PENGGUNAAN ALGORITMA HUFFMAN UNTUK KOMPRES TEKS

Ari Pambudi
STMIK Buddi Tangerang
Jl. Arjuna Utara Tol Tomang Kebun Jeruk, Jakarta 11510
ari.pambudi@yahoo.com

Abstract

Method compression of text by using algorithm Huffman gives storage thrift of data. This method is algorithm that is most famous to compress of text. This algorithm applies system encoding with bit network, where character which the frequency is often is decoded with short bit network and character which the frequency seldom be decoded with long bit network. There is three steps in using algorithm Huffman that is first phase of forming of tree Huffman where at this phase by using this Huffman tree minimization of scanning path length will by the way of putting down information often emerges close to root and information which seldom emerge far from root, second phase is encoding that is way is compiling string binary read from root up to tree leaf Huffman, and third phase that is decoding returns from code which has been formed.

Keywords*: Huffman's Algorithm, Huffman Tree, Encoding, Decoding*

Pendahuluan

Dewasa ini hampir semua organisasi selalu berurusan dengan data, yang pada awalnya data di tulis pada kertas dan disimpan dalam kotak ataupun lemari. Adapun tujuan dari penyimpanan data tersebut agar data dapat digunakan kembali pada masa yang akan datang. Dengan berkembangnya teknologi terutama komputer, maka penyimpanan data sekarang ini mulai berubah kearah penyimpanan secara elektronik kedalam media penyimpan data.

Kehadiran disk baik *floppy disk*, *hardisk* ataupun *flash disk* dewasa ini sangat besar manfaatnya sebagai media penyimpan data, terutama *harddisk* yang digunakan untuk penyimpanan data dengan kapasitas besar. Data yang tersimpan di dalam *hard disk* akan selalu bertambah, karena setiap hari selalu ada transaksi yang disimpan ke dalam database, sehingga seiring dengan waktu berjalan akan semakin cepat penuh pula media penyimpan data tersebut.

Jika media penyimpanan data cepat penuh maka hal itu akan menuntut kita untuk segera menambah media penyimpanan data dengan yang baru, yang berarti perlu adanya pengeluaran untuk pembelian media penyimpanan data yang baru. Untuk mengatasi hal tersebut diperlukan suatu cara yang dapat mengoptimalkan system penyimpanan dengan cara mengompres data yang awalnya berukuran besar menjadi lebih kecil, sehingga menghemat media penyimpanan dan menyimpan daa lebih banyak.

Metode Algoritma Huffman adalah salah satu metode untuk dapat memperkecil ukuran data, dan merupakan pengembangan dari suatu bentuk struktur data, yaitu pohon Binary Huffman. Pohon binary Huffman biasanya digunakan pada system pengaksesan data di memory komputer, dapat diimplementasikan dalam bentuk lain yaitu untuk mengompres teks.

Metode ini sangat sederhana tidak perlu menggunakan rumus-rumus yang terlalu rumit, serta dimana pembentukan kode yang baru algoritma Huffman menggunakan binary tree yang mudah dipahami, kemudian untuk implementasi metode Huffman ini dapat dilakukan dengan menggunakan program sederhana.

Tinjauan Teori

Kompresi adalah sebuah cara untuk memadatakan data sehingga hanya memerlukan ruangan penyimpanan lebih kecil sehingga lebih efisien dalam menyimpannya atau mempersingkat waktu pertukaran data tersebut.

Dasar Algoritma Kompresi

Data digital yang disimpan ke dalam media peyimpanan merupakan sekumpulan angka ataupun karakter yang dikodekan dengan rumus tertentu pada permukaan disk. Informasi tersebut disimpan dalam unit yang disebut bit (binary digit), setiap bit terdiri dari angka 0 dan 1. karena lintasan-lintasan (track) dipermukaan disk lebarnya 8 bit (00000000) maka pengkodean data terkecil pada disk adalah sebesar 8 bit (1 byte).

Sistem pengkodean yang digunakan komputer untuk menyimpan data pada media penyimpanan adalah dalam bentuk kode ASCII (American Standard Code for Information Interchange) yan berjumlah 128 kode (2^8). Kode ini merupakan penyandian dari huruf-huruf alphabet, karakter numeric, simbo-simbol dan instruksi-instruksi kendali, dalam kode ASCII satu byte (8 bit) mewakili satu karakter. Tetapi dalam satu byte belum tentu semua bit berisi penuh dengan informasi terutama file-file yang menyimpan data berupa teks, kode-kode yang dipergunakan adalah kode ASCII berupa karakter alphabet, tanda baca dan numeric (sebanyak 94 karakter) yang bernomor 33-126 (00100001-01111110).

Pada pengcodean ASCII tidak semua bit dalam satu byte berisi penuh dengan informasi yaitu pada bit-bit paling kiri. Hal ini karena pengcodean data pada kode ASCII dimulai dari bit paling kanan, sehingga walaupun suatu kode tidak berjumlah 8 bit, tetap dikodekan 8 bit dengan meletakkan kode'0' pada bit-bit yang tersisa. Namun bit tambahan ini tidak mengandung informasi dan hanya sekedar melengkapi agar jumlah bit perkarakter tetap 8 bit. Penyimpanan pada kode ASCII memiliki kelemahan karena adanya bit-bit kosong yang tidak berisi dengan informasi, tetapi kelemahan tersebut dapat diatasi dengan mengubah system penyimpanan, dimana satu byte tidak mewakili satu kode atau karakter melainkan dapat diisi oleh beberapa kode terutama kode-kode yang pendek, sehingga semua bit dalam satu byte dapat berisi informasi.

Pada kode ASCII karakter-karakter teks tidak dimulai dari urutan pertama melainkan di mulai pada urutan ke 33 (00010001), sehingga kode dari karakter tersebut akan panjang. Penyimpanan pada kode ASCII dapat diperbaiki yaitu dengan membentuk kode baru yang lebih pendek dengan cara meletakkan karakter-karakter yang terdapat pada suatu file (terutama karakter teks) dimulai dari urutan pertama, sehingga kode yang dihasilkan menjadi lebih pendek, yang kemudian dalam system penyimpanannya satu byte dapat diisi dengan beberapa karakter.

Kode yang baru dapat dihasilkan lebih pendek dengan membentuk karakter-karakter yang terdapat pada suatu file ke dalam sebuah pohon biner dan dari pohon biner ini dihasilkan kode baru untuk masing-masing karakter. Pembentukan pohon biner ini dengan cara mengurutkan karakter-karakter yang ada, kemudian karakteryang berdekatan dan setingkat dihubungkan sehingga membentuk simpul gabungan lainnya, begitu seterusnya sampai berbentuk pohon biner dengan simpul terakhir.

Algoritma Huffman

Teknik ini dibuat oleh seorang mahasiswa MIT bernama David Huffman pada tahun 1952, merupakan salah satu metode paling lama dan paling terkenal dalam mengompres. Algoritma ini menggunakan system pengkodean dengan rangkaian bit, dimana karakater yang frekuensinya sering dikodekan dengan rangkaian bit yang pendek dan karakter yang frekuensinya jarang dikodekan dengan rangkaian bit yang panjang.

