KOMPRESI *SHORT MESSAGE SERVICE* MENGGUNAKAN HUFFMAN *CODING* DAN PERULANGAN BIGRAM

SKRIPSI

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

Oleh :

ARYA WIBISANA
0410963006-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2008

LEMBAR PENGESAHAN SKRIPSI

KOMPRESI *SHORT MESSAGE SERVICE* MENGGUNAKAN HUFFMAN *CODING* DAN PERULANGAN BIGRAM

oleh :
ARYA WIBISANA
0410963006 - 96

Telah dipertahankan di depan Majelis Penguji
pada tanggal 11 Desember 2008
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Drs. Muh Arif Rahman, M.Kom	**Candra Dewi, S.Kom., MSc.**
NIP. 131 971 481	**NIP. 132 304 116**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, MSc.
NIP. 132 126 049

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: Arya Wibisana**
NIM	**: 0410963006-96**
Jurusan	**: Matematika**
Program Studi	**: Ilmu Komputer**
Penulis skripsi berjudul	**: Kompresi *Short Message Service* Menggunakan Huffman *Coding* Dan Perulangan 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, 11 Desember 2008
Yang menyatakan,

(Arya Wibisana)
NIM. 0410963006-96

KOMPRESI *SHORT MESSAGE SERVICE* MENGGUNAKAN HUFFMAN *CODING* DAN PERULANGAN BIGRAM

ABSTRAK

Format standar SMS (*Short Message Service*) adalah 160 karakter, dengan masing - masing karakter dikodekan sebanyak 7 bit. Sehingga total untuk satu kali pengiriman sebanyak 1120 bit. Apabila jumlah karakter yang ditulis melebihi batas tersebut, maka otomatis biaya pengiriman akan melebihi satu SMS. Penghematan biaya pengiriman SMS dapat dilakukan dengan melakukan kompresi sebelum dilakukan pengiriman.

Salah satu metode kompresi teks yang cukup populer adalah Huffman *Coding*. Algoritma Huffman menggunakan prinsip pengkodean dimana karakter yang sering dipakai dikodekan dengan rangkaian bit yang pendek, dan karakter yang jarang dipakai dikodekan dengan rangkaian bit yang lebih panjang. Pengkodean Huffman juga dapat dilakukan terhadap bigram. Bigram-bigram yang sering berulang dalam penulisan teks dapat dimanfaatkan untuk menambah besarnya rasio kompresi. Penggabungan metode Huffman dan bigram dilakukan dengan mengkodekan bigram yang sering berulang dengan bit yang lebih pendek, kemudian membuat *header* pada awalan teks yang akan menginisialisasi bigram tersebut.

Berdasarkan pengujian yang telah dilakukan terhadap 25 teks uji, nilai rata-rata rasio dengan menggunakan metode Huffman sebesar 22,37%, sedangkan nilai rata-rata rasio menggunakan metode Huffman dan bigram sebesar 27,05%. Jika ditinjau dari pengiriman SMS dengan rasio tersebut, dalam satu kali pengiriman SMS, rata-rata jumlah karakter yang dapat ditulis sebanyak 203 karakter.

SHORT MESSAGE SERVICE COMMPRESSION USING HUFFMAN CODING AND REPEATING BIGRAM

ABSTRACT

The standard format of SMS (Short Message Service) is 160 character. Each of them is coded by using 7 bit, so the total of one delivery SMS is 1120 bit. The cost of the delivery will be more than one SMS if the number of character is written more than limit above. The effeciency of the delivery cost can be done if the SMS is compressed before sending it.

One of the popular text compressor methods is Huffman Coding. Huffman algorithm uses coding principle, that the most used character is coded with short series bit, and the rarely used character is coded with longer series bit. Huffman coding can combine with bigram. The Bigram that is often repeated in the text can be used to enlarge the level of the compressor ratio. The combination of Huffman method and bigram is done by coded bigram with shorter bit, after that make header on the begining of the text to give initial on that bigram.

The testing result from 25 text shows that the average score ratio uses Huffman method is 22,37 %, while the average score ratio of method Huffman which uses bigram is 27, 05%. With that ratio for one delivery SMS, the average number of the characters can be written as 203 characters.

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat, taufik serta hidayah-Nya sehingga penulis dapat menyelesaikan skripsi ini sebagai salah satu syarat untuk memperoleh gelar Sarjana dalam bidang Ilmu Komputer.

Banyak pihak yang berperan atas terselesaikannya penelitian dan penulisan Skripsi ini. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Drs. Muh. Arif Rahman, M.Kom. selaku dosen pembimbing 1 yang telah memberikan saran dan bimbingan atas skripsi ini.
2. Candra Dewi,. S.Kom, M.Sc selaku dosen pembimbing 2 yang telah memberikan bimbingan dalam penulisan laporan.
3. Dr. Agus Suryanto, M.Sc., selaku ketua jurusan Matematika Fakultas Mipa Universitas Brawijaya Malang.
4. Wayan Firdaus Mahmudy, S.Si, M.T., selaku Ketua Program Studi Ilmu Komputer Universitas Brawijaya Malang.
5. Drs. A Ridok, M.Kom selaku pembimbing akademik
6. Segenap Bapak dan Ibu dosen yang telah mendidik penulis selama menempuh pendidikan di Program Studi Ilmu Komputer, Universitas Brawijaya.
7. Segenap staf dan karyawan di Ilmu Komputer Universitas Brawijaya.
8. Orang tua, adik dan kakak penulis yang telah memberikan semangat dan doa restunya selama menempuh kuliah dan dalam pengerjaan skripsi.
9. Teman-teman Ilmu Komputer Universitas Brawijaya yang telah banyak memberikan bantuan dan ilmu pengetahuan pada penulis selama menempuh kuliah.
10. Semua pihak lain yang telah membantu terselesaikannnya skripsi ini yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, dan memiliki banyak kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Semoga penulisan skripsi ini dapat bermanfaat bagi pembaca.

Malang, Desember 2008

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Proses komunikasi senantiasa terjadi setiap waktu. Berbagai perangkat komunikasi terus dikembangkan dan digunakan untuk menunjang proses tersebut. Salah satu perangkat komunikasi saat ini adalah telepon genggam atau sering juga disebut *handphone* (HP). Dewasa ini penggunaan *handphone* sudah menjadi kebutuhan primer bagi sebagian manusia.

Berbagai macam fitur terdapat dalam *handphone*, salah satunya adalah layanan SMS (*short message service)*. SMS adalah sebuah mekanisme penyampaian pesan pendek dalam jaringan bergerak. Dengan adanya SMS memungkinkan dua orang yang terpisah dapat saling mengirimkan pesan teks melalui *handphone*. SMS merupakan fitur yang sering dipakai oleh pengguna *handphone* di Indonesia, karena dianggap murah padahal sebenarnya sangat mahal jika dibanding negara lainnya.

Terdapat 2 metode pengiriman SMS yaitu *Text Message* dan *Binary Message* (Farid, 2007). Untuk metode pengiriman yang berupa *Text Message,* jumlah karakter yang disediakan sebanyak 160 karakter untuk satu SMS. Karakter-karakter tersebut menggunakan kode ASCII 7 bit, yang berarti masing-masing karakter dikodekan sebanyak 7 bit. Sedangkan untuk *Binary Message* satu SMS sebanyak 1120 bit. Kedua metode tersebut memiliki besar yang sama untuk satu kali pengiriman yaitu sebanyak 140 byte. Apabila jumlah pesan yang ditulis melebihi batas tersebut, maka otomatis biaya pengiriman akan melebihi dari satu SMS. Penghematan biaya pengiriman SMS dapat dilakukan apabila sebelum dilakukan pengiriman, SMS dikompres terlebih dahulu. Oleh sebab itu, pengompresan SMS dirasa cukup penting untuk dilakukan.

Salah satu metode kompresi teks yang cukup populer adalah Huffman *Coding*. Metode ini banyak digunakan karena mempunyai prinsip dasar yang cukup sederhana. Penulisan teks yang berbeda akan menghasilkan penghematan karakter yang berbeda. Algoritma Huffman menggunakan prinsip pengkodean, dimana setiap karakter diberi kode biner yang unik. Karakter yang sering dipakai dikodekan dengan rangkaian bit yang pendek, dan karakter yang jarang dipakai

dikodekan dengan rangkaian bit yang lebih panjang. Dari rangkaian bit tersebut akan dibentuk tabel Huffman.

Pengkodean Huffman juga dapat dilakukan pada 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. Penggabungan metode Huffman dan bigram dilakukan dengan mengkodekan bigram yang sering berulang dengan bit yang lebih pendek, kemudian membuat header pada awalan teks yang akan menginisialisasi bigram tersebut. Kelebihan penggunaan bigram dibanding tipe n-gram yang lain adalah peluang perulangannya paling besar, sehingga sangat cocok untuk teks SMS yang berupa teks pendek.

Dengan memanfaatkan metode pengiriman yang berupa *Binary Message,* metode Huffman Coding dan perulangan bigram dapat dimplementasikan pada kompresi SMS. Tetapi pada kasus SMS, terjadi masalah jika tabel Huffmannya dimasukkan dalam SMS, dilihat dari format SMS yang berupa teks pendek, hanya mempunyai 160 karakter atau sebesar 140 *byte*, sedangkan untuk tabel Huffman yang dibentuk dapat mencapai lebih dari 40 *byte* (Yuku, 2007). Besar kemungkinan jika memasukkan tabel Huffman dalam SMS, *size* dari SMS akan bertambah besar. Untuk menghindari hal tersebut, maka akan digunakan metode Huffman yang mempunyai tabel tetap (*fixed table*).

Penelitian pada kompresi SMS menggunakan metode Huffman pernah dilakukan oleh Nadhira Ayuningtyas (2007). Pada penelitiannya, Huffman *tree* yang dibentuk memusatkan kompresi pada huruf kecil (a-z), sehingga algoritma Huffman yang diterapkan pada SMS tidak efektif untuk mengkompresi SMS yang menggunakan huruf besar.

Kelemahan metode Huffman yang dilakukan oleh Nadhira adalah melesetnya analisa dari tabel huffman yang telah dibentuk. Ketika karakter-karakter yang ditulis dalam SMS merupakan karakter-karakter yang berada diluar prediksi, dapat mengakibatkan *size* dari SMS bertambah besar. Oleh karena itu dalam penelitian ini akan dibuat dua jenis tabel Huffman, hasil kompresi dari kedua tabel tersebut nantinya akan dibandingkan. Tabel yang menghasilkan kompresi terbaik yang selanjutnya digunakan dalam kompresi SMS.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan maka rumusan masalah yang dikerjakan adalah:

1. Bagaimana implementasi kompresi metode Huffman dengan bigram untuk *Short Message Service* (SMS).
2. Bagaimana rasio kompresi metode Huffman dalam melakukan kompresi SMS.
3. Bagaimana rasio kompresi metode Huffman dengan metode bigram untuk kompresi SMS.

1.3 Batasan Masalah

Dari permasalahan diatas, berikut ini diberikan batasan untuk menghindari melebarnya masalah yang akan diselesaikan:

1. SMS yang dikompresi adalah SMS yang menggunakan karakter standar ASCII 7 bit.
2. Karakter yang akan dikompresi adalah huruf latin (besar dan kecil), angka latin, dan beberapa tanda baca yang umum.
3. Jumlah tabel Huffman yang dibuat dibatasi dua tabel.
4. Tree Huffman yang dibentuk merupakan hasil analisa penulis berdasarkan data-data yang penulis dapatkan.

1.4 Tujuan

Tujuan yang ingin dicapai dalam skripsi ini adalah:

1. Mengimplementasikan metode Huffman dengan bigram pada kompresi *Short Messagge Service* (SMS).
2. Menganalisa rasio kompresi metode Huffman dan bigram dalam melakukan kompresi SMS.

1.5 Manfaat

Manfaat yang akan dicapai dari skripsi ini adalah dihasilkannya perangkat lunak yang dapat melakukan kompresi SMS, sehingga biaya pengiriman SMS dapat dihemat semaksimal mungkin.

1.6 Metodologi

Metodologi yang digunakan dalam penyusunan skripsi ini adalah sebagai berikut :

1. Studi literatur
   Studi ini dilakukan dengan cara mencari sekaligus mempelajari beberapa literatur dan artikel mengenai kompresi SMS dan Huffman *Coding*.
2. Pendefinisian dan Analisis Masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi perangkat lunak
   Membuat perancangan model perangkat lunak dan mengimplementasikan hasil rancangan tersebut dengan membuat perangkat lunak kompresi dan dekompresi teks SMS menggunakan Huffman *Coding* dan bigram.
4. Uji coba dan analisa hasil implementasi
   Menguji perangkat lunak, kemudian menganalisa hasil dari implementasi kompresi SMS sesuai dengan tujuan yang dirumuskan sebelumnya, untuk kemudian dievaluasi dan disempurnakan.

1.7 Sistematika Penulisan

Sistematika penulisan skripsi ini adalah sebagai berikut :

BAB I : PENDAHULUAN
Pada bab ini membahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, serta sistematika penulisan skripsi.

BAB II : DASAR TEORI
Bab ini mencantumkan beberapa tinjauan pustaka yang berkaitan dengan penelitian ini, diantaranya : SMS, ASCII 7 bit, kompresi teks, Huffman *Coding* dan bigram.

BAB III : METODOLOGI DAN PERANCANGAN
Bab ini menjelaskan mengenai metode dan perancangan yang akan dilakukan dalam membangun tahapan-tahapan kompresi. Beberapa poin yang dibahas ialah: analisa perancangan dan pembentukan Huffman *tree*, pembentukan tabel Huffman, perancangan proses kompresi dan perancangan proses dekompresi.

BAB IV : IMPLEMENTASI DAN PEMBAHASAN
Bab ini menerangkan hasil implementasi dari metode dan perancangan yang telah dijelaskan pada bab III.

Beberapa hal yang dijelaskan yaitu: implementasi program, uji coba, dan analisa hasil kompresinya.

BAB V : PENUTUP

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan, dan saran untuk pengembangan penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

Bab ini mencantumkan beberapa tinjauan pustaka dan referensi-referensi yang berkaitan dengan penelitian ini. Pada tahap pertama akan dijelaskan mengenai *Short Message Service* (SMS) sebagai aplikasi untuk dilakukannya kompresi. Kemudian dijelaskan juga tentang pengkodean standar dari SMS yang menggunakan kode ASCII.

Pada tahap kedua akan dijelaskan mengenai pengertian dan konsep dasar dari kompresi secara umum. Sedangkan pada tahap ketiga dijelaskan mengenai kompresi Huffman beserta proses-proses yang dilakukan. Selanjutnya dijelaskan mengenai bigram beserta definisinya. Dan pada tahap terakhir akan diberikan rumus rasio kompresi untuk pengujian yang akan dilakukan.

2.1 Short Message Service (SMS)

Short Message Service atau SMS pertama kali ditemukan oleh SGM pioners di Eropa. Standarisasi di bawah lembaga *Eutopan Telecommunications Standards Institute*. SMS dibuat untuk menyediakan infrastruktur transportasi pesan singkat yang mempunyai kapasitas maksimal 140 *bytes*. Skema dasar pengalamatan SMS adalah nomor *mobile phone* yang disebut MSISDN (Sigit, 2007). SMS berbentuk bilangan biner, terbagi atas 2 bagian yaitu *message header* untuk transportasi data dan *messsage body* sebagai *payload*.

SMS saat ini menjadi sebuah fitur mendasar dari setiap telepon selular. Fitur SMS akan berjalan ketika terdapat operator telepon seluler yang menyediakan layanan ini. Dengan SMS memungkinkan dua orang yang memiliki telepon selular dapat saling mengirimkan pesan teks satu sama lain. Teknologi yang mendukung SMS antara lain adalah GSM, TDMA dan CDMA. Dengan didukung oleh ketiga teknologi ini, SMS telah menjadi layanan data bergerak yang bersifat *universal*. Protokol yang bekerja pada SMS ini lebih dikenal dengan nama *Protocol Data Unit* atau PDU (Bambang, 2007). SMS mampu mengirimkan dan menerima secara simultan data berupa teks antar jaringan operator selular.

Perkembangan SMS yang cukup pesat menjadikan layanan ini banyak diminati oleh pengguna telepon seluler. Menurut Ayuningtyas (2007) berbagai keunggulan dari SMS yaitu:

1. *Deliver Oriented Service*, pesan akan selalu diusahakan untuk dikirimkan ke tujuan. Jika suatu tujuan sedang tidak aktif, pesan akan disimpan di SMSC (*short message service centre)* sebagai *server* dan akan dikirim sesegera mungkin setelah nomor tujuan aktif. Pesan juga akan tetap terkirim walaupun pengguna sedang melakukan telepon karena transmisi SMS menggunakan kanal *signaling* bukan kanal suara.
2. Dapat dikirim ke berbagai tujuan/penerima pada saat bersamaan.
3. Dapat dikirim ke berbagai jenis penerima, seperti email, IP, maupun berbagai aplikasi lain.
4. Memiliki beberapa macam kegunaan bila dilakukan integrasi dengan berbagai aplikasi, seperti *chatting*, *voting*, kuis, reservasi, informasi tertentu, dan sebagainya.

Sebuah SMS terdiri atas 160 karakter untuk skema 7 bit, dan ada yang memakai skema 8 bit berjumlah 140 karakter, atau 70 karakter memakai skema 16 bit (Bambang, 2007). Maksud skema disini adalah pemakaian jenis tabel ASCII, jika suatu operator selular memakai skema 7 bit berarti tabel yang digunakan adalah tabel ASCII 7 bit. Untuk operator-operator yang berkembang di Indonesia saat ini kebanyakan memakai skema 7 bit. Penggabungan SMS dan kompresi SMS (terdiri lebih dari 160 karakter) telah dikembangkan. Tetapi fitur-fitur ini belum diimplementasikan oleh semua jaringan operator selular di dunia.

2.2 ASCII 7 bit

Penelitian ini akan membahas tentang SMS yang menggunakan kode ASCII 7 bit. Selain itu karena metode Huffman memodifikasi dan merubah rangkaian bit kode ASCII, maka akan dijelaskan mengenai prinsip penggunaan dan pengkodean ASCII 7 bit pada karakter.

Kode Standar Amerika untuk Pertukaran Informasi atau ASCII (*American Standard Code for Information Interchange*) merupakan suatu standar internasional berupa rangkaian bit kode untuk mewakili teks, agar bisa dimengerti oleh banyak model komputer (Ravi, 2008).

Ada dua jenis kode yang sering digunakan, yaitu ASCII- 8 bit dan ASCII- 7 bit. Kode yang terdapat pada ASCII-8 bit jauh lebih

lengkap dari ASCII-7 bit. Menurut Tino ASCII-7 bit mempunyai kombinasi kode $2^7 = 127$, dengan ketentuan sebagai berikut:

1. 26 kode untuk huruf kapital *(upper case)* dari A –Z.
2. 26 kode untuk huruf kecil *(lower case)* dari a –z.
3. 10 digit desimal dari 0 –9.
4. 34 karakter kontrol untuk informasi status operasi komputer.
5. 32 karakter khusus *(special characters)*.

Kode ASCII dengan nilai kode 0 sampai dengan 31 dan 127 termasuk dalam status karakter-karakter kontrol yang tidak dapat dicetak (*non-printable characters*). Contohnya, tabel ASCII karakter 28 mewakili fungsi "*file separator*", atau karakter 8 yang mewakili *backspace*.

Kode ke 32 adalah karakter *spasi*, menandakan pemisah antara kata, yang dihasilkan oleh *space-bar* dari *keyboard*. Kode 33 sampai 126 dikenal sebagai karakter-karakter yang dapat di cetak (*printable characters*), terdiri dari beberapa huruf, angka, tanda baca, dan beberapa simbol. Tabel 2.1 adalah beberapa contoh karakter ASCII 7 bit yang merupakan *printable characters* (Ravi, 2008).

Tabel 2.1 Kode ASCII-7 Bit

Binary	Oct	Dec	Hex	**Value**
011 0000	60	48	30	0
011 0001	61	49	31	1
011 0010	62	50	32	2
011 0011	63	51	33	3
011 0100	64	52	34	4
011 1010	72	58	3A	:
011 1011	73	59	3B	;
011 1100	74	60	3C	<
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D

2.3 Frekuensi Pemakaian Huruf Dalam Bahasa Indonesia

Penelitian ini memakai huffman dengan tabel yang tetap, oleh sebab itu akan diberikan beberapa data mengenai pemakaian huruf untuk mendukung analisis dari pembentukan *tree* Huffman yang akan dibuat pada bab 3.

Menurut penelitian yang dilakukan Aysar (2008) pada 2 artikel bahasa Indonesia yang berbeda, didapatkan data mengenai frekuensi pemakaian huruf ditunjukkan pada Tabel 2.2 :

Tabel 2.2 Data Frekuensi Pemakaian Huruf I

Rangking	File Indonesia01.txt (sumber tvone.co.id)		File Indonesia02.txt (sumber kompas.com)	
	Huruf	**Frekuensi**	**Huruf**	**Frekuensi**
1	a	690	a	634
2	e	137	n	130
3	n	127	e	126
4	i	104	r	109
5	t	80	u	87
6	u	76	m	85
7	k	72	i	84
8	r	69	t	59
9	d	66	s	57
10	m	59	k	50

Pada penelitian yang dilakukan Galih (2008) terhadap 3 artikel bahasa Indonesia yang berbeda, didapatkan data mengenai frekuensi pemakaian huruf ditunjukkan pada Tabel 2.3 :

Tabel 2.3 Data Frekuensi Pemakaian Huruf II

Rangking	File 1 (sumber detikinet.com)		File 2 (sumber detikinet.com)		File 3 (sumber detikinet.com)	
	Huruf	**Frek**	**Huruf**	**Frek**	**Huruf**	**Frek**
1	a	112	a	137	a	134
2	i	73	i	76	n	79
3	n	60	n	61	e	59
4	e	57	e	59	i	57
5	s	55	s	54	s	43

Menurut *American Cryptogram Association* (2005) menyebutkan data mengenai frekuensi pemakaian huruf dalam bahasa Indonesia adalah seperti ditunjukkan pada Tabel 2.4 :

Tabel 2.4 Data Frekuensi Pemakaian Huruf III

Rangking	Huruf	Frekuensi
1	a	22.72%
2	n	8.79%
3	e	8.68%
4	i	6.00%
5	k	5.52%
6	r	5.27%
7	u	4.94%
8	t	4.85%
9	s	4.70%
10	d	4.54%
11	m	4.14%
12	p	3.16%
13	b	3.10%
14	ng	3.03%
15	h	2.70%
16	l	2.53%
17	y	1.63%
18	o	1.20%
19	g	0.98%
20	j	0.68%
21	w	0.39%
22	c	0.32%
23	f	0.13%
24	v	0.02%

2.4 Kompresi

Kompresi atau *compression* adalah proses pemampatan ukuran sebuah data. Sebuah data terkadang memiliki sebuah informasi yang sama dan berulang-ulang. Sebuah informasi yang sama dan berulang-ulang membuat ukuran sebuah data menjadi besar. Pada transmisi data (*data transmision*), sebuah data berukuran besar akan

membutuhkan waktu *transfer* lebih lama dibandingkan data berukuran kecil. Pada *file*, sebuah *file* yang berukuran besar menyita banyak ruangan pada media penyimpanan. Oleh karena itu, untuk menghindari masalah tersebut, maka dilakukan kompresi data. Dengan mengunakan algoritma-algoritma dan teknik-teknik tertentu, informasi yang sama dan berulang-ulang tersebut dikodekan sedemikian rupa sehingga data tersebut menjadi berukuran lebih kecil. *File* atau data yang sudah dikompres agar bisa digunakan kembali harus dikembalikan lagi seperti semula. Proses pengembalian sebuah *file* yang terkompres menjadi seperti *file* aslinya disebut dekompresi atau *Decompression*.

