Journal of Computing and Informatics Research
ISSN 2808-375X (Media Online)
Vol 1, No 2, Maret 2022
Hal 43−49

Penerapan Algoritma Taboo Codes Pada Kompresi File Text

Naomi Yuni Intan Hutagalung

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

Abstrak-Kebutuhan manusia modern dan tuntutan dari perkembagan teknologi saat ini memaksa setiap individu banyak bekerja menggunakan komputer dan menggunakan data yang serba digital, sehingga mengharuskan setiap orang menggunakan aplikasi pengelola file texs seperti Microsoft Office untuk mengolah sebuah teks konvensional menjadi file teks digital. Hal ini disebabkan file dalam bentuk elektronik digital lebih disukai dari pada dalam bentuk konvensional, karena beberapa faktor tertentu seperti masalah keamanan data, kemudahan dalam memproses, pendistribusian maupun modifikasi data. Kecenderungan manusia pada saat ini dalam mengumpulkan data berupa file teks dan dokumen didalam sebuah komputer yang menyebabkan media penyimpanan pada perangkat komputer akan lebih cepat habis. Terlebih lagi saat ini kebanyakan individu dalam berbagi file teks menggunakan media internet melalui email ataupun media sosial seperti whatsapp dan telegram, sementara itu jaringan internet yang sangat lambat di Indonesia mengakibatkan pengiriman file teks yang berukuran besar menjadi sangat lama. Algoritma Taboo Codes memiliki pendekatan terhadap panjangnya suatu variable. Penggagas Algoritma Taboo Codes adalah Steven Pigeon, prinsip dari Algoritma Taboo Codes yaitu untuk memilih bilangan positif n dan membalikkan pola n untuk mengindikasikan akhir dari kode tersebut, algoritma taboo codes bekerja untuk mengkompresi file teks dengan cara mengganti nilai biner asli dari setiap karakter menjadi nilai biner algoritma taboo codes yang lebih singkat, sehingga mengkompresi file teks menggunakan algoritma taboo codes dapat memperkecil ukuran asli file teks sekitar 20%.

Kata Kunci: File Teks; Elektronic Document; Unified Modelling Language; Kompresi; Taboo Codes

Abstrack-The needs of modern humans and the demands of technological developments today force every individual to work a lot on computers and use all digital data, requiring everyone to use a text file manager application such as Microsoft Office to process a conventional text into a digital text file. This is because files in digital electronic form are preferred over conventional forms, due to certain factors such as data security problems, ease of processing, distribution and modification of data. The tendency of humans at this time to collect data in the form of text files and documents on a computer causes storage media on computer devices to run out faster. Moreover, at this time most individuals share text files using internet media via email or social media such as WhatsApp and telegram, meanwhile the very slow internet network in Indonesia causes the sending of large text files to take a very long time. The Taboo Codes algorithm has an approximation to the length of a variable. The initiator of the Taboo Codes Algorithm is Steven Pigeon, the principle of the Taboo Codes Algorithm, which is to select positive n numbers and reverse the n pattern to indicate the end of the code, the taboo codes algorithm works to compress text files by replacing the original binary value of each character into a binary value. The taboo codes algorithm is shorter, so compressing a text file using the taboo codes algorithm can reduce the original text file size by about 20%.

Keywords: File Text; Dokumen Elektronik; Unified Modeling Language; Compression; Taboo Codes

1. PENDAHULUAN

Kompresi data adalah proses mengubah aliran masukan data (sumber aliran atau data mentah asli) ke aliran data lain (keluaran, *bitstream*, atau aliran terkompresi) yang memiliki ukuran lebih ringkas. Kompresi *file* teks diproses menggunakan proses memampatkan data teks agar sizenya ringkas dan masih mempertahankan isi dari teks dokumen. Jadi diperlukan kualitas kompresi yang baik yaitu ukuran yang minimum dengan tidak mengurangi kualitas tetapi mengurangi ukuran data tersebut[1].

