Jurnal Ilmu Komputer, Teknologi Dan Informasi
Vol 1, No. 2, Juli 2023, Hal 34 - 42
ISSN: **2963-0169** (Online - Elektronik)
https://journal.grahamitra.id/index.php/jurikti

Implementasi Kombinasi Algoritma Tunstall Code Dan Boldi-Vigna Untuk Kompresi File Pdf

Romi Adi Sandra

Fakultas Ilmu Komputer Dan Teknologi Informasi, Prodi Studi Teknik Informatika, Universitas Budi Darma, Kota, Indonesia
Jl. Sisingamangaraja No. 338, Medan, Sumatera Utara, Indonesia
Email: romiadisandra@gmail.com

Abstrak-Kebutuhan file pdf yang digunakan untuk keperluan pengiriman dengan *size* yang terlalu besar tentu sangat mempersulit pengiriman, hal ini dikarenakan banyak media pengiriman yang membatasi jumlah *size* pdf. Masalahnya adalah pada kapasitas penyimpanan yang disediakan, jika file pdf yang terlalu besar dapat mengakibatkan pemborosan ruang penyimpanan dan memperlambat transmisi data. Sehingga dibutuhkan teknik kompresi. Pada penelitian ini akan dilakukan proses kompresi file pdf dengan dua proses algoritma yaitu algoritma *Tunstall Code* dan algoritma *Boldi-Vigna*. Proses kompresi file pdf pertama dilakukan dengan algoritma *Tunstall Code*, kemudian dilanjutkan dengan kompresi kedua menggunakan algoritma *Boldi-Vigna*. Sedangkan untuk proses dekompresi dilakukan sebanyak dua kali yaitu dekompresi pertama menggunakan algoritma *Boldi-Vigna* dan dekompresi kedua menggunakan algoritma *Tunstall Code.* Berdasarkan hasil analisa dengan dua proses kompresi didapatkan *space saving* kompresi file pdf hingga mencapai 62,5% sedangkan rasio kompresi sebanyak 37,5% .

Kata Kunci: *kompresi, pdf, Tunstall Code, Boldi-Vigna*

Abstract-The need for a pdf file that is used for shipping purposes with a size that is too large is certainly very difficult to send, this is because many delivery media limit the number of pdf sizes. the problem is in the storage capacity provided, if the pdf file is too large it can result in wasting storage space and slow down data transmission. So a compression technique is needed. In this study, a pdf file compression process will be carried out with two algorithm processes, namely the Tunstall Code algorithm and the Boldi-Vigna algorithm. The first pdf file compression process is done with yahoo Tunstall Code, then followed by a second compression using yahoo Boldi-Vigna. Meanwhile, the decompression process was carried out twice, namely the first decompression using the Boldi-Vigna algorithm and the second decompression using the Tunstall Code algorithm. Based on the results of the analysis with two compression processes, the storage space for the pdf file compression reaches 62.5% while the compression ratio is 37.5%.

Keywords: *compression, pdf, Tunstall Code, Boldi-Vigna*

1. PENDAHULUAN

Bagi masyarakat umum keberadaan file dengan ekstensi PDF mungkin tidaklah asing, bahkan banyak yang mengetahui aplikasi yang dapat membaca file PDF *Adobe Reader*. Namun, tidak sedikit masyarakat yang mengerti tujuan dari file PDF, mengingat untuk pengolahan data sudah ada *Microsoft Office.* Penggunaan file PDF biasanya identik dalam dunia kerja yang mengarah pada pengolah data. Data yang sudah dibuat dalam *Microsoft Office* seperti *word*, *excel* dan lainnya biasa di simpan dalam ekstensi aslinya atau dalam ekstensi lain, salah satunya adalah ekstensi .PDF. Banyaknya orang yang memilih menggunakan format PDF dikarenakan sangat praktis dan tidak memakan waktu lama untuk membukanya.

File PDF juga sering digunakan untuk keperluan pengiriman via *email* atau media pengiriminan lainya. Banyak perusahaan atau universitas yang meminta untuk mengirimkan suatu file, dirubah terlebih dahulu kedalam bentuk PDF sebelum dikirimkan. Kebutuhan file PDF yang digunakan untuk keperluan pengiriman dengan *size* yang terlalu besar tentu sangat mempersulit pengiriman, hal ini dikarenakan banyak media pengiriman yang membatasi jumlah *size* PDF [1]. Masalah yang terjadi adalah ukuran data file PDF yang besar mengakibatkan pemborosan memori penyimpanan dan memperlambat proses transmisi data. Sehingga dibutuhkan teknik pengecilan data atau biasa disebut dengan teknik kompresi. Kompresi bertujuan untuk mengurangi redudansi data menjadi sekecil mungkin. Kompresi data adalah sebuah cara untuk memadatkan data sehingga hanya memerlukan ruangan penyimpanan lebih kecil sehingga lebih efisien dalam menyimpannya atau mempersingkat waktu pertukaran data tersebut. Kompresi data dapat dibagi menjadi dua, *lossless compression* dan *lossy compression* [2].