Jenis kompresi data berdasarkan *mode* penerimaan data oleh manusia:

1. *Dialoque Mode* yaitu proses penerimaan data dimana pengirim dan penerima seakan berdialog (*real time*), seperti pada contoh *video conference*. Kompresi data pada *mode* ini harus berada dalam batas penglihatan dan pendengaran manusia. Waktu tunda (*delay*) tidak boleh lebih dari 150 ms, dimana 50 ms untuk proses kompresi dan dekompresi dan 100 ms untuk mentransmisikan data dalam jaringan.
2. *Retrieval Mode* yaitu proses penerimaan data tidak dilakukan secara *real time*. *Retrieval Mode* dapat dilakukan secara *fast forward* maupun *fast rewind* di *client*, dan dapat dilakukan *random access* terhadap data dan dapat bersifat interaktif *(*Anynomous, 2005).

Jenis kompresi data berdasarkan output :

1. Lossy Compression

- Teknik kompresi dimana terdapat data yang hilang selama proses kompresi. Artinya data hasil dekompresi tidak sama dengan data sebelum kompresi namun sudah cukup untuk digunakan. Contoh: Mp3, *streaming media*, JPEG, MPEG, dan WMA.
- Kelebihannya yaitu ukuran file lebih kecil dibanding *loseless* namun masih tetap memenuhi syarat untuk digunakan.
- Prinsip dasar dari *Lossy Compression* adalah membuang bagian-bagian data yang tidak digunakan, tidak dirasakan, tidak begitu dilihat oleh manusia sehingga manusia masih

beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi.

2. *Loseless Compression*
   - Teknik kompresi dimana data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi. Contoh aplikasi: ZIP, RAR, GZIP, 7-Zip
   - Data hasil dekompresi tetap sama dengan data sebelum dikompres, artinya tidak terdapat data yang hilang atau data kembali seperti sebelum proses kompresi.

Untuk Huffman *Coding* yang digunakan dalam penelitian ini termasuk dalam metode *Loseless Compression (*Anynomous, 2005).

2.5 Huffman *Coding*

Pada tahun 1951, David A. Huffman berhasil menemukan cara pengkodean biner yang paling efisien yang kemudian diberi nama Huffman c*oding*. Algoritma Huffman menggunakan prinsip pengkodean yang mirip dengan kode Morse, yaitu tiap karakter (simbol) dikodekan hanya dengan rangkaian beberapa bit, dimana karakter yang sering dipakai dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang dipakai dikodekan dengan rangkaian bit yang lebih panjang.

Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal menjadi sekumpulan *codeword*, algoritma Huffman termasuk ke dalam kelas algoritma yang menggunakan metode statik. Metode statik adalah metode yang selalu menggunakan peta kode yang sama, metode ini membutuhkan dua fase (*two-pass*): fase pertama untuk menghitung probabilitas kemunculan tiap simbol dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan di transmisikan. (Wardoyo,2005).

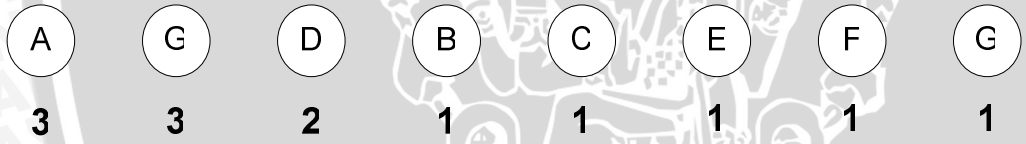
2.5.1 Pembentukan Pohon Huffman (Huffman *Tree*)

Huffman *coding* pada dasarnya adalah himpunan yang berisi sekumpulan kode biner, dimana pada Huffman *coding* ini tidak ada kode biner yang menjadi awal bagi kode biner yang lain. Huffman *coding* direpresentasikan dari pohon biner yang diberikan nilai atau label. Untuk cabang kiri pada pohon biner diberi label 0, sedangkan pada cabang kanan pada pohon biner diberi label 1. Rangkaian bit yang terbentuk pada setiap lintasan dari akar ke daun merupakan

pengkodean untuk karakter yang berpadanan. Pohon biner ini biasa disebut Huffman *tree*. Langkah-langkah pembentukan Huffman *tree* menurut Wardoyo (2005) adalah sebagai berikut:

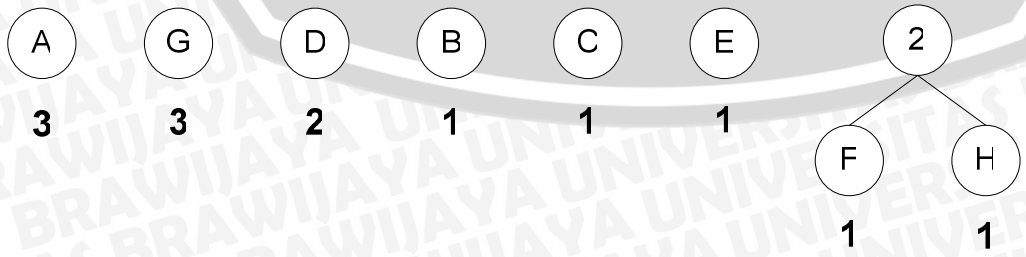
1. Baca semua karakter di dalam teks atau string untuk menghitung frekuensi kemunculan setiap karakter. Setiap karakter penyusun teks dinyatakan sebagai *tree* bersimpul tunggal. Setiap simpul di-*assign* dengan frekuensi kemunculan karakter tersebut.
2. Terapkan strategi algoritma *greedy* sebagai berikut : gabungkan dua buah *tree* yang mempunyai frekuensi terkecil pada sebuah akar. Setelah digabungkan akar tersebut akan mempunyai frekuensi yang merupakan jumlah dari frekuensi dua buah *tree* penyusunnya.
3. Ulangi langkah 2 sampai hanya tersisa satu buah Huffman *tree*. Agar pemilihan dua *tree* yang akan digabungkan berlangsung cepat, maka semua yang ada selalu terurut menaik berdasarkan frekuensi.

Sebagai contoh, akan dilakukan pembentukan tree pada *string* "ABACADGEGHDFG". Dengan rincian frekuensi pemakaian karakter seperti ditunjukkan pada Gambar 2.1.



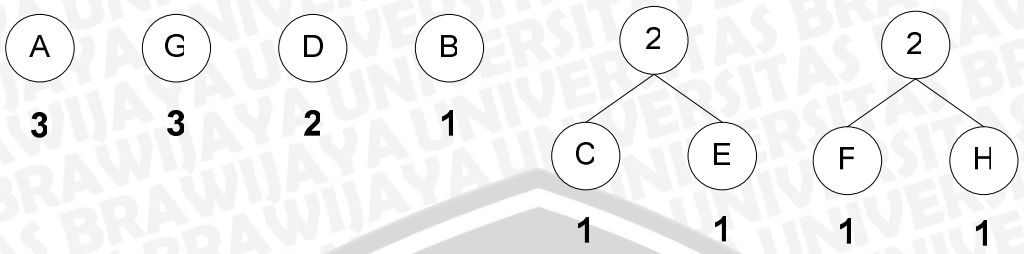
Gambar 2.1 Frekuensi Pemakaian Karakter

Dari Gambar 2.1, dipilih 2 *node* yang mempunyai frekuensi terkecil. Dalam hal ini dipilih *node* 'F' dan 'H'. Kemudian dibuat *tree* baru yang merupakan jumlah dari kedua frekuensi *node* yang telah dipilih. Hasilnya akan membentuk *tree* yang ditunjukkan pada Gambar 2.2.



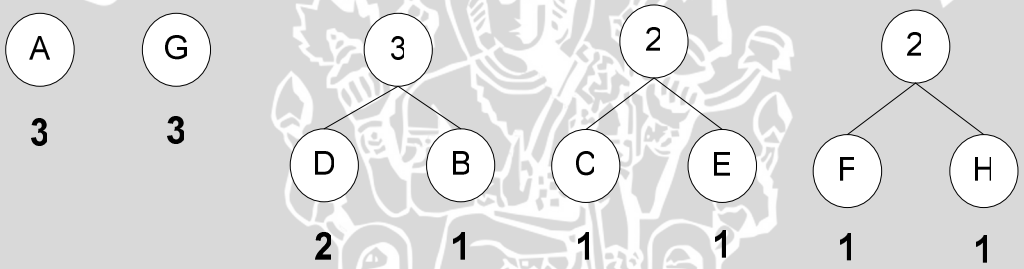
Gambar 2.2 Pembentukan Huffman *Tree* Langkah 1

Pilih 2 *node* lagi yang mempunyai frekuensi terkecil seperti yang pada langkah sebelumnya, prosesnya ditunjukkan pada Gambar 2.3.



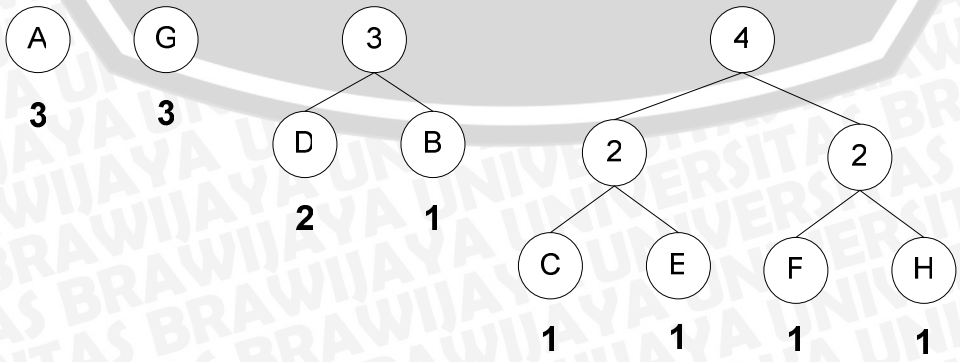
Gambar 2.3 Pembentukan Huffman *Tree* Langkah 2

Sekali lagi dipilih dua dari empat *node* yang tersisa. *Node* yang mempunyai frekuensi paling kecil adalah *node* 'B', kemudian satu *node* lagi dapat dipilih secara bebas dari ketiga *node* yang tersisa. Dalam contoh ini diplih *node* 'B' dan 'D' yang ditunjukkan pada Gambar 2.4.



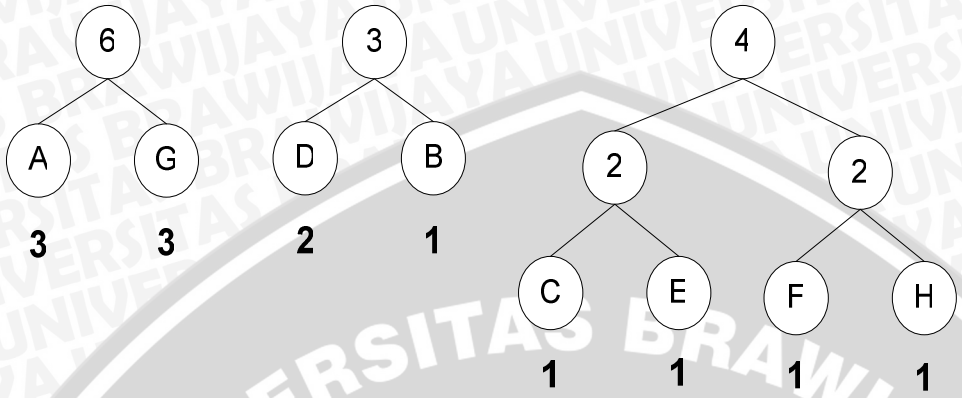
Gambar 2.4 Pembentukan Huffman *Tree* Langkah 3

Dari Gambar 2.4 didapatkan dua *tree* dengan frekuensi yang sama sebanyak 2. Kedua *tree* ini digabungkan untuk membentuk *tree* baru dengan frekuensi sebanyak 4. Hasilnya akan membentuk 4 *tree* yang tersisa, satu mempunyai frekuensi 4, dan tiga dengan frekuensi sebanyak 3, seperti ditunjukkan pada Gambar 2.5.



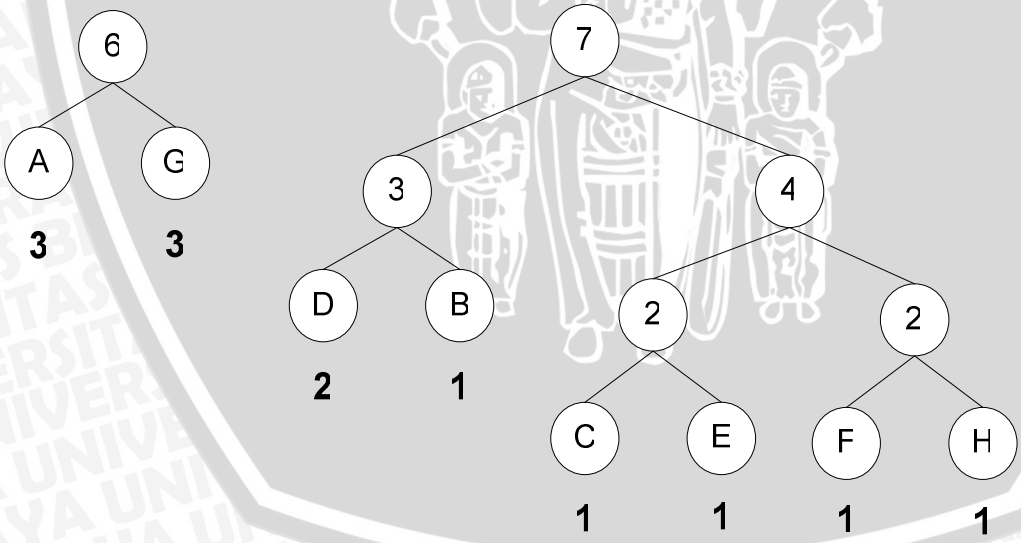
Gambar 2.5 Pembentukan Huffman *Tree* Langkah 4

Dua *node* terakhir (*node* 'A' dan 'B') yang sekarang mempunyai frekuensi terkecil digabungkan sehingga mempunyai frekuensi sebanyak 6. Terbentuk 3 *tree* yang tersisa yang ditunjukkan pada Gambar 2.6.



Gambar 2.6 Pembentukan Huffman *Tree* Langkah 5

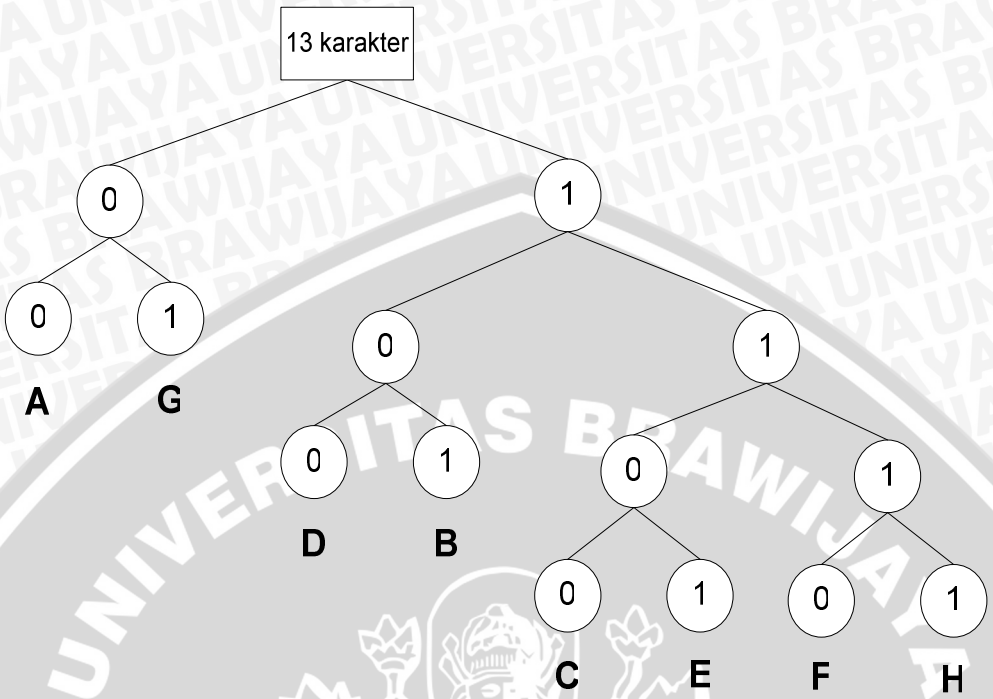
Terdapat 2 *tree* yang mempunyai frekuensi sebanyak 3 dan 4. Dari kedua *tree* tersebut digabungkan sehingga menghasilkan satu *tree* yang mempunyai frekuensi sebanyak 7, proses ditunjukkan pada Gambar 2.7.



Gambar 2.7 Pembentukan Huffman *Tree* Langkah 6

Langkah terakhir 2 *tree* yang tersisa digabungkan membentuk *tree* terakhir yang mempunyai frekuensi sebanyak 13 karakter. Kemudian setiap cabang diberi nilai 1 atau 0. Untuk cabang kiri bernilai 0, dan

cabang kanan bernilai 1. Sehingga hasil Huffman *tree* untuk *string* "ABACADGEGHDFG" ditunjukkan pada Gambar 2.8.



Gambar 2.8 Huffman *Tree* String "ABACADGEGHDFG"

2.5.2 Proses *Encoding*

Encoding adalah cara menyusun *string* biner dari teks yang ada (Wardoyo, 2005). Proses *encoding* untuk satu karakter dimulai dengan melihat Huffman *tree* yang telah dibentuk. Setelah itu, kode untuk satu karakter dibuat dengan menyusun nama *string* biner yang dibaca dari akar sampai ke daun dari Huffman *tree* tersebut.

Langkah-langkah untuk meng-*encoding* suatu string biner adalah sebagai berikut:

1. Tentukan karakter yang akan di-*encoding*
2. Baca dari akar, baca setiap bit yang ada pada cabang yang bersesuaian sampai ketemu *node* dimana karakter itu berada
3. Ulangi langkah 2 sampai seluruh karakter di-*encoding*. Sebagai contoh kita dapat melihat Tabel 2.2, yang merupakan hasil *encoding* untuk Huffman *tree* pada contoh sebelumnya.

Tabel 2.5 Tabel Huffman Hasil *Encoding*

Simbol	Frekuensi	Peluang	Kode Huffman
A	3	3/13	00
B	1	1/13	101
C	1	1/13	1100
D	2	2/13	100
E	1	1/13	1101
F	1	1/13	1110
G	3	3/13	01
H	1	1/13	1111

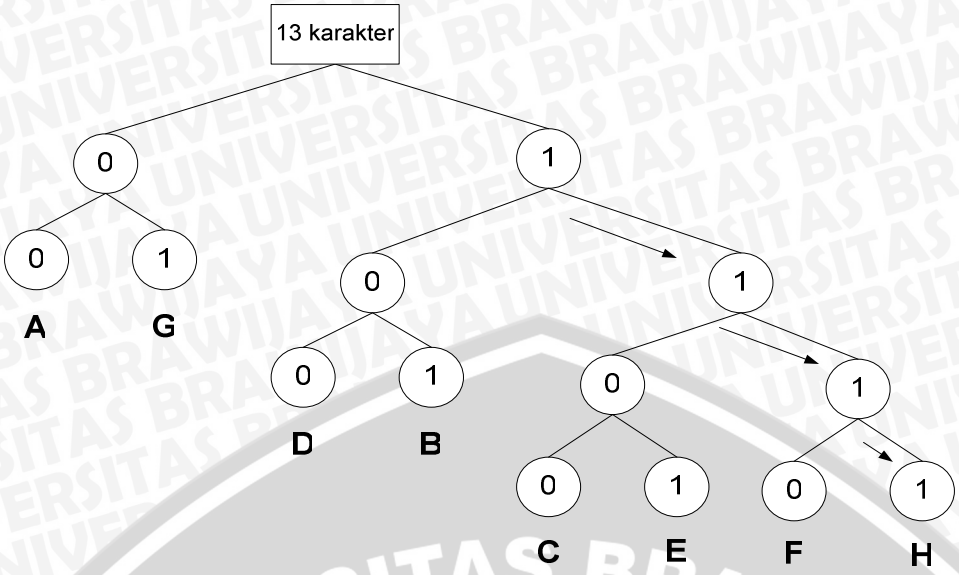
2.5.3 Proses *Decoding*

Decoding merupakan kebalikan dari *encoding*. *Decoding* berarti menyusun kembali data dari *string* biner menjadi sebuah karakter kembali. *Decoding* dapat dilakukan dengan dua cara, yaitu yang pertama dengan menggunakan pohon Huffman dan yang kedua dengan menggunakan tabel kode Huffman.

Langkah-langkah men-*decoding* suatu *string* biner dengan menggunakan Huffman *tree* menurut Wardoyo (2005) adalah sebagai berikut:

1. Baca sebuah bit dari *string* biner.
2. Untuk setiap bit pada langkah 1, lakukan pembacaan pada cabang yang bersesuaian.
3. Ulangi langkah 1 dan 2 sampai bertemu *node*. Kodekan rangkaian bit yang telah dibaca dengan karakter di daun.
4. Ulangi dari langkah 1 sampai semua bit di dalam *string* habis.

Sebagai contoh akan dilakukan pen-*decoding*-an string biner yang bernilai ”1111”. Setelah dilakukan penelusuran pada Huffman *tree,* maka kita akan menemukan bahwa string yang mempunyai kode Huffman “1111” adalah karakter H. Seperti ditunjukkan pada Gambar 2.9.



Gambar 2.9 Proses Decoding

Cara yang kedua adalah dengan menggunakan tabel kode Huffman. Sebagai contoh, akan digunakan kode Huffman pada Tabel 2.2 untuk merepresentasikan *string* "ABACA". Dengan menggunakan Tabel 2.2 *string* tersebut akan direpresentasikan menjadi rangkaian bit : 00 101 00 1100 00. Total jumlah bit yang dibutuhkan sebanyak 13 bit. Dari Tabel 2.2 tampak bahwa kode untuk sebuah simbol atau karakter tidak boleh menjadi awalan dari kode simbol yang lain untuk menghindari ambiguitas dalam proses dekompresi atau *decoding*. Karena tiap kode Huffman yang dihasilkan unik, maka proses *decoding* dapat dilakukan dengan mudah. Contoh: saat membaca kode bit pertama dalam rangkaian bit "0010100110000", didapatkan bit "0", cek kode "0" pada Tabel 2.2, karena tidak terdapat kode yang dimaksud pada Tabel 2.2, maka dilakukan pembacaan bit berikutnya. Didapatkan kode "00", yang kemudian dicocokkan kembali pada tabel dan dapat langsung disimpulkan merupakan pemetaan dari karakter "A". Kemudian baca kode bit selanjutnya, yaitu bit "1". Tidak terdapat kode Huffman "1", lalu baca kode bit selanjutnya, sehingga menjadi "10". Tidak terdapat juga kode Huffman "10", lalu baca lagi kode bit berikutnya, sehingga menjadi "101". Didapatkan rangkaian kode bit "110" adalah pemetaan dari simbol "B".

2.6 Bigram

Bigram merupakan salah satu bagian dari n-gram yang mempunyai ukuran sebanyak dua. Sedangkan n-gram adalah sebuah bagian dari rangkaian sebanyak n materi, yang diberikan pada sebuah rangkaian tertentu. Materi yang dimaksud dalam n-gram dapat berupa karakter, kata, maupun keduanya tergantung dari aplikasi. N-gram biasa digunakan dalam perhitungan statistik language proceessing, sedangkan untuk ouputnya dapat digunakan untuk Statistik mesin penerjemah dan pengecekkan ejaan (Hatem, 2007).

Contoh diberikan rangkaian karakter ”Arya Wibisana”, dari rangkaian tersebut didapatkan bigram yaitu : ”Ar”, ”ry”, ”ya”, ”a ”, ” W”, ”Wi”, ”ib”, ”bi”, ”is”, ”sa”, ”an”, ”na”.

