Majalah Ilmiah INTI, Volume 12, Nomor 3, September 2017
ISSN 2339-210X

PENERAPAN ALGORITMA HUFFMAN DAN SHANNON-FANO DALAM PEMAMPATAN FILE TEKS

Aldi Medana Pratama[1], Nelly Astuti Hasibuan[2], Efori Buulolo[2]

[1] Mahasiswa Teknik Informatika STMIK Budi Darma, Medan, Indonesia
[2] Dosen Tetap STMIK Budi Darma, Medan, Indonesia
[1,2] Jl. Sisingamangaraja No 338 Simp Limun Medan, Indonesia

ABSTRAK

Penerapan merupakan proses untuk mengetahui atau memahami nilai bobot suatu zat, kata, dan lain sebagainya. Disini penerapan dapat diartikan sebagai penentuan nilai ukuran file yang akan diterapkan pada suatu algoritma. Sebuah media penyimpanan seperti floppy disk, hard disk dan CD (Compact Disc) mempunyai kapasitas yang terbatas.Pada dasar ke dua algoritma ini mempunyai cara kerja yang sama. Dimulai dengan pengurutan karakter berdasarkan frekuensinya, pembentukan pohon biner dan diakhiri dengan pembentukan kode. Pada algoritma Huffman, pohon biner dibentuk dari daun hingga akar dan disebut dengan pembentukan pohon dari bawah ke atas. Sebaliknya, pada Shannon-Fano pohon biner dibentuk dari akar hingga daun dan disebut dengan pembentukan pohon dari atas ke bawah

Kata Kunci: Algoritma Huffman, Algoritma Shannon-Fano, File Teks.

I. PENDAHULUAN

Pemampatan data *(data compression)* merupakan suatu masalah dan menjadi salah satu kajian di dalam ilmu komputer yang bertujuan untuk mengurangi ukuran *file* sebelumnya untuk menyimpan atau memindahkan data tersebut ke dalam media penyimpanan *(storage device)*. Media penyimpanan seperti *floppy disk, harddisk* disimpan pada media penyimpanan semakin bertambah dan berukuran besar, maka media penyimpanan tidak dapat menyimpan data tersebut karena melebihi kapasitas. Untuk mengatasi masalah ini pemampatan sangat dibutuhkan.

Metode pertama yang muncul untuk pemampatan data adalah Shannon-Fano coding. Shannon dan Fano (1948) mengembangkan algoritma yang menghasilkan sebuah kode dengan jumlah bit yang lebih sedikit untuk setiap karakter yang terdapat data dengan membangun suatu pohon kode biner dari pada menggunakan kode yang mempunyai panjang tertentu seperti kode ASCII. Gambaran kode yang lebih pendek ini oleh Shannon disebut kode berpanjang peubah (*variable length code*). Jika frekuensi kemunculan semakin tinggi, maka kodenya semakin pendek begitu juga sebaliknya. Pada tahun 1952 David Huffman memperkenalkan algoritma pemampatan yang dinamakan Huffman coding. Metode ini memakai hampir semua karakteristik dari Shannon-Fano coding. Prinsip kode Huffman adalah karakter yang paling sering muncul di dalam data dikodekan dengan kode yang lebih pendek , sedangkan karakter nyang jarang muncul dikodekan dengan kode yang lebih panjang. Algoritma Huffman membangun pohon biner untuk menghasilkan kode prefix (*prefix code*).

Berdasarkan pengkodean ASCII, string (fungsi rekursif) membutuhkan 15 *byte* atau 120 *bit* (15x8=120 *bit*) untuk menyimpang *string* tersebut dalam media penyimpanan. Ukuran *string* tersebut dapat dikurangi apabila dilakukan proses pemampatan dengan meminimumkan jumlah bilangan *bit*. Untuk melakukan proses pemampatan dapat digunakan sebuah algoritma.

Penulis berkeinginan untuk melakukan suatu studi mengenai penerapan kinerja algoritma pemampatan dalam memampatkan *file*, yaitu Huffman dan Shannon-Fano dalam menghasilkan *file* dengan ukuran yang lebih kecil dari ukuran *file* semula. Pemampatan data dilakukan pada tipe *file teks*. *File teks* adalah *file* yang berisi informasi-informasi yang disajikan dalam bentuk *teks* yang merupakan tipe data yang digunakan untuk menggambarkan atau merepresentasikan kumpulan karakter huruf, angka, dan simbol.

