Journal of Informatics, Electrical and Electronics Engineering
ISSN 2807-9507(Media Online)
Vol 2, No 2, Desember 2022
Hal 55-59
https://djournals.com/jieee

Penerapan Kombinasi Algoritma Sequitur Dan Punctured Elias Code Untuk Kompresi File Teks

Juni Leuwarta Ompusunggu

Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: junileuw@gmail.com
Email Penulis Korespondensi: junileuw@gmail.com

Abstrak−File teks memiliki ukuran yang besar sehingga mempengaruhi ruang penyimpanan dan proses pengiriman. ukuran inilah yang menjadi faktor berapa banyak ruang penyimpanan yang akan dipakai dan berapa lama waktu yang dibutuhkan untuk melakukan pengiriman file. semakin panjangnya teks maka ukuraan juga semakin besar. ukuran yang besar dapat diatasi dengan cara melakukan proses kompresi. Kompresi dilakukan untuk mengurangi ukuran dari sebuah file. Algoritma kompresi yang dapat digunakan yaitu algoritma sequitur dan algoritma punctured elias code. Kelebihan algoritma sequitur adalah dengan mengganti 2 karakter berdampingan dengan 1 karakter terminal pada karakter set, sedangkan kelebihan algoritma punctured elias code adalah terdapat dua kode yaitu P1 dan P2. Kompresi file teks dilakukan dengan membaca input string pada file teks (*txt dan *.doc) dan mengkodekan string menggunakan punctured elias code P1 dan P2. Kedua algoritma tersebut akan dikombinasikan untuk menghasilkan ukuran yang lebih kecil dari ukuran sebelumnya.

Kata Kunci: Kompresi; Algoritma Sequitur; Algoritma Punctured Elias Code; File Teks

Abstract−Text files have a large size that affects the storage space and the sending process. This size is a factor in how much storage space will be used and how long it will take to send files. The longer the text, the bigger the size. large size can be overcome by doing the compression process. Compression is done to reduce the size of a file. Compression algorithms that can be used are sequitur algorithm and punctured elias code algorithm. The advantage of the sequitur algorithm is that it replaces 2 characters side by side with 1 terminal character in the character set, while the advantage of the punctured elias code algorithm is that there are two codes, namely P1 and P2. Text file compression is done by reading the input string in the text file (*txt and *.doc) and encoding the string using punctured elias code P1 and P2. The two algorithms will be combined to produce a smaller size than the previous size.

Keywords: Compression; Sequitur Algorithm; Punctured Elias Code Algorithm; Text Files

1. PENDAHULUAN

Suatu proses yang dapat mengubah masukan data ke bentuk aliran data yang lain dengan ukuran yang lebih kecil disebut sebagai proses kompresi. kompresi data dilakukan untuk mengubah aliran data masukan menjadi aliran data lain dengan tujuan untuk membuat ukuran menjadi lebih kecil dari ukuran aslinya. kompresi juga bertujuan untuk mengurangi pemakaian ruang memori dan mempercepat proses transmisi data.[1]

File teks merupakan susunan dari baris data yang berupa karakter huruf, angka, simbol yang merupakan representasi kode ASCII. file teks memiliki beberapa ekstensi, seperti untuk notepad memiliki ekstensi file *.txt, pada Microsoft Word tergantung dari versi MS. Office maka terdapat ekstensi *.doc, *.docx, dan *.rtf. karakter yang tersimpan pada file teks berformat angka 1 dan 0 yang merupakan hasil pembacaan dari kode ASCII. ukuran file teks dipengaruhi dengan banyaknya karakter yang tersimpan pada file tersebut, yang dapat terdiri dari karakter simbol, huruf, angka, maupun gambar. semakin besar ukuran sebuah file maka semakin banyak menggunakan ruang memori dan proses transmisi pun menjadi lebih lambat.