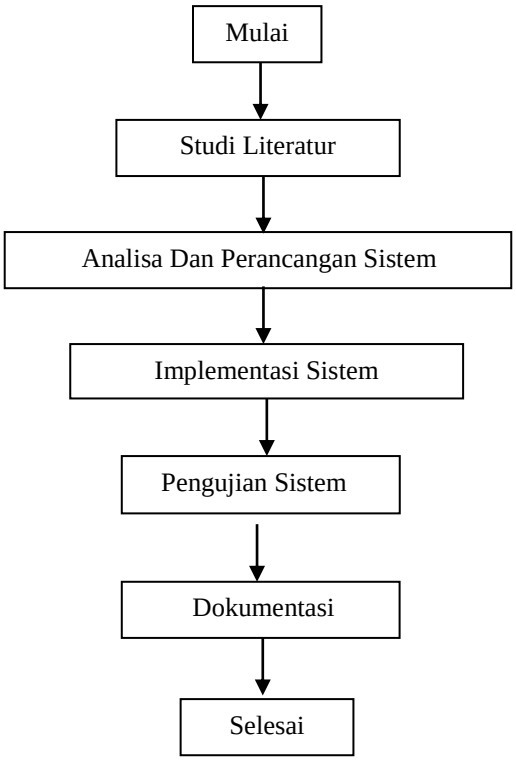
Lossless compression berarti data yang telah dikompres mampu dikembalikan menjadi data aslinya. Dalam *lossy compression*, data yang telah dikompresi tidak dapat dikembalikan menjadi data asli karena hilangnya informasi pada saat proses kompresi. Teknik kompresi menggunakan algoritma dalam melakukan lagkah-langkah perhitungan kompresi [3]. Ada berbagai macam algoritma kompresi data salah satunya adalah algoritma *Tunstall Code* dan Algoritma *Boldi-Vigna*. Algoritma *Tunstall Code* adalah salah satu metode kompresi yang memetakan simbol sumber ke sejumlah bit serta menyortirnya berdasarkan panjang dari variabel data, sehingga dapat ditunjukkan bahwa dalam sebuah kamus yang cukup besar, jumlah bit per huruf dari sumber tidak terbatas . Sedangkkan algoritma *Boldi-Vigna* diperkenalkan oleh Paolo Boldi dan Sebastiano Vigna sebagai keluarga *Variable-Length*. Kedua algoritma tersebut adalah jenis *lossless* dimana algoritma dapat dikombinasikan untuk memperkecil ukuran PDF [4].

Pada penelitian ini, kompresi file PDF dilakukan dengan 2 algoritma. Proses kompresi file PDF pertama dilakukan dengan algoritma *Tunstall Code* kemudian dilanjutkan dengan kompresi kedua menggunakan algoritma *Boldi-Vigna*. Kombinasi ini bertujuan untuk mendapatkan hasil *space saving* dan rasio kompresi yang lebih optimal dari teknik satu kali kompresi.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Adapun tahapan penelitian yang di laksanakan adalah seperti gambar 1 di bawah ini:



Gambar 1. Tahapan Penelitian

Berdasarkan Gambar 1 di atas, Metode pengumpulan data yang digunakan dalam pembahasan penelitian ini adalah sebagai berikut:

a. Studi Literatur
   Merupakan tahap pengumpulan data dengan cara mengumpulkan literatur, jurnal, *paper*, dan buku-buku yang berkaitan dengan judul penelitian, serta mencari informasi dari berbagai sumber di internet untuk mengetahui perkembangan terbaru dari data yang diambil sebagai bahan dalam pembuatan tugas akhir.
b. Analisa dan perancangan sistem
   Tahap ini akan dilaksanakan analisa data, kebutuhan sistem yang digambarkan dalam *flowchart*, dan perancangan antarmuka.
c. Implementasi sistem
   Implementasi dilaksanakan berdasarkan hasil analisa dan perancangan yang telah dilakukan sebelumnya. Dalam tahap ini dilakukan pengkodean (*coding*) dalam menggunakan bahasa pemograman *Microsoft Visual Studio Net* 2008.
d. Pengujian sistem
   Pengujian ini dilakukan pengujian terhadap proses kompresi pada sistem yang telah di bangun. Meliputi input file pdf, proses kompresi, proses dekompresi dan juga mencakup apakah implementasi sistem sudah sesuai dengan teori serta perancangan yang sudah dilakukan sebelumnya.
e. Dokumentasi
   Membuat dokumentasi sistem dari tahap awal sampai pengujian sistem, kemudian untuk disusun dalam format penelitian.

