GIT GEMILANG INFORMATIKA

Vol 2, No2, Juli 2024, Hal. 63-71
ISSN 2985-721X(media online)
https://www.journal.hdgi.org/index.php/git/index

Penerapan Metode Even-Rodeh Code Pada Aplikasi Lirik Lagu Berbasis Android

Riska Devi Hanum[1*], Henky Japina[2]
[1]Teknik Informatika, S1, STMIK Budi Darma, Medan, Indonesia
[2]Fakultas Ekonomi; Prodi Ekonomi Pembangunan; Universitas Asahan
Email: deviriska47@gmail.com, hjapina@gmail.com

Abstrak- Lirik lagu pada aplikasi berbasis android terkadang memiliki syair yang berulang-ulang pada setiap bait nya, maka dari itu diperlukan lah aplikasi kompresi agar tidak memiliki banyak lirik. Dengan menerapkan metode even-Rodeh Code, agar dapat mengetahui kinerja untuk pengkompresian lirik lagu pada android, sehingga lirik lagu yang berukuran besar di kompresi menjadi ukuran yang leih kecil. Aplikasi lirik lagu tersebut merupakan salah satu aplikasi yang memiliki banyak data teks. Dikarenakan aplikasi lirik lagu menyimpan berbagai macam teks lirik dari berbagai lagu yang ada, baik dari lagu daerah, lagu nasional, lagu dalam negeri maupun lagi luar negeri. Adapun faktor yang menjadi pemilihan algoritma selalu digunakan di dalam kompresi data yaitu (memory, kecepatan PC), kecepatan kompresi, ukuran hasil kompresi, besarnya redudansi dan kompleksitas algoritma. Salah satu algoritma kompresi yang dikenal adalah algoritma Even-Rodeh Code. Algoritma Even-Rodeh Code adalah algoritma kompresi data yang mengkodekan karkater-karakter dibuat berdasarkan frekuensi setiap karakter.

Kata kunci : Lirik lagu, *even-rodeh code*

Abstract- Song lyrics in Android-based applications sometimes have repeated verses in each verse, therefore a compression application is needed so that they don't have a lot of lyrics. By applying the Even-Rodeh Code method, we can find out the performance of compressing song lyrics on Android, so that large song lyrics are compressed to a smaller size. This song lyrics application is an application that has a lot of text data. Because the song lyrics application stores various kinds of lyric texts from various existing songs, both regional songs, national songs, domestic and foreign songs. The factors that determine the algorithm selection are always used in data compression, namely (memory, PC speed), compression speed, compression result size, amount of redundancy and algorithm complexity. One of the known compression algorithms is the Even-Rodeh Code algorithm. The Even-Rodeh Code algorithm is a data compression algorithm that encodes characters based on the frequency of each character.

Keywords: Song lyrics, even-rodeh code

1.PENDAHULUAN

Perkembangan teknologi komunikasi dan informasi telah mengubah cara kita mengakses dan menikmati musik. Salah satu inovasi yang berkembang pesat adalah aplikasi lirik lagu berbasis Android, yang memungkinkan pengguna untuk melihat lirik secara real-time saat mendengarkan musik. Namun, seiring dengan peningkatan jumlah pengguna dan kebutuhan akan konten yang semakin besar, tantangan dalam pengelolaan data, khususnya dalam penyimpanan dan pengolahan lirik lagu yang efisien, menjadi semakin nyata.

Masalah utama yang sering dihadapi dalam pengembangan aplikasi lirik lagu adalah besarnya ukuran data lirik, yang dapat berdampak pada kinerja aplikasi, termasuk waktu pemuatan, penggunaan memori, dan konsumsi data. Oleh karena itu, diperlukan metode kompresi data yang efektif untuk mengatasi masalah ini, tanpa mengurangi kualitas atau akurasi dari lirik yang disajikan.