Permasalahan saat ini adalah dapat di atasi atau di tangani dengan adanya pemampatan atau pengkompresian pada data. pemampatan atau pengkompresian data adalah salah satu cara untuk memperkecil data sehingga hanya memerlukan sedikit ruang penyimpanan untuk mempersingkat waktu pertukaran data yang lebih segnifikan atau lebih efesien.jika tanpa adanya proses pengkompresian, mungkin kapasitas pada komputer membutuhkan ratusan bahkan jutaan kapasitas hanya untuk menyimpan sebuah file atau data yang berukuran sangat besar. maka dari itu, dengan adanya proses pengkompreian ukuran kapasitaspada file atau data dapat menjadi jauh lebih kecil.

Untuk mengurangi ukuran dari file teks, ada beberapa algoritma yang digunakan untuk proses kompresi data, seperti algoritma sequitur dan algoritma punctured elias code. algoritma sequitur bekerja dengan mengganti pasangan huruf yang muncul lebih dari satu kali dengan karakter baru yang disebut karakter non-terminal, sehingga isi teks dari file teks dapat menjadi lebih sedikit dari sebelumnya dan mempengaruhi ukuran dari file teks tersebut. kompresi dengan algoritma punctured elias code adalah dengan melakukan perhitungan sesuai dengan algoritmanya dan mengganti kode bit asli karakter dengan kode bit dari hasil perhitungan punctured elias code sehingga menghasilkan ukuran yang lebih sedikit.

Berdasarkan penelitian yang dilakukan oleh Yansyah (2015) yang membandingkan antara algoritma huffman dan punctured elias code untuk kompresi file teks mengatakan bahwa algoritma punctured elias code lebih efektif dibandingkan algoritma huffman jika terdapat banyak karakter berulang berdasarkan penelitian yang dilakukan oleh Prayoga (2019) yang berjudul kompresi file teks dengan menerapkan algoritma sequitur mengatakan bahwa algoritma sequitur lebih efektif jika terdapat pasangan karakter yang muncul lebih dari sekali[2].

Journal of Informatics, Electrical and Electronics Engineering
ISSN 2807-9507(Media Online)
Vol 2, No 2, Desember 2022
Hal 55-59
https://djournals.com/jieee

2. METODOLOGI PENELITIAN

2.1 Kompresi

Kompresi data dalam bidang ilmu komputer, ilmu pengetahuan dan seni adalah sebuah penyajian informasi ke dalam bentuk yang lebih sederhana. Kompresi data atau source coding dalam ilmu komputer dan teori informasi adalah proses mengencode informasi dengan menggunakan lebih sedikit bit dari suatu sumber yang belum di-*encode* melalui pengguna skema pengkodean yang spesifik. kompresi data dapat diartikan juga sebagai proses yang dapat mengubah sebuah aliran data masukan (sumber atau data asli) ke dalam aliran data yang lain (ukuran atau data yang dimampatkan). Yang memiliki ukuran yang lebih kecil. kompresi data menjadi salah satu cabang teori informasi karena kompresi data berkecimpung dengan masalah redudancy dalam suatu informasi [3].

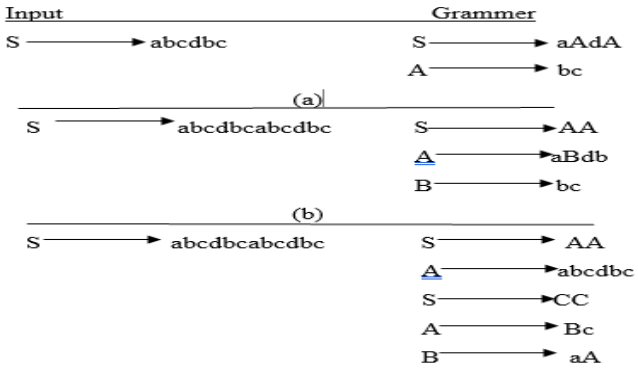
Kompresi merupakan proses untuk menghilangkan berbagai suatu kerumitan yang tidak penting (redudancy) dari suatu informasi dapat memiliki dengan cara memadatkan isi file yang sehingga ukurannya menjadi lebih kecil dengan memaksimalkan kesederhanaannya dan tetap menjaga kualitas cara untuk dari informasi tersebut. maka sebuah data yang sudah dikompres tentunya harus dapat dikembalikan lagi kebentuk aslinya, prinsip ini dinamakan dekompresi. agar supaya dapat merubah data yang terkompres diperlukan dengan cara yang berbeda seperti pada waktu proses kompresi yang dilaksanakan. pada saat dikompres ada terdapat catatan header yang berupa byte-byte yang berisi catatan mengenai isi dari file tersebut[4].

