PEMAMPATAN DATA *(DATA COMPRESSION)* DENGAN KODE HUFFMAN DALAM KOMUNIKASI DATA

Nur Hadi Waryanto & Wahyu Setyaningrum
Jurusan Pendidikan Matematika FMIPA UNY

ABSTRAK

Masalah yang sering muncul dalam komunikasi data adalah ukuran pesan *(message)* yang dikirim terlalu besar sehingga waktu pemgiriman lama dan membutuhkan ruang penyimpanan yang besar pula.

Kedua masalah di atas dapat diatasi dengan memampatkan data atau pesan yang dikirim. Pemampatan data dilakukan dengan mengkodekan setiap karakter dalam pesan tersebut dengan kode yang lebih pendek. Sistem kode yang digunakan adalah sistem kode ASCII kemudian dilakukan pemampatan data *(data compression).* Dalam makalah ini akan dibahas pemampatan data *(data compression)* menggunakan kode Huffman.

Pemampatan data *(data compression)* dengan kode Huffman ternyata dapat digunakan untuk mempersingkat pesan yang dikodekan dengan sistem ASCII, sehingga pesan yang dikirimkan relatif singkat atau pendek dan ruang penyimpanan relatif kecil pula.

Kata kunci : kode Huffman, pemampatan data

A. Pendahuluan

Pesan agar sampai ke alamat tujuan, pengiriman pesan tersebut memanfaatkan komunikasi data. Dalam komunikasi data sering timbul beberapa masalah yaitu pesan yang dikirimkan ukurannya besar dan waktunya lama. Selain itu dalam penyimpanan data, pesan yang berukuran besar akan membutuhkan ruang penyimpanan yang besar pula. Permasalahan ini dapat diatasi dengan mengkodekan pesan tersebut sesingkat mungkin. Pesan yang singkat menyebabkan ukuran pesan menjadi kecil dan membutuhkan ruang penyimpanan yang kecil sehingga pesan dapat dikirimkan dengan cepat. Cara pengkodean inilah yang disebut dengan pemampatan data. Setiap karakter dalam pesan dikodekan dengan kode yang lebih pendek dalam komunikasi data. Kode ASCII merupakan sistem kode yang banyak digunakan saat ini.

Setiap karakter dalam kode ASCII dikodekan dalam 8 bit biner. Contoh beberapa kode ASCII terlihat dalam Tabel 1.

Karakter	Kode ASCII	Karakter	Kode ASCII	Karakter	Kode ASCII
A	01000001	N	01001110	U	01010101
B	01000010	O	01001111	a	01100001
C	01000011	P	01010000	i	01101010
D	01000100	R	01010010	u	01100101
E	01000101	S	01010011	e	01100110
I	01001001	“	00100010	o	01110000
M	01001101	spasi	00100000	L	01001100

Tabel 1 : Tabel Kode ASCII

Jika suatu komunikasi data mengunakan kode ASCII maka sebuah pesan misalnya “SEMINAR NASIONAL MIPA” akan direpresentasikan menjadi rangkaian bit sebagai berikut :

00100010010100110100010101001101010010010100111001000001010100100010000001001110010000010101001101001001010011110100111001000001010011000010000001001110010000010101001101001001010011110100111001000001010011000010000001001101010010010101000001000001001000100

Secara langsung dapat dilihat bahwa sebuah pesan yang hanya terdiri tiga kata jika direpresentasikan dengan kode ASCII tanpa pemampatan data terlihat begitu panjang. Dapat dibayangkan jika pesan tersebut merupakan rangkaian kalimat dalam satu paragraf. Jika dihitung banyaknya bit yang dibutuhkan untuk pesan tersebut adalah 23 x 8 bit = 184 bit. Jumlah bit yang dibutuhkan untuk pesan tersebut dapat diminimumkan dengan memperpendek panjang kode setiap karakter dalam pesan tersebut terutama karakter yang mempunyai frekuensi kemunculannya besar. Hal inilah yang mendasari munculnya kode Huffman. Dengan menggunakan kode Huffman, jumlah bit yang dibutuhkan semakin sedikit yang menyebabkan ukuran pesan menjadi kecil dan ruang penyimpanannyapun kecil sehingga peasn dapat dikirimkan dengan cepat.

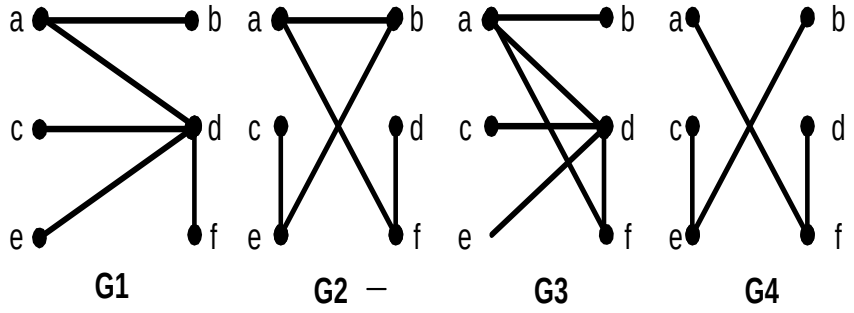
B. PEMBAHASAN

POHON BINER

Definisi 1:

Pohon adalah graf tak-terarah terhubung yang tidak mengandung sirkuit.

Menurut definisi 1 di atas, graf G_1 dan G_2 pada gambar 1 adalah pohon, sedangkan G_3 dan G_4 bukan pohon. G_3 mengandung sirkuit a, d, f, a. Sedangkan G_4 tidak terhubung.