II. TEORITIS

A. Algoritma Huffman

Menurut Darma Putra (2009:279), Prinsip kode Huffman adalah karakter yang paling sering muncul di dalam data dikodekan dengan kode yang jumlah bitnya lebih sedikit, sedangkan karakter yang jarang muncul dikodekan dengan kode yang jumlah bitnya lebih panjang. Algoritma Huffman menggunakan tabel frekuensi kemunculan karakter untuk frekuensi dua buah pohon yang digabungkan. Oleh karena itu, total *cost* pembentukan pohon Huffman adalah jumlah seluruh penggabungan daun-daun. Huffman memberikan sebuah algoritma untuk membangun sebuah kode Huffman dengan masukan *string teksS*={*s1,s2,…,sn*} dan frekuensi kemunculan karakter *F*={*f1, f2 , …, fn*}, dihasilkan keluaran berupa kode *string* biner *C*={*c1,c2,…,cn*} atau disebut kode Huffman.

Langkah-langkah Pemampatan Algoritma Huffman:

1. Data dianalisis dahulu dengan cara membuat tabel frekuensi kemunculan setiap *symboll*

Majalah Ilmiah INTI, Volume 12, Nomor 3, September 2017
ISSN 2339-210X

ASCII, tabel frekuensi tersebut memiliki atribut berupa simbol ASCII dan frekuensi.
2. Dua data yang memiliki frekuensi kemunculan paling kecil dipilih sebagai simpul pertama pada pohon Huffman.
3. Dari dua simpul ini dibuat simpul induk yang mencatat jumlah frekuensi dua simpul pertama.
4. Kemudian dua simpul tersebut dihapus dari table digantikan oleh simpul induk tadi. Simpul ini kemudian dijadikan acuan untuk membentuk pohon.
5. Langkah 3-5 dilakukan berulang-ulang hingga isi tabel tinggal satu saja, data inilah yang akan menjadi simpul bebas atau simpul akar.
6. Setiap simpul yang terletak pada cabang kiri (simpul dengan frekuensi lebih besar) diberi nilai 0 dan simpul yang terletak pada cabang kanan (simpul dengan frekuensi lebih kecil) diberi nilai 1.
7. Pembacaan dilakukan dari simpul akar ke arah simpul daun dengan memperhatikan nilai setiap cabang.

B. Algoritma Shannon-Fano

Menurut Darma Putra, (2009:275) Algoritma Shannon-Fano merupakan suatu metode yang dikenal pertama kali mampu melakukan pengkodean terhadap *symbol* secara efektif adalah metode Shannon-fano. Metode ini dikembangkan secara bersamaan oleh Claude Shannon dari Bell Labs dan RM Fano dariMIT.

Metode ini tergantung pada probabilitas dari setiap *symbol* yang hadir pada suatu data(pesan). Berdasarkan Probabilitas tersebut kemudian dibentuk daftar kode untuk setiap simbol dengan ketentuan sebagai berikut:

1. Setiap simbol berbeda memiliki kode berbeda.
2. Simbol dengan probabilitas kehadiran yang lebih rendah memiliki kode jumlah *bit* yang lebih panjang dan simbol dengan probabilitas yang lebih tinggi memiliki jumlah *bit* yang lebih pendek.
3. Meskipun memiliki panjang kode yang berbeda, simbol tetap dapat didekode secara unik.

C. File Teks

Menurut Sudewa, Ida Bagus Adi(2003:278), *File teks* merupakan *file* yang berisi informasi-informasi yang di sajikan dalam bentuk *teks* yang merupakan kumpulan dari karaktek-karakter atau *string* yang menjadi satu kesatuan sedangkan *string* merupakan tipe data yang digunakan untuk menggambarkan atau mempresentasikan kumpulan karakter (huruf,angka,symbol). Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh masukan data *teks* yang terdiri dari karakter, angka dan tanda baca.

