TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

Penerapan Algoritma Arithmetic Coding Untuk Mengkompresi Record Database Pada Aplikasi Ensiklopedia Flora Berbasis Android

Ratiya Tari, Surya Darma Nasution, Taronisokhi Zebua

Fakultas Ilmu Komputer dan Teknologi Informasi, Prodi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: ratiyata3@gmail.com

Abstrak−Penggunaan database dalam sebuah aplikasi dapat mempengaruhi ukuran (size) aplikasi itu sendiri. Ukuran (size) dari database dipengaruhi oleh banyaknya record-record database, dengan hal tersebut diperlukan suatu teknik kompresi untuk mengkompresi record database pada aplikasi ensiklopedia flora. Komperesi dilakukan untuk mengurangi atau mereduksi ukuran suatu data sehingga ukuran data menjadi lebih kecil dan dengan kompresi mempermudah dalam proses transmisi. Banyak algoritma yang dikembangkan dalam proses kompresi, salah satunya yaitu algoritma arithmetic coding. Pengukuran kinerja hasil kompresi dengan menerapan algoritma arithmetic coding dalam mengkompresi record database aplikasi ensiklopedia flora yaitu menghasilkan Ratio of compression (2,2), Compression Ratio (45%), dan Redudancy (55%).

Kata Kunci: Kompresi; Ensiklopedia Flora; Record Database; Arithmetic Coding; Android

Abstract−The use of a database in an application can affect the size of the application itself. The size of the database is influenced by the number of database records, so it requires a compression technique to compress database records in the flora encyclopedia application. Completion is done to reduce or reduce the size of a data so that the data size becomes smaller and with compression makes it easier to transmit. Many algorithms are developed in the compression process, one of which is the arithmetic coding algorithm. Performance measurement of compression results by applying the arithmetic coding algorithm in compressing the flora encyclopedia application database records, namely producing Ratio of compression (2.2), Compression Ratio (45%), and Redundancy (55%).

Keywords: Compression; Flora Encyclopedia; Record Database; Arithmetic Coding; Android

1. PENDAHULUAN

Database secara umum dapat diartikan sebagai kumpulan dari berbagai macam data. Data tersebut dapat berupa text, gambar, suara, video dan berbagai multimedia lainya. *Database* sangat bermanfaat untuk mengatasi berbagai masalah yang sering terjadi dalam penyusunan data. Penggunaan *database* dalam sebuah aplikasi dapat mempengaruhi ukuran (*size*) aplikasi itu sendiri. Aplikasi yang menggunakan *database* bertujuan untuk mengelompokkan data dalam mempermudah identifikasi data. Semakin banyak data-data di dalam *database*, maka semakin besar pula ukuran aplikasinya. Ukuran (*size*) dari *database* dipengaruhi oleh banyaknya *record-record database* sehingga dengan hal tersebut diperlukan suatu teknik kompresi untuk mengkompresi *record database* nya [1]. Aplikasi yang memiliki kapasitas besar dapat membutuhkan ruang penyimpanan yang besar dan kompresi *record database* sangat membantu dalam mengatasi redudansi data, mempercepat akses data dan dapat mengurangi ukuran sebuah aplikasi [2].

Kompresi merupakan proses pengubahan data yang memperkecil atau memadatkan ukuran berupa kumpulan karakter menjadi bentuk kode sehingga hanya memerlukan ruang penyimpanan yang lebih kecil dan lebih efisien dalam waktu transmisi data [3]. Teknik kompresi memiliki dua sifat, yaitu bersifat *lossless compression* dan lossy *compression*. *Lossy compression* yaitu teknik kompresi dimana data hasil dekompresi tidak sama dengan data sebelum kompresi, biasanya teknik ini membuang bagian-bagian data yang sebenarnya tidak begitu berguna, tetapi tidak begitu dirasakan. Sedangkan *lossless compression* merupakan teknik kompresi dimana data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi [4]. Adapun algoritma yang bersifat *lossless compression*, salah satunya yaitu *arithmetic coding*.

Algoritma *arithmetic coding* merupakan teknik kompresi bersifat *lossless*, yang artinya menghasilkan data yang identik dengan data aslinya dan data tidak berubah atau hilang pada saat proses kompresi atau dekompresi [5]. Algoritma *arithmetic coding* melakukan penggantian satu deretan *symbol input* dengan sebuah bilangan *floating point*. Semakin panjang dan semakin kompleks pesan yang dikodekan, akan semakin banyak bit yang diperlukan untuk keperluan tersebut. *Output* dari pengkodean *aritmathic coding* adalah satu angka yang lebih kecil dari angka 1 dan lebih besar atau sama dengan 0. Angka ini secara unik dapat di *decoding* sehingga output tersebut, tiap simbol yang akan di *encoding* diberi satu set nilai probabilita [4].

Berdasarkan penelitian terdahulu bahwa hasil kompresi dengan menerapkan algoritma *arithmetic coding* sangat baik diimplementasikan untuk file citra karena meghasilkan rasio yang cukup besar. Selain itu, waktu kompresi dan dekompresi tidak memerlukan waktu yang lama. Kemudian yang paling penting hasil dekompresi file sama seperti semula dikarenakan algoritma *arithmetic coding* bersifat *lossless* [6]. Penelitian lainnya mengatakan bahwa algoritma *arithmetic coding* dapat diimplementasikan pada proses kompresi dan dekompresi file citra berekstensi *.bmp, dan tingkat rasio kompresi sebuah citra tidak dipengaruhi oleh ukuran dimensi citra, melainkan tergantung pada banyak atau sedikitnya komposisi warna citra yang bersangkutan [4].

