BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 2, No 2, April 2022
DOI 10.47065/bulletincsr.v2i2.156 | Hal 34−40
https://hostjournals.com/bulletincsr

Implementasi Algoritma Set Partitioning in Hierarchical Trees (SPIHT) dan Algoritma Huffman Untuk Kompresi File Citra

Lubis Irsan

Prodi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: irsanlubis1996@gmail.com

Abstrak−Seiring berkembangnya teknologi, proses pengiriman file tidak perlu dilakukan lagi secara manual. Data atau file tersebut dapat dikirim secara elekronik melalui email. Ukuran file citra terkadang sangat besar dimana semakin baik kualitas citra yang dihasilkan, maka ukuran pixel yang dibutuhkan untuk merekam citra tersebut semakin besar. Dengan ukuran file citra yang sangat besar, pada saat melakukan proses pemindahan bisa saja proses pemindahan gagal, karena media ruang penyimpanan melebihi batasnya. Adapun solusi dalam permasalahan ini adalah bagaimana file citra tersebut dapat dikompresi guna untuk mempercepat pemindahan dan penyimpanan file citra. Kompresi pada file citra dilakukan dengan memperkecil ukuran file citra dengan proses mengurangi bit pada file citra, akan tetapi tidak menghilangkan data informasinya didalamnya.Dengan melakukan kompresi, data yang besar akan berkurang ukurannya sehingga dapat menghemat ruang penyimpanan. Dalam penelitian ini, algoritma yang digunakan adalah swiss army knife data compression, dengan menggunakan metode tersebut, hasil kompresi dari nilai k mempunyai hasil kompresi yang berbeda-beda dari setiap nilainya, dsan hasil kompresi akan menguntungkan dalam melakukan pengiriman dan pemindahan file citra akan semakin mudah.

Kata Kunci: Implementasi; Kompresi File Citra; Set Partitioning in Hierarchical Trees; Huffman

Abstract−As technology develops, the process of sending files no longer needs to be done manually. Data or files can be sent electronically via email. The image file size is sometimes very large, where the better the quality of the resulting image, the bigger the pixel size needed to record the image. With a very large image file size, the transfer process may fail, because the media storage space exceeds its limit. The solution to this problem is how the image file can be compressed in order to speed up the transfer and storage of image files. Image compression is done by reducing the size of the image file by reducing the bits in the image file, but not eliminating the information data in it. By compressing, large data will be reduced in size so as to save storage space. In this study, the algorithm used is the Swiss army knife data compression, using this method, the compression results from the value of k have different compression results for each value, and the compression results will be beneficial in sending and moving image files more easily.

Keywords: Implementation; Image File Compression; Set Partitioning In Hierarchical Trees; Huffman

1. PENDAHULUAN

Kecepatan sangat dibutuhkan dalam pertukaran informasi. Pemindahan data atau sebuah informasi sering dilakukan melalui media penyimpanan ataupun melalui suatu media seperti internet. Kecepatan pengiriman data maupun informasi pada suatu bentuk yang bertipe file citra/gambar akan menjadi bagian utama dalam pertukaran suatu informasi atau data dimasa yang akan datang. Besarnya suatu ukuran data yang akan dipindahkan, sangat dibutuhkan ruang penyimpanan yang cukup untuk melakukan proses pemindahan. Proses pemindahan bisa saja terjadi gagal dilakukan jika ukuran file yang akan dipindahkan lebih besar dari pada ruang yang tersedia, dan salah satu cara untuk menangani hal tersebut adalah dengan teknik kompresi. Tujuan kompresi adalah untuk mengecilkan bentuk file yang besar, sehingga saat melakukan proses pemindahan dan penyimpanan suatu file akan menjadi lebih mudah atau efisien.

