

Penerapan Algoritma Stout Codes Pada Kompresi File Audio Aplikasi Kumpulan Ceramah Ustadz KH. Zainuddin MZ

Aziz Alfaridho Hrp

Program Studi Teknik Informatika, Fakultas Ilmu Komputer & Teknologi Informasi, Universitas Budi Darma, Medan, Indonesia
Jl. Sisingamangaraja No.338, Siti Rejo I, Kec. Medan Kota, Kota Medan, Sumatera Utara, Indonesia
Email: azistaurus97@gmail.com

Abstrak—Pada saat ini perkembangan teknologi sangat berperan penting dalam memudahkan perpindahan data baik audio maupun video, namun dengan banyaknya data yang diterima tidak seimbang dengan kapasitas memori yang dimiliki sehingga membuat memori menjadi lambat bekerja atau lelet maka disarankan untuk melakukan proses kompresi terlebih dahulu terkhusus pada file yang berkapasitas besar. Teknik kompresi biasanya digunakan untuk proses transmisi data dan penyimpanan data. Keuntungan data yang terkompresi antara lain menghemat penyimpanan data memudahkan distribusi data dengan media elektronik maupun media removable seperti flashdisk, CD, DVD, dan lain lain. Berdasarkan keuntunagn seperti di atas penelitian ini menerapkan algoritma kompresi data pada file audio menggunakan algoritma stout code. File audio ceramah Ustadz KH. Zainuddin MZ sangat banyak peminatnya sehingga sangat merugikan jika file tersebut di nikmatin dalam keadaan file yang besar. Oleh karena itu penelitian ini sangat berguna menurut penulis dan di harapkan juga bagi pengguna lain.

Kata Kunci: Kompresi; Ceramah; Audio; Algoritma; Stout Code.

Abstract—*At this time, technological developments play a very important role in facilitating the transfer of data, both audio and video, however, with the amount of data received not being balanced with the memory capacity that is available, making the memory work slowly or slowly, it is recommended to carry out a compression process first, especially on files that large capacity. Compression techniques are usually used for data transmission and data storage processes. The advantages of compressed data include saving data storage, making it easier to distribute data using electronic media or removable media such as flash disks, CDs, DVDs, etc. Based on the advantages above, this research applies a data compression algorithm to audio files using the stout code algorithm. The audio files of Ustadz KH. Zainuddin MZ's lectures are very popular so it would be very detrimental if the files were enjoyed in large files. Therefore, this research is very useful according to the author and it is hoped that it will also be useful for other users.*

Keywords: *Compression; Lecture; Audio; Algorithm; Stout Codes.*

1. PENDAHULUAN

Pada awal keberadaannya telepon hanya digunakan sebagai alat untuk berkomunikasi jarak jauh, namun pada saat ini telepon genggam telah bertransformasi menjadi media untuk mempelajari suatu pokok permasalahan dalam lingkup kehidupan yang berbeda dengan cakupan yang lebih luas, salah satunya dapat memudahkan setiap orang untuk mengakses berbagai bidang ilmu contohnya bidang ilmu agama yang kini tersebar luas di *play store* dalam bentuk aplikasi. Dimana dapat diketahui secara luas, banyak orang yang mencari tentang pelajaran agama dalam sebuah pembahasan ceramah. Mengingat banyak orang yang membutuhkan jawaban seputar agama dari ceramah, maka terciptalah aplikasi yang memuat khusus ceramah-ceramah ustadz kenamaan.

Aplikasi ceramah merupakan sebuah aplikasi dengan kumpulan ceramah-ceramah dengan berbagai pembahasan tentang kehidupan dan amalan yang baik. Dalam aplikasi ceramah ini, penulis mengambil ceramah dari Ustadz KH. Zainuddin MZ yang sudah sangat dikenal dan banyak sekali penggemarnya juga pengikutnya. Keakuratan dalam ceramah dan pembahasan di aplikasi ini di perkuat dengan berbagai acuan dan sumber yakni Alquran dan Hadist dimana sebagai sumber hukum dari umat islam.

Pada aplikasi kumpulan ceramah ini terdapat 49 audio ceramah yang memiliki ukuran yang lumayan besar. Dengan banyaknya jumlah audio tersebut maka juga berpengaruh pada kapasitas penyimpanan yang juga akan semakin membesar, juga akan memakan waktu lebih bila kita ingin mengirim atau membagikan file tersebut kepada orang lain di karenakan ukuranya, bila dilakukan pengkompresian maka proses pengiriman data tersebut menjadi lebih cepat karena ukurannya menjadi semakin kecil. Disini saya melakukan perancangan aplikasi baru dengan memanfaatkan file audio ceramah dari aplikasi yang sudah ada karena dari penjabaran di atas dengan 49 audio ceramah dan besar ukurannya sekitar 20 MB sampai 30 MB sepertinya perlu dilakukan kompresi. kebanyakan aplikasi kumpulan ceramah memiliki kapasitas yang besar sehingga akan membutuhkan banyak ruang untuk menyimpan *file* aplikasi tersebut, banyak orang akan memilih untuk menghapus aplikasi lain agar bisa memasang aplikasi baru di telepon genggamnya, hingga akhirnya akan sangat tidak efektif jika sewaktu waktu aplikasi yang telah dihapus dibutuhkan kembali. Hal ini dapat dihindari apabila aplikasi tersebut melewati proses kompresi terlebih dahulu.

