Penggunaan Kode Huffman dan Kode Aritmatik pada *Entropy Coding*

Wisnu Adityo NIM:13506029
Program Studi Teknik Informatika ITB, Jalan Ganesha no 10 Bandung,
email : raydex@students.itb.ac.id

Abstrak *– Pada jaman yang serba modern ini, kompresi data adalah suatu hal yang esensial. Teknik kompresi ini esensial karena karena ukuran dari data semakin lama semakin besar, tetapi tidak didukung oleh perkembangan dari teknologi penyimpanan data dan bandwidth (untuk kecepatan download data dari internet) yang seimbang. Sementara orang-orang pun menginginkan data dengan kualitas terbaik dengan kuantitas (ukuran) yang minimum.*

Melihat masalah-masalah tadi, maka pemecahannya adalah maksimalisasi kompresi – yaitu mengurangi tempat yang digunakan oleh data yang dikompresi – yang diwujudkan oleh entropy coding. Dalam makalah ini, akan dibahas dua kode yang sering dipakai dalam Entropy coding, yaitu kode Huffman dan kode Aritmatik beserta perbandingan kedua pada pemampatan data.

Kata Kunci*: entropy, kode Huffman, kode Aritmatik, efektifitas kompresi.*

1. PENDAHULUAN

1.1. Introduksi pada kompresi data

Sebagaimana dijelaskan pada abstrak, kompresi data adalah hal yang esensial pada zaman modern ini. Hal yang penting dalam kompresi data adalah "rasio kompresi", atau rasio dari file yang dikompresi dengan file tersebut sebelum dimkompresi. Misalnya, anggap sebuah data berukuran 100 kilobytes (KB). Dengan kompresi data dengan suatu software kompresi data, file tersebut katakanlah dapat direduksi menjadi 50 KB. Oleh karena itu, file tersebut lebih mudah untuk disimpan media penyimpan dan lebih mudah ditransmisikan melalui koneksi internet. Pada kasus spesifik ini, software kompresi tersebut dapat mereduksi ukuran data ini dengan factor pembagi 2, atau mereduksi data ini dengan "rasio kompresi" 2:1.

Data kompresi terbagi menjadi kompresi data secara "lossless" dan "lossy". Kompresi data secara "lossy" adalah kompresi data yang hasil kompresinya, apabila didekompres, akan ada sebagian data yang hilang. Dengan kata lain, kompresi data secara "lossy" tidak dapat mendekompresi data seperti semula, sementara kompresi data secara "loseless" adalah kebalikannya. Kompresi data secara "lossless" akan lebih ditekankan dalam makalah ini.

Kriteria pengkompresian :

- Kualitas data hasil encoding

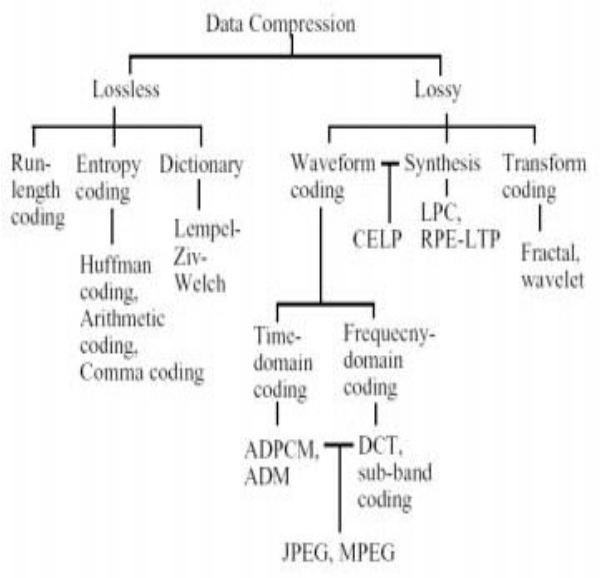
Ukuran hasil kompresi, tidak rusaknya data (lossy)

- Ketepatan proses dekompresi data

Data hasil dekompresi dan sebelum kompresi tetap sama (loseless)

- Kecepatan, rasio dan efisiensi proses kompresi dan dekompresi

Kompresi data secara "lossless" ini digunakan ketika data yang didekompresi harus sama seperti aslinya (sebelum dikompresi). Data-data berbentuk tulisan (text files) misalnya, harus dikompresi menggunakan kompresi data secara "lossless", karena kehilangan sebuah karakter saja dapat berakibat pada kesalahpahaman pada kasus terburuk. Penyimpanan "master source" (sumber yang pasti/terpercaya) dari data gambar, video ataupun suara biasanya pun dikompresi secara "loseless". Akan tetapi, terdapat batasan yang tegas pada kemampuan mengompresi yang didapat dari kompresi data secara "lossless". Rasio kompresi secara "lossless" biasanya berkisar antara 2:1 sampai 8:1. Salah satu teknik kompresi secara "lossless" yang akan dibahas pada makalah ini adalah entropy encoding, yang akan dibahas pada subbab berikutnya. Untuk lebih jelas mengenai klasifikasi kompresi data, silahkan melihat gambar berikut.



