PERBANDINGAN PENGODEAN TEKS MENGGUNAKAN ALGORITME HUFFMAN DAN ALGORITME HUFFMAN WEAVEN HANKAMER PADA APLIKASI MATLAB

Muhammad A. Rofi'udin[1], Beby H.A Manafe[2], Amin A. Maggang[3]

[1,2,3)] Fakultas Sains dan Teknik, Program Studi Teknik Elektro, Universitas Nusa Cendana
Jln.Adisucipto - Penfui, Telp. 0380-881597, Fax. 0380-881557
Email : abizarofi@gmail.com,bebymanafe@staf.undana.ac.id, amin_maggang@staf.undana.ac.id

Info Artikel

Diterima Agu 29, 2022
Direvisi Sep 20, 2022
Disetujui Okt 27, 2022

ABSTRACT

Huffman Algorithm and Huffman Weaven-Hankamer Algorithm are the two-source coding algorithms applied in general for text coding. Both algorithms require more time to complete the processes as they have many stages. Therefore, this research aimed to create a user interface using MATLAB that can simulate the process of both algorithms and provide highly accurate results faster. There were two scenarios for the experiment. Both approaches were conducted eight times using the same number of characters, 15 to 22 thousand. ELSE character was also contained in the information sources but is different in number compared to an ordinary character. The first experiment used more ELSE characters, whereas the second applied lesser ELSE characters compared to the ordinary ones. The results showed that the Huffman algorithm is more efficient in the first scenario than Huffman Weaven-Hankamer, which was a 2.24 percent average difference. Although the Huffman algorithm still performed better in the second experiment, the difference in average efficiency was smaller, 1.48 percent, compared to the first approach. In addition, the results also showed that the Huffman weaven Hankamer Tree was simpler than the Huffman tree.

Keywords: *Text Coding, Huffman Algorithm and Huffman Weaven Hankamer Algorithm.*

ABSTRAK

Algoritme Huffman dan Algoritme Huffman Weaven-Hankamer adalah dua algoritme pengodean sumber yang diterapkan secara umum untuk pengodean teks. Kedua algoritme ini membutuhkan lebih banyak waktu untuk menyelesaikan prosesnya karena memiliki banyak tahapan. Oleh karena itu, penelitian ini bertujuan untuk membuat user interface menggunakan MATLAB yang dapat menyimulasikan proses kedua algoritme dan mampu memberikan hasil yang sangat akurat dengan lebih cepat. Ada dua skenario pada penelitian ini. Terdapat delapan kali percobaan pada masing-masing skenario dengan menggunakan jumlah karakter yang sama yaitu 15 sampai 22 ribu. Karakter ELSE juga terdapat dalam sumber informasi yang digunakan, namun jumlahnya berbeda dibandingkan dengan karakter biasa. Skenario pertama menggunakan lebih banyak karakter ELSE, sedangkan skenario kedua menggunakan karakter ELSE yang lebih sedikit dibandingkan dengan eksperimen biasa. Hasil penelitian menunjukkan bahwa algoritme Huffman lebih efisien pada skenario pertama dibandingkan dengan Huffman Weaven-Hankamer, dimana selisih rata-ratanya 2,24 persen. Meskipun algoritme Huffman masih tampil lebih baik pada percobaan kedua, perbedaan efisiensi rata-rata nya lebih kecil, 1,48 persen, jika dibandingkan dengan pendekatan pertama. Selain itu, hasil juga menunjukkan bahwa pohon algoritme Huffman Weaven Hankamer lebih sederhana dibandingkan dengan pohon algoritme Huffman.

Kata kunci: *pengodean teks, algoritme huffman, algoritme huffman weaven hankamer.*

Penulis Korespondensi:
Muhammad A. Rofi'udin
Program Studi Teknik Elektro Fakultas Sains dan Teknik,
Universitas Nusa Cendana,
Jl. Adisucipto Penfui - Kupang.
abizarofi@gmail.com

1. PENDAHULUAN

Terdapat dua proses pada pengodean sumber yaitu *Encoding* yang terletak di pengirim untuk mengubah pesan menjadi kode-kode biner dan *Decoding* yang terletak di penerima untuk mengubah kode-kode biner yang diterima menjadi pesan. Beberapa contoh pengodean teks pada sumber informasi adalah algoritme Shanon, algoritme Huffman dan algoritme Huffman Weaven Hankamer. Namun diantara algoritme tersebut, yang paling ideal digunakan untuk pengodean teks adalah algoritme Huffman [1].

Algoritme Huffman ditemukan oleh David A. Huffman pada 1952, algoritme Huffman termasuk dalam algoritme *variable codeword length*, ini berarti simbol individual (karakter dalam sebuah sumber teks) digantikan oleh kode biner. Jadi simbol yang muncul cukup banyak dalam teks akan memberikan kode biner yang pendek sementara simbol yang jarang dipakai akan mempunyai kode biner yang lebih panjang [2].

Algoritme Huffman Weaven Hankamer yaitu algoritme yang dimodifikasi dari algoritme Hufman, ditemukan oleh Michael Hankamer pada 1979. Seperti penjelasan pada algoritme Huffman, simbol yang jarang dipakai akan mempunyai kode biner yang lebih panjang daripada simbol yang sering dipakai, hal ini menyebabkan urutan kode biner akan semakin panjang jika banyak simbol yang jarang digunakan pada sumber teks. Oleh karena itu Michael Hankamer memodifikasi algoritme Huffman agar jika kondisi tersebut terjadi, urutan kode biner menjadi lebih kecil dari algoritme Huffman biasa [3].