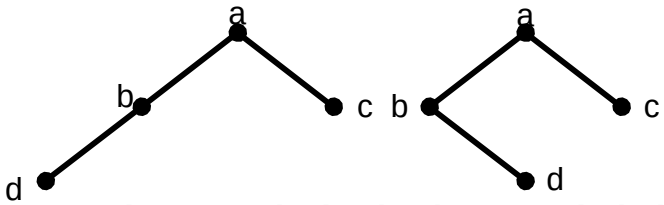


Gambar 1: Pohon dan bukan pohon

Definisi 2:

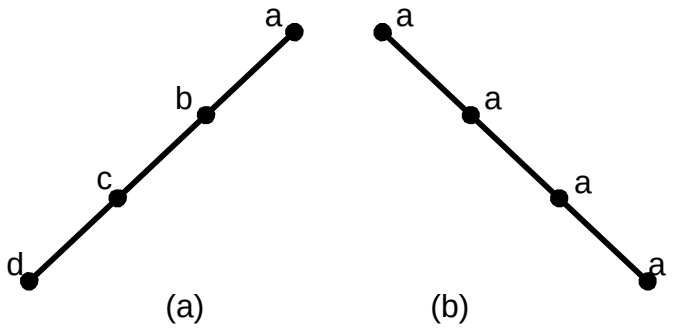
Pohon biner adalah pohon yang setiap simpul cabangnya mempunyai maksimum dua buah anak.

Ada yang menyebut dua anak tersebut dengan anak pertama dan anak kedua atau anak kanan dan anak kiri. Selanjutya, dalam tulisan kami menggunakan istilah anak kiri dan anak kanan. Pohon yang akarnya adalah anak pohon kiri disebut upapohon kiri (*left subtree*), sedangkan pohon yang akarnya adalah anak pohon kanan disebut upapohon kanan (*right subtree*). Karena adanya perbedaan anak/upapohon kiri dan anak/upapohon kanan, maka pohon biner adalah pohon terurut. Dua buah pohon pada Gambar 2 adalah dua buah pohon biner berbeda.



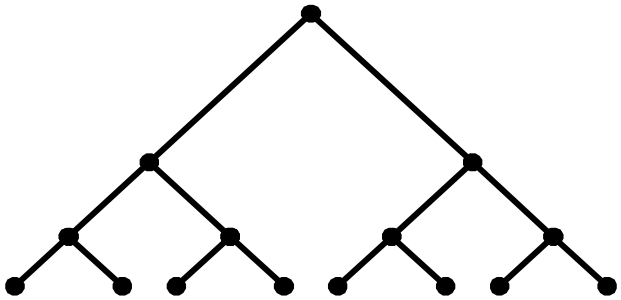
Gambar 2 : Dua buah pohon biner yang berbeda

Pohon yang semua simpulnya terletak di bagian kiri saja atau di bagian kanan saja disebut pohon condong (*skewed tree*). Pohon yang condong ke kiri disebut pohon condong-kiri (*skew left*), pohon yang condonmg ke kanan disebut pohon condong-kanan (*skew right*). Lihat Gambar 3.



Gambar 3 : (a) pohon condong-kiri, dan (b) pohon condong-kanan

Pohon biner penuh (*full binary tree*) adalah pohon yang setiap simpulnya mempunyai tepat dua buah anak, kiri dan kanan, kecuali simpul pada aras bawah (Gambar 4).


Gambar 4 : Pohon biner penuh

Pohon biner penuh dengan tinggi h memiliki jumlah daun sebanyak 2^n, sedangkan jumlah seluruh simpulnya adalah:

$S = 2^0 + 2^1 + 2^2 + \ldots + 2^h = 2^{h+1}-1$

ALGORITMA HAUFFMAN

Algoritma ini digunakan untuk menentukan sebuah pohon biner n-terboboti (P) yang mempunyai panjang lintasan minimum dengan bobot w_1, w_2, w_3, …, w_n. Untuk menunjukkan bagaimana algoritma Huffman bekerja, pertama kita buktikan lemma yang mengungkapkan bahwa dalam pohon biner optimal, daun dengan bobot terbesar terletak di dekat akar. Lemma dan akibatnya berlaku jika semua simpul mempunyai bobot berbeda. Bagaimana jika tidak semua simpul mempunyai bobot berbeda atau dengan kata lain ada simpul yang memiliki bobot sama. Dengan menggunakan algoritma Huffman kita dapat menyelesaikan kasus tersebut.

Lemma:

P pohon biner optimal dengan bobot w_1, w_2, …, w_n. untuk i = 1, 2, …, t dan L_i merupakan level atau tingkatan dari w_i dalam pohon P. Jika $w_i < w_k$ maka $L_j \geq L_k$.

Bukti:

Andaikan $w_i < w_k$ dan $L_j < L_k$ dan P pohon biner optimal. P merupakan pohon yang diperoleh dengan menukar tempat w_i dan w_k pada pohon P.

Maka bobot total P

$W(P) = w_j L_j + w_k L_k$

Sedangkan bobot total P'

$W(P') = w_j L_k + w_k L_j$

$W(P) - W(P') = w_j L_j + w_k L_k - w_j L_k - w_k L_j$

$= (w_k - w_j)(L_k - L_j) > 0$

sehingga $W(P') < W(P)$ dan P bukan pohon biner optimal, kontradiksi dengan pengandaian. Jadi P pohon biner optimal, jika $w_i < w_k$ maka $L_j \geq L_k$.

Akibat *(corollary)*:

Pada pohon biner optimal P, apabila dua bobot terkecil w_1 dan w_2 maka keduanya terletak pada level (tingkatan) terendah.