Penelitian ini menguraikan tentang proses kompresi dan dekompresi *record database* pada aplikasi ensiklopedia *flora* berbasis android berdasarkan algoritma *arithmetic coding*, sehingga memori yang digunakan untuk menyimpan aplikasi menjadi lebih kecil dan aplikasinya dapat berjalan secara optimal di *smartphone*. Aplikasi ensiklopedia *flora* perlu dikompresi karena didalam aplikasi terdapat banyak data-data yang disimpan dalam *database*. Data dalam aplikasi

TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

ensiklopedia *flora* terdapat data teks dan data gambar, dengan hal tersebut maka aplikasi ensiklopedia flora perlu dikompresi agar aplikasi memiliki *size* kecil. Aplikasi yang memiliki ukuran (*size*) kecil jika dijalankan di *smartphone* dapat berjalan lebih lancar dibandingkan dengan aplikasi yang memiliki ukuran (*size*) besar. Pengoptimalan aplikasi dengan melakukan pemampatan dalam *database* dapat dilakukan dengan menerapkan suatu algoritma.

2. METODOLOGI PENELITIAN

2.1 Kompresi Data

Kompresi data merupakan suatu proses mereduksi ukuran suatu data dengan mengubah sekumpulan data menjadi sekumpulan kode yang dapat menghemat kebutuhan tempat penyimpanan dan waktu untuk transmisi data [4]. Tujuan kompresi data adalah untuk mengurangi atau mereduksi ukuran suatu data sehingga ukuran data menjadi lebih kecil dan lebih ringan dalam proses transmisi. Kompresi data merepresentasikan data digital dengan sesedikit bit, akan tetapi juga tetap mempertahankan kebutuhan minimum untuk membentuk kembali data aslinya [2].

Salah satu yang membutuhkan penerapan teknik kompresi adalah *record database*. Kompresi data pada *record database* sangat membantu dalam mengatasi redudansi data, mempercepat akses data dan dapat mengurangi ukuran sebuah aplikasi [2]. Pemanfaatan kompresi juga mempengaruhi kapasitas suatu data dengan meminimalisir data, sehingga ruang penyimpanan tidak menjadi cepat penuh dan mempercepat dalam proses transfer data [7].

2.2 Algoritma Arithmetic Coding

Algoritma *arithmetic coding* melakukan penggantian satu deretan simbol input dengan sebuah bilangan *floating point*. Semakin panjang dan semakin kompleks pesan yang dikodekan, akan semakin banyak bit yang diperlukan untuk keperluan tersebut. *Output* dari pengkodean *arithmetic coding* adalah satu angka yang lebih kecil dari angka 1 dan lebih besar atau sama dengan 0. Angka ini secara unik dapat di *decoding*, tiap simbol yang akan di *encoding* diberi satu set nilai probabilitas. *Arithmetic coding* dapat juga didefenisikan sebagai suatu bagian dari *entropy encoding* yang mengkonversi suatu data ke dalam bentuk data yang lain dengan lebih sering menggunakan sedikit bit dan jarang menggunakan lebih banyak bit karakter [4].

Algoritma *arithmetic coding* menggunakan dua variabel *low* dan *high* untuk mendefinisikan *interval* [*low*, *high*). Berikut ini proses *encoding* dari algoritma *arithmetic coding* [6]:

1. *Set low* = 0,0 (kondisi awal)
2. *Set high* = 1.0 (kondisi awal)
3. *While* (simbol *input* masih ada)
4. Ambil simbol *input*
5. CR = *high – low*
6. *High = low* + CR * *high_range (symbol)*
7. *Low = low* + CR* low_range (*symbol*)
8. *End while*
9. Cetak *low*

Proses *decoding* dari algoritma *arithmetic coding* [6], sebagai berikut :

1. Ambil *encoded-symbol* (ES)
2. *Do*
3. Cari *range* dari *symbol* yang melingkupi ES
4. Cetak *symbol*
5. RC = *high_range-low_range*
6. ES = ES-*low_range*
7. ES = ES / CR
8. *Until symbol* habis

3. HASIL DAN PEMBAHASAN

Database digunakan untuk menyimpan atau mengelolah data untuk menghasilakn sebuah informasi pada sebuah apliaksi. Informasi atau data ini disebut *record database*, semakin banyak *record database* maka semakin besar pula ukuran (*size*) *database*. *Database* juga dapat mempengaruhi ukuran aplikasi, sehingga memerlukan ruang penyimpanan yang besar. Aplikasi yang memiliki ukuran (*size*) besar selain memerlukan ruang penyimpanan yang besar, juga dapat mempengaruhi jalanya aplikasi di *smartphone*. Oleh karena itu, maka diperlukanlah suatu proses kompresi untuk memampatkan data didalam *database*. Proses kompresi *database* dapat dilakukan dengan mengkompresi *record database*-nya. Pengkompresian *record database* sangat perlu dilakukan untuk mengurangi ukuran suatu *database*. Bila proses kompresi *record database* tidak dilakukan, maka akan terjadi masalah redudansi data, kecepatan akses yang lambat dan ukuran sebuah aplikasi yang besar karena menggunakan *database*.

Record database yang dikompres dalam penelitian ini adalah *record database* aplikasi ensiklopedia *flora*. Aplikasi ini dibangun dengan menggunakan *database* untuk mengelompokkan data atau *file* terkait *flora*. Data dalam aplikasi

TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

ensiklopedia *flora* dapat berbentuk data teks dan data gambar. Pengkompresian ini dilakukan selain untuk mengoptimalkan aplikasi di *smartphone* juga mengurangi ukuran (*size*) aplikasi nya. Kompresi *record database* sangat membantu dalam mengatasi redudansi data, mempercepat akses data dan dapat mengurangi ukuran sebuah aplikasi.