III. ANALISA DAN PEMBAHASAN

A. Pemampatan *File Teks* dengan Algoritma *Huffman*

Untuk melakukan proses pemampatan *file teks*, diberikan sebuah file teks dengan file text.txt yang berukuran 356 *bytes* yang terdiri dari rangkaian *string* dan terdiri dari 27 karakter menggunakan algoritma Huffman sebagai berikut:

"Dalam penyimpanan data, file yang berukuran besar juga membutuhkan ruang penyimpanan yang besar. Agar file berukuran kecil dapat digunakan pemampatan data, yaitu isi file dikodekan sesingkat mungkin sehingga ruang penyimpanan yang dibutuhkan sedikit. Pemampatan data dilakukan dengan mengkodekan setiap karakter di dalam file dengan kode yang lebih pendek."

Masukan: 27 karakter yang telah diurutkan dan table frekuensi

Keluran : kode *Huffman* dengan n kode

1. Pengurutan karakter berdasarkan frekuensi kemunculanya.

Karakter-karakter diurutkan di dalam table secara menarik (*asscending order*).

Tabel 1. Karakter yang telah diurutkan berdasarkan frekuensinya

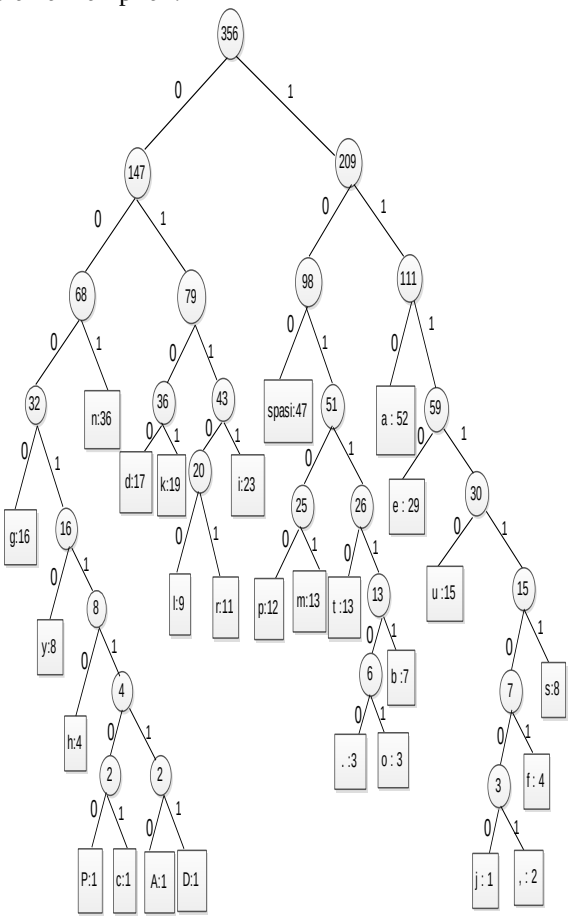
Karakter	Frekuensi	Karakter	Frekuensi
A	1	R	11
D	1	P	12
P	1	M	13
C	1	T	13
J	1	G	15
,	2	D	16
.	3	K	17
O	3	I	19
F	4	E	23
H	4	N	36
B	7	Spasi	47
S	8	A	52
Y	8		
L	9		

2. Pembentukan Pohon Biner

Setiap karakter digambarkan sebagai daun atau pohon bersimpul tunggal. Kemudian gabungan dua daun yang mempunyai frekuensi kemunculan karakter paling kecil untuk membentuk akar. Akar merupakan jumlah dari frekuensi dua daun penyusunannya. Iterasi ini dilakukan hingga terbentuk satu pohon biner. Beri label 0 dan 1 pada setiap sisi pohon biner. Sisi kiri dilabeli dengan 0 dan sisi kanan dilabeli dengan 1. Proses pembentukan pohon biner hingga

Majalah Ilmiah INTI, Volume 12, Nomor 3, September 2017
ISSN 2339-210X

membentuk pohon Huffman dapat dilihat pada halaman lampiran.