Bukti:

Setidaknya ada 2 daun yang terletak pada tingkat/level terendah, misal w_j dan w_k. Jika $w_1 < w_j$ maka $L_1 \geq L_j = L$ berdasarkan lemma maka $L_1 = L$ dan w_1 pada level L. Jika $w_1 = w_j$ maka $L_1 = L_j$, kita dapat menukar tempat w_1 dan w_j tanpa mengubah bobottotal dari P. Begitu pula, dengan menukar tempat w_2 dan w_k, akibatnya w_2 terletak pada level L. Sehingga w_1 dan w_2 terletak pada level L.

Teorema:

Misal $0 \leq w_1 \leq w_2 \leq \ldots \leq w_t$, p' adalah pohon biner optimal dengan bobot w_1, w_2, …, w_t. Akan ditunjukkan bahwa $W(P) = W(P_0)$. Dengan menggunakan akibat dari lemma (*corollary*) dapat dianggap bahwa w_1 dan w_2 pada P_0 terletak pada level/tingkat yang sama. Bobot total dari P_0 tidak akan berubah jika bobot pada level yang sama saling ditukar tempat. Diasumsikan bahwa w_1 dan w_2 adalah anak dari orangtua yang sama b. sehingga tiga simpul tersebut membentuk sub pohon kecil, misal P_b dengan b sebagai akar.

Misal P_0' merupakan pohon dengan bobot w_1, w_2, …, w_t yang didapat dari P_0 dengan mengganti daun b' pada subpohon P_b yang berbobot $w_1 + w_2$. Jika L merupakan level dari simpul b maka dalam perhitungan $W(P_0)$ sub pohon P_b memberikan kontribusi $w_1(L + 1) + w_2(L+1)$. Sedangkan pada perhitungan $W(P_0')$ simpul b' dengan berat $w_1 + w_2$ memberikan kontribusi $(w_1 + w_2)$ L. sehingga

$W(P_0) = W(P_0') + w_1 + w_2$

Karena P' merupakan pohon optimal dengan bobot $w_1 + w_2$, w_3, …, w_t, maka

$W(P') \leq W(P_0')$ sehingga

$W(P) = W(P') + w_1 + w_2 \leq W(P_0') + w_1 + w_2 = W(P_0)$

Didapat $W(P_0) \leq W(P)$, karena P_0 merupakan pohon optimal dengan bobot w_1, w_2, …, w_t maka $W(P) = W(P_0)$

Akibatnya P merupakan pohon biner optimal dengan bobot w_1, w_2, …, w_t.

Berikut ini diberikan salah satu contoh pemampatan data dengan kode Huffman. Misalkan terdapat suatu pesan seperti pada bagian pendahuluan yaitu "SEMINAR NASIONAL MIPA". Jumlah karakter dalam pesan itu adalah 23 termasuk tanda petik dan spasi. Pada bagian pendahuluan pesan tersebut telah direpresentasikan dalam kode ASCII sebelum dipampatkan. Jumlah bit dari pesan tersebut adalah 184 bit. Langkah pertama untuk memampatkan data tersebut adalah menentukan bobot atau frekuensi dari setiap karakter dalam pesan tersebut, seperti terlihat pada Tabel.2.

Karakter	**Bobot(w)/frekuensi**
E	1
R	1
O	1
L	1
P	1
M	2
"	2
S	2
Spasi	2
I	3

N	3
A	4
Jumlah	23

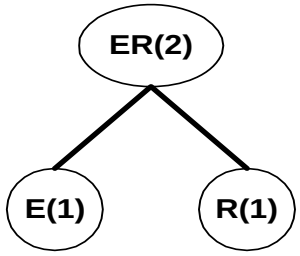
Tabel.2

Setelah bobot masing-masing karakter ditentukan langkah selanjutnya adalah membuat barisan dari masing-masing karakter disertai bobotnya untuk memudahkan dalam pemilihan karakter-karakter yang mempunyai bobot paling kecil. Barisan tersebut adalah sebagai berikut:

E(1), R(1), O(1), L(1), P(1), M(2), “(2), S(2), Spasi(2), I(3), N(3), A(4)

Langkah selanjutnya memilih dua karakter yang mempunyai bobot paling kecil. Jika terdapat lebih dari dua karakter yang mempunyai bobot terkecil pilih secara sembarang, dalam hal ini dipilih karakter E dan R kemudian gabung dan tambahkan bobot masing-masing karakter. Langkah berikutnya adalah membuat barisan dari masing-masing karakter tanpa karakter yang telah dipilih seperti terlihat seperti dibawah ini :

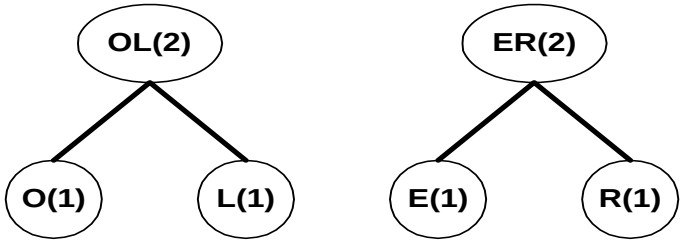
ER(2), O(1), L(1), P(1), M(2), “(2), S(2), Spasi(2), I(3), N(3), A(4)