2.2 Algoritma Sequitur

Sequitur merupakan sebuah algoritma waktu linier yang menyimpulkan tata bebas konteks *(contex-free grammer)* kedalam suatu pemampatan untuk mengurangi masukan yang berulang. Operasi *sequitur* terdiri dari pemastian dua sifat yang berlaku. Ketika menjelaskan algoritma, sifat-sifatnya bertindak sebagai batasan . Algoritma beroperasi menjalankan batasan didalam sebuah tata bahasa yang ketika diagram keunikan (diagram *uniqueness)* dilanggar sebuah aturan baru dibentuk, dan ketika batasan digunakan *(rule unility)* dilanggar aturan yang sia-sia dihapus[7].

Diagram keunikan mempunyai asti bahwa tidak ada pasangan dari simbol atau diagram muncul lebih dari sekali dalam sebuah tata bahasa. Jika hal ini terjadi maka akan melanggar aturan batasan pertama (diagram *uniqueness)* sehingga akan membentuk aturan baru (simbol *non-terminal)* yang akan menggantikan simbol atau diagram yang muncul lebih dari sekali.

Rule unility mempunyai arti bahwa setiap aturan produksi digunakan lebih dari sekali. Dan jika ada aturan yang hanya digunakan sekali maka akan terjadi pelanggaran pada batasan kedua *(rule unility)* sehingga aturan tiga urutan contoh *input* dan *grammar*



Gambar 1. Rule Unility

Setiap algoritma memiliki cara kerja tersendiri, dimana cara kerja ini menjelaskan bagaimana proses atau langkah-langkah yang dimiliki dalam algoritma itu sendiri. Adapun langkah-langkah cara kerja algoritma sequitur adalah sebagai berikut:

1. Mencari nilai awal string dengan memunculkan tabel nilai frekuensi dan setiap karakter atau string akan dikalikan dengan 8 bit
2. Kemudian urutkan karakter berdasarkan frekuensi kemunculan dari yang terbesar hingga terkecil.
3. Menerapkan algoritma sequitur dengan menampilkan tabel agar lebih mudah untuk dilihat pada proses algoritma sequitur ini. Dan proses kerja algoritma ini menyimbolkan setiap pasangan karakter atau string yang muncul lebih dari sekali.
4. Melakukan perhitungan terhadap proses algoritma yang telah dikerjakan sebelumnya dengan memunculkan tabel frekuensi setiap kemunculan karakter atau string yang telah diproses.
5. Hasil yang diketahui kemudian diterapkan kedalam rumus yang berlaku pada proses kompresi tersebut.

2.3 Algoritma Punctured Elias Code

Punctured elias code untuk bilangan integer dirancang oleh Peter Finwick dalam sebuah percobaan untuk meningkatkan performa *the burrows-wheeler transfrom.* Istilah punctured datang dari tempat pengawasan *error* kode-kode (ECC). ECC

terdiri dari data yang asli ditambah sejumlah bilangan dari *check bits*. jika beberapa *check bits* dihilangkan, untuk mempersingkat serangkaian kode itu, hasil kode ditujukan sebagai punctured[5]. Cara untuk membangun kode punctured elias code ini adalah.

