukup baik di pakai dalam mengkompresi teks.
e. Untuk inpelentasi Aritmetic Coding sebaiknya dilakukan dengan perhitungan bilangn biner.

5. SARAN

Beberapa saran untuk pengembangan lebih lanjut yang diberikan oleh penulis adalah :

a. Perbaikan dalam proses pembacaan karakter yang dilakukan dalam algoritma LZW, sehingga dengan demikian proses kompresi dapat memampatkan semua file dengan lebih baik.
b. Pembentukan struktur *header teks* yang lebih efisien sehingga memungkinkan terbentuk sebuah teks terkompresi dengan ukuran yang lebih kecil.
c. Evaluasi dapat dilakukan dengan membandingkan dengan algoritma kompresi teks lainnya.

DAFTAR PUSTAKA

[1] Anton, 2005. *Kompresi dan Teks*. Fakultas Teknik Informatika. Univesitas Kristen Duta Wacan http://lecturer.ukdw.ac.id/anton/download/multimedia6.pdf Tanggal Akses : 12 Februari 2016

[2] Mokhamad, Adi Pn Fazmah Arief Yulianto, Endro Ariyanto 2008 Analisis dan Implementassi Perbandinga Kinerja Algoritma Kompresi Huffman,LZW,dan DMC Pada Berbagai Tipe File Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom Tanggal Akses 5 Februari 2016

[3] Taleumbanua, Pilipus 2011 Analisis Perbandingan Algoritma Kompresi LEMPEL ZIV WELCH, ARITHMETIC CODING, dan RUN-LENGTH ENCODING Pada File

Teks Fakultas Matematika dan Ilmu Pengetahuan Alam, Ilmu Komputer, Universitas Sumatra Utara. Tanggal Akses : 12 Februari 2016

[4] Subarka Aan, Fuad 2010 Rancangan Bangun Aplikasi Kompresi File Menggunakan Metode LZW Berbasis Java Fakultas Sains Teknologi, Teknik Informatika, Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang http://lib.uin-malang.ac.id/files/thesis/fullchapter/05550055.pdf Tanggal Akses : 01 Februari

[5] Petrus Santoso, 2001 Studi Kompresi Data dengan Metode *Arithmetic Coding* Jurusan Teknik Elektro, Fakultas Teknologi Industri – Universitas Kristen Petra http://puslit.petra.ac.id/journals/electrical/ Tanggal Akses : 01 Februari 2016

[6] Suarga, 2006 *Algoritma dan Pemograman*. Andi Offset. Yogyakarta. ISBN : 978-979-29-2745-0.