Salah satu metode kompresi data yang potensial untuk diterapkan dalam konteks ini adalah Even-Rodeh Code. Metode Even-Rodeh Code dikenal sebagai teknik kompresi lossless yang dapat mengurangi ukuran data dengan efisien. Metode ini menggunakan pendekatan kode variabel yang dirancang untuk meminimalkan panjang rata-rata kode, sehingga memungkinkan penyimpanan dan transmisi data yang lebih efektif.

Penelitian ini bertujuan untuk mengeksplorasi penerapan metode Even-Rodeh Code pada aplikasi lirik lagu berbasis Android. Fokus utama dari penelitian ini adalah untuk mengukur efektivitas metode ini dalam mengompresi data lirik lagu, serta untuk mengevaluasi dampaknya terhadap kinerja aplikasi, termasuk kecepatan akses dan penggunaan sumber daya.

Penelitian yang dilakukan oleh Surya Darma Nst dengan judul "Perancangan Aplikasi Kompresi *File* Teks Dengan Menerapkan Algoritma Goldbach Codes" menyimpulkan bahwa hasil kompresi menggunakan algoritma goldbach codes sebelum dan sesudah dikompresi mencapai 50% rasio perbandingannya. Penelitian yang dilakukan oleh Heru Cahya Rustamaji dengan judul "Aplikasi Kompresi Data Menggunakan Metode Huffman Statik Pada Perangkat

GIT GEMILANG INFORMATIKA
Vol 2, No2, Juli 2024, Hal. 63-71
ISSN 2985-721X(media online)
https://www.journal.hdgi.org/index.php/git/index

Mobile Berbasis *Android*" menyimpulkan bahwa berdasarkan analisis terhadap ukuran hasil kompresi bahwa pemampatan setiap *file* tergantung dari karakteristik *file* yang akan dimampatkan, sedangkan dari hasil analisis kecepatan terhadap pemampatan *file* adalah jika ukuran *file* semakin besar maka akan semakin lama proses pengkompresian.
Dengan menerapkan metode Even-Rodeh Code, diharapkan aplikasi lirik lagu dapat berfungsi dengan lebih efisien, memberikan pengalaman pengguna yang lebih baik, dan memungkinkan penghematan pada penggunaan data dan penyimpanan. Hasil penelitian ini diharapkan dapat memberikan kontribusi signifikan dalam pengembangan aplikasi musik yang lebih canggih dan efisien di masa mendatang.

2. METODOLOGI

2.1 Aplikasi

Aplikasi dapat dikatakan suatu prangkat lunak yang siap pakai dengan menjalankan intruksi-intruksi dari *user* atau pengguna, aplikasi banyak diciptakan guna membantu berbagai keperluan seperti untuk laporan, percetakan dan lain-lain sedangkan istilah aplikasi berasal dari bahaa inggris "*application*" yang berarti penerapan, lamaran ataupun penggunaan, jadi aplikasi dapat di simpulkan merupakan program siap pakai yang membantu mencapai tujuan pengguna (Eka Noviansyah, Aplikasi *website* museuum nasional menggunakan *macromedia Dreamweaver MX*, 2008) [4].

Aplikasi juga dapat dikatakan sebagai penggunaan dan penerapan suatu konsep yang menjadi suatu pokok pembahasan. Aplikasi dapat diartikan juga sebagai program komputer yang dibuat untuk menolong manusia dalam melakukan tugas tertentu. Aplikasi *software* yang direncanakan untuk suatu tugas khusus dapat dibedakan menjadi dua jenis, yaitu :

1. Aplikasi *software* spesialis, program dengan dokumentasi tergabung yang dijalankan untuk menjalankan tugas tertentu.
2. Aplikasi *software* paket, suatu program dengan dokumentasi tergabung yang dirancang.