Media Penyimpanan *(Storage)*

Media peyimpanan merupakan suatu tempat yang di gunakan untuk menyimpan data, dimana pada computer media peyimpanan data digolongkan menjadi 2 yaitu:

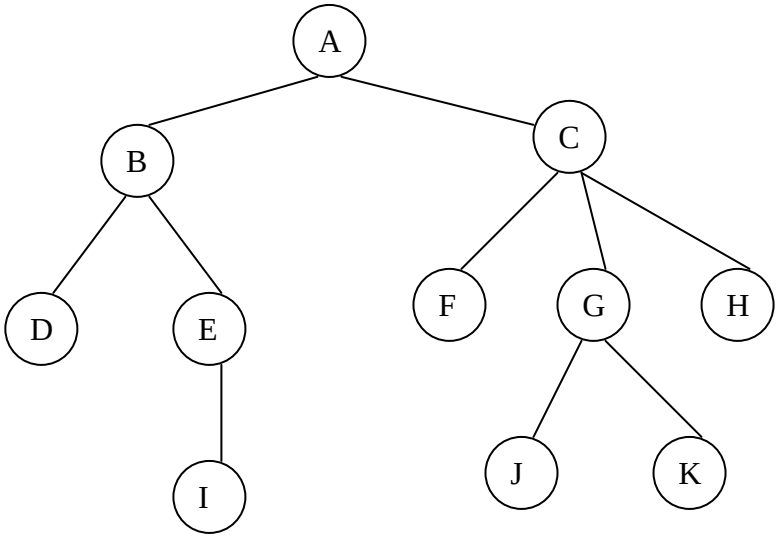
1. **Internal Storage**
   Sering disebut dengan istilah main memory, dimana memory ini hanyalah suatu tempat penampungan data sementara atau hanya diperlukan pada waktu proses sedang berlangsung. Data akan tersimpan pada main memory selama computer dihidupkan tetapi bila computer dimatikan maka data tersebut akan hilang.
2. **External Storage.**
   Adalah memory yang digunakan computer untuk menyimpan data yang bersifat per-

manent (tetap) dengan kapasitas yang besar namun bukan bagian integral dari komputer. Data yang berada dalam eksternal computer tidak langsung dikontrol komputer, karena komputer hanya memproses data yang berada pada main memory, sehingga agar data dapat segera diproses oleh komuter maka data yang berada pada eksternal memory dipindahkan terlebih dahulu ke dalam main meory. Proses pemindahan data ini dikendalikan oleh sebuah interface yang menghubungkan antara eksternal memory dengan main memory. Setelah berada dalam main memory baru bisa diproses oleh komputer, jika proses telah selesai maka data dikembalikan atau disimpan ke eksternal storage. Eksternal storage yang banyak digunakan adalah hard disk, floppy disk ataupun flash disk.

Pembahasan

Struktur Pohon

Merupakan satu bentuk struktur yang mempunyai ciri khusus, dinamai dengan istilah pohon atau tree karena struktur ini digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen data. Secara sederhana struktur pohon bisa didefinisikan sebagai kumpulan elemen yang salah satu elemennya disebut dengan akar (root) dan simpul yang terpecah menjadi sejumlah himpunan yang saling tidak berhubungan satu sama lain disebut subpohon atau cabang.



Sumber: Hasil Olahan Data

Gambar 1
Struktur Pohon

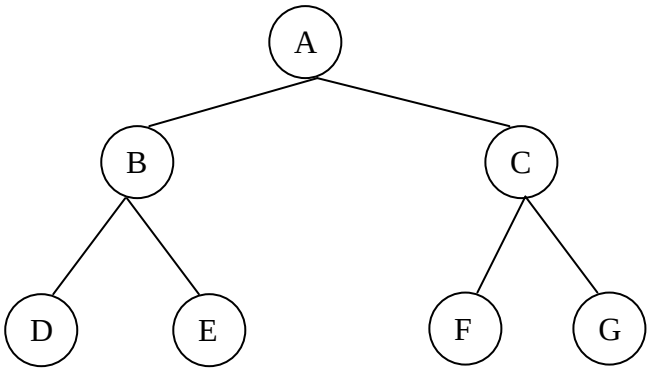
Pada struktur pohon tersebut diatas akarnya adalah A sub pohon yang pertama berakar B dan sub pohon kedua berakar pada C, selanjutnya pada sub pohon B mempunyai dua busa sub pohon yang berakar pada D dan E, sub pohon yang berakar pada E mempunyai sub pohon I, begitu seterusnya untuk sub pohon C, F, G, H dan J, K.

Dari gambar tersebut yangberisi informasi adalah simpul, terdapat 11 simpul yang berupa huruf A, B, C dan seterusnya sampai huruf K. Hubungan antara satu simpul dengan simpul lain dimisalkan seperti halnya sebuah keluarga yaitu ada anak, bapak, paman dan lain-lain. Dari gambar diatas siimpul A adalah bapak dari simpul B dan C. Dengan demikian B dan C bersaudara, begitu juga dengan simpul D dan E adalah anak dari simpul B. Dari urutan-urutan ini dalam struktur data simpul yang berdekatan mempunyai hubungan informasi yang langsung berkaitan.

Pohon Biner

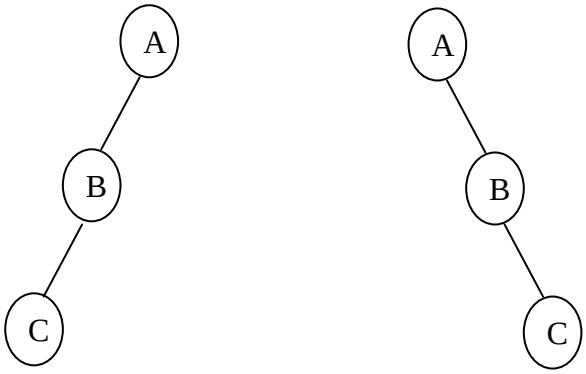
Pohon biner merupakan tipe yang sangat penting dari struktur data yang merupakan pengembangan dari struktur pohon, pohon biner

banyak dijumpai dalam berbagai terapan. Ciri penting yang dimiliki oleh pohon biner yaitu setiap simpul paling banyak hanya mempunyai dua cabang. Pohon biner akan membedakan cabang kiri dan kana, pada pohon biner ada kemungkinan tidak mempunyai cabang. Selain itu masih memungkinkan juga pohon biner hanya mempunyai cabng kanan atau cabang kiri saja yang disebut dengan pohon biner miring. Jika pada setiap simpul memiliki sepasang cabang kiri dan kanan, maka pohon ini disebut pohon biner lengkap.



Sumber: Hasil Olahan Data

Gambar 2
Pohon Biner Lengkap



Sumber: Hasil Olahan Data

Gambar 3
Pohon Biner Miring

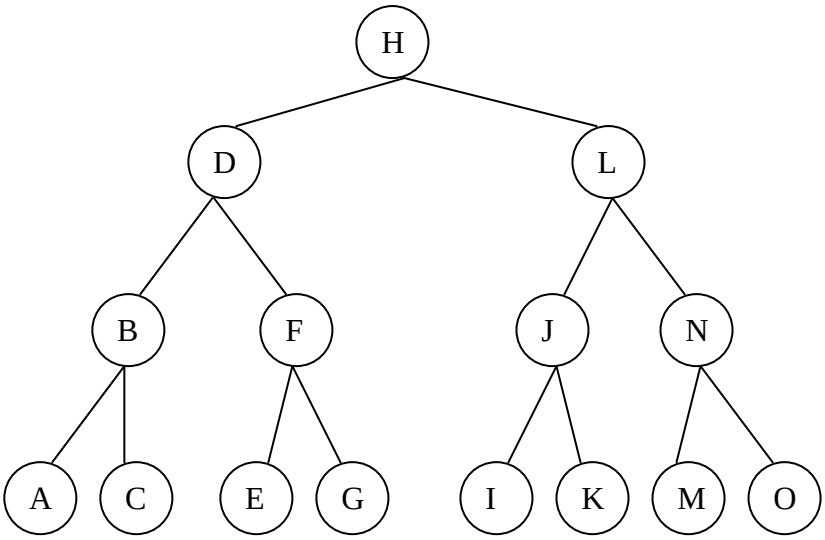
Pada pohon biner dapat dilakukan sejumlah operasi untuk mebaca infromasi yang disimpannya. Dengan melakukan kunjungan secara lengkap dan berurutan pada setiap simpul akan diperoleh informasi yang tersimpan pada pohon biner. Informasi yang tersimpan akan berbeda jika letak simpulnya ditukar atau cara penelusurannya tidak teratur, maka dalam melakukan penelusuran ada tiga yaitu:

a. Preorder
   Proses penelusuran dengan melakukan urutan kunjungan kecabang sebelah kiri, kesimpul dan kemudian kecabang sebelah kanan.
b. inorder
   Proses penelusuran dengan melakukan kunjungan dengan urutan kunjungan terhadap simpul, ke cabang sebelah kiri dan kemudian kunjungan ke cabang kanan.
c. postorder.
   Proses penelusuran dengan urutan kunjungan cabang sebelah kiri, ke cabang sebelah kanan dan kemudian kunjungi simpul.

