Kode Huffman dan Penggunaannya dalam Kompresi SMS

A. Thoriq Abrowi Bastari (13508025)
Teknik Informatika Institut Teknologi Bandung
email: if18025@students.itb.ac.id

ABSTRAK

Dalam makalah ini, akan dibahas konsep serta penggunaan singkat kode Huffman dalam memecahkan permasalahan kompresi SMS (*Short Message Service*). Di zaman yang serba praktis ini efisiensi merupakan salah satu hal yang paling diutamakan. SMS yang sudah tidak bisa dipisahkan dari kehidupan manusia saat ini juga bisa ditingkatkan tingkat efisiensinya. Salah satunya adalah dengan penerapan kode/algoritma Huffman.

Algoritma Huffman merupakan algoritma yang sering digunakan untuk melakukan kompresi karena tingkat efisiensinya yang cukup tinggi. Algoritma ini berdasarkan pada kekerapan munculnya karakter-karakter yang dipakai untuk kemudian dibuat pohon binernya. Proses ini sendiri terdiri dari beberapa langkah. Yang pertama adalah langkah *encoding*, atau pembentukan pohon Huffman yang merupakan pohon biner, dilanjutkan dengan *decoding*.

Pada penerapannya dalam SMS, kode Huffman akan meminimalisasi jumlah bit yang digunakan dengan memanfaatkan statistik jumlah kemunculan karakter/simbol pada SMS. Dengan demikian ukuran file teks SMS yang akan dikirim menjadi lebih kecil dari ukuran normalnya.

Kata kunci: kode Huffman, SMS, kompresi, pohon Huffman.

1. PENDAHULUAN

Dengan perkembangan teknologi informasi yang sudah sangat pesat, interaksi (komunikasi) antar sesama manusia menjadi jauh lebih mudah. Dulu manusia perlu menghabiskan waktu berhari-hari bahkan berminggu-minggu hanya untuk menyampaikan informasi dari satu kota ke kota yang lain. Namun, sekarang tidak sampai sepersekian detik, informasi dengan bobot yang jauh lebih besar bisa disampaikan ke tempat yang berjarak berpuluh-puluh kilometer dari lokasi sang pemberi informasi.

Sejak sekitar sepuluh tahun yang lalu, saat telepon genggam mulai populer di masyarakat, informasi yang kita sampaikan bisa sangat cepat dan tepat ke siapa pun yang ingin dituju. Salah satu aplikasi yang sangat terkenal dan mudah untuk digunakan oleh siapa saja adalah *Short Message Service*, atau lebih dikenal dengan SMS.

Bahkan penggunaan SMS sudah melebihi jumlah penggunaan telepon karena SMS akan selalu mengusahakan agar bagaimanapun sampai ke tujuannya. Walaupun telepon sedang tidak diaktifkan, SMS tetap akan sampai saat telepon tersebut telah diaktifkan. Ini disebabkan karena pada saat tidak berhasil terkirim, SMS akan disimpan dalam SMSC (*Short Message Service Centre*) yang bertindak sebagai *server*. Dengan keuntungan ini saja banyak orang yang lebih memilih untuk mengirim SMS dibandingkan menelepon disamping keuntungan dalam segi biaya yang dikeluarkan.

Namun, SMS juga memiliki keterbatasan. Sebuah SMS maksimal terdiri dari 140 bytes, yang berarti sebuah SMS dapat memuat sejumlah 140 karakter 8-bit, 160 karakter 7-bit atau 70 karakter 16-bit untuk bahasa Jepang, Mandarin dan Korea yang memakai aksara Kanji/Hanja [3]. Dalam mengirimkan SMS seorang pengguna dapat mengirim pesan lebih dari 140 byte, tetapi untuk itu seorang pengguna harus membayar lebih, misalnya pesan yang akan dikirimkan memiliki ukuran 158 byte maka ia harus membayar untuk 2 SMS, jika 320 byte maka sama saja membayar untuk 3 SMS, dan seterusnya. Hal ini terjadi karena pesan yang dikirimkan terdiri lebih dari satu halaman sehingga proses pengiriman pesan akan dilakukan sebanyak jumlah halaman yang ada, jumlah halaman sesuai dengan isi SMS yang diketikkan.

