KOMPRESI *FILE* TEKS BERDASARKAN KARAKTERISTIK HIMPUNAN DENGAN MODEL BIGRAM

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Oleh:
YOSSY HADIWIDJAJA
0610963072

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2011

LEMBAR PENGESAHAN SKRIPSI

KOMPRESI *FILE* TEKS BERDASARKAN KARAKTERISTIK HIMPUNAN DENGAN MODEL BIGRAM

oleh :

YOSSY HADIWIDJAJA
0610963072

Telah dipertahankan di depan Majelis Penguji
pada tanggal 3 Agustus 2011
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Edy Santoso, SSi., M.Kom.	**Nurul Hidayat, S. Pd., MSc.**
NIP 19740414-200312-1-004	**NIP 19680430-200212-1-001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP. 19670907-199203-1-001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: Yossy Hadiwidjaja**
NIM	**: 0610963072**
Jurusan	**: Matematika**
Program Studi	**: Ilmu Komputer**
Penulis skripsi berjudul	**: Kompresi *File* Teks Berdasarkan Karakteristik Himpunan dengan Model Bigram**

Dengan ini menyatakan bahwa :

1. **Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.**
2. **Apabila dikemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 3 Agustus 2011
Yang menyatakan,

(Yossy Hadiwidjaja)
NIM. 0610963072

KOMPRESI *FILE* TEKS BERDASARKAN KARAKTERISTIK HIMPUNAN DENGAN MODEL BIGRAM

ABSTRAK

Karakteristik himpunan dapat digunakan sebagai salah satu metode untuk kompresi *file* teks. Tujuan penelitian ini adalah mengetahui tingkat rasio kompresi *file* teks berdasarkan karakteristik himpunan dengan model bigram. Proses kompresi dimulai dengan memecah isi *file* menggunakan model bigram, pembentukan kamus data, pengubahan indeks isi *file* sesuai dengan indeks kamus, pengubahan indeks *file* menjadi bentuk biner dan pengubahan bentuk biner *file* menjadi desimal. Jika *file* kompresi diubah menjadi *file* asli, maka dilakukan proses dekompresi melalui tahapan pengubahan isi *file* dekompresi menjadi bentuk biner, pemotongan bentuk biner *file* dekompresi sesuai dengan panjang biner indeks tertinggi kamus data, pengubahan ke bentuk desimal dan dicocokkan dengan nilai encode kamus data. Kemudian nilai encode kamus data dicocokkan dengan string kamus data dan menghasilkan *file* asli.

Hasil penelitian menunjukkan bahwa dengan menggunakan algoritma yang didasarkan pada karakteristik himpunan dengan model bigram, dihasilkan persentase rata-rata rasio kompresi untuk *file* dengan ekstensi *.txt sebesar 25.94%, *.htm sebesar 26.44% dan *.rtf sebesar 26.95%. *File* dengan ekstensi *.rtf memiliki tingkat rasio kompresi yang paling tinggi. *File* dengan ekstensi *.htm memiliki tingkat rasio paling rendah. Tingkat rasio yang diperoleh dari proses kompresi dipengaruhi oleh keragaman karakter penyusun setiap *file* itu sendiri.

Kata kunci : kompresi, teks, karakteristik himpunan, bigram.

COMPRESSION TEXT FILE BASED ON SET CHARACTERISTICS WITH BIGRAM MODEL

ABSTRACT

Characteristics of the set can be used as one method for compression of text files. The purpose of this study was to determine the level of compression ratio based on set characteristics of the text files with the bigram model. Compression process begin by breaking down the contents of a file using bigram model, data dictionary creation, changing the contents of the file index according to the dictionary index, changing the file index to binary and conversion binary into decimal. If the compressed file is converted to the original file, then do the conversion process into binary, cut binary decompress accordance with the highest index dictionary length binary data, then conversion to decimal and be matched with a value encode the data dictionary. Then encodes value matched with string data dictionary and generate the original file.

The results showed that by using an algorithm based on the characteristics of the set with a bigram model, the resulting percentage of average compression ratio for files with the extension *. txt for 25.95%, 26.44 & for *. htm and *. rtf of 26.95%. Files with the extension *. rtf has the highest level of the compression ratio. Files with the extension *. htm to have the lowest ratios. Rate ratios obtained from the compression process is influenced by the diversity of characters making up each file.

Keywords: compression, text, the characteristics of the set, bigram.

KATA PENGANTAR

Segala puji dan syukur penulis persembahkan kehadirat Tuhan YME karena atas segala limpahan rahmat, karunia dan berkat-Nya penulis dapat menyelesaikan skripsi dengan judul Kompresi *File* Teks Berdasarkan Karakteristik Himpunan Dengan Model Bigram.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak, untuk itu penulis mengucapkan terima kasih kepada:

1. Edy Santoso, SSi., M.Kom., selaku Dosen Pembimbing I dan Nurul Hidayat, S. Pd., MSc., selaku Dosen Pembimbing II atas ilmu, bimbingan, kritik dan saran yang diberikan kepada penulis selama penyusunan skripsi ini.
2. Dr. Abdul Rouf Alghofari, M.Sc., selaku Ketua Jurusan Matematika Universitas Brawijaya.
3. Reza Andria S, ST., selaku Dosen Penasehat Akademik yang telah memberikan arahan kepada penulis selama menempuh studi di Jurusan Matematika Universitas Brawijaya.
4. Drs. Marji, MT., selaku Ketua Program Studi Ilmu Komputer Jurusan Matematika Universitas Brawijaya.
5. Kedua orang tua dan kakakku yang selalu memberikan doa dan dukungan tak henti-hentinya.
6. Seluruh teman-teman dan semua pihak yang telah membantu dalam penulisan skripsi ini.

Penulis menyadari bahwa dalam skripsi ini masih terdapat banyak kekurangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun dari pembaca, dan semoga skripsi ini dapat memberikan sumbangan yang bermanfaat bagi perkembangan ilmu pengetahuan.

Malang, 3 Agustus 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang pesat telah menjadi peran penting dalam kehidupan terutama untuk pertukaran data informasi. Pertukaran data berhubungan erat dengan sistem transmisi elektronik dari satu terminal komputer ke terminal komputer yang lain. Ukuran data adalah salah satu faktor yang mempengaruhi proses pertukaran data tersebut. Data dengan ukuran besar membutuhkan waktu *transfer* yang lebih lama dibandingkan dengan data yang lebih kecil dan membutuhkan media penyimpanan yang besar pula. Oleh sebab itu ukuran data perlu dilakukan pemampatan (kompresi), sehingga data berukuran lebih kecil dari ukuran semula tanpa mengurangi isi dari data tersebut (Winanti, 2006).

Menurut (Munir, 2008), kompresi adalah proses pengubahan sekumpulan data menjadi bentuk kode dengan tujuan untuk menghemat kebutuhan tempat penyimpanan data dan waktu untuk transmisi data. Kompresi data juga diartikan sebagai proses meminimalkan ukuran data atau berkas dengan mengurangi data yang berulang. Jika data tersebut akan digunakan kembali, maka harus dilakukan proses dekompresi yaitu dekompresi merupakan proses untuk mengembalikan data baru yang telah dihasilkan oleh proses kompresi menjadi data awal (Hananto, 2006).

Metode-metode kompresi data secara keseluruhan dapat dibagi ke dalam dua kelompok besar yakni kelompok metode kompresi data boleh-hilang (*lossy compression*) dan metode kompresi data tak-hilang (*lossless compression*) (Jean, 1996). Metode *lossy compession* adalah suatu metode kompresi data yang menghilangkan sebagian informasi data dari *file* asli dengan tidak menghilangkan informasi yang ada dalam *file* secara keseluruhan. Hasil dekompresi dari data hasil kompresi tidak tepat sama persis (Zulen, 2010). Metode *lossless compression* adalah metode kompresi data yang mana tidak ada informasi data yang hilang atau berkurang jumlahnya selama proses kompresi. Setelah proses dekompresi jumlah bit (*byte*) data atau informasi dalam keseluruhan *file* hasil sama persis dengan *file* aslinya.

Penelitian ini mengkaji tentang kompresi *file* teks berdasarkan karakteristik himpunan. Metode yang digunakan berdasarkan pada sifat himpunan matematika, dimana himpunan adalah sekumpulan objek yang mempunyai syarat tertentu dan jelas yang setiap elemennya hanya muncul satu kali (Didi, 2010). Terdapat tiga fase dalam proses kompresi *file* teks berdasarkan karaktersitik himpunan yaitu fase pembentukan kamus data, fase *encoding* dan fase *decoding*.

Prinsip kompresi *file* teks berdasarkan karaktersitik himpunan adalah karakter penyusun dari sebuah *file* di-*encode* mengacu kamus data yang telah terbentuk. Suatu *file* teks terbentuk dari berbagai macam karakter yang dapat dicetak dan diorganisasi menjadi baris-baris teks (Jatnika, 2011). Apabila dalam *file* tersebut terdapat karakter penyusunnya yang muncul lebih dari satu kali, maka dapat dilakukan proses kompresi agar ukuran *file* tersebut lebih kecil.

Pada penelitian ini dilakukan kompresi *file* teks berdasarkan karakteristik himpunan dengan model bigram. Bigram merupakan potongan sebanyak 2 karakter pada sebuah rangkaian teks tertentu. Bigram-bigram yang sering berulang dalam penulisan teks dapat dimanfaatkan untuk menambah besarnya rasio kompresi (Wibisana, 2008). Model bigram diterapkan pada pembentukan kamus data *file*. Kamus data berisi karakter pembentuk *file* yang digunakan untuk proses *encode* isi *file*. Metode bigram merupakan salah satu metode yang efektif untuk kompresi file teks dibanding tipe n-gram yang lain karena peluang perulangannya paling besar.

1.2 Rumusan Masalah

Rumusan masalah dalam skripsi ini adalah:

1. Bagaimana implementasi kompresi *file* teks menggunakan karakteristik himpunan dengan model bigram.
2. Bagaimana tingkat kompresi metode tersebut pada pengkompresian *file* teks.

1.3 Batasan Masalah

Batasan masalah dalam skripsi ini antara lain:

1. *File* yang digunakan dalam penelitian ini adalah *file* yang berekstensi *.txt, *.htm dan *.rtf.
2. *File* yang digunakan adalah *file* asli, bukan hasil konversi *file* menjadi *file* dengan ekstensi *.txt, *.htm atau *.rtf.

1.4 Tujuan Penelitian

Tujuan penelitian dalam skripsi ini adalah :

1. Mengimplementasikan algoritma yang didasarkan pada karakteristik himpunan dengan model bigram pada suatu *file* teks.
2. Menghitung tingkat rasio kompresi algoritma tersebut dalam melakukan kompresi *file* teks.

1.5 Manfaat Penelitian

Manfaat yang bisa diambil dari tugas akhir ini adalah :

1. Menghasilkan sistem aplikasi komputer yang berfungsi untuk mengkompresi *file* teks (memperkecil ukuran *file* teks) yang didasarkan pada suatu *file* teks.
2. Memperoleh *file* dengan ukuran lebih kecil sehingga menghemat media penyimpanan.

1.6 Metodologi Penelitian

1. Studi Kepustakaan
   Studi ini dilakukan dengan cara mempelajari beberapa literatur dan artikel mengenai kompresi *file* teks. Selain itu juga mempelajari program aplikasi yang sudah ada sebagai acuan dalam perencanaan dan pembuatan Skripsi ini untuk memberikan gambaran yang jelas mengenai aplikasi kompresi *file* teks.

2. Perancangan dan Pembuatan Program
   Dalam hal ini untuk melakukan percobaan dengan program komputer.
3. Pengujian Perangkat Lunak
   Melakukan uji coba *software* pada beberapa data (*file* teks) yang berbeda.
4. Pembuatan Laporan
   Membuat laporan tertulis mengenai Skripsi ini.