Terdapat beberapa penelitian yang menggunakan algoritme Huffman untuk pengodean teks, contohnya pada penelitian dengan judul "Penerapan Algoritme Huffman Dan Shanon Fano Dalam Pemampatan Teks" yang hasilnya algoritme Huffman lebih optimal dibanding palgoritme Shanon Fano untuk, lalu perbedaan antara penelitian tersebut dan penelitian ini terletak pada algoritme yang digunakan dan metodenya.

Teori dan prinsip kerja dari kedua algoritme dipelajari pada mata kuliah Teori Informasi dan Pengodean, dimana mahasiswa melakukan perhitungan manual untuk mendapatkan hasil perbandingan dari kedua algoritme. Proses ini membutuhkan waktu yang lama jika sumber teks nya panjang dan harus dilakukan secara teliti agar tidak terjadi kesalahan. Oleh karena itu, penulis berkeinginan membuat program yang dapat melakukan simulasi dari kedua algoritme tersebut. Dengan demikian, mahasiswa yang mengambil mata kuliah Teori Informasi dan Pengodean dimudahkan dalam melakukan perhitungan pengodean teks dari kedua algoritme dengan sumber teks yang lebih Panjang. Selain itu dengan program simulasi yang dibuat, mahasiswa bisa mendapatkan hasil secara tepat dan cepat.

2. METODE PENELITIAN

Pada penelitian ini terdapat beberapa tahap dalam perancangan program dan menjalankan program diantara lain:

1. Mengimplementasikan algoritme Huffman dan algoritme Huffman Weaven Hankamer untuk pengodean sumber pada aplikasi MATLAB.
2. Menentukan sumber informasi teks yang dijadikan *input* untuk 16 kali percobaan, dengan 8 percobaan (15.000 sampai 22.000 karakter) menggunakan sumber informasi yang karakter ELSE lebih sedikit dan 8 percobaan (15.000 sampai 22.000 karakter) menggunakan sumber informasi yang karakter ELSE lebih banyak.
3. Membuat GUI untuk membuat tampilan aplikasi yang menarik dan mudah digunakan.
4. Melihat dan membandingkan nilai ACL, *Entropy,* Efisiensi dan panjang kode biner dari algoritme Huffman dan algoritme Huffman Weaven Hankamer.

Terdapat beberapa persamaan yang digunakan untuk menghitung hasil dari pengodean yaitu ACL, entropy, dan efisiensi.

ACL (average codeword length) adalah panjang rata-rata kode biner dari sumber informasi, ACL memiliki persamaan yaitu:

$$\text{Acl} = \sum_{i=0}^{q} \text{lenght } f(si) P(si) \qquad (1)$$

Dimana:

lenght f (si) = Jumlah bit kode biner simbol
P (si) = Peluang munculnya simbol
i = Karakter simbol pertama
q = Karakter simbol terakhir

Entropy suatu sumber informasi adalah ukuran jumlah informasi suatu sumber atau parameter yang menyatakan jumlah informasi rata-rata per simbol. Bila S = {S, P} adalah suatu sumber informasi dengan distribusi peluang P = {p1,p2,…,pq}, maka average information yang didapat dari sebuah sampel atau simbol dari S adalah:

$$H(S) = \sum Pi\ log2\ 1/Pi \qquad (2)$$

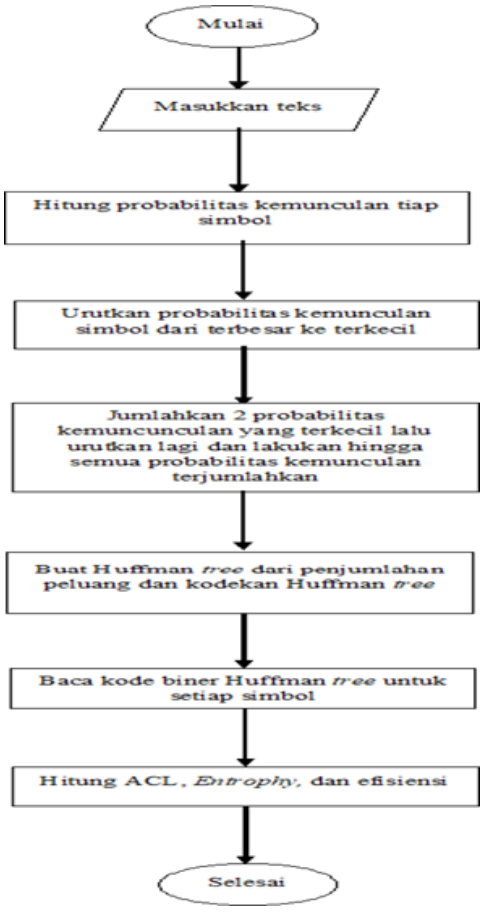
Dimana:

H(S) = Entrophy
Pi = Peluang

Efisiensi merupakan tolak ukur untuk melihat seberapa optimal skema pengodean, didapat dari perbandingan antara entropy dengan panjang kode rata-rata (acl) [6]. Dapat dirumuskan sebagai berikut:

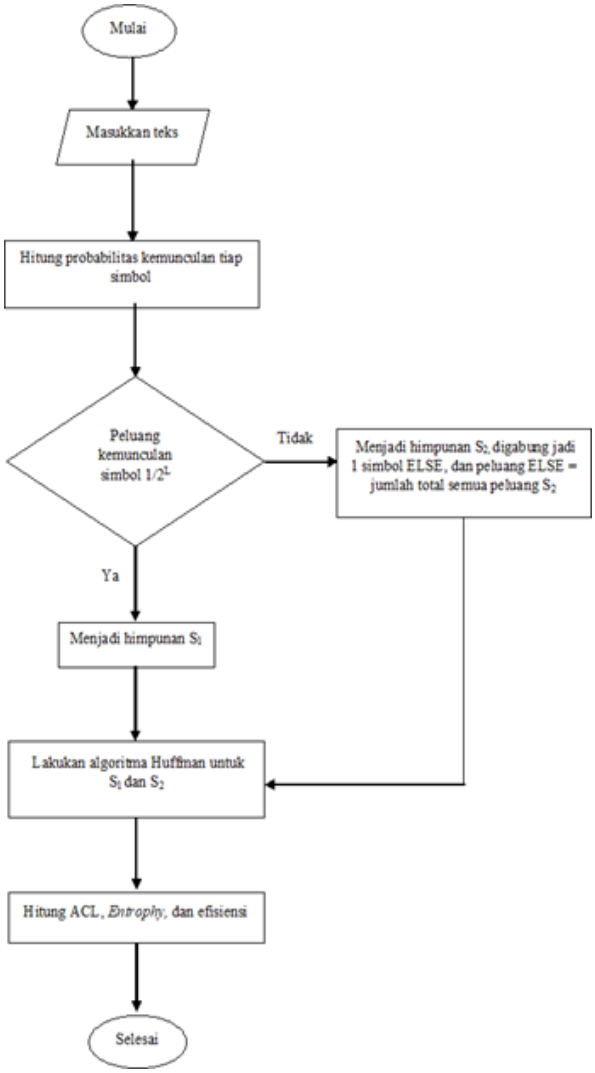
$$\text{Efisiensi} = \frac{H(s)}{ACL} x100\% \qquad (3)$$

Dibawah ini merupakan *flowchart* dari kedua algoritme.



Gambar 2.1 flowchart algoritme Huffman

Pada proses perhitungan ACL, Entrophy, dan Efisiensi menggunakan persamaan (1), (2), dan (3)



Gambar 2.2 flowchart algoritme Huffman Weaven Hankamer

- Bagi himpunan simbol menjadi 2 himpunan yakni S1 dan S2 (ELSE) dengan ketentuan seperti dibawah ini:

$$S1 = \{x;\ p\ (x) > 1/2\text{^}L\} \qquad (4)$$
$$S2 = \{x;\ p\ (x) \leq 1/2\text{^}L\} \qquad (5)$$

Dimana L adalah panjang bit ASCII (American Standard Code for Information Interchange), simbol-simbol yang tergabung pada himpunan S2 digabungkan menjadi 1 simbol dengan nama ELSE, peluang ELSE adalah sama dengan jumlah total semua peluang pada S2.

$$P(x) = \sum \in s2\ p(x) \qquad (6)$$

❖ Lakukan algoritme Huffman untuk mengodekan semua simbol pada S1 dan ELSE, jika sebuah simbol dalam ELSE akan dikirimkan maka kode nya sama dengan kode Huffman untuk ELSE di ikuti dengan karakter ASCII (128 simbol dengan panjang 7 bit) dari simbol yang akan dikirimkan.

3. HASIL DAN PEMBAHASAN

3.1 Algoritme Huffman dan Algoritme Huffman Weaven Hankamer Skema 1

Pada tahap skema 1 ini menggunakan sumber teks dengan 15.000 sampai 22.000 karakter dengan karakter ELSE lebih sedikit. Pada skema 1 ini proses algoritme akan dijelaskan dengan sumber teks 15.000 karakter yang akan dijelaskan pada tahap-tahap dibawah ini:

1. Hitung probabilitas kemunculan tiap simbol dan urutkan dari terbesar ke terkecil.

Pada tahap ini data *input* yang berjumlah 15.000 karakter akan diproses untuk menghitung probabilitas kemunculan tiap simbol, lalu mengurutkan probabilitas dari terbesar ke terkecil seperti pada tabel 3.1 dibawah ini.

Tabel 3.1 Probabilitas Kemunculan Karakter Skema 1

No.	Karakter	Jumlah Kemunculan	Probabilitas
1	A	2279	0.15136
2	Spasi	2168	0.14399
3	N	1410	0.093644
4	I	1050	0.069735
5	E	955	0.063426
6	T	733	0.048682
7	K	613	0.040712
8	U	600	0.039849
9	R	577	0.038321
10	S	572	0.037989
11	M	566	0.03759
12	D	514	0.034137
13	G	490	0.032543
14	P	447	0.029687
15	L	341	0.022647
16	B	297	0.019725
17	O	255	0.016936
18	H	255	0.016936
19	Y	173	0.01149
20	,	132	0.0087667
21	J	123	0.008169
22	.	106	0.0070399
23	F	59	0.0039184
24	C	55	0.0036528
25	2	39	0.0025902
26	1	36	0.0023909
27	V	36	0.0023909
28	3	33	0.0021917
29	9	32	0.0021253
30	W	32	0.0021253
31	0	32	0.0021253
32	(	24	0.0015939
33	)	23	0.0015275

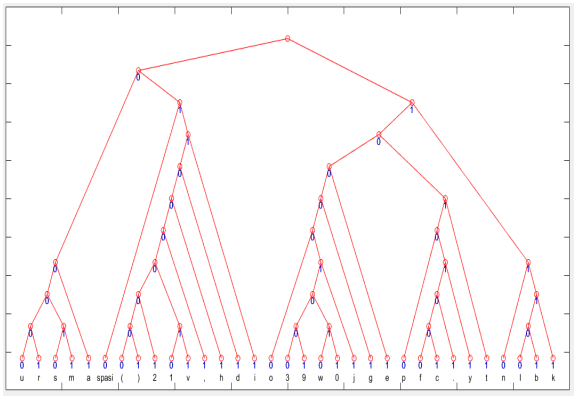
2. Proses penjumlahan semua probabilitas.