Pembacaan Data pada Pohon Biner

Penelusuran data pada pohon biner umumnya lebih banyak dilakukan dengan cara preorder dimana penelusuran data dimulai pada puncak simpul (root) diteruskan kepada cabang sebelah kiri dan dilanjutkan terhadap cabang sebelah kanan. Namun pencarian data dengan cara ini kurang efesien, terutama terhadap node (daun) yang berada pada tingkat paling bawah akan memerlukan penelusuran yang lebih jauh disbandingkan dengan node yang terletak dekat dengan root (puncak), karena panjangnya jalur yang harus ditempuh.

Seandainya kita akan mencari suatu string yang berbunyi “BAIK” akan memerlukan pencarian sebanyak 11 langkah, yaitu 2 langkah untuk B dan masing-masing 3 langkah untuk A, I dan K. Sedangkan untuk mencari karakter “DFLJ” hanya membutuhkan 6 langkah. Dengan demikian huruf A yang berada pada posisi yang jauh dari akar akan membutuhkan langkah pencarian yang lebih panjang, sedangkan huruf A memiliki frekuensi lebih banyak. Hal ini kurang efisien terutama untuk waktu penelusuran, keadaan ini dapat diatasi sewaktu penyusunan pohon biner dengan meletakkan informasi yang sering muncul dekat dengan akar, dan informasi yang jarang muncul jauh dari akar.



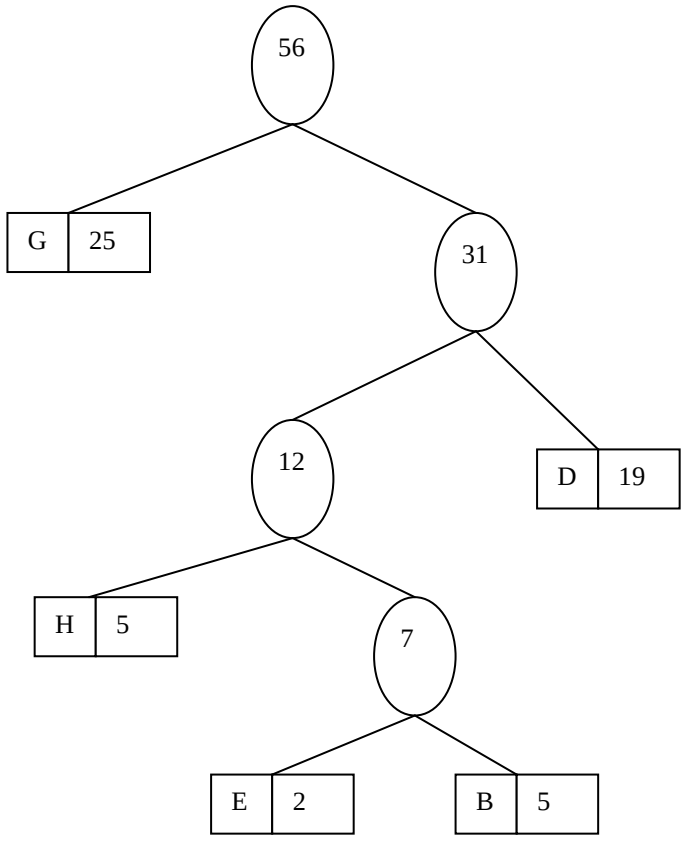
Sumber: Hasil Olahan Data

Gambar 4
Bentuk Data dalam Pohon Biner

Pohon Binary Huffman

Pohon Huffman dikemukan oleh seorang ahli yang bernama DA. Huffman, sehingga pohon yang terbentuk disebut dengan pohon Huffman. Pohon ini akan meminimalkan panjang lintasan penelusuran dengan cara meletakkan informasi yang sering muncul dekat dengan akar(root) dan informasi yang jarang muncul jauh dari akar. Sebagai contoh :
Diketahui ada N buah karakter dengan masing-masing karakter mempunyai frekuensi yang berbeda satu sama lain. Frekuensi yang terbesar menunjukkan bahwa karakter paling sering muncul dan frekuensi terkecil menunjukkan karakter paling jarang muncul. Maka dalam menyusun pohon telusurnya kita harus menempatkan frekuensi yang terbesar paling dekat dengan akar, dan frekuensi yang terkecil paling jauh dari akar, seperti gambar 5.

Pada gambar tersebut, karakter G berada paling dekat dengan akar karena memiliki frekuensi terbesar, sehingga sewaktu proses penelusuran untuk mencari huruf G hanya dibutuhkan 1 langkah. Begitu sebaliknya dengan karakter E dan B yang memiliki frekuensi terkecil berada pada posisi paling bawah. Penyusunan data seperti ini dapat meminimalkan panjang lintasan dalam proses pembacaan data.

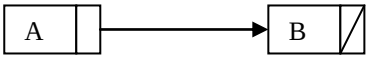


Sumber: Hasil Olahan Data

Gambar 5
Pohon Huffman

Tipe Data Pointer

Dalam mengeksekusi program, kebutuhan memory pada komputer selalu berubah-rubah, maak untuk itu diperlukan suatu tipe data yang dapat digunakan untuk meng-alokasikan ruang memory secara dinamis sesuai dengan kebuthan program. Perubahan alokasi memory secara dinamis ini akan menunjuk lokasi ruang memory lain, maka penunjuk antar lokasi memory ini disebut dengan pointer yang dpat diartikan sebagai "menunjuk ke sesuatu". Pointer adalah sebuah sel yang berisi alamat daris el memori lain dan dalam struktur data pointer digambarkan sebagai anak panah seperti digambarkan di bawah :