File teks merupakan suatu *file* yang berisi informasi – informasi dalam bentuk teks. Data yang berasal dari dokumen pengolahan kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh yang termasuk dalam masukan data teks yang terdiri dari karakter, angka dan tanda baca. *Input* dan *output* data teks dipresentasikan sebagai set karakter atau sistem kode yang dikenal oleh sistem komputer. Ada tiga jenis set karakter umum yang dapat digunakan untuk input dan output pada komputer, yaitu ASCII, EBCDIC, dan Unicode. ASCII (American Code for Information Interchange) adalah suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode, tetapi ASCII bersifat universal. Kode ASCII 8 bit memiliki komposisi bilangan biner yang dimulai dari 00000000 sehingga 11111111[2].

Pada penelitian sebelumnya mengenai kompresi *file* teks yang berjudul "Penerapan Metode Deflate dan Algoritma Goldbach Codes Dalam Kompresi File Teks" oleh Muhammad Rio Irliansyah, Surya Darma Nasution, dan Kurnia Ulfa pada tahun 2017 ,dikatakan bahwa penerapan metode Deflate dan algoritma Goldbach codes dilakukan dengan cara mengkompresi tiap karakter yang terdapat pada *file* teks berdasarkan cara kerja tiap algoritma dan rasio yang dihasilkan berdasarkan proses kompresi *file* teks tersebut menghasilkan *file* dengan ukuran lebih kecil dari sebelum dilakukan proses kompresi[3].

Dari hasil penelitian yang dilakukan oleh Sutardi pada tahun 2014, yang berjudul "Implementasi dan Analisis Kinerja Algoritma Shannonfano Untuk Kompresi File Text", menyimpulkan bahwa *file* teks dengan ukuran yang sama dan memiliki karakter bervariasi akan menghasilkan tingkat pemampatan *file* yang lebih rendah dibandingkan *file* teks yang memiliki karakter yang tidak terlalu bervariasi. *File* teks dengan karakter yang bervariasi akan menghasilkan

pemampatan *file* yang rendah, sedangkan *file* teks dengan karakter yang tidak bervariasi memiliki tingkat pemampatan *file* yang tinggi dengan ukuran *file* yang sama[4].

Untuk mengatasi permasalahan yang ada dari banyaknya *file* teks yang disimpan di dalam komputer, hal ini akan menimbulkan proses pemakaian ruang penyimpanan pada komputer lebih besar dan lebih boros, maka teknik kompresi diperlukan dengan tujuan memperkecil ukuran dari *file* teks berformat *.docx tersebut. Dengan mengkompresi *file* teks tersebut diharapkan dapat menghemat penyimpanan dari *harddisk.* Kompresi memiliki berbagai jenis algoritma yang dapat digunakan. Maka dari itu penulis bermaksud menggunakan algoritma *Taboo Codes* untuk mengompresi *file* teks berformat *.docx tersebut. Diharapkan dengan memperkecil ukuran dari *file* teks yang ada dapat mengefisiensikan penyimpanan yang ada pada *harddisk.*

2. METODOLOGI PENELITIAN

2.1 Kompresi Data

Kompresi data adalah proses mengkonversikan sebuah input data stream (*stream* sumber, atau data mentah asli) menjadi data *stream* lainnya (*bit stream* hasil, atau *stream* yang telah terkompresi) yang berukuran lebih kecil. Pemampatan merupakan salah satu cara dalam ilmu komputer yang bertujuan untuk memadatkan data sehingga hanya memerlukan ruang penyimpanan yang lebih kecil[5]. Kompresi memiliki 2 teknik, antara lain:

1. Kompresi *lossy* adalah kompresi data yang menghasilkan *file* data hasil kompresi yang tidak dapat dikembalikan menjadi *file* data sebelum dikompresi secara utuh. Ketika data hasil kompresi di-decode kembali, data hasil decoding tersebut tidak dapat dikembalikan menjadi sama dengan data asli tetapi ada bagian data yang hilang.
2. Kompresi *lossless* adalah data yang menghasilkan *file* data hasil kompresi yang dapat dikembalikan menjadi *file* data asli sebelum dikompresi secara utuh tanpa perubahan apapun. Kompresi jenis ini ideal untuk kompresi teks. Algoritma yang termasuk dalam metode kompresi lossless diantaranya adalah dictionary coding dan huffman coding.

