BAB II

LANDASAN TEORI

2.1 Data

Kata “data” diadopsi dari bahasa Inggris dan berasal dari kata Yunani “datum” yang berarti “fakta”. Data di komputer memiliki ukuran dalam penyebutannya. Data terkecil dalam komputer disebut dengan *bit*, yaitu sinyal elektronik yang melewati suatu rangkaian digital (prosesor) komputer. *Bit-bit* tersebut selanjutnya dirangkai dan rangkaian tersebut diberi kode lagi yang disebut dengan *character* (Wahyudi, 2008)

2.2 *File* Multimedia

Multimedia dapat diartikan sebagai kombinasi dari macam-macam objek *multimedia*, yaitu teks, gambar, animasi, audio, dan video untuk menyajikan informasi. Setiap objek multimedia memerlukan cara penanganan tersendiri, dalam hal kompresi data, penyimpanan, dan pengambilan kembali untuk digunakan. Multimedia terdiri dari beberapa objek, yaitu teks, grafik, gambar, animasi, audio, dan video (Setiabudi, 2005).

2.3 Dropbox

Dropbox merupakan sebuah layanan gratis yang memungkinkan Anda untuk menyimpan foto, dokumen, dan video dimanapun serta membagikannya untuk orang lain dengan mudah. Dropbox ditemukan pada tahun 2007 oleh Drew

Houston dan Arash Ferdowsi, yang merupakan dua mahasiswa MIT yang lelah karena harus mengirimkan *e-mail* satu sama lain ketika saat bekerja dengan menggunakan lebih dari satu komputer (Dropbox, 2014).

2.4 Kompresi

Proses kompresi data didasarkan pada kenyataan bahwa pada hampir semua jenis data selalu terdapat pengulangan pada komponen data yang dimilikinya, misalnya di dalam suatu teks kalimat akan terdapat pengulangan penggunaan huruf alphabet dari huruf a sampai dengan huruf z. Kompresi data melalui proses *encoding* berusaha untuk menghilangkan unsur pengulangan ini dengan mengubahnya sedemikian rupa, sehingga ukuran data menjadi lebih kecil baik untuk berbagai jenis data seperti data teks, gambar, video, suara, dan lain-lain. Proses pengurangan unsur pengulangan ini dapat dilakukan dengan memakai beberapa teknik kompresi. Misalnya jika suatu komponen muncul berulang kali dalam suatu data, maka komponen tersebut tidak harus dikodekan berulang kali pula tapi dapat dikodekan dengan menulis frekuensi munculnya komponen dan di mana komponen tersebut muncul. Teknik kompresi data lainnya, berusaha untuk mencari suatu bentuk kode yang lebih pendek untuk suatu komponen yang sering muncul. Keberhasilan pengkompresian data tergantung dari besarnya data itu sendiri dan tipe data yang memungkinkan untuk dikompresi (Gozali, 2004).

Aplikasi kompresi data yang dilakukan bertujuan untuk mengurangi redundansi dari data-data yang terdapat dalam sebuah *file*, sehingga dapat disimpan atau ditransmisikan secara efisien. Informasi yang dirasakan redundan

dapat dikompresi untuk meminimalisasi redundansi tersebut. Hal ini dapat pula dilakukan pada *file* multimedia yang beragam tipenya dengan menerapkan metode kompresi yang ada. Tujuan utama dari kompresi pada *file* multimedia adalah untuk mengurangi penggunaan memori, sehingga akan memudahkan penyimpanan, pengolahan, serta pengiriman *file* multimedia tersebut. Dapat disimpulkan bahwa kompresi merupakan proses untuk menghilangkan berbagai kerumitan yang tidak penting (redundansi) dari suatu informasi, dengan memaksimalkan kesederhanaannya dan tetap menjaga kualitas penggambaran dari informasi tersebut (Shofiyah, 2010).

2.5 Kompresi *Lossy*

Kompresi menggunakan *lossy*, beberapa bagian data asli hilang ketika berkas di *decoded*. Keuntungan dari algortima ini adalah bahwa rasio kompresi cukup tinggi. Hal ini dikarenakan cara algoritma *lossy* yang mengeliminasikan beberapa data dari suatu berkas. Namun data yang dieliminasikan biasanya adalah data yang kurang diperhatikan atau di luar jangkauan manusia, sehingga pengeliminasian data tersebut kemungkinan besar tidak akan mempengaruhi manusia yang berinteraksi dengan berkas tersebut. Beberapa algoritma *lossy* digunakan di operasi video dengan hanya menyimpan perbedaan di antara *frame* berturut-turut (Megasari, 2007).

2.6 Kompresi *Lossless*

Kompresi menggunakan *lossless* menjamin bahwa berkas yang dikompresi dapat selalu dikembalikan ke bentuk aslinya. Algoritma *lossless* digunakan untuk kompresi berkas *text*, seperti program komputer (berkas zip, rar), karena ingin mengembalikan berkas yang telah dikompres ke status aslinya (Megasari, 2007).

