Penggunaan Algoritma Huffman dalam Kompresi Teks

Rachmawaty 13509071
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509071@std.stei.itb.ac.id

***Abstrak* —** *Makalah ini membahas tentang definisi dari algoritma Huffman atau kode Huffman, beberapa variasinya, dan cara pembuatan kode Huffman untuk Text Data Compressor. Algoritma Huffman sendiri merupakan salah satu algoritma kompresi yang terkenal terutama dalam mengompresi data berbentuk file teks. Prinsip yang digunakan oleh algoritma Huffman ini adalah karakter atau simbol yang sering mucul atau frekuensi kemunculannya paling tinggi direpresentasikan dengan kode (jumlah bit untuk string) yang paling pendek. Sedangkan untuk karakter atau simbol yang frekuensi kemunculanya paling rendah akan memiliki representasi string bit yang lebih pendek. Kode untuk setiap simbol juga tidak boleh merupakan awalan dari kode yang lain karena pada saat proses decoding bisa menimbulkan keraguan atau ambiguitas.*

Kata Kunci: *teks, algoritma Huffman, pohon Huffman, encoding, decoding*

I. PENDAHULUAN

Dewasa ini, kehidupan dengan dunia digital tidak bisa kita hindari lagi. Hal tersebut menyebabkan semakin membanjirnya data digital yang harus kita simpan, baik itu berupa *file* teks seperti pada *word*, .txt, *email* (surat elektronik) atau bahkan SMS (*Short Message Service*) yang tiap hari kita jumpai, bisa juga berupa *file* gambar dengan format JPEG, BMP, GIF, dan sebagainya, *file* lagu atau video dengan format seperti MP3, MP4, AVI, MOV, dan sebagainya. Semakin banyak data digital seperti contoh di atas yang kita simpan, maka akan semakin menghabiskan ruang kosong yang tersisa pada media penyimpanan data yang kita punya, seperti *hard disk*, CD, DVD, atau *flash disk*. Tempat penyimpanan data tersebut masih memiliki memori yang terbatas untuk kita menyimpan seluruh *file* digital yang kita punya. Meskipun sudah ada *hard disk* dengan kapasitas memori berukuran terabyte, tetap saja dia masih memiliki batasan memori, belum tidak terbatas.

Salah satu cara untuk menghemat ruang penyimpanan data pada media yang ada sekarang ini adalah dengan melakukan kompresi atau pemampatan data. Saat ini terdapat beberapa algoritma kompresi untuk memampatkan data, antara lain algoritma Huffman, LIFO, LZHUF, LZ77 dan variannya (LZ78, LZW, GZIP), *Dynamic* Markov *Compression* (DMC), *Block-Sorting Lossless, Run-Length*, Shannon-Fano, *Arithmetic*, *Prediction by Partial Matching* (PPM), Burrows-Wheeler *Block Sorting*, dan *Half Byte*. Tentunya masing-masing algoritma tersebut memiliki keunggulan yang berbeda dalam pemampatan data. Dalam makalah ini hanya akan dibahas lebih lanjut tentang salah satu algoritma di atas, yakni algoritma Huffman yang biasa digunakan untuk kompresi data, terutama data berupa *file* teks. Penjelasan tentang algoritma Huffman untuk pemampatan data berbentuk teks akan dipaparkan pada bab-bab selanjutnya.

II. DEFINISI DATA DIGITAL, TEKS, KOMPRESI ATAU PEMAMPATAN DATA, ENCODING, DAN DECODING

A. Data Digital

Data merupakan fakta, atau bagian dari fakta yang mengandung arti yang dihubungkan dengan kenyataan, simbol-simbol, gambar-gambar, kata-kata, angka-angka, huruf-huruf, atau simbol-simbol yang menunjukkan suatu ide, objek, kondisi atau situasi dan lain-lain. Data sendiri merupakan bentuk jamak dari datum yang berarti informasi. Jelasnya data itu dapat berupa apa saja dan dapat ditemui di mana saja.

Digital berasal dari kata *digitus*, dalam bahasa Yunani berarti jari jemari. Apabila kita hitung jari jemari orang dewasa, maka berjumlah sepuluh (10). Nilai sepuluh tersebut terdiri dari 2 radix, yaitu 1 dan 0. Oleh karena itu, digital merupakan penggambaran dari suatu keadaan bilangan yang terdiri dari angka 0 dan 1 atau *off* dan *on* (bilangan biner). Semua sistem komputer menggunakan sistem digital sebagai basis datanya atau dapat pula disebut dengan istilah bit (*binary digit*).