2.7 Rasio Kompresi

Rasio kompresi merupakan perbandingan ukuran file asli (sebelum dikompresi) dan ukuran file setelah dikompresi. Persentase kompresi dapat dinyatakan dengan persamaan 2.1 (Sayood, 2000) :

$$R = (1 - (\frac{K}{A}) * 100\% \qquad (2.1)$$

Dimana: A = Ukuran file asli,
K = Ukuran file terkompresi
R = Persentase kompresi

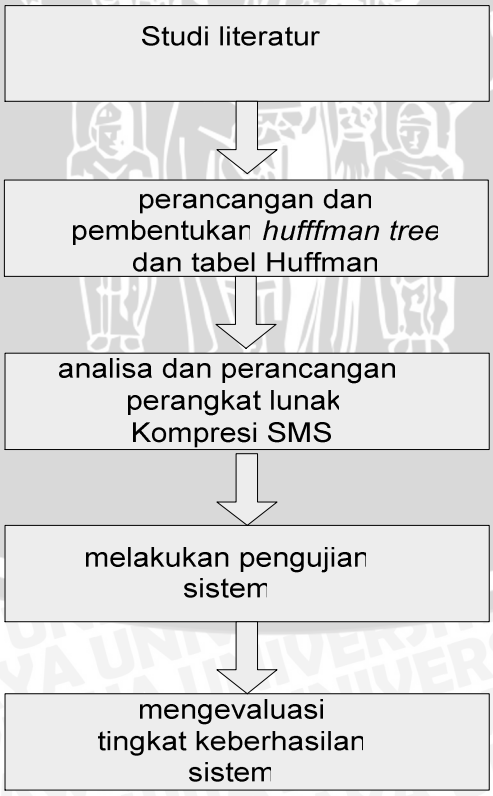
BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini akan dibahas metode perancangan yang digunakan dan langkah-langkah yang dilakukan untuk melakukan kompresi SMS dengan metode Huffman.

Penelitian dilakukan dengan tahapan-tahapan sebagai berikut:

1. Mempelajari literatur yang terkait dengan masalah Kompresi SMS dengan metode Huffman dan bigram.
2. Melakukan analisa perancangan dan pembentukan Huffman *tree* dan tabel Huffman yang sesuai dengan penulisan SMS.
3. Melakukan perancangan sistem kompresi SMS dengan tabel Huffman yang telah dibangun, dan mengimplementasikan menjadi perangkat lunak.
4. Melakukan uji coba sistem dengan mengirim dan menerima pesan SMS yang telah dikompresi melalui sistem.
5. Mengevaluasi tingkat keberhasilan sistem dalam melakukan kompresi SMS

Langkah-langkah yang dilakukan dapat dilihat pada Gambar 3.1



Gambar 3.1 Langkah-langkah penelitian

3.1 Analisa Sistem

Pada subbab analisis sistem ini akan dijelaskan mengenai deksripsi sistem, dan batasan sistem.

3.1.1 Deskripsi Sistem

Sistem yang akan dibangun berupa aplikasi SMS pada *Handphone* yang dapat melakukan kompresi SMS, mengirim SMS, menerima SMS dan kemudian men-*dekompres*-nya untuk mengembalikan file seperti semula. Proses kompresi dilakukan sebelum SMS dikirim, yang bertujuan agar *size* dari SMS dapat lebih kecil sehingga mengurangi biaya pengiriman SMS. Perangkat lunak yang dibangun ditunjukan bagi pengguna telepon genggam khususnya pengguna SMS.

Metode kompresi yang dilakukan menggunakan prinsip dasar dari kode Huffman, yaitu tiap karakter dikodekan dengan rangkaian beberapa bit, dimana karakter yang sering dipakai dikodekan dengan rangkaian bit yang pendek, dan karakter yang jarang dipakai dikodekan dengan rangkaian bit yang lebih panjang. Sistem akan menggunakan metode Huffman yang mempunyai tabel tetap, oleh karena itu akan dilakukan analisa pemakaian karakter-karakter yang sering dipakai ketika orang melakukan SMS. Selanjutnya dilakukan pembentukan tree dan tabel Huffman berdasarkan analisa dan prediksi karakter tersebut.

Sistem juga akan mencari perulangan berupa bigram pada teks SMS. Bigram yang didapat akan dikodekan dengan kode Huffman. Oleh karena itu pembentukan tree dan tabel Huffman tidak hanya dilakukan pada karakter, tetapi dilakukan juga pada Bigram. Bigram tersebut akan dibuat kode Huffmannya dari kode yang telah dikosongkan dan belum terpakai oleh karakter lain. Bigram-bigram yang telah didapat akan dibuat sebagai header yang digunakan untuk mengembalikan nilai bigram yang telah dikodekan pada saat proses dekompresi nanti.

Untuk mengurangi kesalahan prediksi atas karakter yang telah dibuat tabel Huffmannya, maka akan dibuat 2 tabel huffman yang saling berlawanan. Pada saat teks SMS akan dikompres, sistem akan memilih diantara 2 tabel yang menghasilkan rasio kompresi paling baik

Setelah *tree* dan tabel Huffman terbentuk, maka pada tahap selanjutnya adalah melakukan analisa dan perancangan perangkat lunak, perangkat lunak yang dibuat akan melakukan kompresi sebelum SMS dikirim ke operator seluler, kemudian perangkat lunak akan melakukan dekompresi hasil kompresi SMS yang diterima oleh pengguna SMS. Langkah-langkah yang dilakukan sistem secara umum dijelaskan pada Gambar 3.2.

Gambar 3.2 Gambaran umum sistem

3.1.2 Batasan Sistem

Batasan sistem yang akan dikembangkan adalah:

1. Karakter yang akan dikompresi adalah karakter-karakter umum yang digunakan untuk SMS.
2. Analisa pemakaian karakter pada SMS memakai bahasa sehari-hari dalam bahasa Indonesia
3. Jumlah tabel Huffman yang dibuat sebatas dua.
4. Kode Huffman yang dibentuk untuk bigram sebanyak 11.

3.2 Perancangan *Tree* Dan Tabel Huffman

3.2.1 Analisa Pemakaian Karakter

SMS menggunakan kode ASCII 7 bit yang mempunyai variasi karakter sebanyak 128, sedangkan untuk karakter-karakter yang dipakai dalam SMS sebanyak 95, oleh karena itu dalam

pembentukan Huffman *tree* yang akan dilakukan memakai 95 karakter tersebut, yang terdiri dari :

1. 26 kode untuk huruf kapital *(upper case)* dari A –Z.
2. 26 kode untuk huruf kecil *(lower case)* dari a –z.
3. 10 digit desimal dari 0 –9.
4. 34 karakter tanda baca.

Karena tidak semua karakter umum dipakai dalam penulisan SMS. Pemakaian ke 95 karakter dalam penggunaan SMS dapat dianalisis. Analisa pemakaian karakter pada SMS diasumsikan sebagai berikut :

1. Prioritas pertama adalah huruf kecil (a-z), karna hampir selalu digunakan dalam setiap penulisan SMS.
2. Prioritas kedua adalah tanda baca yang sering muncul dalam SMS, seperti : ” ”(spasi). ”.”(titik) dan ”,”(koma).
3. Prioritas ketiga adalah, angka (0-9), dan huruf besar (A-Z), karena jika dilihat dari teks SMS, penggunaan huruf besar hanya terdapat di awal kalimat atau setelah titik.
4. Prioritas terakhir adalah tanda baca yang sangat jarang terpakai dalam penulisan SMS, seperti : ”#”, ”~”, ”^”.

3.2.2 Analisa Penggunaan Perulangan Bigram

Dalam penulisan SMS banyak terdapat perulangan kata yang dipakai. Dengan memanfaatkan kondisi ini, kata-kata yang sering diulang dalam penulisan akan dikodekan dengan sejumlah bit yang lebih pendek. Akan dibuat *header* dalam teks SMS yang akan menginisialisasi kata-kata tersebut. *Header* ini berfungsi pada saat proses dekompresi nanti.

Karena tabel Huffman yang dipakai adalah tabel Huffman yang tetap, maka kode yang dibentuk bersifat tetap. Begitu juga kode yang dibentuk untuk bigram juga tetap, hanya bigram-nya saja yang berbeda. Dari perulangan kata ini akan dibuat 12 kode Huffman yang terdiri dari :

1. Satu kode sebagai bit penanda, yang akan digunakan sebagai tanda pembuka dan penutup *header*.
2. Sebelas kode berupa perulangan bigram kata.

Pembentukan kode untuk bigram hanya dibuat sebanyak sebelas kode, karena bila jumlah kode diperbanyak akan memanjangkan panjang kode huffman untuk bigram tersebut, sedangkan untuk bigram terakhir yaitu bigram ke-11 sudah mempunyai panjang

sebanyak 9 bit dan belum ditambahkan dengan panjang header. Sehingga kompresi nantinya akan tidak efektif jika jumlah kode untuk bigram ditambah.

3.2.3 Analisa Perancangan *Tree* Dan Tabel Huffman

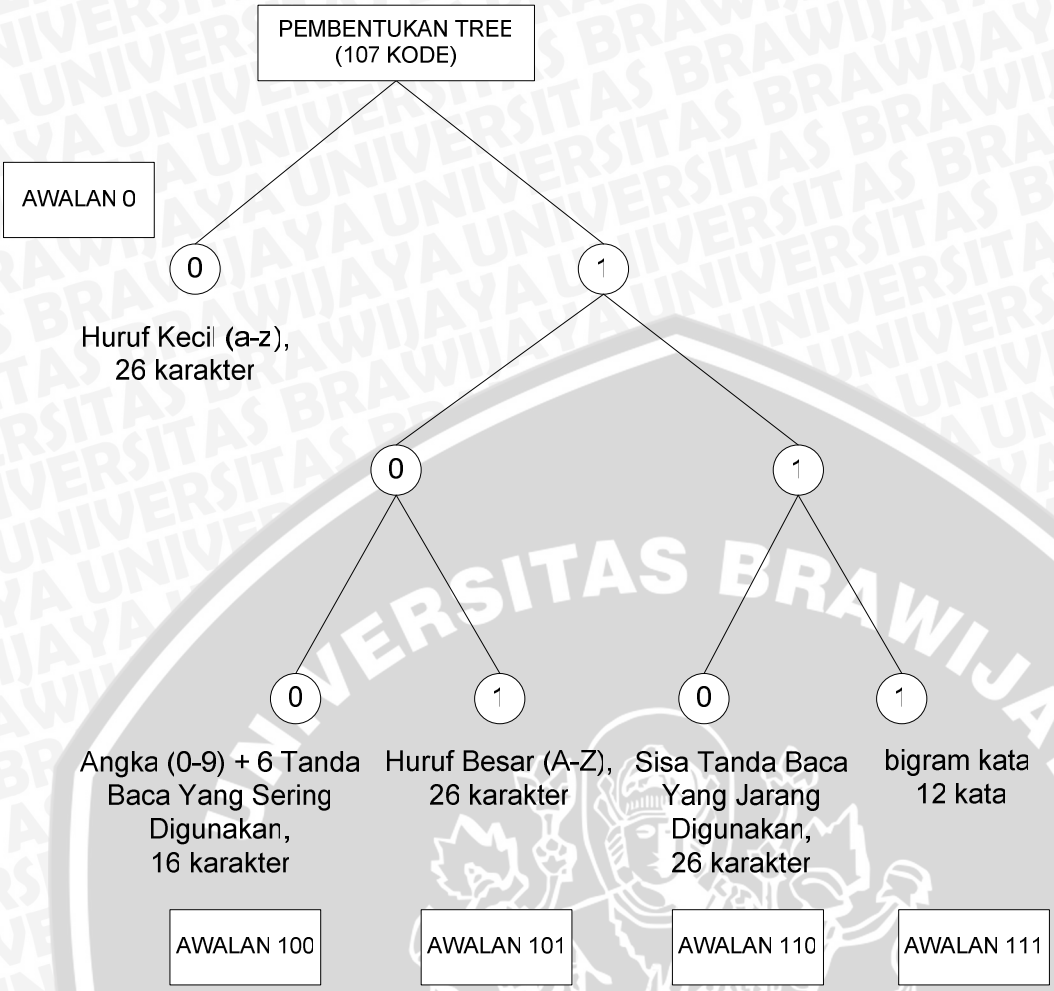
Berdasarkan analisa pada subbab 3.2.1 dan subbab 3.2.2, maka terdapat 95 kode dari karakter dan 12 kode dari bigram sehingga total terdapat 107 kode yang akan dibentuk. Untuk mempermudah pengaturan pembagian kode kompresi maka dari 107 kode akan dibagi menjadi 5 bagian awalan kode, bagian awalan tersebut adalah: 0, 100, 101,110 dan 111.

Berdasarkan ke-5 bagian awalan kode, bagian yang mempunyai awalan 0 dijadikan sebagai pusat kompresi, karena bit yang digunakan paling sedikit dari bagian awalan yang lain. Artinya bagian awalan 0 ditempatkan untuk karakter-karakter yang mempunyai prioritas utama. Untuk 4 bagian yang lain, penempatannya berdasarkan analisa prioritas yang telah dijelaskan sebelumnnya ditambah dengan satu bagian dari kata-kata.

Dari analisa pemakaian karakter, huruf kecil (a-z) merupakan prioritas utama, tetapi dalam penggunaan SMS sering kali ada penulisan yang memakai huruf besar (A-Z) pada semua hurufnya. Oleh karena itu, akan dibentuk 2 tabel Huffman yang berbeda berdasarkan pemakaian huruf. Untuk tabel Huffman 1 menggunakan prediksi yang telah dilakukan sebelumnnya, dengan huruf kecil sebagai prioritas utama kompresi. Sedangkan pada tabel Huffman 2 akan memakai huruf besar (A-Z) sebagai prioritas utama kompresi.

3.2.4 Perancangan Tabel Huffman 1

Tabel Huffman 1 memakai huruf kecil (a-z) sebagai prioritas utama, artinya huruf kecil akan mendapat awalan 0 sebagai pusat kompresi. Gambaran pembagian pembentukan Huffman *tree* untuk tabel Huffman 1 secara umum ditunjukkan pada Gambar 3.3



Gambar 3.3 Pembagian pembentukan Huffman *tree* 1

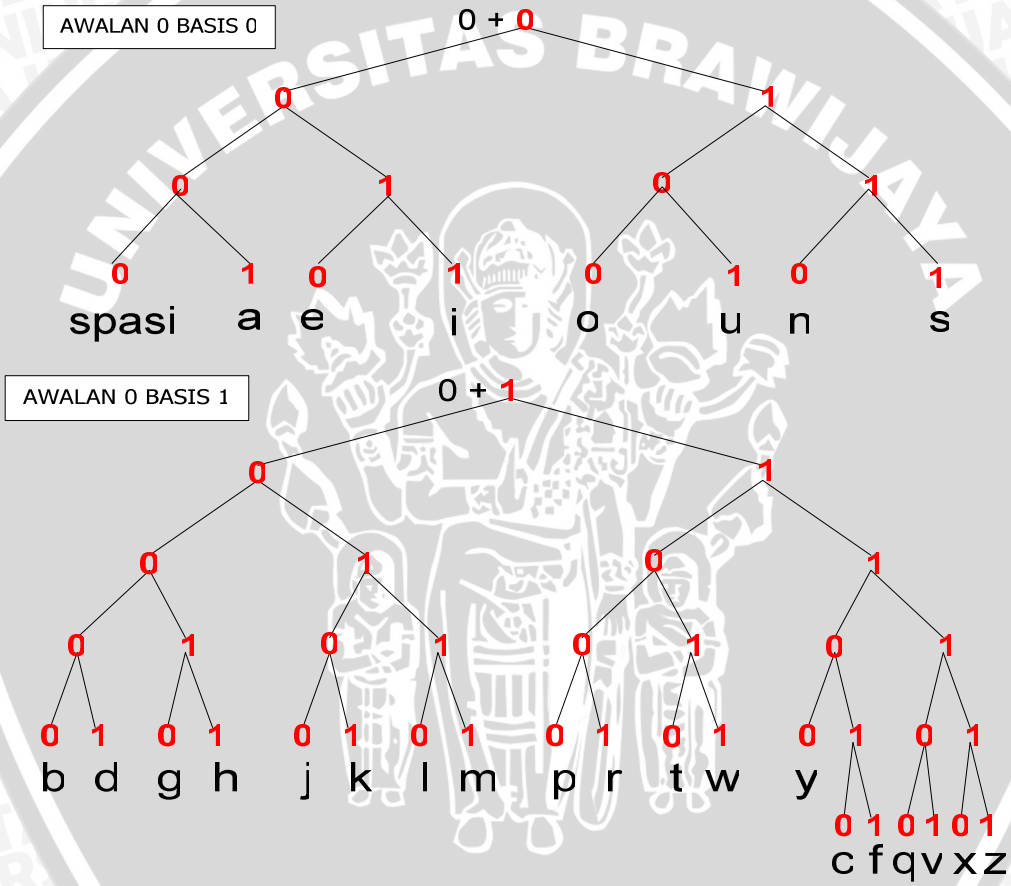
3.2.4.1 Pembentukan *Tree* Dan Tabel Huffman 1 Awalan 0

Awalan 0 ditempatkan untuk karakter-karakter dengan prioritas utama, karena pada awalan 0, kompresi bit per karakter paling banyak dilakukan. Sesuai dengan prioritas yang telah dianalisis sebelumnya, awalan 0 ini ditempatkan pada huruf kecil yang sering sekali dipakai pada penulisan SMS.

Dalam pemakaian huruf kecil a sampai z, tidak semua mempunyai perbandingan pemakaian yang sama. Oleh karena itu dalam pemakaian huruf juga dilakukan analisa berdasarkan prioritas penggunaan. Analisa pemakaian huruf pada SMS mengacu pada dasar teori yang telah ditulis pada bab 2. Sehingga prediksi pemakaian huruf diasumsikan sebagai berikut :

1. Prioritas utama adalah semua huruf vokal (“a”, “i”, “u”, “e”, “o”), dan beberapa huruf konsonan yang paling sering digunakan yaitu “n” dan “s”.
2. Prioritas kedua adalah huruf konsonan yang mempunyai tingkatan sedang dalam penggunaannya. yaitu : “b”, “d”, “g”, “h”, “j”, “k”, “l”, “m”, “p”, “r”, “t”, “w”, “y”.
3. Prioritas ketiga merupakan huruf konsonan yang sangat jarang dipakai dalam penulisan, yaitu : “x”, “z”, “c”, “q”, dan “f”.

Berdasarkan urutan prioritas maka dapat dibentuk *tree* dengan awalan 0 untuk setiap karakter, sesuai dengan Gambar 3.4 .



Gambar 3.4 Pembentukan *tree* awalan 0

Dari *tree* yang telah dibentuk pada Gambar 3.3 maka dapat dibentuk tabel Huffman seperti ditunjukkan pada Tabel 3.1:

Tabel 3.1 Tabel Huffman 1 awalan 0

Karakter	Awalan	Badan	Kode
a	0	0010	00010
b		10000	010000
c		111010	0111010
d		10001	010001
e		0010	00101
f		111011	0111011
g		10010	010010
h		10011	010011
i		0011	00011
j		10100	010100
k		10101	010101
l		10110	010110
m		10111	010111
n		0110	00110
o		0100	00100
p		11000	011000
q		111100	0111100
r		11001	011001
s		0111	00111
t		11010	011010
u		0101	00101
v		111101	0111101
w		11011	011011
x		111110	0111110
y		11100	011100
z		111111	0111111
spasi		0000	00000

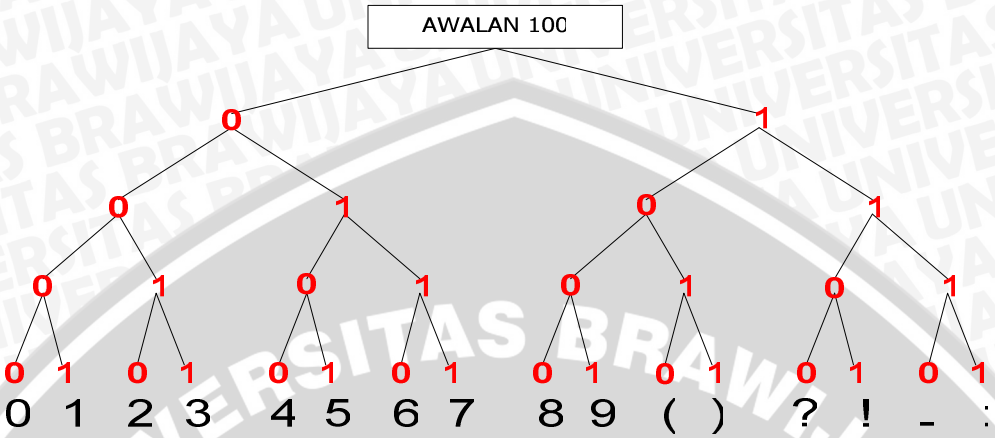
3.2.4.2 Pembentukan *Tree* Dan Tabel Huffman 1 Awalan 100

Awalan 100 ditempatkan untuk karakter-karakter dengan prioritas sedang, dengan kata lain kemungkinan pemakaian karakter-karakter ini ada, tetapi tidak sering digunakan. Dari awalan 100 akan dibentuk 16 kode karakter yang terdiri dari :

1. 10 angka (dari 0 sampai 9)

2. 6 tanda baca yang sering dipakai dalam penulisan. yaitu: "(", ")", "?", "!", "-", ":".

Dari 16 karakter tersebut dapat dibentuk *tree* dengan awalan 100 untuk setiap karakter, sesuai dengan Gambar 3.5 .



Gambar 3.5 Pembentukan *tree* awalan 100

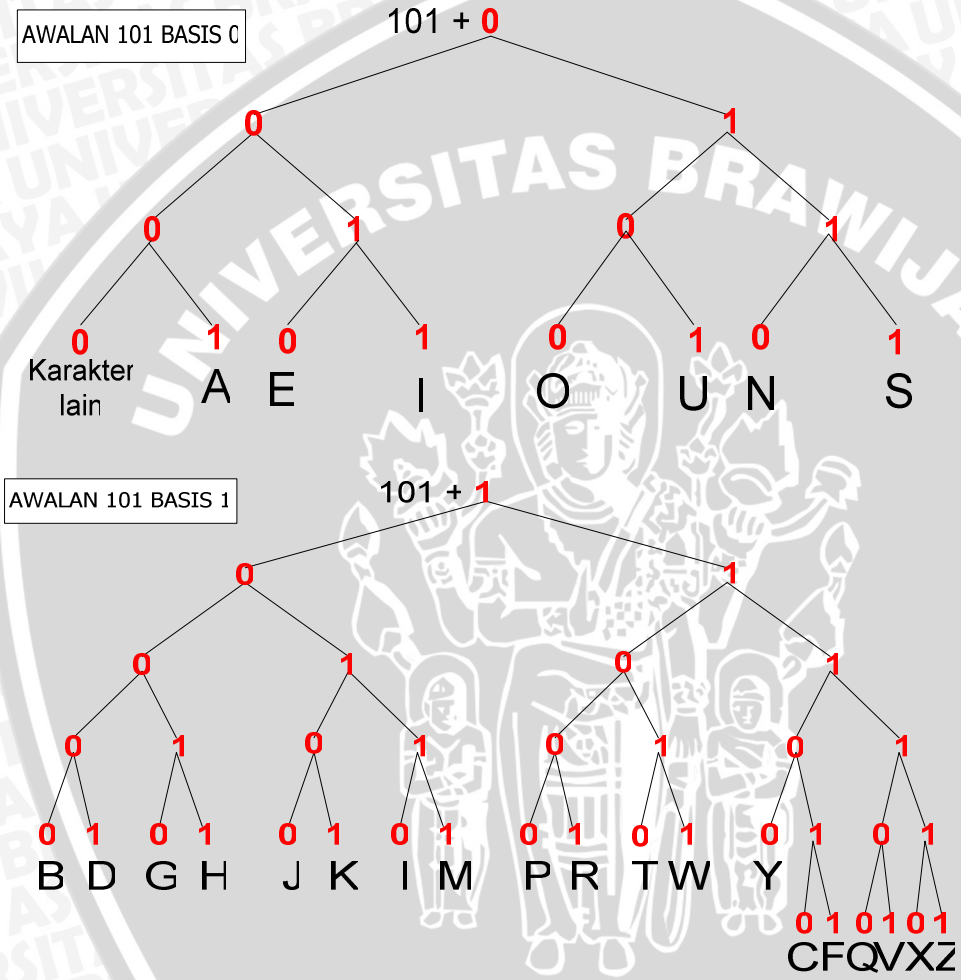
Dari *tree* yang telah dibentuk pada Gambar 3.5 maka dapat dibentuk tabel Huffman seperti ditunjukkan pada Tabel 3.2.