Kompresi sendiri adalah sebuah proses yang dapat mengubah sebuah aliran data masukan data asli ke dalam aliran data yang lain (data yang telah dimampatkan) yang memiliki ukuran yang lebih kecil [1]. Salah satu metode yang digunakan dalam melakukan kompresi atau pemampatan data yaitu algoritma *Stout Codes*. Algoritma *Stout Codes* mengurutkan karakter yang paling banyak muncul ke bit terkecil dan karakter yang langka ke bit terbesar. Dengan begitu, ukuran file dapat diminimalisir dari ukuran aslinya.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian.

Tahapan Penelitian yang digunakan dalam penelitian ini terbagi dalam 5 (lima) tahap antara lain adalah:

a. Studi Literatur

Pada tahap ini penulis mengumpulkan referensi yang diperlukan dalam penelitian, yang mana dilakukan untuk memperoleh data-data atau informasi yang dibutuhkan dalam penelitian ini. Pada tahap studi literature dilakukan pengumpulan buku, jurnal, *e-book*, artikel, makalah, *student papper*, maupun situs internet yang membahas algortima *Stout code* untuk dipelajari lebih lanjut.

b. Analisis dan Perancangan Sistem

Pada tahapan analisis dan perancangan sistem akan dilaksanakan perancangan *flowchart*, *interface*, UML dan perancangan aplikasi kompresi *file* gambar dengan menerapkan algoritma *Stout code*.

c. Implementasi Sistem

Pada tahap implementasi sistem ini dilakukan implementasi terhadap hasil perancangan dengan cara melakukan penulisan program.

d. Pengujian Sistem

Pengujian sistem yang sudah dikembangkan dengan sistematika yang sudah dirancang sedemikian rupa untuk melihat perangkat lunak memberikan hasil yang diinginkan.

e. Dokumentasi

Dalam tahap dokumentasi dilakukan penyusunan laporan dari hasil perancangan sistem dalam format penulisan penelitian.

2.2 Kompresi.

Kompresi sendiri adalah sebuah proses yang dapat mengubah sebuah aliran data masukan (data asli) ke dalam aliran data yang lain (data yang telah dimampatkan) yang memiliki ukuran yang lebih kecil [1]. Kompresi data dalam bidang ilmu komputer, ilmu pengetahuan dan seni adalah sebuah penyajian informasi ke dalam bentuk yang lebih sederhana. Kompresi data dapat di artikan juga sebagai proses yang dapat mengubah sebuah aliran data masukan (sumber atau data asli) ke dalam aliran data yang lain (keluaran atau data yang dimampatkan) yang memiliki ukuran yang lebih kecil [2].

Berdasarkan penguraian diatas maka dapat disimpulkan bahwa kompresi merupakan sebuah proses untuk memampatkan data dengan cara menghilangkan bagian-bagian dari data tersebut, namun bagian yang dihilangkan merupakan bagian-bagian yang tidak kentara oleh indera manusia, contohnya dengan menghilangkan bagian-bagian *pixel* suatu gambar hal ini dilakukan untuk memperkecil kapasitas penyimpanan suatu data.

$$\text{rasio kompresi} = 100 - \left(\frac{\text{ukuran file setelah dikompresi}}{\text{ukuran file sebelum dikompresi}} \right) \times 100\% \dots \dots \dots (1)$$

2.3 Audio.

Audio merupakan salah satu elemen dalam multimedia, di mana elemen ini dapat dirasakan dengan indera pendengaran. *Audio* terdiri dari kata yang diucap, suara-suara, musik dan bahkan gangguan (*noise*). *Audio* sendiri adalah gelombang yang merambat akibat dari tekanan yang melalui beberapa media (benda padat, cair, atau gas). Suara terdiri dari *frekuensi* dan dapat didengar dalam jangkauan pendengaran tingkatan yang cukup kuat untuk didengar [5].

2.4 Algoritma.

Algoritma merupakan teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat denngan jumlah kata terbatas tetapi tersusun secara logis dan sistematis [8]. Dalam makalah pendeknya Quentin stout memperkenalkan dua keluarga RI dan SI dari rekursi, kode panjang variable untuk bilangan bulat, mirip dengan dan lebih umum dari Elias omega dan Even – Rodeh code. Stout code bersifat universal dan asimtotik optimal sebelum membaca sebelumnya pembaca diinginkan bawah panjangn L dari Bilangan bulat n diberikan oleh $L = 1 + \log_2 n$ (persmaan(2.1)).

Dua kelompok kode tergantung pada pilihan parameter interger 1 sekali a nilai(lebih besar dari atau sama dengan 2) untuk 1 telah dipilih kata sandi dalam keluarga pertama terdiri dari awalan RI (n) dan akhiran 0n akhiran adalah nilai biner dari n dalam bit L didahului oleh permisa 0 tunggal. itu awalan RI (n) terdiri dari grup panjang dimulai dengan panjang L dari n yaitu menduhului panjangn L1 dan L maka panjang L2 dari L1 dan seterusnya sampai panjang Li tercapai yang cukup pendek untuk masuk dalam kelompok 1- bit. Perhatikan bahwa kelompok panjang (kecuali mungkin yang paling kiri) dimulai dengan 1. dengan demikian permisa 0 tunggal menunjukkan akhir kelompok panjang. Sebagai contoh kita memilih 1 = 2 dan mengkodekan interger 985 – bit n yang nilainya tepat tidak relevan.