Gambar 5. Pohon biner I

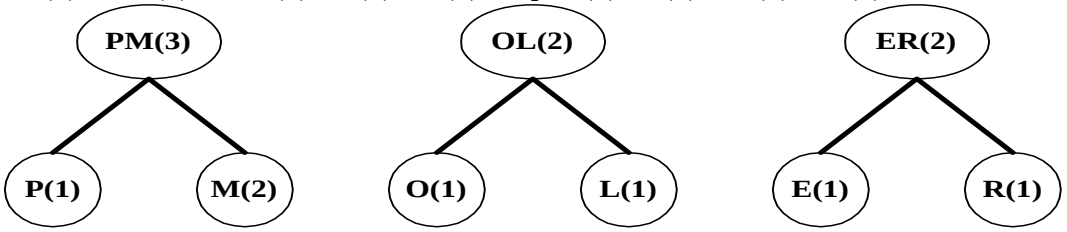
Pohon biner dapat dibuat setelah semua proses pemilihan masing-masing karakter selesai. Pohon biner dalam halini dibuat terlebih dahulu dengan maksud untuk mempermudah pemahaman. Proses-proses pemilihan masing-masing karakter selanjutnya adalah sebagai berikut :

ER(2), **OL(2)**, P(1), M(2), “(2), S(2), Spasi(2), I(3), N(3), A(4)



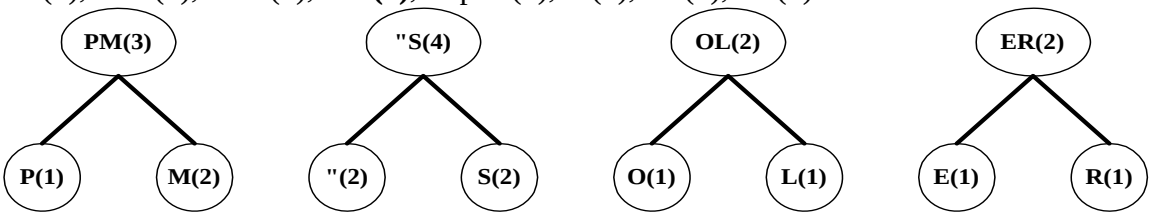
Gambar 6. Pohon biner II

ER(2), OL(2), **PM(3)**, “(2), S(2), Spasi(2), I(3), N(3), A(4)



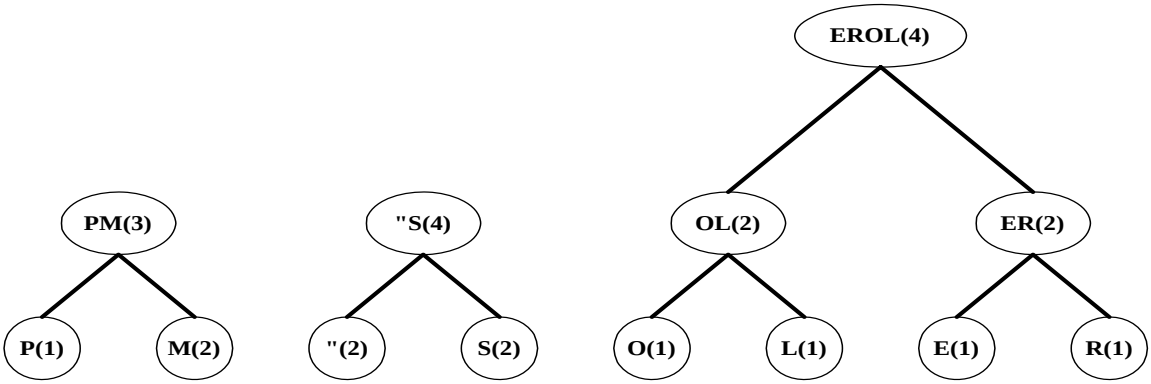
Gambar 7. Pohon biner III

ER(2), OL(2), PM(3), **“S(4)**, Spasi(2), I(3), N(3), A(4)



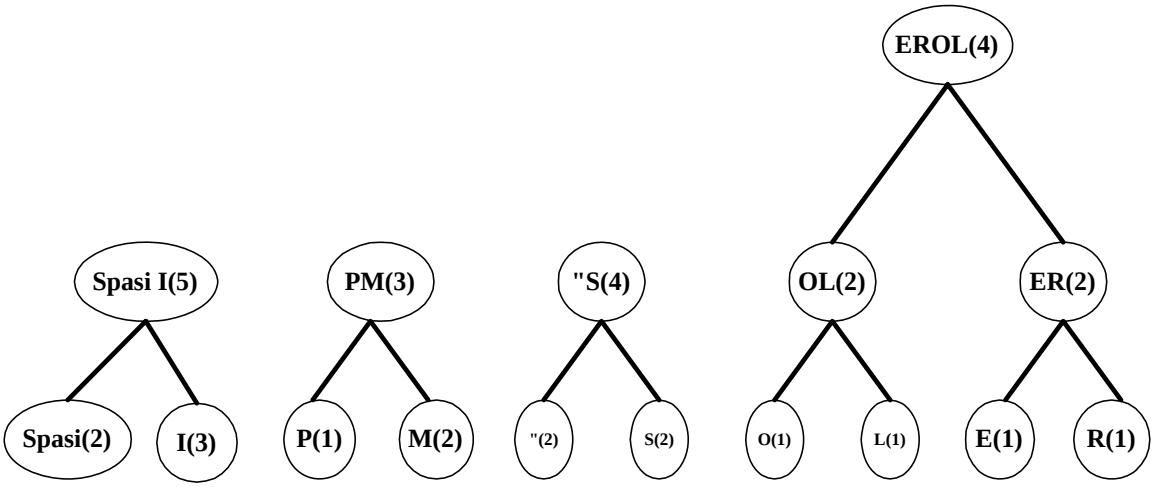
Gambar 8. Pohon biner IV

EROL(4), PM(3), “S(4), Spasi(2), I(3), N(3), A(4)