Proses kompresi adalah proses *encoding* yang menghasilkan data yang sudah dikompresi yang disebut aliran data *encoded*. Sebaliknya aliran data yang telah dikompresi harus dilakukan proses dekompresi untuk menghasilkan kembali aliran data yang asli. Karena proses dekompresi menghasilkan *decoding* dari aliran data yang sudah dikompresi maka hasilnya adalah aliran data *decoded*. Tingkat pengurangan data yang dicapai sebagai hasil dari proses kompresi disebut rasio kompresi. Rasio ini merupakan perbandingan antara panjang data string asli dengan panjang data string yang sudah dikompresi, seperti dituliskan dalam persamaan berikut:

$$\text{Rasio} = \frac{\text{ukuran file asli}}{\text{ukuran file terkompresi}}$$

Jika dinyatakan dalam prosentase maka dituliskan dalam persamaan berikut:

$$\text{P} = \left(\frac{1 - \text{ukuran file terkompresi}}{\text{ukuran file asli}}\right) \times 100\%$$

Yang berarti ukuran *file* berkurang sebesar P (dalam persentase) dari ukuran semula. Semakin tinggi rasio tingkat suatu teknik kompresi data maka semakin efektif teknik kompresi tersebut[6].

2.2 Algoritma *Taboo Codes*

Algoritma Taboo Codes memiliki pendekatan terhadapat panjangnya suatu variable. Penggagas Algoritma *Taboo Codes* adalah Steven Pigeon. Prinsip dari Algoritma Taboo yaitu untuk memilih bilangan positif n dan membalikkan pola n untuk mengindikasikan akhir dari kode tersebut[1].

Terdapat dua tipe yang yang ada dalam Algoritma Taboo diantaranya:

1. Tipe pertama dari Algoritma *Taboo Codes* yaitu block-based dan memiliki panjang berupa kelipatan dari n. Block based Algoritma *Taboo Codes* dari integer adalah string dari n- bit blocks, dimana nilai n dipilih oleh user dan blok terakhir memiliki sedikit pola taboo yang tidak dapat mucul pada blok lainnya. n-bit blok memilik nilai 2n , jika salah satu nilai dicadangkan, masing-masing sisa kode blok dapat memiliki salah satu yang ada pada pola bit 2n -1.
2. Tipe kedua dari Algoritma *Taboo Codes* yaitu panjang total kode ini tidak terbatas pada kelipatann. Tipe ini dikatakan unconstrained dan ditunjukkan bahwa tipe ini memiliki relasi terhadap nomor fibonaci ke-n. Adapun tabel dari algoritma *Taboo Codes* dapat dilihat ditabel ini :

Tabel 1. Pola *Taboo Code*s [1].

m	Code	m	Code	m	Code	m	Code
0	01 00	**4**	01 10 00	**8**	10 11 00	**12**	01 01 01 00
1	10 00	**5**	01 11 00	**9**	11 01 00	**13**	01 01 10 00
2	11 00	**6**	10 01 00	**10**	11 10 00	**14**	01 01 11 00
3	01 01 00	**7**	10 10 00	**11**	11 11 00	**…**	

2.3 File Teks

File teks merupakan suatu *file* yang berisi informasi – informasi dalam bentuk teks. Data yang berasal dari dokumen pengolahan kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh yang termasuk dalam masukan data teks yang terdiri dari karakter, angka dan tanda baca. *Input* dan *output* data teks dipresentasikan sebagai set karakter atau sistem kode yang dikenal oleh sistem komputer. Ada tiga jenis set karakter umum yang dapat digunakan untuk *input* dan *output* pada komputer, yaitu ASCII, EBCDIC, dan Unicode. ASCII (*American Code for Information Interchange*) adalah suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode, tetapi ASCII bersifat universal. Kode ASCII 8 bit memiliki komposisi bilangan biner yang dimulai dari 00000000 sehingga 11111111.

Total kombinasi yang dihasilkan adalah 256, dimulai dari kode 0 hingga 255 yang terdapat dalam sistem bilangan desimal. EBCDIC (*Extended Binary Codec Decimal Interchange Code*) merupakan salah satu set karakter yang diciptakan oleh komputer merk IBM. EBCDIC terdiri dari 256 karakter yang masing – masing karakternya berukuran 8 bit. Adanya keterbatasan pada kode ASCII dan EBCDIC, maka dibuat standar kode internasional baru yang merupakan kode 16 bit yang disebut Unicode. Unicode merupakan suatu standar industri yang dirancang untuk mengizinkan teks dan simbol dari semua tulisan di dunia agar dapat ditampilkan dan dimanipulasi secara konsisten oleh komputer[7].