Akhiran adalah string 986 – bit 0n dan awalan di mulai dengan grup L = 985 = 11110110012 panjang L1 dari grup ini adalah 1010 = 10102 panjang L2 dari kelompok kedua adalah 4 = 1002 dan panjang L3 adalah 3 = 112 lengkap oleh karena itu codeword adalah 11|100|1010|1111011001|0|n perhatikan bagaimana kelompok panjang di mulaidengan 1, yang menyiratkan bahwa setiap kelompok diikuti oleh 1, kecuali kelompok panjang terakhir yang diikuti oleh 0. Penguraian sederhana. Dekoder dimulai dengan membaca bit pertama. Ini adalah panjang grup berikutnya. Semakin banyak grup panjang dibaca, hingga grup ditemukan yang diikuti oleh 0. Ini menunjukkan bahwa grup berikutnya adalah n itu sendiri. Dengan latar belakang ini, definisi rekursif dari awalan Rl mudah dibaca dan memahami. Kami

menggunakan notasi $L = 1 + \lceil \log_2 n \rceil$ dan dilambangkan dengan $B(n, l)$ representasi biner l -bit (kode beta) dari integer n . Jadi, $B(12, 5) = 01100$. Untuk $l \geq 2$, awalan didefinisikan oleh:

$$S_l(n) = \begin{cases} B(n, l), & \text{for } 0 \leq n \leq 2^l - 1 \\ R_l(L - l - 1) B(n, L), & \text{for } n \geq 2^l \end{cases} \quad (2)$$

Mereka yang akrab dengan Even – Rodeh code mungkin sudah menyadari bahwa kode ini identik dengan R3. Lebih lanjut, kode omega Elias (Bagian 3.4) berada di antara R2 dan R3 dengan dua perbedaan: (1) kode omega mengkodekan jumlah $L - 1$ dan (2) pemisah 0 ditempatkan di sebelah kanan n .

Tabel 1. Kode Omega mengkodekan jumlah L

R2(985) = 11 100 1010 1111011001	R2(31,925) = 111001111 111110010110110
R2(985) = 100 1010 1111011001	R2(31,925) = 100 1111 111110010110110
R2(985) = 1010 1111011001	R2(31,925) = 1111 111110010110110
R2(985) = 01010 1111011001	R2(31,925) = 01111 111110010110110
R2(985) = 001010 1111011001	R2(31,925) = 001111 111110010110110

Tabel pendek mencantumkan awalan $R_l(n)$ untuk $2 \leq l \leq 6$ dan 985-bit dan 31.925-bit integer. Jelas bahwa awalan terpendek diperoleh untuk nilai parameter $l = L = 1 + \lceil \log_2 n \rceil$. Nilai l yang lebih besar menghasilkan awalan yang sedikit lebih lama, sedangkan nilai yang lebih kecil membutuhkan kelompok yang lebih panjang dan karenanya menghasilkan awalan yang jauh lebih lama. Manipulasi aljabar dasar menunjukkan bahwa rentang panjang terbaik L untuk parameter yang diberikan l diberikan oleh $[2s, 2e - 1]$ di mana $s = 2l - 1$ dan $e = 2l - 1 = 2s - 1$ (ini adalah rentang panjang terbaik L , bukan bilangan bulat terbaik n). Tabel berikut mencantumkan interval ini untuk beberapa nilai l dan menjelaskan bahwa parameter kecil, seperti 2 dan 3, cukup untuk sebagian besar aplikasi praktis kompresi data. Parameter $l = 2$ terbaik untuk bilangan bulat yang panjangnya 2 hingga 7 bit, sedangkan $l = 3$ yang terbaik untuk bilangan bulat yang panjangnya 8 hingga 127 bit.

Keluarga kedua kode Stout serupa, tetapi dengan awalan yang berbeda dilambangkan dengan $S_l(n)$. Untuk nilai l kecil, keluarga ini menawarkan beberapa peningkatan dibandingkan kode R_l . Secara khusus, ini menghilangkan sedikit redundansi yang ada dalam kode R_l karena grup panjang tidak boleh 0 (itulah sebabnya grup panjang dalam kode omega mengkodekan $L - 1$ dan bukan L). Awalan S_l mirip dengan awalan R_l dengan perbedaan bahwa grup panjang untuk L mengkodekan $L - 1 - l$. Jadi, $S_2(985)$, awalan dari bilangan bulat 985-bit n , dimulai dengan grup panjang 10-bit 1111011001 dan menambahkannya ke grup panjang untuk $10 - 1 - 2 = 7 = 1112$. Untuk ini didahului grup panjang untuk $3 - 1 - 2 = 0$ sebagai dua bit 00. Hasilnya adalah awalan 15-bit 00 | 111 | 1111011001, lebih pendek dari 19 bit $R_2(985)$. Contoh lain adalah $S_3(985)$, yang dimulai dengan 1111011001 yang sama dan bergantung padanya kelompok panjang untuk $10 - 1 - 3 = 6 = 1102$. Rekursi berhenti pada titik ini karena 110 adalah grup l -bit. Hasilnya adalah 13-bit codeword 110 | 1111011001, sekali lagi lebih pendek dari 17 bit $R_3(985)$.

Melihat bahwa kolom paling kiri mencantumkan nilai L , misalnya, panjang bilangan bulat yang dikodekan, dan bukan bilangan bulat itu sendiri. Grup panjang mempertahankan nilainya hingga grup yang mengikutinya menjadi semua 1, di mana titik grup bertambah 1 dan grup yang mengikuti diatur ulang ke 10 ... 0. Semua grup panjang, kecuali mungkin yang paling kiri, mulai dengan 1. Perilaku reguler yang akan dijabarkan di bawah ini adalah hasil dari pilihan $L - 1 - l$. Berikut ini di bawah merupakan kode $S_2(n)$ dan kode $S_3(n)$.