1.7 Sistematika Penulisan

BAB I : PENDAHULUAN

Dalam bab ini menjelaskan pentingnya kompresi *file* teks khususnya untuk menghemat waktu dan tempat penyimpanan untuk transmisi data yang termuat dalam latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

BAB II : DASAR TEORI

Dalam bab ini menjelaskan mengenai dasar-dasar teori yang menjadi acuan dalam proses pembuatan aplikasi kompresi *file* teks.

BAB III : METODE DAN PERANCANGAN

Dalam bab ini menjelaskan tentang metode yang digunakan dan tahapan-tahapan perhitungan kompresi *file* teks.

BAB IV : IMPLEMENTASI DAN PEMBAHASAN

Dalam bab ini menjelaskan mengenai implementasi program, uji coba dan analisanya.

BAB V : PENUTUP

Dalam bab ini berisi kesimpulan dan saran.

BAB II
TINJAUAN PUSTAKA

2.1 Himpunan

Himpunan adalah kumpulan dari obyek-obyek yang mempunyai sifat tertentu dan didefinisikan secara jelas. Kumpulan ini dapat berupa daftar, koleksi atau kelas. Sedangkan obyek-obyek dalam kumpulan dapat berupa benda, orang, bilangan-bilangan atau huruf. Obyek-obyek ini disebut anggota, unsur atau elemen dari himpunan tersebut (Noeryanti, 2010).

Nama himpunan ditulis menggunakan huruf besar, misalnya S, A, atau B, sementara elemen himpunan ditulis menggunakan huruf kecil (a, c, z). Format penulisan himpunan ditunjukkan pada Tabel 2.1.

Tabel 2.1 Penulisan Himpunan

	NOTASI	CONTOH
Himpunan	huruf besar	S
Elemen Himpunan	huruf kecil (jika merupakan huruf)	a
Kelas	huruf tulisan tangan	C

Himpunan-himpunan bilangan seperti bilangan kompleks, riil, bulat dan sebagainya, menggunakan notasi khusus yang ditunjukkan pada Tabel 2.2.

Tabel 2.2 Penulisan bilangan asli, bulat, rasional, riil, dan kompleks

Bilangan	Asli	Bulat	Rasional	Riil	Kompleks
Notasi	N	Z	Q	R	C

Simbol-simbol khusus yang digunakan dalam teori himpunan ditunjukkan pada Tabel 2.3.

Tabel 2.3 Simbol-simbol khusus dalam teori himpunan

SIMBOL	ARTI
$\{\ \}$ atau Φ	Himpunan kosong
$\cup$	Operasi gabungan dua himpunan
$\cap$	Operasi irisan dua himpunan
$\subseteq$, $\subset$	Subhimpunan, Subhimpunan sejati
$\supseteq$, $\supset$	Superhimpunan, Superhimpunan sejati
A^C	Komplemen
$P(A)$	Himpunan kuasa

2.1.1 Fungsi Karakteristik

Fungsi karakteristik menunjukkan apakah sebuah elemen terdapat dalam sebuah himpunan atau tidak (Kustiawan, 2011) dan merupakan fungsi yang dimiliki oleh suatu peubah acak X baik diskrit maupun peubah acak kontinu (Rosida, 2006).

$$\chi_A(x) = \begin{cases} 1, & \text{jika } x \in A \\ 0, & \text{jika } x \notin A \end{cases} \qquad (2.1)$$

Jika A={ apel, jeruk, mangga, pisang} maka:

χ_A(apel) = 1
χ_A(durian) = 0
χ_A(utara) = 0
χ_A(pisang) = 1
χ_A(singa) = 0

2.2 Bit

Bit adalah singkatan dari *Binary Digit* (digit bilangan biner) yang merupakan unit terkecil dalam penyimpanan dan komunikasi informasi dalam teori komputasi dan informasi digital (Proboyekti, 2010). Bit sebagai sebuah satuan adalah jumlah informasi yang dapat dibawa oleh dua pilihan yang mempunyai kemungkinan yang sama. Bit dalam komputer ditetapkan sebagai variabel kuantitas yang

memiliki dua nilai kemungkinan. Dua nilai kemungkinan ini kemudian dinotasikan dalam angka numerikal 0 dan 1. Tentang bit ditunjukkan dalam Tabel 2.5.

Table 2.5 Bit

No.	Bit	Jumlah Bit
1	0	1 bit
2	1	1 bit
3	110	3 bit
4	10011101	8 bit

2.3 Byte

Byte adalah singkatan dari *binary term* yaitu satu unit data yang terdiri dari 8 bit. Satu *byte* dapat merepresentasikan satu karakter tunggal seperti huruf, angka atau tanda baca. MByte sama dengan 1 juta *byte* atau 1.048.576 *byte* (Sumari, 2010). Tabel 2.6 menunjukkan tentang *byte*.

Tabel 2.6 *Byte*

Karakter	Bit	Byte		Karakter	Bit	Byte
A	01000001	65		1/4	10111100	188
B	01000010	66		.	00101110	46
C	01000011	67		:	00111010	58
a	01100001	97		$	00100100	36
b	01100010	98		\	01011100	92

2.4 ASCII

ASCII (*American Standard Code for Information Interchange*) merupakan sistem coding karakter 7 bit. ASCII adalah karakter-karakter yang dijadikan standar untuk pertukaran informasi secara internasional (Proboyekti, 2010). Jumlah karakter ASCII adalah 256 karakter yang terdiri dari alpabet a-z, A-Z, 0-9, dan simbol-simbol. Masing-masing karakter diwakili oleh suatu bilangan. Tabel 2.7 menunjukkan kode ASCII 7 bit.

Tabel 2.7 Kode ASCII

Posisi Bit			7	0	0	0	0	1	1	1	1
			6	0	0	1	1	0	0	1	1
			5	0	1	0	1	0	1	0	1
4	3	2	1								
0	0	0	0	NUL	DLE	SP	0	@	P	`	P
0	0	0	1	SOH	DC1	!	1	A	Q	a	Q
0	0	1	0	STX	DC2	"	2	B	R	b	R
0	0	1	1	ETX	DC3	#	3	C	S	c	S
0	1	0	0	EOT	DC4	$	4	D	T	d	T
0	1	0	1	ENQ	NAK	%	5	E	U	e	U
0	1	1	0	ACK	SYN	&	6	F	V	f	V
0	1	1	1	BEL	ETS	'	7	G	W	g	W
1	0	0	0	BS	CAN	(	8	H	X	h	X
1	0	0	1	HT	EM	)	9	I	Y	i	Y
1	0	1	0	LF	SUB	*	:	J	Z	j	Z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS	-	=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	_	o	DEL

2.5 Sistem Bilangan

System bilangan (*number system*) adalah suatu cara untuk mewakili besaran dari suatu item fisik. Sistem bilangan yang banyak digunakan oleh manusia adalah *system* bilangan desimal, yaitu sistem bilangan yang menggunakan 10 macam simbol untuk mewakili suatu besaran (Firdausy, 2010).

2.5.1 Biner

Biner adalah bilangan yang menggunakan dua simbol (0,1). Bilangan biner juga disebut bilangan berbasis 2, setiap biner digit disebut bit (Arifia, 2010). Tabel 2.8 menunjukkan bilangan biner.

Tabel 2.8 Biner

Desimal	Biner	Base ^{Exponent}
1	1	$(1x2^0)$
50	110010	$(0x2^0) + \ldots + (1x2^5)$
100	1100100	$(0x2^0) + \ldots + (1x2^6)$
200	11001000	$(0x2^0) + \ldots + (1x2^7)$

Cara yang dipakai untuk mengonversi bilangan desimal ke biner dengan pembagian ulang angka desimal oleh 2, menghasilkan deretan dari sisa 0 atau 1. Deretan sisa tersebut bila dibaca dari arah terbalik akan menghasilkan angka biner ekivalen dari angka desimal yang dikonversikan (Arifia, 2010).

Contoh :

126des = ? bin

	Hasil Bagi	Nilai Sisa	
126 / 2 =	63	0	
63 / 2 =	31	1	
31 / 2 =	15	1	Urutkan
15 / 2 =	7	1	
7 / 2 =	3	1	
3 / 2 =	1	1	
1 / 2 =	→	1	

Dengan menuliskan nilai sisa mulai dari bawah ke atas, diperoleh angka biner 1111110bin**.**

Konversi bilangan biner ke desimal dilakukan dengan menjumlahkan hasil perkalian semua bit biner dengan beratnya.

Contoh :

1010011 = 83

1010011bin = 1 X 2^6 + 0 X 2^5 + 1 X 2^4 + 0 X 2^3 + 0 X 2^2 + 1 X 2^1 + 1 X 2^0

= 64+0+16+0+0+2+1

= 83

Dari contoh tersebut, dapat diperoleh rumus umum untuk mendapatkan nilai desimal dari radiks bilangan tertentu (Anggraini, 2010):

$$(N)r = [(d_0 \times r^0) + (d_1 \times r^1) + (d_2 \times r^2) + ... + (d_n \times r^n) +]_{10} ... \quad (2.2)$$

Dimana: N = Nilai
r = Radiks
d_0, d_1, d_2 = digit dari yang terkecil (paling kanan) untuk d_0

2.6 Bigram

Model bigram adalah model n-gram yang hanya melibatkan dua buah elemen (Sukamto, 2009). Model N-gram adalah sebuah tipe model probabilistik untuk memperkirakan elemen selanjutnya pada sebuah urutan. N-gram digunakan untuk berbagai area statistik dari pemrosesan bahasa alami dan analisis urutan genetik. Sebuah n-gram adalah sebuah sub-urutan dari sejumlah n elemen dari urutan yang diberikan.

Contoh bigram yang diterapkan pada rangkaian karakter "mama_menanak_nasi" diperoleh bigram "ma", "ma", "_m", "en", "an", "ak", "_n", "as", "i".

Berdasarkan hasil penelitian Wibisana (2008) mengenai kompresi *short messsage service* menggunakan *huffman coding* dan perulangan bigram, penambahan metode bigram dalam kode Huffman dapat menambah besarnya rasio kompresi dengan rata-rata penambahan sebesar 5,85%.

2.7 Kompresi

Kompresi adalah pengubahan data yang berupa kumpulan karakter menjadi bentuk kode dengan tujuan untuk menghemat kebutuhan tempat penyimpanan dan waktu transmisi data (Callista, 2010). Menurut Anton (2005), kompresi adalah proses mengkodekan informasi menggunakan bit lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem encoding tertentu.

Pengiriman data hasil kompresi dapat dilakukan jika pihak pengirim atau yang melakukan kompresi dan pihak penerima memiliki aturan yang sama dalam hal kompresi data. Pihak pengirim

harus menggunakan algoritma kompresi data yang sudah baku dan pihak penerima juga menggunakan teknik dekompresi data yang sama dengan pengirim sehingga data yang diterima dapat dibaca atau di-dekode kembali dengan benar (Anton, 2005).

2.7.1 Jenis Teknik Kompresi

Teknik kompresi dapat diklasifikasikan menjadi tiga, yaitu (Restyanditi, 2008):teknik kompresi yang

1. *Entropy coding* adalah teknik kompresi yang menggunakan proses *lossless*. Teknik ini berdasarkan urutan data dan tidak memperhatikan semantik data (arti data).
2. *Source coding* adalah teknik kompresi dengan menggunakan proses *lossy*. Teknik ini berkaitan dengan data semantik dan media.
3. *Hybrid coding* adalah teknik kompresi dengan menggunakan kombinasi atau gabungan dari metode *entrophy coding* dan *source coding*.

Teknik kompresi data dapat dibagi menjadi dua kategori besar, yaitu (Widhiartha, 2008):

1. *Lossy Compression* menyebabkan adanya perubahan data dibandingkan sebelum dilakukan proses kompresi. Sebagai gantinya *lossy compression* memberikan derajat kompresi lebih tinggi. Tipe ini cocok untuk kompresi *file* suara digital dan gambar digital. *File* suara dan gambar secara alamiah masih bisa digunakan walaupun tidak berada pada kondisi yang sama sebelum dilakukan kompresi.
2. *Lossless Compression* memiliki derajat kompresi yang lebih rendah tetapi dengan akurasi data yang terjaga antara sebelum dan sesudah proses kompresi. Kompresi ini sesuai untuk basis data, dokumen atau *spreadsheet*. Pada *lossless compression* ini tidak diijinkan ada bit yang hilang dari data pada proses kompresi.