Maka, data digital merupakan suatu metoda penyimpanan dalam lingkungan komputer, dimana data tersebut disimpan dan diolah secara digital. Data digital banyak ragamnya, seperti yang telah disebutkan pada bab pendahuluan, yaitu seperti *file* teks, gambar digital, lagu, video, animasi, dan lain sebagainya yang juga memiliki beragam ekstensi untuk tiap jenis data digital tersebut.

B. Teks

Teks adalah data yang terdiri dari karakter-karakter yang menyatakan kata-kata atau lambang-lambang untuk berkomunikasi oleh manusia dalam bentuk tulisan. Di dalam sistem komputer, teks ini dikodekan dalam suatu standardisasi, seperti ASCII, dimana pada kode tersebut terdapat nilai numerik maupun angka maupun tanda baca,

serta simbol lainnya. Teks ini digunakan oleh sistem komputer untuk penyimpanan maupun dalam proses pengiriman datanya. Teks sendiri dibedakan berdasarkan nilai biner yang diolah sedemikian rupa oleh komputer. Contoh dari teks ini adalah karakter yang diperlakukan sebagai karakter abjad yang biasa kita baca. misalnya pada lembar kerja, rumus, dan lain-lain.

C. *Kompresi atau Pemampatan Data*

Kompresi atau pemampatan data adalah sebuah cara untuk memadatakan data sehingga data tersebut berukuran lebih kecil. Kompresi data bertujuan untuk mengurangi ukuran data tanpa merusak tujuan dari data tersebut Hal ini membuat data yang dimampatkan memerlukan ruang penyimpanan yang lebih kecil sehingga lebih efisien dalam menyimpannya dan tidak memakan memori yang banyak pada tempat penyimpanan data kita. Dengan melakukan pemampatan atau kompresi data juga dapat mempersingkat waktu pertukaran data atau waktu transmisi data tersebut. Contoh nyatanya adalah kebutuhan akan kecepatan akses internet dewasa ini. Apabila data yang dikirim berukuran sangat besar, maka waktu transmisi data tersebut akan lama, untuk itulah dibutuhkan kompresi atau pemampatan data sehingga kita bisa melakukan pengiriman data via internet dengan lebih cepat. Contoh perangkat lunak yang dapat kita gunakan untuk melakukan kompresi atau pemampatan antara lain Winzip/Winrar, 7-Zip, IZArc, dan lain sebagainya.

D. *Encoding*

Encoding adalah proses untuk mengubah sinyal ke dalam bentuk yang dioptimasi untuk keperluan transmisi data atau penyimpanan data. *Encoding* dalam bahasa Indonesia disebut sebagai penyandian.

E. *Decoding*

Decoding atau dalam bahasa Indonesia disebut penguraian kode merupakan kebalikan dari *encoding*, yaitu merupakan sebuah proses untuk menyusun kembali data yang telah dikodekan sebelumnya sehingga informasi yang diterima dapat dibaca dan diolah.

III. ALGORITMA HUFFMAN, VARIASI ALGORITMA HUFFMAN, DAN POHON HUFFMAN

A. *Algoritma Huffman*

Algoritma Huffman atau biasa juga disebut kode Huffman dikembangkan oleh David A. Huffman saat beliau masih menjadi seorang mahasiswa Ph.D di sebuah institut teknologi terkemuka di dunia, yakni MIT (Massachussets Institute of Technology). Algoritma Huffman dipublikasikan pada tahun 1952 dalam *paper*nya yang berjudul "*A Method for the construction of minimum-redudancy codes*".