3. HASIL DAN PEMBAHASAN

Algoritma *Taboo Codes* merupakan algoritma kompresi yang bersifat *lossless*, algoritma *Taboo Codes* mengkompresi *file* dengan frekuensi karakter pada objek yang akan dilakukan kompresi. Algoritma *Taboo Codes* bekerja dengan cara mengurutkan karakter yang sering muncul dan menggantikan bilangan biner dari setiap karakter menjadi nilai kebenaran berdasarkan tabel kebenaran algoritma *Taboo Codes*.

Permasalahan yang diangkat dari penelitian ini adalah untuk mendapatkan algoritma kompresi yang lebih baik yang akan digunakan untuk memperkecil ukuran *file* teks berformat *.docx dengan menggunakan algoritma *Taboo Codes*. Pada penelitian ini prosedur kerjanya adalah dengan cara mengkompresi *file* teks dan *file* teks yang sudah dikompresi kemudian didekompresi lagi untuk mengembalikan *file* teks ke format semula. Adapun diagram kompresi dan dekompresi *file* teks dapat dilihat dari gambar di bawah ini :

Gambar 1. Prosedur Kompresi Dekompresi *File* Teks.

3.1 Penerapan Algoritma *Taboo Codes*

Penelitian ini akan membahasa 2 proses utama yaitu proses kompresi dan proses dekompresi, dalam penelitian ini penulis akan mengkompresi *file* teks berformat *.docx menggukan algoritma *Taboo Codes*, algoritma ini merupakan algoritma yang bersifat *lossless* sehingga *file* yang dikompresi tidak hilang banyak dan ketika didekompresi *file* akan kembali ke ukuran semula sehingga tidak mempengaruhi isi *file* yang dikompresi dan tidak menghilangkan teks.

Algoritma *Taboo Codes* memiliki pendekatan terhadapat panjangnya suatu variabel. Penggagas Algoritma *Taboo Codes* adalah Steven Pigeon. Prinsip dari Algoritma *Taboo Codes* yaitu untuk memilih bilangan positif n dan membalikkan pola n untuk mengindikasikan akhir dari kode tersebut. Terdapat dua tipe yang yang ada dalam Algoritma *Taboo Codes* diantaranya:

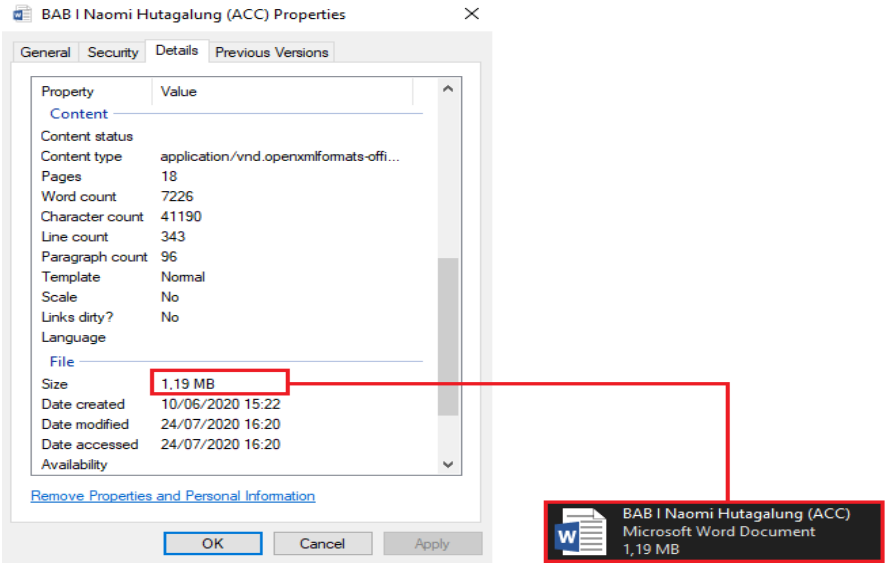
1. Tipe pertama dari Algoritma *Taboo Codes* yaitu block-based dan memiliki panjang berupa kelipatan dari n. Block based Algoritma *Taboo Codes* dari integer adalah string dari n- bit blocks, dimana nilai n dipilih oleh user dan blok terakhir memiliki sedikit pola taboo yang tidak dapat mucul pada blok lainnya. n-bit blok memilik nilai 2 n , jika salah satu nilai dicadangkan, masing-masing sisa kode blok dapat memiliki salah satu yang ada pada pola bit 2 $n - 1$.
2. Tipe kedua dari Algoritma *Taboo Codes* yaitu panjang total kode ini tidak terbatas pada kelipatann. Tipe ini dikatakan *unconstrained* dan ditunjukkan bahwa tipe ini memiliki relasi terhadap nomor fibonaci ke-n. Adapun kode kebenaran dari Algoritma *Taboo Codes*, dapat dilihat dari tabel dibawah ini :