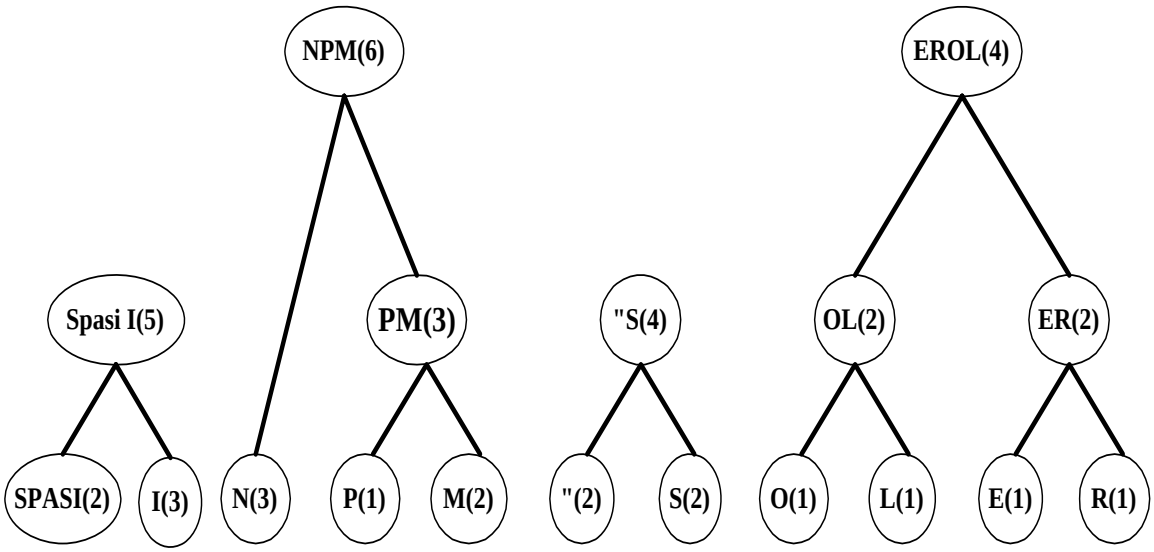
Gambar 9. Pohon biner V

EROL(4), PM(3), “S(4), **SpasiI(5)**, N(3), A(4)



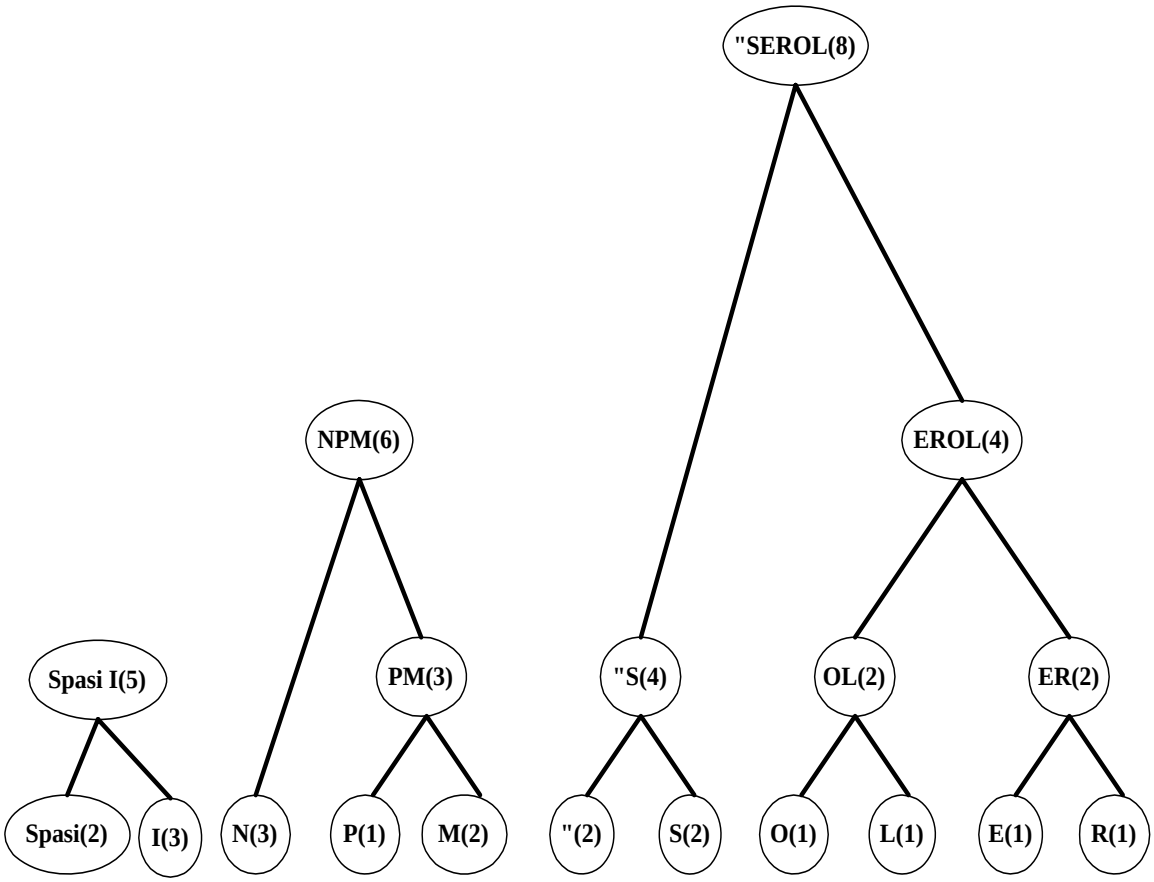
Gambar 10. Pohon biner VI

EROL(4), **NPM(6)**, “S(4), SpasiI(5), A(4)



