JURNAL TEKNISI **E-ISSN: 2775-0965**
(Jurnal Teknologi Komputer dan Sistem Informasi)
Agustus 2023, Vol 3, No 2, Hlm. 49 – 53
Available online at http://jurnal.goretanpena.com/index.php/teknisi

PENINGKATAN RASIO KOMPRESI ALGORITMA RLE UNTUK KOMPRESI TEKS DENGAN MENGGUNAKAN ALGORITMA BWT DAN SEQUITUR

Rian Syahputra[1], Surya Darma Nasution[2], Alwin Fau[3]
[1,3]Sistem Informasi, Universitas Budi Darma
[2]Teknik Informatika, Universitas Budi Darma
email: ryansyah93@gmail.com

Abstract: To reduce the size can be done with compression techniques. Some of the compression techniques that are often used are Run Length Encoding. The RLE algorithm allows a significant size reduction by reducing the number of symbols in a row to a number of the number of it. But for text compression the RLE algorithm still cannot be used because there are no symbols or letters that line up. To improve the results of text compression from the RLE algorithm, data changes can be used first, such as the Burrows-Wheeler Transform algorithm or the Sequitur algorithm. The RLE algorithm cannot compress text because a row of text words is arranged randomly so that the word has a meaning, BWT and Sequitur managed to make changes to the sample text, but RLE still cannot execute it properly, so the RLE algorithm is not suitable for text compression.

Keywords: compression; RLE; BWT; sequitur; comparison

Abstrak: Untuk melakukan pengurangan ukuran bisa dilakukan dengan teknik kompresi. Beberapa teknik kompresi yang sering digunakan adalah Run Length Encoding. Algoritma RLE memungkinkan pengurangan ukuran yang signifikan dengan mempersingkat jumlah dari simbol yang berderet dengan angka dari jumlah simbol yang berderet tersebut. Namun untuk kompresi teks algoritma RLE masih belum bisa digunakan karena tidak ada simbol atau huruf yang berderet. Untuk meningkatkan hasil kompresi teks dari algoritma RLE dapat digunakan perubahan data terlebih dahulu, dengan algoritma Burrows-Wheeler Transform atau algoritma Sequitur. Algoritma RLE tidak bisa untuk kompresi teks karena deretan dari suatu kata teks disusun secara acak sehingga kata tersebut memiliki suatu arti, BWT dan Sequitur berhasil melakukan perubahan pada teks sampel, namun RLE tetap tidak bisa mengeksekusinya dengan baik, sehingga algoritma RLE tidak cocok untuk kompresi teks.

Kata kunci: kompresi; RLE; BWT; sequitur; perbandinga kinerja

PENDAHULUAN

Mengumpulkan dan menyimpan file merupakan kebiasaan dari setiap manusia. Pengumpulan file dilakukan untuk menyimpan dan menjadi bukti digital dari yang telah dilakukan, seperti file dokumen, audio, video, dan gambar. Untuk file dokumen atau file teks biasanya disimpan sebagai bukti digital berisi informasi yang penting dan tidak dapat dihapus. Untuk itu maka penyimpanan file dokumen yang banyak bisa mengakibatkan pemakaian memori yang cukup banyak pula [1].

Salah satu upaya yang bisa digunakan untuk mengurangi ukuran adalah teknik kompresi. Kompresi dilakukan untuk mengurangi ukuran data dengan cara memadatkan data sehingga ukuran menjadi lebih kecil dan memori penyimpanan lebih sedikit digunakan, selain itu juga proses transmisi menjadi lebih cepat [2].

Salah satu teknik pada kompresi adalah Run Length Encoding (RLE). Algoritma RLE biasanya digunakan untuk kompresi file citra. Algoritma ini bekerja dengan cara dengan

mengurangi ukuran fisik pengulangan string dari deretan karakter / byte data [3]. Tetapi algoritma RLE juga bisa digunakan untuk kompresi data teks, seperti penelitian yang dilakukan Umar Mansyuri dengan judul Kompresi data teks dengan metode RLE mengatakan algoritma RLE bekerja dengan menghitung kemunculan simbol lalu menuliskan jumlah simbol tersebut satu kali diikuti dengan jumlah kemunculannya. Teknik kompresi dengan RLE ini biasanya digunakan untuk data yang memiliki kesamaan dan berdekatan, yang memiliki banyak kesamaan pola [4].