1. Ambil bilangan biner dari n,
2. Reversed (balikkan bit-bitnya), dan siapkan flag untuk menunjukkan jumlah bit yang bernilai 1 di dalam n
3. Untuk setiap bit di dalam n kita siapkan flag dari 1 dan akhir flag dengan 0
4. Gabungkan flag dengan bilangan biner yang sudah dibalikkan (reversed).

Tabel 1. Punctured Elias Code

n	Binary Of n	Reserved	Flag	Flag Reserved	P1
0	0	-	-	-	0
1	1	1	10	10 | 1	101
2	10	01	10	10 | 01	1001
3	11	11	110	110 | 11	11011
4	100	001	10	10 | 001	10001
5	101	101	110	110 | 101	110101
6	110	011	110	110 | 011	110011
7	111	111	1110	1110 | 111	1110111
8	1000	0001	10	10 | 0001	100001

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Analisa sistem merupakan tahap awal dalam sebuah penelitian yang bertujuan mengetahui masalah terkait dalam pembuatan sebuah sistem dan menggambarkan proses-proses yang ada di dalam sistem untuk menghasilkan keluaran yang sesuai dengan kebutuhan pemakai. Pada tahap ini akan dijelaskan secara umum cara kerja algoritma *adaptive coding* dan algoritma *sequitur* dalam kompresi file teks. Algoritma *adaptive huffman coding* merupakan teknik kompresi dengan cara melakukan pengkodean dalam bentuk bit untuk mewakili data karakter sedangkan algoritma *sequitur* merupakan teknik kompresi yang bekerja dengan cara merubah atau mengganti karakter yang sama yang memiliki jumlah lebih dari satu.

Hasil kompresi dari algoritma sequitur dan algoritma *punctured elias code* akan dibandingkan dengan menggunakan metode pembanding eksponensial untuk mengetahui algoritma mana yang lebih efektif dan efisien dalam mengompresi file teks.

Analisa kebutuhan sistem membahas secara garis besar apa saja yang dibutuhkan dalam membangun sistem tersebut. Ada beberapa kebutuhan yang harus harus dimiliki sistem yaitu membaca file string dan file hasil kompresi, melakukan proses kompresi dan dekompresi file teks berdasarkan algoritma adaptive huffman coding atau sequitur dan mampu membandingkan hasil kompresi adaptive huffman coding dan sequitur dengan metode pembandingan eksponesial.

Langkah-langkah proses kombinasi kompresi dengan algoritma sequitur dan punctured elias code adalah sebagai berikut:

1. Membaca string yang terdapat pada file teks.
2. Merubah karakter menjadi huruf kecil
3. Melakukan proses kompresi dengan algoritma sequitur
   a. Mengganti pasangan karakter yang muncul lebih dari satu kali dengan simbol non-terminal (A, B, C, ...)
   b. Setelah tidak ada lagi pasangan karakter yang muncul lebih dari satu kali, maka lakukan proses perhitungan, tetapi dilanjutkan ke perhitungan dengan Punctured elias code
4. Hasil dari tahap 3.a dilanjutkan ke perhitungan dengan punctured elias code
5. Karakter yang sudah diganti dengan bit dari punctured elias code disimpan kembali ke file teks.

Berikut adalah langkah-langkah proses kombinasi dekompresi dengan algoritma sequitur dan punctured elias code:

1. Baca isi file teks yang telah terkompresi.
2. Selanjutnya hapus flag bit dan padding dari nilai keseluruhan bit.
3. Pengecekan bit yaitu dengan cara melakukan cek bit dari bit pertama dengan tabel kode punctured elias code. Jika ditemukan bit yang sesuai dengan tabel kode punctured elias code maka ubah nilai string yang sesuai sehingga akan mendapatkan hasil semula sebelum dikompresi.
4. Lalu setelah dari dekompresi punctured elias code, lakukan pengembalian karakter non-terminal ke pasangan karakter yang ada pada tabel kamus aturan sequitur.