2.2 Kompresi

Kompresi data adalah ilmu atau seni yang merepresentasikan informasi dalam bentuk yang lebih *compact* . Istilah kompresi tersebut diterjemahkan dari kata bahasa Inggris "*compression*" yang berarti pemampatan [5]. Dalam bidang teknik, kompresi berarti proses memampatkan sesuatu yang berukuran besar sehingga menjadi kecil. Dengan demikian, kompresi data berarti teknik untuk mengurangi ukuran data agar penyimpanannya jauh lebih padat dan juga untuk mengurangi waktu pengiriman data tersebut. Kompresi data bertujuan untuk mengurangi jumlah bit yang digunakan untuk menyimpan atau mengirimkan informasi. Tujuan daripada kompresi data tidak lain adalah untuk mengurangi data yang berlebihan (*Redundancy Data)* sehingga ukuran data menjadi lebih kecil dan lebih ringan sehingga mengurangi biaya untuk penyimpanan [6] .

2.3 Konsep Kompresi

Proses kompresi sendiri didasarkan pada bahwa isi *file* akan dibaca secara *per byte* (8 bit) sehingga menghasilkan nilai pembacaan antara 0 hingga 255. Suatu metode pada kompresi data akan menghasilkan bit-bit (satuan terkecil pembentuk data) data baru yang lebih pendek dibandingkan oleh bit-bit data sebelum dikompresi. Bit-bit data yang lebih pendek tersebut biasanya tidak akan bisa dibaca oleh komputer sebelum dilakukan proses *encoding*. Didalam komputer satu karakter direpresentasikan oleh bilangan ASCII (*American Standard Code For Information Interchange*) sebanyak delapan bit dalam bilangan biner. Jika ternyata jumlah bit-bit data tersebut bukan merupakan kelipatan delapan. Maka dibentuk variabel baru sebagai penambahan ke bit-bit data itu agar bit-bit data tersebut habis dibagi delapan. Variabel ini adalah *padding* dan *flag bits* [6].

a. *Flag Bits*
   Flag bits adalah penambahan bilangan biner sepanjang delapan bit setelah *padding* dimana *flag bits* ini adalah sejumlah bilangan yang memberikan sebuah tanda bahwa terdapat n buah *padding* di dalam bit-bit data hasil dari kompresi. Penambahan *flag bits* ini dimaksudkan untuk mempermudah dalam membaca bit-bit data hasil kompresi pada saat proses dekompresi. Contoh misalkan bit-bit data yang telah diberikan *padding* adalah 1001011**0**. Karena terdapat 1 bit penambahan *padding* maka *flag bits*nya adalah bilangan biner dari 1 dengan panjang 8 bit yaitu 00000001. Sehingga bit-bit datanya menjadi 1001011**00000001** setelah diberikan *flag bits* [7].
b. *Padding*
   Padding adalah penambahan bit 0 sebanyak kekurangan jumlah bit-bit data pada hasil proses kompresi sehingga jumlah keseluruhan bit-bit data tersebut merupakan kelipatan delapan (habis dibagi delapan). Contoh misalkan dihasilkan bit-bit data hasil kompresi yaitu 1001011. Terdapat 7 bit data dalam bilangan biner. Maka dilakukan penambahan bit 0 sebanyak 1 kali agar jumlah bit-bit data tersebut habis dibagi delapan. Sehingga bit bit data itu menjadi 1001011**0** setelah diberikan *padding*[5].

2.4 Ukuran Kinerja Kompresi

Ketika mengukur performa dari suatu algoritma kompresi, biasanya akan fokus pada efisiensi dari ruang penyimpanan dan efisiensi waktu termasuk dalam faktor lainnya. Performa suatu algoritma kompresi juga bergantung pada tipe dan struktur dari sumber masukan, yang mana juga akan tergantung pada apakah termasuk dalam kategori kompresi *lossless* atau *lossy*. Untuk menilai dan mengetahui keefektifan suatu algoritma kompresi ataupun untuk mengetahui perbandingan *file* sebelum dan setlah di kompresi, maka diperlukan beberapa parameter kinerja kompresi yang harus diperhatikan sebagai berikut. Ada beberapa faktor pembanding yang digunakan dalam penelitian ini, yaitu *Ratio of Compression* (RC), *Compression Ratio* (CR), *Redundancy* (Rd) .

a. *Ratio of Compression* (Rc)
   Ratio of Compression (Rc) adalah perbandingan antara ukuran data sebelum dikompresi dengan ukuran data setelah dikompresi .

$$Rc = \frac{ukuran\ data\ sebelum\ dikompresi}{ukuran\ data\ setelah\ dikompresi} \quad (1)$$

b. *Compression Ratio* (Cr)
   Compression Ratio (Cr) adalah persentasi besar data yang telah dikompresi yang didapat dari hasil perbandingan antara ukuran data setelah dikompresi dengan ukuran data sebelum dikompresi .

$$Rc = \frac{ukuran\ data\ setelah\ dikompresi}{ukuran\ data\ sebelum\ \ dikompresi} \times 100\% \quad (2)$$

c. *Saving Percentages / Space Saving* (SS)
   Saving Percentages menghitung persentasi penyusutan dari data sebelum dilakukan kompresi [3].

$$SS = \frac{ukuran\ data\ sebelum - ukuran\ data\ setelah}{ukuran\ data\ sebelum} \times 100\% \quad (3)$$

2.5 Algoritma Tunstall Code

Algoritma *Tunstall Code* adalah algoritma kompresi data yang dikemukakan oleh Brian Parker Tunstall sebagai subjek thesis PhD sewaktu Georgia Institute of Technology pada tahun 1967 dimana dalam algoritma ini tahapan awal yang dilakukan yaitu membangun sebuah tabel yang memuat simbol, frekuensi, dan kolom probabilitas .