1.2. Entropy Coding

1.2.1Entropy Coding dan metodenya

Entropi secara umum dapat diinterpretasikan jumlah rata-rata minimum dari jumlah pertanyaan ya/tidak untuk menentukan harga spesifik dari variable x. Dalam konteks bahasan, entropi merepresentasikan batas bawah (lower bound) dari jumlah rata-rata bit per satu nilai input – yaitu rata-rata panjang code word digunakan untuk mengodek input.

Menurut rumus, entropi adalah sebuah himpunan elemen unik $e_i,\ldots,e_n$ dengan peluang pi,…

$$H(pi...pn) = -\sum_{\forall i} pi \log 2\, pi$$

Contoh entropi :
Misal elemen A dan B dengan pA=0.5 dan pB=0.5

Maka H(A,B)= - pA $\log_2$ pA - pB $\log_2$ pB
H(A,B)= - 0.5 $\log_2$ 0.5 – 0.5 $\log_2$ 0.5
H(A,B)= 1

Hasil diatas menunjukkan bahwa dibutuhkan 1 bit secara rata-rata per symbol untuk merepresentasikan data tersebut

Sementara metode dari entropy encoding adalah pengkodean data dengan BPS (bits per symbol) untuk tiap alphabet yang mendekati nilai entropi, karena semakin dekat BPS dari alphabet tersebut dengan nilai entropi, semakin efisien pula kode kompresi tersebut.

Pada makalah ini, kode yang termasuk dalam entropy encoding yang akan dibahas adalah kode Huffman dan kode Aritmatik.

\
1.2.2Kamus Kode

- Codeword – sebuah susunan kode biner untuk merepresentasikan tiap karakter pada suatu data

- Fixed-length codes (kode dengan panjang tertentu) – kode yang tiap codeword-nya memiliki panjang yang sama (banyak bit)

  Misal : A-001, B-110, C-101, D-111

- Variable-length codes (kode dengan panjang berubah-ubah) – tiap codeword dapat berbeda-beda panjang

  Misal : A-0, B-101, C-01

- Prefix code (kode prefiks) – tiap kode tidak dapat menjadi header (bagian awal) dari kode lainnya

Misal : A-1, B-01, C-001

- Uniquely decodable code (kode yang didekode secara unik) – kode yang hanya mempunyai satu kemungkinan string sumber (string source) untuk memproduksinya

2. PENJELASAN KODE HUFFMAN DAN KODE ARITMATIK

2.1. Huffman Coding

Karakteristik kode Huffman :
• Tiap symbol diberikan suatu variable-length codes bergantung pada frekuensinya. Semakin tinggi frekuensinya, semakin pendek codeword-nya
•Jumlah bit untuk tiap codeword-nya adalah sebuah nilai integral
• Kode huffman adalah sebuah kode prefix
• Kode Huffman pun sebuah variable-length code
• Kode Huffman adalah kode prefix dan variable-length code yang optimal, bila bila diberikan peluang kemunculan tiap symbol
•Codeword dapat didapatkan dengan membentuk pohon Huffman

Untuk melihat bagaimana cara kerja Kode Huffman, terlebih dahulu kita harus membentuk pohon Huffman. Berikut ini adalah tahapan dalam membentuk pohon Huffman :

Inisialisasi :
- Buat daun dari tiap symbol X dengan beban= pX
- Berat bias berbentuk probabilitas ataupun banyaknya kemunculan dari symbol

Algoritma :

1. Buat node daun untuk tiap symbol, dan biarkan tiap node berisi probabilitas atau tinghkat kemunculan dari symbol. Kumpulan dari node ini disortir berdasarkan berkurangnya probablilitas

2. Buatlah sebuah node baru dari dua buah node tak berorangtua dengan probabilitas terendah dan buatlah node tersebut menjadi node orangtua dari kedua node tersebut. Isi dari node baru adalah jumlah dari isi dua node tersebut

3. Ulangi langkah 2 hingga sebuah node takberorangtua tercipta. Node ini adalah sebuah akar dari pohon Huffdman'

4. Pasangkan digit 0 dan 1 pada setiap sisi kanan dan kiri (atau atas dan bawah bergantung pada orientasi pohon)

5. Untuk menemukan kode dari symbol, ikuti tiap sisi dari node akar hingga ke node daun, dan gabungkan tiap digit yang telah dilewati