Gambar 1. Pohon *Huffman* yang telah Terbentuk

3. Pembentukan Kode *Huffman*

Barisan angka-angka 0 dan 1 pada sisi pohon dari akar ke daun menyatakan kode Huffman untuk karakter yang bersesuain. Telusuri pohon biner dari akar ke daun untuk membentuk kode *Huffman.*

Tabel 2. Kode *Huffman* yang telah terbentuk

Karakter (C)	Frekuensi (f)	Kode Huffman	Panjang Kode (shf)	Total Panjang (f*shf)
A	1	00011110	8	8
D	1	00011111	8	8
P	1	00011100	8	8
C	1	00011101	8	8
J	1	11111000	8	8
,	2	11111001	8	16
.	3	1011100	7	21
O	3	1011101	7	21
F	4	1111101	7	28
H	4	000110	6	24
b	7	101111	6	42
s	8	111111	6	48
y	8	00010	5	40
l	9	01100	5	45
r	11	01101	5	55
P	12	10100	5	60
m	13	10101	5	65
t	13	10110	5	65
u	15	11110	5	75
g	16	0000	4	64
d	17	0100	4	68
k	19	0101	4	76
i	23	0111	4	92
e	29	1110	4	116
n	36	001	3	108
spasi	47	100	3	141
a	52	110	3	156
Total	356		151	1466

Ukuran *file teks* sebelum pemampatan dengan kode ASCII (8 *bit*) adalah 356 *bytes* atau 356 x 8 *bit* = 2848 *bit*

Ukuran *file teks* setelah pemampatan menggunakan algoritma *Huffman* :

Kode *Huffman* = 1466 *bit*

$$= \frac{1466}{151/27}$$

= 262,25

= 262 *bytes*

Dengan demikian, untuk menyimpan *file teks* yang terdiri dari 356 karakter dengan menggunkan algoritma *Huffman* dibutuhkan 262 *bytes*.

File teks tersebut dimampatkan dengan rasio sebesar:

$$\text{Rasio Pemampatan n} = \left(\frac{\text{Hasil Pemampatan}}{\text{Ukuran } \textit{File} \text{ Semula}}\right) * 100\%$$

$$n = \left(\frac{262}{356}\right) * 100\%$$

= 73,59%

Majalah Ilmiah INTI, Volume 12, Nomor 3, September 2017
ISSN 2339-210X

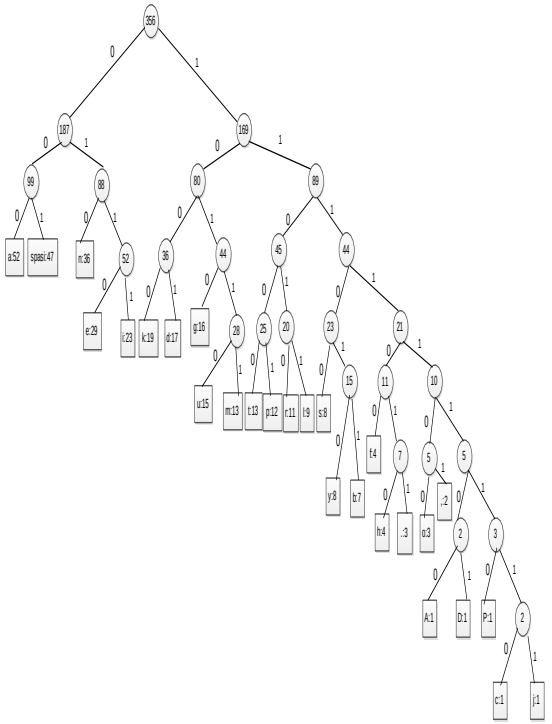
B. Pemampatan *File Teks* dengan Algoritma *Shannon-Fano*

Tabel 3. Karakter yang telah diurutkan berdasarkan frekuensinya

Karakter	Frekuensi	Karakter	Frekuensi
A	52	S	8
Spasi	47	Y	8
N	36	B	7
E	29	F	4
I	23	H	4
K	19	.	3
D	17	O	3
G	16	,	2
U	15	A	1
M	13	D	1
T	13	P	1
P	12	C	1
R	11	J	1
L	9		

2. Pembentukan Pohon Biner

Pohon biner dibentuk dengan pembagian tabel karakter yang telah diurutkan ke dalam dua bagian jumlah frekuensi masing-masing subbagian mendekati atau sama. Berikan label 0 pada subpohon kiri dan label 1 pada subpohon kanan. Ulangi pembagian ini secara rekursif hingga masing-masing karakter menjadi daun kode yang bersesuaian. Proses perulangan pembagian pohon dapat dilihat pada lampiran.