Untuk mengatasi hal tersebut, dibuatlah aplikasi yang melakukan kompresi dari isi SMS tersebut. Kompresi ini dapat dilakukan lewat berbagai metode, tetapi salah satu cara yang cukup mudah adalah dengan kode Huffman. Kode Huffman memanfaatkan statistik untuk memberikan kode-kode tertentu pada setiap karakter sehingga karakter yang sering muncul akan memiliki kode yang sederhana, yang secara otomatis akan membuat ukuran dari SMS menjadi lebih kecil.

2. KODE HUFFMAN

2.1 Sejarah Singkat

Sesuai dengan namanya, kode Huffman ditemukan oleh David A. Huffman pada tahun 1951. Pada saat itu ia masih menjadi salah seorang mahasiswa di MIT. Ide tersebut ia dapatkan ketika ia mendapatkan tugas untuk ujian akhir dari dosennya. Huffman tidak begitu saja menemukan kode Huffman karena pada awalnya ia sama sekali tidak terpikirkan hal tersebut. Pada saat itu ia ditugasi membuat sebuah kode biner yang efisien.

Setelah lama berpikir akhirnya ia mendapatkan sebuah ide untuk membuat sebuah kode biner yang didasarkan pada kekerapan dan ia berhasil menunjukkan bahwa cara ini adalah yang paling efisien. Konsep ini sebenarnya telah tercetus sebelumnya oleh dosen dan juga profesornya di MIT yang bernama Robert M. Fano bersama dengan Claude Shannon, namun disempurnakan oleh Huffman karena ia membangun pohon biner (pohon Huffman) dari bawah ke atas, bukan dari atas ke bawah seperti pada kode/algoritma ciptaan profesornya.

2.2 Proses Pembentukan Kode Huffman

Seperti yang telah disebutkan pada Bab Pendahuluan, kode Huffman memanfaatkan statistik untuk memperoleh kode yang paling efisien. Dari sekumpulan karakter-karakter yang ada, dihitung jumlah kemunculannya, lalu pohon Huffman akan dibentuk berdasarkan data tersebut. Dengan begitu, karakter yang lebih sering muncul akan memiliki kode yang lebih singkat, dan sebaliknya karakter yang jarang muncul akan memiliki kode yang lebih rumit.

Kelebihan lain dari kode Huffman adalah sifatnya yang prefiks. Ini berarti setiap kode tidak mungkin menjadi prefiks (awalan) bagi kode yang lainnya. Ini akan memudahkan jalannya proses *decoding*.

Dilihat dari tipe peta kodenya, kode/algoritma Huffman termasuk dalam kategori algoritma dengan metode statik. Ini berarti kode Huffman menggunakan peta kode yang sama. Metode yang dipakai terdiri dari dua fase, yang pertama adalah fase untuk menghitung jumlah kemunculan (kekerapan) karakter-karakter di dalam *string*. Fase yang kedua adalah proses translasi *string* tersebut menjadi rangkaian bit yang berasal dari kode Huffman yang dibuat.

Langkah pertama dan terpenting yang dilakukan untuk memperoleh kode Huffman adalah membentuk pohon Huffman. Pohon Huffman merupakan sebuah pohon biner yang merepresentasikan karakter-karakter yang muncul berikut kekerapannya. Seluruh karakter yang ada akan berada pada simpul daun (simpul tak beranak). Semua simpul dalam (simpul yang memiliki anak) yang ada pada pohon Huffman merupakan gabungan dari dua atau lebih karakter. Pohon ini disusun sedemikian rupa sehingga orang tua dari simpul memiliki kekerapan yang lebih besar dari anaknya, atau lebih tepatnya memiliki kekerapan yang merupakan jumlah kekerapan dari anak kiri dan anak kanannya.

Algoritma pembentukan pohon Huffman adalah sebagai berikut:

1. Pilih dua simpul karakter/simbol dengan kekerapan/peluang yang paling kecil. Kombinasikan kedua simpul tadi menjadi sebuah simpul orang tua dengan kekerapan yang merupakan hasil penjumlahan dari kedua simpul anaknya.
2. Lalu pilih lagi simpul dengan kekerapan yang paling kecil, termasuk simpul orang tua yang baru dibuat. Gabungkan keduanya untuk membuat simpul orang tua (langkah yang sama seperti langkah 1).
3. Ulangi kedua langkah tersebut sampai semua simpul habis dan berada pada pohon Huffman.