Pada tahap ini, dua probabilitas terkecil dijumlahkan lalu diurutkan lagi seperti yang dapat dilihat pada Tabel 3.1. Dua probabilitas kemunculan terkecil adalah karakter " (" dan ") " jadi probabilitas kedua karakter tersebut dijumlahkan, P "(" + P ")" = 0,0015939 + 0,0015275 = 0,0031214. Proses ini dilakukan berulang hingga semua probabilitas dijumlahkan dan jumlahnya sama dengan satu.

Lalu untuk algoritme Huffman Weaven Hankamer karakter yang memiliki probabilitas kemunculan digabung menjadi satu himpunan ELSE, dimana probabilitas karakter ELSE dijumlahkan menjadi 1.

3. Pembuatan Huffman tree

Setelah proses penjumlahan probabilitas selesai, maka proses selanjutnya adalah membuat Huffman *tree* dari kedua algoritme untuk menentukan kode biner dari masing-masing karakter.



Gambar 3.1 Huffman Tree Skema 1

a) Algoritme Huffman

Berdasarkan Gambar 3.1 dapat ditentukan kode biner untuk masing-masing karakter, dengan ketentuan sisi kiri untuk kode biner 0 dan sisi kanan untuk kode biner 1. Contoh untuk karakter " f " memiliki kode biner 10101000, dan untuk kode biner semua karakter dapat dilihat pada Tabel 3.2 dibawah ini.

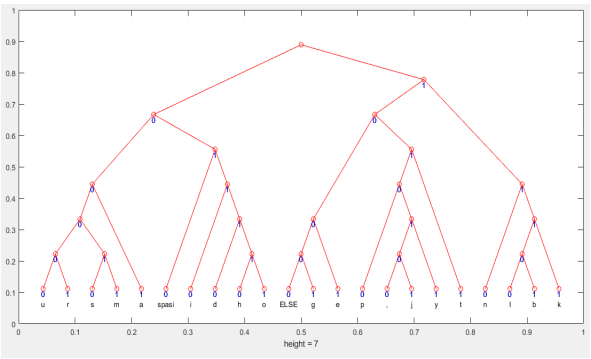
Tabel 3.2 Kode Biner Huffman Skema 1

	Kode Biner Huffman	
1.	a	001
2.	spasi	010
3.	n	110
4.	i	0111
5.	e	1001
6.	t	1011
7.	k	1111
8.	u	00000
9.	r	00001
10.	s	00010
11.	m	00011
12.	d	01101
13.	g	10001
14.	p	10100
15.	l	11100
16.	b	11101
17.	h	011001
18.	o	100000
19.	y	101011
20.	,	0110001
21.	j	1000011
22.	.	1010101
23.	f	10101000
24.	c	10101001
25.	2	011000001
26.	1	011000010
27.	v	011000011
28.	3	100001000
29.	9	100001001
30.	w	100001010
31.	0	100001011
32.	(	0110000000
33.	)	0110000001

Dapat dilihat pada Tabel 3.2 bahwa karakter dengan panjang kode biner terkecil adalah " a " yaitu tiga bit, dan karakter dengan kode biner terbesar adalah ") " yaitu 10 bit, serta jumlah total bit kode biner adalah 300 bit.

b) Algoritme Huffman Weaven Hankamer

Selanjutnya untuk Huffman *tree* dari algoritme Huffman Weaven Hankamer dapat dilihat pada gambar 3.2:



Gambar 3.2 Huffman Weaven Hankamer Tree Skema 1

Perbedaan dengan algoritme Huffman adalah pada kode biner karakter. Jika pada algoritme Huffman karakter "f " memiliki kode biner 10101000, pada Huffman Weaven Hankamer karakter " f " memiliki kode biner 100001100110. Perbedaan ini karena " f " termasuk karakter ELSE dimana probabilitasnya $<1/2^L$ Oleh karena itu kode biner karakter " f " adalah kode biner ELSE ditambah kode ASCII dari karakter " f ". Selanjutnya untuk kode biner semua karakter Huffman Weaven Hankamer *tree* dapat dilihat pada Tabel 3.3 berikut ini.

Tabel 3.3 Kode Biner Huffman Weaven Hankamer Skema 1

	Kode Biner Huffman Weaven	
1.	a	001
2.	spasi	010
3.	n	110
4.	i	0110
5.	e	1001
6.	t	1011
7.	k	1111
8.	u	00000
9.	r	00001
10.	s	00010
11.	m	00011
12.	d	01110
13.	ELSE	10000
14.	g	10001
15.	p	10100
16.	l	11100
17.	b	11101
18.	h	011110
19.	o	011111
20.	y	101011
21.	,	1010100
22.	j	1010101
23.	ELSE)	100000101001
24.	ELSE (	100000101000
25.	ELSE 0	100000110000
26.	ELSE w	100001110111
27.	ELSE 9	100000111001
28.	ELSE 3	100000110011
29.	ELSE v	100001110110
30.	ELSE 1	100000110001
31.	ELSE 2	100000110010
32.	ELSE c	100001100011
33.	ELSE f	100001100110
34.	ELSE .	100000101110

Berdasarkan Tabel 3.3 Kode Biner Huffman Weaven Hankamer, karakter dengan panjang kode biner terkecil adalah karakter " a " yaitu 3 bit, dan karakter dengan kode biner terbesar adalah karakter yang termasuk ELSE yaitu 12 bit. Dapat dilihat juga bahwa karakter yang termasuk ELSE ada 12 dan yang tidak termasuk ELSE ada 21 karakter, lalu untuk jumlah total

kode biner adalah 107-bit dimana nilai ini lebih kecil dibandingkan pada algoritme Huffman.