Tabel 2. Kode $S_2(n)$ dan Kode $S_3(n)$

L	S2(n)	S3(n)
1	01	001
2	10	010
3	11	011
4	00 100	100
5	00 101	101
6	00 110	110
7	00 111	111
8	01 1000	000 1000
15	01 1111	000 1111
16	10 10000	001 10000
32	11 100000	010 100000
64	00 100 1000000	011 1000000
128	00 101 10000000	100 10000000
256	00 110 100000000	101 100000000
512	00 111 1000000000	110 1000000000
1024	01 1000 10000000000	111 10000000000
2048	01 1001 100000000000	000 1000 100000000000

Awalan S2 (64), misalnya, dimulai dengan grup 7-bit $1000000 = 64$ dan bergantung padanya $S2 (7-1-2) = S2 (4) = 00 | 100$. Kami menekankan lagi bahwa tabel hanya mencantumkan awalan, bukan codeword lengkap. Setelah ini dipahami, tidak sulit untuk melakukannya lihat bahwa kode Stout kedua adalah kode awalan. Setelah codeword diberikan, itu tidak akan menjadi awalan dari codeword lainnya. Jadi, misalnya, awalan dari semua codewords untuk bilangan bulat 64-bit dimulai dengan awalan 00 100 dari bilangan bulat 4-bit, tetapi setiap codeword untuk bilangan bulat 4-bit memiliki 0 mengikuti 00 100, sedangkan codewords untuk bilangan bulat 64-bit memiliki 1 mengikuti 00 100.

3. HASIL DAN PEMBAHASAN

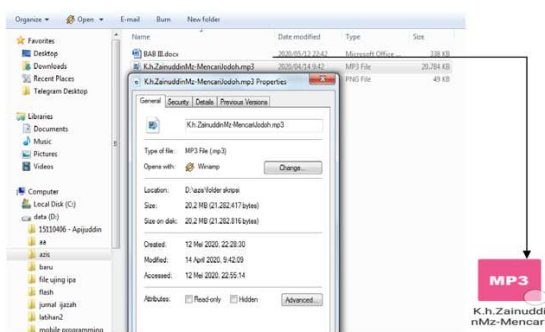
Analisa merupakan suatu penguraian dari sebuah pokok pembahasan, dalam hal ini pembahasan mengenai perancangan aplikasi kompresi audio menggunakan algoritma *Stout Code*. Masalah yang akan dianalisa adalah bagaimana cara mengompres file audio yang ukurannya besar sehingga ukurannya menjadi lebih kecil agar lebih mudah dalam proses penyimpanan maupun pemindahan data.

3.1 Penerapan Algoritma Stout Code.

Berdasarkan analisa, *kompresi files Audio* memiliki ukuran lebih besar dari ukuran yang lainnya. Dengan melakukan kompresi file Audio, file yang berukuran besar akan dikompresi menjadi ukuran yang kecil dan akan mengurangi alokasi penyimpanan. Dalam penelitian ini, akan dibahas 2 proses utama yaitu proses kompresi dan dekompresi, dan peneliti akan mengompresi sebuah *fileAudio* dengan algoritma *stout codes*. Contoh *file Audio* yang akan dikompres adalah *file* berektensi *MP3* yang akan dilakukan sebuah proses kompresi. Sebelum file dikompresi, terlebih dahulu dilakukan pembacaan pada file audio untuk mendapatkan file berupa biner. Membaca biner yang terdapat pada file audio menggunakan aplikasi *Binary Viewer* untuk mencari biner pada aplikasi file *MP3*. Berikut adalah contoh file audio yang akan dikompresi dan dekompresi.

Tabel 3. Sempel File audio yang akan dikompresi

Nama File	K.h.ZainuddinMz-MencariJodoh.mp3
Extension File	MP3
Size	21.28 MB



Gambar 1. File Audio Yang Akan Dikompresi

Data View			
Address...	Text (ASCII)	Hexadecimal	
000 000 000	I D 3 * * *	49 44 33 03 00 00 00 00	
000 000 008	* { T S S E * *	11 7B 54 53 53 45 00 00	
000 000 016	* * * * * L	00 1F 00 00 01 FF FE 4C	
000 000 024	* a * v * f * *	5 00 61 00 76 00 66 00 35	
000 000 032	* 4 * . * 2 * *	9 00 34 00 2E 00 32 00 39	
000 000 040	* * . 1 * 0 * *	4 00 2E 00 31 00 30 00 34	
000 000 048	* * * T I T 2 * *	00 00 00 54 49 54 32 00	
000 000 056	* * * * * 1 o	00 00 14 00 00 00 6C 6F	
000 000 064	* e d i . b l o	67 65 64 69 2E 62 6C 6F	
000 000 072	* g a p o t . c o	67 73 70 6F 74 2E 63 6F	
000 000 080	* m T X X X * *	61 54 58 58 58 00 00 00	
000 000 088	* * * * * m *	27 00 00 01 FF FE 6D 00	
000 000 096	* a * j * o * *	61 00 6A 00 6F 00 72 00	
000 000 104	* _ * b * r * *	5F 00 62 00 72 00 61 00	
000 000 112	* n . d * * * *	6E 00 64 00 00 00 FF FE	
000 000 120	* d * a * s * h *	64 00 61 00 73 00 68 00	
000 000 128	* * T X X X * *	00 00 54 58 58 58 00 00	
000 000 136	* * * * * m	00 25 00 00 01 FF FE 6D	
000 000 144	* i * n * o * *	00 69 00 6E 00 6F 00 72	
000 000 152	* _ * v * e * *	00 5F 00 76 00 65 00 72	