Citra/gambar merupakan hal yang vital dan menjadi bagian integral dalam kehidupan sehari-hari. Pada kepentingan tertentu, citra/gambar digunakan sebagai alat untuk mengungkapkan pertimbangan (reason), penggambaran (represent), ingatan (memories), pendidikan, komunikasi, hiburan dan lain sebagainya. Ukuran file citra/gambar relatif besar dimana semakin bagus kualitas gambar yang dihasilkan, maka ukuran pixel yang dibutuhkan untuk merekam gambar tersebut semakin besar. Proses kompresi adalah salah satu alternatif yang sangat dibutuhkan untuk memperkecil kapasitas suatu file citra/gambar, agar tidak terlalu boros dalam menggunakan ruang penyimpanan. Dengan ukuran suatu data yang telah dilakukan proses pengecilan atau pengurangan bit terhadap suatu file gambar, pada saat melakukan proses pengiriman atau pemindahan file gambar yang telah dilakukan pengurangan bit, maka proses pemindahan terhadap file akan semakin cepat.

Metode kompresi yang baik untuk kompresi citra adalah algoritma Set Partitioning In Hierarchical Trees (SPIHT). SPIHT dapat mengkompresi menjadi beberapa ruang partisi yang sesuai dengan jumlah partisi yang diinginkan. Teknis yang digunakan untuk SPIHT adalah menerapkan ambang batas untuk penetapan nilai matrik menjadi beberapa bagian dan jumlah partisi dapat dikatakan sebagai hasil kompresi dari SPIHT. Namun kompresi SPIHT hanya bekerja berdasarkan komposisi warna R, G, dan B, cara tidak siknifikan untuk mengubah ukuran citra. Permasalahan dari cara kompresi SPIHT tersebut dianggap tidak tepat dalam kompresi sehingga dibutuhkan algoritma lain untuk meningkatkan hasil kompresi citra dengan menggunakan algoritma Huffman. Tujuan dari penerapan algoritma Huffman ini tidak lain adalah menciptakan hasil kompresi warna dari algoritma SPIHT lebih kecil berdasarkan frekuensi. Karena ambang batas dari SPIHT membuat nilai matrik lebih seragam.

Dari perolehan penelitian sebelumnya yang telah dilakukan oleh Kharisma Mahesa dan karpen pada tahun 2017 telah berhasil membuktikan bahwa mengkompresi citra digital menggunakan metode Huffman dan dapat mengembalikan citra yang terkompresi kedalam bentuk semula dalam proses implementasi adalah citra digital dengan format Bitmap (*bmp) dengan kedalaman warna 24 bit [1].

ISSN 2774-3659 (Media Online)
Vol 2, No 2, April 2022
DOI 10.47065/bulletincsr.v2i2.156 | Hal 34−40
https://hostjournals.com/bulletincsr

Penelitian lainnya juga dilakukan untuk mengkompresi file citra, seperti yang dilakukan oleh Leni Marlina dkk telah berhasil membuktikan bahwa kompresi data yang telah dilakukan dengan menggunakan elias delta code sangat menghemat ruang penyimpanan dan dapat memanfaatkan semua tipe data[2].

Konsep kerja kompresi berdasarkan algoritma elias delta code dilakukan dengan mengubah nilai pixel citra, kemudian dengan memanfaatkan urutan kemunculan nilai-nilai pixel pada citra yang akan dikompres. Berdasarkan hal tersebut penulis menggunakan algoritma Set Partitioning In Hierarchical Trees untuk mengetahui kinerja kompresi apabila dilakukan terhadap kompresi file citra/gambar, sehingga file gambar yang berukuran besar akan dikompresi menjadi ukuran yang lebih kecil, dimana file gambar akanmembutuhkan jangka pemindahan yang makin singkat dibandingkan pada file gambar yang bukan terkompresi serta memperkecil ruang alokasi penyimpanan data.

2. METODOLOGI PENELITIAN

2.1 Kompresi Data

File kompresi dimaksudkan sebagai *file-file* yang digabungkan menjadi satu dengan tujuan untuk memperoleh ukuran *file* yang lebih kecil dibandingkan dengan *file* aslinya. *File* yang dikompresi memungkinkan *file* lebih cepat ketika di download dan lebih banyak data yang tersimpan dalam media penyimpanan eksternal [3],[4].