Berikut ini akan ditunjukkan contoh penggunaan kode Huffman untuk melakukan kompresi pada *string*. Contoh kali ini akan menggunakan *string* "STRUKTURDISKRIT". Untuk mempermudah pekerjaan pertama akan dibuat tabel yang berisi karakter-karakter yang terdapat dalam *string* beserta kekerapannya secara terurut membesar. Didapatkan kekerapan dari karakter D, I, S, U, K, T, dan R secara berurutan adalah 1, 2, 2, 2, 2, 3, dan 3.

Tabel 1. Tabel karakter (subpohon) beserta kekerapannya secara terurut

Karakter (Subpohon)	Kekerapan
D	1
I	2
S	2
U	2
K	2
T	3
R	3

Dua karakter yang memiliki kekerapan paling kecil adalah D dan salah satu di antara I, S, U, dan K. Untuk kasus seperti ini dibebaskan untuk memilih yang mana saja. Misalnya dipilih karakter D dan I. Maka akan terbentuk simpul baru, yaitu simpul DI dengan kekerapan 3, yang didapat dari penjumlahan kekerapan D dengan kekerapan I. Setelah terbentuk simpul DI maka didapatkan tabel baru berikut:

Tabel 2. Tabel karakter (subpohon) beserta kekerapannya secara terurut

Karakter (Subpohon)	Kekerapan
S	2
U	2
K	2
(D, I)	3
T	3
R	3

Dari Tabel 2 didapatkan karakter-karakter dengan kekerapan paling rendah adalah S, U, dan K. Kita pilih S dan U untuk membentuk simpul baru SU. Dalam pembuatan pohon Huffman simpul baru ini dengan simpul U dan S akan membentuk suatu subpohon yang baru pula. Lalu akan terbentuk tabel yang baru.

Tabel 3. Tabel karakter (subpohon) beserta kekerapannya secara terurut

Karakter (Subpohon)	Kekerapan
K	2
(D, I)	3
T	3
R	3
(S, U)	4

Dari Tabel 3 terlihat bahwa simpul K dan DI memiliki kekerapan yang paling rendah sehingga akan membentuk simpul KDI seperti terlihat pada Tabel 4.

Tabel 4. Tabel karakter (subpohon) beserta kekerapannya secara terurut

Karakter (Subpohon)	Kekerapan
T	3
R	3
(S, U)	4
(K, (D, I))	5

Simpul T dan R memiliki kekerapan terendah dengan angka yang sama dan akan terbentuk simpul baru SU dengan kekerapan 4.

Tabel 5. Tabel karakter (subpohon) beserta kekerapannya secara terurut

Karakter (Subpohon)	Kekerapan
(S, U)	4
(K, (D, I))	5
(T, R)	6

Dua subpohon (SU dan KDI) akan membentuk satu subpohon yang sama, yang direpresentasikan dengan simpul SUKDI.

Tabel 6. Tabel karakter (subpohon) beserta kekerapannya secara terurut

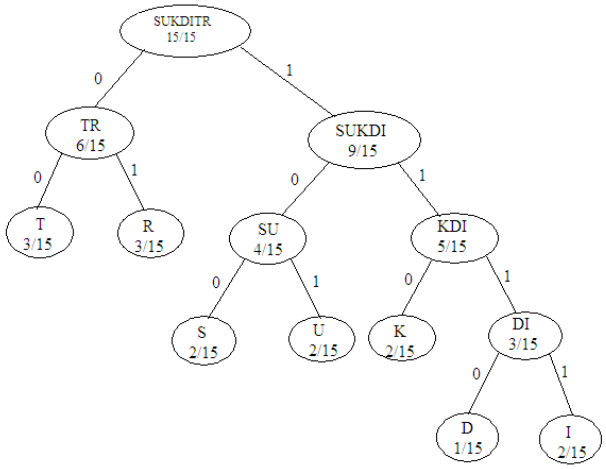
Karakter (Subpohon)	Kekerapan
(T, R)	6
((S, U), (K, (D, I)))	9

Hanya tersisa dua simpul lagi, yaitu simpul TR dan simpul SUKDI. Keduanya akan membentuk satu simpul lagi, yaitu simpul TRSUKDI. Simpul ini merupakan simpul dengan *aras* (tingkat) 0 pada pohon Huffman.