File Name: K.h.ZainuddinMz-MencariJodoh.mp3 | Size: 21,282,417 Bytes |

Gambar 2. Isi file audio Dari Hasil Binary Viewer

Berdasarkan contoh diatas di dapat nilai biner viewer, adapun nilai hexa yang akan di jadikan sampel untuk penelitian ini pada file Audio sampel D:\Azis\FolderSkrripsi\K.h.ZainuddinMz-MencariJodoh.mp3 tersebut adalah:

Tabel 4. Nilai Hexadesimal berdasarkan binary viewer

00	00	14	00	00	00	6C	6F	67
65	64	69	2E	62	6C	6F	67	73
70	6F	74	2E	63	6F	6D		

3.1.1 Kompresi Algoritma Stout Codes.

Dalam proses kompresi algoritma stout codes yaitu dengan mengurutkan frekuensi dari karakter yang muncul lalu merubah karakter tersebut menjadi *codeword*. Berikut ini langkah-langkah untuk mengkompresi file audio menggunakan algoritma *stoutcodes*:

- Buat daftar frekuensi kemunculan tiap-tiap karakter dan mengurutkan dari frekuensi terbesar sampai frekuensi terkecil.

Tabel 5. Daftar Frekuensi Kemunculan Karakter

N	Hexsa	ASCII Biner	Bit	Frekuensi	Bit x Frekuensi
1	00	00000000	8	5	40
2	6F	01101111	8	4	32
3	6C	01101100	8	2	16
4	67	01100111	8	2	16
5	2E	00101110	8	2	16
6	14	0010100	8	1	8
7	65	01100101	8	1	8
8	64	01100100	8	1	8
9	69	01101001	8	1	8
10	62	01100010	8	1	8
11	73	01110011	8	1	8
12	70	01110000	8	1	8
13	74	01110100	8	1	8
14	63	01100011	8	1	8
15	6D	01101101	8	1	8
Total Bit					200

- Mencari codeword dari setiap karakter yang ada berdasarkan data yang ada pada tabel 5. Untuk mencari codeword, pertama sekali menentukan nilai l , pada penelitian ini menggunakan nilai $l = 2$. Setelah menentukan nilai l , maka masukan rumus berikut:

- Untuk nilai $0 \leq n \leq 2^l - 1$

Karena nilai $l=2$, maka:

$$0 \leq n \leq 2^2 - 1 \Rightarrow 0 \leq n \leq 3$$

- $n=1$, maka codeword yang dihasilkan adalah nilai biner dari n yaitu 1 dan diambil sebanyak nilai l yaitu 2 sehingga codewordnya : 01
- $n=2$, maka codeword yang dihasilkan adalah nilai biner dari n yaitu 10 dan diambil sebanyak nilai l yaitu 2 sehingga codewordnya : 10
- $n=3$, maka codeword yang dihasilkan adalah nilai biner dari n yaitu 11 dan diambil sebanyak nilai l yaitu 2 sehingga codewordnya : 11

- Untuk nilai $n \geq 2^l$

Karena nilai $l=2$, maka:

$$n \geq 2^2 \Rightarrow n \geq 4$$

- $n=4$ dengan nilai biner 100, maka nilai $L=3$ diambil dari panjang bit dari nilai biner n dan langkah selanjutnya mencari nilai R .
 $R_2(3-1-2) = 0$, nilai binernya 0 dan diambil sebanyak nilai l , yaitu 2 sehingga nilai $R=00$.
 $R_1(L-1-l)B(n,L)=00100$
- $n=5$ dengan nilai biner 101, maka nilai $L=3$
 $R_2(3-1-2) = 0$, nilai binernya 0 dan diambil sebanyak nilai l , yaitu 2 sehingga nilai $R=00$.
 $R_1(L-1-l)B(n,L)=00101$
- $n=6$ dengan nilai biner 110, maka nilai $L=3$
 $R_2(3-1-2) = 0$, nilai binernya 0 dan di ambil sebanyak nilai l , yaitu 2
 Sehingga nilai $R=00$
 $R_1(L-1-l)B(n,L)=00110$
- $n=7$ dengan biner 111, maka nilai $L=4$
 $R_2(3-1-2) = 0$, nilai binernya 1 dan diambil sebanyak nilai l , yaitu 2
 Sehingga nilai $R=00$