Gambar 2 Pohon *Shannon-Fano* yang telah Terbentuk

2. Pembentukan Kode *Shannon-Fano*

Barisan angka-angka 0 dan 1 pada sisi pohon dari akar ke daun menyatakan kode Shannon-Fano untuk karakter yang bersesuaian. Telusuri pohon biner dari akar ke daun untuk membentuk kode *Shannon-Fano*.

Tabel 4. Kode *hannon-Fano* yang telah terbentuk

Karakter (C)	Frekuensi (f)	Kode Shannon-Fano	Panjang Kode (shf)	Total Panjang (f*shf)
A	52	000	3	156
Spasi	47	001	3	141
N	36	010	3	108
E	29	0110	4	116
I	23	0111	4	92
K	19	1000	4	76
D	17	1001	4	68
G	16	1010	4	64
U	15	10110	5	75
M	13	10110	5	65
T	13	11000	5	65
P	12	11001	5	60
R	11	11010	5	55
L	9	11011	5	45
S	8	11100	5	40
Y	8	111010	6	48
B	7	111011	6	42
F	4	111100	6	24
H	4	1111010	7	28
.	3	1111011	7	21
O	3	1111100	7	21
,	2	1111101	7	14
A	1	11111000	8	8
D	1	11111101	8	8
P	1	11111110	8	8
C	1	111111110	9	9
J	1	111111111	9	9
Total	356		152	1466

Ukuran *file teks* setelah pemampatan menggunakan algoritma *Shannon-Fano*:

Majalah Ilmiah INTI, Volume 12, Nomor 3, September 2017
ISSN 2339-210X

kode *Shannon-Fano* = 1466 bit

$$= \frac{1466}{152/27}$$

= 260,39

= 260 *bytes*

Dengan demikian, untuk menyimpan *file teks* yang terdiri dari 356 karakter dengan menggunakan algoritma *Shannon-Fano* dibutuhkan 260 *bytes*.
File teks tersebut dimampatkan dengan *rasio* sebesar:

$$\text{Rasio Pemampatan } n = \frac{\text{Hasil Pemampatan}}{\text{Ukuran } \textit{File} \text{ Semula}} * 100\%$$

$$n = \left(\frac{260}{356} \right) * 100\%$$

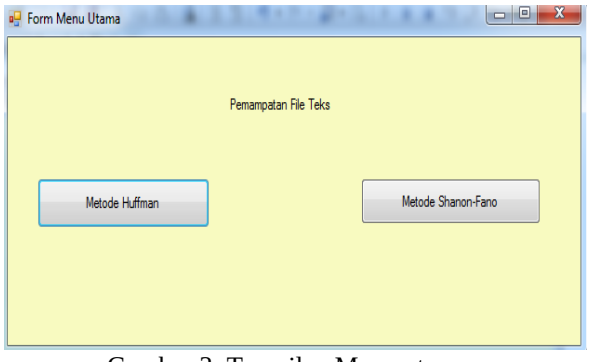
= 73,03%

IV. IMPLEMENTASI

Pada bab ini akan dilakukan implementasi dan pengujian terhadap sistem yang baru. Tahapan ini dilakukan setelah perancangan selesai dilakukan dan selanjutnya akan diimplementasikan pada bahasa pemrograman yang akan digunakan. Setelah implementasi maka dilakukan pengujian sistem yang baru dimana akan dilihat kekurangan-kekurangan pada aplikasi yang baru untuk selanjutnya diadakan pengembangan sistem.

1. Tampilan Menu Utama

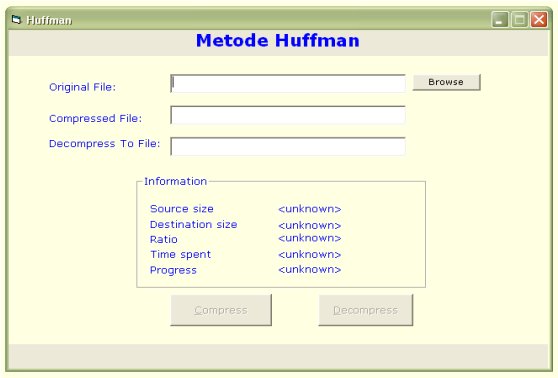
Saat pertama kali aplikasi penerapan *file teks* dijalankan, maka akan tampil tombol Metode Huffman dan Metode Shannon-Fano.