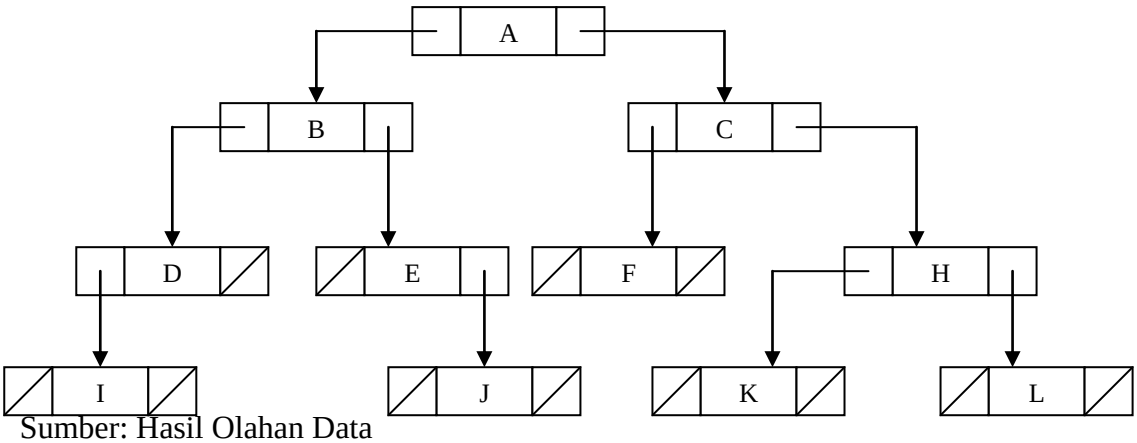


Sumber: Hasil Olahan Data

Gambar 6
Pointer

Pada gambar tersebut, panah berasal daris el A menuju sel B ini berarti informasi A berada pada sel B. Jika pada sel tidak mempunyai alamat memori lain maka dalam penggambarannya medan pointernya di isi garis miring (nil), seperti sel B diatas. Pointer ini dipakai dalam pembentukkan linked list (senarai berantai), dimana struktur linked list ini dalam memory computer dapat membentuk struktur pohon biner yang terdiri atas rangkaian elemen yang saling berkaitan antara satu sel dengan sel yang lainnya dihubungkan dengan pointer.

Dalam menyusun pohon biner pada memory, sel terdiri dari medan informasi dan dua pointer. Medan informasi berisi informasi, satu pointer menunjuk cabang ke kanan dan yang satunya lagi menunjuk ke cabang kiri sseperti gambar berikut :



Sumber: Hasil Olahan Data

Gambar 7
Pohon Biner dengan Linked List

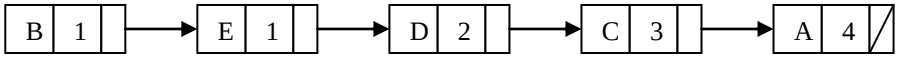
Algoritma Huffman

Selain digunakan untuk meminimalkan waktu penelusuran, teknik ini dapat diaplikasikan terhadap pengompresan file, dengan metode yang disebut algoritma Huffman. Pengompresan dilakukan dengan membuat kode baru terhadap karakter-karakter yang terdapat pada suatu file dengan cara membentuknya dalam pohon binary Huffman. Kode yang baru ini digunakan untuk menyimpan data. Sehingga data yang semula memerlukan ruang penyimpanan besar dapat diminimalkan.

Adapun proses dari algoritma Huffman dalam pembentukkan kode yang baru sebagai berikut :
Seandainya kita mempunyai sebuah teks string :CADEBAC ACAD" yang jika disimpan dalam kode ASCII adalah:
A = 01000001
B = 01000010
C = 01000011
D = 01000100
E = 01000101
Dalam penyimpanannya string diatas akan membutuhkan ruang penyimpanan sebesar 88 bit/11 byte. Namun penyimpanan string ini

dapat diminnimalkan dengan mengompesnya, yang tahapannya seperti berikut ini :

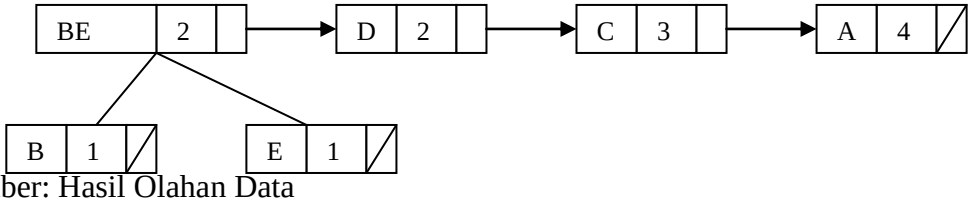
1. Melakukan penghitungan frekuensi dari maisng-masing karakter A, B, C, D dan E dimana frekuensi A= 4, B=1, C=3, D=2 dan E=1. Kemudian frekuensi ini diurutkan secara menaik, dari frekuensi terkecil samapai frekuensi terbesar.
2. Kemudian frekuensi yang sudah diurutkan masing-masing dibuat nodenya terdiri dari karakter dan frekuensinya yang dibentuk menjadi senarai berantai (linked list).



Sumber: Hasil Olahan Data

Gambar 8
Pengurutan Frekuensi dalam Linked List

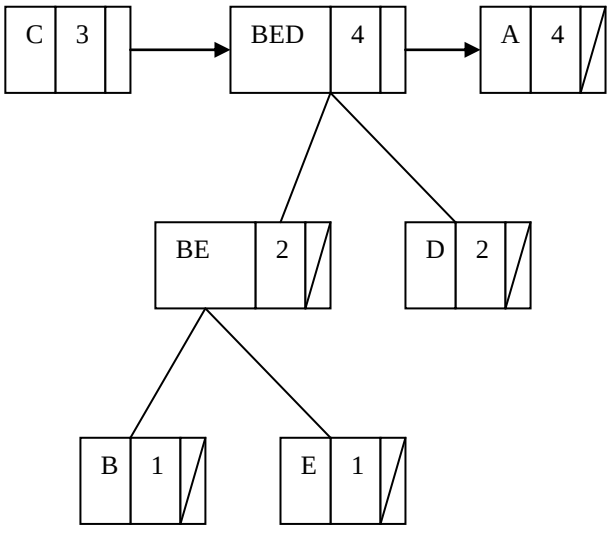
3. Selanjutnya adalah memilih dua node dengan frekuensi terkecil (dua node paling kiri), kedua node ini di lepas dari senarai berantai. Dari dua node ini dibentuk node baru yaitu (BE,2) berisi gabungan dua node terkecil, node baru ni menjadi akar (root) dari dua node yang lainnya (D, C, dan A) dalam linked list.



Sumber: Hasil Olahan Data

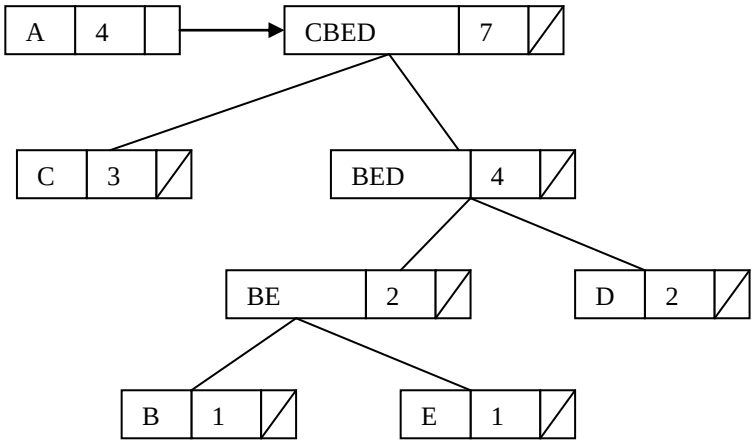
Gambar 9
Pembentukan Pohon Huffman 1

4. Berikutnya lakukan seperti langkah nomor 3 dengan memilih dua node terkecil yang masih berda dlam linked list yaitu BE, D, C dan A. Kemudian buat node baru (BED,4) yang menjadi root dari dua node terkecil yang dilepas dari linked list. Node yang baru terbentuk diurutkan dengan linked list bersama node C dan A . Demikian cara ini diulangi terus sampai semua linked list terbentuk menjadi binary tree, seperti gambar berikut :