Untuk meningkatkan hasil kompresi algoritma RLE, bisa digunakan algoritma Burrow-Wheeler Transform (BWT) dan algoritma Sequitur. Algoritma BWT juga termasuk algoritma yang melakukan transformasi pada blok data menjadi bentuk baru namun tetap mengandung karakter yang sama, hanya saja urutannya yang berbeda. Transformasi BWT cenderung mengelompokan karakter secara berurutan sehingga meningkatkan peluang untuk menemukan karakter yang sama secara berurutan [5]. Algoritma sequitur beroperasi menjalankan batasan di sebuah tata bahasa yang ketika aturan diagram keunikan dilanggar, maka akan membentuk aturan baru, dan ketika batasan aturan kegunaan dilanggar, maka aturan yang tidak digunakan akan dihapus [1][6][7]. Sehingga dengan demikian untuk meningkatkan kualitas kompresi sebuah dokumen digunakanlah algoritma RLE dan untuk memberikan kualitas kompresi yang lebih baik digunakanlah algoritma Burrow-Wheeler Transform (BWT) sehingga hasil kompresi menjadi sangat baik.

METODE

Run-length encoding (RLE) merupakan suatu bentuk yang sangat sederhana dari kompresi data, di mana data berjalan yang berurutan memiliki nilai data yang sama berturut-turut maka disimpan sebagai nilai data tunggal dan dihitung panjangnya [3]. Kompresi dari data teks yang dilakukan jika ada huruf atau simbol yang sama yang berturut-turut. Kompresi dan dekompresi data menggunakan algoritma RLE merupakan suatu teknik yang digunakan untuk memadatkan data yang berisi karakter yang berulang secara berderet. Saat karakter yang sama berderet sebanyak empat kali atau lebih (lebih dari tiga), algoritma ini mengkompres data.

RLE memiliki dua tipe yaitu RLE tipe 1 dan RLE tipe 2.
Data mentah: ABCCCCCCCDEFGGG = 15 karakter, menggunakan RLE tipe 1 (minimal memiliki 4 huruf yang sama berderet) maka ditulis: ABC7! DEFG!3 = 11 karakter. RLE tipe 1 ini memiliki suatu karakter yang menjadi penanda yaitu '!'. Salah satu kelemahan teknik RLE tipe 1 adalah jika terdapat karakter angka, maka sulit menentukan mana tanda mulai dan mana tanda akhir?

Selanjutnya dalam RLE tipe 2 digunakanlah *flag* dengan bilangan negatif untuk menandai berapa banyak jumlah karakter tersebut.

Data: ABCCCCCCCDEFGGG = 15 Karakter
Dengan RLE tipe 2: -2AB7CDEF3G = 11 Karakter .

Contoh lain:
Data; AB12CCCCCDEEEEF = 15 Karakter, Dengan RLE tipe 2; -4AB125CD4EF = 12 Karakter

Teknik kompresi algoritma RLE ini lebih berguna untuk data yang memiliki banyak kesamaan dan berdekatan yang memilki kesamaan pola [4].

Sequitur adalah algoritma waktu linier yang menyimpulkan tata bahasa bebas konteks (context free grammar) ke dalam suatu proses pemampatan untuk mengurangi data yang berulang. Operasi sequitur sendiri terdiri dari dua sifat pemastian yang berlaku. Saat menjelaskan algoritma sequitur, sifat-sifatnya bertindak sebagai sebuah batasan yang berlaku. Algoritma beroperasi menjalankan batasan pada sebuah tata bahasa yang dimana ketika aturan digram keunikan dilanggar, maka sebuah aturan baru akan dibentuk, dan ketika batasan aturan kegunaan dilanggar, maka aturan yang tidak digunakan akan dihapus. Berikut ini menguraikan bagaimana dua batasan ini terjadi:

1. Diagram Keunikan (Diagram Uniqueness) memiliki arti bahwa tidak ada suatu

JURNAL TEKNISI **E-ISSN: 2775-0965**
(Jurnal Teknologi Komputer dan Sistem Informasi)
Agustus 2023, Vol 3, No 2, Hlm. 49 – 53
Available online at http://jurnal.goretanpena.com/index.php/teknisi

pasangan dari simbol memiliki kemunculan lebih dari satu kali didalam sebuah tata bahasa. Jika terjadi maka akan melanggar aturan keunikan sehingga akan dibentuk aturan baru menggunakan simbol non-terminal yang menggantikan simbol yang muncul lebih dari satu kali.

2. Aturan Kegunaan (Rule Utility) memiliki arti yaitu setiap aturan yang diproduksi harus digunakan lebih dari satu kali. Jika ada aturan yang hanya digunakan satu kali, maka terjadi pelanggaran pada batasan aturan kegunaan, sehingga aturan tersebut akan dihapus [1].