Algoritma atau kode Huffman menggunakan sebuah metoda yang spesifik dalam pemilihan representasi untuk tiap simbol, yang menghasilkan kode berupa jumlah bit untuk string yang merepresentasikan simbol dari sumber data yang paling sering muncul atau yang frekuensi kemunculannya dalam sumber data tinggi menggunakan kode dalam bentuk jumlah bit untuk string yang lebih pendek atau lebih sedikit dibandingkan dengan simbol yang frekuensi kemunculannya rendah. Huffman berhasil membuat metode kompresi yang paling efisien. Tidak ada pemetaan lain dari sumber simbol yang individual ke bit yang unik untuk string akan menghasilkan sebuah keluaran dengan rata-rata ukuran yang lebih kecil. Sebuah metoda yang kemudian ditemukan untuk menciptakan sebuah kode Huffman dalam waktu yang linier jika probabilitas masukan (disebut juga sebagai bobot) telah diurutkan.

Untuk himpunan dari simbol-simbol dengan distribusi probabilitas yang seragam, kode Huffman ekuivalen dengan *block encoding* biner sederhana, seperti kode ASCII. Meskipun algoritma Huffman yang asli optimal untuk kode simbol-simbol (contohnya sekumpulan simbol-simbol yang tidak berkaitan) dengan sebuah masukan yang diketahui probabilitas distribusinya, algoritma ini tidak optimal jika aturan untuk simbol-simbol ini tidak ada atau ketika bobot dari fungsi probabilitasnya tidak diketahui, tidak terdistribusi merata, atau tidak independen. Kode untuk setiap simbol yang dihasilkan tidak boleh merupakan awalan dari kode yang lain karena pada saat proses decoding bisa menimbulkan keraguan atau ambiguitas.

B. *Variasi Algoritma Huffman*

Beberapa variasi dari algoritma Huffman, yaitu:

1. n-*ary* Huffman Coding
   Algoritma ini menggunakan alphabet {0,1,…,n-1} untuk *encoding* suatu pesan dan membuat pohon n-*ary*. Pendekatan ini merupakan ide dari Huffman yang terdapat pada *paper*nya yang asli. Algoritma yang sama dengan yang digunakan pada kode biner (n=2), kecuali pada n buah simbol yang kemunculannya paling sedikit diambil secara bersamaan, tidak diambil 2 yang frekuensi kemunculannya paling sedikit saja. Untuk n>2, tidak semua himpunan dati sumber kata-kata dapat membentuk sebuah pohon n-*ary* untuk kode Huffman dengan tepat. Pada kasus ini, tambahan probabilitas 0 harus ditambahkan. Hal ini dilakukan karena pohon harus membentuk kontraktor n ke 1. Pada kode biner, ini merupakan kontraktor 2 ke 1, himpunan dengan ukuran berapapun dapat membentuk kontraktor ini. Jika jumlah dari sumber kata-kata kongruen dengan 1 modulo n-1, maka dapat membentuk pohon Huffmam yang benar.

2. *Adaptive* Huffman *Coding*
   Sebuah varias iyang disebut *adaptive* Huffman *coding* melingkupi perhitungan probabilitas secara dinamis berdasarkan frekuensi terakhir dari sekuens sumber simbol-simbol, dan perubahan struktur pohon kode untuk mencocokannya dengan estimasi probabilitas yang baru.

3. Huffman *template Algorithm*
   Sering kali, bobot yang digunakan dalam implementasi kode Huffman merepresentasikan probabilitas numeric, tetapi algoritma yang diberikan tidak memerlukannya. Yang diperlukan hanya sebuah cara untuk mengurutkan bobot dan menambahkannya. Algoritma yang menggunakan dasar algoritma Huffman dapat menggunakan beragam jenis bobot (harga, frekuensi, bobot sepasang, bobot tidak numerik) dan mengkombinasikan banyak metoda (tidak hanya penjumlahan). Algoritma seperti ini dapat menyelesaikan masalah minimalisasi lain, seperti masalah pada desain rangkaian digital.

4. *Length-limited* Huffman *Coding*
   Kode Huffman dengan panjang dibatasi adalah salah satu varian Kode Huffman dengan tujuan yang sama yaitu membangun sebuah pohon dengan panjang lintasan berbobot minimum, tetapi ada tambahan aturan, yaitu panjang tiap kode harus lebih kecil dari konstanta tertentu yang diberikan.

5. Huffman *Coding* with *Unequal Letter Costs*
   Pada masalah penentuan kode Huffman biasa, diasumsikan tiap simbol pada sumber dibangun dengan harga yang sama. Kode yang memiliki panjang N, akan selalu memiliki harga N, tidak peduli berapa banyak digitnya yang 0, dan berapa banyak digitnya yang 1, dan lainnya. Dengan asumsi seperti ini, meminimalisasi harga total suatu pesan dan meminimalisasi jumlah digit merupakan hal yang sama.