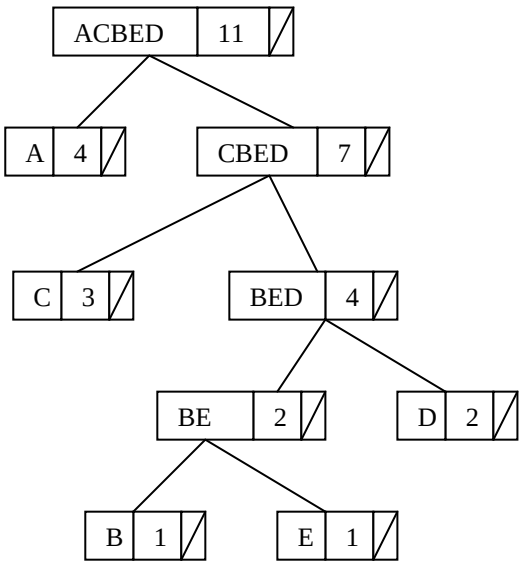
Sumber: Hasil Olahan Data

Gambar 10
Pembentukan Pohon Huffman 2



Sumber: Hasil Olahan Data

Gambar 11
Pembentukan Pohon Huffman 3



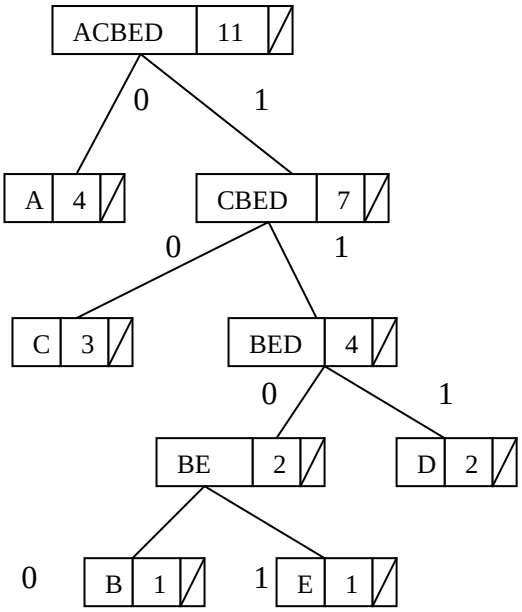
Sumber: Hasil Olahan Data

Gambar 12
Pembentukan Pohon Huffman 4

Pada gambar 12 ditemukan bentuk terakhir dari pohon Huffman dengan meletakkan karakter yang memiliki frekuensi terbesar dekat dengan akar dan karakter memiliki frekuensi terkecil paling jauh dari akar.

Pembentukkan Kode Huffman

Setelah pohon biner Huffman terbentuk dilanjutkan dengan pemberian kode ‘0’ pada cabang kiri dan kode ‘1’ pada cabang kanan. Hal ini dilakukan pada setiap percabangan pada setiap simpul sehingga akan diperoleh pohon Huffman dengan kode 0 dan 1 pada setiap cabangnya, seperti gambar berikut:



Sumber: Hasil Olahan Data

Gambar 13
Pengkodean Pohon Huffman

Untuk mendapatkan kode baru dari masing-masing karakter dengan pengkodean pohon binary Huffman, maka untuk karakter B dilakukan penelusuran dari akar (puncak) melewati semua cabang sampai kepada node b akan didapat kode ‘1100’. Begitu juga untuk karakter yang lainnya sehingga didapat kode baru

A = 0
B = 1100
C = 10
D = 111
E = 1101

Dari kode baru yang terbentuk maka string CADEBACACAD dapat diminimalkan penyimpanannya dengan mengkodekan menjadi :

C A D E B A C A C A D
10011111011100010010 0111= 24 bit

String diatas jika disimpan dengan kode ASCII akan membutuhkan ruang penyimpanan sebesar 88 bit, sedangkan jika disimpan dengan menggunakan kode Huffman membutuhkan ruang penyimpanan sebesar 24 bit atau 3 byte.

Pembuktian Hipotesa

Dari hipotesa yang menyatakan bahwa "Pengompresan efektif apabila pada suatu data jumlah karakternya sedikit dengan perbedaan yang besar antara karakter berfrekuensi terbesar dengn karakter berfrekuensi terkecil dan begitupun sebaliknya".

Sebagaimana diuraikan di atas bahwa pembentukan kode Huffman berdasarkan atas jumlah karakter-karakter yang terdapat dalam suatu data. Pembentukkan kode Huffman sangat dipengaruhi oleh beberapa factor dalam proses pengoptimalan terhadap penyimpanan data, faktor-faktor tersebut adalah :

- Jumlah karakter yang terdapat pada suatu file.
- Penyebaran atau beda frekuensi antar karakter pada suatu file, terutama karakter berfrekuensi terbesar dengan karakter berfrekuensi terkecil.

Dari dua faktor di atas maka terdapat beberapa kemungkinan yang akan dihasilkan pada proses pengoptimalan penyimpanan data :

Maksimal

Jika jenis karakter sedikit dan perbedaan yang besar antara karakter berfrekuensi terbesar dengan karakter berfrekuensi terkecil, seperti contoh berikut:

Tabel 1
Contoh Pembentukan kode Huffman

Karakter	Frekuensi	Pohon Huffman	Kode Huffman	Kode x Frek
A	10		00000	50 bit
B	5		000010	30 bit
C	8		000010	48 bit
D	30		0001	120 bit
E	15		00100	75 bit
F	20		00101	100 bit
G	40		0011	160 bit
H	190		01	380 bit
I	195		10	390 bit
J	212		11	424 bit
10	725	JUMLAH		1777 bit

Sumber: Hasil Olahan Data

Pada Tabel 1 diatas terdapat 10 karakter dengan perbedan yang besar antara karakter berfrekuensi terbesar dan karakter berfrekuensi terkecil yaitu 207 (212-5). Pada contoh ini perbedaan frekuensi cukup besar pembentukkan kode Huffman dapat mengoptimalkan panjang kode yang sebelumnya 725 byte menjadi 222 byte (1777 bit : 8). Pengoptimalan ini sangat efektif dengan menghemat lebih dari 500 byte, hal ini dikarenakan karkater-karakter yang berfrekuensi besar pada pohon Huffman berda sangat dekat dengan akar (level 20 sehingga menghasilkan kode yang sangat pendek (2 bit).

Kurang Maksimal

Jika jumlah karakter sedikit dan perbedaan yang kecil antara karakter berfrekuensi terbesar dengan karakter berfrekuensi terkecil seperti contoh berikut :

Tabel 2
Contoh Pembentukkan kode Huffman

Karakter	Frekuensi	Pohon Huffman	Kode Huffman	Kode x Frek
A	72		000	216 bit
B	73		001	219 bit
C	74		010	222 bit
D	75		011	225 bit
E	76		100	228 bit
F	77		101	231 bit
G	68		1100	272 bit
H	69		1101	276 bit
I	70		1110	280 bit
J	71		1111	284 bit
10	725	JUMLAH		2453

Sumber: Hasil Olahan Data

Pada table diatas terdapat 10 karakter dengan perbedaan yang kecil antara karakter berfrekuensi terbesar dan karakter berfrekuensi terkecil sebesar 9 (77-68). Dari contoh ini perbedaan frekuensi yang kecil mampu mengoptimalkan panjang kode yang sebelumnya 725 byte menjadi 307 byte (245 bit:8), pengoptimalan ini kurang optimal dibandingkan dengan kemungkinan pertama, karena karakter berfrekuensi terbesar berada jauh dari akar dan posisinya tidak jauh berbeda dengan karakter berfrekuensi terkecil (berada pada level 4 & 5), sehingga kode yang dihasilkan berfrekuensi besar masih cukup panjang.