Masukkan karakter pertama s untuk membuat hasil dari $S \rightarrow abcababcd$:
Mengulang
Mencocokkan pada non-terminal yang ada:
$S \rightarrow AcAab$ → $A \rightarrow ab$
$S \rightarrow AcAa$
Membuat sebuah non-terminal baru :
$S \rightarrow AcAAc$ $A \rightarrow ab$
→ $B \rightarrow Ac$
Memindahkan non-terminal :
$S \rightarrow AcAAc$ $A \rightarrow ab$
$S \rightarrow BAB$ → $B \rightarrow Ac$
Memasukkan karakter baru :
$S \rightarrow BABd$
Sampai tidak ada karakter yang tersisa (*Ervin, 2011*)

Gambar 1. Tahapan Sequitur

Algoritma Burrows-Wheller Transform (BWT) pertama kali diperkenalkan pada tahun 1994 oleh Michael Burrows dan David Wheller. Contoh algoritma BWT dengan 1 dimensi p = [3, 2, 5, 3, 1, 4, 2, 6]. Untuk proses BWT tahap pertama dilakukan perputaran dengan memindahkan nilai dari yang paling kanan ke paling kiri untuk setiap baris data [8].

index	Step 1							
0	3	2	5	3	1	4	2	6
1	2	5	3	1	4	2	6	3
2	5	3	1	4	2	6	3	2
3	3	1	4	2	6	3	2	5
4	1	4	2	6	3	2	5	3
5	4	2	6	3	2	5	3	1
6	2	6	3	2	5	3	1	4
7	6	3	2	5	3	1	4	2

Gambar 2. Tahap 1 BWT

Tahap ke dua yaitu setiap baris disusun dengan teknik leksikografi (berdasarkan kamus yaitu urutan). Dan Tahap ke tiga atau terakhir dari algoritma BWT adalah hasil nilai pada indeks terakhir setelah tahap 2 diambil sebagai hasil BWT.

index	Step 2							
0	1	4	2	6	3	2	5	3
1	2	5	3	1	4	2	6	3
2	2	6	3	2	5	3	1	4
3	3	1	4	2	6	3	2	5
4	3	2	5	3	1	4	2	6
5	4	2	6	3	2	5	2	1
6	5	3	1	4	2	6	3	2
7	6	3	2	5	3	1	4	2

index	Step3
0	3
1	3
2	4
3	5
4	6
5	1
6	2
7	2

Gambar 3. Tahap 2 dan 3 BWT

Nilai asli dari p berada pada baris ke lima dan output BWT dinyatakan dengan:

$$L = [3\,3\,4\,5\,6\,1\,2\,2]^T,$$

Gambar 4. Hasil BWT

Pada indeks ke empat, hasil dapat ditulis seperti berikut, BWT = [Indeks L], diamana L merupakan hasil dari BWT dan indeks sebagai penanda lokasi asli dari urutan secara leksikografi. Algoritma BWT merupakan proses transformasi data yang dapat dikembalikan (reserve) sehingga nilai asli dapat kembali ke bentuk semula. Hanya diperlukan BWT L dan index untuk mengembalikan ke urutan aslinya. Berikut tahap reserve BWT:

Langkah (3i-2): Letakan kolom n di depan kolom 1, ..., -1.

Langkah (3i-1): Urutkan panjang string i yang dihasilkan secara leksikografi.

Langkah (3i): Letakan list yang terurut pada tabel.

index	(i=1)			(i=2)			(i=3)		
	a	b	c	a	b	c	a	b	c
0	3	1	1...3	31	14	14...3	314	142	142...3
1	3	2	2...3	32	25	25...3	325	253	253...3
2	4	2	2...4	42	26	26...4	426	263	263...4
3	5	3	3...5	53	31	31...5	531	314	314...5
4	6	3	3...6	63	32	32...6	632	325	325...6
5	1	4	4...1	14	42	42...1	142	426	426...1
6	2	5	5...2	25	53	53...2	253	531	531...2
7	2	6	6...2	26	63	63...2	263	632	632...2

Gambar 5. Dekompresi BWT-1

index	(i=4)			(i=5)			(i=6)		
	a	b	c	a	b	c	a	b	c
0	3142	1426	1426...3	31426	14263	14263...3	314263	142632	142632...3
1	3253	2531	2531...3	32531	25314	25314...3	325314	253142	253142...3
2	4263	2632	2632...4	42632	26325	26325...4	426325	263253	263253...4
3	5314	3142	3142...5	53142	31426	31426...5	531426	314263	314263...5
4	6325	3253	3253...6	63253	32531	32531...6	632531	325314	325314...6
5	1426	4263	4263...1	14263	42632	42632...1	142632	426325	426325...1
6	2531	5314	5314...2	25314	53142	53142...2	253142	531426	531426...2
7	2632	6325	6325...2	26325	63253	63253...2	263253	632531	632531...2