4. Hitung nilai ACL, entrophy, dan efisiensi

Pada tahap ini, nilai entrophy, ACL, dan efisiensi dihitung berdasarkan hasil yang telah didapatkan pada proses sebelumnya menggunakan persamaan 1, 2, dan 3. Hasilnya dapat dilihat pada Tabel 3.4 dibawah ini.

Tabel 3.4 Entrophy, ACL, Efisiensi Skema 1

No.	Jumlah Karakter	Entrophy		ACL		Efisiensi	
		Huffman	Huffman Weaven	Huffman	Huffman Weaven	Huffman	Huffman Weaven
1.	15.000	4,162	4,162	4,195	4,289	99,21%	97,03%
2.	16.000	4,102	4,102	4,152	4,235	98,80%	96,86%
3.	17.000	4,188	4,188	4,225	4,341	99,14%	96,49%
4.	18.000	4,139	4,139	4,176	4,269	99,12%	96,94%
5.	19.000	4,108	4,108	4,15	4,259	98,97%	96,45%
6.	20.000	4,166	4,166	4,209	4,303	98,99%	96,82%
7.	21.000	4,122	4,122	4,161	4,237	99,08%	97.29%
8.	22.000	4,202	4,202	4,237	4,326	99,18%	97,13%
Rata-rata		4,148	4,148	4,188	4,282	99,06%	96,82%

Berdasarkan hasil pada Tabel 3.4 dapat dilihat bahwa nilai *entrophy* untuk kedua algoritme hasilnya sama. Hal ini karena *entrophy* adalah ukuran jumlah informasi yang proses perhitungan nya tidak dipengaruhi oleh hasil kode biner masing-masing algoritme.

Selanjutnya untuk Nilai ACL, dapat dilihat bahwa algoritme Huffman memiliki nilai ACL yang lebih kecil dibanding algoritme Huffman Weaven Hankamer. Hal ini karena karakter ELSE pada algoritme Huffman Weaven Hankamer memiliki kode biner yang lebih panjang daripada Algoritme Huffman. Contohnya untuk kara nilai efisiensi algoritme Huffman lebih besar dibandingkan dengan algoritme Huffman Weaven Hankamer dengan selisih rata-rata 2,24 %.

3.2 Algoritme Huffman dan Algoritme Huffman Weaven Hankamer Skema 2

Pada tahap skema 2 ini menggunakan sumber teks dengan 15.000 sampai 22.000 karakter yang mana karakter ELSE nya lebih banyak. Skema 2 ini akan dijelaskan proses algoritme dengan sumber teks berjumlah 15.000 karakter. Penjelasannya dapat dilihat pada setiap tahapan berikut:

1. Perhitungan probabilitas kemunculan tiap simbol dan Pengurutan simbol dari Probabilitas terbesar.

Pada tahap ini data input yang berjumlah 15.000 karakter diproses untuk menghitung probabilitas kemunculan tiap simbol, dan kemudian simbol diurutkan berdasarkan probabilitas terbesar ke terkecil seperti pada tabel 3.5 dibawah ini.

Tabel 3.5 Probabilitas Kemunculan Karakter Skema 2

No.	Karakter	Jumlah Kemunculan	Probabilitas
1.	A	2453	0.1612
2.	Spasi	2333	0.15332
3.	N	1300	0.085431
4.	E	986	0.064796
5.	I	954	0.062693
6.	T	703	0.046198
7.	K	622	0.040875
8.	R	602	0.039561
9.	S	592	0.038904
10.	M	533	0.035027
11.	D	518	0.034041
12.	U	509	0.033449
13.	G	441	0.028981
14.	P	383	0.025169
15.	L	342	0.022475
16.	b	302	0.019846
17.	o	264	0.017349
18.	h	213	0.013998
19.	y	184	0.012092
20.	1	141	0.009266
21.	,	135	0.0088717
22.	.	134	0.0088059
23.	P	132	0.0086745
24.	j	102	0.006703
25.	c	81	0.005323
26.	w	74	0.004863
27.	v	47	0.0030887
28.	f	46	0.0030229
29.	6	16	0.0010515
30.	3	11	0.00072288
31.	4	11	0.00072288
32.	5	10	0.00065716
33.	8	9	0.00059144
34.	7	8	0.00052573
35.	/	7	0.00046001
36.	z	4	0.00026286
37.	9	3	0.00019715
38.	Q	1	6.5716e-05
39.	R	1	6.5716e-05
40.	T	1	6.5716e-05
41.	A	1	6.5716e-05
42.	!	1	6.5716e-05
43.	@	1	6.5716e-05
44.	#	1	6.5716e-05
45.	$	1	6.5716e-05
46.	%	1	6.5716e-05
47.	^	1	6.5716e-05
48.	&	1	6.5716e-05
49.	*	1	6.5716e-05

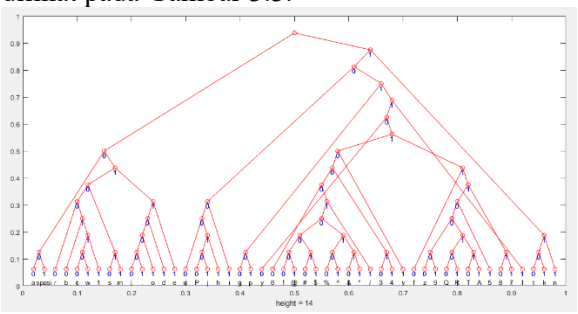
Pada Tabel 4.5 Probabilitas Kemunculan Karakter dapat dilihat bahwa karakter " a " memiliki probabilitas kemunculan terbesar yaitu 0,1612 dan karakter " * " memiliki probabilitas terkecil yaitu 0,000065716.