Tabel 2. Kode Algoritma *Taboo Codes*[1]

M	Code
0	01 00
1	10 00
2	11 00
3	01 01 00
4	01 10 00
5	01 11 00
6	10 01 00
7	10 10 00

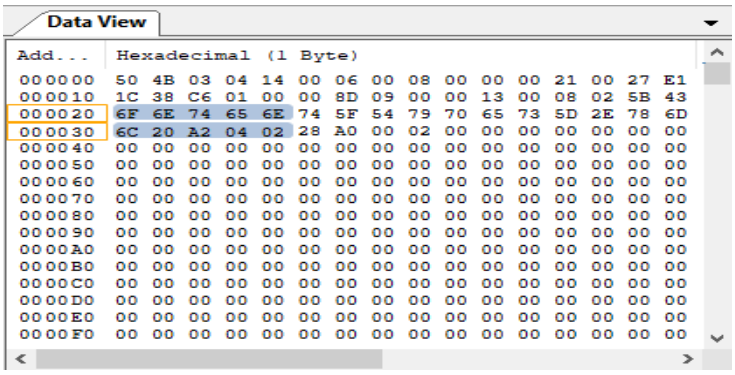
M	Code
8	10 11 00
9	11 01 00
10	11 10 00
11	11 11 00
12	01 01 01 00
13	01 01 10 00
14	01 01 11 00

File yang akan di kompresi dalam penelitian ini adalah *file* berformat *.docx, maka dari itu penulis memutuskan untuk mengambil sample yang akan dikompresi adalah hasil penelitian penulis sendiri. Berikut adalah tampilan detail dari dokumen yang akan dikompresi.



Gambar 2. Contoh *File* *.docx yang akan dikompresi.

Dalam penelitian ini, penulis akan mengkompresi *file* teks berformat *.docx yang berisikan hasil skripsi BAB I penulis. Sebelum mengkompresi contoh *file* dokumen diatas tahap awal adalah mencari nilai bilangan heksadesimal dari *file* contoh menggunakan apilikasi *binary viewer*, hasil bilangan heksadesimal yang telah didapat akan diseleksi/diambil beberapa baris nilai heksadesimal. Penulis mengambil *sample* bilangan heksadesimal sebanyak 2 kolom dan 5 baris sehingga total 10 nilai bilangan heksadesimal.



Gambar 3. *Sample* Bilangan Heksadesimal

Selanjutnya *sample* bilangan heksadesimal yang sudah dipilih akan diurutkan berdasarkan frekuensi kemunculan dari setiap karakter, dan setiap bilangan heksadesimal akan diubah menjadi bilangan biner basis 8.

Tabel 3. *String* karakter yang belum dikompresi

m	Karakter	Frekuensi	ASCII (binary)	Bit	Frek×Bit
1	6F	1	01101111	8	8
2	6E	2	01101110	8	16
3	74	1	01110100	8	8
4	65	1	01100101	8	8

m	Karakter	Frekuensi	ASCII (binary)	Bit	Frek×Bit
5	6C	1	01101100	8	8
6	20	1	00100000	8	8
7	A2	1	10100010	8	8
8	04	1	00000100	8	8
9	02	1	00000010	8	8
		Total Bit			80

Berdasarkan tabel kode ASCII diatas, satu karakter bernilai delapan bit bilangan biner. Sehingga 22 karakter pada *string* mempunyai nilai biner sebanyak 80 bit. Algoritam *Taboo Codes* bekerja dengan memulai *sorting* jumlah frekuensi kemunculan setiap karakter yang akan di kompresi. Adapun *string* bit sebelum dikompresi adalah sebagai berikut : 01101111 01101110 01110100 01100101 01101110 01101100 00100000 10100010 00000100 00000010.