2.2 Kompresi Data

Kompresi data telah dimanfaatkan dalam berbagai aspek multimedia. gambar, audio maupun video yang kita dapatkan dari *web* merupakan *file* teks yang telah terkompresi. TV-HD juga merupakan hasil kompresi MPEG-2, kebanyakan modem juga melakukan proses kompresi data, dan beberapa sistem berkas otomatis mengkompresi data ketika data tersimpan (Blelloch, 2013). Sehingga alasan kita membutuhkan kompresi data yaitu karena tegnologi ini memudahkan kita mendapat informasi atau data yang berkualitas dan tidak menghabiskan ruang penyimpanan kita.
Kompresi data adalah proses mengkonversi atau mengubah data input atau data asli menjadi data yang memiliki ukuran yang lebih kecil. Kompresi data menjadi popular dikarenakan dua hal yakni perilaku manusia yang suka menumpuk data dan tidak suka membuangnya dan ketidak senangan manusia dalam menunggu transfer data yang lama (Salomon, 2007). Dalam melakukan kompresi dilakukan 2 metode yaitu Kompresi dan Decompresi. Kompresi merupakan metode atau algoritma untuk mencapai tujuan kompresi pada data asli. Dibalik setiap algoritma terdapat ide, model matematika atau teknik implementasi untuk mencapai tujuan dalam mengkompresi data dengan mempertimbangkan aspek efisiensi dan keefektifan kompresi. Dekompresi merupakan teknik untuk mengembalikan data yang telah dikompresi supaya data dapat kembali ke dalam bentuk yang semula atau data awal (Pu, 2016)

1.3 Metode *even-rodeh code*

Algoritma *Even-Rodeh Code* Merupakan algoritma kompresi yang di kembangkan oleh Shimon Even dan Michael Rodeh pada tahun 1978 (Salomon, 2007). Algoritma *Even-Rodeh Code* adalah algoritma komperesi data yang mengkodekan setiap krakter dengan menggunakan beberapa rangkain bit. Pembentukan bit yang mewakili masing-masing krakter dibuat bedasarkan frekuensi kemunculan tiap karakter. Cara untuk membangun kode *Even-Rodeh Code* adalah sebagai berikut [7] :

1. Hitung panjang bit.
2. jika panjang bit 0 <= n <=3 maka nilai n diubah ke biner, tambahkan 0 di depan nilai biner sehingga bit menjadi 3 digit.
3. Jika panjang bit 4 <= n <= 7 maka nilai n diubah ke biner, tambahkan 0 di belakang nilai biner sehingga bit menjadi 4 digit.

GIT GEMILANG INFORMATIKA

Vol 2, No2, Juli 2024, Hal. 63-71
ISSN 2985-721X(media online)
https://www.journal.hdgi.org/index.php/git/index

4. Jika panjang bit n >= 8 maka nilai n diubah ke biner, tambahkan 0 dibelakang nilai biner kemudian angka ditambahkan di depan nilai biner sebanyak jumlah digit nilai biner.

Berikut beberapa daftar kode dan jumlah bit kode *Even-Rodeh Code* berdasarkan frekuensi karakter dapat dipilih pada tabel 2.1 dan tabel 2.2

Tabel 1 Kode Algoritma *even-rodeh code*

N	Even-Rodeh
0	000
1	001
2	010
3	011
4	100 0
7	111 0
8	100 1000 0
15	100 1111 0
16	100 10000 0
32	110 100000
100	111 110 100 0
1000	110 1100100 0

Sumber : Abdullah, A. R. 2016

Tabel 2 Jumlah kode *Even-Rodeh* karakter

N	Jumlah bit *Even-Rodeh*
0-3	3
4-7	4
8-15	8
16-31	9
32-63	110
64-127	11

Sumber: Abdullah, A. R. 2016

Contoh sederhana pada proses komperesi dan dikomperesi *file* dengan metode *Even-Rodeh Code* pada string “Maria menari” adalah sebagai berikut :

1. Proses komperesi *file* dengan algoritma *Even-Rodeh Code.*

Tabel 3 *String* yang sudah dikomperesi dengan *Even-Rodeh*

Karter	*Even-Rodeh Code*	Bit	Frekuensi	Bit x Frekuensi
A	000	3	3	9
M	001	3	2	6
R	010	3	2	6
I	011	3	2	6
Spasi	1000	4	1	4
E	1010	4	1	4
N	1100	4	1	4