Tabel 3.2 Tabel Huffman 1 awalan 100

Karakter	Awalan	Badan	Kode
0	100	0000	1000000
1		0001	1000001
2		0010	1000010
3		0011	1000011
4		0100	1000100
5		0101	1000101
6		0110	1000110
7		0111	1000111
8		1000	1001000
9		1001	1001001
(		1010	1001010
)		1011	1001011
?		1100	1001100
!		1101	1001101
-		1110	1001110
:		1111	1001111

3.2.4.3 Pembentukan *Tree* Dan Tabel Huffman 1 Awalan 101

Awalan 101 ditempatkan untuk huruf-huruf besar dari A sampai Z. Dengan memakai analisa prioritas huruf yang telah dijelaskan sebelumnya pada analisa huruf kecil, huruf A sampai Z dibagi juga menjadi 3 prioritas. Oleh karena itu untuk pembentukan *tree* pada awalan 101 menyerupai pembentukan *tree* pada awalan 0. Proses pembentukan *tree* dapat dilihat pada Gambar 3.6.



Gambar 3.6 Pembentukan *tree* awalan 101

Kode 1010000 ditempatkan untuk karakter lain pada penulisan sms yang berupa simbol langka atau karakter unikode lainnya. Kode untuk karakter tersebut diberi awalan 1010000 diikuti kode karakter aslinya sebanyak 16 bit. Dari penggabungan tersebut akan menghasilkan kode khusus yang akan diproses lebih lanjut.

Dari hasil pembentukan *tree* pada Gambar 3.7 dapat dibentuk tabel Huffman seperti ditunjukkan pada Tabel 3.3.

Tabel 3.3 Tabel Huffman 1 awalan 101

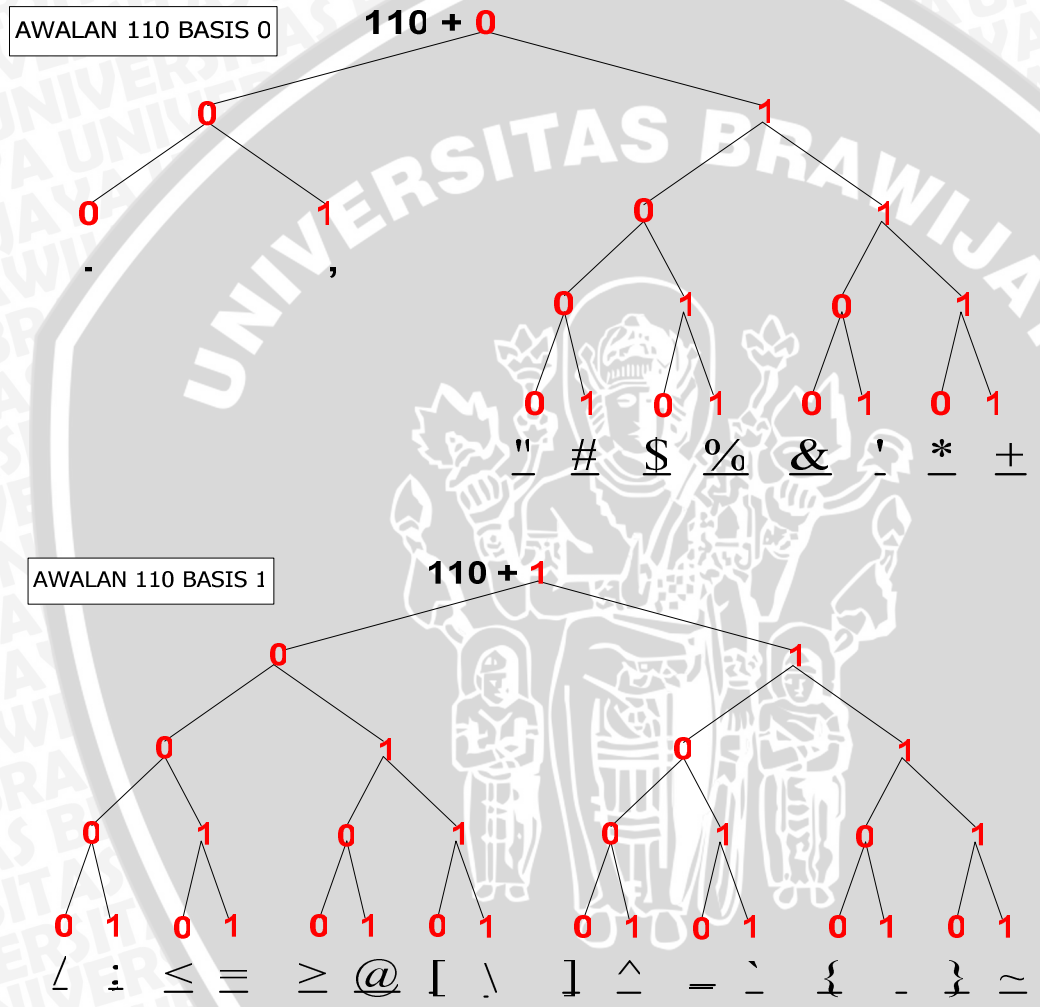
Karakter	Awalan	Badan	Kode
A	101	0001	101000
B		10000	10110000
C		111010	101111010
D		10001	10110001
E		0001	1010001
F		111011	101111011
G		10010	10110010
H		10011	10110011
I		0011	1010011
J		10100	10110100
K		10101	10110101
L		10110	10110110
M		10111	10110111
N		0110	1010110
O		0100	1010100
P		11000	10111000
Q		111100	101111100
R		11001	10111001
S		0111	1010111
T		11010	10111010
U		0101	1010101
V		111101	101111101
W		11011	10111011
X		111110	101111110
Y		11100	10111100
Z		111111	101111111
Karakter Lain		0000	1010000

3.2.4.4 Pembentukan *Tree* Dan Tabel Huffman 1 Awalan 110

Awalan 110 ditempatkan untuk karakter-karakter sisa yang kesemuanya merupakan tanda baca. Dari 95 karakter yang ada sebanyak 69 karakter telah dibuat kode Huffmannya, sisa karakter sebanyak 26 akan dibangun *tree*-nya dengan asumsi sebagai berikut :

1. Dari sisa 26 karakter yang berupa tanda baca terdapat 2 karakter yang menjadi prioritas utama karena sering dipakai dalam penulisan, yaitu : “.”(titik) dan “,”(koma).
2. Sisa 24 karakter akan dikodekan dengan jumlah bit yang sama, karakter-karakter ini merupakan karakter yang sangat jarang dipakai dalam penulisan.

Dari kriteria diatas maka pembentukan *tree* ditunjukan pada Gambar 3.7.



Gambar 3.7 Pembentukan *tree* awalan 110

Dari hasil pembentukan *tree* pada Gambar 3.8 dapat dibentuk tabel Huffman seperti ditunjukkan pada Tabel 3.4.

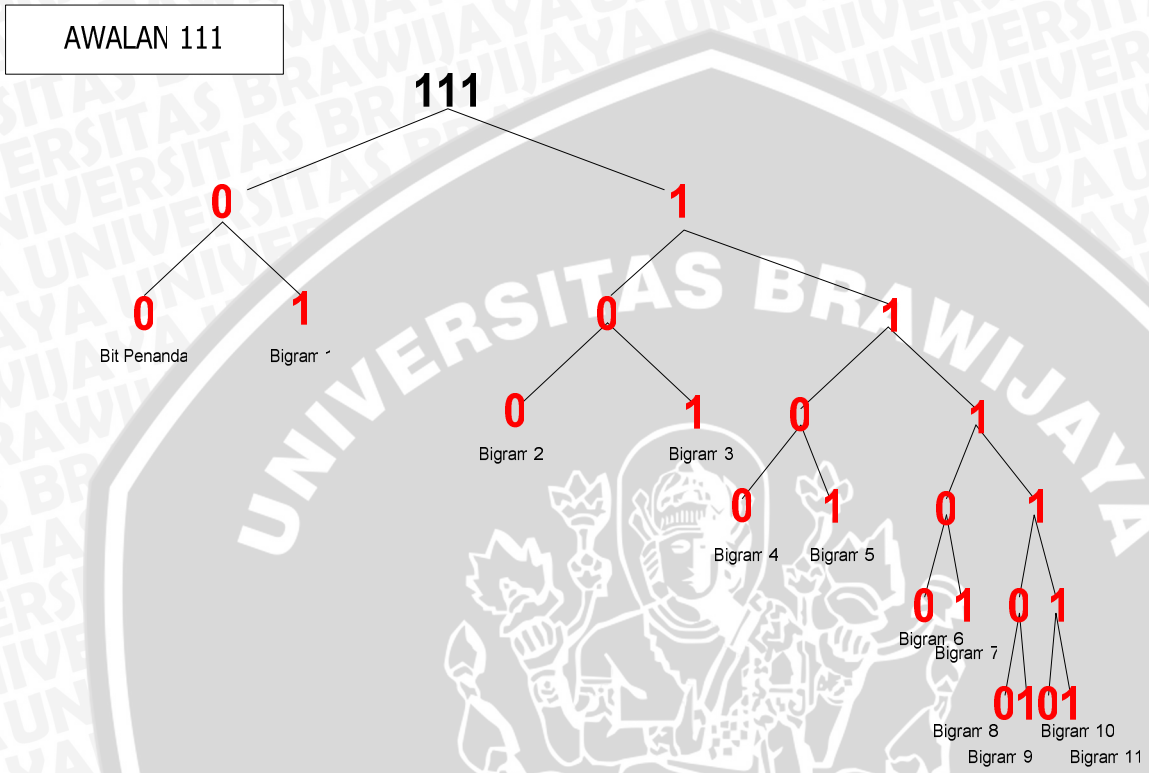
Tabel 3.4 Tabel Huffman 1 awalan 110

Karakter	Awalan	Badan	Kode
.	110	000	110000
,		001	110001
"		01000	11001000
#		01001	11001001
$		01010	11001010
%		01011	11001011
&		01100	11001100
'		01101	11001101
*		01110	11001110
+		01111	11001111
/		10000	11010000
:		10001	11010001
<		10010	11010010
=		10011	11010011
>		10100	11010100
@		10101	11010101
[		10110	11010110
\		10111	11010111
]		11000	11011000
^		11001	11011001
_		11010	11011010
`		11011	11011011
{		11100	11011100
\| | | 11101 | 11011101 |
} | | 11110 | 11011110 |
~ | | 11111 | 11011111 |

3.2.4.5 Pembentukan *Tree* Dan Tabel Huffman 1 Awalan 111

Terdapat 12 kode yang akan dibentuk dan dibangun *tree*-nya dengan memakai awalan 111. Seperti yang dijelaskan pada subbab 3.2.2 akan disediakan satu kode sebagai bit penanda, kemudian dari 11 kode sisanya akan dibentuk sesuai prinsip kode Huffman. Bigram yang mempunyai frekuensi perulangan paling banyak akan dikodekan dengan bit yang lebih pendek. Oleh karena itu nantinya

akan ada proses sorting berdasarkan frekuensi perulangan. bigram 1 adalah bigram yang mempunyai frekuensi perulangan paling tinggi, sedangkan bigram 11 adalah bigram yang mempunyai frekuensi perulangan paling rendah. Sehingga dapat dibentuk *tree* seperti pada Gambar 3.8.



Gambar 3.8 Pembentukan *tree* awalan 111

Dari hasil pembentukan *tree* pada Gambar 3.9 dapat dibentuk tabel Huffman seperti ditunjukkan pada Tabel 3.5.

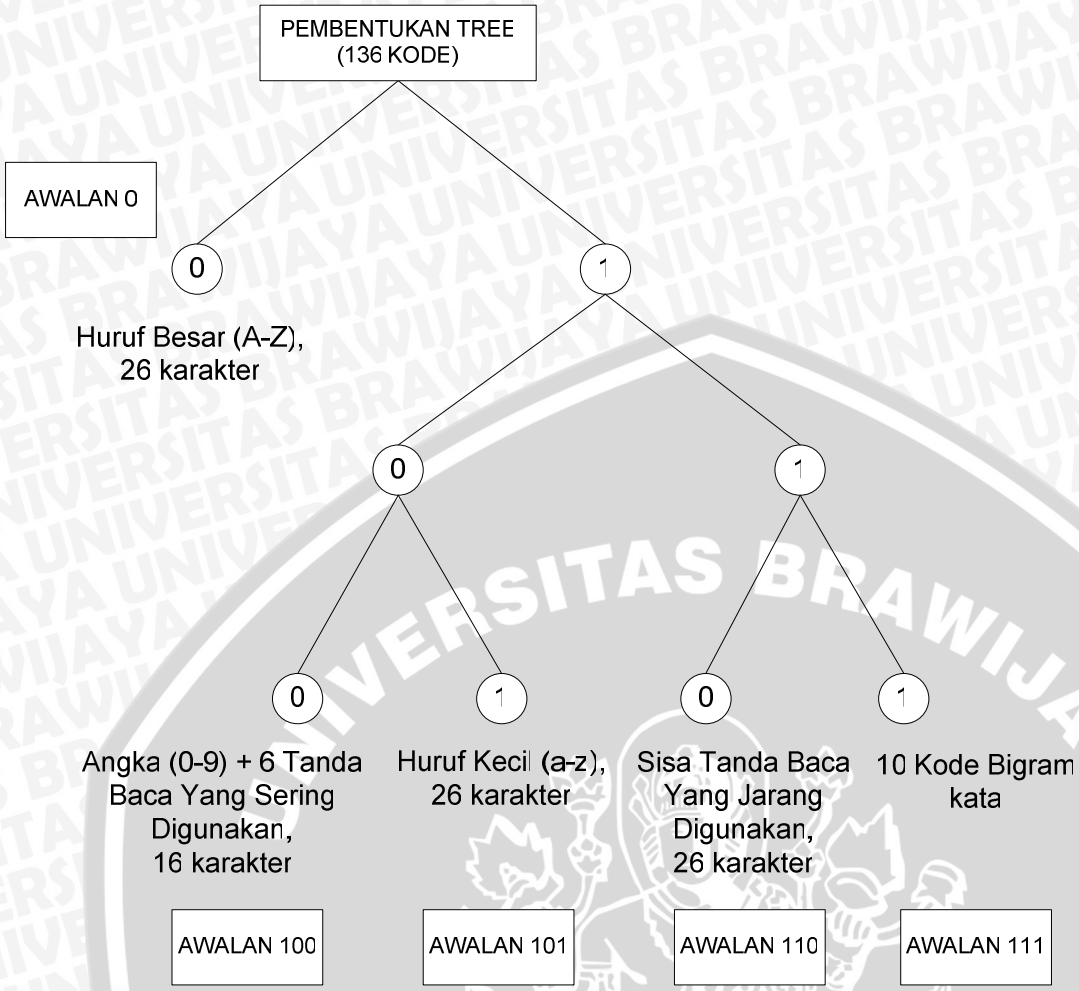
Tabel 3.5 Tabel Huffman 1 awalan 111

Kata	Awalan	Badan	Kode
Bit Penanda Header	111	00	11100
Bigram 1		01	11101
Bigram 2		100	111100
Bigram 3		101	111101
Bigram 4		1100	1111100
Bigram 5		1101	1111101
Bigram 6		11100	11111100
Bigram 7		11101	11111101
Bigram 8		111100	111111100
Bigram 9		111101	111111101
Bigram 10		111110	111111110
Bigram 11		111111	111111111

3.2.5 Perancangan Tabel Huffman 2

Tabel Huffman 2 memakai huruf besar (A-Z) sebagai prioritas utama, artinya huruf besar akan mendapat awalan 0 sebagai pusat kompresi. Sedangkan untuk huruf kecil mendapat awalan 101. Selanjutnnya untuk pembentukan tabel Huffman 2 akan sama prosesnya seperti pada tabel Huffman 1.

Gambaran pembagian pembentukan Huffman *tree* untuk tabel Huffman 2 secara umum ditunjukkan pada Gambar 3.9



Gambar 3.9 Pembagian pembentukan Huffman *tree* 2

Dalam penggunaan tabel Huffman 2 terdapat satu kode tambahan sebagai bit penanda yang akan digunakan untuk menandakan bahwa tabel yang dipakai adalah tabel Huffman 2 pada proses dekompresi. Kode tersebut ditempatkan pada awalan 101 yang merupakan kode untuk huruf kecil (a-z) dan dibangun dengan membuat cabang dari tree huruf "y". Sehingga penambahan pada tabel Huffman 2 ditunjukkan pada Tabel 3.6.

Tabel 3.6 Penambahan tabel Huffman 2 awalan 111

Karakter	Awalan	Badan	Kode
y	101	111001	101111001
Bit Penanda		111000	101111000

3.3 Perancangan Sistem

Pada subbab ini akan dijelaskan mengenai berbagai proses yang terjadi dalam membangun sistem kompresi SMS, antara lain membahas tentang proses kompresi, proses dekompresi beserta struktur data dan *flowchart*-nya.

3.3.1 Struktur Data

Struktur data dari sistem yang akan dibangun ditunjukkan pada Tabel 3.7.

Tabel 3.7 Struktur Data

Variabel	Tipe Data
cc	array of String[128]
kode_huffman2	array of String[127]
cari_kata	String
cari_kode	integer
Bigram	array of String[30]
Kode_Bigram	array of Stting[10]
Jum_bigram	integer

1. Tabel Huffman 1 akan menggunakan 2 *array* untuk menyimpan kode huffman yang telah dibentuk, *array* pertama bertipe *string* yaitu *array* cc[0..127], *array* cc indeks ke 32-127 adalah kode untuk karakter, *Array* kedua yaitu *array* kode_bigram digunakan sebagai kode untuk bigram yang akan dicari
2. Tabel Huffman 2 menggunakan struktur data yang sama seperti tabel Huffman 1. Yaitu dua *array* bertipe string: *array* cc2[0..128] dan *array* kode_bigram[0..10].
3. Variabel *string* cari_kata, untuk pengecekan kode yang diterjemahkan adalah kata atau karakter.
4. Variabel *integer* cari_kode, yang digunakan untuk proses dekompresi
5. Variabel *integer* jum_bigram, digunakan sebagai jumlah bigram yang didapat.

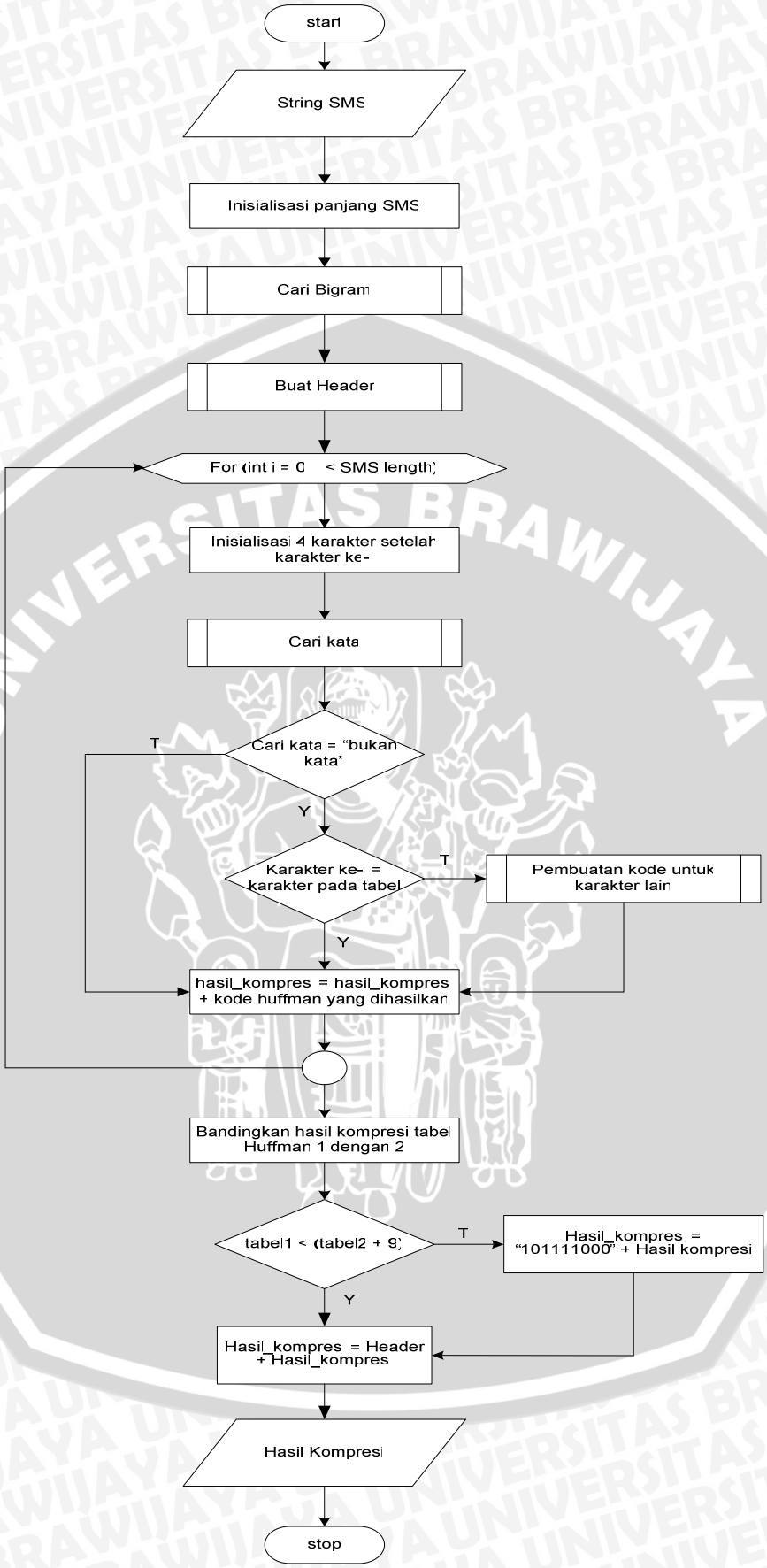
3.3.2 Perancangan Proses Kompresi

Proses kompresi dilakukan sebelum pesan SMS akan dikirim. Setelah pengguna SMS selesai menulis pesan, pesan akan dihitung

dengan memakai 2 tabel yang telah dibentuk sebelumnya. Untuk pemakaian tabel Huffman 2, pada awalan SMS akan ditambahkan bit penanda yaitu : ”101111000”. Dari kompresi kedua tabel tersebut, tabel yang menghasilkan kompresi yang paling baik adalah tabel yang dipakai. Langkah-langkah yang dilakukan adalah sebagai berikut:

1. Input data *string* teks_SMS
2. Lakukan pencarian bigram.
3. Lakukan proses sorting terhadap bigram yang telah ditemukan berdasarkan frekuensi perulangan.
4. Lakukan pembuatan *header* dari bigram yang ditemukan
5. Lakukan perulangan dari karakter pertama sampai karakter terakhir pada teks SMS.
6. Lakukan pengecekan pada setiap karakter dengan memanggil fungsi cari_kata, apakah karakter tersebut merupakan bigram yang telah dicari sebelumnnya atau tidak. Jika ya maka cari_kata akan mengembalikan nilai berupa kode huffman dari bigram yang dimaksud, dan akan ditambahkan pada *string* hasil_kompresi.
7. Jika bukan bigram, lakukan pengecekan pada setiap karakter, apakah karakter tersebut, ada di dalam tabel Huffman yang telah dibuat. Jika tidak ada maka memanggil fungsi karakter_sisa, fungsi karakter_sisa akan membuat nilai baru untuk kode huffman karakter tersebut dengan fungsi tertentu, dan kode yang telah dibuat ditambahkan pada *string* hasil_kompresi.
8. Jika karakter tersebut ada maka tambahkan kode Huffman karakter tersebut pada *string* hasil_kompresi.
9. Output *string* hasil_kompresi
10. Bandingkan Output *string* hasil_kompresi menggunakan tabel Huffman 1 dengan *string* hasil_kompresi menggunakan tabel Huffman 2. Jika penggunaan tabel Huffman 2 menghasilkan kompresi yang lebih baik, maka pada awal *string* hasil_kompresi ditambahkan bit penanda : ”101111000”.
11. Tambahkan header didepan teks hasill kompresi.
12. *Output* adalah *string* hasil_kompresi yang telah dibandingkan.