Setelah itu sortir simbol sesuai dengan probabilitas terbesar. Kemudian lakukan literasi, untuk mengetahui berapa literasi yang harus dilakukan yaitu dengan memasukkan ke dalam rumus $N + (N - 1) \leq 2^n$. Kemudian lakukan literasi sesuai dengan hasil k yang diperoleh. Untuk melakukan peliterasian, pertama-tama pilah simbol sesuai dengan probabilitas terbesar, kemudian hapus simbol dengan probabilitas tertinggi. Misalkan kita ingin membuat N-bit *Tunstall codewords*, jumlah *codewords* adalah 2N. Codewords merupakan representasi bit tiap simbol. Adapun langkah-langkah algoritma ini, adalah:

a. Buat sebuah pohon dengan simpul akar yang memiliki probabilitas 1.0. Hubungkan 0 dan 1 ke *root*, dua node daun yang dihasilkan masing-masing memiliki probabilitas terjadinya 1 dan 0, yang masing-masing adalah Prob (1) dan

Prob (0).

b. Node daun dengan probabilitas tertinggi dibagi menjadi dua cabang dengan 0 dan 1 sebagai label. Setelah membelah, jumlah simpul daun meningkat 1.
c. Langkah 2 diulangi hingga jumlah total node daun sama dengan 2N.
d. Tetapkan *codeword* dengan panjang yang sama (panjang = N) ke node *leaf*.

2.6 Algoritma Boldi-Vigna

Zeta (ζ) kode juga dikenal sebagai *Boldi-Vigna code*, diperkenalkan oleh Paolo Boldi dan Sebastiano Vigna sebagai keluarga *Variable-Length Code* yang merupakan pilihan terbaik untuk kompresi. Dimulai dengan hukum Zipf, seorang kuasa hukum empiris [Zipf 07] diperkenalkan oleh Linguis George K. Zipf. Menyatakan bahwa frekuensi setiap kata dalam bahasa alami kira-kira berbanding terbalik dengan posisinya dalam tabel frekuensi .

Boldi-Vigna kode zeta dimulai dengan bilangan bulat k positif yang menjadi menyusut oleh Faktor kode. Himpunan semua bilangan bulat positif dibagi menjadi $[2^0, 2^k - 1]$, $[2^k, 2^{2k} - 1]$, $[2^{2k}, 2^{3k} - 1]$, dan secara umum $[2^{hk}, 2^{(h+1)k} - 1]$. Panjang setiap interval adalah $2^{(h+1)k} - 2^{hk}$.

Diberikan interval [0, z-1] dan sebuah integer x di interval ini, pertama kita hitung $s = \lfloor \log_2 z \rfloor$. Jika $x < 2^s - z$, dikodekan sebagai unsur xth elemen pada interval ini, pada s - 1 bit. Jika tidak, maka dikodekan sebagai $(x - z - 2^s)$ th elemen pada interval di s bit. Dengan latar belakang ini, di sini dibahas bagaimana kode zeta dibangun. Mengingat bilangan bulat n akan dikodekan, kami mempekerjakan k untuk menentukan *interval* di mana n berada. Salah satu yang diketahui, nilai-nilai h dan k yang digunakan dengan cara yang sederhana untuk membangun kode zeta n dalam dua bagian, nilai h + 1 di *unary* (sebagai nol h diikuti dengan 1), diikuti oleh minimal kode biner dari $n - 2^{hk}$ dalam *interval* $[0, 2^{(h+1)k} - 2^{hk} - 1]$.

3. HASIL DAN PEMBAHASAN

Pada saat ini banyak orang-orang yang menggunakan file pdf untuk saling bertukar informasi. Contohnya dalam mengirim berkas-berkas yang berhubungan dengan file seperti *word* teks yang selalu dirubah kedalam file pdf. Banyak perusahaan atau universitas meminta terhadap pengirim file untuk merubah data file kedalam bentuk pdf terlebih dahulu sebelum dikirim. Kebutuhan file pdf yang digunakan untuk keperluan pengiriman dengan *size* yang terlalu besar tentu sangat mempersulit pengiriman, hal ini dikarenakan banyak media pengiriman yang membatasi jumlah *size* pdf. Masalahnya adalah pada kapasitas penyimpanan yang disediakan, jika file pdf yang terlalu besar dapat mengakibatkan pemborosan ruang penyimpanan dan memperlambat transmisi data. Berdasarkan pada masalah di atas dan pada masalah bab sebelumnya, makan dibutuhkan sebuah teknik pengecilan ukuran file pdf.