2. Proses penjumlahan semua probabilitas

Seperti pada percobaan sebelumnya, tahap ini juga menjumlahkan dua probabilitas terkecil dan kemudian diurutkan lagi. Proses ini dilakukan hingga semua probabilitas dapat dijumlahkan dan totalnya sama dengan 1. Demikian juga untuk algoritme Huffman Weaven Hankamer, karakter yang memiliki probabilitas kemunculan $1/2^L$ digabung menjadi satu himpunan ELSE dimana probabilitas nya dijumlahkan jadi satu.

3. Pembuatan Huffman *tree*

Setelah proses penjumlahan probabilitas selesai maka proses selanjutnya adalah membuat huffman *tree* untuk menentukan kode biner dari masing-masing karakter, huffman *tree* dapat dilihat pada Gambar 3.3.



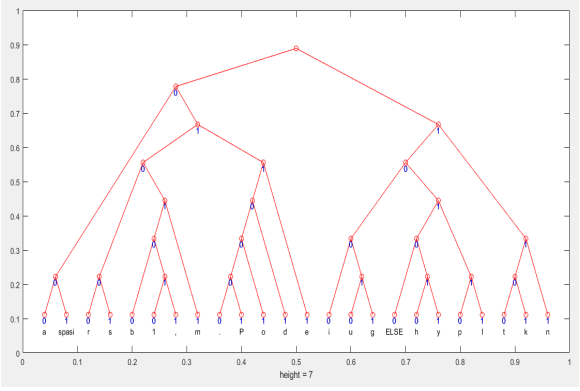
Gambar 3.3 Huffman Tree Skema 2

Jika Gambar 3.3 Huffman *tree* skema 2 diatas dibandingkan dengan Gambar Huffman *tree* skema 1, terlihat bahwa Huffman *tree* skema 2 lebih rumit dibanding Gambar Huffman *tree* skema 1. Selanjutnya untuk kode biner karakter dapat dilihat pada Tabel 3.6.

Tabel 3.6 Kode Biner Huffman Skema 2

Kode Biner Huffman		
1.	A	000
2.	Spasi	001
3.	N	111
4.	E	0111
5.	I	1001
6.	t	1100
7.	k	1101
8.	r	01000
9.	s	01010
10.	m	01011
11.	d	01101
12.	u	10000
13.	g	10100
Kode Biner Huffman		
14.	p	10101
15.	l	10111
16.	b	010010
17.	o	011001
18.	h	100011
19.	y	101100
20.	1	0100111
21.	,	0110000
22.	.	0110001
23.	P	1000100
24.	j	1000101
25.	c	01001100
26.	w	01001101
27.	v	10110101
28.	f	10110110
29.	6	1011010000
30.	3	1011010010
31.	4	1011010011
32.	5	1011011101
33.	8	1011011110
34.	7	1011011111
35.	/	10110100011
36.	z	101101110000
37.	9	101101110001
38.	Q	1011011100100
39.	R	1011011100101
40.	T	1011011100110
41.	A	1011011100111
42.	!	10110100010000
43.	@	10110100010001
44.	#	10110100010010
45.	$	10110100010011
46.	%	10110100010100
47.	^	10110100010101
48.	&	10110100010110
49.	*	10110100010111

Berdasarkan Tabel 3.6 Kode Biner Huffman dapat dilihat bahwa karakter " a " memiliki panjang kode biner terpendek yaitu 3 bit, dan karakter dengan panjang kode biner terbesar adalah karakter yang memiliki probabilitas kemunculan 0,000065716 yaitu 14 bit.



Gambar 3.4 Huffman Weaven Hankamer Tree Skema 2

Jika membandingkan Gambar 3.4 Huffman Weaven Hankamer *tree* dengan Gambar 3.3 Huffman *tree,* maka Huffman Weaven Hankamer *tree* diatas terlihat lebih ringkas karena pada percobaan tahap ini karakter ELSE lebih banyak daripada karakter biasa.

Tabel 3.7 Kode Biner Huffman Weaven Hankamer Skema 2

Kode Biner Huffman Weaven		
1.	a	000
2.	spasi	001
3.	n	111
4.	e	0111
5.	i	1000
6.	t	1100
7.	k	1101
8.	r	01000
9.	s	01001
10.	m	01011
11.	d	01101
12.	u	10010
13.	g	10011
14.	ELSE	10100
15.	p	10110
16.	l	10111
17.	b	010100
18.	o	011001
19.	h	101010
20.	y	101011
21.	l	0101010
22.	,	0101011
23.	.	0110000
24.	P	0110001
25.	ELSE *	101000101010
26.	ELSE &	101000100110
27.	ELSE ^	101001011110
28.	ELSE %	101000100101
29.	ELSE $	101000100100
30.	ELSE #	101000100011
31.	ELSE @	101001000000
32.	ELSE !	101000100001
33.	ELSE A	101001000001
34.	ELSE T	101001010100
35.	ELSE R	101001010010
36.	ELSE Q	101001010001
37.	ELSE 9	101000111001
38.	ELSE z	101001111010
39.	ELSE /	101000101111
40.	ELSE 7	101000110111
41.	ELSE 8	101000111000
42.	ELSE 5	101000110101
43.	ELSE 4	101000110100
44.	ELSE 3	101000110011
45.	ELSE 6	101000110110
46.	ELSE f	101001100110
47.	ELSE v	101001110110
48.	ELSE w	101001110111
49.	ELSE c	101001100011
50.	ELSE j	101001101010

Berdasarkan Tabel 3.7 Kode Biner Huffman Weaven Hankamer dapat dilihat bahwa karakter " a " memiliki panjang kode biner terpendek yaitu 3 bit, dan karakter dengan panjang kode biner terbesar adalah karakter ELSE yaitu 12 bit.