Konsep penerapan kompresi *record database* di aplikasi ensiklopedia *flora* yaitu melakukan kompresi *record database* yang *file* gambar dengan menerapkan algoritma *arithmetic coding*. Pengaplikasian kompresi dan dekompresi data yang akan dilakukan hanya menampilkan proses kerja dari algoritma *arithmetic coding*, agar dapat mengetahui bagaimana proses dari algoritma *arithmetic coding.*

Aplikasi ensiklopedia *flora* memiliki data yang disimpan dalam *database*, yang mana *database* memiliki *record database* sehingga nantinya akan dilakukan proses kompresi dan dekompresi *record database*.

3.1 Penerapan Arithmetic Coding Pada Kompresi Database

Aplikasi ensiklopedia *flora* memiliki *database* untuk menyimpan *file.* Salah satu isi dari *record database* yaitu berupa *file* gambar. *Record database* inilah yang nanti akan di kompresi dengan algoritma *arithmetic coding*.

Kompresi *record database* yang berupa *file* gambar memerlukan aplikasi *binary viewer* untuk membaca nilai *heksa* dari *file* gambar yang akan disimpan menjadi *record database* aplikasi ensiklopedia.

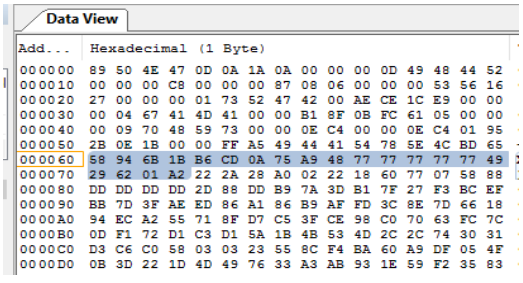
Dapat dijelaskan bahwa saat melakukan kompresi *record database* yang berbentuk file gambar, terlebih dahulu dilakukan pengubahan gambar tersbut menjadi nilai *heksa*. Nilai *heksa* dari *file* gambar didapatkan dengan menggunakan aplikasi *binary viewer*, kemudian ambil sampel nilai *heksa* dari *file* gambar tersebut. Nilai *heksa* inilah yang kemudian dilakukan proses kompresi dengan menerapkan algoritma *arithmetic coding*. Setelah mendapatkan hasil proses kompresi, selanjutnya melakukan proses dekompresi terhadap hasil kompresi.

Dapat dilihat dari penjelasan tentang proses kompresi *record database file* gambar dengan algoritma *arithmetic coding* bahwa proses kompresi dilakukan di luar aplikasi. Kompresi dan dekompresi dilakukan untuk melihat bagaimana proses dari penerapan algoritma *arithmetic coding.*

seperti yang dijelaskan di atas yaitu *record database* berupa *file* gambar. Proses kompresi *record database file* gambar memerlukan nilai *heksa* dalam proses kompresi, untuk mempermudah kompresi menggunakan algoritma *arithmetic coding*. Mendapatkan nilai ASCII yang *heksa* menggunakan aplikasi *binary viewer*. Berikut ini merupakan *file* gambar dari *record database* yang akan dikompresi :

Gambar 1. *File* Gambar *Flora* dari *Record Database*

Setelah menentukan *file* gambar dari *record database* yang akan dikompresi, maka selanjutnya mencari nilai heksa dengan menggunakan bantuan aplikasi *binary viewer*, sebagai berikut :



Gambar 2. Nilai *Heksa File* Gambar Menggunakan *Binary Viewer*

Berdasarkan gambar di atas dapat dilihat nilai *heksa* atau dapat disebut simbol, bahwa *sample* yang diambil terdapat dua puluh nilai heksa yaitu “58 94 6B 1B B6 CD 0A 75 A9 48 77 77 77 77 77 49 29 62 01 A2”. Data inilah yang akan dikompresi dengan menggunakan algoritma *arithmetic coding.*

1. Proses Kompresi
   Berikut ini merupakan langkah-langkah untuk kompresi dengan algoritma *arithmetic coding* :
   a. Membuat daftar frekuensi kemunculan tiap-tiap simbol
      Daftar frekuensi kemunculan dari setiap simbol dibuat dengan menghitung berapa kali kemunculan nilai-nilai *heksa* yang sama. Berdasarkan nilai-nilai *heksa* citra yang telah ditentukan untuk dikompresi

TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

Tabel 1. Pendataan Simbol *Arithmetic Coding*

No	*Heksa*	Frekuensi
1	58	1
2	94	1
3	6B	1
4	1B	1
5	B6	1
6	CD	1
7	0A	1
8	75	1
9	A9	1
10	48	1
11	77	5
12	49	1
13	29	1
14	62	1
15	01	1
16	A2	1
	Jumlah	20

b. Menghitung probabilitas frekuensi kemunculan setiap simbol
Nilai-nilai probabilitas frekuensi didapatkan dengan membagi nilai frekuensi kemunculan setiap nilai *heksa* dari gambar dengan banyaknya keseluruhan nilai *heksa*. Keseluruhan nilai *heksa* yaitu dua puluh nilai *heksa*. Berikut ini tabel meghitung probabilitas frekuensi.

Tabel 2. Probabilitas Frekuensi Kemunculan Simbol

No	*Heksa*	Frekuensi	Probabilitas
1	58	1	1/20 = 0,05
2	94	1	1/20 = 0,05
3	6B	1	1/20 = 0,05
4	1B	1	1/20 = 0,05
5	B6	1	1/20 = 0,05
6	CD	1	1/20 = 0,05
7	0A	1	1/20 = 0,05
8	75	1	1/20 = 0,05
9	A9	1	1/20 = 0,05
10	48	1	1/20 = 0,05
11	77	5	5/20 = 0,25
12	49	1	1/20 = 0,05
13	29	1	1/20 = 0,05
14	62	1	1/20 = 0,05
15	01	1	1/20 = 0,05
16	A2	1	1/20 = 0,05
	Jumlah	20	1.0

c. Menghitung jangkauan setiap simbol, dengan titik terendah (*low*) adalah 0,0 dan titik tertinggi (*high*) adalah 1,0. Nilai *high* diberikan pada nilai *heksa* yang pertama, kemudian nilai *low* pada nilai *heksa* terakhir. Setelah menentukan nilai *high* dan *low*, maka selanjutnya melakukan perhitungan secara berurut yaitu dengan cara nilai *high* (1.0) pada *heksa* pertama dikurang probabilitas. Hasil pengurangan merupakan nilai *low* di *heksa* pertama dan nilai *high* di *heksa* kedua, begitu perhitungan seterusnya sampai pada *heksa* yang terakhir. Perhitungan jangkauan setiap simbol dapat dilihat pada tabel 3.