- $R1(L-1-1)B(n,L)=010111$
- e) $n=8$ dengan biner 1000, maka nilai $L=4$
 $R2(4-1-2) = 1$, nilai binernya 1 dan diambil sebanyak nilai 1, yaitu 2
 Sehingga nilai $R=01$
 $R1(L-1-1)B(n,L)=011000$
- f) $n=9$ dengan biner 1001, maka nilai $L=4$
 $R2(4-1-2) = 1$, nilai binernya 1 dan diambil sebanyak nilai 1, yaitu 2
 Sehingga nilai $R=01$
 $R1(L-1-1)B(n,L)=011001$
- g) $n=10$ dengan biner 1010, maka nilai $L=4$
 $R2(4-1-2)= 1$, nilai binernya 1 dan diambil sebanyak nilai 1, yaitu 2
 Sehingga nilai $R=01$
 $R1(L-1-1)B(n,L)=011010$
- h) $n=11$ dengan biner 1011, maka nilai $L=4$
 $R2(4-1-2) = 1$, nilai binernya 1 dan diambil sebanyak nilai 1, yaitu 2
 Sehingga nilai $R=01$
 $R1(L-1-1)B(n,L)=011011$
- i) $n=12$ dengan biner 1100, maka nilai $L=4$
 $R2(4-1-2) = 1$, nilai binernya 1 dan diambil sebanyak nilai 1, yaitu 2
 Sehingga nilai $R=01$
 $R1(L-1-1)B(n,L)=011100$
- j) $n=13$ dengan biner 1101, maka nilai $L=4$
 $R2(4-1-2) = 1$, nilai binernya 1 dan diambil sebanyak nilai 1, yaitu 2
 Sehingga nilai $R=01$
 $R1(L-1-1)B(n,L)=011101$
- k) $n=14$ dengan biner 1110, maka nilai $L=4$
 $R2(4-1-2) = 1$, nilai binernya 1 dan diambil sebanyak nilai 1, yaitu 2
 Sehingga nilai $R=01$
 $R1(L-1-1)B(n,L)=011110$
- l) $n=15$ dengan biner 1111, maka nilai $L=4$
 $R2(4-1-2) = 1$, nilai binernya 1 dan diambil sebanyak nilai 1, yaitu 2
 Sehingga nilai $R=01$
 $R1(L-1-1)B(n,L)=011111$
3. Mengganti karakter yang ada dengan codeword yang telah dihasilkan, dan dapat dilihat pada tabel 4:

Tabel 6. Pergantian karakter menjadi codeword

N	Hexsa	Codeword	Bit	Frekuensi	Bit x Frekuensi
1	00	01	2	5	10
2	6F	10	2	4	8
3	6C	11	2	2	4
4	67	00100	5	2	10
5	2E	00101	5	2	10
6	14	00110	5	1	5
7	65	010111	6	1	6
8	64	011000	6	1	6
9	69	011001	6	1	6
10	62	011010	6	1	6
11	73	011011	6	1	6
12	70	011100	6	1	6
13	74	011101	6	1	6
14	63	011110	6	1	6
15	6D	011111	6	1	6
Total					101

Berdasarkan pada tabel di atas dapat dibentuk nilai bit baru hasil kompresi dari susunan nilai hexadecimal sampel awal sebelum kompresi yaitu, 00, 00, 14, 00, 00, 00, 6C, 6F, 67, 65, 64, 69, 2E, 62, 6C, 6F, 67, 73, 70, 6F, 74, 2E, 63, 6F, 6D. (tanpa ada tanda koma dan spasi) menjadi nilai *bit biner*:

“0101001100101011110001000101110110000110010010101101011100010001101101110010011101001010111101001111”

Kemudian sebelum di dapatkan hasil keseluruhan akhir kompresi dilakukan penambahan *stringbit* itu sendiri yaitu *padding* bit dan *flag* bit. Jika sisa bagi panjang *string* bit terhadap 8 adalah 0 maka tambahan 00000001. Nyatakan dengan bit akhir. Sedangkan jika sisa bagi panjang *string* bit terhadap 8 adalah $n(1, 2, 3, 4, 5, 6, 7)$ maka tambahkan 0 sebanyak

$7 - n + "1"$ di akhir *string bit*. Nyatakan dengan L. Lalu tambahkan bilangan biner dari $9 - n$. nyatakan dengan bit akhir. karena jumlah string bit 101 tidak habis dibagi 8 dan sisanya 5 bit, nyatakan sisa bagi tersebut dengan nilai n . maka tambahkan 0 sebanyak 0 sebanyak $7 - n + "1"$ di akhir string bit. Nyatakan dengan L. Lalu tambahkan bilangan biner dari $9 - n$. Nyatakan dengan bit akhir.

$$\begin{aligned} 7 - n + "1" \\ 7 - 5 + "1" = 001 \\ \text{Bit Akhir } 9 - n \\ \text{Bit Akhir} = 9 - 5 = 4 = 00000100 \end{aligned}$$

Gambar 3. Perhitungan Penambahan Bit

01010011001010111100010001011101100001100100101011010111000100011011011100100111010010101110100111110010000100 Total panjang bit keseluruhan setelah ada penambahan bit adalah $101 + 3 + 8 = 112$. Selanjutnya lakukan pemisahan bit menjadi beberapa kelompok. Setiap kelompok terdiri dari 8 bit.

01010011-00101011-11000100-01011101-10000110-01001010-11010111-00010001-10110111-00100111-01001010-11110100-11111001-00000100

01010011	00101011	11000100	01011101	10000110	01001010
11010111	00010001	10110111	00100111	01001010	11110100
11111001	00000100				

Berdasarkan pada pembagian kelompok nilai biner, didapatkan 6 kelompok nilai biner baru yang sudah terkompresi beserta nilai biner penambahan bit. Setelah pembagian dilakukan, maka nilai yang sudah dibagi dirubah kedalam suatu karakter dengan terlebih dahulumencairi nilai desimal dari *string bit* tersebut menggunakan kode ASCII untuk mengetahui nilai yang sudah terkompresi. Nilai simbol yang sudah terkompresi dapat dilihat pada tabel di bawah ini:

Tabel 7. Nilai Desimal terkompresi

Biner	Nilai Desimal Terkompresi	Simbol
01010011	83	S
00101011	43	+
11000100	196	Ä
01011101	93	]
10000110	134	+
01001010	74	J
11010111	215	×
00010001	17	11
10110111	183	.
00100111	39	'
01001010	74	J
11110100	232	È
11111001	249	Ð
00000100	4	4

Setelah mengetahui hasil kompresi tersebut, hasil kompresi dapat di ukur atau di hitung sebagai berikut:

$$Cr = 100\% - \frac{\text{ukuranbitdata setelah dikompresi}}{\text{ukuranbitdata sebelum dikompresi}} \times 100\%$$

$$Cr = 100\% - \frac{112 \text{ bit}}{200 \text{ bit}} \times 100\%$$

$$Cr = 100\% - 56\%$$

$$Cr = 44\%$$

Selanjutnya proses dekompresi hal yang dilakukan adalah menganalisa keseluruhan bit hasil dari kompresi sebelumnya. Proses dekompresi sangat penting untuk membuktikan bahwa pengkompresia yang dilakukan sudah benar. Adapun bit keseluruhan hasil kompresi dapat dilihat pada tabel yang di jabarkan seperti di bawah berikut:

Tabel 8. Hasil Tekompres

Simbol	Nilai Desimal Terkompresi	Biner
S	83	01010011
+	43	00101011
Ä	196	11000100
]	93	01011101
+	134	10000110

Simbol	Nilai Desimal Terkompresi	Biner
J	74	01001010
×	215	11010111
11	17	00010001
.	183	10110111
,	39	00100111
J	74	01001010
	232	11110100
Đ	249	11111001
4	4	00000100

Berdasarkan pada tabel di atas maka diambil seluruh nilai biner dan digabungkan menjadi:
 “01010011001010111100010001011101100001100100101011010111000100011011011100100111010010101111010
 01111100100000100”

Selanjutnya adalah dengan mengembalikan *binary* menjadi *string* bit semula dengan menghilangkan biner yang ditebalkan. Untuk mengembalikan *binary* menjadi *string* bit semula dapat dilakukan melalui langkah berikut ini. Lakukan pembacaan pada 8 bit terakhir, hasil pembacaan berupa bilangan desimal. Nyatakan hasil pembacaan dengan n. Hilangkan bit pada bagian akhir sebanyak 7 + n. Setelah dilakukan perhitungan pembacaan bit akhir. Nilai biner yang dihilangkan sebanyak 8 bit pada akhir. n = 5. Hilangkan 7 + n atau 7 + 5 = 12. Penjelasan diatas menunjukan bahwa bit akhir harus dihilangkan. Hasil pengembalian *binary* menjadi *string* bit semula dapat dilihat sebagai berikut ini:
 “01010011001010111100010001011101100001100100101011010111000100011011011100100111010010101111010
 011111”

Berdasarkan perhitungan dengan algoritma *stout codesstring bit* pada diatas berjumlah 101 bit seperti diawal sehingga dilakukan pembacaan *string* bit awal. Adapun tabel hasil perhitungan diatas adalah sebagai berikut:

Tabel 9. Pengecekan Bit

Indeks	Nilai	Keterangan
1	0	Tidak Ada
2	01	Ada pada table (00)
3	0	Tidak Ada
4	01	Ada pada tabel (00)
5	0	Tidak Ada
6	00	Tidak Ada
7	001	Tidak Ada
8	0011	Tidak ada
9	00110	Ada Pada tabel (14)
10	0	Tidak Ada
11	01	Ada pada tabel (00)
12	0	Tidak Ada
13	01	Ada pada tabel (00)
14	0	Tidak Ada
15	01	Ada pada tabel (00)
16	1	Tidak Ada
17	11	Ada pada tabel (6C)
18	1	Tidak Ada
19	10	Ada pada tabel (6F)
20	0	Tidak Ada
21	00	Tidak Ada
22	001	Tidak Ada
23	0010	Tidak Ada
24	00100	Ada Pada tabel (67)
25	0	Tidak Ada
26	01	Tidak Ada
27	010	Tidak Ada
28	0101	Tidak Ada
29	01011	Tidak Ada
30	010111	Ada pada tabel (65)
31	0	Tidak Ada
32	01	Tidak Ada
33	01	Tidak Ada
34	010	Tidak Ada
35	0101	Tidak Ada

Indeks	Nilai	Keterangan
36	011000	Ada pada tabel (64)
37	0	Tidak Ada
38	01	Tidak Ada
39	011	Tidak Ada
40	0110	Tidak Ada
41	01100	Tidak Ada
42	011001	Ada pada tabel (69)
No	Biner	Keterangan
43	0	Tidak Ada
44	01	Tidak Ada
45	001	Tidak Ada
46	0010	Tidak Ada
47	0010	Tidak Ada
48	00101	Ada pada tabel (2E)
49	0	Tidak Ada
50	01	Tidak Ada
51	011	Tidak Ada
52	0110	Tidak Ada
53	01101	Tidak Ada
54	011010	Ada pada tabel (62)
55	1	Tidak Ada
56	11	Ada pada tabel (6C)
57	1	Tidak Ada
58	10	Ada pada tabel (6F)
59	0	Tidak Ada
60	00	Tidak Ada
61	001	Tidak Ada
62	00100	Ada pada tabel (67)
63	0	Tidak Ada
64	01	Tidak Ada
65	011	Tidak Ada
66	0110	Tidak Ada
67	01101	Tidak Ada
68	011011	Ada pada tabel (73)
69	0	Tidak Ada
70	01	Tidak Ada
71	011	Tidak Ada
No	Biner	Keterangan
72	0111	Tidak Ada
73	01110	Tidak Ada
74	011100	Ada pada tabel (70)
75	1	Tidak Ada
76	10	Ada pada tabel(6F)
77	0	Tidak Ada
78	01	Tidak Ada
79	011	Tidak Ada
80	0111	Tidak Ada
81	01110	Tidak Ada
82	011101	Ada pada tabel (74)
83	0	Tidak Ada
84	00	Tidak Ada
85	001	Tidak Ada
86	0010	Tidak Ada
87	00101	Ada pada tabel (2E)
88	0	Tidak Ada
89	01	Tidak Ada
90	011	Tidak Ada
91	0111	Tidak Ada
92	01111	Tidak Ada
93	011110	Ada pada tabel (63)
94	1	Tidak Ada