Tidak Maksimal

Jika jumlah karakter besar dan perbedaan yang besar antara karakter berfrekuensi terbesar dengan karakter berfrekuensi terkecil. Kondisi ini pada proses pembentukan kode

Huffman tidak optimal, karena jumlah karakter yang banyak akan membentuk pohon Huffman yang panjang (dalam).

Karakter berfrekuensi terbesar memang berada dekat dengan akar, akan tetapi karakter berfrekuensi kecil akan berada pada tingkatan level sangat jauh dari akar, sehingga kode yang terbentuk untuk karakter berfrekuensi kecil akan panjang dan bisa melebihi 8 bit (melebihi kode ASCII).

Untuk pengoptimalan penyimpanan data kondisi seperti ini kurang baik, karena pada satu sisi terjadi penghematan oleh karakter berfrekuensi besar. Akan tetapi sebaliknya juga terjadi pemborosan oleh karakter-karakter berfrekuensi kecil sehingga pengoptimalan terhadap data sangat sedikit.

Hampir tidak ada Pengoptimalan

Jika jumlah karakter besar dan perbedaan yang kecil antara karakter berfrekuensi terbesar dengan karakter berfrekuensi terkecil. Kondisi ini pada pembentukkan kode Huffman sangat tidak baik, karena posisi dari karakter berfrekuensi besar dan karakter berfrekuensi kecil sama-sama berada pada tingkatan (level) yang jauh dari akar, sehingga kode baru yang terbentuk untuk masing-masing karakter akan panjang, mendekati panjang kode ASCII (8 bit) atau bahkan bisa melebihnya. Jika dilakukan pengoptimalan pada data seperti ini tidak akan efektif karena data yang dioptimalkan ukurannya tidak akan jauh berbeda dengan data yang asli.

Struktur Node Algoritma Huffman

Dalam penyusunan pohon Huffman pada senarai berantai, setiap node mempunyai 3 medan pointer. Dua medan (sel) menunjuk ke cabang kiri dan cabang kanan sedangkan pointer ke tiga digunakan untuk menggabungkan ke node berikutnya. Dimana struktur nodenya dapat digambarkan sebagai berikut:

Tabel 3
Struktur Node

Info	Frekuensi	Next
Kiri	Kanan	Kode

Sumber: Hasil Olahan Data

Untuk penyusunan program struktur node ini dapat dideklarasikan sebagai berikut :

```
struct node
{
char Info;
char kode;
long frekuensi;
struct node *kiri, * kanan, * next;
}
```

Dari deklarasi diatas node merupakan record yang terdiri drai 6 field yaitu info, kode, frekuensi, kiri, kanan dan next. Info berisi karakter, frekuensi berisi frekuensi karakter, kiri dan kanan merupakan pointer yang menunjuk cabang kiri dan kanan, sedangkan next adalah pointer untuk menggandeng node yang lain selama masih berada dalam linked list, sedangkan kode berisi kode dari pengcodean pohon Huffman.

Penyimpanan Kode Huffman pada disk

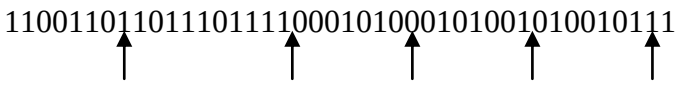
Pengompresan file dengan metode Huffman merupakan suatu model kompresi yang termasuk ke dalam phisikal kompresi, dimana penyimpanannya data tidak lagi disimpan dalam bentuk kode ASCII namun disimpan dalam kode yang baru. Akan tetapi computer dalam melakukan pembacaaan data menggunakan kode ASCII, untuk membaca data pada file yang terkompress dalam kode hufmmman, data tersebut harus diubah kembali ke dalam kode ASCII.

Untuk memudahkan proses pengembalian dat ke bentuk semula, maka pada file baru harus diikut sertakan daftar kode ASCII dari karakter yang dikodekan ke dalam kode Huffman. Maka dalam penyimpanannya pada awal file terkompresdibentuk sebuah table dari karakter-karakter yang terdapat dalam suatu file, yang disusun secara berurutan antar kode ASCII beserta kode Huffmannya dan jumlah bit dari kode Huffman. Hal ini sngat penting agar tidak terjadi kehilangan informasi dalam proses pengembalian data ke bentuk semula.

Setelah table dari kode terbentuk kemudian diikuti dengan penyimpanan data dalam kode Huffman. Sistem penyimpanan data dengan kode Huffman prosesnya dimulai dari bit paling kiri berbeda dengan kode ASCII yang dimulai dari kanan. Penyimpanan data pada

media penyimpanan tetap byte perbyte karena lebarnya track pada disk adalah 1byte, tetapi satu byte dapat diisi lebih dari satu karakter tergantung dari panjang kode Huffman yang terbentuk untuk masing-masing karakter.

Data yang berupa rangkaian bit dalam kode Huffman akan dibagi-bagi sebesar 8 bit yang kemudian disimpan ke dalam media penyimpanan seperti contoh berikut :



Pada rangkaian kode Huffman diatas data yang dibagi-bagi sebesar 8 bit akan disimpan pada lintasan/track disk. Pada contoh di atas byte terakhir hanya terdapat 4 bit yaitu 0111, penyimpanan kode ini harus tetap dalam 8 bit , bit kosong yang tersisa pada sebelah kanan akan diisi dengan bit '0' menjadi 01110000. Bit 0000 yang ditambahkan hanya berfungsi sebagai mencukupkan menjdi 8 bit namun tidak mengandung informasi. Pengisian dengan bit kosong ini hanya terjadi sekali yaitu akhir data, jika panjang rangkaian data dalam kode Huffman tidak habis dibagi 8. Hal ini berbea dengan kode ASCII yang kemungkinan akan terjadi hamper setiap byte penyimpanan terutama terhadap karakter-karakter ASCII 1 – 127. Jadi penyimpanan data dalam kode Huffman tidak akan ditemukan bit-bit kosong tanpa menyimpan informasi kecuali hanya pada akhir data.

Penguraian Kode Huffman

Keistimewaan dari pengkodean algo-ritma Huffman adalah kode yang dihasilkannya tunggal dan tidak terdapat suatu kode pendek yang akan membentuk kode yang lebih panjang. Ini berarti suatu kode pendek yang mewakili satu karakter tidak akan merangkap untuk kode lain yang lebih panjang, sebagai contoh 100 tidak akan membentuk kode baru seperti 10001. Hal ini dikarenakan sewaktu pembentukan pohon Huffman semua karakter tunggal berada pada daun dan tidak terdapat satupun karakter yang tunggal berada pada simpul. Keistimewaan kode Huffman ini sangat membantu sewaktu penguraiannya menjadi bentuk data yang semula ke dalam kode ASCII, sehingga pada proses pengambilan dank e bentuk semula tidak akan terjadi kesalahan atau kehilangan informasi yang tersimpan.