Tabel 3. Jangkauan Setiap *Range*

No	*Heksa*	Frekuensi	Probabilitas	Jangkauan *Range*	
				Low	*High*
1	58	1	1/20 = 0,05	0,95	1,0
2	94	1	1/20 = 0,05	0,9	0,95
3	6B	1	1/20 = 0,05	0,85	0,9
4	1B	1	1/20 = 0,05	0,8	0,85
5	B6	1	1/20 = 0,05	0,75	0,8
6	CD	1	1/20 = 0,05	0,7	0,75
7	0A	1	1/20 = 0,05	0,65	0,7

TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

No	*Heksa*	Frekuensi	Probabilitas	Jangkauan *Range*	
				Low	*High*
8	75	1	1/20 = 0,05	0,6	0,65
9	A9	1	1/20 = 0,05	0,55	0,6
10	48	1	1/20 = 0,05	0,5	0,55
11	77	5	5/20 = 0,25	0,25	0,5
12	49	1	1/20 = 0,05	0,2	0,25
13	29	1	1/20 = 0,05	0,15	0,2
14	62	1	1/20 = 0,05	0,1	0,15
15	01	1	1/20 = 0,05	0,05	0,1
16	A2	1	1/20 = 0,05	0,0	0,05
	Jumlah	20	1.0		

d. Diinisialisasi nilai titik terendah (*low*) sebagai 0,0 dan titik tertinggi (*high*) sebagai 1,0. Kemudian dilakukan perhitungan *low* dan *high* sesuai data setiap karakter dengan rumus kompresi *arithmetic coding*, berikut ini *proseudocode* algoritma *arithmetic coding* :

Algoritma *Arithmeric Coding* :
Set *low* = 0.0
Set *high* = 1.0
While (simbol *input* masih ada) *do*
Ambil simbol input
Coderange = high – low
*High = low + Coderange *high_range* (*symbol*)
*Low = low + Coderange *low_range* (*symbol*)
End While
Cetak *Output* dari proses *arithmetic coding*.

Kompresi *arithmetic coding* dilakukan dengan mengelompokkan dua nilai *heksa* untuk mempermudah mencari *low* dan *high*. Nilai *low* dan *high* yang diambil adalah perhitungan yang kedua yaitu (0,995-0,9975). Perhitungan nilai kedua diambil dikarenakan melakukan pengelompokkan nilai *heksa*. Nilai perhitungan kedua inilah yang nanti dilakukan *rescaling* untuk mencari nilai *biner*.

Tabel 4. Kompresi *Arithmetic Coding*

Hexsadesimal		*Low / High*	
	Low	0,0 + (1,0 – 0,0) * 0,95	0,95
58	*High*	0,0 + (1,0 – 0,0) * 1,0	1,0
94	*Low*	0,95 + (1,0 – 0,95) * 0,9	0,995
	High	0,95 + (1,0 – 0,95) * 0,95	0,9975
	Low	0,0 + (1,0 – 0,0) * 0,85	0,85
6B	*High*	0,0 + (1,0 – 0,0) * 0,9	0,9
1B	*Low*	0,85 + (0,9 – 0,85) * 0,8	0,89
	High	0,85 + (0,9 – 0,85) * 0,85	0,8925
	Low	0,0 + (1,0 – 0,0) * 0,75	0,75
B6	*High*	0,0 + (1,0 – 0,0) * 0,8	0,8
CD	*Low*	0,75 + (0,8 – 0,75) * 0,7	0,785
	High	0,75 + (0,8 – 0,75) * 0,75	0,7875
	Low	0,0 + (1,0 – 0,0) * 0,65	0,65
0A	*High*	0,0 + (1,0 – 0,0) * 0,7	0,7
75	*Low*	0,65 + (0,7 – 0,65) * 0,6	0,68
	High	0,65 + (0,7 – 0,65) * 0,65	0,6825
	Low	0,0 + (1,0 – 0,0) * 0,55	0,55
A9	*High*	0,0 + (1,0 – 0,0) * 0,6	0,6
48	*Low*	0,55 + (0,6 – 0,55) * 0,5	0,575
	High	0,55 + (0,6 – 0,55) * 0,55	0,5775
	Low	0,0 + (1,0 – 0,0) * 0,25	0,25
77	*High*	0,0 + (1,0 – 0,0) * 0,5	0,5
77	*Low*	0,25 + (0,5 – 0,25) * 0,25	0,3125
	High	0,25 + (0,5 – 0,25) * 0,5	0,375
	Low	0,0 + (1,0 – 0,0) * 0,25	0,25
77	*High*	0,0 + (1,0 – 0,0) * 0,5	0,5
77	*Low*	0,25 + (0,5 – 0,25) * 0,25	0,3125
	High	0,25 + (0,5 – 0,25) * 0,5	0,375
	Low	0,0 + (1,0 – 0,0) * 0,25	0,25

TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

Hexsadesimal		*Low / High*	
77	*High*	0,0 + (1,0 – 0,0) * 0,5	0,5
49	*Low*	0,25 + (0,5 – 0,25) * 0,2	0,3
	High	0,25 + (0,5 – 0,25) * 0,25	0,3125
	Low	0,0 + (1,0 – 0,0) * 0,15	0,15
29	*High*	0,0 + (1,0 – 0,0) * 0,2	0,2
62	*Low*	0,15 + (0,2 – 0,15) * 0,1	0,155
	High	0,15 + (0,2 – 0,15) * 0,15	0,1575
	Low	0,0 + (1,0 – 0,0) * 0,05	0,05
01	*High*	0,0 + (1,0 – 0,0) * 0,1	0,1
A2	*Low*	0,05 + (0,1 – 0,05) * 0,0	0,05
	High	0,05 + (0,1 – 0,05) * 0,05	0,0525

Representasi dari nilai *heksa* "58 94 6B 1B B6 CD 0A 75 A9 48 77 77 77 77 77 49 29 62 01 A2" yang telah dimampatkan diambil nilai *biner* yang berada di antara (0,995 – 0,9975), (0,89 – 0,8925), (0,785 – 0,7875), (0,68 - 0,6825), (0,575 - 0,5775), (0,3125 – 0,375), (0,3125 – 0,375), (0,3 – 0,3125), (0,155 - 0,1575), (0,05 - 0,0525).

Setelah mendapatkan nilai *low* dan *high*, maka selanjutnya mencari nilai *biner* antara. Nilai *biner* antara dilakukan untuk mencari nilai *biner* di antara penggabungan dua nilai *heksa* pada tabel 3.4.

Nilai *biner* didapatkan dengan cara melakukan *rescaling* pada nilai *heksa*. Cara melakukan *rescaling* dapat dilakukan dengan membuat tabel untuk mempermudah mendapatkan nilai *biner*. Pertama menentukan nilai *low* dan *high* dari *heksa* yang dipilih (0,995 – 0,9975). Sesuai dengan algoritma *arithmetic coding* bahwa nilai *low* 0,0 dan nilai *high* 1,0 ini merupakan ketentuan mutlak. Buat diantara 1,0 dan 0,0 simbol X, simbol X merupakan nilai tengah antara 1,0 dan 0,0 dapat dilihat pada tabel 3.5. Kemudian buat penyataan bahwa X>=*low* dan X<=*high* (*low* dan *high* disini merupakan hasil nilai heksa 0,995-0,9975), hal ini merupakan acuan bahwa X>=*low* dan X<=*high* harus bernilai *true* untuk menyelesaikan perhitungannya. Cara mendapatkan nilai tengah (X) yaitu nilai (*high-low*)/2+*low*, jika pernyataan X>=*low* bernilai *false* dan X<=*high* bernilai *true* maka nilai biner 1. Sebaliknya jika pernyataan X>=*low* bernilai *true* dan X<=*high* bernilai *false* maka nilai biner 0.

Nilai *biner* yang bernilai 1, maka nilai *high* tetap sama dengan *high* sebelumnya dan nilai *low* nya sama dengan nilai X pada perhitungan sebelumnya, dapat dilihat pada tabel 3.5 di kolom 2 dan 3 (dihitung mulai dari kolom berwarna). Nilai *biner* yang bernilai 0, maka nilai *high* nya sama dengan nilai X sebelumnya dan nilai *low* nya sama dengan *low* sebelumnya, dapat dilihat pada tabel 3.6 kolom 5 dan 6. Kemudian jika penyataan X>=*low* dan X<=*high* sudah bernilai *true* maka untuk menentukan nilai *biner* nya yaitu melihat nilai selisih terendah antara *high* dan *low*, jika nilai selisih *high* yang terendah maka nilai *biner* 1 dan jika sebaliknya maka nilai *biner* 0. Nilai selisih terletak diakhir perhitungan, lihat tabel 3.5 yaitu (0,0025 dan 0,002815). Nilai selisih *high* didapat dari pengurangan nilai *high* terakhir dengan nilai *high* dari *heksa* yaitu (1,0 - 0,9975= 0,0025). Selisih nilai *low* didapat dari pengurangan nilai *low* yang *heksa* dengan nilai *low* terakhir yaitu (0,995 - 0,9921875= 0,002815). Lihat tabel 3.5 sebagai berikut :

a. Biner antara (0,995 – 0,9975)

Tabel 5. *Rescaling* Nilai *Heksa* 58 dan 94

Hexsa	Low	High							
	0,995	0,9975							
		1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
		X	0,5	0,75	0,875	0,9375	0,96875	0,984375	0.9921875
		0,0	0,0	0,5	0,75	0,875	0,9375	0,96875	0,984375
		X>=0,995	False	False	False	False	False	False	False
58		X<=0,9975	True	True	True	True	True	True	True
94			1	1	1	1	1	1	1
		1,0	1,0		0,0025				
		X	0,99609375						
		0,0	0,9921875		0,002815				
		X>=0,995	True						
		X<=0,9975	True						
			1		11111111				

b. Biner antara (0,89 – 0,8925)

Tabel 6. *Rescaling* Nilai *Heksa* 6B dan 1B

Hexsa	Low	High							
	0,89	0,8925							
6B		1,0	1,0	1,0	1,0	1,0	0,9375	0,90625	0,01375
1B		X	0,5	0,75	0,875	0,9375	0.90625	0,890625	
		0,0	0,0	0,5	0,75	0,875	0,875	0,875	0,015

TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

	X>=0,89	False	False	False	True	True	True	
	X<=0,8925	True	True	True	False	False	True	
		1	1	1	0	0	1	111001

c. Biner antara (0,785 – 0,7875)