Menurut penelitian Anggraini (2010) mengenai kompresi *file* teks berdasarkan karakteristik himpunan, diperoleh hasil bahwa semakin besar tingkat rasio, maka ukuran *file* kompresi yang dihasilkan akan semakin kecil. *File* teks dengan ekstensi **.htm* merupakan *file* dengan tingkat rasio paling besar dengan jumlah rasionya mencapai 2,24 (44%) dan *file* teks berekstensi **.pas* memiliki tingkat rasio yang paling kecil sebesar 1,09 (8%).

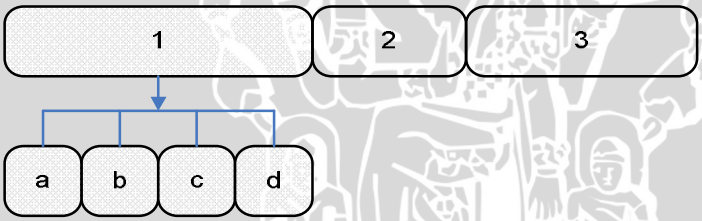
2.7.2 Rasio Kompresi

Proses kompresi merupakan proses encoding yang menghasilkan data terkompresi (aliran data encoded). Tingkat pengurangan data yang didapatkan dari hasil proses kompresi disebut rasio kompresi. Rasio kompresi adalah perbandingan antara panjang data asli dengan panjang data yang sudah dikompresi. Persamaan rasio ditunjukkan dalam persamaan 2.2 (Syarif, 2010):

$$Rasio = (1 - (\frac{Ukuran\_kompresi}{Ukuran\_asli}) * 100\% \quad (2.3)$$

2.8 Struktur File Kompresi

File kompresi terdiri dari dua bagian yaitu *header file* dan data kompresi. *Header file* berisi informasi mengenai data kompresi. Informasi tersebut berguna untuk proses dekompresi. Sehingga *file* hasil ekstrak sesuai dengan *file* aslinya (Anggraini, 2010). Bagian *file* dapat dilihat pada gambar 2.1.



Gambar 2.1 Struktur *file* kompresi.

Keterangan Gambar 2.1 :

1. Bagian *header file*

   Bagian *header file* terdiri atas 4 bagian, yaitu:

   a. Ekstensi *file* asli
   b. Jumlah blok data
   c. Panjang kamus
   d. Panjang data terkompresi

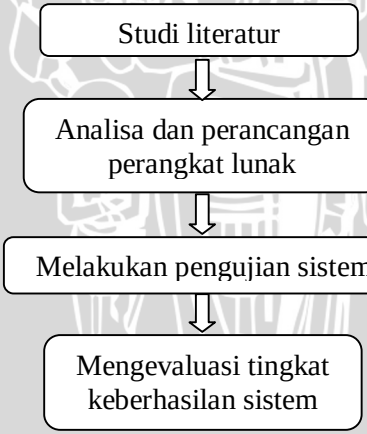
2. Data kamus, yang berisi jenis karakter penyusun *file* teks.
3. Data kompresi, yang berisi nilai hasil kompresi dari algoritma yang didasarkan pada karakteristik himpunan pada suatu *file.*

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini akan dibahas metode dan rancangan yang digunakan, serta langkah-langkah yang dilakukan dalam pembuatan perangkat lunak untuk melakukan kompresi *file* menggunakan algoritma yang didasarkan pada karakteristik himpunan dengan model bigram. Tahapan pembuatannya adalah sebagai berikut :

1. Mempelajari literatur yang terkait dengan kompresi *file* teks dan bigram.
2. Menganalisa dan merancang perangkat lunak dengan metode yang digunakan.
3. Membuat perangkat lunak berdasarkan analisis dan perancangan yang dilakukan.
4. Uji coba perangkat lunak.
5. Mengevaluasi tingkat keberhasilan system dalam melakukan kompresi.

Langkah-langkah yang dilakukan dapat dilihat pada Gambar 3.1.



Gambar 3.1 Langkah-langkah penelitian

3.1 Analisa Perangkat Lunak

Pada subbab ini akan dibahas mengenai semua hal yang diperlukan dalam proses pembuatan kompresi pada *file*.

3.1.1 Deskripsi Umum Perangkat Lunak

Perangkat lunak yang akan dibuat berupa aplikasi kompresi dan dekompresi yang dapat digunakan oleh user untuk mengkompresi *file* teks. Hasil yang diperoleh yaitu *file* yang terkompresi semaksimal mungkin dengan tujuan menghemat tempat pada suatu media penyimpanan.

Metode yang digunakan dalam perangkat lunak ini menggunakan algoritma yang didasarkan pada karakteristik himpunan dengan model bigram. Model bigram berfungsi untuk pembentukan kamus karakter yang diperoleh dari jenis karakter penyusun *file*.

Proses-proses yang dilakukan untuk mengkompresi *file* adalah sebagai berikut :

1. User memasukkan *file* yang akan dikompresi.
2. Pembentukan kamus karakter dengan model bigram yang diperoleh dari karakter unik penyusun *file*.
3. Setiap karakter penyusun *file* yang di-*encode* mengacu pada isi kamus karakter yang telah terbentuk.

3.1.2 Batasan Perangkat Lunak

Batasan dari perangkat lunak yang akan dikembangkan adalah:

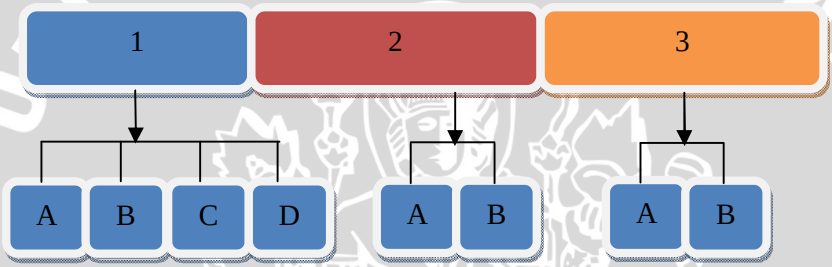
1. Pengkompresian hanya dilakukan pada sebuah *file* teks tunggal.
2. *File* yang digunakan dalam penelitian ini adalah *file* yang berekstensi *.txt, *.htm dan *.rtf.
3. *File* yang digunakan adalah *file* asli, bukan hasil konversi *file* menjadi *file* dengan ekstensi *.txt, *.htm atau *.rtf.

3.2 Perancangan Perangkat Lunak

Berdasarkan analisa yang telah dilakukan, berikut ini akan dibahas mengenai arsitektur dan proses yang terjadi pada perangkat lunak yang akan dibangun.

3.2.1 Perancangan Struktur *File* Kompresi

Struktur *file* kompresi terdiri dari tiga bagian, yaitu *header file*, data kamus dan data kompresi yang memodifikasi struktur file yang telah dijelaskan pada bab 2. Informasi tentang data kompresi disimpan di *header file* yang berfungsi untuk mengembalikan (*dekompresi*) *file* kompresi menjadi *file* aslinya. Struktur *file* kompresi ditunjukkan pada Gambar 3.2.



Gambar 3.2 Struktur *file* kompresi

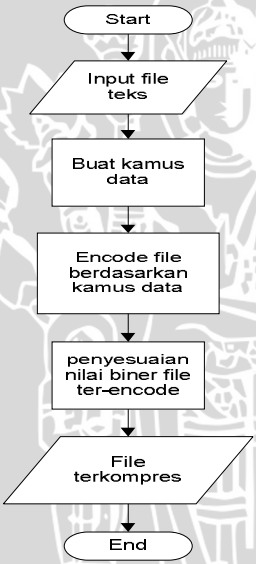
Keterangan gambar :

1. *Header file*
   Header file terdiri dari 4 bagian, yaitu :
   a. Ekstensi *file* asli.
   b. Panjang biner penyesuaian.
   c. Panjang file yang telah berubah menjadi model bigram.
   d. Panjang biner penyesuaian.
   e. Panjang sisa biner penggabungan
2. Data kamus
   Data kamus terdiri dari :
   a. Karakter unik penyusun *file* dengan model bigram.
   b. Panjang data kamus

3. Data kompresi
   Data kompresi terdiri dari :
   a. Isi hasil kompresi.
   b. Panjang hasil kompresi

3.2.2 Perancangan Proses Kompresi

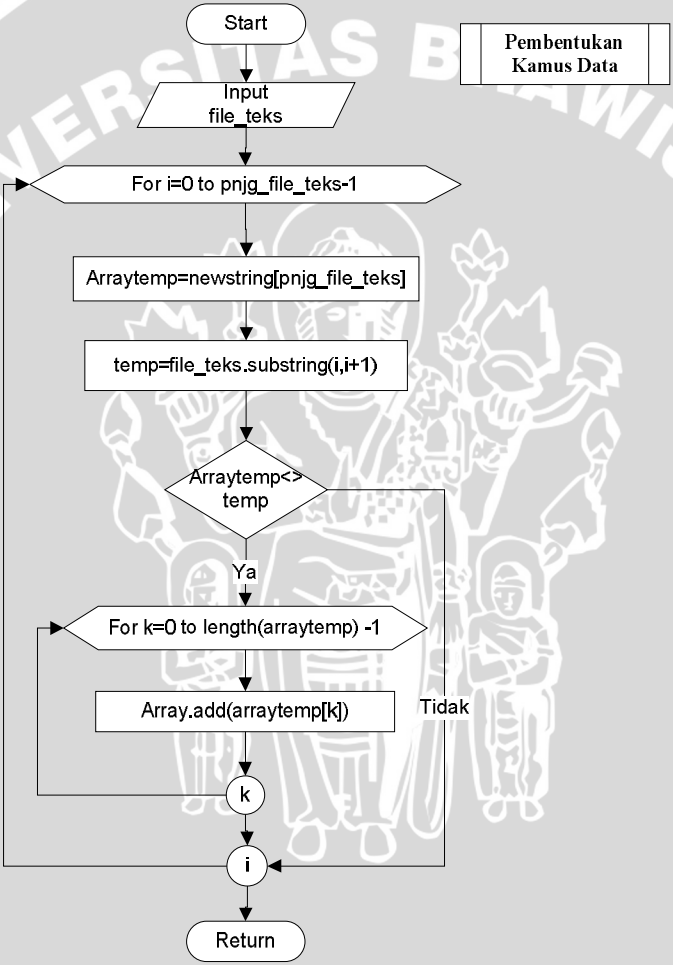
Proses kompresi bertujuan untuk mengubah ukuran suatu file menjadi lebih kecil sehingga menghemat kebutuhan tempat penyimpanan dan waktu transmisi data. Metode yang dapat digunakan untuk kompresi data salah satunya dengan menggunakan karakteristik himpunan dengan model bigram. Proses kompresi berdasarkan karakteristik himpunan dengan model bigram dapat dijelaskan pada Gambar 3.3.



Gambar 3.3 Proses Kompresi

3.2.2.1 Pembentukan Kamus Data

Proses kompresi diawali dengan pembentukan kamus data dari *file* teks yang didasarkan pada model bigram. Setelah terbentuk kamus data, maka dilakukan proses kompresi dengan mengacu pada kamus data tersebut.



Gambar 3.4 *Flowchart* proses pembentukan kamus_data

Langkah-langkah untuk pembentukan kamus data sebagai berikut:

1. Input data teks.
2. Lakukan pencarian bigram dengan melakukan *looping* pada seluruh *file* teks.
3. Dilihat isi temp dan arraytemp, jika sama masukkan bigram pada arraytemp. Jika tidak sama maka arraytemp sama dengan temp.
4. Output array kamus.
   Proses pembentukan kamus secara keseluruhan dijelaskan pada Gambar 3.4.