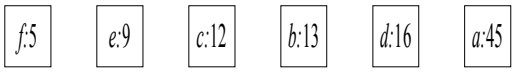
Untuk lebih jelasnya, silahkan ikuti contoh berikut

Misalkan data panjangnya 100 karakter dan disusun oleh huruf-huruf *a*, *b*, *c*, *d*, *e*, dengan frekuensi kemunculan setiap huruf sebagai berikut:

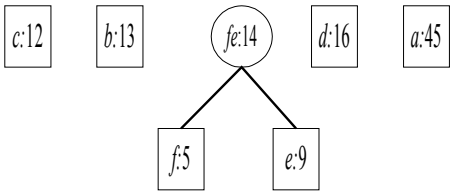
Karakter	*a*	*b*	*c*	*d*	*e*	*f*
Frekuensi	45	13	12	16	9	5

Maka dengan mengikuti algoritma yang telah dicantumkan diatas akan didapatkan pohon Huffman :

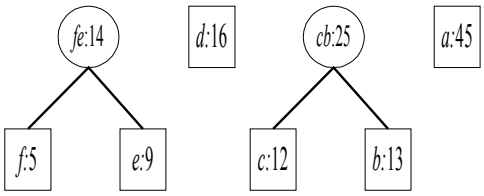
1.



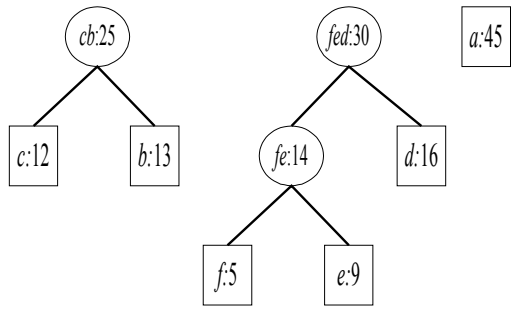
2.



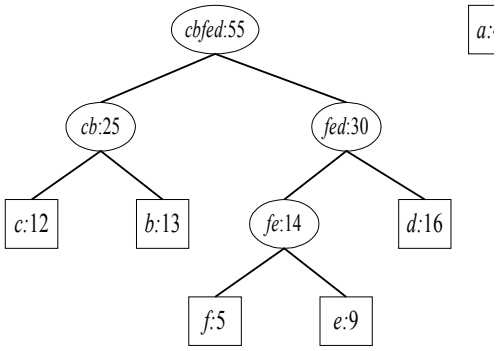
3.



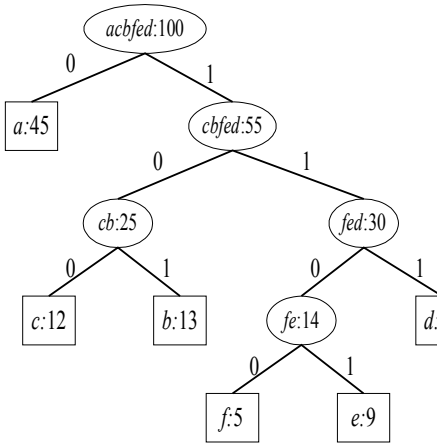
4.



5.



6



Untuk menghasilkan kode Huffman untuk bcd misalnya, didapatkan bcd = 101100111 dan untuk mengdekodekannya, kita hanya harus mengikuti sesuai algoritma dan mendapatkan bcd kembali

Kemampuan kompresi Kode Huffman

Misalkan kita memiliki 100.000 karakter yang akan kita simpan secara kompak. Kita teliti bahwa kemunculan karakter ada pada tabel. Ada beberapa cara untuk merepresentasikan informasi file tersebut. Dalam kasus ini kita reperesentasikan dalam kode biner. Setiap karakter direpresentasikan dengan string biner yang unik. Jika kita menggunkan kode panjang tetap (*fixed-length code*), kita mebutuhkan 3 bit untuk merepresentasikan tiap karakter:
A=000, B = 001, ...F=101
Metode ini membutuhkan 300.000 bit untuk merepresentasikan 100.000 karakter tersebut. Dengan teknik Hufman kita akan menghitung banyak kemuculan tiap karakter dan membentuk pohon Huffman sehingga panjang tiap karakter dapat dihemat dan dengan kode Huffman kita bisa memperkecil ukuran penyimpanan hanya menjadi 224.00 bit. Dan kita telah melakukan kompresi data sebanyak 25 %.