Proses Kompresi dengan metode Huffman secara keseluruhan dijelaskan pada Gambar 3.10.



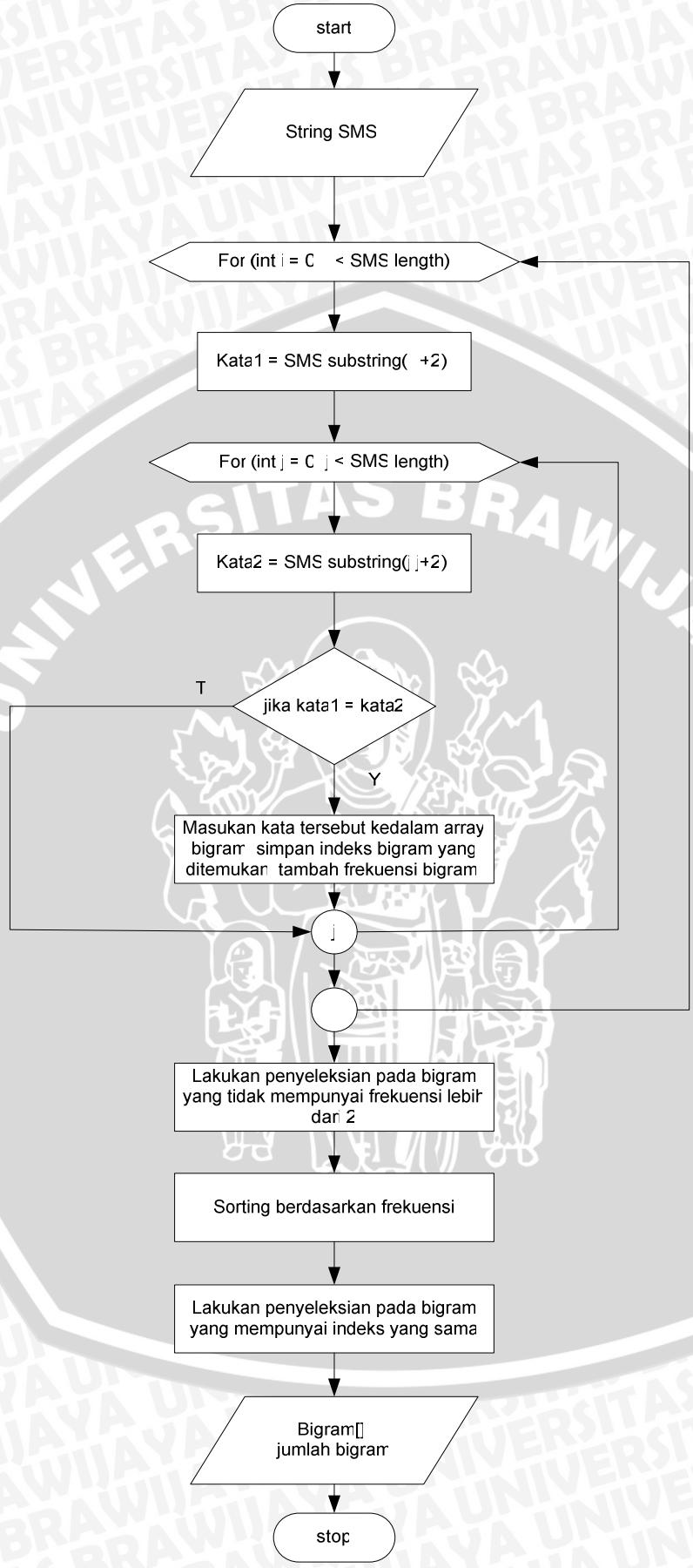
Gambar 3.10 *Flowchart* proses kompresi

3.3.2.1 Proses Pencarian Bigram

Dari pembentukan kode Huffman didapatkan kode untuk bigram sebanyak 9. Jika ditemukan bigram lebih dari 9 maka akan diambil bigram yang terbaik, yaitu yang mempunyai frekuensi lebih besar. Untuk itu akan dilakukan proses *sorting* berdasarkan frekuensi.

Bigram yang dicari tidak bisa memiliki indeks yang sama, karena akan merusak perhitungan frekuensi. maka jika ditemukan bigram yang saling berdekatan, juga akan diambil bigram yang memiliki frekuensi paling banyak melalui proses *sorting*. Proses pencarian bigram dapat dilihat pada Gambar 3.11. Dan langkah-langkah yang dilakukan adalah sebagai berikut :

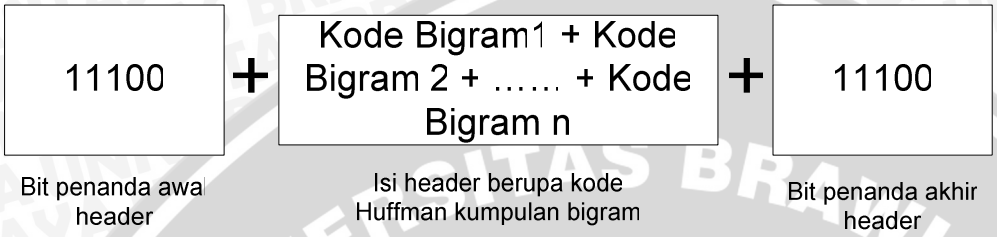
1. Input data *string* teks_SMS
2. Lakukan pencarian bigram dengan melakukan *looping* bercabang, inisialisasi variabel kata1 dan kata2, yang merupakan bigram dari indeks yang berbeda.
3. Lakukan perbandingan antara variabel kata1 dan kata 2, jika sama maka tambahkan frekuensi kata tersebut, masukkan kata tersebut kedalam *array* bigram dan simpan indeks bigram tersebut.
4. Jika tidak sama maka lanjutkan *looping* dengan bigram kata yang berbeda.
5. Lakukan penyeleksian terhadap bigram yang tidak memiliki frekuensi perulangan sebanyak batasan yang diberikan, bigram tersebut kemudian akan dihapus dari *array* bigram.
6. Lakukan *Sorting* berdasarkan frekuensi.
7. Lakukan penyeleksian pada bigram yang mempunyai indeks yang sama.
8. Output *array* bigram dan Jumlah bigram.



Gambar 3.11 *Flowchart* proses pencarian bigram

3.3.2.2 Proses Pembuatan *Header*

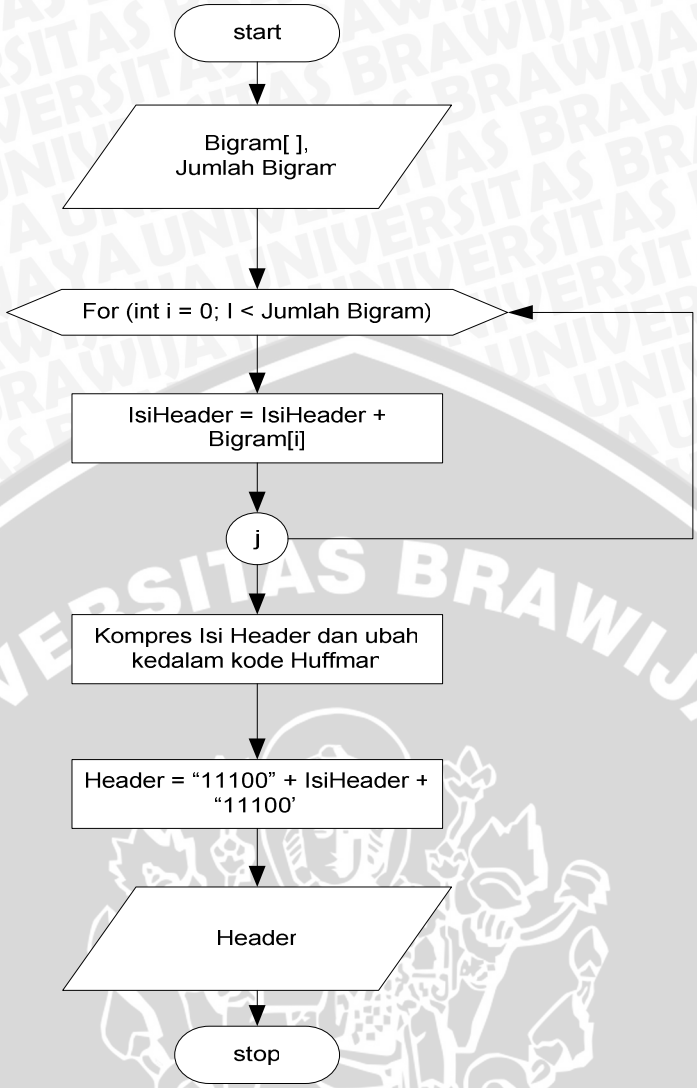
Pembuatan *header* digunakan saat proses dekompresi, dengan tujuan untuk menginisialisasi bigram yang dipakai dalam proses kompresi Header diawali dan diakhiri dengan bit penanda yaitu "11100". Isi dari header adalah bigram kata yang telah dicari pada proses kompresi. Struktur dari header dapat dilihat pada Gambar 3.12 berikut



Gambar 3.12 Struktur *header*

Untuk lebih jelasnya *flowchart* proses pembuatan *header* dapat dilihat pada Gambar 3.11. Dan langkah-langkah yang dilakukan adalah sebagai berikut

1. Input *array* Bigram[], Jumlah Bigram
2. Lakukan perulangan untuk menginisialisasi isi *header* yang berupa kumpulan bigram, perulangan dilakukan sebanyak jumlah bigram.
3. Kompres dan ubah isi *header* menjadi kode Huffman yang telah dibentuk dengan cara memanggil fungsi cari kata..
4. Tambahkan awalan dan akhiran pada isi *header* berupa "11100"
5. Output *header*



Gambar 3.13 *Flowchart* proses pembuatan *header*

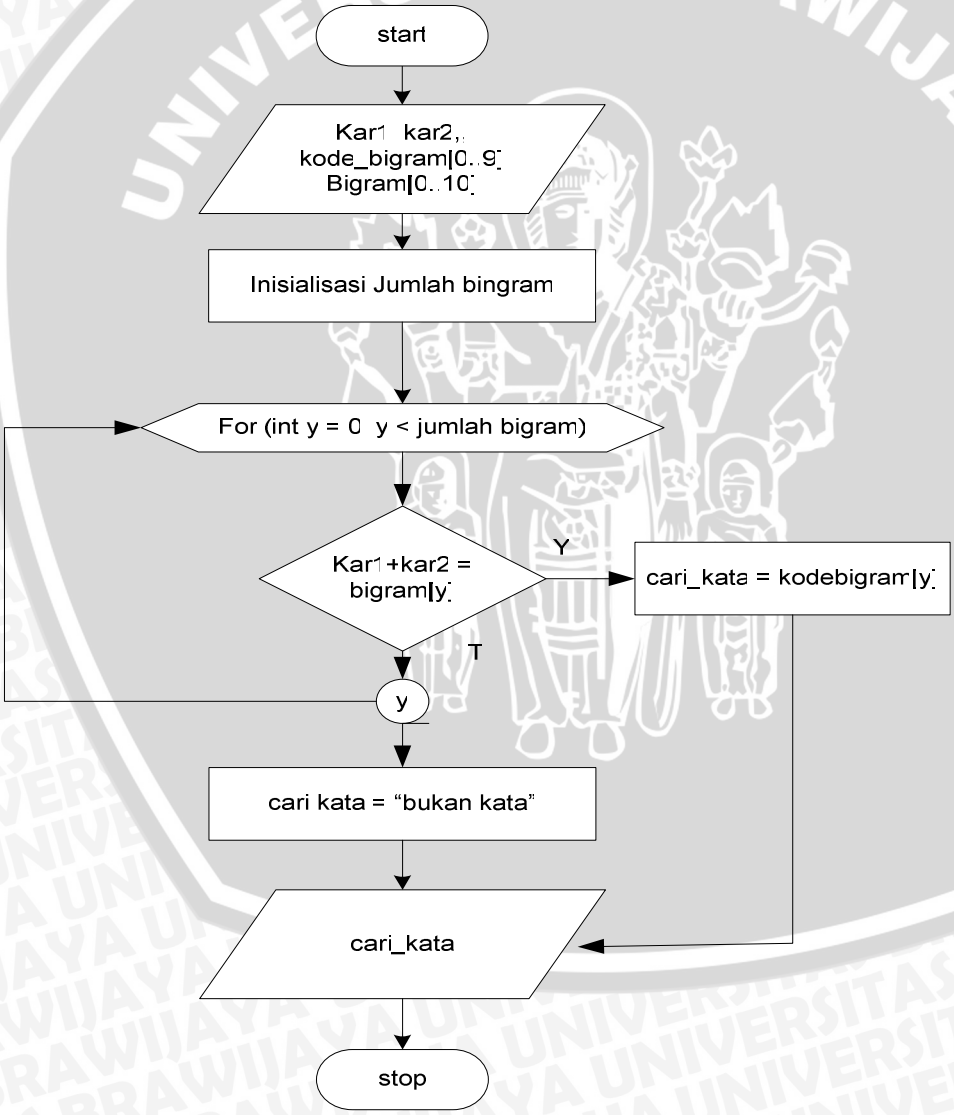
3.3.2.3 Proses Pencarian Kata

Proses pencarian kata pada tahap kompresi digunakan untuk membedakan saat akan mengkompresi karakter, atau saat mengkompresi bigram. Proses ini diproses oleh sebuah fungsi yang mempunyai tipe *string*. Fungsi cari_kata akan memberikan nilai "bukan kata" ketika kumpulan karakter yang dijadikan inputan, tidak terdapat dalam 11 bigram kata yang telah dicari sebelumnnya. Sedangkan jika kumpulan karakter tersebut merupakan salah satu bigram, maka fungsi ini akan memberi nilai *string* berupa kode Huffman dari kata tersebut.

Bigram hanya terdiri dari dua karakter, oleh karena itu fungsi cari_kata mendapatkan input 2 karakter yang saling berurutan. Untuk lebih jelasnya *flowchart* proses pencarian data dapat dilihat pada

Gambar 3.13. Dan langkah-langkah yang dilakukan adalah sebagai berikut:

1. Input karakter pada SMS ke-i, karakter ke-i+1, karakter ke-i+2, *array* kata[0..32], *array* bigram[0..8]
2. Lakukan inisialisasi jumlah bigram
3. Lakukan pengecekan bigram dengan proses looping sebanyak jumlah bigram. Jika kumpulan karakter tersebut terdapat dalam array bigram, maka fungsi akan mengembalikan nilai string berupa kode Huffmanbigram yang tersebut.
4. Jika kata yang dimaksud tidak ada dalam *array* bigram, maka fungsi mengembalikan nilai ”bukan kata”.
5. Output *string* cari_kata.

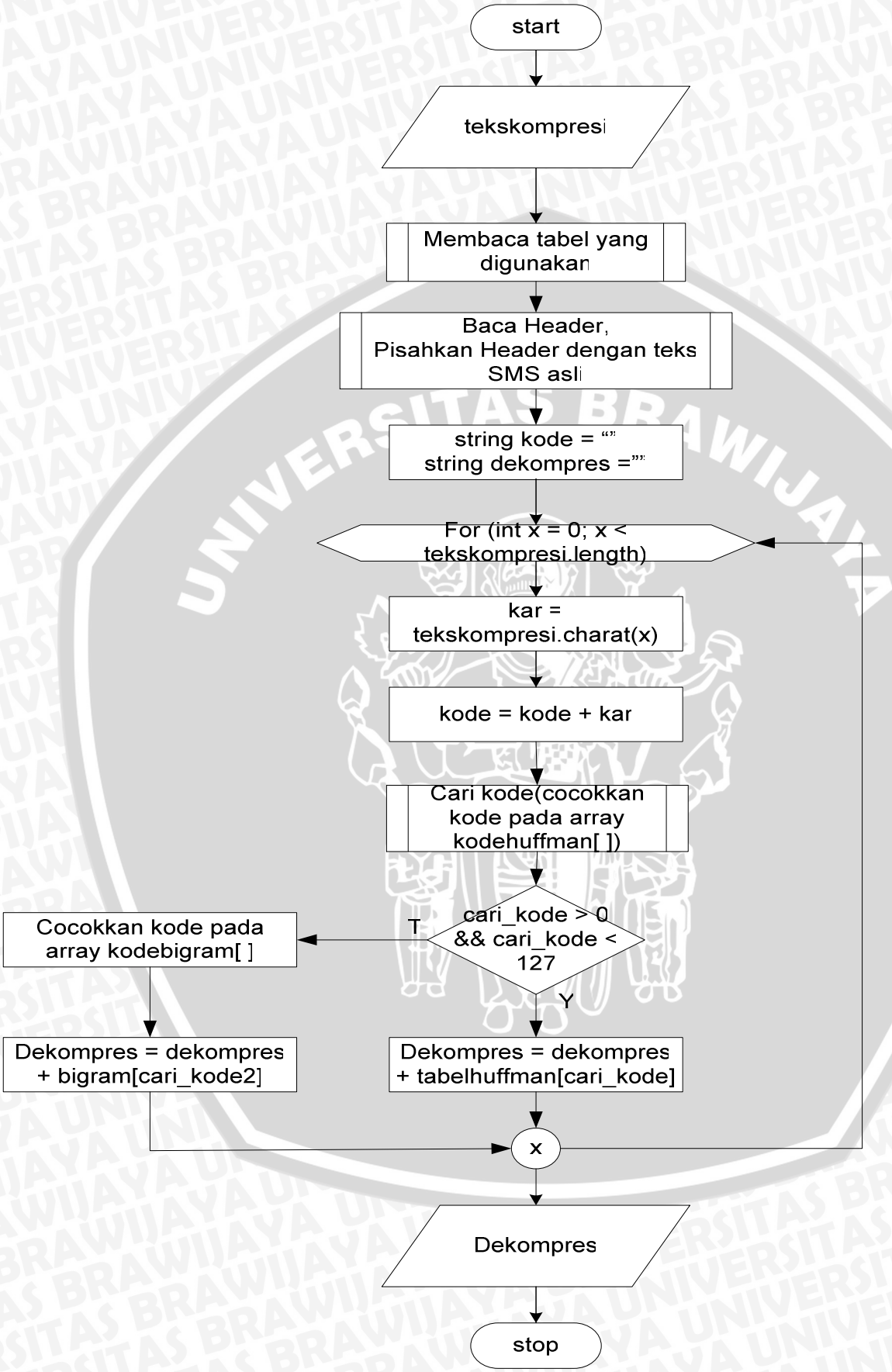


Gambar 3.14 *Flowchart* proses pencarian kata

3.3.3 Perancangan Proses Dekompresi

Proses dekompresi dilakukan ketika SMS kompresi yang telah dikirim sudah diterima. Untuk bisa membaca pesan SMS, proses dekompresi harus dilakukan. Proses dekompresi secara keseluruhan dijelaskan pada Gambar 3.14, dan langkah-langkah yang dilakukan adalah sebagai berikut:

1. Input Teks SMS yang telah dikompresi.
2. Pembacaan tabel yang digunakan pada proses kompresi sebelumnya. Proses pengenalan ini adalah dengan melihat 9 bit pertama dari teks SMS. jika 9 bit pertama adalah ” 101111000”, maka inisialisasi nilai tabel = 2, sedangkan jika tidak maka inisialisasi nilai tabel = 1. Untuk nilai tabel = 2, 9 karakter pertama sebagai bit penanda dihapus, agar tidak mengganggu jalannya proses dekompresi. kemudian *array* yang digunakan adalah array tabel Huffman 2. Untuk nilai tabel = 1; *array* yang digunakan adalah *array* tabel Huffman 1.
3. Membaca Header, menginisialisasi bigram yang dipakai dalam header, kemudian memisahkan header dari teks SMS.
4. Insialisasi *variable* kode dan dekompres.
5. Lakukan perulangan dari karakter SMS pertama sampai karakter SMS terakhir.
6. Inisialisasi *variable* karakter, yang merupakan karakter ke-x (sesuai *looping*) dalam SMS.
7. Inisialisasi *String* kode adalah gabungan dari bit-bit karakter dari *variable* kar. *Variable* ini yang akan dicocokkan dengan kode-kode Huffman yang telah dibentuk sebelumnya.
8. Pemanggilan fungsi cari kode, fungsi ini akan mencocokan *string* kode dengan kode Huffman pada *array* kodehuffman[ ]. Nilai kembalian dari fungsi ini adalah *integer* dari indeks *array* tabel Huffman yang cocok. Indeks bernilai dari 0-127.
9. Jika cari_kode tidak bernilai antara 1-127, maka lakukan pengecekan pada array kodebigram[ ]. Hasil dari pencarian kode yang ditemukan akan mengembalikan nilai berupa bigram kata yang telah diinisialisasi oleh pembacaan *header*.
10. Jika cari_kode bernilai antara 1-127, maka *String* dekompres ditambahkan dengan kata atau karakter yang dimaksud sesuai dengan indeks nilai cari_kode..
11. *Output* teks hasil dekompresi.

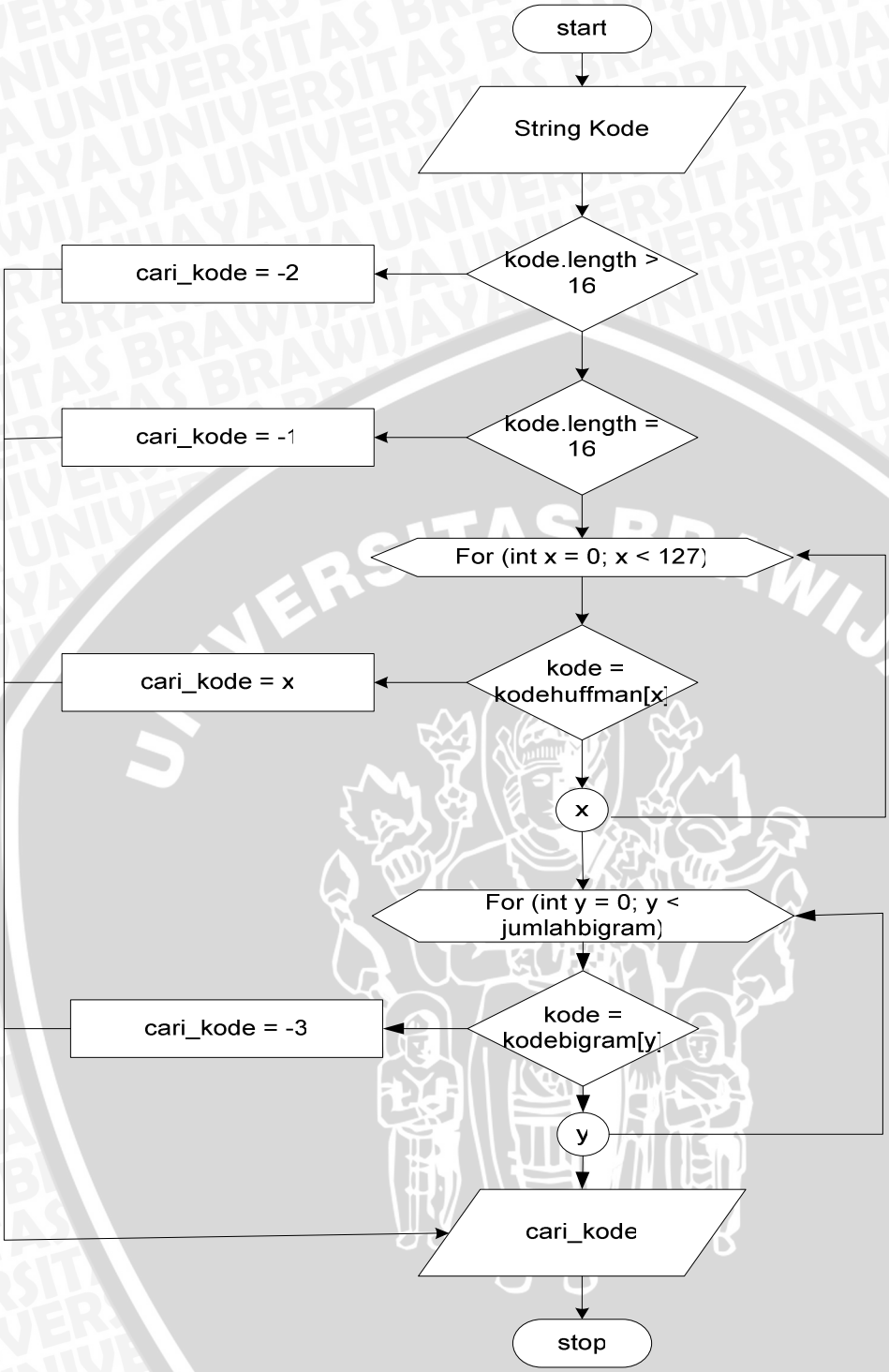


Gambar 3.15 *Flowchart* proses dekompresi

3.3.3.1 Proses Pencarian Kode (Decoding)

Ketika proses dekompresi berlangsung, rangkaian bit akan dicocokan berdasarkan proses *looping* dengan fungsi cari kode. Rangkaian bit karakter atau kata yang cocok akan membuat fungsi kode mengembalikan nilai dari indeks *array* pada tabel huffman yang telah dibentuk. Proses kompresi secara keseluruhan dijelaskan pada Gambar 3.15, dan langkah-langkah yang dilakukan adalah sebagai berikut:

1. Input *string* kode
2. Lakukan pengecekan panjang kode, panjang kode > 16 untuk kondisi pencarian kode yang tidak ketemu maka kembalikan nilai cari_kode = -2.
3. Untuk panjang kode = 16, adalah kondisi untuk melakukan proses dekompresi pada karakter-yang tidak terdapat dalam tabel, nilai cari_kode = -1.
4. Lakukan *looping* sebanyak jumlah kode yang dibentuk sebelumnya, yaitu 143.
5. Jika kode cocok dengan kode huffman yang telah dibentuk maka nilai kode adalah nilai indeks dari *array*, yaitu x.
6. Jika kode tidak cocok lakukan pengecekan apakah kode tersebut terdapat pada array kodebigram[ ], jika terdapat maka kembalikan nilai indeks = -3.
7. Output nilai cari_kode.