GIT GEMILANG INFORMATIKA

Vol 2, No2, Juli 2024, Hal. 63-71
ISSN 2985-721X(media online)
https://www.journal.hdgi.org/index.php/git/index

Total bit	39

Sumber: Abdullah, A. R. 2016

Berdasarkan tabel 2.1 maka string ”Maria Menari” setelah dikomperesi menjadi sebagai berikut : “00100001001100010000011010110000010011”.

Dengan uraian sebagai berikut :

001 000 010 011 000 1000 001 1010 1100 000 010 011

Sebelum di tulis ke sebuah *file* hasil komperesi dilakukan penambahan bit-bit *padding* dan flag bits diawal dan di ahir *string* bit. Bit-bit dihasilkan dari panjang *tring* bit itu sendiri apakah habis dibagi delapan dan dan berapa sisanya jika dibagi delapan. Karena jumlah *string* bit adalah 39, tidak habis dibagi delapan dan sisanya 7. Sehingga dibutuhkan penambahan bit 0 sebanyak 1 kal, maka *padding* adalah “0” dan flag-nya yaitu “00000001”.

2.4 Data

Pada dasarnya data adalah kumpulan informasi atau keterangan-keterangan dari suatu hal yang di peroleh melalui pengamatan atau pencarian ke sumber-sumber tertentu. Data yang diperoleh dapat menjadi suatu anggapan atau fakta karena memang belum diolah lebih lanjut, setelah diolah melalui penelitian atau percobaan maka suatu data dapat menjadi bentuk yang lebih kompleks seperti suatu *database*. Dari segi bahasa kata data diambil dari kata datum yang dalam bahas romawi diartikan sebagai sesuatu yang diberikan, defenisi sesungguhnya dari data adalah diberikan bukan memberikan, karena jika memberikan maka data tersebut sudah menjadi informasi yang baku dan diakui kebenarannya. Istilah data memang lebih banyak ditemui pada bidang komputer atau dalm lingkup suatu penelitian.
Data adalah suatu fakta mentah ataupn rincian peristiwa yang masih belum diolah serta kadang tak bisa diterima akal pikiran dari penerima data tersebut, sehingga data harus diolah lebih dulu menjadi informasi supaya bisa diterima oleh penerima. Suatu data juga bisa berbentu angka, simbol, karakter, gambar, suara, ataupun tanda-tanda yang bisa dijadikan sebagai suatu informasi. Sementara sebuah informasi bisa menjadi data jika informasi tersebut dipakai kembali untuk pengolahan pada sistem berikutnya. Didalam ilmu komputer data merupakan segala sesuatu yang telah disimpan pada memori berdasrkan format tertentu.

2.5 Teks

Teks adalah kumpulan dari karakter – karakter atau string yang menjadi satu kesatuan. Teks yang memuat banyak karakter didalamnya selalu menimbulkan masalah pada media penyimpanan dan kecepatan waktu pada saat transmisi data. Data teks merupakan *file* yang berisi informasi-informasi dalam bentuk teks. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, merupakan contoh masukan data teks yang terdiri dari karakter, angka dan tanda baca.
Masukan dan keluaran data teks direpresentasikan sebagai set karakter atau sistem kode yang dikenal oleh sistem komputer. Salah satu set karakter yang umum digunakan untuk masukan dan keluaran pada komputer yaitu ASCII (*American Standard Code for Information Interchange*). ASCII digunakan oleh untuk menunjukkan teks. Kode ASCII memiliki komposisi bilangan biner sebanyak 8 bit, dimulai dari 00000000 dan 11111111. Total kombinasi yang dihasilkan sebanyak 256, dimulai dari kode 0 hingga 255 dalam sistem bilangan desimal.