2.2 Algoritma Set Partitioning In Hierarchical Trees (SPIHT)

Algoritma SPIHT adalah algoritma yang dapat mengkompresi menjadi beberapa ruang partisi yang sesuai jumlah partisi yang diinginkan. Teknis yang digunakan untuk SPIHT adalah menerapkan ambang batas untuk penetepan nilai matrik menjadi beberapa bagian dan jumlah partisi dapat dikatakan sebagai hasil kompresi dari SPIHT. Pengkodean partisi set menggunakan rekursi partisi atau pemisahan set sampel yang dipandu oleh uji ambang batas. Proses SPIHT dari beberapa sumber di sampaikan dengan cara yang berbeda, akan tetapi maksud dan tujuannya sama, yaitu menerapkan urutan partisi dan menempatkan pixel untuk output dengan operasi ambang batas(thereshold) pada setiap pixel kunjungannya. Ambang batas juga menggunakan parameter list insignificant pixel(LIP), list insignificant set(LIS), dan list set pixel(LSP)[5].

2.3 Algoritma Huffman

Algoritma hufffman menggunakan prinsip pengkodean yang mirip dengan kode morse yaitu tiap karakter dikodekan hanya dengan rangkain beberapa bit, dimana karakter yang sering muncul dikodekan dengan rangkaian bit yang lebih panjang, karena prosesnya yang menggunakan kode ini, membuat alagoritma Huffman sebagai algoritma keluarga dengan variabel *codeword length*. Cara kerja algoritma Huffman ini adalah sebagai berikut[6]:

a. Menghitung banyaknya jenis karakter dan jumlah dari masing-masing karakter yang terdapat dalam sebuah file.
b. Menyusun setiap jenis karakter dengan urutan jenis karakter yang jumlahnya paling sedikit ke yang jumlahnya paling banyak.
c. Membuat pohon biner berdasarkan ukuran karakter dari yang jumlahnya terkecil ke yang terbesar, dan memberi kode untuk tiap karakter.
d. Mengganti data yang ada dengan kode bit berdasarkan pohon biner.
e. Menyimpan jumlah kode bit untuk kode bit yang terbesar, jenis karakter yang diurutkan dari frekuensi keluarnya terbesar ke terkecil beserta data yang sudah berubah menajadi kode bit sebagai data hasil kompresi.

2.3 Citra Digital

Citra digital adalah citra yang dapat diolah oleh komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan sebuah larik (array) yang berisi nilai-nilai real maupun komplek yang direpresentasikan dengan deretan bit tertentu[7]-[10].

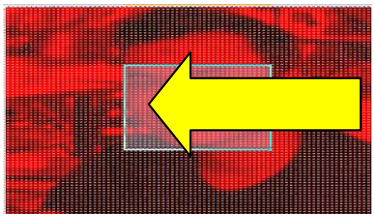
3. HASIL DAN PEMBAHASAN

Untuk melakukan analisa dengan tepat disediakan satu *file* sampel citra, sampel *file* citra yang disediakan yaitu irsanuji.jpeg yang berukuran 3KB, resolusi 80x80. Berikut merupakan sampel *file* citra yang akan dikompresi.

Gambar 1. Sampel citra input irsanuji.jpeg 80x80

Karena sampel citra yang akan di uji terlalu luas, maka matrik yang akan di uji diperkecil lagi menjadi ukuran 32x32. Berikut merupakan gambar matrik yang akan diuji.

BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 2, No 2, April 2022
DOI 10.47065/bulletincsr.v2i2.156 | Hal 34−40
https://hostjournals.com/bulletincsr



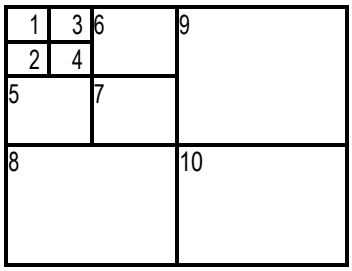
Gambar 2. Pengambilan 32x32 pada keabuan merah.