   Kode Huffman dengan bobot sumber tidak sama merupakan generalisasi, dimana asumsi tersebuat tidak lagi berlaku. Huruf-huruf pada sumber dari *encoding* alfabet, mungkin memiliki panjang yang tidak seragam, karena karakteristiki tertentu dari media transmisi. Contohnya adalah *encoding* alfabet pada sandi Morse, dimana sebuah ‘garis’ memakan waktu pengiriman yang lebih lama daripada sebuah ‘titik’, dengan demikian bobot garis pada pengiriman lebih besar. Tujuannya adalah untuk meminimalisasi panjang kode rata-rata. Tetapi tidak cukup hanya dengan meminimalisasi jumlah symbol yang digunakan dalam pesan. Belum ada algoritma yang diketahui dapat menyelesaikan masalah ini dengan tingkat efisiensi yang sama dengan kode Huffman konvensional.

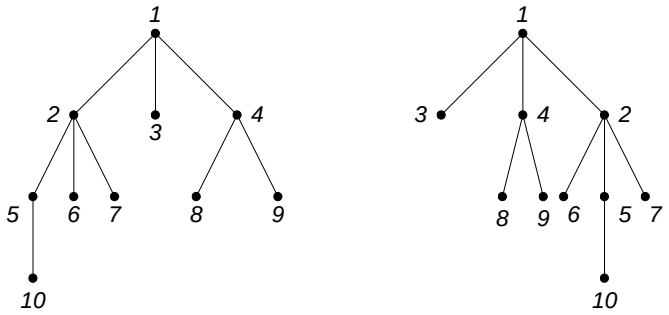
6. *Optimal Alphabetic Binary Trees* (Hu-Tucker *Coding*)
   Pada masalah penentuan kode Huffman yang biasa, diasumsikan bahwa setiap kode dapat mengacu pada masukan simbol apapun. Pada versi alfabetis ini, urutan alfabetis pada masukan dan keluaran harus sama atau identik. Ini juga dikenal sebagai Kode Hu-Tucker. Pohon biner alfabetis optimal ini sering digunakan sebagai pohon pencarian biner (BST/*Binary Search Tree*).

7. *The Canonical* Huffman *Code*
   Kode yang didapatkan dari sumber yang diurutkan kembali, yang disebut Kode Huffman kanonik ini adalah kode yang sering digunakan, karena kemudahannya dalam *encoding*/*decoding*. Teknik menentukan kode ini sering disebut Kode Huffman-Shannon-Fano, karena optimalitasnya seperti Kode Huffman dan probabilitas bobotnya yang alfabetis seperti Kode Shannon-Fano. (P.S. Robert M. Fano adalah dosen David A. Huffman ketika ia mahasiswa doctoral. Sedangkan Claude Shannon adalah kolega Fano yang sekaligus penemu keilmuan teori informasi (*information theory*)).

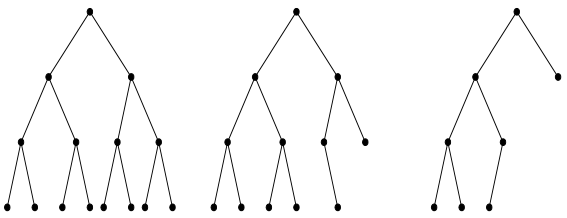
C. Pohon Huffman

Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Sedangkan,graf secara matematis didefinisikan sebagai berikut:

> Graf G didefinisikan sebagai pasangan himpunan (V, E), yang dalam hal ini:
>
> V=himpunan tidak kosong dari simpul-simpul (*vertices* atau *node*)={v_1, v_2, …,v_n}
> dan
> E=himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul={e_1, e_2, …,e_n}
>
> atau dapat ditulis singkat notasi G=(V,E).



Terdapat beberapa jenis pohon, antara lain pohon merentang, pohon berakar, pohon terurut, pohon n-*ary*, dan pohon biner. Kode Huffman sendiri merupakan salah satu bentuk aplikasi atau terapan dari pohon biner. Pohon biner merupakan kasus khusus pohon n-*ary* jdengan n=2.