Gambar 6. Dekompresi BWT-2

JURNAL TEKNISI **E-ISSN: 2775-0965**
(Jurnal Teknologi Komputer dan Sistem Informasi)
Agustus 2023, Vol 3, No 2, Hlm. 49 – 53
Available online at http://jurnal.goretanpena.com/index.php/teknisi

index	*(i=7)*		
	a	*b*	*c*
0	3142632	1426325	1426325...3
1	3253142	2531426	2531426...3
2	4263253	2632531	2632531...4
3	5314263	3142632	3142632...5
4	6325314	3253142	**3253142...6**
5	1426325	4263253	4263253...1
6	2531426	5314263	5314263...2
7	2632531	6325314	6325314...2

Gambar 7. Dekompresi BWT-3

Output dari proses *reserve transform* menampilkan bahwa nilai p asli juga berada pada indeks ke empat, itulah mengapa indeks sering dimanfaatkan untuk merepresentasikan output/hasil dari proses *forward transform* [8].

HASIL DAN PEMBAHASAN

Pada penelitian ini akan digunakan sampel teks ”*compression*”. Sebagai pendahuluan kita coba untuk kompresi teks sampel menggunakan algoritma RLE, sebagai berikut:

compression => *compre1sion*

Dari hasil diatas bisa dilihat bahwa hasil kompresi teks dengan algoritma RLE tidak mengalami pengurangan jumlah simbol, *compression* berjumlah 11 simbol dan setelah dikompresi menjadi *compre1sion* jumlahnya tetap 11.

Selanjutnya sampel teks *compression* akan dirubah terlebih dahulu dengan menggunakan algoritma BWT, dengan *Left-Cycle-Shift*, seperti berikut:

c	o	m	p	r	e	s	s	i	o	n
o	m	p	r	e	s	s	i	o	n	c
m	p	r	e	s	s	i	o	n	c	o
p	r	e	s	s	i	o	n	c	o	m
r	e	s	s	i	o	n	c	o	m	p
e	s	s	i	o	n	c	o	m	p	r
s	s	i	o	n	c	o	m	p	r	e
s	i	o	n	c	o	m	p	r	e	s
i	o	n	c	o	m	p	r	e	s	s
o	n	c	o	m	p	r	e	s	s	i
n	c	o	m	p	r	e	s	s	i	o

Gambar 8. Tahap Pertama

Setelah tahap pertama, maka dilanjutkan tahap ke dua, yaitu dengan menyusun hasil dari tahap pertama secara leksikografi atau seperti urutan kamus.

c	o	m	p	r	e	s	s	i	o	n
e	s	s	i	o	n	c	o	m	p	r
i	o	n	c	o	m	p	r	e	s	s
m	p	r	e	s	s	i	o	n	c	o
n	c	o	m	p	r	e	s	s	i	o
o	m	p	r	e	s	s	i	o	n	c
o	n	c	o	m	p	r	e	s	s	i
p	r	e	s	s	i	o	n	c	o	m
r	e	s	s	i	o	n	c	o	m	p
s	i	o	n	c	o	m	p	r	e	s
s	s	i	o	n	c	o	m	p	r	e

Gambar 9. Tahap Kedua

Dari tahap kedua kita ambil kolom yang paling akhir sebagai hasil dari algoritma BWT yaitu ”nrsoocimpse”. Selanjutnya kita akan kompresi hasil BWT tersebut menggunakan algoritma RLE sebagai berikut:

nrsoocimpse = > nrs1ocimpse

Dari hasil diatas kita bisa lihat bahwa BWT telah berhasil mengubah urutan dari teks sampel *compression* menjadi nrsoocimpse, namun setelah hasil BWT dikompres dengan RLE, tidak terjadi pengurangan ukuran juga.

Selanjutnya teks sampel *compression* akan kita ubah dengan menggunakan algoritma Sequitur, karena algoritma ini bekerja dengan diagram keunikan pengulangan pasangan simbol, maka untuk teks sampel akan kita tambahkan kata tambahan *computer* menjadi *compression compass*, untuk memungkinkan algoritma Sequitur dapat bekerja secara maksimal, sebagai berikut:

Teks compression compass ditemukan keunikan (*Diagram* Uniqueness) yaitu “co” muncul lebih dari satu kali, maka diganti dengan simbol baru yaitu A, maka rumusnya ”co => A” sehingga menjadi Ampression Ampass, lalu ditemukan kembali keunikan ”Am” muncul lebih dari satu kali, maka diganti dengan simbol baru yaitu B, maka rumusnya ”Am => B”, selanjutnya disini aturan kegunaan (*Rule Utility*) yaitu simbol A tidak digunakan di kamus, karena sudah berada di simbol B, maka simbol A akan dihapus dan dikembalikan seperti semula menjadi ”Am => com => B” dengan hasil Bpression Bpass.