Gambar 3. Tampilan Menu utama

2. Tampilan Halaman Menu Utama

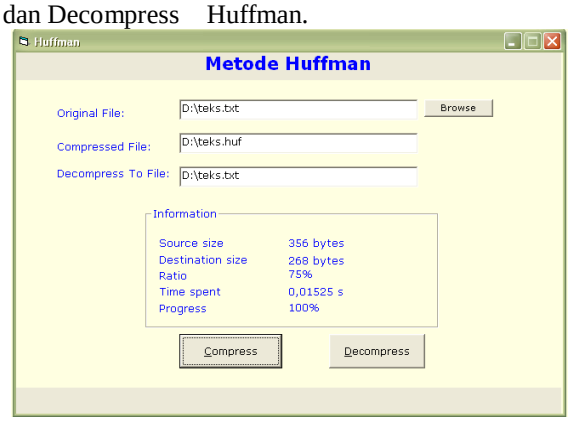
Pada halaman pertama terdapat beberapa menu atara lain form untuk memampatkan *file* dengan Algoritma Huffman dan Algoritma Shannon-Fano.



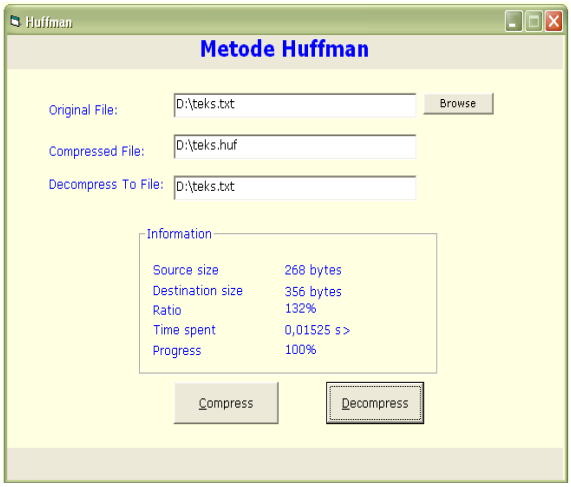
Gambar 4. Halaman Utama

3. Tampilan Halaman Hasil Algoritma Huffman

Pada halaman ini merupakan hasil halaman Compress dan Decompress Huffman.



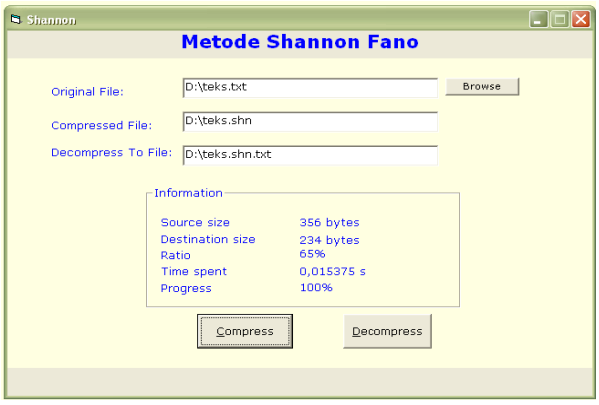
Gambar 5. Hasil *Compress File Teks* Dengan Algoritma Huffman



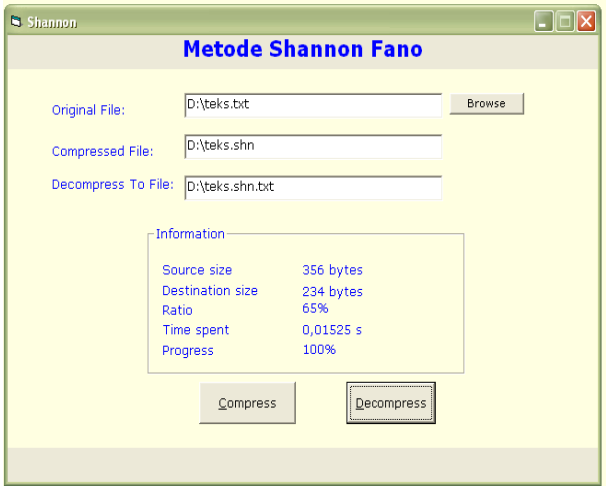
Gambar 6. Hasil *Decompress File Teks* Dengan Algoritma Huffman

4. Tampilan Halaman Hasil Algoritma Shannon-Fano

Pada halaman ini merupakan hasil halaman Compress dan Decompress Shannon-Fano.