Setelah menentukan kemunculan setiap karakter dan mencari nilai bilangan biner dari setiap karakter, langkah selanjutnya adalah mengurutkan setiap karakter berdasarkan frekuensi kemunculan yang paling banyak pada setiap karakter. Setelah diurutkan berdasarkan frekuensi terbanyak, bilangan biner sebelum dikompresi pada setiap karakter diganti dengan kode kebeneran dari algoritma *taboo codes* yang diurutkan berdasarkan frekuensi kemunculan.

Tabel 4. Data setelah dikompresi menggunakan algoritma *Taboo Codes*

N	Char	Biner	Bit	Frek	Bit × Frek
1	6E	0100	4	2	8
2	6F	1000	4	1	4
3	74	1100	4	1	4
4	65	010100	6	1	6
5	6C	011000	6	1	6
6	20	011100	6	1	6
7	A2	100100	6	1	6
8	04	101000	6	1	6
9	02	101100	6	1	6
		Total Bit			52

Setelah kode berhasil di *encode* berdasarkan kode kebenaran agloritma *Taboo Codes*, tahap selanjutnya adalah menyusun kembali kode-kode yang telah dihasilkan dari proses kompresi sesuai dengan posisi karakter pada *string* seperti pada tabel 6.

Tabel 5. *String bit* hasil kompresi dengan algoritma *Taboo Codes*

6F	**6E**	**74**	**65**	**6E**
1000	0100	1100	010100	0100
6C	**20**	**A2**	**04**	**02**
011000	011100	100100	101000	101100

Setelah setiap karakter disusun kembali menggunakan kode kebenaran algoritma *taboo codes*, tahap selanjutnya menyusun kembali hasil kompresi setiap karakter, sehingga diperoleh *string bit* sebagai berikut :

1000 0100 1100 010100 0100 011000 011100 100100 101000 101100.

Dalam hal penyimpanan data digunakan aturan 8 bit, sehingga untuk menyesuaikan data dengan aturan penyimpanan *string bit* sebanyak kekurangan dari kelipatan 8. Setelah itu, untuk mengetahuii banyaknya *padding* dan *flagging* yang akan ditambahkan pada *string bit* kompresi, maka total dari keseluruhan kompresi *string bit* akan dibagi 8. Jika sisa bagi total seluruh *string bit* terhadap 8 adalah 0 maka akan ditambahkan *string bit* 00000001, sedangkan jika sisa bagi panjang *string bit* terhadap 8 tidak habis dibagi, maka akan ditambahkan *padding* dan *flagging*.

Tabel 6. Penambahab *padding* dan *flagging*

Padding	Flagging
$7 - n$ + "1" 7 – 4 + "1" = 0001	$9 - n$ 9 – 4 = 5 00000101

Tabel 8. Hasil Penambahan *padding* dan *flagging*

sehingga diperoleh *string bit* sebagai berikut : 1000 0100 1100 010100 0100 011000 011100 100100 101000 101100 0001 00000101

Tabel 8. menampilkan total dari keseluruhan bit yang telah ditambahkan *padding* dan *flagging*, panjang *string bit* awal adalah 52 setelah ditambahkan *padding* dan *flagging* panjang total keseluruha *string bit* menjadi 64 bit. Selanjutnya total seluruh *string bit* akan dibagi menjadi 8 bit dalam satu kelompok *string bit*, dan *string bit* yang sudah dikelompokkan

akan diubah kedalam bilangan desimal, bilangan desimal yang diperoleh dari bilangan kelompok biner selanjutnya akan diubah menjadi berbagai karakter unik yang terdapat pada tabel ASCII.