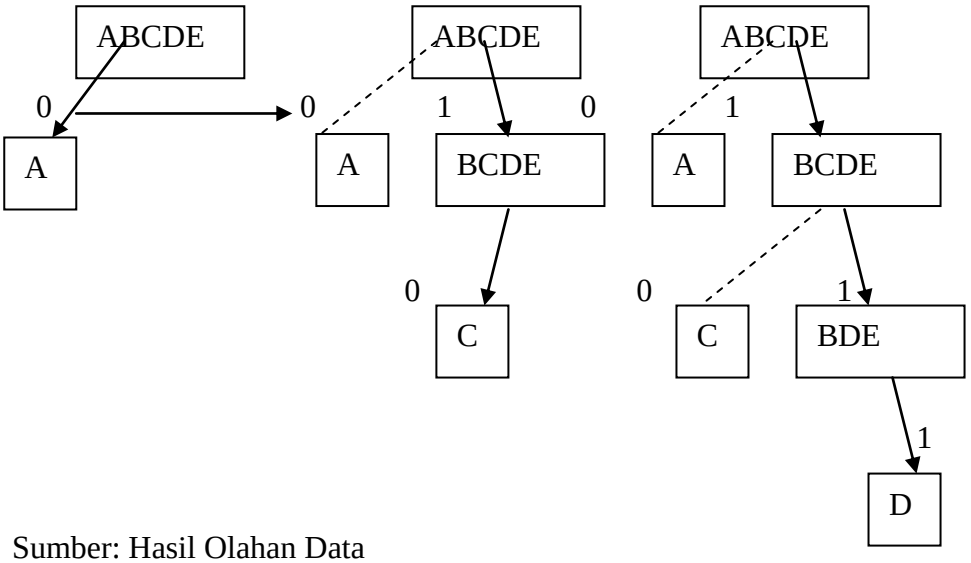
Dari contoh string 'CADEBACCAD' yang dikodekan dalam kode Huffman menjadi '1001111101110001001001111', untuk menguraikannya kembali ke bentuk semula, terlebih dahulu haruslah dibentuk pohon binary Huffman. Pembentukkan pohon Huffman ini berdasarkan table kode Huffman pada table 1, dimana kode Huffman adalah :

A	=	0
B	=	1100
C	=	10
D	=	111
E	=	1101

Dalam pembentukkan pohon Huffman yang pertama dibuat adalah node akar, kemudian pembentukkan cabang-cabang berdasarkan kode Huffman dari masing-masing karakter. Pembentukkan cabang dimulai dari kode terpendek kemudian secara berurut diikuti kode yang lebih panjang. Jika kode berupa bit '0' maka dibentuk cabang kiri, dan jika kode '1' dibentuk cabang kanan. Seperti pada table di atas kode terpendek adalah '0' (karakter A) maka dibentuk cabang kiri dari akar dengan kode '0' dan nodenya berisi karakter 'A'. Untuk kode berikutnya yaitu '10' (karakter C), berdasarkan kode ini dibentuk cabang kanan dengan kode '1' yang nodenya berisi BCDE, kemudian dilanjutkan dengan membentuk cabang kiri dengan kode '0' nodenya berisi karakter C. (Gambar 14).

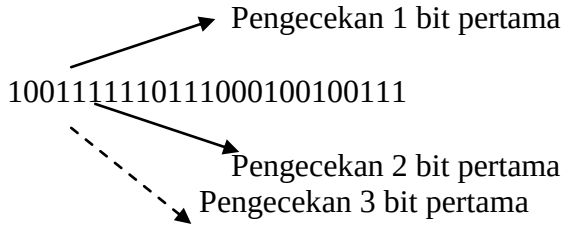
Proses pemberian kode ini selalu dimulai pada node akar dengan penelusuran mengikuti kode Huffman dari masing-masing karakter yang ada terletak pada node daun. Setelah pohon Huffman terbentuk kembali seperti pada gambar 10, maka untuk menguraikan data yang dalam kode Huffman ke bentuk semula, adalah dengan langkah-langkah sebagai berikut :

1. Penguraian kode dimulai dari awal data yaitu dari sebelah kiri rangkaian kode Huffman, seandainya penguraian dimulai dari kanan maka informasi yang dihasilkan akan salah.



Sumber: Hasil Olahan Data

Gambar 14
Pembentukkan pohon biner berdasarkan Kode Huffman

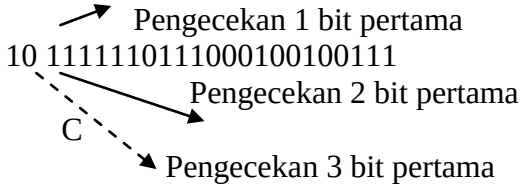
2. Dalam penguraian kode dilakukan dengan memilih-milih data bit per bit yaitu dengan melakukan penelurusan pada pohon Huffman yang trebentuk. Penelusuran dilakukan dari node akar dengan mengecek rangkaian kode terhadap pohon Huffman sampai ditemukannya karakter tunggal dari kode yang ada. Proses pengecekan dimulai dari bit awal rangkaian data seperti berikut:
3. 

Sumber: Hasil Olahan Data

Gambar 15
Penguraian kode Huffman langkah pertama.

Pada contoh di atas pengecekan pertama dilakukan terhadap bit'1' terhadap pohon Huffman, karena penelusuran kode 1 tidak berakhir pada node daun maka dilanjutkan pada level node berikutnya (tingkat lebih tinggi) disini ditemukan kode '0' yang berakhir pada node daun dengan karakter C. Berarti untuk kode 10 dari rangkaian kode Huffman terbentuk karakter C.

4. Setelah ditemukan karakter dari kode yang pertama, maka pengecekan dilanjutkan terhadap bit setelah kode 'C' dan kembali dilakukan penelusuran dari akar. Jika penelusuran tidak berakhir pada node daun dilanjutkan dengan node berikutnya sampai ditemukan nde tunggal (berakhir pada daun) seperti berikut :



Gambar 16
Penguraian kode Huffman langkah kedua

5. Pada contoh diatas pengecekan 1 bit kedua yaitu bit 0, sewaktu penelusuran pada pohon Huffman langsung berakhir pada daun dan ditemukan bahwa kode ini adalah karakter A.

Pada proses pengeckan dengan penelusuran pada pohon Huffman ini dilakukan terus sampai bit terakhir, maka akan diperoleh hasil seperti berikut:

10 0 111 1101 1100 0 10 0 10 0 111
C A D E B A C A C A D

Hasil dari penguraian kode Huffman ini kemudian dikembalikan ke bentuk semula dalam kode ASCII yaitu :

A = 01000001 D = 01000100
B = 01000010 E = 01000101
C = 01000011

Spesifikasi Program

Input: Nama file yang akan dikompres diikuti dengan ekstensionnya.

Proses: Proses pengompresan dari file terdiri dari beberapa tahap :

a. Pembacaan karakter apa saja yang terdapat dalams ebuah file yang akan di compress dan menghitung frekuensi dari masing-masing karakter tersebut.
b. Pembuatan node-node untuk masing-masing pasangan karakter di ikuti frekuensinya yang disusun dalam linked list dengan menggunakan pointer.
c. Pembuatan pohon Huffman terhadap node-node yang telah tersusun.
d. Pemberian tanda 0 dan 1 sebagai pengkodean terhadap pohon Huffman yang telah terbentuk.
e. Mengkorversi data yang dalam kode ASCII ke dalam kode Huffman menjadi file baru yang terkompress.

Output: Sebuah file baru yang telah terkompres berasal dari file input dengan ukuran yang telah berkurang dari file yang aslinya.

Implementasi Program

Pengompresan file dengan algoritma Huffman ini dapat diimplementasikan ke dalam program computer dengan menggunakan bahasa pemrograman C. Proses pengompresan ini dikelompokkan menjadi beberapa kelompok program yang terdiri dari proses pembaca data, pembuatan pohon Huffman, pengkodean terhadap pohon Huffman dan penguraian data dalam kode Huffman.