Berdasarkan gambar 2, sampel citra yang akan di uji adalah irsanuji.jpg keabuan merah,dengan ukuran matrik yang sudah di perkecil menjadi 32x32. Proses *Set Partitioning in Hierarchical Trees* (SPIHT) dari beberapa sumber disampaikan dengan cara yang berbeda, akan tetapi tujuan dan maksud penyampain algoritma SPIHT adalah sama, yaitu menerapkan urutan partisi, dan menempatkan *pixel* untuk output dengan operasi ambang batas (*threshold*) pada setiap *pixel* kunjungannya.

Tabel 1. Partisi 10 matrik dengan parameter LIP, LIS, dan LSP

Partisi	LIP	LIS	LSP
0	0	240	256
1	24	216	232
2	48	192	208
3	72	168	184
4	96	144	160
5	120	120	136
6	144	96	112
7	168	72	88
8	192	48	64
9	216	24	40
10	240	0	16

Matrik citra keabuan merah akan dikompresi berdasarkan kunjungan setiap pixelnya dengan tujuan penyesuaian kerangka partisi seperti pada gambar 3



Gambar 3. Struktur partisi matrik SPIHT

Untuk mendapatkan posisi matrik 1 pada gambar 3 dilakukan dengan cara melihat nilai pembanding ambang batas (*threshold*) setiap partisi. Nilai banding part1 dengan LIP = 24 , LIS = 216, dan LSP = 232. Nilai tersebut dibandingkan dengan nilai citra matrik irsanuji.jpg keabuan merah = x yang diawali pada posisi 0,0 yaitu 167 (nilai 0,0 = 167 dapat dilihat pada gambar 3)

Part1(0,0) :

Jika nilai x_0 > LIP**{**

True = 167 > 24, maka x0 – LIS

= 167 – 216

= – 49, nilai warna terkecil adalah 0, maka – 49 dianggap 0

= 0

Part1(0,1) :

Jika nilai x_1 > *LIP***{**

True = 178 > 24, maka x1 – *LIS*

= 178 – 216

= – 38, nilai warna terkecil adalah 0, maka – 38 dianggap 0

= 0

Part1(0,2) :

Jika nilai x_2 > *LIP***{**

True = 184 > 24, maka x2 – *LIS*

ISSN 2774-3659 (Media Online)
Vol 2, No 2, April 2022
DOI 10.47065/bulletincsr.v2i2.156 | Hal 34−40
https://hostjournals.com/bulletincsr

= 184 – 216
= – 32, nilai warna terkecil adalah 0, maka – 32 dianggap 0
= 0
Part1(0,3) :
Jika nilai x_3 > *LIP***{**
True = 191 > 24, maka x3 – *LIS*
= 191 – 216
= – 25, nilai warna terkecil adalah 0, maka – 25 dianggap 0
= 0
Part1(0,4) :
Jika nilai x_4 > *LIP***{**
True = 206 > 24, maka x4 – *LIS*
= 206 – 216
= – 10, nilai warna terkecil adalah 0, maka – 10 dianggap 0
= 0
Part1(0,5) :
Jika nilai x_5 > *LIP***{**
True = 212 > 24, maka x5 – *LIS*
= 212 – 216
= – 4, nilai warna terkecil adalah 0, maka – 4 dianggap 0
= 0
Part1(0,6) :
Jika nilai x_6 > *LIP***{**
True = 226 > 24, maka x6 – *LIS*
= 226 – 216
= 10
Part1(0,7) :
Jika nilai x_7 > *LIP***{**
True = 227 > 24, maka x7 – *LIS*
= 227 – 216
= 11

Berikutnya nilai banding part2 dengan LIP = 48, LIS = 192, dan LSP = 208. Nilai tersebut dibandingkan dengan nilai citra matrik irsanuji.jpg keabuan merah = x yang diawali pada posisi 0,0 yaitu 167 (nilai 0,0 = 167 dapat dilihat pada gambar 3.6)

Part2(0,0) :
Jika nilai x_0 > LIP**{**
True = 167 > 48, maka x0 – LIS
= 167 – 192
= –25, nilai warna terkecil adalah 0, maka – 25 dianggap 0
= 0
Part2(0,1) :
Jika nilai x_1 > *LIP***{**