Tabel 7. Tabel karakter (subpohon) beserta kekerapannya secara terurut

Karakter (Subpohon)	Kekerapan
(T, R), ((S, U), (K, (D, I)))	15

Dari data-data yang didapat dari Tabel 1-7 dapat dibuat sebuah pohon Huffman sebagai berikut (angka yang berada di bawah nama simpul adalah peluang kemunculannya):



Gambar 1. Pohon Huffman untuk *string* "STRUKTURDISKRIT"

Setelah menggambarkan pohon Huffman maka kita dapat dengan mudah menentukan kode dari masing-masing karakter, atau kode Huffman dari *string* "STRUKTURDISKRIT". Ini dapat dilakukan dengan menelusuri pohon Huffman dari akar yang paling atas. Kode setiap karakter adalah lintasan yang dilalui dari akar sampai ke simpul daun karakter tersebut. Misalnya, jika kita ingin mendapatkan kode untuk karakter R maka telusuri dari simpul akar paling atas lalu ke simpul TR, dan sampai ke simpul R. Lintasan yang dilewatinya adalah 01, maka itulah kode Huffman yang didapatkan untuk karakter R. Kode Huffman untuk masing-masing karakter dapat dilihat secara langsung pada tabel di bawah ini:

Tabel 8. Tabel karakter beserta kode Huffmannya

Karakter	Kode Huffman
D	1110
I	1111
S	100
U	101
K	110
T	00
R	01

Setelah diperoleh kode Huffman untuk masing-masing karakter pada *string* "STRUKTURDISKRIT" maka kita bisa mendapatkan rangkaian bit-nya. Berdasarkan Tabel 8 rangkaian bit-nya adalah 100000110111000101011110111110011001111100, dengan jumlah total 42 bit. Ini jauh lebih efisien daripada *string* dengan kode ASCII (*American Standard Code for Information Interchange*). Karena setiap karakter memiliki 8 bit, maka untuk 15 karakter diperlukan 8 x 15 = 120 bit. Pada contoh ini, kode Huffman dapat menghemat sampai sekitar 70% dari total memori yang diperlukan. Ini menunjukkan bahwa metode ini sangat membantu dalam meningkatkan efisiensi memori.

Kompleksitas waktu dari algoritma Huffman juga dapat dicari. Dalam melakukan satu kali proses iterasi pada saat menggabungkan dua buah pohon dengan kekerapan terkecil membutuhkan waktu O(log n). Untuk membentuk pohon Huffman, proses tersebut akan diulang sebanyak n kali. Sehingga, kompleksitas waktu algoritma Huffman adalah O(n log n).

3. PENGGUNAAN KODE HUFFMAN DALAM APLIKASI KOMPRESI SMS

Dalam melakukan kompresi SMS ada banyak teknik yang digunakan, namun cara yang umum digunakan adalah dengan membuat pohon Huffman yang tetap. Langkah awalnya adalah menyusun tabel kode Huffman yang baru untuk semua karakter yang akan digunakan. Masing-masing karakter terdiri atas prefiks dan *body*. Prefiks menunjukkan jenis karakter misalnya huruf besar atau kecil, dan *body* berisi bit datanya. Dalam tabel tersebut (Tabel 9), hanya sebagian karakter yang ditampilkan.

Seperti yang telah dibahas pada contoh pada bab yang sebelumnya, karakter-karakter yang memiliki kekerapan tinggi akan terletak semakin dekat dengan akar paling atas pada pohon Huffman. Ini berarti kode Huffman yang dimilikinya juga lebih singkat. Pada tabel di bawah kode Huffman untuk huruf kecil akan memiliki prefiks 0. Selanjutnya adalah huruf besar yang memiliki kekerapan lebih kecil dibanding huruf kecil. Huruf besar memiliki prefiks 100. Setelah itu adalah angka dan beberapa jenis simbol dengan prefiks 111. Lalu tanda baca yang sering digunakan akan memiliki kode awal (prefiks) 110. Pengecualian adalah untuk karakter 'e' yang memiliki prefiks 1011. Karakter yang lain mendapat prefiks 1111110. Ini adalah karakter-karakter yang sangat jarang digunakan. Berikut adalah tabel yang menunjukkan kode untuk setiap karakter:

Tabel 9. Tabel karakter beserta kode Huffmannya dalam berbagai aplikasi kompresi SMS