3.2.2.2 Proses *Encode* Data

Proses *encode* data mengacu pada kamus data yang telah terbentuk. Hasil yang didapatkan dari proses ini adalah isi *file* berubah menjadi bentuk angka sesuai dengan indeks dari kamus data. Langkah-langkah untuk proses *encode* data sebagai berikut:

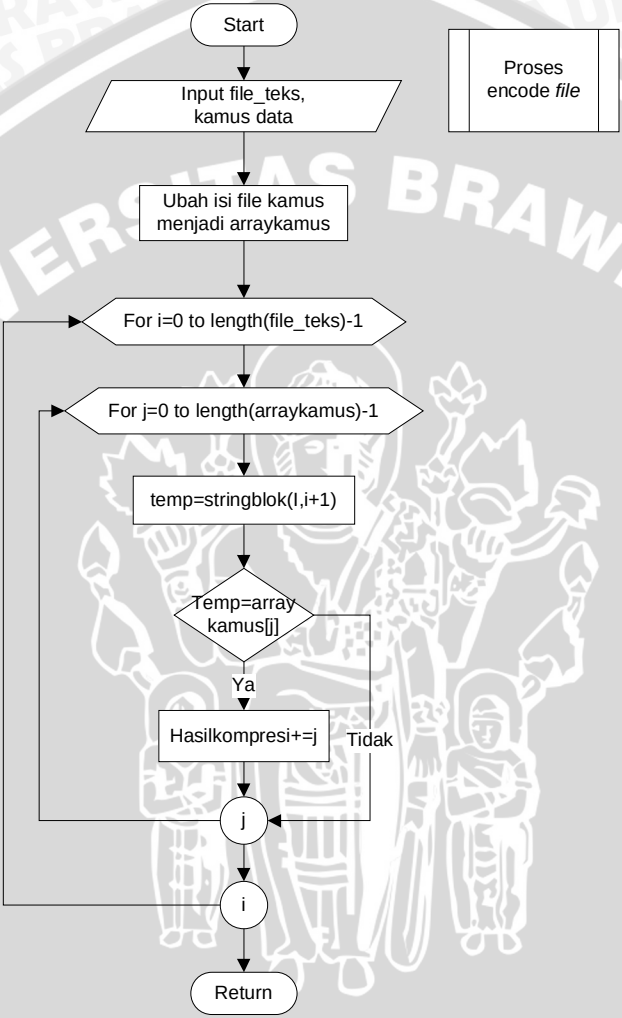
1. Input *file* teks dan kamus data.
2. Lakukan pengubahan isi *file* kamus menjadi bentuk array.
3. Dilihat isi kamus data dan isi *file* teks, apabila sama maka ubah isi *file* teks dengan indeks kamus data. Jika tidak sama maka lakukan langkah selanjutnya.
4. Output *file* ter-*encode*.
   Proses encode data secara keseluruhan dijelaskan pada Gambar 3.5.

3.2.2.3 Proses Penyesuaian Biner

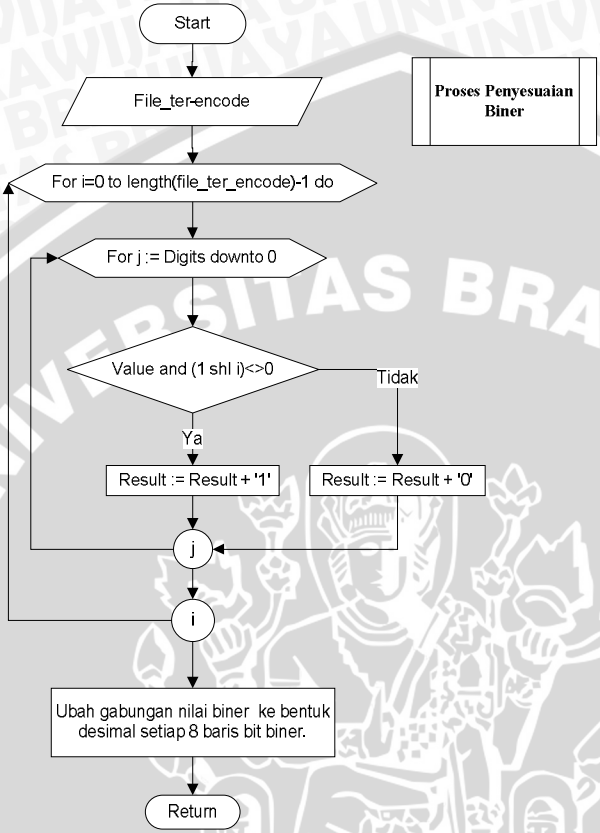
Setelah diperoleh *file* ter-*encode,* langkah selanjutnya adalah penyesuaian nilai biner. Isi *file* ter-*encode* diubah ke dalam bentuk biner dan disesuaikan panjangnya. Langkah-langkah untuk proses penyesuaian biner sebagai berikut:

1. Input *file* ter-*encode.*
2. Ubah isi *file* ter-*encode* menjadi bentuk biner sesuai dengan *value.*
3. Jika panjang biner tidak sama dengan panjang biner indeks tertinggi, maka ubah panjang biner sesuai dengan panjang biner indeks tertinggi.
4. Ubah gabungan nilai biner yang telah disesuaikan menjadi bentuk desimal setiap delapan baris bit biner.

5. Output *file* kompresi
   Proses penyesuaian biner secara keseluruhan dijelaskan pada Gambar 3.6.



Gambar 3.5 *Flowchart* proses *encode* file



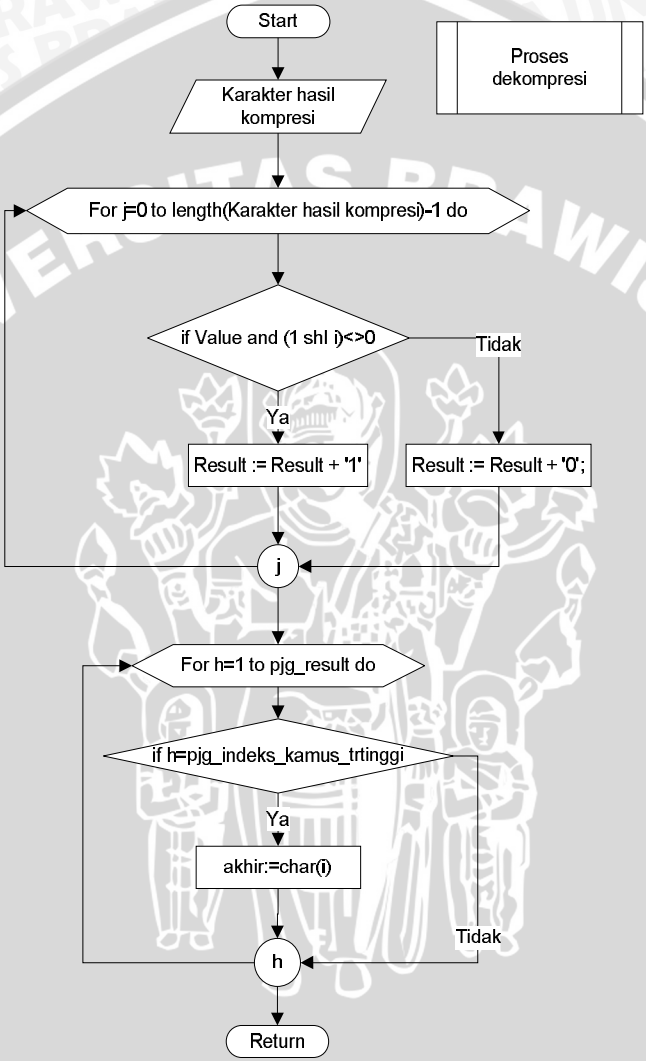
Gambar 3.6 *Flowchart* proses penyesuaian biner hasil kompresi dan pengubahan kebentuk desimal

3.2.3 Perancangan Proses Dekompresi

Proses dekompresi perlu dilakukan untuk membaca isi *file* terkompresi yang bertujuan untuk menghasilkan *file* asli. Langkah-langkah proses dekompresi sebagai berikut:

1. Input *file* kompresi.
2. Ubah isi *file* kompresi ke dalam bentuk biner.
3. Ambil nilai biner sebanyak panjang biner indeks tertinggi dan ubah ke dalam bentuk desimal. Nilai desimal sama dengan nilai dari indeks kamus data.
4. Lakukan pengambilan isi kamus data sesuai dengan nilai desimal yang dihasilkan.

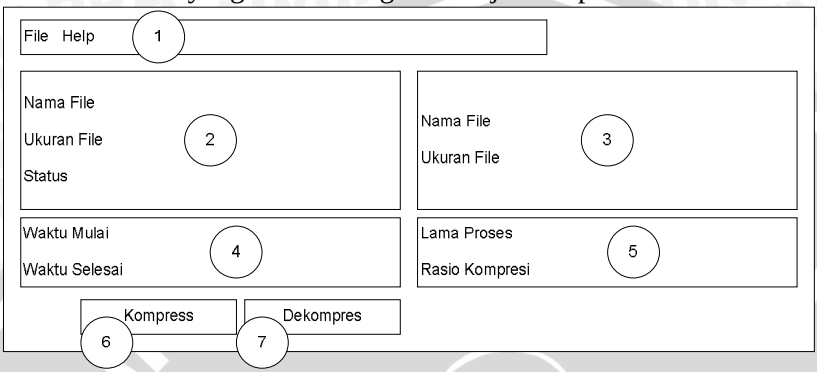
5. Output *file* dekompresi.
   Proses dekompresi secara keseluruhan dijelaskan pada Gambar 3.7.



Gambar 3.7 *Flowchart* proses dekompresi.

3.3 Rancangan Antar Muka Sistem

Antarmuka yang akan dibangun ditunjukkan pada Gambar 3.8.



Gambar 3.8 Rancangan antarmuka aplikasi Kompresi *File* Teks

Berdasarkan Karakteristik Himpunan dengan Model Bigram Keterangan bagian-bagian dalam rancangan antarmuka adalah sebagai berikut:

1. *Main* menu berisi menu *File* dan *Help*. Menu *File* digunakan untuk membuka *file* yang akan diproses. Menu *Help* berisi petunjuk atau keterangan tentang aplikasi.
2. *Frame* 2 berisi keterangan tentang *file* input yang akan dikompresi ataupun dekompresi, yaitu berupa nama *file*, ukuran *file* dan status *file* (*compress* atau *uncompress*).
3. *Frame* 3 berisi keterangan tentang *file* output hasil dari proses kompesi maupun dekompresi, yaitu nama *file* disertai direktori penyimpanannya dan ukuran *file*.
4. *Frame* 4 menampilkan waktu ketika proses mulai dijalankan serta waktu ketika proses berakhir.
5. *Frame* 5 menampilkan waktu yang dibutuhkan selama proses berlangsung.
6. *Button* Compress berfungsi untuk melakukan kompresi *file* yang telah dipilih oleh *user*.
7. *Button* Dekompress berfungsi untuk men-dekompres *file*

3.4 Rancangan Uji Coba dan Evaluasi Hasil

File hasil uji coba sistem kompresi dan dekompresi digunakan untuk evaluasi kinerja sistem. Tujuan dari uji coba ini adalah untuk mengetahui tingkat keberhasilan dari penerapan algoritma yang berdasarkan karakteristik himpunan dengan model bigram pada suatu *file* teks.

3.4.1 Rancangan Uji Coba

Uji coba terhadap sistem dilakukan dengan meng-*input*-kan *file* teks dengan berbagai tipe *file* dan ukuran. Uji coba ini dilakukan untuk mengetahui pengaruh tipe dan ukuran *file* terhadap hasil kompresi yang didapatkan.

3.4.2 Analisa dan Evaluasi Hasil

Hasil yang didapatkan dari proses kompresi akan dievaluasi dengan memperhatikan tipe *file*, ukuran dan rasio kompresi yang diperoleh. Untuk hasil yang didapatkan pada proses dekompresi, ukuran dan kesesuaian *file* akan dievaluasi dengan memperhatikan *file* asli sebelum dilakukan proses kompresi.

Rancangan tabel untuk hasil dari uji coba yang dilakukan ditampilkan pada Tabel 3.1 dan Tabel 3.2.