True = 178 > 48, maka x1 – *LIS*
= 178 – 192
= –14, nilai warna terkecil adalah 0, maka –14 dianggap 0
= 0
Part2(0,2) :
Jika nilai x_2 > *LIP***{**
True = 184 > 48, maka x2 – *LIS*
= 184 – 192
= –8, nilai warna terkecil adalah 0, maka -8 dianggap 0
= 0
Part2(0,3) :
Jika nilai x_3 > *LIP***{**
True = 191 > 48, maka x3 – *LIS*
= 191 – 192
= -1, nilai warna terkecil adalah 0, maka -1 dianggap 0
= 0

ISSN 2774-3659 (Media Online)
Vol 2, No 2, April 2022
DOI 10.47065/bulletincsr.v2i2.156 | Hal 34−40
https://hostjournals.com/bulletincsr

Part2(0,4) :
Jika nilai x_4 > *LIP***{**
True = 206 > 48, maka x4 – *LIS*
= 206 – 192
= 14
Part2(0,5) :
Jika nilai x_5 > *LIP***{**
True = 212 > 48, maka x5 – *LIS*
= 212 – 192
= 20
Part2(0,6) :
Jika nilai x_6 > *LIP***{**
True = 226 > 48, maka x6 – *LIS*
= 226 – 192
= 34
Part2(0,7) :
Jika nilai x_7 > *LIP***{**
True = 227 > 48, maka x7 – *LIS*
= 227 – 192
= 35
Seterusnya proses yang sama, untuk pengerjaan semua pixel part2 keabuan merah, hasilnya dapat terlihat seperti pada tabel 2.

Tabel 2. Ratio hasil kompresi.

Kompresi	ukuran		Ratio
Citra Asli	3	KB	1
10	2,88	KB	0,96
9	2,75	KB	1,00
8	2,63	KB	1,05
7	2,50	KB	1,10
6	2,38	KB	1,15
5	2,25	KB	1,21
4	2,13	KB	1,28
3	2,00	KB	1,35
2	1,88	KB	1,44
1	1,75	KB	1,53

Pada penelitian ini tidak melakukan pengujian dekompresi. Penelitian ini hanya mencoba menguji kompresi dengan SPIHT, untuk dekompresi sebenarnya adalah kebalikan dari kompresi, namun hasil dekompresi tidak akurat kembali sepertri semula, hal ini dikarenakan penerapan ambang batas (*threshold*). Berdasarkan urutan data citra matrik hasil partisi 10 yang di atas, jika didistribusikan dalam bentuk tabel, maka nilai citra matrik partisi 10 tersebut akan terlihat seperti tabel 3 di bawah ini:

Tabel 3. Struktur binner pohon Huffman

No Urut	Character	Frequency	Assignment	Space Savings
1	244	40	11110100	40×8 - 40×6 = 80 bits
2	245	31	11110101	31×8 - 31×6 = 62 bits
3	242	29	11110010	29×8 - 29×6 = 58 bits
4	215	24	11010111	24×8 - 24×7 = 24 bits
5	232	20	11101000	20×8 - 20×7 = 20 bits
6	237	19	11101101	19×8 - 19×7 = 19 bits
7	248	19	11111000	19×8 - 19×7 = 19 bits
8	251	19	11111011	19×8 - 19×7 = 19 bits
9	240	18	11110000	18×8 - 18×7 = 18 bits
10	241	17	11110001	17×8 - 17×7 = 17 bits
11	250	17	11111010	17×8 - 17×7 = 17 bits
12	235	17	11101011	17×8 - 17×7 = 17 bits
13	243	16	11110011	16×8 - 16×7 = 16 bits
14	224	16	11100000	16×8 - 16×7 = 16 bits
15	227	16	11100011	16×8 - 16×7 = 16 bits
16	158	16	10011110	16×8 - 16×7 = 16 bits

ISSN 2774-3659 (Media Online)
Vol 2, No 2, April 2022
DOI 10.47065/bulletincsr.v2i2.156 | Hal 34−40
https://hostjournals.com/bulletincsr