4. Hitung nilai ACL, entrophy, dan efisiensi

Pada proses ini, nilai entrophy, ACL, dan efisiensi berdasarkan hasil yang telah didapatkan pada proses sebelumnya akan dihitung menggunakan persamaan 1, 2, dan 3. Hasilnya dapat dilihat pada Tabel 3.8 dibawah ini.

Tabel 3.8 Entrophy, ACL, Efisiensi Skema 2

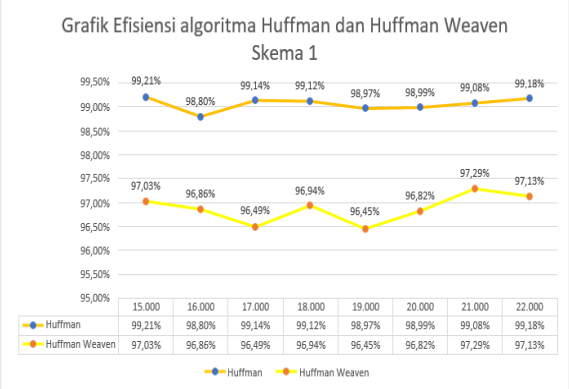
No.	Jumlah Karakter	Entrophy		ACL		Efisiensi	
		Huffman	Huffman Weaven	Huffman	Huffman Weaven	Huffman	Huffman Weaven
1.	15.000	4,176	4,176	4,217	4,294	99,03%	97,25%
2.	16.000	4,25	4,25	4,287	4,389	99,12%	96,82%
3.	17.000	4,097	4,097	4,147	4,201	98,79%	97,52%
4.	18.000	4,134	4,134	4,178	4,25	98,94%	97,28%
5.	19.000	4,178	4,178	4,221	4,261	98,98%	98,06%
6.	20.000	4,035	4,035	4,135	4,178	98,79%	97,76%
7.	21.000	4,124	4,124	4,171	4,228	98,86%	97,52%
8.	22.000	4,176	4,176	4,216	4,284	99,03%	97,48%
Rata-rata		4,146	4,146	4,196	4,260	98,94%	97,46%

Berdasarkan hasil pada Tabel 3.8, Nilai entrophy untuk kedua algoritme, sama seperti percobaan sebelumnya adalah sama. Sedangkan nilai ACL algoritme Huffman lebih kecil dibanding algoritme Huffman Weaven Hankamer dengan selisih rata-rata 0,064. Selain itu untuk nilai efisiensi, dapat dilihat berdasarkan Tabel 3.8 bahwa nilai efisiensi algoritme Huffman lebih besar dibandingkan dengan algoritme Huffman Weaven Hankamer dengan selisih rata-rata 1,48 %.

3.3 Perbandingan hasil Algoritme Huffman dan Algoritme Huffman Weaven Hankamer dan GUI

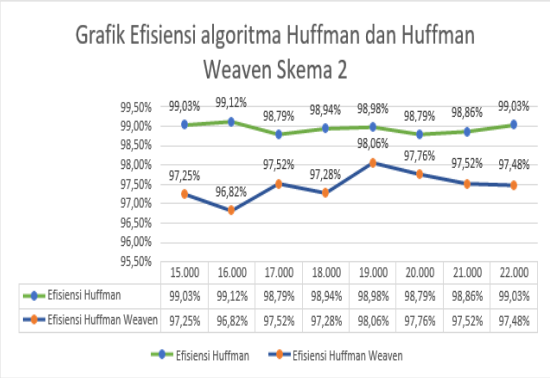
1. Perbandingan hasil Algoritme Huffman dan Algoritme Huffman Weaven Hankamer

Hasil dari dari skema 1 dan skema 2 kemudian dibandingkan untuk melihat algoritme yang lebih optimal berdasarkan nilai efisiensi seperti yang dapat dilihat pada grafik gambar 3.5.



Gambar 3.5 Grafik Efisiensi Skema 1

Dari Gambar 3.5 dapat dilihat bahwa dengan 8 kali percobaan dengan sumber teks yang karakter ELSE nya lebih sedikit, algoritme Huffman memiliki *range* efisiensi dari 98,80% sampai 99,21%. Sedangkan efisiensi algoritme Huffman Weaven Hankamer berkisar dalam *range* 96,45% sampai 97,29%. Sehingga selisih rata-rata nya adalah 2,24%.
Selanjutnya untuk skema 2 dengan 8 kali percobaan dengan sumber teks yang karakter ELSE nya lebih banyak grafik efisiensi nya dapat dilihat pada gambar dibawah ini

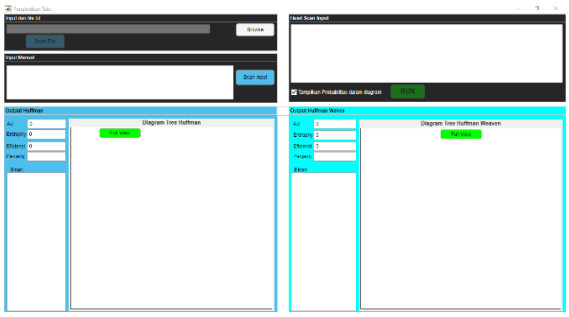


Gambar 3.6 Grafik Efisiensi Skema 2

Dari Gambar 3.6 Grafik Efisiensi dapat dilihat bahwa dengan 8 kali percobaan dengan sumber teks yang karakter ELSE nya lebih banyak, algoritme Huffman memiliki range efisiensi dari 98,79% sampai 99,12%. Sedangkan algoritme Huffman Weaven Hankamer memiliki *range* efisiensi dari 96,82% sampai 98,06%, dan selisih rata-rata nya adalah 1,48%.