Tabel 3.1 Rancangan tabel hasil uji coba untuk proses kompresi

No	Nama *file*	Ukuran *file* asli (*byte*)	Ukuran *file* terkompresi (*byte*)	Rasio Kompresi

Tabel 3.2 Rancangan tabel hasil uji coba untuk proses dekompresi

No	Nama *file*	Ukuran *file* terkompresi (*byte*)	Ukuran *file* hasil dekompresi (*byte*)

3.5 Contoh Perhitungan Manual

Terdapat sebuah *file* teks dengan karakter penyusunnya: "MAMA_DAN_NANA_MENANAK_NASI". Karakter penyusun *file* tersebut terdiri dari 26 karakter yang membentuk kamus data:

0. MA
1. _D
2. AN
3. _N
4. A_
5. ME
6. NA
7. K_
8. SI

Langkah selanjutnya adalah proses *encode* data *file* teks dengan kamus data menjadi acuannya. Hasil yang diperoleh dari proses kompresi dapat ditunjukkan pada Gambar 3.9.

Gambar 3.9 Hasil encode file

Bentuk kompresi tersebut kemudian diubah ke dalam bentuk biner dan dilakukan penyesuaian panjang biner. Kamus data yang diperoleh berjumlah sembilan, sehingga indeks tertingginya adalah delapan. Bentuk biner dari delapan adalah 1000 yang memiliki panjang empat baris bit. Panjang biner dari nilai hasil *encode* setiap karakter disamakan dengan panjang biner indeks tertinggi dari jumlah kamus data yang ditunjukkan pada Tabel 3.3.

Tabel 3.3 Penyesuaian panjang biner

Indeks *File*	Karakter *File*	Nilai Hasil *Encode*	Biner Penyesuaian
0	MA	0	0000
1	MA	0	0000
2	_D	1	0001
3	AN	2	0010
4	_N	3	0011
5	AN	2	0010
6	A_	4	0100
7	ME	5	0101
8	NA	6	0110
9	NA	6	0110
10	K_	7	0111
11	NA	6	0110
12	SI	8	1000

Hasil penyesuaian panjang biner pada Tabel 3.3 kemudian digabungkan ke dalam satu deret baris.

0000+0000+0001+0010+0011+0010+0100+0101+0110+0110+0111+0110+1000

⇩

000000000001001000110010010001010110011001110110100 0

Setelah penggabungan nilai biner, kemudian nilai biner tersebut diubah dalam bentuk desimal. Perubahan ke bentuk desimal dilakukan setiap delapan baris nilai biner.

00000000||00010010||00110010||01000101||01100110||01110110||1000

⇩

Desimal :

0	18	50	69	102	118	8

Setelah perubahan bentuk biner menjadi bentuk desimal diperoleh informasi mengenai data kompresi, yaitu :

**File* asli

Ukuran : (26 karakter * 1 byte) = 26 byte

**File* kompresi

Header

Ekstensi *file* = 1 *byte*

Panjang kamus data = 1 *byte*

Panjang data kompresi = 1 *byte*

Panjang biner penyesuaian = 1 *byte*

Data kamus : 7 jenis karakter penyusun *file* = 10 byte

Data kompresi = 7 *byte*

Ukuran *file* kompresi = {header+data kamus}
= {ekstensi *file*+panjang kamus data+panjang data kompresi+data kamus}
= {1+1+1+1+10+7}
= 20 *byte*

Rasio kompresi dapat dihitung menggunakan persamaan 2.2 pada halaman 15, yaitu:

$$\text{Rasio kompresi} = \left(1 - \frac{\text{ukuran file kompresi}}{\text{ukuran file asli}}\right) * 100\%$$

$$= \left(1 - \frac{21}{26}\right) * 100\%$$

$$= 19\%$$

Sehingga ukuran *file* setelah melewati proses kompresi berkurang 19% dari ukuran *file* asli.

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada Bab ini akan dibahas implementasi sistem dan uji coba sistem serta evaluasi hasil. Implementasi sistem merupakan implementasi rancangan yang sudah dilakukan pada Bab 3. Sedangkan pembahasan adalah penjelasan dari masing-masing implementasi tersebut.

Uji coba dilakukan terhadap sistem pengkompresian sebuah *file* teks dengan menerapkan algoritma yang didasarkan pada karakteristik himpunan dengan model bigram. Hasil dari uji coba ini akan digunakan untuk mengevaluasi tingkat keefektifan yang dilakukan oleh sistem yang telah dibuat.

Perangkat lunak yang digunakan dalam pengembangan sistem pengkompresian ini adalah:

1. Sistem operasi yang digunakan adalah Microsoft Windows XP Professional.
2. *Software* yang digunakan untuk mengembangkan sistem adalah Borland Delphi 7

Perangkat keras yang digunakan untuk mengembangkan sistem pengkompresian sebuah *file* teks dengan menerapkan algoritma yang didasarkan pada karakteristik himpunan dengan model bigram adalah:

1. Prosesor Intel Pentium Dual Core 2.16 Ghz
2. Memori 2 GB
3. Harddisk dengan kapasitas 250 GB

4.1 Implementasi Program

Berdasarkan analisa dan rancangan sistem pada Bab 3, maka pada subbab ini akan dijelaskan implementasi proses-proses tersebut.

4.1.1 Implementasi *Open File*

Pada subbab ini akan dijelaskan implementasi yang digunakan untuk membuka *file*. Implementasi *open file* ini memberikan informasi mengenai *file* yang dibuka, diantaranya informasi mengenai nama *file*, ukuran *file* dan status dari *file*. Kode program ditunjukkan pada Gambar 4.1.

```
procedure TForm1.Open1Click(Sender: TObject);
var
  SearchRec: TSearchRec;
begin
  if OpenDialog1.Execute then
       label7.Caption:=get_file_name(opendialog1.FileName);
if  FindFirst(OpenDialog1.FileName,  faAnyFile,  SearchRec)  =
0 then
   Label8.Caption   :=   FloatToStrF(SearchRec.Size/1048576,
ffFixed, 7, 2)+' MB';
  FindClose(SearchRec);

  get_extention_file(opendialog1.FileName);
   if get_extention_file(opendialog1.FileName)= 'chi' then
        begin
          label9.Caption := 'Compress';
        end
        else
        begin
          label9.Caption := 'Uncompress';
        end;

    if label9.Caption = 'Compress' then
    begin
      Button1.Enabled := False;
      Button2.Enabled := True;
    end
    else
    begin
      Button1.Enabled := True;
      Button2.Enabled := False;
    end;

  end;
```

Gambar 4.1 Implementasi *Open File*.

4.2 Implementasi Prosedur dan Fungsi Kompresi

Pada subbab ini akan dijelaskan beberapa fungsi atau prosedur yang digunakan dalam proses kompresi, yaitu :

4.2.1 Fungsi untuk Memperoleh Nama *File*

get_file_name adalah fungsi yang digunakan untuk mendapatkan nama dari suatu *file*. Proses dari fungi get_file_name adalah mengambil semua karakter dibelakang tanda “/”. Fungsi get_file_name ditunjukkan pada Gambar 4.2.

```
function get_file_name(filez : string):string;
var I : integer;
    s : string;
begin
s := '';
for i:=length(filez) downto 1 do
 begin
  if filez[i] <> '\' then
    begin
      s := s+ filez[i];
    end
  else
    break;
 end;
 result := ReverseString(s);
end;
```

Gambar 4.2 Fungsi get_file_name.

4.2.2 Fungsi untuk Memperoleh *Extention File*

get_extention_file adalah fungsi yang digunakan untuk mendapatkan ekstensi dari suatu *file*. Proses fungsi get_extention_file mengambil semua karakter dibelakang tanda '.'. Fungsi get_extention_file ditunjukkan pada gambar 4.3.

```
function get_extention_file(filez : string):string;
var i : integer;  s : string;
begin
s := '';
for i:=length(filez) downto 1 do
 begin
  if filez[i] <> '.' then
    begin
      s := s+ filez[i];
    end
  else
    break;
 end;
 result := ReverseString(s); end;
```

Gambar 4.3 Fungsi get_extention_file.

4.2.3 Fungsi untuk Mengubah Nilai Biner menjadi Integer

biner_to_integer adalah fungsi yang digunakan untuk mengubah nilai biner ke nilai integer. Proses perubahan dilakukan sepanjang nilai biner yang diinputkan. Fungsi biner_to_integer ditunjukkan pada Gambar 4.4.

```
function biner_to_integer(Value: String): byte;
var
  i, iValueSize: Integer;
begin
  Result := 0;
  iValueSize:= Length(Value);
  for i:= iValueSize downto 1 do
    if Value[i] = '1' then Result := Result+(1 shl
(iValueSize-i));
end;
```

Gambar 4.4 Fungsi biner_to_integer.

4.2.4 Fungsi untuk Mengubah Nilai Integer menjadi Biner dengan Panjang Biner dapat Disesuaikan

integer_to_biner adalah fungsi yang digunakan untuk mengubah nilai integer ke nilai biner. Panjang biner yang dihasilkan dapat disesuaikan dengan value yang diinputkan. Fungsi integer_to_biner ditunjukkan pada Gambar 4.5.

```
function integer_to_biner(Value: byte; Digits: byte):
String; var
 i: Integer;
begin
 Result := '';
 for i := Digits downto 0 do
  if Value and (1 shl i)<>0 then
   Result := Result + '1'
  else
   Result := Result + '0';
end;
```

Gambar 4.5 Fungsi integer_to_biner.

4.2.5 Fungsi untuk Mengubah Nilai Integer menjadi Biner

menjadi_bentuk_biner adalah fungsi yang digunakan untuk mengubah nilai integer menjadi bentuk biner. Fungsi menjadi_bentuk_biner ditunjukkan pada Gambar 4.6

```
function menjadi_bentuk_biner(a : integer): string;
//merubah nilai integer ke bentuk biner
var s : string;
begin
  s := '';
  if a = 0 then
    s := '0'


else
```

```
  begin
  while a > 0 do
   begin
    if a mod 2 = 0 then
     s := s + '0'
    else
     s := s + '1';
    a := a div 2;
   end;
  end;
   result := ReverseString(s);
end;
```

Gambar 4.6 Fungsi menjadi_bentuk_biner

4.2.6 Proses Mengubah Isi *File* menjadi Potongan-potongan Dua Karakter

Proses kompresi diawali dengan merubah isi *file* menjadi bentuk bigram, yaitu teks diubah menjadi potongan-potongan dua karakter. Karakter yang diproses tersebut adalah karakter ASCII dari isi *file*, sehingga keseluruhan proses dilakukan pada karakter ASCII tersebut, bukan pada karakter aslinya. Besarnya *array* untuk penyimpanan hasil pemotongan karakter model bigram adalah setengah dari besarnya *file* asli. Pemotongan dilakukan pada keseluruhan isi teks. Hasil pemotongan kemudian disimpan pada *record file*_kompress. Data yang disimpan pada *record* meliputi penggabungan dua karakter hasil setiap pemotongan, karakter pertama dan karakter kedua hasil pemotongan. Dua karakter hasil pemotongan disimpan di file_gabung2.char2, karakter pertama dan karakter kedua hasil pemotongan disimpan di file_gabung2.byte1 dan gabung2.byte1. Hasil pemotongan tersebut selanjutnya digunakan untuk pembentukan kamus data. Proses perubahan isi *file* menjadi potongan-potongan dua karakter ditunjukkan pada Gambar 4.7.