No Urut	Character	Frequency	Assignment	Space Savings
17	234	16	11101010	16×8 - 16×7 = 16 bits
18	228	15	11100100	15×8 - 15×7 = 15 bits
19	246	15	11110110	15×8 - 15×7 = 15 bits
20	239	15	11101111	15×8 - 15×7 = 15 bits
21	253	15	11111101	15×8 - 15×7 = 15 bits
22	236	15	11101100	15×8 - 15×7 = 15 bits
23	156	14	10011100	14×8 - 14×7 = 14 bits
24	233	14	11101001	14×8 - 14×7 = 14 bits
25	249	13	11111001	13×8 - 13×7 = 13 bits
26	183	13	10110111	13×8 - 13×7 = 13 bits
27	255	13	11111111	13×8 - 13×7 = 13 bits
28	229	11	11100101	11×8 - 11×8 = 0 bits
29	238	11	11101110	11×8 - 11×8 = 0 bits
30	168	10	10101000	10×8 - 10×8 = 0 bits
31	190	10	10111110	10×8 - 10×8 = 0 bits
32	252	10	11111100	10×8 - 10×8 = 0 bits
33	154	10	10011010	10×8 - 10×8 = 0 bits
34	178	10	10110010	10×8 - 10×8 = 0 bits
35	231	10	11100111	10×8 - 10×8 = 0 bits
36	167	10	10100111	10×8 - 10×8 = 0 bits
37	184	9	10111000	9×8 - 9×8 = 0 bits
38	185	9	10111001	9×8 - 9×8 = 0 bits
39	182	9	10110110	9×8 - 9×8 = 0 bits
40	179	9	10110011	9×8 - 9×8 = 0 bits
41	230	9	11100110	9×8 - 9×8 = 0 bits
42	56	8	00111000	8×8 - 8×8 = 0 bits
43	162	8	10100010	8×8 - 8×8 = 0 bits
44	164	8	10100100	8×8 - 8×8 = 0 bits
45	152	8	10011000	8×8 - 8×8 = 0 bits
46	153	8	10011001	8×8 - 8×8 = 0 bits
47	225	8	11100001	8×8 - 8×8 = 0 bits
48	226	8	11100010	8×8 - 8×8 = 0 bits
49	254	8	11111110	8×8 - 8×8 = 0 bits
50	186	8	10111010	8×8 - 8×8 = 0 bits
51	191	8	10111111	8×8 - 8×8 = 0 bits
52	49	7	00110001	7×8 - 7×8 = 0 bits
53	175	7	10101111	7×8 - 7×8 = 0 bits
54	187	7	10111011	7×8 - 7×8 = 0 bits
55	163	7	10100011	7×8 - 7×8 = 0 bits
56	165	7	10100101	7×8 - 7×8 = 0 bits
57	176	7	10110000	7×8 - 7×8 = 0 bits
58	188	7	10111100	7×8 - 7×8 = 0 bits
59	171	6	10101011	6×8 - 6×8 = 0 bits
60	57	6	00111001	6×8 - 6×8 = 0 bits
61	173	6	10101101	6×8 - 6×8 = 0 bits
62	151	6	10010111	6×8 - 6×8 = 0 bits
63	223	6	11011111	6×8 - 6×8 = 0 bits
64	172	6	10101100	6×8 - 6×9 = -6 bits
65	155	5	10011011	5×8 - 5×9 = -5 bits
66	180	5	10110100	5×8 - 5×9 = -5 bits
67	189	5	10111101	5×8 - 5×9 = -5 bits
68	137	5	10001001	5×8 - 5×9 = -5 bits
69	166	5	10100110	5×8 - 5×9 = -5 bits
144	99	1	01100011	1×8 - 1×11 = -3 bits
145	45	1	00101101	1×8 - 1×11 = -3 bits
146	35	1	00100011	1×8 - 1×11 = -3 bits
147	100	1	01100100	1×8 - 1×11 = -3 bits
148	27	1	00011011	1×8 - 1×11 = -3 bits
149	18	1	00010010	1×8 - 1×11 = -3 bits
150	17	1	00010001	1×8 - 1×11 = -3 bits
151	22	1	00010110	1×8 - 1×11 = -3 bits
152	38	1	00100110	1×8 - 1×11 = -3 bits