Selanjutnya masih ditemukan keunikan yaitu ”Bp” muncul lebih dari satu kali maka akan diganti dengan simbol C dan simbol B akan dihapus karena tidak digunakan lagi menjadi ”Bp => comp => C” sehingga hasil menjadi Cression Cass. Masih ditemukan diagram keunikan yaitu ”ss” diganti ke simbol D

dengan rumus ”ss => D” menjadi hasil akhir yaitu ”CreDion CaD”. Lalu hasil dari sequitur kita kompresi dengan algoritma RLE seperti berikut:

CreDion CaD => CreDion CaD

Dari hasil RLE diatas maka tidak terjadi pengurangan jumlah huruf walau sudah dilakukan perubahan dengan menggunakan algoritma Sequitur. Setelah dilakukan penelitian dan perubahan data teks sampel menggunakan algoritma BWT dan Sequitur, tidak ditemukan adanya pengurangan jumlah huruf setelah dilakukan proses RLE. Sebagai hasil pembahasan dapat dilihat pada tabel dibawah:

Tabel 1. Perbandingan Hasil

Algoritma	Hasil RLE	
	Sebelum	Sesudah
BWT-RLE	nrsoocimpse	nrs1ocimpse
Sequitur-RLE	CreDion CaD	CreDion CaD

SIMPULAN

BWT dan Sequitur berhasil merubah teks sampel, namun algoritma RLE tetap tidak mampu untuk mengeksekusinya. Algoritma RLE tidak cocok jika harus digunakan untuk kompresi teks, karena teks tidak memiliki simbol yang berderetan karena teks ataupun kata tersusun dari huruf yang acak sehingga memiliki suatu arti. Algoritma RLE memang khusus ditujukan untuk kompresi citra yang memiliki nilai yang sama yang berderetan, begitu juga dengan file dimana nilai hexadecimal dari sebuah file memiliki kesamaan nilai yang berderetan.

DAFTAR PUSTAKA

[1] Y. Darnita, K. Khairunnisyah, and H. Mubarak, “Kompresi Data Teks Dengan Menggunakan Algoritma Sequitur,” *Sistemasi*, vol. 8, no. 1, p. 104, 2019, doi: 10.32520/stmsi.v8i1.429.

[2] S. R. Saragih and D. P. Utomo, “Penarapan Algoritma Prefix Code Dalam Kompresi Data Teks,” *KOMIK (Konferensi Nas. …*, vol. 4, no. 1, pp. 249–252, 2020, doi: 10.30865/komik.v4i1.2691.

[3] A. Jalaludin and Y. Melita, “Implementasi Metode Run Length Encoding untuk Kompresi Citra,” *J. Tek.*, vol. 3, no. 2, pp. 249–254, 2012.

[4] U. Mansyuri, “Kompresi Data Teks Dengan Metode Run Length Encoding,” *J. SIMASI*, vol. 1, no. 2, pp. 102–109, 2021.

[5] A. Dony Mahendra, E. Suryani, and A. Aziz, “Analisis Perbandingan Kinerja Kombinasi Algoritma BWT-RLE-MTF-Huffman Dan BWT-MTF-RLE-Huffman Pada Kompresi File,” *J. Teknol. Inf. ITSmart*, vol. 1, no. 2, p. 107, 2016, doi: 10.20961/its.v1i2.606.

[6] S. Siahaan, “Penerapan Algoritma Sequitur Pada Kompresi Record Database Pada Database,” *JURIKOM (Jurnal Ris. Komputer)*, vol. 6, no. 5, pp. 511–516, 2019, [Online]. Available: https://www.ejurnal.stmik-budidarma.ac.id/index.php/jurikom/article/view/1644

[7] E. Buulolo, “Sequitur Algorithm For Particular Character Compression Or Words Always Returned,” *Int. J. Informatics Comput. Sci. (The IJICS)*, vol. 1, no. 1, pp. 15–17, 2017.

[8] A. Sulistyanto, “Peningkatan Kompresi Teks Shannon-Fano Dengan Kombinasi Burrows-Wheller Transform (BWT) Run- Length Encoding (RLE) dan Shannon-Fano,” vol. 2, pp. 2–4, 2015.