4.2.7 Proses Pembentukan Kamus Data

Kamus data terdiri dari potongan-potongan dua karakter pembentuk *file* teks. Proses pembentukan kamus data dimulai dengan melihat keseluruhan dari file_gabung2. Isi file_gabung2 yang pertama secara otomatis tersimpan pada kamus. Selanjutnya dilihat pada seluruh isi dari file_gabung2 dengan kamus pertama, apabila ditemukan kesamaan maka yang disimpan adalah yang terdapat pada kamus. Jika tidak ditemukan kesamaan, maka yang disimpan adalah karakter yang terdapat di file_gabung2. Proses ini dilakukan hingga

semua karakter penyusun file_gabung2 tersimpan di kamus data. Isi dari kamus data terdiri dari gabungan dua karakter yang disimpan pada kamus.char_2 dan karakter pembentuk gabungan dua karakter yang disimpan pada kamus.byte1 dan kamus.byte2. Kamus data digunakan untuk mengubah isi *file* menjadi indeks dari isi karakter penyusun kamus data yang sesuai dengan isi *file*. Proses pembentukan kamus data ditunjukkan pada Gambar 4.8.

```
  a:=0;
  while not eof(file_utama) do
   begin
      read(file_utama, b);
      file_ascii[a] := b;
      inc(a);
      Gauge1.Progress:=a;
   end;
  panjang:=0;
  if a mod 2 > 0 then
  begin
    panjang:= (a div 2)+1
  end
  else
  begin
    panjang:= a div 2
  end;
  pjg_file_2:=panjang;SetLength(file_gabung2,
panjang); d:=0; e:=0;
  while d<a do
  begin
    if d=a-1 then
    begin
      file_gabung2[e].char_2:= char(file_ascii[d]);
      file_gabung2[e].byte1:= file_ascii[d];
      file_gabung2[e].byte2:=0;
      d:=d+1;
    end;
    if d<a-1 then
    begin
      file_gabung2[e].char_2:=
char(file_ascii[d])+char(file_ascii[d+1]);
      file_gabung2[e].byte1:= file_ascii[d];
      file_gabung2[e].byte2:= file_ascii[d+1];
      inc(e);
      d:=d+2;
    end;
  end;
```

Gambar 4.7 Proses merubah isi *file* menjadi potongan-potongan dua karakter.

```
 setlength(temp_kamus, pjg_file_2);
  f:=0;
  jmlh_kamus:=1;
  for g:=0 to pjg_file_2-1 do
  begin
    if  g=0 then
    begin
      temp_kamus[f].char_2:=file_gabung2[g].char_2;
      temp_kamus[f].byte1:=file_gabung2[g].byte1;
      temp_kamus[f].byte2:=file_gabung2[g].byte2;
    end
    else
    begin
      ketemu:=true;
      for h:=0 to length(temp_kamus)-1 do
      begin
       if      file_gabung2[g].char_2=temp_kamus[h].char_2
then
        begin
          ketemu:=false;
        end;
      end;

      if ketemu then
      begin
        temp_kamus[f].char_2:=file_gabung2[g].char_2;
        temp_kamus[f].byte1:=file_gabung2[g].byte1;
        temp_kamus[f].byte2:=file_gabung2[g].byte2;
        jmlh_kamus:=jmlh_kamus+1;
      end;
    end;
    f:=f+1;
  end;

  i:=0;
  setlength(kamus, jmlh_kamus);
  for j:=0 to length(temp_kamus)-1 do
  begin
    if temp_kamus[j].char_2<>'' then
    begin
      kamus[i].char_2:=temp_kamus[j].char_2;
      kamus[i].byte1:=temp_kamus[j].byte1;
      kamus[i].byte2:=temp_kamus[j].byte2;
      kamus[i].nilai_encode:=i;
      i:=i+1;
    end;
  end;
```

Gambar 4.8 Proses pembentukan kamus data.

4.2.8 Proses Perubahan Isi file_gabung2 menjadi Indeks Kamus Data dan Perubahan menjadi Bentuk Biner

Proses mengubah isi *file* menjadi indeks kamus data adalah dengan mencari kesamaan dari isi file_gabung2 dengan isi dari kamus data. Jika terdapat kesamaan, maka isi file_gabung2 akan diubah menjadi indeks kamus data yang cocok tersebut. Isi file_gabung2 yang telah diubah menjadi indeks kamus data, selanjutnya diubah menjadi bentuk biner. Perubahan menjadi bentuk biner disesuaikan dengan panjang biner indeks tertinggi. Proses perubahan isi file_gabung2 menjadi indeks kamus data dan perubahan menjadi bentuk biner dapat ditunjukkan pada Gambar 4.9.

4.2.9 Proses Perubahan Isi *File* Bentuk Biner menjadi Bentuk Desimal.

Perubahan menjadi bentuk desimal dilakukan pada setiap delapan digit biner. Proses tersebut menghasilkan nilai *encode* dari file_gabung2. Proses perubahan isi *file* yang telah menjadi bentuk biner menjadi bentuk desimal dapat ditunjukkan pada Gambar 4.10.

```
  for g:=0 to pjg_file_2-1 do
    begin
      index:=-1;
      for h:=0 to length(kamus)-1 do
      begin
        if file_gabung2[g].char_2=kamus[h].char_2 then
        begin
          index:=h;
          break;
        end
      end;
      file_gabung2[g].index:=index;
    end;
    pjg_kamus:=length(kamus);
  pjg_idx_kamus:=pjg_kamus-1;
  bin_pjg_idx_kamus:=menjadi_bentuk_biner(pjg_idx_kamus);
  pjg_biner_penyesuaian:=length(bin_pjg_idx_kamus)-1;
  for k:=0 to length(kamus)-1 do
  begin
kamus[k].biner_penyesuaian:=integer_to_biner(kamus[k].nila
i_encode, pjg_biner_penyesuaian);
  end;
  gabung_bin_enc :='';
  for l:=0 to length(file_gabung2)-1 do
    begin
      temp_biner :='';
```

```
      for m:=0 to length(kamus)-1 do
      begin
        if     file_gabung2[l].index=kamus[m].nilai_encode
then
        begin
          temp_biner:=kamus[m].biner_penyesuaian;
        end;

      end;
      file_gabung2[l].biner_penyesuaian:=temp_biner;
      gabung_bin_enc          :=gabung_bin_enc          +
file_gabung2[l].biner_penyesuaian;
end;
```

Gambar 4.9 Proses perubahan isi *file* menjadi indeks kamus data dan perubahan menjadi bentuk biner.

```
if length(gabung_bin_enc) mod 8 = 0 then
  begin
    setlength(arr_8_byte, length(gabung_bin_enc) div 8);
    sisa:=0;
  end
  else
  begin
    setlength(arr_8_byte, (length(gabung_bin_enc) div 8) +
1);
    sisa:=length(gabung_bin_enc) mod 8;
  end;
  ix := 0; s_temp:='';
  for l:=1 to length(gabung_bin_enc) do
  begin
    s_temp := s_temp + gabung_bin_enc[l];
    if                (length(s_temp)=8)                or
(l=length(gabung_bin_enc))then
    begin
      arr_8_byte[ix]:=biner_to_integer(s_temp);
      s_temp:= '';
      inc(ix);
    end;
  end;
```

Gambar 4.10 Proses perubahan isi *file* bentuk biner menjadi bentuk desimal.

4.3 Implementasi Prosedur dan Fungsi Dekompresi

Pada subbab ini akan dijelaskan proses yang digunakan dalam proses dekompresi yang telah didesain pada Bab 3, yaitu sebagai berikut :

4.3.1 Proses Perubahan Isi Nilai Encode

Pada proses ini semua isi nilai encode dari *file* kompresi diubah menjadi bentuk biner dengan panjang maksimal setiap nilai encode adalah delapan digit. Hasil perubahan menjadi bentuk biner tersebut digabungkan menjadi satu dan disimpan pada encode_biner1. Proses perubahan isi nilai encode dari *file* kompres menjadi bentuk biner ditunjukkan pada Gambar 4.11.

```
encode_biner1:='';
  if psisa_biner_gabung=0 then
  begin
    for e:=0 to pjg_encode_angka-1 do
    begin

encode_biner1:=encode_biner1+integer_to_biner(isi_encode[
e],7);
    end;
    encode_biner2:='';
  end
  else
  begin
    for e:=0 to pjg_encode_angka-2 do
    begin
encode_biner1:=encode_biner1+integer_to_biner(isi_encode[
e],7);
    end;
encode_biner2:=integer_to_biner(isi_encode[pjg_encode_ang
ka-1], psisa_biner_gabung-1);
  end;
  encode_biner1:=encode_biner1+encode_biner2;
```

Gambar 4.11 Proses perubahan isi nilai encode

4.3.2 Proses Pengelompokkan Isi encode_biner1 Sesuai dengan Panjang Biner Potong

Panjang biner potong ini merupakan panjang biner indeks tertinggi dari kamus data. Dilihat pada seluruh isi encode_biner1, apabila panjang biner indeks tertinggi sama dengan jumlah digit biner maka disimpan pada *array* hasil. Proses pengelompokkan isi encode_biner1 sesuai dengan panjang biner potong ditunjukkan pada Gambar 4.12.

4.3.3 Proses Perubahan menjadi *File* Asli

Setelah pengelompokkan isi encode_biner1 sesuai dengan panjang biner potong, kemudian dilakukan perubahan menjadi

bentuk desimal. Proses selanjutnya adalah mengembalikan *file* kompresi menjadi seperti *file* asli. Isi dari *array* hasil disesuaikan dengan nilai kamus_decode.nilai_encode. Apabila nilai desimal terdapat kesamaan dengan nilai indeks kamus_decode.nilai_encode, maka nilai indeks kamus_decode.nilai_encode disesuaikan dengan nilai dari kamus_decode.byte1 dan kamus_decode.byte2. Nilai dari kamus_decode.byte1 dan kamus_decode.byte2 merupakan hasil dari proses dekompresi. Proses perubahan menjadi *file* asli dapat ditunjukkan pada Gambar 4.13.

```
ii:=0; temp:='';setlength(hasil, pfile_2_angka);
  for h:=1 to length(encode_biner1) do
  begin
    temp:=temp+encode_biner1[h];
    if           (length(temp)=pbin_pot_angka)           or
(h=length(encode_biner1))then
    begin
      hasil[ii]:=temp;
      temp:=''; inc(ii);
    end;
  end;
```

Gambar 4.12 Proses pengelompokkan isi encode_biner1 sesuai dengan panjang biner potong

```
pjg_hasil:=length(hasil); setlength(akhir, pjg_hasil);
  hasil_integer:=0;
  for i:=0 to pjg_hasil-1 do
  begin
ketemu:=false;idxketemu:=0;
hasil_integer:=BinToInt(hasil[i]);
    for j:=0 to pkms_angka-1 do
    begin
      if hasil_integer=kamus_decode[j].nilai_encode then
        begin
          ketemu:=true;
          idxketemu:=j;
        end;
    end;
    if ketemu then
    begin
 if  (i=pjg_hasil-1)  and  (kamus_decode[idxketemu].byte2=0)
then
akhir[i]:=char(kamus_decode[idxketemu].byte1)
        else
akhir[i]:=char(kamus_decode[idxketemu].byte1)+char(kamus_de
code[idxketemu].byte2);
      end  end;
```

Gambar 4.13 Proses perubahan menjadi *file* asli.