ISSN 2774-3659 (Media Online)
Vol 2, No 2, April 2022
DOI 10.47065/bulletincsr.v2i2.156 | Hal 34−40
https://hostjournals.com/bulletincsr

Maka dari data pada tabel 3 di atas dapat dihasilkan pengubahan data hexadesimal ke bilangan biner yang diurutkan dapat dilihat di bawah ini. Urutan biner yang ganjil berwarna merah jambu dan genap berwarna biru. Untuk warna-warna pada bit biner ini digunakan sebagai mempermudah melihat batas antar bit biner.

Gambar 4. Data biner ganjil dan genap

Dari uturan data biner di atas dapat dibandingkan dengan hasil kompresi Huffman yang telah berupa Huffman coding. Huffman coding diurutkan sesuai urutan karakternya dapat dilihat pada bagian bawah. Terlihat perubahan urutan berdasarkan warna yang dibentuk dari hasil encoding Huffman.

Gambar 5. Hasil encoding Huffman

4. KESIMPULAN

Berdasrkan hasil analisa yang telah dilakukan maka penulis mengambil kesimpulan yaitu berdasrkan prosedur kompresi dengan menggunakan algoritma SPIHT dan Huffman telah melakukan prose *file* gambar, yang menggunakan sampel berekstensi *JPEG sehingga aplikasi yang diharapkan dapat berjalan sesuai dengan teknik kompresi. Berdasarkan penelitian ini yang telah menggunakan algoritma SPIHT dan Huffman, telah membuktikan *file* gambar yang semulanya memiliki ukuran *file* besar dapat dikompresi menjadi ukuran yang kecil, dan akan memberikan manfaat ruang memori yang lebih sedikit.

REFERENCES

[1] K.Mahesa,"Rancang Bangun Aplikasi Kompresi Dan Dekompresi Pada Citra Digital Menggunakan Metode Huffman", Jurnal Processor, vol. 12, no. 1, pp. 997–1012, pekan baru, 2017.
[2] L. Marlina, A. Putera, U. Siahaan, H. Kurniawan, and I. Sulistianingsih, “Data Compression Using Elias Delta Code", Int. J. Recent Trends Eng, Res., vol. 3, no. 8, pp. 210–217, jakarta, 2017.
[3] Jubilee Enterprise,"Rahasia Manajemen File", PT Elex Media Komputindo, Jakarta, 2010.
[4] DMW Simamora, G Ginting, dan Y Hasan, Implementasi Algoritma Run Length Encoding Pada Kompresi File MP3, Jurikom, ISSN : 2407-389X Pp: 5-9, Medan, 2016.
[5] Piyu Tsai, Yu-Chen Hu b, Chin-Chen Chang, “Menggunakan Partisi Set Dalam Hierarki Hierarki Untuk Otentikasi Gambar Digital.” Pers, Pasal D I et al. 2003. 18: 813–22.
[6] K.M.S Eka Prayoga,”Implementasi algoritma Huffman dan Run Length Encoding pada aplikasi kompresi berbasis web,"J.ilmiah Teknol.Inf. Terap. vol.IV.2018.
[7] E. S. Mulyanta,"Dari Teori Hingga Praktik:Pengolahan Digital Image dengan Photoshop CS2", C.V Andi Offset, Yogyakarta, 2006.
[8] S. S. M. K. Dr.pulung Nurtantio Andono, S.T, M.Kom, T.sutojo,"Konsep Pengolahan Citra Digital", C.V Andi Offset, Yogyakarta, 2015.
[9] Darma Putra," Pengolahan Citra Digital", C.V Andi Offset, Yogyakarta, 2010.
[10] T. S. M. Pulung Nurtantio Andono," Pengolahan Citra Digital", C.V Andi Offset, Yogyakarta, 2017.