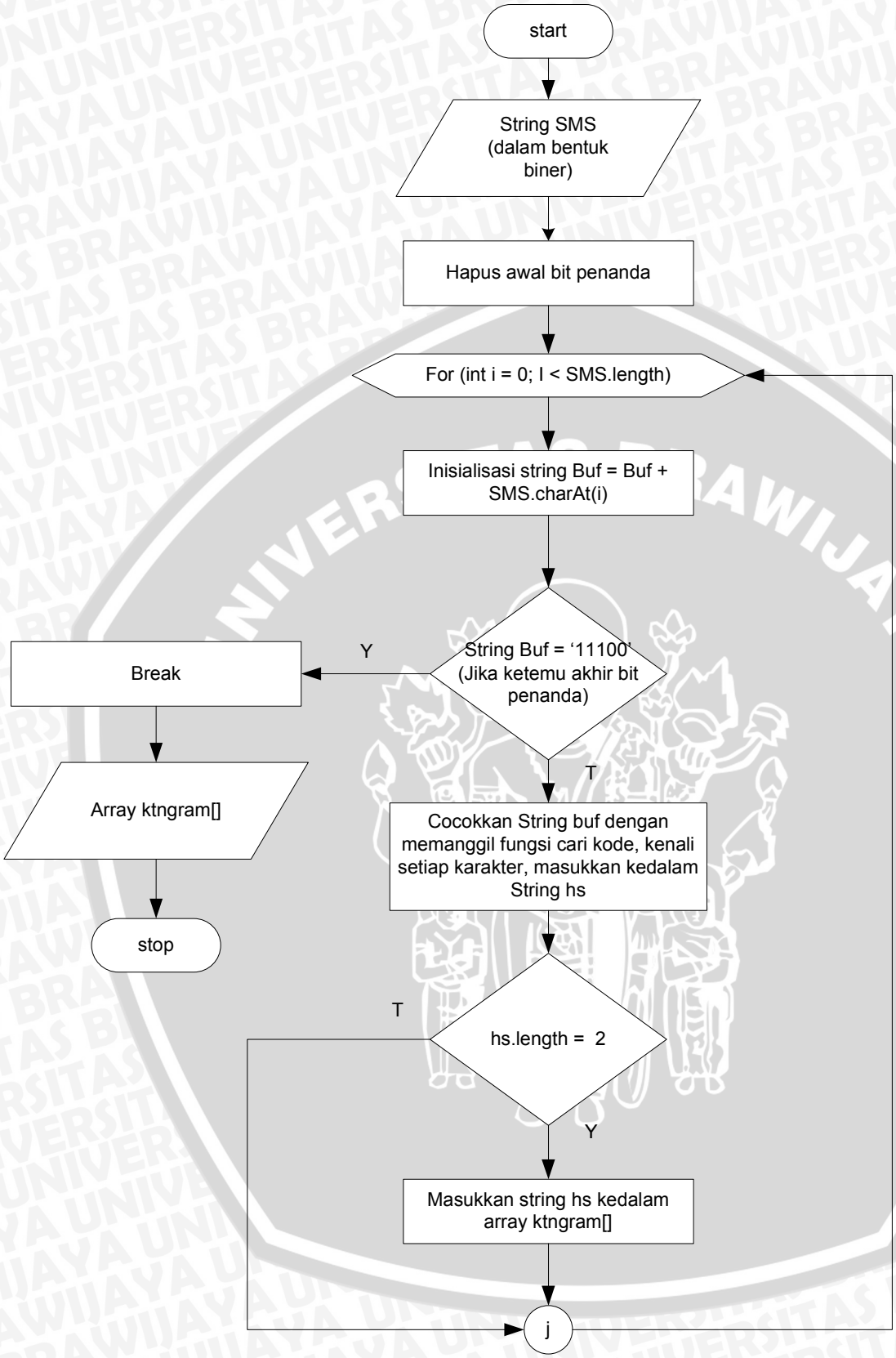
Gambar 3.16 *Flowchart* proses pencarian kode

3.3.3.2 Proses Pembacaan *Header*

Proses pembacaan *header* dilakukan untuk menginisialisasi bigram apa saja yang dibuat sewaktu proses kompresi. Tahapan proses secara keseluruhan dijelaskan pada Gambar 3.16, dan langkah-langkah yang dilakukan adalah sebagai berikut:

1. Input string SMS dalam bentuk biner.

2. Hapus awal bit penanda agar tidak menggangu proses pencarian kode.
3. Lakukan perulangan sebanyak panjang SMS.
4. Inisialisasi *String* buf, nilai buf merupakan kumpulan kode biner yang akan terus ditambahkan sebelum ketemu karakter yang dicari.
5. Jika String buf bernilai '11100' yang merupakan bit penanda, berarti kondisi ini menyatakan akhir dari header, keluar dari proses looping (*Break*).
6. Jika tidak maka lakukan pencarian karakter dengan memanggil fungsi cari kode, setelah karakter dikenali maka masukkan kedalam *String* hs.
7. Lakukan pengecekan apakah panjang *string* hs adalah 2, jika iya maka telah didapatkan bigram. Masukkan bigram tersebut kedalam array ktngram[].
8. Output Array ktngram[].



Gambar 3.17 *Flowchart* pembacaan *header*

3.4 Perancangan Pengujian

Pengujian dilakukan pada 25 teks SMS yang mempunyai jumlah karakter dan variasi pemakaian kata yang berbeda, Dari ke 25 teks SMS dibagi menjadi 5 berdasarkan prosentase pendapatan bigram, yaitu dengan karakteristik sebagai berikut :

1. SMS 1 - 5 dengan prosentase bigram sebanyak 0%.
2. SMS 6 - 10 dengan prosentase bigram sebanyak 1 - 30%.
3. SMS 11 - 15 dengan prosentase bigram sebanyak 31% - 45%%.
4. SMS 16 - 20 dengan prosentase bigram sebanyak 46% - 60%.
5. SMS 21 - 25 dengan prosentase bigram sebanyak 60 - 80%%.

3.4.1 Pengujian Rasio Kompresi

Pengujian yang akan dilakukan pada perangkat lunak ini adalah pengujian rasio SMS sebelum dikompresi dibandingkan dengan teks SMS setelah kompresi. Terdapat dua jenis pengujian, yaitu :

1. Uji coba pertama dilakukan hanya dengan menggunakan metode Huffman, uji coba ini bertujuan untuk membandingkan hasil rasio kompresi sebelum dan sesudah ditambahkan metode bigram.
2. Uji coba kedua dilakukan dengan menggunakan metode Huffman digabungkan dengan bigram.

Tabel yang digunakan untuk pengujian rasio kompresi dengan masing – masing metode ditunjukkan seperti pada Tabel 3.8 dan 3.9

Tabel 3.8 Tabel pengujian rasio kompresi dengan metode Huffman

SMS ke-	Jumlah Karakter	Rasio

Tabel 3.9 Tabel pengujian rasio kompresi dengan metode Huffman dan bigram

SMS ke-	Jumlah Karakter	Bigram yang Berbeda	Total Seluruh Bigram	Prosentase Bigram Pada Teks	Rasio

Penghitungan rasio kompresi menggunakan persamaan 2.1 Sedangkan untuk prosentase bigram pada teks dihitung dengan persamaan berikut :

$$P = (\frac{J * 2}{K}) * 100\% \qquad (3.1)$$

Keterangan :
P : Prosentase bigram pada teks.
K : Jumlah total bigram yang didapatkan
R : Jumlah karakter pada teks.

3.5 Contoh perhitungan

Diketahui terdapat teks SMS yang akan dikirim dengan panjang 53 karakter, seperti ditunjukkan pada Gambar 3.17

hai namaku Wibisana, nanti pagi aku mau pergi ke sana

Gambar 3.18 Contoh teks SMS

Pada tahap selanjutnya sesuai dengan metode kompresi yang telah dijelaskan sebelumnya, dari ke 53 karakter pada contoh akan dicari bigram kata yang mempunyai perulangan. Dari pencarian kata didapatkan 2 bigram yang berulang 4 kali yaitu : "i (spasi)". "na", dan didapatkan 1 bigram yang berulang 3 kali yaitu : "u (spasi)". Pada Gambar 3.18 bigram "i (spasi)" ditunjukkan dengan warna biru, bigram "na" ditunjukkan dengan warna merah, dan bigram "u (spasi)" ditunjukkan dengan warna kuning.

"hai namaku Wibisana, nanti pagi aku mau pergi ke sana"

Gambar 3.19 Contoh Pencarian Bigram

Bila teks pada contoh diubah kedalam bentuk biner sesuai dengan tabel ASCII, maka dihasikan teks seperti ditunjukkan pada Gambar 3.19.

```
1101000110000111010010100000110111011000
1110110111000110101111101010100000101011
1101001110001011010011110011110000111011
0110000101011000100000110111011000011101
1011101001101001010000011100001100001110
1111101001010000011000011101011111010101
0000110110111000011110101010000011100001
0010111100101100111110100101000001101011
1001010100000111001111000011101110100001
```

Gambar 3.20 Bentuk teks biner awal

Setelah dilakukan pencarian kata dan bigram, langkah berikutnya adalah menerjemahkan setiap karakter atau kata yang ada, ke dalam tabel Huffman yang telah dibentuk. Pada contoh kali ini digunakan tabel Huffman 1. Perhitungan kompresi dapat dilihat pada Tabel 3.10.

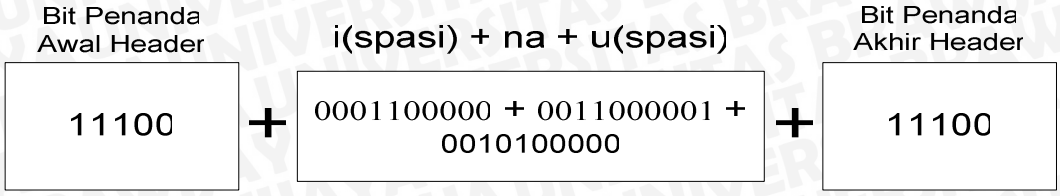
Tabel 3.10 Tabel perhitungan kompresi

Kar / Bigram	Kode ASCII	Kode Huffman	Bit Terkompresi
h	1101000	010011	1
a	1100001	00001	2
i(spasi)	110100101000000	11101	9
na	11011101100001	111100	8
m	1101101	010111	1
a	1100001	00001	2
k	1101011	010101	1
u(spasi)	11101010100000	111101	8
W	1010111	10111011	-1
i	1101001	00011	2
b	1100010	010000	1
i	1101001	00011	2
s	1110011	00111	2
a	1100001	00001	2
na	11011101100001	111100	8

Tabel 3.10 Tabel perhitungan kompresi (lanjutan)

Kar / Bigram	Kode ASCII	Kode Huffman	Bit Terkompresi
,	0101100	110001	1
(spasi)	0100000	00000	2
na	11011101100001	111100	8
n	1101110	00110	2
t	1110100	011010	1
i(spasi)	11010010100000	11101	9
p	1110000	011000	1
a	1100001	00001	2
g	1100111	010010	1
i(spasi)	11010010100000	11101	9
a	1100001	00001	2
k	1101011	010101	1
u(spasi)	11101010100000	111101	8
m	1101101	010111	1
a	1100001	00001	2
u(spasi)	11101010100000	111101	8
p	1110000	011000	1
e	1100101	00101	2
r	1110010	011001	1
g	1100111	010010	1
i(spasi)	11010010100000	11101	9
k	1101011	010101	1
e	110 0101	00101	2
(spasi)	0100000	00000	2
s	1110011	00111	2
a	1100001	00001	2
na	11011101100001	111100	8
Jumlah	**371**	**234**	**137**

Untuk menginisialisasi bigram pada proses dekompresi maka akan dibuat header, header pada contoh digambarkan pada Gambar 3.20



Header = '11100000110000000110000010010100000111000'

Gambar 3.21 Header

Teks terdiri dari 53 karakter dengan panjang masing-masing karakter adalah 7 bit. Panjang teks biner yang dihasilkan sebanyak 371 bit. Setelah dilakukan kompresi panjang biner pada setiap karakter akan berbeda, dari ke-53 karakter, dihasilkan panjang teks biner sebanyak 234 bit ditambah panjang header 40 bit menjadi total 274 bit. Dalam contoh ini didapatkan penghematan sebanyak 97 bit. Hasil teks biner yang sudah dikompresi ditunjukkan pada Gambar 3.21.

```
11100000110000000110000010010100000111000100
11000011110111110001011100001010101111101101
11011000110100000001100111000011111001100010
00001111000011001101011101011000000010100101
11010000101010111110101011100001111101011000
00101011001010010111010101010010100000001110
0001111100
```

Gambar 3.22 Bentuk teks biner akhir

Dari contoh didapatkan data sebagai berikut: terdapat teks sepanjang 53 karakter dengan 3 bigram yang berbeda, dengan total seluruh bigram sebanyak 11, panjang teks biner awal sebanyak 274, dan panjang teks biner akhir sebanyakmaka didapatkan prosentase bigram dan rasio sebanyak :

Prosentase Bigram = ((11 * 2) / 53) * 100%
= 41,5%
Rasio Kompresi = (1- (274 / 371))* 100%
= 26,1%

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam subbab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem kompresi SMS adalah sebagai berikut :

- Handphone
  1. *Support* Java.
  2. *Support Mobile Information Device Profile* 2.0 (MIDP 2.0)
  3. 64 Mb *Phone Memory*.
- Personal Computer
  1. Prosesor Intel P4 2,8 Ghz
  2. Memori 256 Mb
  3. Harddisk dengan kapasitas 80 GB
  4. Monitor 15”
  5. Keyboard
  6. Mouse

4.1.2 Lingkungan perangkat lunak

Perangkat lunak yang digunakan dalam pengembangan sistem kompresi SMS adalah sebagai berikut :

1. Sistem Operasi Windows XP.
2. Jcreator Pro.
3. JDK Version 6.0.
4. Java Wireless Toolkit 2.5.2.
5. Microsoft Excel.

4.2 Implementasi Program

Berdasarkan analisa dan perancangan proses yang terdapat pada subbab 3.3, maka pada subbab ini akan dijelaskan implementasi proses-proses tersebut.

4.2.1 Implementasi Kompresi

Sesuai Pada subbab 3.3 urutan proses kompresi adalah sebagai berikut: pencarian bigram, mengurutkan bigram sesuai dengan frekuensi terbanyak (*sorting*), pembuatan *header*, menerjemahkan setiap karakter dan bigram kedalam kode Huffman, dan langkah terakhir yaitu merubah *string* biner ke dalam bentuk *byte*.

4.2.1.1 Pencarian Bigram

Tahap awal dari proses kompresi adalah memanggil prosedur caribigram. Terdapat dua tahap pada prosedur pencarian bigram, yang pertama yaitu pencarian bigram, dan kedua adalah menyeleksi bigram yang mempunyai indeks sama. Prosedur pencarian bigram tahap pertama ditunjukan pada gambar 4.1.

```
public static void caribigram(String kata)
{
String kumpngram[] = new String[500];
String ngram, cocokkata;
int index[] = new int[600]; int frekuensi[] = new int[600];
int tempindex[][] = new int[600][600];
int frek=0,xindex=0,yindex=0,hitindex=0,indexpakai=0;
jumngram=0; fix=0;
for(int i = 0; i < kata.length()-1; i++)
{
    ngram= kata.substring(i,i+2);     //bigram 1
    for(int j = 0; j < kata.length()-1; j++)
    {
        cocokkata=kata.substring(j,j+2);      //bigram 2
        if(ngram.equals(cocokkata))   //dibandingkan
        {
                frek = frek +1;
                tempindex[jumngram][yindex]=j;  //catat index
                tempindex[jumngram][yindex+1]=j+1;
                yindex=yindex+2;
        }
    }
    yindex=0;
    if(frek>2)         //Seleksi bigram yang kurang dari 2
    {
        kumpngram[jumngram]=ngram;
        frekuensi[jumngram]=frek;
        jumngram=jumngram+1;
    } frek=0;
}
```

Gambar 4.1 Prosedur pencarian bigram

Pencarian bigram dilakukan dengan membandingkan dua *string* satu persatu, jika didapatkan nilai antara kedua *string* sama, maka frekuensi ditambah dan indeksnya dicatat dengan memasukkannya kedalam *array* tempindex.

Indeks pada bigram adalah letak karakter pertama dan karakter terakhir bigram tersebut pada teks SMS. Jika terdapat bigram yang mempunyai indeks sama, maka bigram yang dipilih adalah bigram yang mempunyai frekuensi lebih besar melalui proses *sorting*. Oleh karena itu pada tahap kedua prosedur caribigram akan menyeleksi bigram-bigram yang mempunyai indeks yang sama, Prosedur ini ditunjukan pada Gambar 4.2.

```
//cari indeks sama
for(int l = 1; l < jumngram; l++)
{
        for(int k = 0; k < frekuensi[l]*2; k++)
    {
        for(int m = 0; m < indexpakai; m++)
        {
if((tempindex[l][k]>=index[m])&&(tempindex[l][k]<=index[m+1]))
                {
                        m++;
                }
                else
                {
                        m++;
                        hitindex=hitindex+1;
                }
        }
    }

        if(hitindex==((frekuensi[l]*indexpakai)))
        //menyamakan nilai
        {
                fixngram[fix]=kumpngram[l];
                fixfrek[fix]=frekuensi[l];
                fix++;
                for(int n = 0; n < (frekuensi[l]*2)+1; n++)
        {
                index[indexpakai+n]=tempindex[l][n];  //indeks
dari bigram yang dipakai disimpan kedalam array indeks
        }
        indexpakai=(indexpakai+(frekuensi[l]*2));
        //nilai dari indeks pakai bertambah
        }
        hitindex=0;
}
}
```

Gambar 4.2 Prosedur pencararian bigram tahap seleksi indeks

Seleksi indeks yang sama pada bigram dilakukan dengan cara membandingkan indeks dari setiap bigram dengan indeks dari seluruh bigram yang telah terpakai. Indeks dari seluruh bigram yang telah terpakai dimasukkan didalam *array* indeks. Jika nilai indeks pada bigram tidak terdapat pada *array* indeks, maka nilai variabel hitindex bertambah. Nilai hitindex nantinya akan dibandingkan dengan nilai frekuensi dikalikan jumlah indeks yang telah tepakai. Jika nilai tersebut sama berarti indeks bigram tersebut belum terpakai dan bigram dimasukkan kedalam *array* fixngram. *Array* fixngram inilah yang merupakan *output* dari prosedur cariBigram.

4.2.1.2 *Sorting* (Pengurutan)

Sorting digunakan agar dapat mencari bigram yang mempunyai frekuensi paling banyak yang nantinya akan diberi kode Huffman yang lebih pendek. Prosedur untuk melakukan proses *sorting* ditunjukkan pada Gambar 4.3.

```
public static void sorting(String a[], int b[], int jum, int
index[][] )
{
String temp="";
int temp2 =0;
int temp3=0;
for(int i=0;i<jum-1;i++)
{
        for(int j=0;j<jum-i;j++)
        {
        if(b[j] < b[j+1])
        {
                temp=a[j];      //tukar bigram
                a[j]=a[j+1];
                a[j+1]=temp;
                temp2=b[j];     //tukar frekuensi
                b[j]=b[j+1];
                b[j+1]=temp2;

                for(int k=0;k<(b[j+1]*4);k++)
                {
                        temp3=index[j][k];      //tukar indeks
                        index[j][k]=index[j+1][k];
                        index[j+1][k]=temp3;
                }
        }
        }
}
}
```

Gambar 4.3 Prosedur *Sorting*

4.2.1.3 Pembuatan *Header*

Setelah didapatkan bigram pada proses sebelumnya, tahap berikutnya adalah membuat *header* yang berisi bigram-bigram tersebut. *Header* ditempatkan diawal teks SMS. Prosedur untuk melakukan pembuatan *header* ditunjukkan pada Gambar 4.4.

```
public static void header()
{
    stheader ="11100";//      bit penanda
    String temp="";

    if (fix >= 9)      //      inisialisasi jumlah bigram
    {
        fix = 9;
    }else
    {
        fix = fix;
    }

    for(int i = 0; i < fix; i++)
    {

        temp=temp+fixngram[i]; // kumpulan bigram
    }

    for(int j = 0; j < temp.length(); j++)//ubah ke Huffman
    {
        char kar = temp.charAt(j);
       if(kar >= '\200') //>=200 diluar kode ASCII
       stheader = stheader+ kompresSisanya(kar);
       else if(cc[kar] != null)
       stheader = stheader+ cc[kar];
       else
       stheader = stheader + kompresSisanya(kar);
    }

    stheader=stheader+"11100";

}
```

Gambar 4.4 Prosedur Pembuatan *Header*

Header diawali dan diakhiri dengan bit penanda yaitu “11100”, dan ditengah-tengah *header* berisi bigram-bigram yang telah didapatkan sebelumnnya. Seluruh bigram digabungkan kedalam satu variabel yaitu *string* temp. Proses berikutnya adalah menerjemahkan kumpulan bigram menjadi kode Huffman. Output prosedur adalah *string* stheader.