4.4 Prosedur Penyimpanan

Pada Subbab ini membahas proses penyimpanan *file* hasil kompresi dan dekompresi, yaitu :

4.4.1 Proses Penyimpanan *File* Hasil Kompresi

Pada proses kompresi, data yang disimpan pada *header file* meliputi ekstensi asli, panjang biner penyesuaian, panjang file yang telah berubah menjadi model bigram, panjang biner penyesuaian, panjang sisa biner penggabungan dan data kamus. Untuk data kamus yang disimpan meliputi karakter unik penyusun *file* dengan model bigram dan panjang data kamus. Pada data kompresi yang disimpan meliputi isi hasil kompresi dan panjang hasil kompresi. Ekstensi *file* hasil kompresi adalah *.chi. Proses penyimpanan *file* hasil kompresi dapat dilihat pada Gambar 4.14.

```
if nama_file='html' then
    AssignFile(kompress_final,       copy(nama_file,       0,
length(nama_file)-5)+'chi')
  else
  AssignFile(kompress_final,        copy(nama_file,       0,
length(nama_file)-4)+'chi');

  Label17.Caption:=copy(nama_file,    0,    length(nama_file)-
4)+'chi';
  Rewrite(kompress_final);
  sizeasli :=filesize(file_utama);

pjg_file_2_string:=inttostr(panjang);
pjg_file_2_byte:=length(pjg_file_2_string);
  write(kompress_final, pjg_file_2_byte);

  for o:=1 to length(pjg_file_2_string) do
  begin
    ex:=ord(pjg_file_2_string[o]);
    write(kompress_final, ex);
  end;

  exte:=get_extention_file(OpenDialog1.FileName);
  lex:=length(exte);
  write(kompress_final, lex);

  for m:=1 to lex do
  begin
    ex:=ord(exte[m]);
    write(kompress_final, ex);
  end;
```

```
pjg_kamus_string:=inttostr(pjg_kamus);
pjg_kamus_byte:=length(pjg_kamus_string);
write(kompress_final, pjg_kamus_byte);

for m:=1 to length(pjg_kamus_string) do
begin
  ex:=ord(pjg_kamus_string[m]);
  write(kompress_final, ex);
end;

pjg_bin_potong_string:=inttostr(pjg_biner_penyesuaian);
pjg_bin_potong_byte:=length(pjg_bin_potong_string);
write(kompress_final, pjg_bin_potong_byte);
for o:=1 to length(pjg_bin_potong_string) do
begin
  ex:=ord(pjg_bin_potong_string[o]);
  write(kompress_final, ex);
end;
for yy:=0 to pjg_kamus-1 do
begin
  kamus_final1:=kamus[yy].byte1;
  kamus_final2:=kamus[yy].byte2;
  write(kompress_final, kamus_final1);
  write(kompress_final, kamus_final2);
end;
write(kompress_final, sisa);

ecd := length(arr_8_byte);
ecd_string:=inttostr(ecd);
ecd_byte:=length(ecd_string);
write(kompress_final, ecd_byte);
for m:=1 to ecd_byte do
begin
  ex:=ord(ecd_string[m]);
  write(kompress_final, ex);
end;

for n:=1 to ecd do
begin
  write(kompress_final, arr_8_byte[n]);
end;
```

Gambar 4.14 Proses penyimpanan *file* hasil kompresi.

4.4.2 Proses Penyimpanan *File* Hasil Dekompresi

Ekstensi *file* hasil dekompresi sama dengan ekstensi *file* sebelum dilakukan proses kompresi yang didapatkan dari ext. Keseluruhan isi dari *array* akhir akan disimpan menjadi bentuk teks. Proses penyimpanan *file* hasil dekompresi dapat dilihat pada Gambar 4.15.

```
nama_file_dekompress:=OpenDialog1.FileName+'.'+ext;
  AssignFile(hasil_akhir, nama_file_dekompress);

nama_file_dekompress:=get_file_name(OpenDialog1.FileName)+'.
'+ext;
  Label17.Caption:=nama_file_dekompress;
  Rewrite(hasil_akhir);
  for k:=0 to length(akhir)-1 do
  begin
    write(hasil_akhir, akhir[k]);
  end;
```

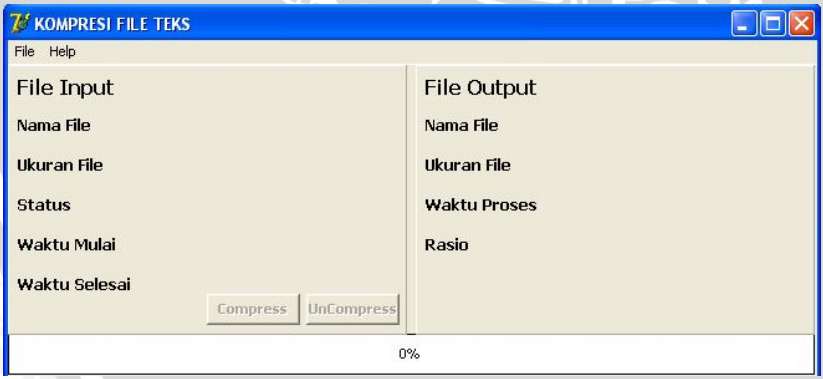
Gambar 4.15 Proses penyimpanan *file* hasil dekompresi

4.5 Implementasi Perhitungan Rasio Kompresi

Rasio kompresi merupakan perbandingan antara ukuran *file* asli dengan ukuran *file* terkompresinya. Rasio kompresi menunjukkan tingkat keefektifan sebuah sistem kompresi. Semakin tinggi rasio yang dicapai oleh sebuah sistem, maka akan semakin efektif pula proses kompresi yang dilakukan terhadap *file* tersebut.

4.6 Implementasi Antarmuka

Berdasarkan pada rancangan antarmuka pada Bab 3 maka dihasilkan antarmuka yang merupakan tampilan awal dari system ditunjukkan pada Gambar 4.16.



Gambar 4.16 Antarmuka aplikasi kompresi *file* teks berdasarkan karakteristik himpunan dengan model bigram.

Proses kompresi dilakukan dengan cara user memilih terlebih dahulu *file* yang akan dikompresi. Ketika user memilih suatu *file*, maka informasi mengenai nama, ukuran *file* dan status dari *file*.

4.7 Implementasi Uji Coba dan Evaluasi Hasil

4.7.1 Rancangan Evaluasi

Pengujian dilakukan terhadap 60 *file* yang terdiri dari tiga jenis ekstensi, yaitu *.txt, *.htm dan *.rtf. *File* yang diuji adalah sama, namun berbeda ekstensi. Masing-masing ekstensi terdiri dari 20 *file*. Uji coba dilakukan terhadap berbagai ukuran *file*. Data *input* yang digunakan dalam uji coba ditunjukkan pada Tabel 4.1, Tabel 4.2 dan Tabel 4.3.

Tabel 4.1 Data *input* untuk uji coba file *.txt.

No	Nama File	Ekstensi	Ukuran File (bytes)
1	satukan	*.txt	2901
2	USB	*.txt	3635
3	NX	*.txt	8048
4	dokumen	*.txt	8236
5	mobile	*.txt	8561
6	koneksi	*.txt	9964
7	bab	*.txt	12733
8	arwin	*.txt	14197
9	tes	*.txt	15877
10	makalah	*.txt	22746
11	anon	*.txt	26420
12	semut	*.txt	30338
13	coba	*.txt	1164
14	README	*.txt	35812
15	tips	*.txt	36459
16	sql	*.txt	38003
17	runlength	*.txt	48185
18	hernawan	*.txt	60476
19	news	*.txt	139289
20	dasar	*.txt	271145

Table 4.2 Data *input* untuk uji coba file *.htm

No	Nama File	Ekstensi	Ukuran File (bytes)
1	satukan	*.htm	25870
2	USB	*.htm	27332
3	NX	*.htm	33411
4	dokumen	*.htm	30761
5	mobile	*.htm	32989
6	koneksi	*.htm	35989
7	bab	*.htm	42579
8	arwin	*.htm	41648
9	tes	*.htm	41981
10	makalah	*.htm	56728
11	anon	*.htm	72963
12	semut	*.htm	59160
13	coba	*.htm	22782
14	README	*.htm	110415
15	tips	*.htm	84668
16	sql	*.htm	78628
17	runlength	*.htm	113144
18	hernawan	*.htm	114146
19	news	*.htm	261477
20	dasar	*.htm	442377

Table 4.3 Data *input* untuk uji coba file *.rtf

No	Nama File	Ekstensi	Ukuran File (bytes)
1	satukan	*.rtf	3337
2	USB	*.rtf	4656
3	NX	*.rtf	8785
4	dokumen	*.rtf	8489
5	mobile	*.rtf	9283
6	koneksi	*.rtf	10842
7	bab	*.rtf	14321
8	arwin	*.rtf	15895
9	tes	*.rtf	16784
10	makalah	*.rtf	25071
11	anon	*.rtf	31619
12	semut	*.rtf	31326
13	coba	*.rtf	32285
14	README	*.rtf	43441
15	tips	*.rtf	40343
16	sql	*.rtf	41559
17	runlength	*.rtf	53853
18	hernawan	*.rtf	66532
19	news	*.rtf	149112
20	dasar	*.rtf	302742

4.7.2 Hasil Uji Coba

Uji coba proses kompresi untuk sebuah *file* teks telah menghasilkan sebuah *file* terkompresi dengan ekstensi *.chi. Rata-rata persentase rasio kompresi yang dicapai oleh sistem untuk *file* dengan ekstensi *.txt adalah 25.52%. Rasio tertinggi dihasilkan oleh *file* dengan nama dokumen dengan nilai rasio sebesar 35.31%. Rasio kompresi terendah adalah *file* dengan nama USB dengan nilai rasio sebesar 8.06%. Hasil uji coba rasio kompresi terhadap sistem untuk proses kompresi *file* teks dengan ekstensi *.txt ditunjukkan pada Tabel 4.4.

Tabel 4.4 Rasio kompresi *file* *.txt.

No	Nama File	Ukuran File Asli (bytes)	Ukuran File Terkompresi (bytes)	Waktu Proses (detik)	Rasio Kompresi (%)
1	satukan	2901	2338	0	19.41
2	USB	3635	3342	0	8.06
3	NX	8048	5522	0	31.39
4	dokumen	8236	5328	0	35.31
5	mobile	8561	6561	0	23.36
6	koneksi	9964	6622	0	33.54
7	bab	12733	9778	1	23.21
8	arwin	14197	10495	1	26.08
9	tes	15877	11437	1	27.96
10	makalah	22746	15966	1	29.81
11	anon	26420	20769	2	21.39
12	semut	30338	20403	2	32.75
13	coba	1164	943	0	18.99
14	README	35812	27048	3	24.47
15	tips	36459	27256	4	25.24
16	sql	38003	28245	3	25.68
17	runlength	48185	32175	6	33.23
18	hernawan	60476	44111	8	27.06
19	news	139289	108618	49	22.02
20	dasar	271145	190034	167	29.91

Uji coba proses kompresi untuk sebuah *file* teks telah menghasilkan sebuah *file* terkompresi dengan ekstensi *.chi. Rata-rata persentase rasio kompresi yang dicapai oleh sistem untuk *file* dengan ekstensi *.htm adalah 26.43%. Rasio tertinggi dihasilkan oleh *file* dengan nama dokumen dengan nilai rasio sebesar 31.26%. Rasio kompresi terendah adalah *file* dengan nama makalah dengan nilai rasio sebesar 22.66%. Hasil uji coba rasio kompresi terhadap sistem untuk proses kompresi *file* teks yang berekstensi *.htm ditunjukkan pada Tabel 4.5.

Tabel 4.5 Rasio kompresi *file* *.htm.

No	Nama File	Ukuran File Asli (bytes)	Ukuran File Terkompresi (bytes)	Waktu Proses (detik)	Rasio Kompresi (%)
1	satukan	25870	18058	2	30.20
2	USB	27332	20902	3	23.53
3	NX	33411	25126	2	24.80
4	dokumen	30761	21144	2	31.26
5	mobile	32989	24930	3	24.43
6	koneksi	35989	26845	4	25.40
7	bab	42579	31967	4	24.92
8	arwin	41648	31292	5	24.87
9	tes	41981	31366	4	25.23
10	makalah	56728	41636	7	22.66
11	anon	72963	53498	12	26.68
12	semut	59160	43122	8	27.12
13	coba	22782	15519	2	30.12
14	README	110415	79164	29	28.30
15	tips	84668	61211	16	27.70
16	sql	78628	57040	14	27.46
17	runlength	113144	80698	30	28.68
18	hernawan	114146	81869	30	28.28
19	news	261477	200996	165	23.13
20	dasar	442377	336323	435	23.97

Uji coba proses kompresi untuk sebuah *file* teks telah menghasilkan sebuah *file* terkompresi dengan ekstensi *.chi. Rata-rata persentase rasio kompresi yang dicapai oleh sistem untuk *file* dengan ekstensi *.rtf adalah 25.22%. Rasio tertinggi dihasilkan oleh *file* dengan nama dokumen dengan nilai rasio sebesar 34.47%. Rasio kompresi terendah adalah *file* dengan nama USB dengan nilai rasio sebesar 12.61%. Hasil uji coba rasio kompresi terhadap sistem untuk proses kompresi *file* teks yang berekstensi *.rtf ditunjukkan pada Tabel 4.6.