2. GUI

Pada penelitian ini GUI dibuat agar tampilan dan penggunaan program menjadi mudah digunakan, tampilan dari GUI dapat dilihat pada gambar 3.7



Gambar 3.7 tampilan GUI

Selanjutnya berikut petunjuk penggunaan program:

1. Program memiliki 2 pilihan input yaitu dari file txt dan input manual, untuk input dari file txt langkah pertama yaitu tekan tombol browse untuk membuka file, lalu tekan scan file untuk menghitung jumlah probabilitas tiap karakter. Untuk input manual terdapat box untuk memasukan sumber informasi, lalu tekan scan input untuk menghitung jumlah probabilitas tiap karakter.

2. Selanjutnya hasil perhitungan probabilitas tiap karakter akan muncul pada box hasil scan input, pada box ini akan ditampilkan jumlah total karakter sumber informasi dan probabilitas kemunculan tiap karakter, tekan run untuk program melakukan pengodean algoritme Huffman dan algoritme Huffman Weaven Hankamer, selain itu juga terdapat check box untuk memilih menampilkan probabilitas karakter pada huffman tree.

3. Setelah itu program akan menampilkan 2 hasil dari masing-masing program, untuk nilai ACL, entrophy, Efisiensi dan panjang total kode biner terletak pada bagian kiri atas, untuk hasil kode biner terdapat pada box binari, dan untuk hasil Huffman tree terdapat pada box diagram huffman tree, pada box ini terdapat tombol full view untuk melihat Huffman tree secara lebih besar.

4. KESIMPULAN

Berdasarkan hasil yang telah diperoleh dapat disimpulkan bahwa:

1. Penelitian ini telah berhasil membangun program perhitungan menggunakan GUI untuk algoritme Huffman dan algoritme Huffwan Weaven pada *software* MATLAB R2019a.

2. Pada skema 1 algoritme Huffman lebih optimal dibanding algoritme Huffman Weaven Hankamer dengan selisih rata-rata 2,24 %. Pada skema 2 hasil algoritme Huffman tetap lebih optimal, namun hasil algoritme Huffman Weaven Hankamer mendekati hasil algoritme Huffman dibanding skema 1 dengan selisih rata-rata yang lebih kecil yaitu 1,48 %.
3. Algoritme Huffman Weaven Hankamer lebih ringkas dibandingkan dengan Algoritme Huffman untuk hasil Huffman tree. Selanjutnya, ditemukan juga bahwa semakin banyak karakter ELSE maka algoritme Huffman Weaven Hankamer lebih mendekati optimal. Selain itu, semakin besar probabilitas kemunculan karakter, maka panjang kode biner akan semakin kecil, begitu pun sebaliknya.
4. Program yang telah dibuat hanya cocok untuk *file* dengan format txt.

DAFTAR PUSTAKA

[1] Akhmad Pahdi, "Algoritme Huffman Dalam Pemampatan Dan Enksripsi Data", STMIK Banjarbaru, 2017.

[2] Purwanto, Hari, "Penerapan Algoritme Huffman Pada Kompresi File Wave". Jurnal Universitas Surya Dharma vol. 2, 2015.

[3] Micahel Hankamer, "A Modified Huffman Procedure with Reduced Memory Requirement", *Proc. IEEE* vol. 27, 1979.

[4] Claude E. Shannon, "A Mathematical Theory of Communication", *Bell Sys Tech Jour.,* vol. 27, pp. 398-403, Juli 1948.

[5] Stephane Le Goff, "Information Theory and Coding Module", Newcastle University, 2012.

[6] Johanis Bowakh, Beby Manafe, "Bahan Ajar Teori Informasi dan Pengodean", Fakultas Sains dan Teknik Undana, 2007.

[7] David A. Huffman, "A Method for the Construction of Minimum Redundancy Codes", *Proc. IRE* vol 40, pp. 1098-1101, September 1952.

[8] Gunaidi A. Away, "The Shortcut of MATLAB Programming", INFORMATIKA, 2014.

[9] Thomas M. Cover, Joy A. Thomas, "Element of Information Theory", Wiley-Interscience, 2006.

[10] Danny Dimas Sulistio, "Perbandingan Algoritme Huffman Statis Dengan Algoritme Huffman Adaptif Pada Kompresi Data Teks", FMIPA IPB, 2004.

[11] Lenti, Febri Nova, "Visualisasi Pengodean Huffman Dengan Pohon Biner" *in Seminar Nasional Riset Teknologi Informasi,* Yogyakarta, 2009. pp. 211-218.

[12] Erna Zuni Astuti, Erwin Yudi Hidayat, "Kode Huffman Untuk Kompresi Pesan", Techno.COM vol. 12, pp. 117-126, Mei 2013.

[13] R. G. Gallager, "Information Theory and Reliable Communication", New York: John Wiley, 1968.

[14] G. O. H. Katona, T. O. H. Nemetz, "Huffman Code and Self-information", *IEEE Trans. Inf. Theory,* vol. 22, no. 3, pp. 337-340, Mei 1976

[15] S. Mohajer, P. Pakzad, A. Kakhbod, "Tight Bounds on The Redundancy of huffman Codes", *IEEE Trans. Inf. Theory,* vol. 58, no. 11, pp. 6737-6746, November 2012