Ilustrasi pohon biner dapat dilihat pada gambar di atas. Pohon biner adalah pohon yang setiap simpul cabangnya mempunyai paling banyak dua buah anak. Untuk memperoleh kode Huffman diperlukan pembentukan pohon Huffman terlebih dahulu. Langkah-langkah untuk mendapatkan kode Huffman dengan menggunakan pohon Huffman ini akan dijelaskan pada bab selanjutnya.

IV. Cara Mendapatkan Kode Huffman beserta Contoh Penyelesaian Kasusnya

Bila kita memiliki sebuah teks. Maka kita bisa memapatkannya dengan kode Huffman. Teknik yang dilakukan adalah dengan menciptakan sebuah pohon biner yang terdiri dari beberapa simpul atau *nodes*. Simpul tersebut dapat berupa simpul daun atau simpul internal.

Pertama, kita membuat sebuah tabel yang berisikan daftar simbol yang ada. Kemudian kita menghitung frekuensi kemunculannya atau kekerapannya. Lalu kita hitung peluang atau probabilitasnya. Dari tabel yang kita buat ini, kita akan lebih mudah untuk membuat pohon Huffman.

Proses dimulai dengan simpul-simpul daun yang memiliki kerapatan dari simbol yang mereka representasikan. Kemudian, sebuah simpul baru yang anak-anaknya adalah dua buah simpul dengan kerapatan yang paling rendah, dan kerapatan dari simpul yang baru setara dengan penjumlahan dari kerapatan anak-anaknya. Dengan dua buah simpul sebelumnya digabungkan menjadi sebuah simpul yang baru, prosedur yang sama diulang hingga tersisa hanya satu buah simpul, yakni pohon Huffman.

Algoritma pembuatannya yang paling sederhana menggunakan *priority queue* dimana simpul dengan probabilitas paling rendah diberikan prioritas paling tinggi:

1. Buat sebuah simpul daun untuk tiap simbol dan tambahkan ke dalam prioritas antrian.
2. Selama masih ada lebih dari satu simpul dalam antrian:
    1. Buang dua simpul dengan prioritas paling tinggi (peluang paling kecil) dari antrian.
    2. Buat sebuah simpul internal baru dengan kedua simpul tadi sebagai anak dan dengan peluang yang setara dengan penjumlahan peluang dari kedua simpul tadi.
    3. Tambahkan simpul baru ke dalam antrian.
3. Simpul yang tersisa merupakan simpul akar dan pohon pun telah lengkap.

Dengan lebih sederhana, algoritma pembentukan pohon Huffman:

1. Pilih dua simbol dengan peluang paling kecil dan kita buat menjadi daun. Kedua simbol ini dikombinasikan sebagai simbol orang tua, dengan peluang merupakan jumlah peluang kedua anaknya.
2. Selanjutnya, pilih dua symbol berikutnya, termasuk symbol baru, yang mempunyai peluang terkecil.
3. Ulangi langkah 1 dan 2 sampai seluruh simbol habis.

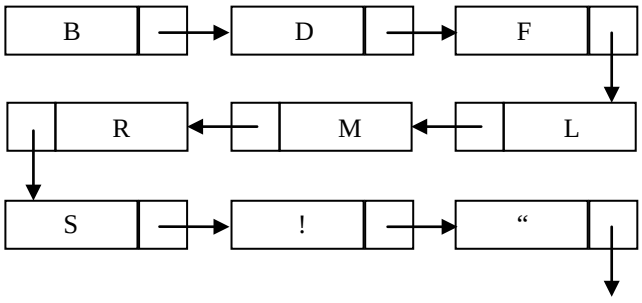
Contoh dari pembuatan pohon Huffman hingga memperoleh kode Huffman dipaparkan di bawah ini.

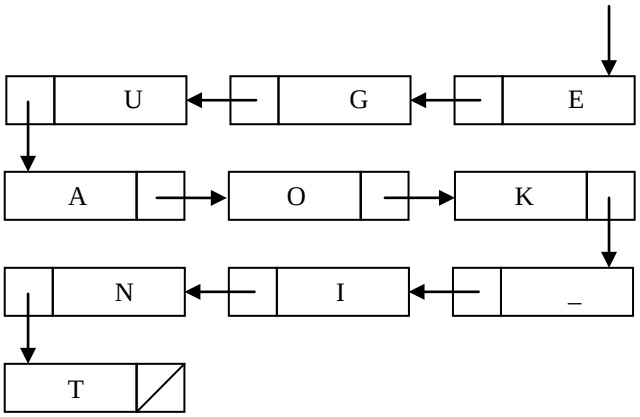
Bila kita mempunyai teks seperti di bawah ini: “TEKNIK_INFORMATIKA_INSTITUT_TEKNOLOGI _BANDUNG!”