3.1.1 Penerapan Algoritma Punctured Elias Code

Pada tahap ini akan dilakukan kompresi dan dekompresi beberapa string dengan algoritma punctured elias code. Berikut ini merupakan contoh proses kompresi dengan metode punctured elias code.

1. Input karakter yang akan dijadikan teks sebagai berikut.
   String : JULEHA AKAN MAKAN
   Jumlah : 17
   Karakter : {J, U, L, E, H, A, Sp, K, N, M}
2. Membuat tabel karakter dan frekuensi

Pada langkah ini akan dilakukan proses perhitungan ukuran bit awal dari string di atas.

Tabel 2. Tabel Karakter dan frekuensi

Char	Ascii Code	Ascii Binary	Bit	Frekuensi	Bit x Frekuensi
J	74	01001010	8	1	8
U	85	01010101	8	1	8
L	76	01001100	8	1	8
E	69	01000101	8	1	8
H	72	01001000	8	1	8
A	65	01000001	8	5	40
Sp	32	00010000	8	2	16
K	75	01001011	8	2	16
N	78	01001110	8	2	16
M	77	01001101	8	1	8
		Total			136

3.1.2 Penerapan Algoritma Sequitur

Pada tahapan ini string di atas akan kita proses dengan menggunakan algortima sequitur. Berikut adalah tahapan algoritma sequitur:

a. Baca string, dan ubah semua karakter menjadi huruf kecil (*lowercase*).
b. Mencari pasangan karakter yang muncul lebih dari 1 kali.
c. Ganti pasangan karakter yang muncul lebih dari 1 kali dengan simbol non-terminal (A, B, C, ..).
d. Ulangi langkah ke-2 (b) sampai tidak ada pasangan karakter lagi.

Tabel 3. Kompresi Sequitur

Proses	String	Pasangan Karakter	Penggantian Simbol	Aturan	String Hasil
1	juleha akan makan	ak	A	A = ak	juleha Aan mAan
2	juleha Aan mAan	Aa	B	A = ak B = Aa	juleha Bn mBn
3	juleha Bn mBn	Bn	C	A = ak B = Aa C = Bn	juleha C mC

Dari tabel 3. proses sequitur didapat hasil akhir string adalah “juleha C mC”. Melihat dari algoritma kompresi sequitur yang akan menghitung ukuran akhir dari string setelah diproses, maka dalam penelitian ini tidak akan dilakukan perhitungan akhir, karena penelitian ini akan melakukan proses kombinasi antara sequitur dengan Punctured elias code, maka setelah hasil akhir dari algoritma sequitur yaitu “juleha C mC”, akan dilanjutkan ke proses algoritma punctured elias code.

Setelah didapat hasil akhir string dari algoritma sequitur, yaitu “juleha C mC”, selanjutnya akan dilakukan pengurutan karakter berdasarkan frekuensi tertinggi ke terendah dan dilanjutkan perhitungan akhir dengan algortima punctured elias code, hasil dapat dilihat pada tabel 4. dibawah.

String : juleha C mC
Jumlah : 11

Tabel 4. Kompresi dengan Algoritma *Punctured elias code*

Char	*Punctured Elias Code* (P1)	Bit	Frekuensi	Bit x Frekuensi
Sp	0	1	2	2
C	101	3	2	6
j	1001	4	1	4
u	11011	5	1	5
l	10001	5	1	5
e	110101	6	1	6
h	110011	6	1	6
a	1110111	7	1	7
m	100001	6	1	6
	Total			47

Dari tabel tersebut dapat dibentuk bit dan string sebelum dikompresi yaitu "juleha C mC" Menjadi string bit "1001, 11011, 10001, 110101, 110011, 1110111, 0, 101, 0, 100001, 101". Sebelum ditulis kesebuah file hasil kompresi dilakukan penambahan string bit itu sendiri apakah habis dibagi 8 dan berapa sisanya jika dibagi 8.