Tabel 7. *Rescaling* Nilai *Heksa* B6 dan CD

Hexsa	Low	High						
	0,785	0,7875						
		1,0	1,0	1,0	1,0	0,875	0,8125	0,8125
		X	0,5	0,75	0,875	0,8125	0,78125	0,796875
		0,0	0,0	0,5	0,75	0,75	0,75	0,78125
		X>=0,785	False	False	True	True	False	True
B6		X<=0,7875	True	True	False	False	True	False
CD			1	1	0	0	1	0
		1,0	0,796875	0,789063	0,001563			
		X	0,789063	0,785157				
		0,0	0,78125	0,78125	0,00375			
		X>=0,785	True	True				
		X<=0,7875	False	True				
			0	1	11001001			

d. Biner antara (0,68 – 0,6825)

Tabel 8. *Rescaling* Nilai *Heksa* 0A dan 75

Hexsa	Low	High							
	0,68	0,6825							
		1,0	1,0	1,0	0,75	0,75	0,6875	0,6875	0,6875
		X	0,5	0,75	0,625	0,6875	0,65625	0,671875	0,6796875
		0,0	0,0	0,5	0,5	0,625	0,625	0,65625	0,671875
		X>=0,68	False	True	False	True	False	False	False
0A		X<=0,6825	True	False	True	False	True	True	True
75			1	0	1	0	1	1	1
		1,0	0,6875		0,68359375		0,00109375		
		X	0,68359375		0,681640625				
		0,0	0,6796875		0,6796875		0,0003125		
		X>=0,68	True		True				
		X<=0,6825	False		True				
			0		0		101011100		

e. Biner antara (0,575 – 0,5775)

Tabel 9. *Rescaling* Nilai *Heksa* A9 dan 48

Hexsa	Low	High							
	0,575	0,5775							
		1,0	1,0	1,0	0,75	0,625	0,625	0,59375	0,578125
		X	0,5	0,75	0,625	0,5625	0,59375	0,578125	0,5703125
		0,0	0,0	0,5	0,5	0,5	0,5625	0,5625	0,5625
		X>=0,575	False	True	True	False	True	True	False
A9		X<=0,5775	True	False	False	True	False	False	True
48			1	0	0	1	0	0	1
		1,0	0,578125		0,578125		0,000625		
		X	0,57421875		0,57617188				
		0,0	0,5703125		0,57421875		0,00078125		
		X>=0,575	False		True				
		X<=0,5775	True		True				
			1		1		100100111		

f. Biner antara (0,3125-0,375)

Tabel 10. *Rescaling* Nilai *Heksa* 77 dan 77

Hexsa	Low	High
77	0,3125	0,375

TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

77		1,0	1,0	0,5	0,5	0,125
		X	0,5	0,25	0,375	
		0,0	0,0	0,0	0,25	0,0625
		X>=0,3125	True	False	True	
		X<=0,375	False	True	True	
			0	1	0	010

g. Biner antara (0,3125-0,375)

Tabel 11. *Rescaling* Nilai *Heksa* 77 dan 77

Hexsa	Low	High				
	0,3125	0,375				
		1,0	1,0	0,5	0,5	0,125
		X	0,5	0,25	0,375	
77		0,0	0,0	0,0	0,25	0,0625
77		X>=0,3125	True	False	True	
		X<=0,375	False	True	True	
			0	1	0	010

h. Biner antara (0,3-0,3125)

Tabel 12. *Rescaling* Nilai *Heksa* 77 dan 49

Hexsa	Low	High					
	0,3	0,3125					
		1,0	1,0	0,5	0,5	0,375	0,0625
		X	0,5	0,25	0,375	0,3125	
77		0,0	0,0	0,0	0,25	0,25	0,05
49		X>=0,3	True	False	True	True	
		X<=0,3125	False	True	False	True	
			0	1	0	0	0100

i. Biner antara (0,155-0,1575)

Tabel 13. *Rescaling* Nilai *Heksa* 29 dan 62

Hexsa	Low	High						
	0,155	0,1575						
		1,0	1,0	0,5	0,25	0,25	0,1875	0,03
		X	0,5	0,25	0,125	0,1875	0,15625	
29		0,0	0,0	0,0	0,0	0,125	0,125	0,03
62		X>=0,155	True	True	False	True	True	
		X<=0,1575	False	False	True	False	True	
			0	0	1	0	1	00101

j. Biner antara (0,05-0,525)

Tabel 14. *Rescaling* Nilai *Heksa* 01 dan A2

Hexsa	Low	High							
	0,05	0,0525							
		1,0	1,0	0,5	0,25	0,125	0,0625	0,0625	0,0625
		X	0,5	0,25	0,125	0,0625	0,03125	0,046875	0,0546875
		0,0	0,0	0,0	0,0	0,0	0,0	0,03125	0,046875
		X>=0,05	True	True	True	True	False	False	True
01		X<=0,0525	False	False	False	False	True	True	False
A2			0	0	0	0	1	1	0
	1,0	0,0546875	0,0021875						
	X	0,05078125							
	0,0	0,046875	0,003125						
	X>=0,05	True							
	X<=0,0525	True							
		1	00001101						

Perhitungan di atas berhenti saat didapatkan nilai yang lebih kecil atau sama dengan *low* dan lebih besar atau sama dengan *high*, yaitu (0,995 – 0,9975) dengan nilai biner 11111111; (0,89 – 0,8925) dengan nilai biner 111001; (0,785 –

TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

0,7875) dengan nilai biner 11001001; (0,68 - 0,6825) dengan nilai biner 101011100; (0,575 - 0,5775) dengan nilai biner 100100111; (0,3125 – 0,375) dengan nilai biner 010; (0,3125 – 0,375) dengan nilai biner 010; (0,3 – 0,3125) dengan nilai biner 0100; (0,155 - 0,1575) dengan nilai biner 00101; (0,05 - 0,0525) dengan nilai biner 00001101. Bilangan biner digabungkan menjadi satu-kesatuan sehingga menjadi 1111111111001110010011010111001001001110100100100001010000110 1.