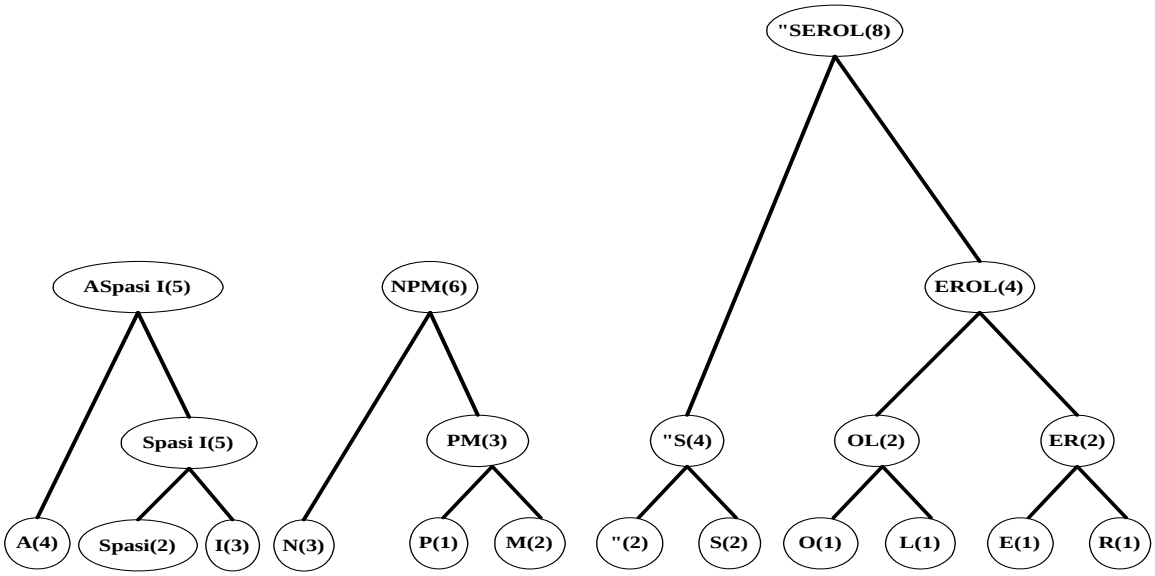
Gambar 11. Pohon biner VII

“SEROL(8), PMN(6), SpasiI(5), A(4)