Tabel 9. Pengelompokan bit dan pencarian karakter unik

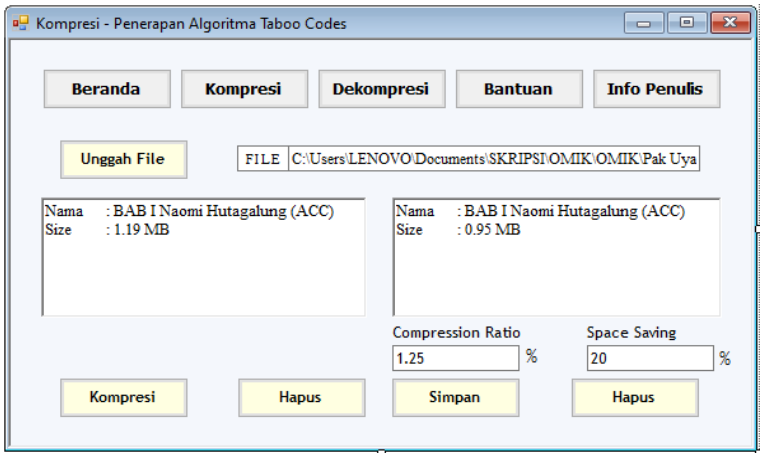
No	String bit hasil kompresi	Bilangan Desimal	Karakter unik
1	10000100	132	Ä
2	11000101	197	┼
3	00010001	17	◄
4	10000111	135	Ç
5	00100100	36	$
6	10100010	162	Ó
7	11000001	193	♪
8	00000101	5	♣

Berdasarkan hasil kompresi *file* teks diatas menggunakan algoritma *Taboo Codes*, maka didapatkan hasil kompresi *string bit* yang lebih sedikit yaitu sebanyak 64 *string bit* dari *string bit* awal sebanyak 80 *string bit*, sebagai berikut : 10000100110001010001000110000111001001001010001011000001000000101.

1. *Compression Ratio*
   Salah satu parameter pengukur kinerja algoritma adalah hasil perbandingan antara ukuran data sebelum dikompresi dengan ukuran data setelah dikompresi pada data yang terkompresi yang biasa disebut *Compression Ratio* dengan cara kerja sebagai berikut :
   $$CR = \frac{ukuran\ data\ sebelum\ dikompresi}{ukuran\ data\ setelah\ dikompresi} \times 100\%$$
   Hasil perhitungan yang didapatkan dari *sample file* *.docx yang dikompresi menggunakan algoritma *taboo codes* adalah sebagai berikut:
   $$CR = \frac{ukuran\ data\ sebelum\ dikompresi}{ukuran\ data\ setelah\ dikompresi} \times 100\%$$
   $$CR = \frac{80\ bit}{64\ bit} \times 100\%$$
   $$CR = 1{,}25\ \%$$
2. *Space Saving*
   Space Saving merupakan besarnya ruang penyimpanan yang akan dihemat setelah proses kompresi dilakukan. *Space Saving* dirumuskan sebagai persentase selisih data awal sebelum dikompresi dengan hasil data yang telah dikompresi.
   $$SS = 1 - \frac{ukuran\ file\ kompresi}{ukuran\ file\ asli} \times 100\%$$
   $$SS = 1 - \frac{64}{80} \times 100\%$$
   $$SS = (1 - 0.8) \times 100\%$$
   $$SS = 0.2 \times 100\%$$
   $$SS = 20\%$$

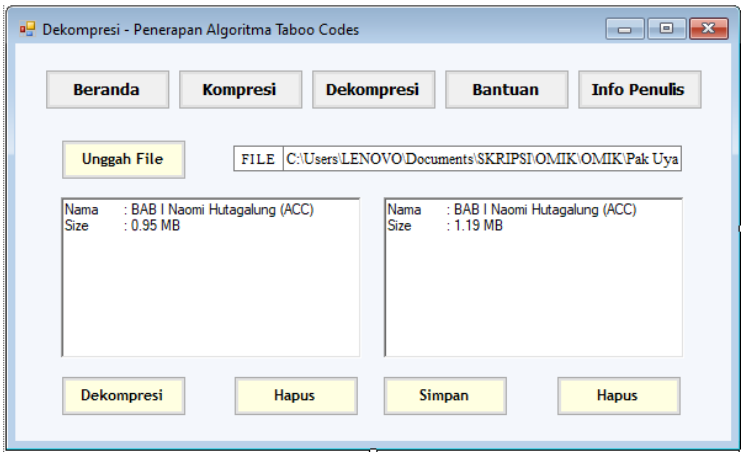
3.2 Implementasi

Proses kompresi pada pengujian akan dimulai dengan mencari *file* teks berformat *.docx yang akan dikompresi, kemudian klik tombol kompresi untuk memulai proses kompresi, selanjutnya akan muncul hasil kompresi dari *file* teks.