Berdasarkan proses di atas, maka nilai *heksa* "58 94 6B 1B B6 CD 0A 75 A9 48 77 77 77 77 77 49 29 62 01 A2" dari citra setelah dikompresi, dapat direpresentasikan menjadi 1111111111001110010011010111001001001110100 10010010000101000011 01.

Berdasarkan perhitungan dengan algoritma *Arithmetic Coding string bit* diatas berjumlah 63 bit, maka selanjutnya hasil tersebut dibagi menjadi 8 bit setiap kelompok. Dengan total 63 bit, tidak habis dibagi delapan dan menyisakan 7, atau dengan kata lain 63 Mod 8 = 7. Nyatakan hasil bagi tersebut dengan "n". Selanjutnya menambahkan *padding* dan *flagging.*

Lalu masukan rumus berikut untuk menambahkan *padding*:

7 - n + "1"

7 - 7 + "1" = 1

Masukan rumus berikut untuk menambahkan *flagging*:

9 - n

9 - 7 = 2 =00000010

Total panjang bit keseluruhan setelah ada penambahan *padding* dan *flagging* adalah 63+1+8=72 bit.

1111111111001110010011010111001001001110100100100001010000110 1**100000010**

Setelah itu langkah berikutnya membagi *string* bit menjadi per 8 bit, kemudian merubahnya menjadi karakter.

Tabel 15. Merubah Bilangan Biner ke Karakter

No	Biner	Desimal	Karakter
1	11111111	255	ÿ
2	11100111	231	ç
3	00100110	38	&
4	10111001	185	¹
5	00100111	39	'
6	01001001	73	I
7	00001010	10	""
8	00011011	27	
9	00000010	2	┐

Hasil dari kompresi dengan mengambil sampel 20 nilai *heksa*, dapat dilihat pada gambar dibawah ini :

ÿç&¹'I""

Gambar 3. Hasil *String bit* Kompresi *Arithmetic Coding*

2. Pengukuran Kinerja Kompresi

Selanjutnya menghitung pengukuran analisis kinerja kompresi data. Total bit yang diperoleh sebelum dikompresi adalah sebanyak 160 bit dan ukuran setelah dikompresi adalah sebanyak 72 bit, maka kinerja dari kompresi data sebagai berikut :

a. *Ratio of compression* (Rc)

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

$$Rc = \frac{160}{72}$$

$$Rc = 2{,}2$$

b. *Compression Ratio* (Cr)

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

$$Cr = \frac{72}{160}\ x100\%$$

$$Cr = 45\ \%$$

c. *Redudancy* (Rd)

Rd = 100 % - Cr

Rd = 100 % - 45 %

Rd = 55 %

Berdasarkan hasil pengukuran kinerja kompresi, bahwa Rc 2, menunjukkan hasil dari ukuran data sebelum dikompresi dan setelah dikompresi mengalami pengurangan *ratio* gambar yaitu 2,2. *Compression ratio* (*C*r) menunjukan *persentase* dari data yang telah terkompresi dengan data yang belum di kompresi dan dinyatakan dengan persen, yaitu

TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

45%. *Redudancy* menunjukkan kelebihan yang terdapat didalam data sebelum dikompresi. Hasil dari *redundancy* yaitu selisih antara ukuran data sebelum dikompresi dengan hasil *compression ratio*.

3.2 Dekompresi Record Database dengan Algoritma Arithmetic Coding

Proses dekompresi dilakukan setelah melakukan proses kompresi terlebih dahulu. Hasil dari proses kompresi yang telah dilakukan yaitu berupa karakter aneh atau dapat dilihat pada tabel 3.15. Karakter aneh tersebut dirubah kembali menjadi bilangan *biner* untuk melakukan proses dekompresi. Tabel 3.16 menunjukkan hasil pengubahan karakter terkompresi menjadi desimal dan biner.

Tabel 16. Merubah Karakter ke Biner

No	Karakter	Desimal	Biner
1	ÿ	255	11111111
2	ç	231	11100111
3	&	38	00100110
4	¹	185	10111001
5	'	39	00100111
6	I	73	01001001
7	“”	10	00001010
8		27	00011011
9		2	00000010

Berikut ini total keseluruhan bit dari tabel merubah karakter ke *string* bit “ 1111111111100111001001101011100100100111010010010000101000011011000000010”. Selanjutnya menghilangkan *padding* dan *flagging*, untuk menghilangkan *padding* dan *flagging* dengan cara mengambil 8 bit terakhir dan merubahnya menjadi bilangan desimal, kemudian nyatakan dengan “n”.

n = **00000010** = 2

Kemudian gunakan rumus “7+n” seperti berikut :

7 + n = 7 + 2 = 9

Hilangkan dari string bit sebanyak 9 bit terakhir.

Berdasarkan dari langkah-langkah untuk menghilangkan *padding* dan *flagging*, maka jumlah *string bit* yaitu berjumlah 63 bit seperti diawal, yaitu :

“111111111100111001001101011100100100111010010010000101000011101”

Dilakukan pembacaan *string bit* dari kiri ke kanan lalu ikuti tabel *rescaling* hingga menemukan nilai *biner* antara. Adapun tabel pengecekan bit adalah sebagai berikut :

Tabel 17. Pengecekan *bit*

No.	**Nilai**	**Keterangan**
1	1	Tidak Ada
2	11	Tidak Ada
3	111	Tidak Ada
4	1111	Tidak Ada
5	11111	Tidak Ada
6	111111	Tidak Ada
7	1111111	Tidak Ada
8	11111111	Biner antara (0,995-0,9975)
9	1	Tidak Ada
10	11	Tidak Ada
11	111	Tidak Ada
12	1110	Tidak Ada
13	11100	Tidak Ada
14	111001	Biner antara (0,89-0,8925)
15	1	Tidak Ada
16	11	Tidak Ada
17	110	Tidak Ada
18	1100	Tidak Ada
19	11001	Tidak Ada
20	110010	Tidak Ada
21	1100100	Tidak Ada
22	11001001	Biner antara (0,785-0,7875)
23	1	Tidak Ada
24	10	Tidak Ada
25	101	Tidak Ada