Indeks	Nilai	Keterangan
95	10	Ada pada tabel (6F)
96	0	Tidak Ada
97	01	Tidak Ada
98	011	Tidak Ada
99	0111	Tidak Ada
100	01111	Tidak Ada
101	011111	Ada pada tabel (6D)

Maka dari penjabaran diatas dapat dibentuk tabel *stout codes* dan nilai *hexa*awal. Berdasarkan hasil dekompresi di atas didapati nilai sampel hexadesimal berdasarkan binary viewer sehingga dihasilkan *string* semula seperti tabel di bawah:

Tabel 10. Tabel *Hexa* dan *stout codes*

N	Hexsa	Codeword	Bit	Frekuensi	Bit x Frekuensi
1	00	01	2	5	10
2	6F	10	2	4	8
3	6C	11	2	2	4
4	67	00100	5	2	10
5	2E	00101	5	2	10
6	14	00110	5	1	5
7	65	010111	6	1	6
8	64	011000	6	1	6
9	69	011001	6	1	6
10	62	011010	6	1	6
11	73	011011	6	1	6
12	70	011100	6	1	6
13	74	011101	6	1	6
14	63	011110	6	1	6
15	6D	011111	6	1	6
Total					101

Berdasarkan hasil dekompresi di atas didapati nilai *hexadesimal* awal sebelum kompresi sebagai berikut, 00, 00, 14, 00, 00, 00, 6C, 6F, 67, 65, 64, 69, 2E, 62, 6C, 6F, 67, 73, 70, 6F, 74, 2E, 63, 6F, 6D

4. KESIMPULAN

Berdasarkan pembahasan dan evaluasi sebelumnya dan analisa terhadap penerapan algoritma *Stout codes* terhadap kompresi *file* audio aplikasi kumpulan ceramah ustadz KH. Zainuddin MZ, maka dihasilkan kesimpulan yaitu Proses kompresi dan dekompresi pada *file* audio dilakukan dengan proses *encoding* dan *decoding* yang terdapat pada metode algoritma *Stout codes*. Algoritma *Stout Codes* dapat diterapkan untuk melakukan kompresi karakter yang terdapat pada *file* audio ceramah sehingga ukuran *file* menjadi lebih kecil. Aplikasi kompresi *file* audio telah selesai dirancang dengan menggunakan *Eclipse* dimana *file* audio didalamnya telah terkompresi dengan menggunakan algoritma *stout code*.

REFERENCES

- [1] S. David, "Hand Book Of Data Compression," in *Compression*, 5th ed., California: Springer London Dordrecht Heidelberg New York, 2010, p. 2.
- [2] H. Rahmat, "Implementasi Kompresi Data Pada Jaringan Komputer Menggunakan Algoritma Zlib," *Kompresi*, vol. 09, pp. 2–7.
- [3] D. Salomon, *Data Compression*, 4th ed. California: Springer Verlag London New York, 2007.
- [4] Fathansyah, "Basis Data. Revisi Kedua," in *Informatika*, Bandung, 2015.
- [5] Y. Arifin, S. Kom, M. Y. Ricky, S. Kom, V. Yesmaya, and S. Kom, *DIGITAL MULTIMEDIA*. Jakarta Barat, DKI Jakarta Raya 11480: PT WIDIA INOVASI NUSANTARA, 2015.
- [6] P. Studi, I. Komputasi, U. Telkom, and L. Firmansah, "Kompresi Data Audio Lossless format FLAC menjadi Audio Lossly format MP3 dengan Algoritma Huffman Shift Coding AUDIO LOSSY FORMAT MP3 DENGAN ALGORITMA HUFFMAN," vol. 2, no. 3, pp. 8066–8073, 2015.
- [7] C. R. C. P. Llc, *THE TRANSFORM AND DATA COMPRESSION*. 2001.
- [8] Fl.Sigit Suyantoro, Ed., *ALGORITMA DAN PEMOGGRAMAN*, 2nd ed. Yogyakarta: C.V Andi, 2012.
- [9] R. A.S, *REKAYASA PERANGKAT LUNAK TERSTRUKTUR DAN BERORIENTASI OBJEK*. Bandung: Informatika Bnadung, 2016.
- [10] A. Nugroho, *Pemograman JAAVA untuk Aplikasi Basis Data dengan Teknik XP Menggunakan IDE Eclipse*, 1st ed. Yogyakarta, 2007.
- [11] M. S. Alfa Satyaputra, *JAVA for Beginners with eclipse 42 Juno*. Jakarta, 2012.