4.2.1.4 Proses Kompresi Kode Huffman (Encoding)

Tahap ini merupakan dasar dari seluruh proses kompresi. Seluruh proses yang telah dijelaskan sebelumnya dijalankan dalam fungsi kompres. dalam proses ini masing – masing karakter dan bigram akan diubah menjadi kode Huffman yang telah dibentuk sebelumnya. Fungsi untuk melakukan proses *encoding* ditunjukkan pada Gambar 4.5.

```
public static byte[] kompres(String st)
{
  stheader="";
  jadi="";
  String crkata;
  int pjgsms=(st.length()-1);
  caribigram(st);      //
  header();

  for(int i = 0; i < st.length(); i++)
  {     crkata="bukan kata";
        char kar = st.charAt(i);
        if(i!=pjgsms)
        {
               char kar2 =st.charAt(i+1);
               char kar3 ='k';
               crkata = carikata(kar, kar2, kar3;
               //cari kata bwt ngembaliin nilai kode bigram

        }
        if(crkata.equals("bukan kata"))//jika bukan bigram
        {
               if(kar >= '\128')
               jadi = jadi + kompresSisanya(kar);
               else
               if(cc[kar] != null)
               jadi = jadi + cc[kar];
               else
               jadi = jadi + kompresSisanya(kar);

        }
        else//jika bigram
        {
               i++;
               jadi = jadi + crkata;
        }
  }
 jadi=stheader+jadi;
 return stringKeByteA(jadi;
}
```

Gambar 4.5 Prosedur Proses Kompresi

Fungsi kompres mendapatkan inputan berupa teks SMS, kemudian akan memanggil prosedur caribigram( ) dan header( ). *String* crkata adalah *string* yang akan membedakan sewaktu akan mengkompresi karakter atau bigram. Nilai *string* crkata didapatkan dari kembalian nilai fungsi carikata( ).

Ketika kembalian nilai *string* crkata adalah "bukan kata", maka akan dilakukan proses kompresi karakter. Proses ini adalah dengan melihat nilai indeks kode ASCII dari karakter yang akan dikompresi, jika nilai indeks lebih dari 128, maka karakter tersebut bukan merupakan kode ASCII 7 bit sehingga dilakukan pembuatan kode khusus, dengan cara memanggil prosedur kompresSisannya( ). Jika nilai indeks karakter tersebut terdapat di *array* kode huffman, maka akan langsung diberikan nilai kode Huffman karakter tersebut.

4.2.1.5 Pencarian Kata

Fungsi pencarian kata dipanggil saat proses kompresi, fungsi ini yang membedakan ketika akan mengkompresi karakter atau bigram. Fungsi mendapat masukkan dua karakter, yang mana dari kedua karakter ini akan dicocokkan dengan bigram-bigram yang telah dicari sebelumnnya. Jika dua karakter tersebut adalah bigram maka hasil kembalian dari fungsi ini adalah kode bigram kedua karakter tersebut. Fungsi carikata ditunjukkan pada Gambar 4.6.

```
static String carikata(char kar1, char kar2)
{
        String carikata = "";
        carikata = carikata.valueOf(kar1);
        carikata = carikata+carikata.valueOf(kar2);

        String hasil ="bukan kata";


        for (int i = 0; i< fix; i++)//meriksa bigram
        {
                if(carikata.equals(fixngram[i]))
                {
                        hasil =kdngram[i];
                        break;
                }
        }
        return hasil;
}
```

Gambar 4.6 Fungsi Pencarian Kata

4.2.2 Implementasi Dekompresi

Pada subbab 3.4 inputan dari proses dekompresi adalah teks SMS dalam bentuk biner yang telah terkompresi, tujuan proses ini dilakukan adalah agar teks SMS dapat terbaca oleh *user*. Urutan proses pada tahap ini adalah sebagai berikut: pembacaan tabel Huffman, pembacaan *header*, inisialisasi bigram, dan langkah terakhir menerjemahkan kumpulan bit biner menjadi karakter. Fungsi untuk melakukan proses dekompresi ditunjukkan pada Gambar 4.7.

```
public static String dekompres(byte ba[])
{

init();
String st = byteAKeString(ba);
bacaheader(st);
String hs = "";
String buf = "";
for(int i = 0; i < teks.length(); i++)
{
        char kar = teks.charAt(i);
        buf = buf + kar;
        int c = cariKode(buf);
        if(c >= 0) //dekompresi untuk karakter
         {
                buf = "";
                hs = hs + (char)c;
         } else
         if(c == -2) //pencarian karakter tidak ketemu
         {
                 buf = "";      //nilai buf dikosongkan
         }else
         if (c == -3) //dekompresi untuk bigram
         {
                for(int j = 0; j < jngramdek; j++)
                {
            if(kdngram[j] != null && kdngram[j].equals(buf))
                {
                        buf = "";
                        hs = hs + ktngram[j];
                }
                }
          }
}

return hs;
}
```

Gambar 4.7 Fungsi Dekompres

Setelah menerima inputan berupa teks SMS dalam bentuk biner yang bertipe *byte*, fungsi dekompres akan memanggil fungsi byteAkeString() untuk merubah tipe *byte* menjadi *string*. Kemudian memanggil prosedur bacaheader() untuk menginisialisasi bigram.

Proses dekompresi dilakukan dengan menginisialisasi *string* buf yang berupa kumpulan kode biner. *String* buf ini nantinya akan dijadikan sebagai inputan ketika memanggil fungsi carikode(). Jika kembalian dari fungsi carikode() lebih besar dari 0, maka langsung dikembalikan ke nilai indeks dari karakter pada kode ASCII. Jika kembalian fungsi -3, maka proses yang dilakukan adalah melakukan proses dekompresi secara bigram.

4.2.2.1 Pencarian Kode

Fungsi carikode akan menentukan proses *decoding* dari kode Huffman. Proses pencarian kode ditunjukkan pada Gambar 4.8.

```
static int cariKode(String buf)//cari kode
{
        if(buf.length() > 23)
                return -2;
        if(buf.length() ==23&&
        buf.substring(0,7).equals(ccSISANYA))
          return Integer.parseInt(buf.substring(7), 2);

for(int i = 0; i < 128; i++)
{
        if(cc[i] != null && cc[i].equals(buf))
                return i;
}

for(int i = 0; i < jngramdek; i++)
{
        // memeriksa apakah string buf sama dengan kdngram
       if(kdngram[i] != null && kdngram[i].equals(buf))
                return -3;
}
         return -1;
}
```

Gambar 4.8 Fungsi Pencarian Kode

Ada beberapa nilai kembalian dari fungsi ini, nilai -2 untuk kode Huffman yang sudah melebihi panjang batas, karena dalam tabel Huffman yang telah dibuat sebelumnnya tidak terdapat kode yang panjangnnya melebihi dari 23. Kode biner akan dicocokkan satu-persatu, bila kode biner terdapat pada array cc[], maka langsung

mengembalikan nilai indeks kode ASCII karakter terssebut. Nilai kembalian pada proses ini yaitu lebih besar dari 0.

Kode biner juga dicocokkan pada *array* bigram yang telah diinisialisasi pada saat proses pembacaan *header*, jika cocok maka nilai kembalian = -3. Untuk nilai kembalian -1 berarti kode biner tidak ada yang cocok baik pada *array* karakter maupun pada *array* bigram. Pada saat proses dekompresi nanti nilai kembalian -1 tidak akan diproses.

4.2.2.2 Pembacaan *Header*

Prosedur untuk melakukan proses pembacaan *header* ditunjukkan pada Gambar 4.9.

```
public static void bacaheader(String st)
{
teks.delete(0,teks.length());
teks.append(st); jngramdek=0;
if(st.substring(0,5).equals("11100"))
{
        teks.delete(0,5);//delete bit penanda
        for(int i = 0; i <teks.length() ; i++)
        {
                char kar = teks.charAt(i);
                buf = buf + kar;
                int c = cariKode(buf);
                if(buf.equals("11100"))  //akhir bit penanda
                {
                        teks.delete(0,i+1);
                        break ;
                }
                if(c >= 0)
                {
                        buf = "";
                        hs = hs + (char)c;
                } else
                if(c == -2)
                {
                        buf = "";
                }
                if(hs.length()==2) //ditemukan bigram
                {
                        ktngram[jngramdek]=hs;
                        jngramdek=jngramdek+1;
                        hs="";
                }
        }
}
}
```

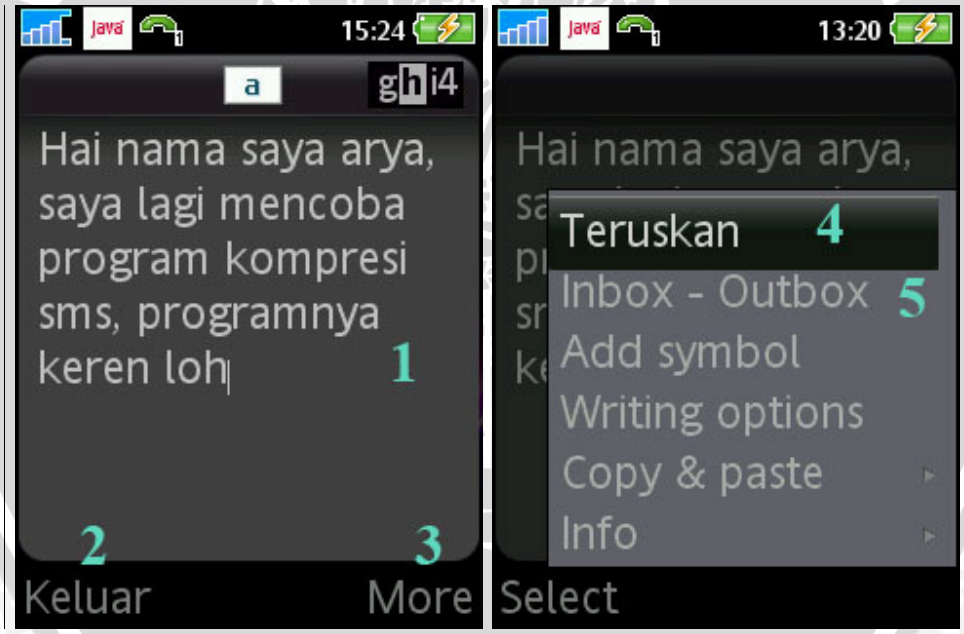
Gambar 4.9 Fungsi Pembacaan *Header*

Proses ini diawali dengan mengenali 5 bit awal dari SMS, jika merupakan bit penanda maka proses pembacaan *header* dijalankan. Tahap-tahap proses pembacaan *header* menyerupai proses dekompresi, hanya dalam pembacaan *header* proses akan berhenti jika ketemu bit penanda akhir.

Setiap kali prosedur mengenali dua karakter, maka karakter tersebut dikenali sebagai satu bigram. Bigram tersebut kemudian disimpan didalam *array* ktngram[]. *Output* dari prosedur ini adalah *array* ktngram[], dan jumlah bigram yang disimpan dalam variabel jngramdek.

4.3 Implementasi Antarmuka

Implementasi antarmuka dibuat menyerupai aplikasi SMS yang terdapat secara umum. Untuk tampilan pada uji coba kali ini menggunakan *handphone* Sony Ericcsson K800i. Tampilan *form* utama dapat dilihat pada Gambar 4.10.

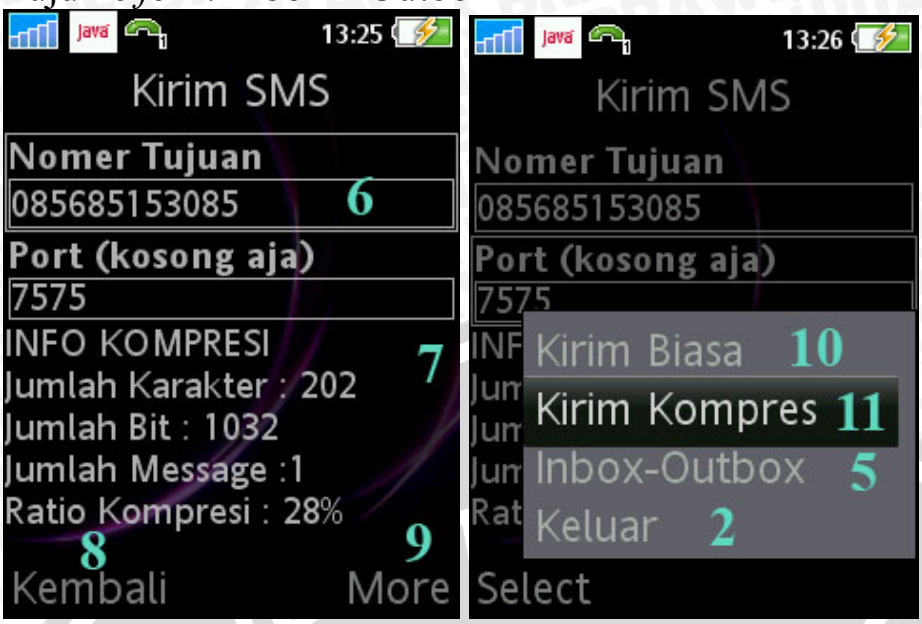


Gambar 4.10 Gambar *form* utama (Textbox SMS)

Keterangan :

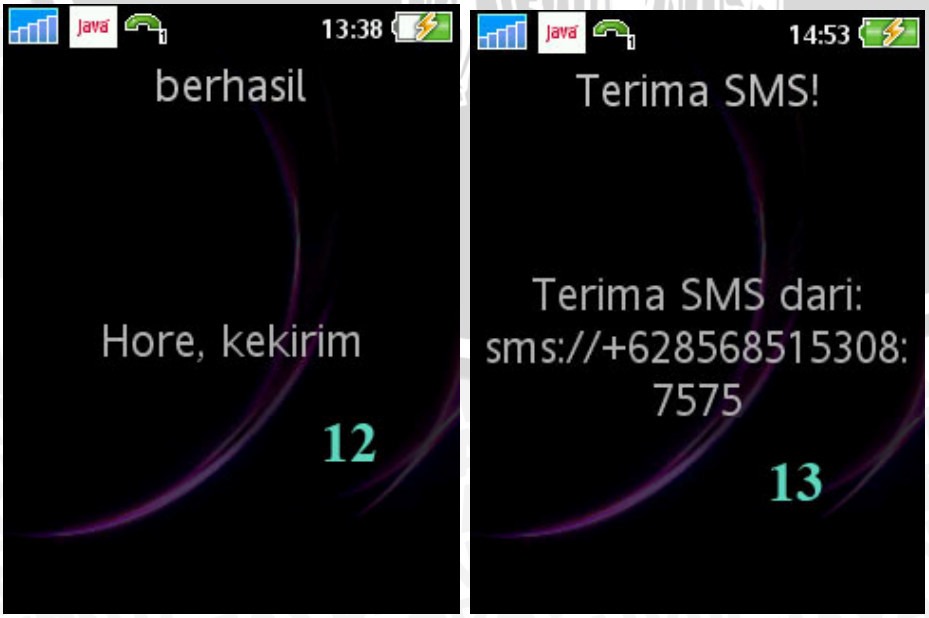
1. *Form* utama → digunakan untuk menulis teks SMS.
2. Keluar dari aplikasi
3. *Command* "menu" pada form utama
4. *Command* "Teruskan" → untuk meneruskan proses kompresi setelah menulis pesan SMS, *command* akan mengarah ke dalam *form* info..

5. Menuju ke *form* Inbox – Outbox



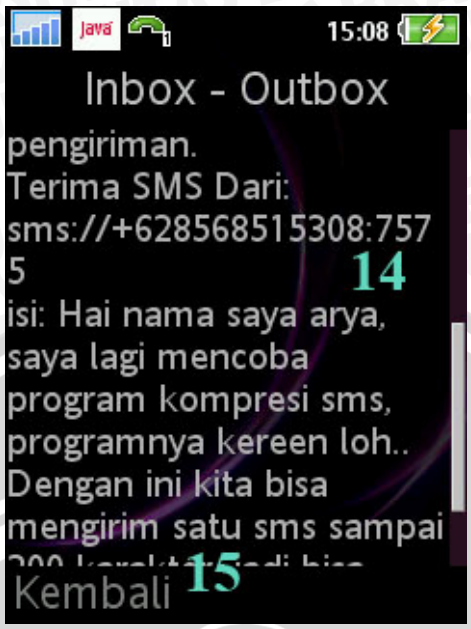
Gambar 4.11 Gambar form info kompresi

6. *Textfield* nomor tujuan yang akan dikirimi SMS.
7. *Form* info kompresi, berisi tentang info kompresi SMS mengenai jumlah karakter, jumlah bit, jumlah SMS, dan rasio kompresi.
8. Menuju ke *form* utama
9. *Command* "menu" pada *form* info kompresi.
10. *Command* "Kirim Biasa" → digunakan untuk mengirimkan SMS secara normal tanpa dilakukan kompresi.
11. *Command* "Kirim Kompres" → digunakan untuk mengirimkan SMS dengan kompresi.



Gambar 4.12 Gambar mengirim dan menerima SMS

12. *Message alert* pemberitahuan SMS telah berhasil terkirim.
13. *Message alert* pemberitahuan baru menerima SMS.



Gambar 4.13 Gambar form Inbox - Outbox

14. *Form* Inbox – Outbox, SMS yang dikirim dan diterima akan disimpan didalam *form* ini.
15. *Command* kembali ke *form* utama.

4.4 Implementasi Sistem

4.4.1 Hasil Uji Rasio

Pengujian rasio kompresi SMS dilakukan dengan 2 tahap pengujian. Pengujian pertama hanya menggunakan metode kompresi Huffman, dan pengujian kedua menggunakan Huffman dan bigram.

Pengujian dilakukan pada 25 teks SMS yang mempunyai 5 jenis prosentase bigram yang berbeda. Bagian pertama adalah data uji yang mempunyai prosentase bigram sebanyak 0%, bagian kedua sebanyak 1%-30%, bagian ketiga sebanyak 31%-45%, bagian keempat sebanyak 46%-60%, dan bagian kelima dengan prosentase bigram sebanyak 60%-80%. Jumlah karakter paling sedikit dari ke 25 teks uji adalah 21 karakter, dan yang paling tinggi adalah 430 karakter.

Hasil uji rasio pada pengujian pertama ditunjukkan pada Tabel 4.1, sedangkan uji rasio pada pengujian kedua ditunjukkan pada Tabel 4.2, Data mengenai teks SMS uji dapat dilihat pada halaman lampiran.

Tabel 4.1 Data kompresi SMS menggunakan Huffman

SMS ke-	Jumlah Karakter	Rasio
1	32	22.32%
2	37	23.38%
3	131	22.03%
4	23	22.36%
5	54	22.75%
6	70	21.53%
7	39	19.78%
8	152	22.09%
9	140	20.51%
10	140	22.45%
11	169	22.91%
12	430	21.99%
13	222	21.75%
14	64	24.33%
15	225	23.05%
16	195	21.32%
17	132	23.38%
18	186	23.12%
19	277	23.34%
20	21	21.77%
21	50	22.00%
22	46	22.05%
23	23	22.98%
24	23	22.36%
25	84	23.64%

Pada Tabel 4.1 data rasio kompresi paling tinggi didapatkan pada teks SMS ke-14 dengan jumlah karakter sebanyak 64 dan rasio kompresi sebesar 24.33%. Sedangkan rasio kompresi paling kecil didapatkan SMS ke-7 dengan jumlah karakter sebanyak 39 dan rasio kompresi sebesar 19.78%.

Tabel 4.2 Data Kompresi SMS Menggunakan Huffman + Bigram

SMS ke-	Jumlah Karakter	Bigram yang Berbeda	Total Seluruh Bigram	Prosentase Bigram Pada Teks	Rasio
1	32	0	0	0.00%	22.32%
2	37	0	0	0.00%	23.38%
3	131	0	0	0.00%	22.03%
4	23	0	0	0.00%	22.36%
5	54	0	0	0.00%	22.75%
6	70	1	4	11.27%	21.53%
7	39	1	3	15.38%	18.68%
8	152	2	12	15.79%	24.34%
9	140	3	13	18.57%	22.65%
10	140	4	17	24.29%	25.20%
11	169	5	27	31.95%	27.73%
12	430	11	80	37.21%	28.50%
13	222	11	44	39.64%	23.29%
14	64	3	14	43.75%	26.15%
15	225	10	53	47.11%	26.79%
16	195	9	47	48.21%	27.25%
17	132	9	33	50.00%	25.22%
18	186	10	50	53.76%	28.19%
19	277	10	77	55.60%	33.06%
20	21	2	6	57.14%	21.09%
21	50	3	16	64.00%	35.14%
22	46	3	15	65.22%	34.47%
23	23	1	9	78.26%	34.16%
24	23	1	9	78.26%	41.14%
25	84	5	50	78.57%	38.78%

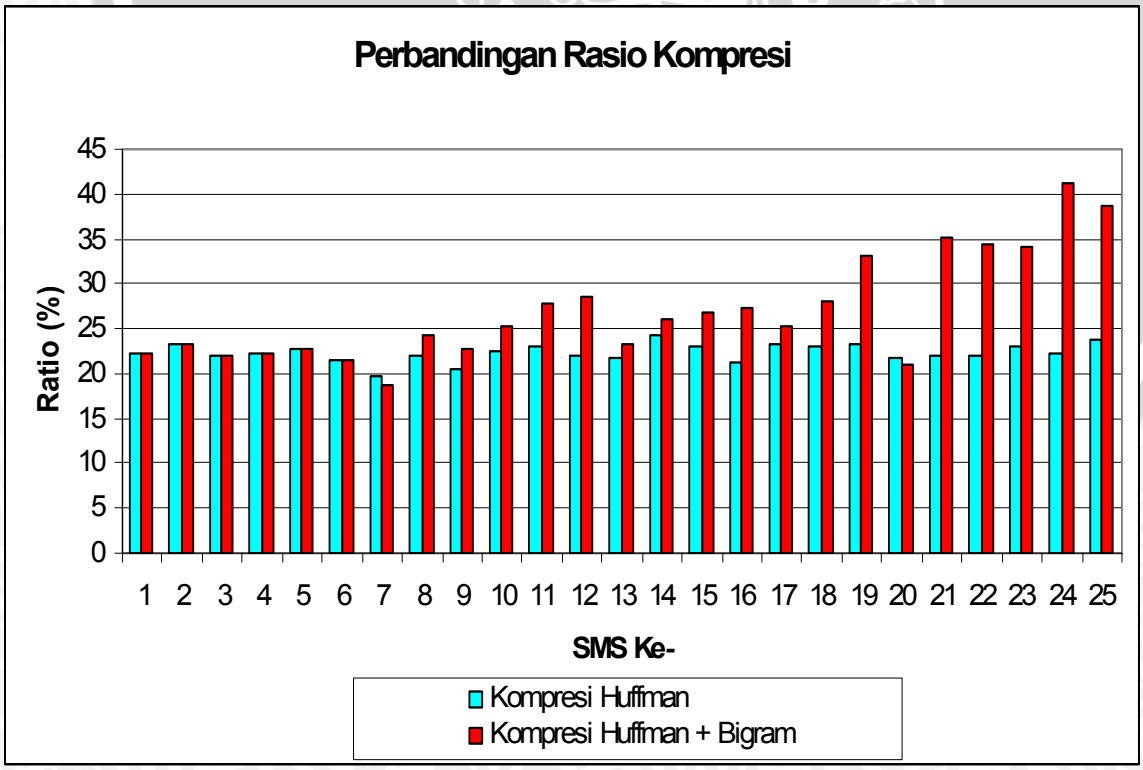
Pada Tabel 4.2 uji coba dilakukan dengan metode Huffman dan perulangan bigram. Pada Tabel 4.2 dari ke 25 teks uji dibagi menjadi 5 bagian. Teks uji 1 - 5 mempunyai prosentase bigram sebesar 0%,

teks uji 6 - 10 mempunyai prosentase bigram antara 0% - 30%, teks uji 11 - 15 dengan prosentase bigram 31% - 45%, teks uji 16 - 20 dengan prosentase bigram 46% - 60% dan teks uji 21 - 25 dengan prosentase bigram 60% - 80%. Hasil rasio kompresi berdasarkan prosentase pendapatan bigram dapat dilihat pada Tabel 4.3.

Tabel 4.3 Rasio Kompresi Huffman + Bigram Berdasarkan Prosentase Bigram

SMS ke-	Prosentase Bigram Pada Teks	Rasio Rata-rata
1-5	0%	22.57%
6-10	1%-30%	22.48%
11-15	31%-45%	26.49%
16-20	46%-60%	26.96%
21-25	60%-80%	36.74%

Data rasio kompresi paling tinggi didapatkan pada teks SMS ke-24 dengan jumlah karakter sebanyak 23 dan rasio kompresi sebesar 41,14%. Sedangkan rasio kompresi paling kecil didapatkan SMS ke-7 dengan jumlah karakter sebanyak 39 dan rasio kompresi sebesar 18.68%. Perbandingan hasil kompresi antara kedua metode kompresi pada Tabel 4.1 dan Tabel 4.2 dapat dilihat pada Gambar 4.5.