2.2. Arithmetic Coding

Karakteristik kode Aritmatik :

- Meng-assign codeword ke seluruh input stream
- Membaca input stream secara symbol demi symbol, menggabungkan lebih dan lebih bit kedalam codeword setiap kalinya
- Codeword adalah angka yang merepresentasikan subseksi segmental yang berdasar pada probailitas symbol
- Mengkode symbol dengan bit angka non-integer, yang menghasilkan hasil yang sangan baik (mendekati nilai entropi)

Untuk memperlihatkan` cara kerja kode Aritmatik, kita pertama-tama membuat table, yang berisi karakter-karakter dengan probabilitas dan hasil penjumlahannya. Hasil penjumlahan yang kumulatif mendefinisikan "interval", yang berkisar dari nilai bawah hingga suatu besaran yang mendekati nilai atas. Susunan karakter yang dimasukkan tidak penting.

simbol	probabilitas	interval
C:	0.3	0.0 : 0.3
B:	0.2	0.3 : 0.5
A:	0.5	0.5 : 1.0

Now each character can be coded by the shortest binary fraction whose value falls in the character's probability interval:

sbl	prob	interval	fraksi biner
C:	0.3	|0.0 : 0.3|	0
B:	0.2	|0.3 : 0.5|	0.011=3/8= 0.375
A:	0.5	|0.5 : 1.0|	0.1 = 1/2 = 0.5

Tabel diatas menunjukkan bahwa tiap karakter dapat direpresentasikan oleh codeword minimum. Akan tetapi kode Aritmetik tidak hanya berhenti disini dan mentranslasi karakter individual menjadi codeword. Dengan pendekatan lebih dalam, perepresentasian fraksi biner untuk melengkapi keseluruhan pesan dapat dilakukan

Untuk permulaan, marilah berpikir untuk mengirim pesan berisi seluruh kemungkinan permutasi dari dua karakter. Dengan mengalikan probabilitas dari dua karakter didapatkan probabilitas dari string yang berisi dua karakter tersebut dan menciptakan sekumpulan interval dengan probabilitas tersebut.

str	prob	intrv	fraksi biner
CC:	0.09	0.00 : 0.09	0.000=0.0625
CB:	0.06	0.09 : 0.15	0.001=0.125

```
CA:  0.15 0.15 : 0.30 0.01 =0.25
BC:  0.06 0.30 : 0.36 0.0101=0.3125
BB:  0.04 0.36 : 0.40 0.011 0.375
BA:  0.10 0.40 : 0.50 0.0111=0.4375
AC:  0.15 0.50 : 0.65 0.1  = 0.5
AB:  0.10 0.65 : 0.75 0.1011=0.6875
AA:  0.25 0.75 : 1.00 0.11=0.75
```

Semakin besar probabilitas dari string, semakin prndek fraksi biner yang diperlukan untuk merepresentasikannya

Sementara untuk pendekodean kita ambil contoh codeword CA dari atas (0.25). Hasil ini diambil dari table diatas

string	probabilitas	interval
C:	0.3	0.0 : 0.3
B:	0.2	0.3 : 0.5
A:	0.5	0.5 : 1.0

Nilai 0.25 jelas masuk dalam interval untuk “C”, jadi c adalah karakter pertama. Kita pun masih bisa melihat lebih dalam untuk mencari karakter selanjutnya dengan mengurangi 0.25 dengan nilai bawah “C”, yang kebetulan adalah 0, dan membagi hasilnya dengan interval probabilitas “C” yaitu 0.3

```
(0.25 - 0) / 0.3  =  0.83333
```

Hasilnya jatuh kepada interval untuk “A” oleh karenanya “A” adalah karakter selanjutnya.

3. PERBANDINGAN EFISIENSI SECARA TEORITIS

3.1 Komparasi dari segi kecepatan
Hasil pengamatan secara teoritis menyatakan bahwa Huffman Coding lebih cepat dari Aritmetic Coding

3.2 Komparasi dari segi ukuran kompresi data
Hasil Pengamatan secara teoritis menyatakan bahwa Arithmetic Coding lebih efisien (data kompresi lebih kecil) dari Huffman Coding

4. KESIMPULAN

4.1 Kelebihan masing-masing kode
Huffman Coding : lebih cepat
Arithmetic Coding : lebih efisien

4.2 Kekurangan masing-masing kode
Huffman Coding : kurang efisien
Arithmetic Coding : kurang cepat

4.3 Kode Huffman dan Kode Aritmatik dapat digunakan secara efisien pada pemampatan data

DAFTAR REFERENSI

[1] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] Wikipedia (2007). http://www.en.wikipedia.org/wiki/Huffman_coding. Tanggal akses: 29/12/2007 pukul 15.00

[3] Wikipedia (2007). http://www.en.wikipedia.org/wiki/Entropy_encoding. Tanggal akses: 1/1/2007 pukul 10.19

[4] Wikipedia (2007). http://www.en.wikipedia.org/wiki/Arithmetic_encoding. Tanggal akses: 1/1/2007 pukul 11.20