Pertama, kita membuat tabel sebagai berikut:

Simbol	Kekerapan	Peluang	Kode Huffman
A	3	3/48	
B	1	1/48	
D	1	1/48	
E	2	2/48	
F	1	1/48	
G	2	2/48	
I	6	6/48	
K	4	3/48	
L	1	1/48	
M	1	1/48	
N	6	6/48	
O	3	3/48	
R	1	1/48	
S	1	1/48	
T	6	6/48	
U	2	2/48	
“	2	2/48	
_	4	4/48	
!	1	1/48	

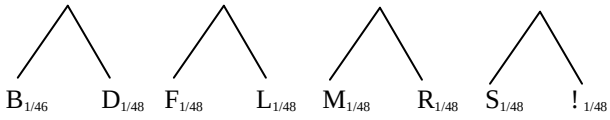
Kode Huffman akan kita isi terakhir setelah pohon terbentuk. Sekarang setelah tabel tersebut dibuat, yang kita lakukan adalah mencari symbol yang memiliki peluang terkecil. Bila dibuat *priority queue*, maka antrian yg terbentuk adalah sebagai berikut:



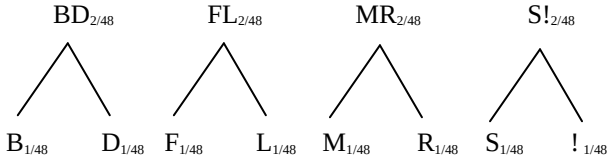


Pada tabel sebelumnya dapat kita lihat bahwa peluang terkecil adalah 1/48 dan yang memiliki peluang 1/48 ada 8 buah simbol, yaitu ‘B’, ‘D’, ‘F’, ‘L’, ‘M’, ‘R’, ‘S’, dan ‘!’.

Kita bisa membuat daun dari delapan simbol tersebut.

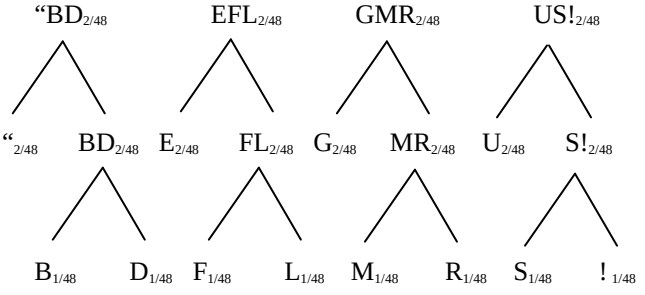


Kemudian kita membuat orangtua dari tiap dua daun tersebut menjadi seperti ini dengan probabilitas merupakan penjumlahan dari kedua anaknya.

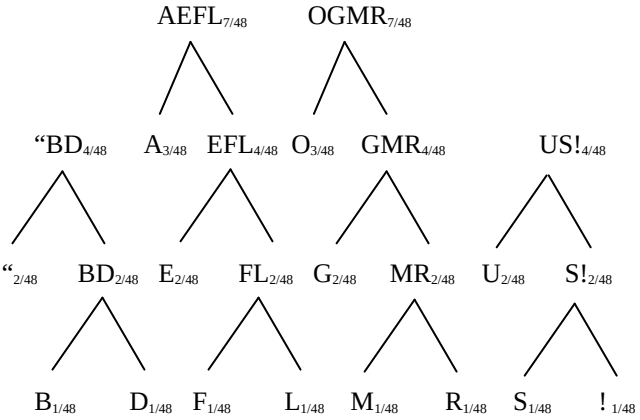


Setelahnya, symbol yang telah dimasukkan ke dalam pohon Huffman dihapus dari antrian. Seperti *queue* yang memegang prinsip FIFO, yakni *First In First Out.*Di mana simbol dengan prioritas paling kecil di taruh pertama pada antrian (dalam kasus ini B) akan dihapus paling pertama dari antrian.