Gambar 4. Tampilan Proses Kompresi

Proses dekompresi pada pengujian akan berfungsi untuk mengolah kembali *file* teks yang sudah dikompresi agar kembali menjadi ukuran *file* asli. Berikut adalah tampilan proses dekompresi.



Gambar 5. Tampilan Proses Dekompresi

Proses pengujian menggunakan sistem telah dilakukan terhadap *file* teks berformat *.docx menggunakan algoritma *Taboo Codes* dan didapatkan hasil ukuran *file* yang telah dikompresi lebih kecil ukurannya dibandingkan dengan ukuran *file* asli. Pada hasil pengujian ini akan ditampilkan beberapa contoh *file* teks berformat *.docx yang telah dikompresi.

Tabel 10. Hasil Kompresi

No	File Original		Hasil Kompresi		
	Nama File	Ukuran	Nama File	Ukuran	Rasio
1.	BAB I Naomi Hutagalung (ACC).docx	1.19 MB	BAB I Naomi Hutagalung.docx	0.95 MB	1.25
2.	BAB II (Naomi Hutagalung).docx	1.18 MB	BAB II (Naomi Hutagalung).docx	0.94 MB	1.25
3.	Laporan Kerja Praktek.docx	1.75 MB	Laporan Kerja Praktek.docx	1.40 MB	1.25

4. KESIMPULAN

Berdasarkan hasil dari pengujian yang telah dilakukan, penulis mengambil beberapa kesimpulan dari hasil penelitian yang dilakukan, adapun beberapa kesimpulan yang diambil sebagai Penerapan algoritma *Taboo Codes* untuk mengkompresi *file* teks berformat *.docx, dilakukan dengan membaca nilai hekasadesimal menggunakan aplikasi *binary viewer*, lalu diubah menjadi nilai biner baru dan dicocokan dengan tabel kebenaran dari algoritma *Taboo Codes* dan diubah menjadi karakter baru. Hasil analisa yang didapatkan saat mengkompresi *file* teks menggunakan algoritma *Taboo Codes* berhasil mengkompresi sample *file* teks yang ukuran sebelumnya sebersar 1,19 MB dan setelah dikompresi ukurannya menjadi 0,95 MB. Adapun perancangan aplikasi kompresi *file* teks dengan menggunakan bahasa pemrograman *Microsoft Visual Basic 2008* sangat cocok untuk merancang aplikasi kompresi *file* teks berformat .*docx dengan menerapkan algoritma *Taboo Codes* , dan didalam sistem yang dirancang memiliki beberapa *form* antara lain *form* beranda, *form* kompresi, *form* dekompresi, *form* bantuan dan *form* info penulis.

REFERENCES

[1] D. Salomon, *Data Compression The Complete Reference FourthEdition*, vol. 53, no. 9. 2007.
[2] B. Fahrudin, "Penerapan Algoritma Backtracking pada Permainan Capsa Banting," *JURIKOM (Jurnal Ris. Komputer)*, vol. 3, no. 6, pp. 30–33, 2016.
[3] M. R. Irliansyah, S. D. Nasution, and K. Ulfa, "Penerapan Metode Deflate Dan Algoritma Goldbach Codes Dalam Kompresi File Teks," *KOMIK (Konferensi Nas. Teknol. Inf. dan Komputer)*, vol. 1, no. 1, pp. 186–189, 2017.
[4] Sutardi, "Implementasi dan analisis kinerja algoritma shannon- fano untuk kompresi file text," vol. 6, no. 1, pp. 53–60, 2014.
[5] D. Salomon, "handbook of data comression," 2008.
[6] D. Toradmalle, J. Muthukuru, and B. Sathyanarayana, "Compression techniques: Key to effective data transmission," *Int. J. Recent Technol. Eng.*, vol. 8, no. 3, pp. 3178–3181, 2019, doi: 10.35940/ijrte.C4906.098319.
[7] J. Enterprise, *Rahasia Manajemen File*. Jakarta, 2010.
[8] S. H. Mulyani, "Sistem Informasi E-Document Pada Badan Penjamin Mutu Akademik Universitas Respati Yogyakarta," *J. Teknol. Inf.*, vol. IX, p. 18, 2014.
[9] P. Sulistyorini, "Pemodelan Visual dengan Menggunakan UML dan Rational Rose," vol. XIV, no. 1, pp. 23–29, 2009.