Gambar 4.14 Grafik perbandingan rasio kompresi

4.4.2 Pembahasan dan Analisa Hasil

Hasil evaluasi penghitungan rasio kompresi menunjukkan nilai rasio yang bervariasi pada tiap data uji. Pada metode pengujian pertama yang hanya menggunakan metode Huffman, nilai rasio kompresi cenderung tidak banyak berubah antara masing – masing teks SMS. Rentang nilai rasio hanya antara 19,78% - 24,33%, hal itu disebabkan karena metode yang dipakai adalah huffman dengan tabel yang tetap, sehingga jumlah karaker tidak berpengaruh pada besarnya rasio. Dalam metode ini hal yang berpengaruh adalah pemakaian karakter pada teks, jika karakter yang dipakai merupakan karakter-karakter yang mempunyai kode Huffman yang pendek, maka nilai rasio semakin besar, tetapi jika karakter-karakter yang dipakai mempunyai kode Huffman yang panjang maka nilai rasio semakin kecil. contoh dapat dilihat pada teks SMS uji ke-14. Rasio rata-rata metode 1 sebesar 22,37%.

Pengujian kedua dilakukan dengan metode Huffman dan bigram. Rata - rata nilai rasio kompresi untuk metode ini yaitu sebesar 27,05%. Rentang nilai rasio dari ke 25 teks uji cukup tinggi yaitu antara 18.68% - 41.14%, hasil rasio yang bervariasi ini dipengaruhi oleh beberapa hal yaitu jumlah karakter pada teks, pemakaian karakter dalam teks, jumlah bigram yang didapat, frekuensi dari setiap bigram yang didapat, maupun dari prosentase pendapatan bigram pada teks.

Berdasarkan Gambar 4.14, teks uji ke 1 – 5 mempunyai rasio kompresi yang sama pada kedua metode pengujian, hal ini disebabkan karena tidak ada bigram yang didapatkan dalam teks atau dengan kata lain prosentase bigramnya 0%, dalam kasus ini *header* otomatis tidak ditulis. Rasio rata-rata untuk teks dengan prosentase bigram 0% yaitu sebesar 22,57%.

Untuk teks uji ke 6 - 10 dengan prosentase bigram sebesar 1% - 30% menghasilkan rasio rata - rata sebesar 22,48%. Rasio rata-rata kompresi teks uji ke 6-10 lebih kecil dari rasio kompresi rata-rata teks uji ke 1 - 5 (lihat Tabel 4.3), hal ini disebabkan karena *header* yang dibuat lebih panjang daripada hasil kompresi dari bigram. Jika bigram yang diperoleh mempunyai frekuensi perulangan kecil dan variasi jumlah bigram yang berbeda banyak, maka nilai kompresi pada bigram juga akan kecil dan akan tidak sebanding dengan panjangnya *header*. Karena semakin banyak jumlah bigram yang berbeda maka semakin panjang *header* dan semakin besar *size* dari

file. Kasus seperti ini terjadi pada teks uji ke-7. Rasio kompresi dengan metode Huffman dan bigram (pengujian II) lebih kecil dari metode pengujian ke I yang hanya memakai metode Huffman. Perbandingan perhitungan kompresi teks uji ke-7 pada kedua pengujian dapat dilihat pada Gambar 4.15.

"rapat jam 11, harap datang tepat waktu!"

```
jumlah karakter = 39
jumlah bigram yang berbeda = 1
bigram 1 = "at"
frekuensi perulangan = 3
Kode Huffman untuk "at" = 00010011010 (11 bit)
Kode Huffman untuk bigram "at" = 11101 (5 bit)
Header = 111000001001101011100 (21 bit)

pengujian II (Huffman + Bigram) = 21 + 5 + 5 + 5 = 36

pengujian I (Huffman)           = 11 + 11 + 11 = 33
```

Gambar 4.15 Perbandingan pengujian I dan II pada teks uji ke-7

Gambar 4.15 menunjukkan panjang *header* ditambah dengan tiga kali panjang kode Huffman untuk bigram "at" (pengujian ke 2) lebih besar daripada tiga kali panjang kode Huffman untuk kata "at" (pengujian ke 1). Dalam kasus teks ke-7 ini metode pengujian I menunjukkan hasil yang lebih baik.

Pada teks uji ke 6 rasio kompresi pada metode pengujian I maupun II, sama-sama menghasilkan nilai rasio yang sama. Hal ini menunjukkan bahwa panjang *header* ditambah hasil kompresi dari bigram pada pengujian II sebanding dengan hasil kompresi pada pengujian I.

Teks uji ke 11-15 dengan prosentase bigram 31% - 45% menghasilkan rasio rata-rata sebesar 26,49%. Lebih tinggi dibanding teks uji 1 - 10, dalam tahap ini peran bigram untuk menambah besar rasio kompresi dapat terlihat, prosentase bigram teks uji 11 - 15 cukup tinggi menunjukkan nilai rasio kompresi yang lebih baik.

Pada teks uji ke-12 didapatkan rasio kompresi yang cukup tinggi sebesar 28,50%. Sedangkan pada teks uji ke-13 rasio kompresi yang rendah sebesar 23,29%. Data kompresi teks uji ke 12 dan 13 dapat dilihat pada Tabel 4.4.

Tabel 4.4 Data hasil kompresi teks SMS ke- 12 dan ke- 13

	SMS 12		SMS 13	
	Bigram Kata	**Frekuensi**	**Bigram Kata**	**Frekuensi**
Bigram 1	ka	16	n	6
Bigram 2	ng	11	s	5
Bigram 3	ra	10	e	5
Bigram 4	d	9	t	5
Bigram 5	te	8	le	5
Bigram 6	m	6	do	3
Bigram 7	a	5	ss	3
Bigram 8	bi	5	y	3
Bigram 9	se	4	te	3
Bigram 10	p	3	co	3
Bigram 11	ca	3	re	3
Total Seluruh Bigram		80		44
Bigram yang Berbeda		11		11
Jumlah Karakter		430		222
Prosentase Bigram		37.21%		39.64%
Rasio		28.50498		23.29472

Pada teks SMS ke 12 dan ke 13 sama-sama mempunyai jumlah variiasi bigram yang berbeda sebanyak 11 bigram. Teks SMS ke 12 mempunyai prosentase bigram lebih kecil daripada teks SMS ke 13, tetapi rasio kompresi SMS ke 12 lebih tinggi, hal ini disebabkan karena pada teks SMS ke 13 frekuensi pada setiap bigram sangat kecil dan tersebar merata, sedangkan pada teks SMS ke- 12 frekuensi pada setiap bigram lebih besar, dan terpusat pada bigram pertama yang mempunyai frekuensi sebanyak 16. Bigram-bigram awal mempunyai kode Huffman yang pendek, frekuensi yang tinggi pada bigram-bigram awal akan menyebabkan semakin tinggi nilai rasio kompresi, sehingga hasil kompresi pada teks SMS ke 12 memperoleh hasil yang maksimal.

Pada teks uji dengan prosentase 46% - 60% dapat dilihat pada Tabel 4.3 rasio rata-rata kompresi semakin membaik dibandingkan dengan teks uji yang prosentase bigramnnya lebih rendah, tetapi

tidak selalu teks yang prosentase bigramnya tinggi menghasilkan nilai rasio yang maksimal, contoh pada teks SMS ke-20 nilai rasio sangat kecil dikarenakan jumlah karakter yang terlalu sedikit sedangkan terdapat 2 buah bigram yang berbeda, menyebabkan panjang header besar dan mengurangi nilai rasio kompresi.

Nilai rasio rata-rata tertinggi didapat pada teks yang mempunyai prosentase bigram 60% - 80% yaitu sebesar 36,73%. Banyaknya bigram pada teks menyebabkan hasil kompresi jauh lebih baik. Teks yang mempunyai prosentase bigram dan jumlah karakter yang sama tetapi memiliki variasi bigram yang lebih sedikit maka akan menghasilkan nilai kompresi yang lebih baik. Contoh pada Tabel 4.5 yang merupakan data SMS ke 22 dan 23.

Tabel 4.5 Data hasil kompresi teks SMS ke- 23 dan ke- 24

	SMS 23		SMS 24	
	Bigram Kata	**Frek**	**Bigram Kata**	**Frek**
Bigram 1	pa	5	lu	9
Bigram 2	pi	4	-	-
Total Seluruh Bigram		9		9
Bigram yang Berbeda		2		1
Jumlah Karakter		23		23
Prosentase Bigram		78.26%		78.26%
Rasio		34.16149		41.14286

Pada Tabel 4.4 dapat dilihat kedua teks memiliki Jumlah karakter dan Prosentase bigram yang sama, yang membedakan hanyalah jumlah bigram yang berbeda (variasi bigram). Pada teks SMS ke 24 hanya mempunyai satu jenis bigram sebanyak 9, sedangkan pada teks SMS ke 23 mempunyai 2 jenis bigram sebanyak masing-masing 5 dan 4. Pada kasus seperti ini rasio kompresi yang lebih tinggi yaitu yang mempunyai variasi bigram yang lebih sedikit. Karena dengan adanya variasi bigram yang lebih sedikit, kompresi dapat berpusat pada bigram yang kode Huffmannya pendek dan panjang *header* otomatis lebih kecil.

Berdasarkan Tabel 4.1 dan Tabel 4.2 didapatkan nilai rata-rata rasio kompresi pada pengujian I yang hanya mengunakan metode Huffman sebesar 22,37%. Sedangkan rata-rata rasio kompresi pada pengujian ke II dengan menggunakan metode Huffman dan bigram yaitu sebesar 27,05%. Penambahan metode perulangan bigram pada kode Huffman dapat menambah rasio kompresi sampai sebesar 18,78% (teks uji ke 24) dengan rata-rata penambahan sebesar 5,85%.

Berdasarkan prosentase pendapatan bigram pada pengujian II rata-rata rasio kompresi pada ke-5 bagian yang diuji, rasio paling tinggi didapatkan pada bagian ke 5 yaitu teks dengan prosentase bigram 60% - 80%.

Jika ditinjau dari pengiriman SMS jumlah karakter yang tersedia dalam satu kali pengiriman adalah 160 karakter, dengan metode kompresi Huffman dan bigram jumlah rata-rata karakter dapat ditingkatkan dalam sekali pengirman yaitu sebesar :

Jumlah karakter rata-rata = K + (K * R)
= 160 + (160 * (27,05/100))
= 203, 28 karakter

Keterangan :
K : Jumlah karakter SMS normal
R : Rata-rata rasio Kompresi Huffman + bigram

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, kesimpulan yang diperoleh antara lain:

1. Model kompresi metode Huffman dan bigram dapat diimplementasikan pada kompresi *Short Message Service* (SMS).
2. Berdasarkan pengujian yang telah dilakukan, diperoleh nilai rata-rata rasio dengan menggunakan metode Huffman sebesar 22,37%.
3. Berdasarkan pengujian yang telah dilakukan, diperoleh nilai rata-rata rasio metode Huffman dengan bigram sebesar 27,05%. Atau jika ditinjau dari pengiriman SMS dengan rasio tersebut, dalam satu kali pengiriman SMS, rata-rata jumlah karakter yang dapat ditulis sebanyak 203 karakter.
4. Penambahan metode bigram dalam kode Huffman dapat menambah besarnya rasio kompresi dengan rata-rata penambahan sebesar 5,85% pada setiap teks SMS.

5.2 Saran

Saran yang diberikan untuk mengembangkan penelitian ini antara lain:

1. Melakukan penelitian lebih lanjut tentang analisa pemakaian karakter pada pembuatan *tree* Huffman dengan data - data yang lebih akurat.
2. Memperbanyak jumlah tabel Huffman yang digunakan dengan variasi pemusatan kompresi yang berbeda-beda.
3. Memperbanyak kode Huffman untuk bigram, dan mengganti bit penanda pada *header* dengan bit yang lebih pendek.
4. Melakukan kompresi metode Huffman dengan Trigram, Fourgram dan n-gram yang lain.
5. Menggunakan kompresi metode Huffman dan bigram untuk mengkompresi data teks yang lain, seperti file .txt.

DAFTAR PUSTAKA

American Cryptogram Association. 2005. "Letter Frequency Statistics". http://www.cryptogram.org/cdb/words/frequency.html. Tanggal akses : 5 Maret 2008.

Antonius, A. H. 2004. "Pemrograman Mobile Java dengan MIDP 2.0". Yogyakarta : Andi publishing.

Anynomous. 2005. "Kompresi Dan Teks". Fakultas Teknik Informatika Universitas Kristen Duta Wacana. http://lecturer.ukdw.ac.id/anton/download/multimedia6.pdf., Tanggal akses : 5 Maret 2008.

Ayuningtyas, N. 2007. "Implementasi Kode Huffman dalam Aplikasi Kompresi Teks pada Layanan SMS". Jurusan Teknik Informatika Institut Teknologi Bandung.

Aysar, A. 2008. "Huruf Paling Sering Muncul". http://komputasi.wordpress.com/2008/11/09/bahasa-indonesia-huruf-paling-sering-muncul/. Tanggal akses : 25 november 2008.

Bambang, W. 2007. "Short Message Service (SMS)". http://bambangwinarno.multiply.com/journal. Tanggal akses : 18 April 2008.

Farid. 2007. "SMS Murah untuk Semua Merk HP". http://www.costfix.net/2007/12/19/costfix/smsmurah-untuk-semua merk-hp-dan-operatorselular/. Tanggal akses : 5 Maret 2008.

Galih, G. 2008. "The Most Appear Letter". http://blog.galih.biz/knowledge/the-most-appear-letter/. Tanggal akses : 25 November 2008.

Hatem, M. 2007. “N-gram and Fast Pattern Extraction Algorithm”. http://www.codeproject.com/KB/recipes/Patterns .aspx. Tanggal akses : 21 April 2008.

Sayood, K. 2000. “Introduction to Data Compression”. Second Edition. Morgan Kaufmann Publishers.

Salomon, D. 2004. “Data Compression: The Complete Reference”. New York: Springer-Verlag, Inc.

Owen, L. A. 2004. “From ASCII Coding To Huffman Coding”. http://www.cs.duke.edu/csed/poop/huff/info/. Tanggal akses : 21 April 2008.

Raharjo, B., Imam. H dan A. Haryono. 2007. ”Tuntunan Pemrograman Java Untuk Handphone”. Bandung : Informatika.

Ravi, K. 2008. “ASCII Table: 7-bit”. Dept. of Physiology University of Wisconsin. Madison. http://www.physiology. wisc.edu/comp/docs/ascii/. Tanggal akses : 25 November 2008.

Sigit, N. 2007. “Teknologi SMS”. Telcom Club Yogyakarta. http://telcomclub.gramaweb.com/artdetail.php?id=10. Tanggal akses : 21 April 2008.

Tino, D. “Pengantar Teknologi Informasi – Pengkodean”. http://www.dwiantoro.com/documents/Modul_5_PTI.pdf. Tanggal akses : 21 April 2008.

Wardoyo, I., P. Kusdinar dan I.H. Taufik. 2005. “Kompresi Teks dengan Menggunakan Algoritma Huffman”. Jurusan Teknik Informatika Sekolah Tinggi Teknologi Telkom.

Yuku. 2007. “Kompresi Teks SMS”. http://www.kejut.com/kompresisms. Tanggal akses : 24 Februari 2008.

Lampiran 1

Data Latih Yang Digunakan

SMS ke-	Jumlah Karakter	Isi
1	32	saya sudah nyampe, bapak dimana?
2	37	ok nanti saya tunggu di dpn kntor
3	131	asadafagahajakalaqawaearatayauaiaoapazaxacavabanama eqewereteyeueieoepelekejehegefedesezexecevebenem isiniridiitipigifikili usunudu
4	23	doakan smg berhsl teman
5	54	she smart an independence, i don't think she needs me.
6	70	Bntr y kenalan dulu sprt biasanya, q pst knalan dgan org2 yg minta code
7	39	rapat jam 11, harap datang tepat waktu!
8	152	hehe da tidur? ya uda gapapa kok, pasti kamu baca sms in waktu da pg hr,dimana atahari ud mw terbit, kl gt sekalian met pg aja deh. enjoy your day then.
9	140	A special occurr if the cW denotes an empty try in th dictionary. This can happen because of the explned lagging behind the coding algorithm
10	140	nanti bk puasa dimn don? sm kita kita aj, di kampus, lah awakmu lapo plg, ni ank2 mw jalan nyari takjil di soekarno hatta, melok ta? reneo.
11	169	kemaren saya sudah coba hubungi, tapi tidak ada jawaban, coba langsung ditanya aja ke kantornya. minta kejelasan tentang acara nikahannya. oya untuk tempatnya sudah oke?
12	430	Fungsi pencarian kata dipanggil saat proses kompresi, fungsi ini yang membedakan ketika akan mengkompresi karakter atau bigram. Fungsi mendapat masukkan dua karakter,

		yang mana dari kedua karakter ini akan dicocokkan dengan bigram-bigram yang telah dicari sebelumnnya. Jika dua karakter tersebut adalah bigram maka hasil kembalian dari fungsi ini adalah kode bigram kedua karakter tersebut. Fungsi carikata ditunjukkan pada Gambar
13	222	Work has been done on lossy text compression on a word level, but to my knowledge it has not been done on the letter level as I described. With such a method, text files could meaningfully be compressed tremendous amounts.
14	64	hai namaku Wibisana, nanti pagi aku mau pergi ke sana beli nanas
15	225	masalah tidak terdapatnya kemunculan suatu bit pada stat dapat diatasi dengan menginisialisasi model awal state dengan satu. probabilitas dihtung menggunakan frekuensi reletif dari dua transisi yang keluer dari state yg baru.
16	195	Since all possible onecharacter strings are already in the dictionary, each encoding step begins with a one character prefix, so the first string searched for in the dictionary has two characters
17	132	Dan untuk aplikasinya sendiri, walaupun aplikasi pada ponsel penerima dalam keadaan mati, pesan yang dikirim dari aplikasi yang sama
18	186	bagian-bagian data yang sebenarnya tidak begitu berguna, tidak begitu dirasakan,tidak begit dilihat oleh manusia sehingga manusia masih beranggapan bahwa data tesebut masih bisa dgunakan
19	277	kau lang Kupersembahka satulagu spesial buat kamu, dan kuharap kau jadi tau isi hatiku, kupendam rasa ini jauh didalam, sudah sekian lama aku suka, lewat lagu ini ingin

		kucurahkan isi hati yang tak pernah terungka kau lang kau lang kau lang kau lang kau lang kau lang kau lang
20	21	sari berlari kemari !
21	50	cici cici ke cina pake naci waktu kecil kena nanas
22	46	kata kamu tamu kakak takkan bertemu muka kakak
23	23	papi papi pai pipa pipa
24	23	lulu luka lupa lulululu
25	84	ayo ayo ayo chayo, ayo aku bisa, ayo kamu bisa, ayo ayo samua bisa, ayo bisa, chayo

Lampiran 2

Data hasil pencarian bigram SMS ke 1 - 5

	SMS 1		SMS 2		SMS 3		SMS 4		SMS 5	
	Kata	Frek	Kata	Frek	Kata	Frek	Kata	Frek	Kata	Frek
Bigram 1										
Bigram 2										
Bigram 3										
Bigram 4										
Bigram 5										
Bigram 6										
Bigram 7										
Bigram 8										
Bigram 9										
Bigram 10										
Bigram 11										
Total Seluruh Bigram		0		0		0		0		0
Bigram yang Berbeda		0		0		0		0		0
Jumlah Karakter		32		33		132		23		54
Prosentase Bigram		0.00%		0.00%		0.00%		0.00%		0.00%
Rasio		22.32		23.38		22.03		22.36		22.75

Lampiran 3

Data hasil pencarian bigram SMS ke 6 - 10

	SMS 6		SMS 7		SMS 8		SMS 9		SMS 10	
	Kata	Frek	Kata	Frek	Kata	Frek	Kata	Frek	Kata	Frek
Bigram 1	an	4	at	3	a	8	t	5	i	5
Bigram 2					d	4	e	4	ta	5
Bigram 3							in	4	d	4
Bigram 4									an	3
Bigram 5										
Bigram 6										
Bigram 7										
Bigram 8										
Bigram 9										
Bigram 10										
Bigram 11										
Total Seluruh Bigram		4		3		12		13		17
Bigram yang Berbeda		1		1		2		3		4
Jumlah Karakter		71		39		152		140		140
Prosentase Bigram		11.27%		15.38%		15.79%		18.57%		24.29%
Rasio		21.53%		18.681		24.342		22.653		25.204

Lampiran 4

Data hasil pencarian bigram SMS ke 11 - 15

	SMS 11		SMS 12		SMS 13		SMS 14		SMS 15	
	Kata	Frek	Kata	Frek	Kata	Frek	Kata	Frek	Kata	Frek
Bigram 1	a	10	ka	16	n	6	na	6	at	8
Bigram 2	an	7	ng	11	s	5	I	5	d	8
Bigram 3	t	4	ra	10	e	5	u	3	an	6
Bigram 4	su	3	d	9	t	5			si	6
Bigram 5	ah	3	te	8	le	5			s	5
Bigram 6			m	6	do	3			en	5
Bigram 7			a	5	ss	3			al	3
Bigram 8			bi	5	y	3			t	3
Bigram 9			se	4	te	3			un	3
Bigram 10			p	3	co	3			m	3
Bigram 11			ca	3	re	3			el	3
Total Seluruh Bigram		27		80		44		14		53
Bigram yang Berbeda		5		11		11		3		10
Jumlah Karakter		169		430		222		64		225
Prosentase Bigram		31.95%		37.21%		39.64%		43.75%		47.11%
Rasio		27.726		28.505		23.295		26.146		26.794

Lampiran 5

Data hasil pencarian bigram SMS ke 16 - 20

	SMS 16		SMS 17		SMS 18		SMS 19		SMS 20	
	Kata	Frek	Kata	Frek	Kata	Frek	Kata	Frek	Kata	Frek
Bigram 1	in	7	an	5	an	9	ka	14	ar	3
Bigram 2	e	7	ik	4	a	8	u	13	i	3
Bigram 3	ar	7	ri	4	b	7	la	12		
Bigram 4	ch	5	p	4	gi	5	ng	11		
Bigram 5	ct	5	da	4	da	5	i	6		
Bigram 6	s	5	a	3	si	4	at	5		
Bigram 7	on	4	pl	3	se	3	pe	4		
Bigram 8	t	4	as	3	na	3	ah	4		
Bigram 9	er	3	ma	3	ti	3	,	4		
Bigram 10					t	3	n	4		
Bigram 11										
Total Seluruh Bigram		47		33		50		77		6
Bigram yang Berbeda		9		9		10		10		2
Jumlah Karakter		195		132		186		277		21
Prosentase Bigram		48.21%		50.00%		53.76%		55.60%		57.14%
Rasio		27.253		25.216		28.187		33.058		21.088

Lampiran 6

Data hasil pencarian bigram SMS ke 21 - 25

	SMS 21		SMS 22		SMS 23		SMS 24		SMS 25	
	Kata	Frek	Kata	Frek	Kata	Frek	Kata	Frek	Kata	Frek
Bigram 1	ci	7	ka	8	pa	5	lu	9	ay	10
Bigram 2	na	5	mu	4	pi	4			o	9
Bigram 3	ke	4	ta	3					,	5
Bigram 4									sa	5
Bigram 5									bi	4
Bigram 6										
Bigram 7										
Bigram 8										
Bigram 9										
Bigram 10										
Bigram 11										
Total Seluruh Bigram		16		15		9		9		50
Bigram yang Berbeda		3		3		1		1		5
Jumlah Karakter		50		46		23		23		84
Prosentase Bigram		64.00%		65.22%		78.26%		78.26%		78.57%
Rasio		35.1429		34.472		34.1615		41.1429		38.776