Tabel 4.6 Rasio kompresi *file* *.rtf.

No	Nama File	Ukuran File Asli (bytes)	Ukuran File Terkompresi (bytes)	Waktu Proses (detik)	Rasio Kompresi (%)
1	satukan	3337	2667	0	20.17
2	USB	4656	4069	0	12.61
3	NX	8785	6555	0	25.38
4	dokumen	8489	5563	0	34.47
5	mobile	9283	7072	1	23.82
6	koneksi	10842	7874	1	27.38
7	bab	14321	10811	0	24.50
8	arwin	15895	11568	0	27.22
9	tes	16784	12118	1	27.80
10	makalah	25071	17469	1	30.32
11	anon	31619	24256	3	23.29
12	semut	31326	21088	2	32.68
13	coba	32285	21914	3	32.12
14	README	43441	32348	4	25.54
15	tips	40343	29994	4	25.65
16	sql	41559	30730	3	26.15
17	runlength	53853	35640	6	33.82
18	hernawan	66532	48318	9	27.38
19	news	149112	106609	56	28.50
20	dasar	302742	211621	211	30.10

Hasil uji coba untuk proses dekompresi ditunjukkan pada Tabel 4.7. Proses dekompresi dari sistem dapat mengembalikan sebuah *file* *.chi menjadi *.txt. Waktu proses dekompresi yang dibutuhkan oleh sistem dipengaruhi oleh komposisi data, proses pembacaan header *file* kompresi serta perangkat keras yang digunakan oleh user.

Table 4.7 Hasil dekompresi *file* *.txt.

No	Nama File	Ukuran File terkompresi (Bytes)	Ukuran File Dekompresi (Bytes)	Waktu Proses (detik)
1	satukan	2338	2901	0
2	USB	3342	3635	0
3	NX	5522	8048	0
4	dokumen	5328	8236	0
5	mobile	6561	8561	1
6	koneksi	6622	9964	0
7	bab	9778	12733	0
8	arwin	10495	14197	0
9	tes	11437	15877	1
10	makalah	15966	22746	2
11	anon	20769	26420	2
12	semut	20403	30338	2
13	coba	943	1164	0
14	README	27048	35812	4
15	tips	27256	36459	3
16	sql	28245	38003	4
17	runlength	32175	48185	6
18	hernawan	44111	60476	11
19	news	108618	139289	85
20	dasar	190034	271145	294

Proses dekompresi dari sistem dapat mengembalikan sebuah *file* *.chi menjadi *.htm. Waktu proses dekompresi yang dibutuhkan oleh sistem dipengaruhi oleh komposisi data, proses pembacaan header *file* kompresi serta perangkat keras yang digunakan oleh user. Hasil uji coba untuk proses dekompresi ditunjukkan pada Tabel 4.8.

Tabel 4.8 Hasil dekompresi *file* *.htm.

No	Nama File	Ukuran File terkompresi (Bytes)	Ukuran File Dekompresi (Bytes)	Waktu Proses (detik)
1	satukan	18058	25870	2
2	USB	20902	27332	3
3	NX	25126	33411	4
4	dokumen	21144	30761	2
5	mobile	24930	32989	3
6	koneksi	26845	35989	4
7	bab	31967	42579	5
8	arwin	31292	41648	6
9	tes	31366	41981	6
10	makalah	41636	56728	10
11	anon	53498	72963	16
12	semut	43122	59160	12
13	coba	15519	22782	1
14	README	79164	110415	45
15	tips	61211	84668	25
16	sql	57040	78628	21
17	runlength	80698	113144	45
18	hernawan	81869	114146	48
19	news	200996	261477	314
20	dasar	336323	442377	947

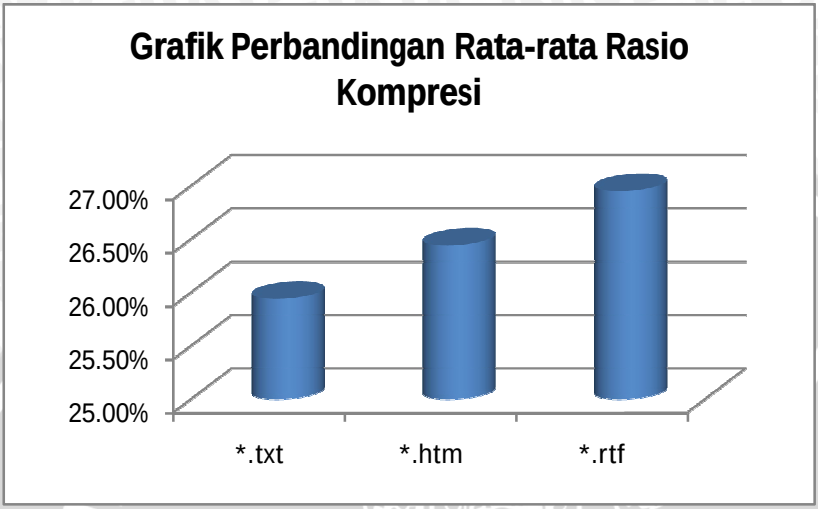
Proses dekompresi dari sistem dapat mengembalikan sebuah *file* *.chi menjadi *.rtf. Waktu proses dekompresi yang dibutuhkan oleh sistem dipengaruhi oleh komposisi data, proses pembacaan header *file* kompresi serta perangkat keras yang digunakan oleh user. Hasil uji coba untuk proses dekompresi ditunjukkan pada Tabel 4.9.

Table 4.9 Hasil dekompresi *file* *.rtf.

No	Nama File	Ukuran File terkompresi (Bytes)	Ukuran File Dekompresi (Bytes)	Waktu Proses (detik)
1	satukan	2667	3337	0
2	USB	4069	4656	0
3	NX	6555	8785	0
4	dokumen	5563	8489	1
5	mobile	7072	9283	0
6	koneksi	7874	10842	0
7	bab	10811	14321	1
8	arwin	11568	15895	1
9	tes	12118	16784	1
10	makalah	17469	25071	1
11	anon	24256	31619	3
12	semut	21088	31326	1
13	coba	21914	32285	3
14	README	32348	43441	6
15	tips	29994	40343	5
16	sql	30730	41559	6
17	runlength	35640	53853	8
18	hernawan	48318	66532	13
19	news	106609	149112	81
20	dasar	211621	302742	361

4.7.3 Analisa dan Evaluasi Hasil

Kompresi *file* teks berdasarkan karakteristik himpunan dengan model bigram dilakukan pada 60 *file* yang terdiri dari *file* teks dengan ekstensi *.txt, *.htm dan *.rtf. *File* yang diuji sama, namun berbeda ekstensi. Setiap jenis *file* teks tersebut terdiri dari 20 *file*. Persentase rata-rata rasio kompresi disajikan pada Gambar 4.17.



Gambar 4.17 Grafik perbandingan rata-rata rasio kompresi *file* teks berdasarkan karakteristik himpunan dengan model bigram.

Berdasarkan Gambar 4.17 dapat diketahui bahwa persentase rata-rata rasio kompresi yang dihasilkan untuk *file* dengan ekstensi *.txt sebesar 25.94%, *file* dengan ekstensi *.htm persentase rata-rata rasio kompresi yang dihasilkan sebesar 26.44% dan persentase rata-rata rasio kompresi yang dihasilkan untuk *file* dengan ekstensi *.rtf sebesar 26.95%. Persentase tertinggi rata-rata rasio kompresi diperoleh *file* dengan ekstensi *.rtf karena mempunyai isi header dan karakter dengan model bigram yang beragam.

File dengan nama dokumen.txt merupakan *file* yang memiliki rasio kompresi tertinggi. Rasio kompresi meningkat karena frekuensi kemunculan model bigram tinggi yang menyebabkan jumlah kamus data kecil, yaitu sebesar 337. *File* dengan nama USB.txt merupakan *file* yang memiliki rasio kompresi terendah karena frekuensi kemunculan karakter dengan model bigram rendah yang menyebabkan jumlah kamus data besar, yaitu sebesar 524. Apabila jumlah kamus data besar, maka *file* hasil kompresi (*.chi) yang disimpan semakin besar. Sebaliknya, jika jumlah kamus data kecil, maka *file* hasil kompresi (*.chi) yang disimpan semakin kecil.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pembahasan yang telah disampaikan dapat disimpulkan sebagai berikut :

1. Algoritma yang didasarkan pada karakteristik himpunan dengan model bigram pada *file* teks telah menghasilkan ukuran *file* kompresi yang lebih kecil dari ukuran *file* semula.
2. Persentase rata-rata rasio kompresi untuk *file* dengan ekstensi *.txt sebesar 25.94%, *.htm sebesar 26.44% dan *.rtf sebesar 26.95%.

5.2 Saran

Perlu dilakukan penelitian lebih lanjut mengenai kompresi *file* teks berdasarkan karakteristik himpunan menggunakan model bigram dengan menggabungkan algoritma seperti *Lempel Ziv Welch (*LZW*)* dan Huffman.

DAFTAR PUSTAKA

Abdurohman, M. 2002. "Buku Ajar Bahasa Pemrograman Tingkat Rendah". Sekolah Tinggi Teknologi Telkom. Bandung.

Anggraini, I. N. 2010. "Kompresi *File* Teks Berdasarkan Karakteristik Himpunan". Fakultas MIPA. Universitas Brawijaya. Malang.

Anton, 2005. "Kompresi dan Teks". Universitas Kristen Duta Wacana. Yogyakarta.

Arifia, N. 2010. "Sistem Digital". Computer Science and Information Technology. Universitas Gunadarma. Jakarta.

Callista, N. 2010. "Aplikasi Greedy Pada Algoritma *Huffman* Untuk Kompresi Teks". Institut Teknologi Bandung. Bandung.

Didi. 2010. "Pengantar Teori Himpunan".

Firdausy, K. 2011. "Sistem Bilangan". Universitas Ahmad Dahlan. Yogyakarta.

Jatnika, Ihsan. 2006. "Teknik Kompresi Data". Universitas Pendidikan Indonesia.

Kustiawan, C. "Fungsi Karakteristik". Fakultas MIPA. Universitas Pendidikan Indonesia.

Lulu. 2010. "Sistem *File"*. Computer Science and Information Technology. Universitas Gunadarma. Jakarta.

Munir, R. 2006. "Diktat Kuliah IF2153 Matematika Diskrit". Program Studi Teknik Informatika. Institut Teknologi Bandung. Bandung.

Noeryanti, 2010. "Teori Himpunan", Fakultas MIPA. Institut Ilmu dan Teknologi Yogyakarta. Yogyakarta.

Proboyekti, U. 2010. “Pengantar Teknologi Informasi”. Universitas Kristen Duta Wacana. Yogyakarta.

Restyandito, 2010. “Metode Statistik Kompresi Data”. Universitas Kristen Duta Wacana. Yogyakarta.

Rosida, Ermawati. 2006. “Fungsi Karakteristik Suatu Peubah Acak dan Penerapannya”. Universitas Muhamadiyah Malang. Malang.

Sukamto, R. A. 2009. ”Penguraian Bahasa Indonesia Dengan Menggunakan Pengurai *Collins”*. Institut Teknologi Bandung.

Sumari, A. D. W. 2010. “Teknologi *Real-Time* : Konsep dan Aplikasi”. Lanud Iswahjudi.

Syarif, Abdusy. 2011. “Kompresi Suara, Gambar dan Vidieo Digital (Prinsip dan Teknik)”. Universitas Mercu Buana. Jakarta.

Wibisana, A. 2008. “Kompresi *Short Message Service* Menggunakan *Huffman Coding* dan Perulangan Bigram”. Fakultas MIPA. Universitas Brawijaya. Malang.

Widhiartha, P. 2008. “Pengantar Kompresi Data”. Fakultas Mipa. Universitas Pendidikan Indonesia. Bandung.

Winanti, Winda. 2006. “Pemampatan Data Dengan Kode Huffman. Institut Teknologi Bandung”. Bandung.

Zulen, A. A. 2010. “Penerapan Pohon Biner *Huffman* Pada Kompresi Citra”. Institut Teknologi Bandung. Bandung.