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Pada penelitian ini akan dilakukan proses analisa algoritma kompresi *Even Rodeh Code* dalam mengompresi isi *record database* aplikasi lirik lagu. Proses kompresi ini ditujukan untuk mengatasi masalah pemakaian ruang penyimpanan yang besar dengan cara mengurangi karakter yang sama sehingga dapat menghemat ruang penyimpanan. Hasil dari proses kompresi berupa karakter baru yang telah diganti akan disimpan ke dalam isi *record database*. Lalu teks tersebut akan dilakukan proses kompresi terlebih dahulu sebelum disimpan pada *database*. Keluaran dari hasil kompresi dalah teks atau simbol baru yang ukuranya telah berubah dari kapasitas awal. Proses dekompresi adalah proses pengembalian teks terkompresi menjadi karakter awal untuk ditampilkan pada halaman *output*.

GIT GEMILANG INFORMATIKA

Vol 2, No2, Juli 2024, Hal. 63-71
ISSN 2985-721X(media online)
https://www.journal.hdgi.org/index.php/git/index

3.1.1 Contoh Kasus

Berikut ini adalah contoh proses kompresi teks dengan metode *Even-Rodeh Code* dan teks yang akan menjadi *input* adalah "JADI AKU SEBENTAR SAJA". Penjelasan teks yang belum dikompresi terdapat pada tabel 1.

Karakter	Penjelasan
J	String J diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka J bernilai 74. Dengan ASCII Code (Binary) yaitu 01001010 sebanyak 8 bit. Frekuensi string J dalam "JADI AKU SEBENTAR SAJA" sebanyak 2. Setelah itu Bit dikalikan Frekuensi. 8 x 2 = 16. Jadi keseluruhan bit untuk string J adalah 16 bit.
A	String A diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka A bernilai 65. Dengan ASCII Code (Binary) yaitu 010000001 sebanyak 8 bit. Frekuensi string A dalam "JADI AKU SEBENTAR SAJA" sebanyak 5. Setelah itu Bit dikalikan Frekuensi. 8 x 5 = 40. Jadi keseluruhan bit untuk string A adalah 40 bit.
D	String D diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka D bernilai 68. Dengan ASCII Code (Binary) yaitu 01000100 sebanyak 8 bit. Frekuensi string R dalam "JADI AKU SEBENTAR SAJA" sebanyak 1. Setelah itu Bit dikalikan Frekuensi. 8 x 1 = 8. Jadi keseluruhan bit untuk string D adalah 8 bit.
I	String I diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka I bernilai 73. Dengan ASCII Code (Binary) yaitu 01001001 sebanyak 8 bit. Frekuensi string I dalam "JADI AKU SEBENTAR SAJA" sebanyak 1. Setelah itu Bit dikalikan Frekuensi. 8 x 1 = 8. Jadi keseluruhan bit untuk string Spasi adalah 8 bit.
Spasi	String Spasi diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka Spasi bernilai 32. Dengan ASCII Code (Binary) yaitu 00100000 sebanyak 8 bit. Frekuensi string Spasi dalam "JADI AKU SEBENTAR SAJA" sebanyak 3. Setelah itu Bit dikalikan Frekuensi. 8 x 3 = 24. Jadi keseluruhan bit untuk string Spasi adalah 24 bit.
K	String K diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka K bernilai 75. Dengan ASCII Code (Binary) yaitu 01001011 sebanyak 8 bit. Frekuensi string U dalam "JADI AKU SEBENTAR SAJA" sebanyak 1. Setelah itu Bit dikalikan Frekuensi. 8 x 1 = 8. Jadi keseluruhan bit untuk string K adalah 8 bit.
U	String U diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka U bernilai 85. Dengan ASCII Code (Binary) yaitu 01010101 sebanyak 8 bit. Frekuensi string U dalam "JADI AKU SEBENTAR SAJA" sebanyak 1. Setelah itu Bit dikalikan Frekuensi. 8 x 1 = 8. Jadi keseluruhan bit untuk string U adalah 8 bit.
S	String S diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka S bernilai 83. Dengan ASCII Code (Binary) yaitu 01010011 sebanyak 8 bit. Frekuensi string S dalam "JADI AKU SEBENTAR SAJA" sebanyak 2. Setelah itu Bit dikalikan Frekuensi. 8 x 2 = 16. Jadi keseluruhan bit untuk string S adalah 16 bit.
E	String E diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka E bernilai 69. Dengan ASCII Code (Binary) yaitu 01000101 sebanyak 8 bit. Frekuensi string U dalam "JADI AKU SEBENTAR SAJA" sebanyak 2. Setelah itu Bit dikalikan Frekuensi. 8 x 2 = 16. Jadi keseluruhan bit untuk string E adalah 16 bit.
B	String B diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka B bernilai 82. Dengan ASCII Code (Binary) yaitu 01010010 sebanyak 8 bit. Frekuensi string B dalam "JADI AKU SEBENTAR SAJA" sebanyak 1. Setelah itu Bit dikalikan Frekuensi. 8 x 1 = 8. Jadi keseluruhan bit untuk string B adalah 8 bit.
N	String N diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka N bernilai 78. Dengan ASCII Code (Binary) yaitu 01001110 sebanyak 8 bit. Frekuensi string N dalam "JADI AKU SEBENTAR SAJA" sebanyak 1. Setelah itu Bit dikalikan Frekuensi. 8 x 1 = 8. Jadi keseluruhan bit untuk string N adalah 8 bit.