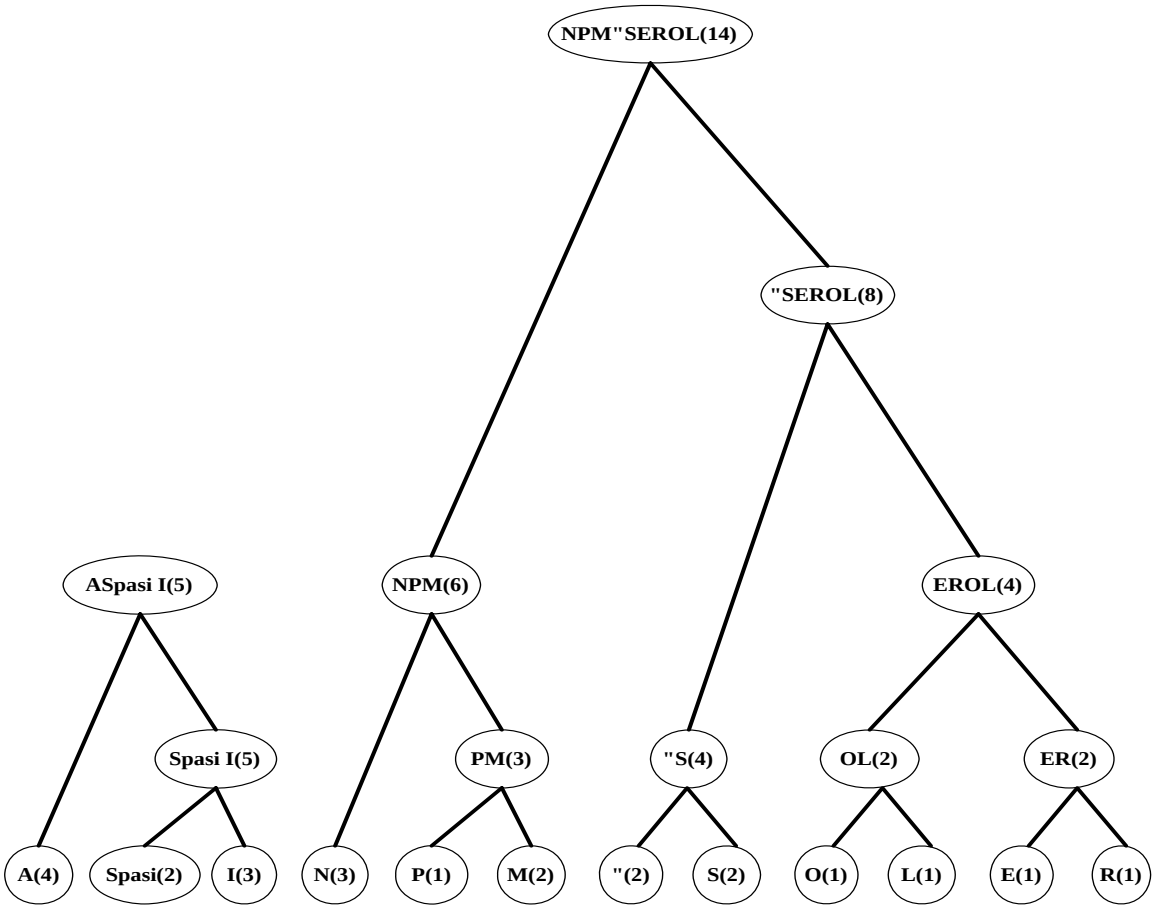
Gambar 12. Pohon biner VIII

“SEROL(8), PMN(6), **ASpasiI I(9)**



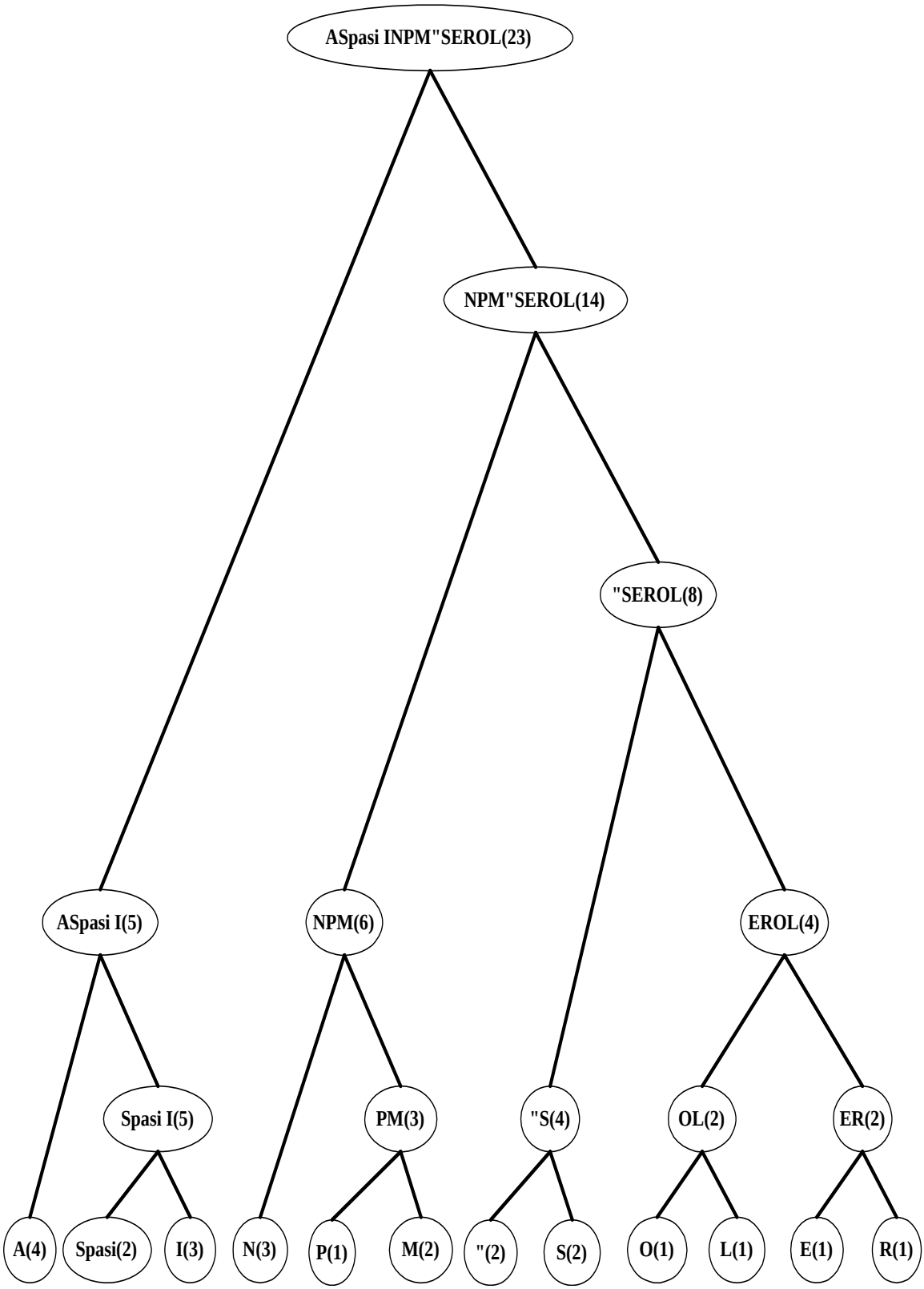
Gambar 13. Pohon biner IX

NPM“SEROL(14), SpasiIA(9)