Teknik tersebut adalah kompresi data. Kompresi data ini akan mengecilkan ukuran file pdf dari ukuran sebelumnya. File pdf yang sudah dikecilkan ukuranya, dapat dikembalikan semula dengan teknik dekompresi. Pada penelitian ini akan dilakukan proses kompresi file pdf dengan dua proses algoritma yaitu algoritma *Tunstall Code* dan algoritma *Boldi-Vigna*. Proses kompresi file pdf pertama dilakukan dengan algoritma *Tunstall Code*, kemudian dilanjutkan dengan kompresi kedua menggunakan algoritma *Boldi-Vigna*. Sedangkan untuk proses dekompresi dilakukan sebanyak dua kali yaitu dekompresi pertama menggunakan algoritma *Boldi-Vigna* dan dekompresi kedua menggunakan algoritma *Tunstall Code*, sehingga didapatkan ukuran file pdf awal sebelum terkompresi. Tujuan dari melakukan dua kali kompresi pada file pdf ini adalah untuk mendapatkan hasil berupa parameter kompresi yang lebih maksimal dan optimal seperti *space saving* dan rasio kompresi.

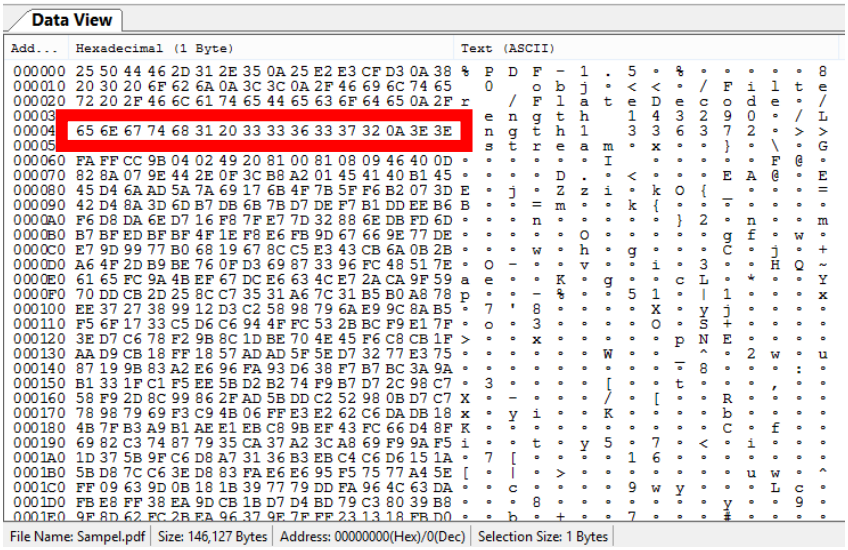
3.1 Penerapan Algoritma

Dalam melakukan kompresi file pdf, sebelumnya dilakukan analisa file pdf tersebut. Berdasarkan pada batasan masalah, file pdf yang dikompresi berupa file yang berisi karakter *string* (teks), selain itu tidak bisa. Sebelum melakukan kompresi file pdf, terlebih dahulu menentukan file yang akan dikompresi. Pada contoh kasus ini file pdf yang akan dikompresi diberi nama Sampel.pdf seperti pada gambar di bawah ini:

Gambar 2. File PDF Sampel

Berdasarkan pada gambar 1 di atas, file pdf memiliki ukuran 143 KB. Untuk mendapatkan nilai *string* dari file PDF untuk keperluan hitungan manual maka dibutuhkan aplikasi bantuan. Adapun hasil dari *string* file pdf berupa Sampel.pdf dapat dilihat pada gambar di bawah ini:

Jurnal Ilmu Komputer, Teknologi Dan Informasi
Vol 1, No. 2, Juli 2023, Hal 34 - 42
ISSN: **2963-0169** (Online - Elektronik)
https://journal.grahamitra.id/index.php/jurikti



Gambar 2. Nilai File PDF Sampel

Berdasarkan pada gambar di atas, untuk memudahkan hitungan manual, maka penulis hanya mengambil contoh *string* file pdf sampel pada baris ke-5 saja yaitu nilai hexadesimal file pdf seperti dibawah ini:

String : 65, 6E, 67, 74, 68, 31, 20, 33, 33, 36, 33, 37, 32, 0A, 3E, 3E

Berdasarkan pada *string* yang akan dijadikan sampel dan contoh kasus pada penelitian ini akan dikompresi menggunakan algoritma *Tunstall Code* terlebih dahulu, kemudian dilanjutkan menggunakan algoritma *Boldi-Vigna*.