GIT GEMILANG INFORMATIKA

Vol 2, No2, Juli 2024, Hal. 63-71
ISSN 2985-721X(media online)
https://www.journal.hdgi.org/index.php/git/index

T	String T diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka T bernilai 84. Dengan ASCII Code (Binary) yaitu 01010100 sebanyak 8 bit. Frekuensi string T dalam "JADI AKU SEBENTAR SAJA" sebanyak 1. Setelah itu Bit dikalikan Frekuensi. 8 x 1 = 8. Jadi keseluruhan bit untuk string T adalah 8 bit.
R	String R diubah dalam bilangan ASCII maka ASCII Code (Decimal) maka R bernilai 82. Dengan ASCII Code (Binary) yaitu 01010010 sebanyak 8 bit. Frekuensi string R dalam "JADI AKU SEBENTAR SAJA" sebanyak 1. Setelah itu Bit dikalikan Frekuensi. 8 x 1 = 8. Jadi keseluruhan bit untuk string R adalah 8 bit.

Untuk mengukur ukuran teks itu didalam komputer dapat dilihat pada tabel 2:

Karakter	ASCII Code	ASCII Code (Binary)	Bit	Frekuensi	Bit x Frekuensi
J	74	01001010	8	2	16
A	65	01000001	8	5	40
D	68	01000100	8	1	8
I	73	01001001	8	1	8
Spasi	32	00100000	8	3	24
K	75	01001011	8	1	8
U	85	01010101	8	1	8
S	83	01010011	8	2	16
E	69	01000101	8	2	16
B	66	01000010	8	1	8
N	78	01001110	8	1	8
T	84	01010100	8	1	8
R	82	01010010	8	1	8
				Total Bit	200

. Dari Tabel 2 dapat dibentuk *String* bit sebelum dikompresi yaitu

"0100101001000001010001000100100100100000010010110101010100100000010100110100010101000001001001110010101000101010010".

Dengan uraian sebagai berikut :

01001010 01000001 01000100 01001001 00100000 01000001 01001011
J A D I Spasi A K
01010101 00100000 01010011 01000101 01000010 01000101 01001110
U Spasi S E B E N
01010100 01000001 01010010 00100000 01010011 01000001 01001010
T A R Spasi S A J
01000001
A

1. Kompresi Berdasarkan Algoritma *Even-Rodeh Code*

Dibawah ini adalah tabel kode *Even-Rodeh Code* yang akan digunakan pada saat proses kompresi.
Contoh untuk pencarian kode *Even-Rodeh Code* adalah jika
n = 3 maka biner dari n adalah 11, karena panjang bit kurang dari 3 maka ditambahkan 0 didepan nilai biner sehingga menjadi 3 bit yaitu 001.