Karakter	**Prefiks**	***Body***	**Hasil**
a	0	0000	00000
b		00101	000101
c		1101	01101
d		01011	001011
f		11001	011001
g		10101	010101
h		1000	01000
i		0001	00001
j		110001001	0110001001
k		1100011	01100011
l		1001	01001
m		10100	010100
n		0110	00110
o		0100	00100
A	100	0000	1000000
B		00101	10000101
C		1101	1001101
D		01011	10001011
F		11001	10011001
G		10101	10010101
H		1000	1001000
I		0001	1000001
J		110001001	100110001001

Lanjutan dari Tabel 9.

spasi	101	1	1011
e		0	1010
!	110	0011	1100011
“		0110	1100110
‘		0100	1100100
0	111	0000	1110000
1		0001	1110001
3		0011	1110011
4		0100	1110100
sisanya		1111110	1111110

Dengan menggunakan tabel di atas, sebuah *file* SMS biasa bisa memiliki ukuran yang lebih kecil dari biasanya. Untuk membuktikan hal ini, akan dicoba perbandingan dengan menggunakan kode ASCII yang biasa digunakan dalam SMS, yaitu ASCII 7 bit. Berikut adalah tabel yang menunjukkan contoh beberapa kode ASCII yang dipakai pada SMS:

Tabel 10. Tabel karakter beserta kode ASCII yang dipakai pada SMS

Kode ASCII	Karakter
0110000	0
0110001	1
0110010	2
1000000	@
0111111	?
1000001	A
1000010	B
1000011	C
1100001	A
1100010	B

Sebagai contoh, akan dicoba perbandingan antara kode ASCII dan kode Huffman. Untuk sebuah SMS yang berisi rangkaian karakter “oi km d mana”, dengan kode ASCII yang biasa dipakai, SMS akan berukuran 7 x 12 = 84 bit. Sementara itu, jika memakai kode Huffman seperti yang ditunjukkan pada Tabel 9, ukurannya adalah 5 + 5 + 4 + 8 + 6 + 4 + 6 + 4 + 6 + 5 = 53 bit (angka-angka tersebut didapatkan dari rangkaian bit untuk masing-masing karakter). Untuk kasus ini penggunaan kode Huffman dapat menghemat sampai sekitar 37% memori. Dengan pemanfaatan aplikasi seperti ini pengguna dapat menambahkan informasi yang ingin disampaikan dalam satu SMS sekaligus mempercepat proses transfer data karena ukurannya yang lebih kecil.

4. KESIMPULAN

Kode Huffman merupakan suatu cara pengodean yang sangat efisien. Kode ini didasarkan pada statistik kekerapan munculnya setiap karakter yang ada sehingga terbentuk serangkaian kode yang dapat merepresentasikan kumpulan karakter-karakter tersebut dengan sesedikit mungkin. Keuntungan lain dari kode Huffman adalah sifatnya yang prefiks, berarti setiap kode tidak mungkin menjadi prefiks (awalan) bagi kode yang lainnya. Ini akan memudahkan jalannya proses *decoding*.

Dengan banyak kelebihannya, kode Huffman sering dipakai dalam metode kompresi data. Salah satunya adalah kompresi SMS. Setiap karakter akan memiliki kode yang terdiri dari prefiks dan *body*, keduanya merepresentasikan kekerapan masing-masing sehingga karakter dengan kekerapan yang tinggi akan memiliki kode yang lebih singkat dan karakter dengan kekerapan yang rendah, seperti simbol-simbol yang jarang digunakan, akan memiliki kode yang lebih panjang. Dengan begitu, akan terbentuk kode yang jauh lebih efisien.

REFERENSI

[1] Munir, Rinaldi. Diktat Kuliah IF2153 Matematika Diskrit. Program Studi Teknik Informatika, Institut Teknologi Bandung. 2006.

[2] Wikipedia, The Free Encyclopedia, *Huffman Coding.* 2007. *http://en.wikipedia.org/wiki/Huffman_coding*. Waktu akses : Jumat, 18 Desember 2009 pukul 20.30 WIB.

[3] Purwanto, Heri. *Aplikasi Kompresi SMS Teks (Short Message Service) dengan Menggunakan Algoritma Huffman Kanonik dan LZW (Lempel-Ziv-Welch).* 2006, 1.