Proses Pembaca Data

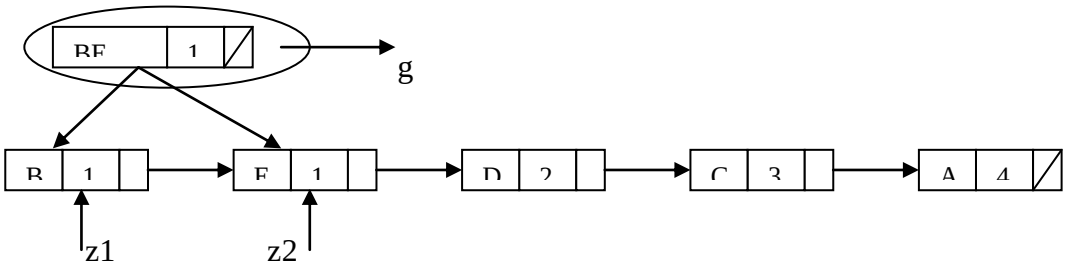
Proses ini adalah merupakan proses dari pembacaan data yang meliputi dari :

- Pembacaan data
- Penghitungan frekuensi masing-masing karakter
- Membentuk node dan mengurutkannya secara assending.

Proses Pembentukan Pohon Huffman

Pembentukan pohon Huffman merupakan proses penggabungan dua node dengan frekuensi terkecil sebagaimana yang telah diuraikan diatas. Maka dalam aplikasinya pembentukan pohon Huffman diawali dengan memberi tanda terhadap dua node terkecil dengan z1 dan z2, dimana z1 menunjukkan node frekuensi terkecil pertama dan z2 menunjukkan node berfrekuensi terkecil kedua. Untuk menggabungkan kedua node ini dibentuk node baru yaitu q yang isi dari node ini merupakan gabungan dari dua node berfrekuensi terkecil (z1 dan z2).

Node baru yang terbentuk mempunyai cabang kiri dan cabang kanan dimana cabang kiri adalah pointer kiri yang menunjuk z1, sedangkan cabang kanan adalah pointer kanan yang menunjuk z2.



Sumber: Hasil Olahan Data

Gambar 17
Pembentukkan Pohon pada memory

Setelah node yang baru terbentuk, selanjutnya node yang baru ini diurutkan kembali dengan node-node lain yang masih berada dalam linked list. Kemudian proses pembentukkan pohon Huffman dilanjutkan terhadap linked list yang masih ada dan proses ini berlangsung terus sampai semua node tidak lagi berbentuk linked list atau semua node terbentuk menjadi pohon Huffman.

Pengkodean Pohon Huffman

Proses pengkodean dilakukan terhadap pohon Huffman yang telah terbentuk untuk menandai pohon Huffman yang akan menghasilkan kode baru bagi masing-masing karakter. Pemberian kode dimulai dari kara (root) pohon Huffman, dengan mengikuti aturan telusur pohon preoder. Proses ini dimulai dengan memeriksa medan pointer kiri dan kemudian medan pointer kanan dari node root. Jika ditemui cabang pada pointer kiri, maka diberikan kode ‘0’ dan jika ditemukan cabang pointer kanan maka diberikan kode ‘1’, kode-kode ini ditempatkan dalam medan kode pada masing-masing node. Proses ini berlangsung terus sampai pada node paling akhir. Agar proses pengkodean dapat berlangsung sampai node yang terakhir baik cabang sebelah kanan maupun sebelah kiri
maka pada program ini dilakukan proses rekursif.

Pembacaan Ulang

Proses pembacaan ulang melakukan pembacaan ulang terhadap kode-kode yang terdapat pada pohon Huffman dan kemudian mengkonversi data pada file asli dalam kode ASCII menjadi kode Huffman. Kemudian menyimpan data tersebut menjadi file baru dalam bentuk kode Huffman.

Setelah selesai proses pembacaan ulang dan pembuatan file yang baru kemudian file yang asli dihapus dan nama file yang baru (file terkompres) diganti dengan nama file yang asli. Pada akhirnya proses ini akan dihasilkan sebuah file baru yaitu file yang telah dikompres dengan metode Huffman.

Rasio Kompresi

Sebuah program kompresi akan mengkompres ukuran dari sutau file sehingga akan menghasilkan file baru yang ukurannya lebih kecil dari pada file yang asli, proses dari pengompresan dapat digambarkan sebagai berikut :

Sumber: Hasil Olahan Data

Gambar 18
Proses Pengompresan

Dari gambar tersebut dapat dihitung rasio kompresi dari file yang dikompres, rumus ratio kompresi adalah dengan menghitung perban-dingan antar file yang telah dikompres dengan file asli sebelum di kompres yaitu:

$$\text{Rasio Kompresi} = \frac{\text{Besar File di kompres}}{\text{Besar File asli}} \times 100\%$$

Sumber: Hasil Olahan Data

Gambar 19
Rumus Rasio Kompresi

Karena pengompresan algoritma Huffman berdasarkan karakter-karakter yang terdapat pada suatu file, maka ratio kompresi sangat tergantung pada jumlah karakter dan perbedaan antara karakter berfrekuensi terbesar dengan karakter berfrekuensi terkecil. Semakin keciljumlah karakter maka akan semakin kecil rasio kompresi dan semakin besar beda antara karakter berfrekuensi terkecil dengan karakter berfrekuensi terbesar maka akan semakin kecil rasio kompresi. Semakin kecil rasio kompresi semakin besar penghematan yang dapat dilakukan terhadap ruang penyimpanan.

Kesimpulan

1. Sebuah bentuk struktur data yaitu pohon binary Huffman dapat diimplementasikan untuk aplikasi lain yaitu mengompres data
2. Pengompresan dilakukan berdasarkan jumlah frekuensi dari karakter-karakter pada suatu data, kemudian mengkodekan karakter-karakter tersebut menjadi kode baru yang lebih pendek, sehingga menghemat ruang penyimpanan data.
3. Rasio pengompresan akan semakin kecil apabila pada file yang dikompres jumlah karakternya sedikit tetapi memiliki perbedaan yang besar antara karakter berfrekuensi terbesar dengan karakter berfrekuensi terkecil. Sebaliknya rasio pengompresan akan semakin besar jika jumlah karakter lebih banyak dan memiliki perbedaan yang kecil antara karakter berfrekuensi terbesar dengan karakter berfrekuensi terkecil.
4. Pengompresan lebih efektif dilakukan terhadap kelompok file Database, file pengolah kata, dan file spreadsheet dengan ratio pengompresan lebih kecil.
5. Pengompresan tidak efektif dilakukan terhadap file eksekusi dan file program karena file hasil pengompresan ukuran tidak jauh berbeda dan bahkan bisa melebihi file asli.

Daftar Pustaka

Data Structures and Algorithm: Introduction: http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/introduction.html

Efraim Turban, R. Kelly Rainer, Richard E. Potter, "Pengantar Teknologi Informasi", Salemba, Jakarta, 2006.

Held, Gilbert, *"Data Compression Tecniques and Aplication hardware and Software Consideration"*, Georgia Jhon Willey LTD, 1987.

H.M. Jogiyanto, "Konsep Dasar Pemrograman Bahasa C", Andi Offset, Yogyakarta, 1993.

Huffman Coding, http://www.en.wikipedia.org/wiki/huffman_coding.

Kompresi Data, http://id.wikipedia.org/wiki/Kompresi_data.

M.Tenen Baum , Auron and J.Augensten, Moshe, Data Structures using Pascal, Englewood: Prentice Hall. Inc 1981.

Practical Huffman Coding, http://www.compressonsult.com/huffman/

Santoso, P. Insap, "Struktur Data Menggunakan Turbo Pascal 6.0", Andi Offset, Yogyakarta, 1993.

Struktur Data, http://id.wikipedia.org/wiki/Struktur_data

Suryanto, ony, "Pemampatan File dengan Algoritma Huffman", Dinastindo, Jakarta, 1995.

Slamet Sumantri, Nursalim, FX. Makaliwe, C. Hendrik, C Wibisono, Wahyu, "Pengantar Struktur Data", Elex Media Komputindo, Jakarta, 1993.