Tabel 3 Kode *Even-Rodeh Codes*

N	Even-Rodeh
0	000

GIT GEMILANG INFORMATIKA
Vol 2, No2, Juli 2024, Hal. 63-71
ISSN 2985-721X(media online)
https://www.journal.hdgi.org/index.php/git/index

1	001
2	010
3	011
4	100 0
5	101 0
6	110 0
7	111 0
8	100 1000 0
9	100 1001 0
10	100 1010 0
11	100 1011 0
12	100 1100 0

Sebelum dilakukan proses kompresi, terlebih dahulu *string* yang akan dikompresi diurutkan terlebih dahulu dari frekuensi terbesar ke frekuensi terkecil. Setelah diurutkan maka gunakan kode *Even Rodeh* untuk proses kompresi. Berikut adalah proses kompresi dengan metode *Even-Rodeh Code* pada string "JADI AKU SEBENTAR SAJA".
3.Proses kompresi *file* dengan algoritma *Even-Rodeh Code.*

Tabel 4 *String* yang sudah dikompresi dengan *Even-Rodeh*

Karakter	Even Rodeh Code	Bit	Frekuensi	Bit cax Frekuensi
A	000	3	5	15
Spasi	001	3	3	9
J	010	3	2	6
S	011	3	2	6
E	100 0	4	2	8
D	101 0	4	1	4
I	110 0	4	1	4
K	111 0	4	1	4
U	100 1000 0	8	1	8
B	100 1001 0	8	1	8
N	100 1010 0	8	1	8
T	100 1011 0	8	1	8
R	100 1100 0	8	1	8
			Total Bit	96

Berdasarkan tabel 4 maka *string* "JADI AKU SEBENTAR SAJA" setelah dikomperesi menjadi sebagai berikut:
"0100001010110000100011101001000000101110001001001010001001010010010110000100110000010110000100000"
Dengan uraian sebagai berikut :

010 000 1010 1100 001 000 1110 10010000 001 011 1000 10010010 1000
J A D I Sp A K U Sp S E B E
10010100 10010110 000 10011000 001 011 000 010 000
N T A R Sp S A J A

GIT GEMILANG INFORMATIKA

Vol 2, No2, Juli 2024, Hal. 63-71
ISSN 2985-721X(media online)
https://www.journal.hdgi.org/index.php/git/index

Hasil kompresi didapatkan ukuran *string* bit menjadi 96 bit, selanjutnya bit yang didapat dibagi 8. Jika sisa bagi *string* bit adalah 0 maka tidak perlu dilakukan penamahan *padding* dan *flagbits*. Setelah hasil kompresi dibagi 8, maka didapat karakter baru yang menjadi hasil kompresi, berikut uraian hasil kompresi setelah dibagi 8:

01000010	10110000	10001101	01000000	10111000	10010010	10001001
B	°	Ì		ç	©	Æ	ë

01001001	01100001	00110000	01011000	01000000
I	a	0	X	@

Dari hasil kompresi dengan Even-Rodeh Code maka dapat dihitung kinerja kompresinya yaitu:

1. *Ratio of Compression (Rc)*

$$Rc = \frac{Ukuran\ Data\ Sebelum\ Dikompresi}{Ukuran\ Data\ Setelah\ Dikompresi}$$

$$Rc = \frac{176}{96}$$

$$Rc = 1{,}83$$

2. *Compression Ratio (Cr)*

$$Cr = \frac{Ukuran\ Data\ Setelah\ Dikompresi}{Ukuran\ Data\ Sebelum\ Dikompresi}\ x\ 100\%$$

$$Cr = \frac{96}{176}\ x\ 100\%$$

$$Cr = 54{,}54\ \%$$

3. *Redundancy (Rd)*

Rd = 100% - *Cr*
Rd = 100% - 54,54%
Rd = 45,46%

Setelah didapat hasil kompresi, maka berikutnya akan dilakukan proses dekompresi. Dari *string* bit hasil kompresi akan dilakukan pembacaan dari indeks ke-0 sampai indeks ke-n. Indeks ke-0 = "0" maka dilakukan pencocokan pada tabel kode even rodeh, dan tidak ada yang cocok. Karena tidak ada yang cocok, maka pembacaan dilanjutkan dengan menambahkan indeks berikutnya ke indeks sebelumnya, maka indeks ke-0 akan ditambah dengan indeks ke-1 = "01" maka dilakukan pencocokan pada tabel kode *Even Rodeh*, dan tidak ada yang cocok. Karena tidak ada yang cocok, maka pemacaan dilanjutkan dengan menambahkan indeks berikutnya ke indeks sebelumnya yang belum ada kecocokan, maka indeks ke-0 ditambah indeks ke-1 dan ditambah indeks ke-2 = "010" dan dilakukan pencocokan pada tabel kode *Even Rodeh* maka didapat karakter "J". Setelah ditemukan kecocokan pada indeks ke-2 maka pembacaan berikut dimulai dari indeks berikutnya yaitu indeks ke-3 dan begitu seterusnya sampai indeks ke-n sehingga akan menghasilkan *string* awal yaitu "JADI AKU SEBENTAR SAJA". Proses pencocokan dapat dilihat pada tabel 3.5:

010	000	1010	1100	001	000	1110	10010000	001	011	1000	10010010	1000
J	A	D	I	Sp	A	K	U	Sp	S	E	B	E

10010100	10010110	000	10011000	001	011	000	010	000
N	T	A	R	Sp	S	A	J	A

5.KESIMPULAN

Berdasarkan hasil akhir kompresi Lirik Lagu maka penulis menguraikan beberapa kesimpulan. Adapun kesimpulan-kesimpulan tersebut terdiri dari:

1. Algoritma Even-Rodeh Code memiliki langkah-langkah untuk pencarian kode bit yang memiliki minimal 3 bit.
2. Algoritma Even-Rodeh Code sangat cocok digunakan untuk kompresi teks karena teks pada aplikasi lirik lagu memiliki karakter yang berulang-ulang.
3. Aplikasi Lirik Lagu dibangun menggunakan *Eclipse Juno* versi 2.0.

GIT GEMILANG INFORMATIKA
Vol 2, No2, Juli 2024, Hal. 63-71
ISSN 2985-721X(media online)
https://www.journal.hdgi.org/index.php/git/index

Referensi

[1] B. S. Hasugian, "PERANAN KRIPTOGRAFI SEBAGAI KEAMANAN SISTEM INFORMASI PADA USAHA KECIL DAN MENENGAH," *Jurnal Warta*, vol. 53, Juli 2017.

[2] S. and P. D. Atika, "DIGITAL SIGNATURE DENGAN ALGORITMA SHA-1 DAN RSA SEBAGAI AUTENTIKASI," *Jurnal Cendikia*, vol. XVI, Oktober 2018.

[3] M. A. J, P. O. C van and S. A. Vanstone, Handbook of Applied Cryptography, USA: CRC Press, 1996.

[4] H. Jogiyanto, Analisa dan Desain Sistem Informasi: Pendekatan Terstruktur Teori dan Praktik Aplikasi Bisnis, Yogyakarta: ANDI, 2005.

[5] R. Munir, Kriptografi, Yogyakarta: Informatika Bandung, 2006.

[6] M. Hanafi, "Aplikasi Pengajuan Surat Digital Menggunakan Algoritma RSA Pada LPPM UIN SUSKA," Pekan Baru, 2019.