Sekarang ada 3 buah simbol dengan peluang yang sama 2/48, yaitu ‘E’, ‘G’, ‘U’, dan ‘ “ ’. Lalu, kita gabungkan menjadi seperti ini.

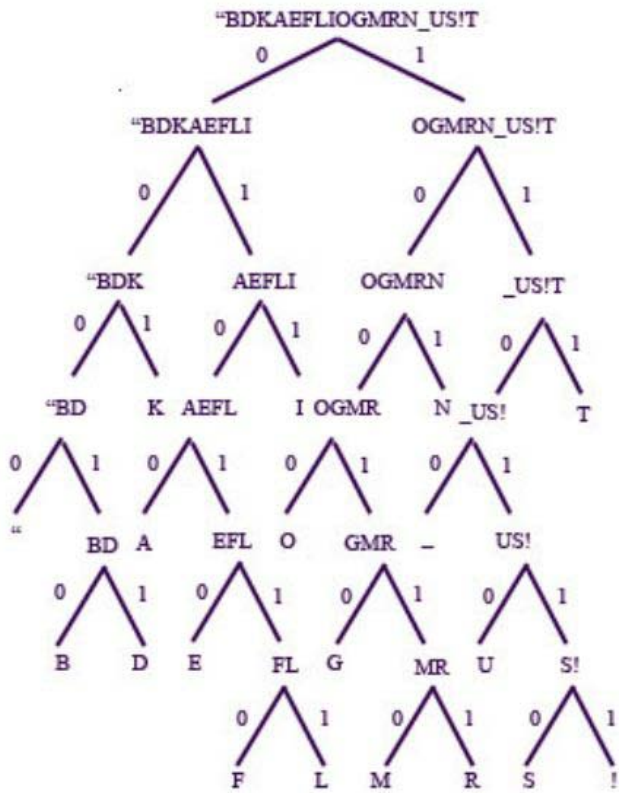


Lalu, kita beranjak dengan simbol yang memiliki kekerapan atau peluang lebih tinggi yakni 3/48. Dimana hanya terdapat dua buah simbol dengan kekerapan 3/48, yaitu ‘A’ dan ‘O’. Kali ini, saya akan meletakkan A dengan EFL dan O dengan GMR. Dalam kasus seperti ini, penempatannya bebas, bisa juga dengan “BD maupun US!. Simpul yang ditambahkan dan menjadi seperti ilustrasi di bawah ini.



Setelah itu kita masukan simbol dengan probabilitas lebih tinggi dari 3/48 dan paling rendah di antara sisa simbol yang ada. Lakukan hal yang sama sampai semua simbol telah masuk ke dalam pohon Huffman. Dalam contoh kasus ini, akan berhenti pada simbol dengan probabilitas yang paling tinggi, yakni ‘I’, ‘N’, atau ‘T’. Kemudian, beri angka 0 di *path* kiri dan 1 di *path* kanan. Angka inilah yang nantinya akan ditelusuri dari akar hingga daun sehingga membentuk kode Huffman untuk tiap symbol.

Pohon Huffman lengkap yang terbentuk adalah sebagai berikut.



Bentuk pohon yang dapat dibuat bisa bermacam-macam. Tidak hanya seperti contoh pohon di atas saja. Dari pohon

tersebut diperoleh Kode Huffman untuk tiap symbol dan sekaligus melengkapi tabel yang kita buat sebelumnya.

Simbol	Kekerapan	Peluang	Kode Huffman
A	3	3/48	0100
B	1	1/48	00010
D	1	1/48	00011
E	2	2/48	01010
F	1	1/48	010110
G	2	2/48	10010
I	6	6/48	011
K	4	3/48	001
L	1	1/48	010111
M	1	1/48	100110
N	6	6/48	101
O	3	3/48	1000
R	1	1/48	100111
S	1	1/48	110110
T	6	6/48	111
U	2	2/48	11010
“	2	2/48	0000
_	4	4/48	1100
!	1	1/48	110111

Biasanya satu simbol dalam teks direpresentasikan dengan 8 bit, sehingga 48 karakter x 8 = 384 bit (1 Byte=8 bit). Sedangkan setelah menggunakan algoritman Huffman, jumlah bit yang merepresentasikan string seluruhnya dapat dihitung dengan menjumlahkan kekerapan tiap simbol dikali dengan jumlah bit dari kode Huffmannya.