10011101 11000111 01011100 11111011 10101010 0001101

Jumlah string 47 jika dibagi 8 maka memiliki sisa 7 (dinayatakan sebagai n) agar bit dapat hasil dibagi 8 maka dapat ditambahkan padding bit 0 sebanyak 7 – n + "1", maka 7 – 7 + "1" = 1. Karena 7-7 habis, maka hanya menambah padding "1" pada akhir string bit. Sehingga hasilnya:

10011101 11000111 01011100 11111011 10101010 00011011

Kemudian tambahkan flagbit yaitu biner dari decimal dengan rumus 9 – n = 9 – 7 = 2 = 00000010, sehingga menjadi:

10011101 11000111 01011100 11111011 10101010 00011011 00000010

Lalu ubah masing-masing 8 bit menjadi karakter kembali dan disimpan pada file.

10011101 = •
11000111 = Ç
01011100 = \
11111011 = û
10101010 = ª
00011011 =
00000010 =

Dan dapat dihitung kinerja kompresinya menurut parameter yang sudah ditentukkan yaitu:

Compression ratio (Cr)

$$Cr = \frac{\text{ukuran data setelah di kompresi}}{\text{ukuran data sebelum di kompresi}} \times 100\%$$

$$Cr = \frac{47 \text{ bit}}{136 \text{ bit}} \times 100\%$$

$$Cr = 34.55\%$$

Maka dari hasil compression ratio didapatkan bahwa sebanyak 34.55% sudah berhasil dimampatkan.

4. KESIMPULAN

Kesimpulan yang dapat diambil setelah melakukan perancangan aplikasi kompresi file teks dengan menerapkan kombinas algoritma sequitur dan punctured elias code Proses kompresi teks digunakan dengan kombinasi algoritma sequitur dan punctured elias code sehingga menghasilkan ukuran yang lebih kecil jika hanya memakai satu algoritma. Proses kombinasi dengan algoritma sequitur yaitu dengan cara mengurangi huruf pada teks sehingga huruf yang ada pada file teks lebih sedikit dan dilanjutkan dengan proses kompresi menggunakan algoritma punctured elias code untuk perhitungan akhir.

REFERENCES

[1] Y. Darnita, K. Khairunnisyah, and H. Mubarak, "KOMPRESI DATA TEKS DENGAN MENGGUNAKAN ALGORITMA SEQUITUR," *SISTEMASI*, 2019, doi: 10.32520/stmsi.v8i1.429.
[2] D. A. Yansyah, "Perbandingan Metode Punctured Elias Code Dan Huffman Pada Kompresi File Text," *J. Ris. Komput.*, 2015.
[3] A. Wibowo, "Kompresi data menggunakan metode huffman," *Semantik*, 2012.
[4] E. P. Ervin Umi; Krisnawati, Lucia D., "Kompresi Data Teks Menggunakan Pendekatan Grammar Compression Dengan Algoritma Sequ Itur," *J. Inform.*, 2007.
[5] S. Neuburger, "The Burrows-Wheeler transform," *ACM SIGACT News*, 2010, doi: 10.1145/1753171.1753177.
[6] J. Rumbaugh, "Unified Modeling Language (UML)," in *Encyclopedia of Software Engineering*, 2010.
[7] C. Rupp, S. Queins, die SOPHISTen, C. Rupp, S. Queins, and die SOPHISTen, "Use-Case-Diagramm," in *UML 2 glasklar*, 2012.
[8] 2013 Rosa & Salahuddin, "UML, Use Case Diagram, Activity Diagram, Class Diagram," in *Rekayasa Perangkat Lunak Terstruktur*, 2013.
[9] Y. Amrizal and R. Kurniati, "Game Aritmatika Berbasis Android," *INOVTEK Polbeng - Seri Inform.*, 2016, doi: 10.35314/isi.v1i2.121.
[10] S. K. Alfa Satyaputra, M.Sc, Eva Maulina Aritonang, *JAVA for Beginners with eclipse 4.2 Juno*. Jakarta: PT Elex Media Komputindo Kelompok Gramedia, Anggota IKAPI, 2012.