TIN: Terapan Informatika Nusantara
Vol 2, No 7, Desember 2021, Hal 433-444
ISSN 2722-7987 (Media Online)
Website https://ejurnal.seminar-id.com/index.php/tin

No.	Nilai	Keterangan
26	1010	Tidak Ada
27	10101	Tidak Ada
28	101011	Tidak Ada
29	1010111	Tidak Ada
30	10101110	Tidak Ada
31	101011100	Biner antara (0,68-0,6825)
32	1	Tidak Ada
33	10	Tidak Ada
34	100	Tidak Ada
35	1001	Tidak Ada
36	10010	Tidak Ada
37	100100	Tidak Ada
38	1001001	Tidak Ada
39	10010011	Tidak Ada
40	100100111	Biner antara (0,575-0,5775)
41	0	Tidak Ada
42	01	Tidak Ada
43	010	Biner antara (0,3125-0,375)
44	0	Tidak Ada
45	01	Tidak Ada
46	010	Biner antara (0,3125-0,375)
47	0	Tidak Ada
48	01	Tidak Ada
49	010	Tidak Ada
50	0100	Biner antara (0,3-0,3125)
51	0	Tidak Ada
52	00	Tidak Ada
53	001	Tidak Ada
54	0010	Tidak Ada
55	00101	Biner antara (0,155-0,1575)
56	0	Tidak Ada
57	00	Tidak Ada
58	000	Tidak Ada
59	0000	Tidak Ada
60	00001	Tidak Ada
61	000011	Tidak Ada
62	0000110	Tidak Ada
63	00001101	Biner antara (0,05-0,0525)

Penjabaran dari pengecekan nilai bit, dapat disimpulkan hasil dekompresi seperti pada tabel 18. :

Tabel 18. Tabel Hasil Dekompresi *Arithmetic Coding*

No	*Arithmetic Coding*	Nilai *Heksa*/Karakter
1	11111111	58 94
2	111001	6B 1B
3	11001001	B6 CD
4	101011100	0A 75
5	100100111	A9 48
6	010	77 77
7	010	77 77
8	0100	77 49
9	00101	29 62
10	00001101	01 A2

4. KESIMPULAN

Berdasarkan dari penelitian yang telah dilakukan, maka dapat diambil beberapa kesimpulan dari pembahasan sebelumnya. Adapun kesimpulan yang dapat diambil Berdasarkan prosedur kompresi yang dilakukan berdasarkan algoritma *arithmetic coding* telah berhasil melakukan proses kompresi *record database* pada aplikasi ensiklopedia *flora*, sehingga proses kompresi dapat berjalan sesuai dengan teknik kompresi. Penerapan algoritma *arithmetic coding* untuk mengkompresi

record database aplikasi ensiklopedia *flora*, berdasarkan contoh kompresi dalam penelitian ini mengahasilkan 72 bit setelah dikompresi, *Ratio Compression* 2,2, *Compression Ratio* 45%, *Redudancy* 55%.

REFERENCES

[1] N. B. Batubara and T. Zebua, "IMPLEMENTASI METODE EVEN-RODEH CODE UNTUK KOMPRESI KITAB UNDANG-UNDANG HUKUM PIDANA (KUHP) BERBASIS ANDROID," *KOMIK*, vol. 3, pp. 123–129, 2019.
[2] S. Siahaan, "Penerapan Algoritma Sequitur Pada Kompresi Record Database Pada Database," *JURIKOM*, vol. 6, no. 5, pp. 511–516, 2019.
[3] S. D. Nasution, "PERANCANGAN APLIKASI KOMPRESI FILE TEKS DENGAN MENERAPKAN ALGORITMA GOLDBACH CODES," *J. Ilm. INFOTEK*, vol. 1, no. 1, pp. 104–109, 2013.
[4] A. K. Saputra, Sutardi, and I. P. Ningrum, "Aplikasi Kompresi File Citra Menggunakan Algoritma Arithmetic Coding Berbasis Java," *semanTIK*, vol. 1, no. 2, pp. 1–10, 2015.
[5] H. T. Sihotang, "PERANCANGAN DAN IMPLEMENTASI ALGORITMA ARITHMETIC CODING UNTUK APLIKASI KOMPRESI DATA VIDEO DAN AUDIO," vol. 2, no. 1, pp. 58–64, 2018.
[6] F. A. Sianturi, "Kompresi File Citra Digital Dengan Arithmetic Coding," *JTIUST*, vol. 03, no. 1, pp. 45–51, 2018.
[7] H. C. Rustamaji and B. Yuwono, "APLIKASI KOMPRESI DATA MENGGUNAKAN METODE HUFFMAN STATIK PADA PERANGKAT MOBILE," *TELEMATIKA*, vol. 11, no. Heru C, pp. 9–18, 2014.
[6] Ihsan and D. P. Utomo, "Analisis Perbandingan Algoritma Even-Rodeh Code Dan Algoritma Subexponential Code Untuk Kompresi File Teks," KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer), vol. 4, no. 1, 2020.
[7] S. R. Saragih and D. P. Utomo, "Penarapan Algoritma Prefix Code Dalam Kompresi Data Teks," KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer), vol. 4, no. 1, 2020.
[8] Lamsah and D. P. Utomo, "Penerapan Algoritma Stout Codes Untuk Kompresi Record Pada Databade Di Aplikasi Kumpulan Novel," KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer), vol. 4, no. 1, 2020.