3.1.1 Kompresi Berdasarkan Algoritma *Tunstall Code*

Untuk memulai proses kompresi, terlebih dahulu setiap *string* sampel file pdf dirubah kedalam bentuk hexa dan binner kemudian diurutkan dengen memulai dari frekuensi terbesar (banyaknya huruf yang sama) ke yang terkecil. Adapun urutan nilai file pdf sampel yang sudah disusun dapat dilihat pada tabel di bawah ini :

Tabel 1. Urutan Karakter

Hexa	*Binner*	*Bit*	*Frekuensi*	*Bit * Frek*
33	00110011	8	3	24
3E	00111110	8	2	16
65	01100101	8	1	8
6E	01101110	8	1	8
67	01100111	8	1	8
74	01110100	8	1	8
68	01101000	8	1	8
31	00110001	8	1	8
20	00100000	8	1	8
36	00110110	8	1	8
37	00110111	8	1	8
32	00110010	8	1	8
0A	00001010	8	1	8
Total				**128 Bit**

Berdasarkan pada tabel 1 di atas, satu nilai hexa/karakter memiliki nilai delapan bit, dan nilai biner didapat dari melihat tabel ASCII. Adapun jumlah nilai hexa file pdf sebanyak 16 nilai dengan total ukuran 128 bit. Setelah nilai bit didapatkan dari semua nilai, selanjutnya adalah melakukan kompresi pertama menggunakan algoritma *Tunstall Code*. Penkompresian dilakukan dengan mengalikan nilai frekuensi dari nilai hexa file pdf yang muncul dengan nilai frekuensi bit dari algoritma *Tunstall Code*. Adapun untuk mendapat nilai bit algoritma *Tunstall Code* adalah sebagai berikut:

a. Buat sebuah pohon dengan simpul akar yang memiliki probabilitas 1.0. Hubungkan 0 dan 1 ke *root*, dua node daun yang dihasilkan masing-masing memiliki probabilitas terjadinya 1 dan 0. Node daun dengan probabilitas tertinggi dibagi menjadi dua cabang dengan 0 dan 1 sebagai label. Setelah membelah, jumlah simpul daun meningkat 1.
b. Langkah 2 diulangi hingga jumlah total node daun sama dengan 2N.
c. Tetapkan *codeword* dengan panjang yang sama (panjang = N) ke node *leaf*.

Adapun tabel nilai *Tunstall Code* yang didapatkan dari sejumlah banyaknya string file pdf yang akan dikompresi adalah sebagai berikut:

Tabel 2. Kode Nilai *Tunstall*

N	*Codeword Tunstall*
1	00
2	01
3	10
4	11
5	101
6	100
7	110
8	111
9	1000
10	1001
11	1010
12	1011
13	1100

Selanjutnya dilakukan kompresi string sampel file pdf menggunakan algoritma *Tunstall Code* dengan mengalikan banyaknya frekuensi nilai hexa yang muncul dengan banyaknya bit nilai *Tunstall*. Adapun proses kompresi tersebut dapat dilihat pada tabel di bawah ini:

Tabel 3. Kompresi Pertama Dengan Nilai Tunstall

N	*Hexa*	*Codeword Tunstall Code*	*Jumlah Bit Tunstall*	*Frekuensi Karakter*	*Bit * Frek*
1	33	00	2	3	6
2	3E	01	2	2	4
3	65	10	2	1	2
4	6E	11	2	1	2
5	67	101	3	1	3
6	74	100	3	1	3
7	68	110	3	1	3
8	31	111	3	1	3
9	20	1000	4	1	4
10	36	1001	4	1	4
11	37	1010	4	1	4
12	32	1011	4	1	4
13	0A	1100	4	1	4
Total					**46 bit**

Berdasarkan pada tabel 3 di atas dapat dibentuk nilai bit baru hasil kompresi dari susunan nilai hexa file pdf sebelum kompresi yaitu "65, 6E, 67, 74, 68, 31, 20, 33, 33, 36, 33, 37, 32, 0A, 3E, 3E" menjadi nilai bit biner : Nilai hexa pertama "65" pada tabel 3 kode Tunstallnya adalah 10 Sehingga untuk hasil seterusnya dapat dilihat pada tabel di bawah ini:

Tabel 4. Susunan Bit Baru Kompresi Pertama

No	*Hexa*	*Kode Tunstall*
1	65	10
2	6E	11
3	67	101
4	74	100
5	68	110
6	31	111
7	20	1000
8	33	00
9	33	00
10	36	1001
11	33	00
12	37	1010
13	32	1011
14	0A	1100
15	3E	01
16	3E	01
Total		**46 bit**

Jurnal Ilmu Komputer, Teknologi Dan Informasi
Vol 1, No. 2, Juli 2023, Hal 34 - 42
ISSN: **2963-0169** (Online - Elektronik)
https://journal.grahamitra.id/index.php/jurikti

Sehingga jika digabung menghasilkan nilai bit baru yaitu:

"10111011001101111000000010010010101011110001 01"

Proses selanjutnya adalah melakukan penambahan atau *padding* dan *flag bits*. *Padding* dilakukan jika jumlat bit hasil kompresi tidak habis dibagi 8 atau memiliki sisa. Sedangkan *Flag bits* adalah nilai angka *padding* yang dijadikan biner. Karena jumlah bit hasil kompresi *Tunstall* 46 jika dibagi 8 maka memiliki sisa 6 maka ditambahkan *padding* sebanyak kurangnya sisa yaitu 2 bit nilai binner "**00**". Dengan demikian, *flags bit*nya adalah nilai binner dari angka 2 (*padding*) yaitu "**00000010**". Sehingga *string* bit yang terbentuk keseluruhanya adalah:

"1011101100110111100000001001001010101111000101**0000000010**"

Adapun total keseluruhan bit adalah 46 + 2 (*padding*) + 8 (*flagsbit*) = 56 bit**.**

3.1.2 Kompresi Berdasarkan Algoritma *Boldi-Vigna*

Sebelum melakukan kompresi kedua menggunakan algoritma *Boldi-Vigna*, terelebih dahulu hasil kompresi pertama dipisah menjadi 8 bit dan disusun dari frekuensi terbesar hingga terkecil.

Tabel 5. Nilai Hexa dan Binner Hasil Kompresi Pertama

Binner	Hexa
10111011	BB
00110111	37
10000000	80
10010010	92
10101111	AF
00010100	14
00000010	2

Adapun bit hasil kompresi pertama yang telah disusun dapat dilihat pada tabel di bawah ini:

Tabel 6. Urutan Nilai Hasil Kompresi Pertama

Binner	*Nilai Hexa*	*Bit*	*Frekuensi*	*Bit * Frek*
10111011	BB	8	1	8
00110111	37	8	1	8
10000000	80	8	1	8
10010010	92	8	1	8
10101111	AF	8	1	8
00010100	14	8	1	8
00000010	2	8	1	8
Total				**56 Bit**

Berdasarkan pada tabel 6 di atas, adapun jumlah nilai hexa (binner) sebanyak 7 nilai dengan total ukuran 56 bit. Setelah nilai bit didapatkan dari semua nilai, selanjutnya adalah melakukan kompresi kedua menggunakan algoritma *Boldi-Vigna*. Penkompresian dilakukan dengan mengalikan nilai frekuensi dari setiap nilai hexa yang muncul dengan nilai frekuensi bit dari algoritma *Boldi-Vigna*.

Adapun nilai dari *Boldi-Vigna* yang didapat sesuai dengan jumlah nilai hexa yang dikompresi dapat dilihat pada tabel berikut:

Tabel 7. Kode *Boldi-Vigna*

n	Kode *Boldi-Vigna*
1	1000
2	10010
3	10011
4	10100
5	10101
6	10110
7	10111

Selanjutnya dilakukan kompresi kedua menggunakan algoritma *Boldi-Vigna* dengan mengalikan banyaknya frekuensi nilai hexa yang muncul dengan banyaknya bit nilai *Boldi-Vigna*. Adapun proses kompresi tersebut dapat dilihat pada tabel di bawah ini:

Tabel 8. Kompresi Kedua Dengan Nilai Boldi-Vigna

N	*Hasil Kompresi Pertama*		*Boldi-Vigna*	*Jumlah Bit Boldi-Vigna*	*Frekuensi Binner*	*Bit * Frek*
	Binner	*Hexa*				
1	10111011	BB	1000	4	1	4
2	00110111	37	10010	5	1	5
3	10000000	80	10011	5	1	5
4	10010010	92	10100	5	1	5
5	10101111	AF	10101	5	1	5
6	00010100	14	10110	5	1	5
7	00000010	2	10111	5	1	5
Total						**34 bit**

Berdasarkan pada tabel 8 di atas dapat dibentuk nilai bit baru hasil kompresi dari susunan nilai hexa kompresi pertama yaitu "BB, 37, 80, 92, AF, 14, 2" (tanpa tanda koma) menjadi nilai bit biner baru:

Misal nilai hexa pertama "BB" pada tabel 8 kode *Boldi-Vignaya* adalah "1000" Sehingga untuk hasil seterusnya dapat dilihat pada tabel di bawah ini :

Tabel 9. Susunan Bit Baru Kompresi Kedua

No	*Hexa*	*Kode Boldi-Vigna*
1	BB	1000
2	37	10010
3	80	10011
4	92	10100
5	AF	10101
6	14	10110
7	2	10111
Total		**34 bit**

Sehingga jika digabung menghasilkan nilai bit baru yaitu:

"1000100101001110100101011011010111".

Proses selanjutnya adalah melakukan penambahan atau *padding* dan *flag bits*. *Padding* dilakukan jika jumlat bit hasil kompresi tidak habis dibagi 8 atau memiliki sisa. Sedangkan *Flag bits* adalah nilai angka *padding* yang dijadikan biner. Karena jumlah bit hasil kompresi kedua *Boldi-Vigna* 34 tidak habis dibagi 8 dan memiliki sisa 2 maka ditambahkan *padding* sebanyak kurangnya sisa yaitu 6 bit nilai binner "**000000**". Dengan demikian, *flags bit*nya adalah nilai binner dari angka 6 (*padding*) yaitu "**00000110**". Sehingga *string* bit yang terbentuk keseluruhanya adalah:

"1000100101001110100101011011010111**00000000000110**"

Adapun total keseluruhan bit adalah 48 bit. Berdasarkan hasil kedua kompresi maka, didapatkan hasil kompresi akhir berupa pengecilan ukuran *string* sampel file pdf dari 128 bit menjadi 48 bit. Adapun hasil akhirnya setiap bit dipecah menjadi 8 bit dapat dilihat pada tabel di bawah ini:

Tabel 10. Hasil Akhir Kompresi

No	*Binner*	*Nilai Hexa*	*Bit*	*Frekuensi*	*Bit * Frek*
1	10001001	89	8	1	8
2	01001110	4E	8	1	8
3	10010101	95	8	1	8
4	10110101	B5	8	1	8
5	11**000000**	C0	8	1	8
6	**00000110**	6	8	1	8
Total					**40 Bit**

Kemudian hasil kompresi nilai desimal dirubah kedalam bentuk karakter dengan melihat tabel ASCII. Adapun nilai karakter yang dihasilkan dari proses kedua kompresi:

Tabel 11. Karakter Hasil Kompresi

No	*Binner*	*Nilai Hexa*	*Karakter*
1	10001001	89	‰
2	01001110	4E	N
3	10010101	95	•
4	10110101	B5	µ
5	11000000	C0	À

No	*Binner*	*Nilai Hexa*	*Karakter*
6	00000110	6	ACK

3.1.3 Menghitung Parameter Hasil Kompresi

Berdasarkan hasil kompresi dengan *Tunstall Code dan Boldi-Vigna* pada proses sebelumnya, adapun dapat dihitung proses kinerja kompresinya berupa parameter yaitu:

a. *Ratio of Compression* (R_C)

$$Rc = \frac{Ukuran\ data\ sebelum\ dikompresi}{Ukuran\ data\ setelah\ dikompresi}$$

$$Rc = \frac{128}{48}$$

Rc = 2,7

b. *Compression Ratio* (C_R)

$$Cr = \frac{Ukuran\ data\ setelah\ dikompresi}{Ukuran\ data\ sebelum\ dikompresi} x100\%$$

$$Cr = \frac{48}{128} x100$$

Cr = 37,5 %

c. *Space Saving* (S_s)

$$Ss = \frac{Ukuran\ data\ sebelum\ dikompresi - setelah\ kompresi}{Ukuran\ data\ sebelum\ dikompresi} x100\%$$

$$Ss = \frac{128-48}{128} x100$$

Ss = 62,5 %

4. KESIMPULAN

Berdasarkan pengujian yang telah dilakukan sebelumnya, dapat disimpulkan bahwa Kompresi dan dekompresi menggunakan algoritma *Tunstall Codes* dan *Boldi-Vigna* berhasil dilakukan terhadap file PDF dengan membaca setiap *string* file PDF, Proses implementasi kompresi mendapatkan hasil berupa parameter *Compression Ratio,Ratio of Compression* dan *Space Saving* dengan menghitung setiap nilai pengurangan ukuran file sesuai rumus parameter yang sudah ditetapkan pada aplikasi. Dan Penerapan algoritma *Tunstall Codes* dan *Boldi-Vigna* pada aplikasi kompresi file PDF, mendapatkan hasil ukuran kompresi file PDF dengan rata-rata < 50 % .

REFERENSI

[1] P. D. F. Adobe, "101-quick overview of PDF file format," *Adobe Syst. [cit. 2.5. 2013] Dostupné z http//partners. adobe. com/public/developer/tips/topic tip31. html Lit. Lit.*, 2010.

[2] S. Shanmugasundaram and R. Lourdusamy, "A comparative study of text compression algorithms," *Int. J. Wisdom Based Comput.*, vol. 1, no. 3, pp. 68–76, 2011.

[3] D. A. Rachesti, T. W. Purboyo, and A. L. Prasasti, "Comparison of Text Data Compression Using Huffman, Shannon-Fano, Run Length Encoding, and Tunstall Methods," *Int. J. Appl. Eng. Res.*, vol. 12, no. 23, pp. 13618–13622, 2017.

[4] S. Mehfuz and U. Tiwari, "A Tunstall based lossless compression algorithm for Wireless Sensor Networks," in *2015 Annual IEEE India Conference (INDICON)*, 2015, pp. 1–4.

[5] A. P. U. Siahaan, "Implementasi Teknik Kompresi Teks Huffman," *None*, vol. 10, no. 2, p. 101651, 2016.

[6] K. Geofandy, E. A. Nathaniel, and H. Agung, "Kompresi File Menggunakan Konversi Biner Hexadecimal dan Algoritma Huffman Encoding," *J. Ilm. Teknol. Infomasi Terap.*, vol. 5, no. 3, pp. 36–46, 2019.

[7] T. S. Waruwu and K. Telaumbanua, "Kombinasi Algoritma OTP Cipher dan Algoritma BBS dalam Pengamanan File," *J. SIFO Mikroskil*, vol. 17, no. 1, pp. 119–126, 2016.