A = 3 x 4 = 12
B = 1 x 5 = 5
D = 1 x 5 = 5
E = 2 X 5 = 10
F = 1 x 6 = 6
G = 2 x 5 = 10
I = 6 x 3 = 18
K = 4 x 3 = 12
L = 1 x 6 = 6
M = 1 x 6 = 6
N = 6 x 3 = 18
O = 3 x 4 = 12
R = 1 x 6 = 6
S = 1 x 6 = 6
T = 6 x 3 = 18
U = 2 x 5 = 10
“ = 2 x 4 = 8
_ = 4 x 4 = 16
! = 1 x 6 = 6

Maka jumlah bit yang diperlukan setelah melalui kompresi dengan algortima Huffman adalah 12+5+5+10+6+10+18+12+6+6+18+12+6+6+18+10+8+16+6=190 bit.

V. KESIMPULAN

Kebutuhan akan data digital dengan memori yang lebih kecil namun tetap memiliki kualitas tinggi sehingga tidak memakan ruang penyimpanan data serta untuk menambah kecepatan saat transmisi data semakin tinggi. Oleh karena itu, dibutuhkan suatu perangkat lunak yang berfungsi untuk melakukan kompresi atau pemampatan data. Perangkat lunak tersebut dapat menggunakan berbagai macam algortima kompresi, salah satunya adalah algoritma Huffman.

Algoritma Huffman dapat digunakan untuk melakukan kompresi data terutama data berbentuk teks seperti pada Microsof Word, *file* .txt, untuk email, dan sebagainya. Dengan meng*encode* teks menggunakan algortima Huffman, akan diperoleh kode Huffman yang dapat merepresentasikan simbol-simbol atau karakter yang ada dalam teks dengan jumlah bit yang lebih sedikit dari seharusnya namun dengan kualitas yang tetap sama. Untuk malekukan *decoding* ada syarat yang harus ada saat *encoding* yakni kode dari yang merepresentasikan salah satu symbol tidak boleh mengandung kode awalan dari kode yang merepresentasikan simbol lainnya agar tidak menimbulkan ambiguitas saat proses *decoding*.

Untuk kompleksitas algoritma Huffman dapat ditinjau dari keefektifitasan yang dimiliki dari prioritas antrian struktur data yaitu O(log n), dan sebuah pohon dengan n daun memiliki 2n-1 simpul, maka algoritma Huffman bila dinyatakan dalam notasi big-O memiliki orde O(n log n).

VI. UCAPAN TERIMA KASIH

Makalah ini dibuat untuk memenuhi tugas mata kuliah Struktur Diskrit. Penulis mengucapkan terima kasih kepada dosen Struktur Diskrit, yakni Ibu Harlili dan Bapak Rinaldi Munir yang senantiasa mengajar dan membimbing kami.

Ucapan terima kasih juga diberikan kepada semua pihak yang telah membantu dalam pembuatan makalah ini sehingga makalah ini dapat penulis selesaikan.

DAFTAR REFERENSI

Ir. Rinaldi Munir, M.T., *Diktat Kuliah IF2151 Matematika Diskrit*, Bandung, 2004, ch 8-9
http://id.wikipedia.org/wiki/Kompresi_data
Tanggal akses 8 Desember 2010
http://en.wikipedia.org/wiki/Huffman_coding
Tanggal akses 9 Desember 2010
http://www.total.or.id/info.php?kk=data
Tanggal akses 9 Desember 2010
http://www.total.or.id/info.php?kk=teks
Tanggal akses 9 Desember 2010
http://www.compressconsult.com/huffman
Tanggal akses 9 Desember 2010
http://id.wikipedia.org/wiki/Encoding
Tanggal akses 11 Desember 2010
http://id.wikipedia.org/wiki/Digital

Tanggal akses 11 Desember 2010
http://www.total.or.id/info.php?kk=digital%20data
Tanggal akses 11 Desember 2010
http://en.wikipedia.org/wiki/Decoding
Tanggal akses 11 Desember 2010

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 15 Desember 2010

Rachmawaty
13509071