Gambar 7. Hasil *Compress File Teks* Dengan Algoritma Shannon-Fano



Gambar 8. Hasil *Decompress File Teks* Dengan Algoritma Shannon-Fano

V. KESIMPULAN

Berdasarkan dari hasil penelitian yang telah dilakukan, maka dapat disimpulkan yaitu:

1. Pada dasarnya pengolahan *file teks* melalui operasi pemampatan dengan menggunakan algoritma Huffman dan Shannon-Fano mempunyai cara kerja yang sama. Dimulai dengan pengurutan karakter berdasarkan frekuensinya, pembentukan pohon biner dan diakhiri dengan pembentukan kode. Pada algoritma Huffman, pohon biner dibentuk dari daun hingga akar dan disebut dengan pembentukan pohon dari bawah ke atas. Sebaliknya, pada Shannon-Fano pohon biner dibentuk dari akar hingga daun dan disebut dengan pembentukan pohon dari atas ke bawah.
2. Berdasarkan hasil analisa terhadap algoritma Huffman dan Shannon-Fano dalam pemampatan *file teks* yang baik adalah Shannon-Fano namun tidak berarti algoritma Shannon-Fano adalah algoritma yang baik untuk memampatkan file-file lain.

REFERENCE

[1] Abdul Kadir dan Terra Ch. Triwahyuni 2013. ”Pengantar Teknologi Informasi” :Dasar Sistem Komputer. Yogyakarta : Andi Offset.

[2] Abdul Kadir 2013.Pengenalan Algoritma :Flowchart. Yogyalarta : Andi Offset.

[3] Rahmat Pryanto 2009. Langsung Bisa Visual Basic.NET 2008: Pendahuluan. Yogyakarta Andi Offset.

[4] Rosa AS – M Shalahuddin 2011. Rekayasa Perangkat Lunak.: “Pemodelan Dan UML”.Bandung 2011

[5] Darma Putra 2010. Pengolahan Citra Digital .: “Kompresi Data Citra” Yogyakarta :Andi Offset

[6] Sumber Jurnal Volume 5 No. 2, November 2009 (Nita Cristina, Sri Suwarno, R. Gunawan Sentosa, Jurusan Teknik Informatika Fakultas Teknik Kristen Duta Wacana, Yogyakarta.)

[7] Haryanto et al. (2017) ‘Internet Protocol Security as the Network Cryptography System’, International Journal of Scientific Research in Science and Technology, 3(6), pp. 223–226.

[8] Nasution, S. D. et al. (2017) ‘Data Security Using Vigenere Cipher and Goldbach Codes Algorithm’, International Journal of Engineering Research & Technology (IJERT), 6(1), pp. 360–363.

[9] Nasution, S. D. and Mesran (2016) ‘Goldbach Codes Algorithm for Text Compression’, IJournals: International Journal of Software & Hardware Research in Engineering, 4(December), pp. 43–46.

[10] Yogie Adriasan, “*Penerapan Algoritma Huffman Dalam Pemampatan File Teks*”, Vol. VII, No. 1, 2013

[11] T.Sutoyo,S.Si.,M.Kom, “*Teori Pengolahan Citra Digital*”, Penerbit Andi, Yogyakarta, Edisi 1, 2009

[12] Josua Marinus Silaen, “*Studi Perbandingan Algoritma Huffman Dan Shannon-Fano Dalam Pemampatan File Teks*”, Vol. VII, No. 1, 2014

[13] Yuni Sugiarti, “*Analisis & Perancangan UML Generated VB. 6*”, Penerbit Graha Ilmu, Yogyakarta, Edisi 1, 2013

[14] Muhammad Sadeli, 2009, Pemrograman Database dengan Visual Basic .NET 2008 untuk Orang Awam, MAXIKOM, Palembang.