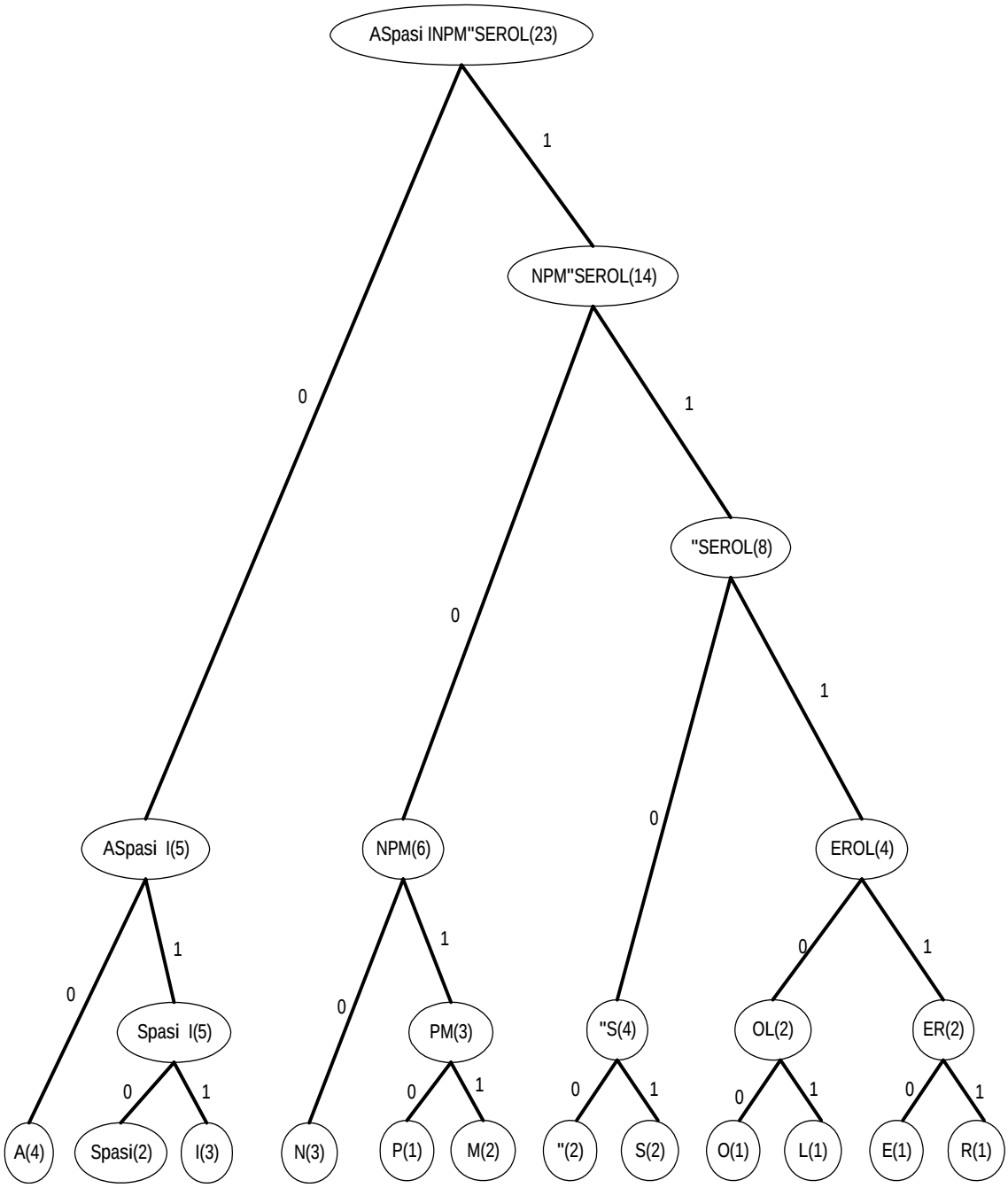
Gambar 14. Pohon biner X

“SEROLPMNSpasiIA(23)



Gambar 15. Pohon biner XI

Setelah prosedur pemilihan masing-masing karakter selesai, pohon biner terakhir diberikan kode untuk setiap karakter dengan memberi label 0 untuk untuk setiap cabang (sisi) kiri dan label 1 untuk setiap cabang (sisi) kanan. Sehingga didapatkan pohon Huffman seperti gambar 16.



Gambar 16. Pohon Huffman untuk Tabel 2

Dengan membuat lintasan dari akar ke daun pada Gambar 16, akan dihasilkan kode untuk masing-masing karakter seperti pada Tabel 3.

Karakter	Kode Huffnan
E	11110
R	11111
O	11100
L	11101
P	1010
M	1011
“	1100
S	1101
SPASI	010
I	011
N	100
A	00

Tabel.3

Berdasarkan Tabel.3, karakter yang mempunyai bobot paling besar mempunyai kode dengan jumlah bit minimum dan jumlah seluruh bit dari pesan di atas adalah 80 bit. Terlihat bahwa dengan menggunakan kode Huffman pesan diatas dapat menghemat ruang penyimpanan sebanyak 104 bit.

KESIMPULAN

Kode Huffman dapat digunakan untuk memampatkan data (data compression) dalam komunikasi data. Pemampatan data (data compression) dengan kode Huffman dapat digunakan untuk mempersingkat pesan yang dikodekan dengan sistem ASCII, sehingga pesan yang dikirimkan relatif singkat atau pendek dan ruang penyimpanan

relatif kecil pula. Langkah-langkah pemampatan data dengan algoritma Huffman adalah sebagai berikut:

1. Menentukan bobot atau frekuensi dari setiap karakter dalam pesan.
2. Membuat barisan dari masing-masing karakter disertai bobotnya.
3. Memilih dua karakter yang mempunyai bobot terkecil, kemudian digabungkan dan bobot masing-masing karakter dijumlahkan. Gabungan dua karakter ini diperlakukan sebagai karakter baru.
4. Selanjutnya, pilih dua karakter berikutnya, termasuk karakter baru, yang mempunyai bobot terkecil.
5. Prosedur yang sama dilakukan pada 2 karakter berikutnya yang mempunyai bobot terkecil.
6. Setelah prosedur pemilihan karakter selesai, kemudian pohon biner yang terbentuk diberi label 0 untuk setiap cabang kiri dan 1 untuk setiap cabang kanan.
7. Kode Huffman untuk setiap karakter diperoleh dengan membaca label pada pohon biner dari akar ke daun.

Daftar Pustaka

Lipschutz, Seymour. (1997). *Schaum's Outline of Theory and Problems of Discrete Mathematics.* New York: Mc Graw-Hill. Inc.

Munir, Rinaldi. (2001). *Matematika Diskret.* Bandung: Informatika Bandung.

Ross, Kenneth A. (1992). *Discrete Mathematics*. New Jersey: Prentice-Hall.

Truss, J. K. (1994). *Discrete Mathematics for Computer Scientists*. London: Addison-Wesley Publisihng Company.