2.7 Run Length Encoding

Run Length Encoding (RLE) adalah algoritma kompresi data yang bersifat *lossless*. Algoritma ini cukup mudah untuk dipahami dan diterapkan untuk kompresi data. Metode kompresi ini sangat sederhana, dimana dengan memindahkan pengulangan *byte* yang sama secara terus menerus (Rahandi, 2012).

RLE meng*encode* urutan *byte* yang memiliki nilai yang sama sebagai sebuah *codeword*. Sebagai contoh terdapat urutan *byte* 77 77 77 77 77 77 77 dapat dikodekan menjadi 7 77 (terdapat 7 buah angka 77).

2.7.1 Proses *Encoding*

Untuk melakukan *encoding* dipakai algoritma sebagai berikut.

1. Baca *file input* dari data yang akan dikompresi.
2. Lihat apakah terdapat deretan karakter yang sama secara berurutan lebih dari dua karakter, jika terdapat hal tersebut lakukan kompresi. Pada contoh di atas deretan karakter yang sama secara berurutan sebanyak 7 karakter, jadi lebih dari dua dan dapat dilakukan kompresi.

3. Berikan *byte* penanda pada file kompresi, bit penanda disini berupa duplikat dari karakter yang dikompresi. Deretan *byte* yang boleh dipilih sembarang asalkan digunakan secara konsisten pada seluruh bit penanda hasil kompresi. *Byte* penanda ini berfungsi untuk menandai bahwa karakter selanjutnya adalah karakter yang dikompresi sehingga tidak membingungkan pada saat mengembalikan file yang sudah dikompresi ke file aslinya.
4. Tambahkan deretan *byte* untuk menyatakan jumlah karakter yang sama berurutan.
5. Tambahkan deretan bit yang menyatakan karakter yang berulang.

2.7.2 Proses *Decoding*

Untuk melakukan *decoding* dipakai algoritma sebagai berikut.

1. Baca *file* hasil kompresi data.
2. Lihat karakter pada hasil kompresi satu-persatu dari awal sampai akhir, jika ditemukan *byte* penanda, lakukan proses pengembalian.
3. Lihat karakter setelah *byte* penanda, lihat nilai yang disimpan untuk menentukan jumlah karakter yang berurutan.
4. Lihat karakter pada *byte* penanda, kemudian lakukan penulisan karakter tersebut sebanyak bilangan yang telah diperoleh pada karakter sebelumnya (langkah 3).

2.8 Arithmetic Coding

Ide dasar dari *Arithmetic Coding* adalah membuat suatu garis peluang dari 0 sampai 1 dan memberikan interval pada setiap karakter dari teks *input* berdasarkan peluang kemunculannya. Semakin tinggi peluang yang dimiliki suatu karakter, maka semakin besar interval yang akan diperoleh. Setelah semua karakter memiliki interval, maka dilakukan pengkodean untuk menghasilkan satu bilangan *output*. Sebagai contoh untuk teks ABCAD, distribusi peluang dari teks tersebut dapat dilihat pada tabel berikut:

Tabel 2.1 Distribusi Probabilitas untuk Kata 'ABCD'

KARAKTER	PROBABILITAS
A	0.4
B	0.2
C	0.2
D	0.2

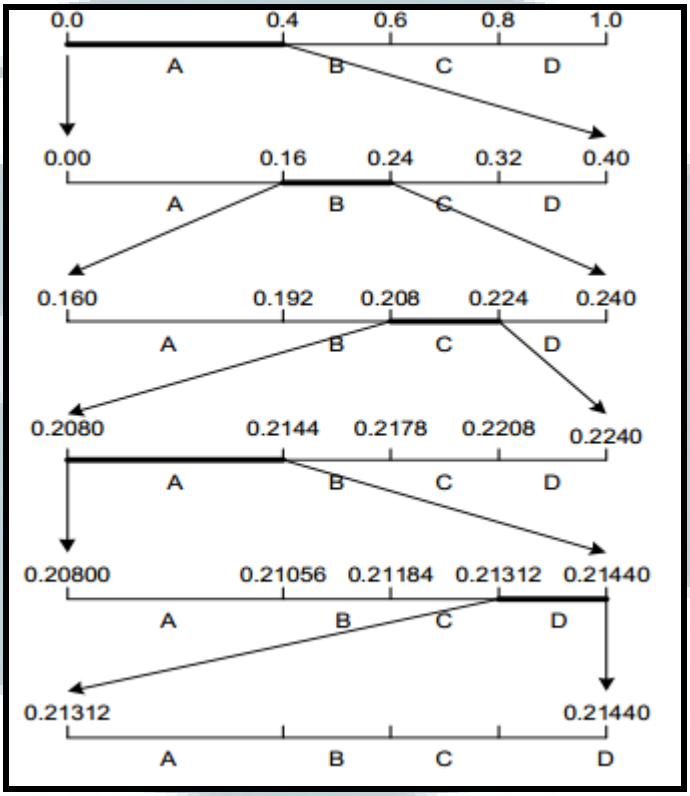
Tabel 2.2 di bawah ini menunjukkan interval yang diberikan pada masing-masing karakter:

Tabel 2.2 Tabel Probabilitas Kata 'ABCAD'

KARAKTER	PROBABILITAS	RANGE
A	0.4	[0, 04)
B	0.2	[0.4, 0.6)
C	0.2	[0.6, 0.8)
D	0.2	[0.8, 1.0)

Karakter pertama yang terbaca pada proses pengkodean akan mempersempit panjang garis peluang menjadi sebesar interval yang dimiliki oleh karakter tersebut. Pada garis peluang yang baru akan dilakukan partisi lagi dengan porsi yang sama seperti pada garis peluang sebelumnya. Ilustrasi dari keseluruhan

proses partisi dan garis peluang untuk teks ABCAD dapat dilihat pada gambar di bawah ini.



Gambar 2.1 Proses partisi pada peluang untuk teks ABCAD

Untuk hasil kompresi diambil satu nilai yang berada di dalam interval terakhir, misalnya nilai batas bawahnya yaitu 0.21312.

Algoritma pengkodean aritmetika dapat menghitung garis peluang yang baru tanpa harus melakukan pembagian interval dari setiap garis peluang yang baru (Silalahi, 2010)

2.8.1 Proses *Encoding*

Untuk melakukan *encoding* dipakai algoritma sebagai berikut.

- Set *low* = 0.0
- Set *high* = 1.0
- *While* (simbol masih ada) *do*
  - Ambil simbol *input*
  - CR = *high* - *low*
  - *High* = *low* + CR**high_range*(simbol)
  - *Low* = *low* + CR**low_range*(simbol)
- End *While*
- Cetak *Low*

Di sini '*Low'* adalah *output* dari proses *Arithmetic Coding* dan CR adalah *Compression Range* dari setiap tahap kompresi, dimana nilai CR akan selalu berubah setiap simbol dilakukan kompresi. Sebagai contoh akan dilakukan kompresi untuk kata 'TELEMATIKA'. Berikut merupakan tabel probabilitas untuk kata 'TELEMATIKA'.

Tabel 2.3 Tabel Probabilitas untuk Kata 'TELEMATIKA'

KARAKTER	PROBABILITAS	RANGE
A	2/10	0.00 - 0.20
E	2/10	0.20 - 0.40
I	1/10	0.40 - 0.50
K	1/10	0.50 - 0.60
L	1/10	0.60 - 0.70
M	1/10	0.70 - 0.80
T	2/10	0.80 - 1.00

Pertama ambil karakter ‘T’. Nilai CR adalah 1-0 = 1. *High_range*(T) = 1.00, *Low_range*(T) = 0.80. Kemudian didapatkan nilai *high* = 0.00 + CR*1.00 = 1.00 dan *low* = 0.00 + CR*0.80 = 0.80. Selanjutnya diambil karakter ‘E’. Nilai CR adalah 1.0 – 0.8 = 0.2. *High_range*(E) = 0.40, *Low_range*(E) = 0.20. Kemudia didapatkan nilai *high* = 0.80 + CR*0.4 = 0.88 dan *low* = 0.80 + CR*0.2 = 0.84. Untuk proses selanjutnya akan diringkaskan pada tabel berikut.

Tabel 2.4 Proses *Encoding* untuk Kata ‘TELEMATIKA’

S	Low	High	CR
	0.0	1.0	1.0
T	0.8	1.0	0.2
E	0.84	0.88	0.04
L	0.864	0.868	0.004
E	0.8648	0.8656	0.0008
M	0.86536	0.86544	8E-05
A	0.86536	0.86544	1.6E-05
T	0.8653728	0.865376	3.2E-06
I	0.86537408	0.8653744	3.2E-07
K	0.86537424	0.865374272	3.2E-08
A	0.86537424	0.865374246	6.4E-09

Dari proses ini didapatkan nilai *Low* = 0.86537424. Nilai inilah yang akan ditransmisikan untuk membawa pesan telematika.

2.8.2 Proses *Decoding*

Untuk melakukan *decoding* dipakai algoritma sebagai berikut.

- Ambil *encoded-symbol* (ES)
- *Do*
  - Cari *range* dari simbol yang melingkupi ES

- Cetak simbol
- CR = *high_range – low_range*
- ES = ES – *low_range*
- ES = ES / CR

- *Until* simbol habis

Dalam hal ini simbol habis bisa ditandai dengan simbol khusus (seperti simbol *End of Message*) atau dengan menyertakan panjang pesan waktu dilakukan transmisi.

Untuk pesan yang tadi di-*encode* (ES = 0.86537424) dilakukan proses *decoding* seperti berikut. Didapatkan *range* simbol yang melingkupi ES adalah simbol/karakter 'T'.

- *Low_range* = 0.8
- *High_range* = 1.0
- CR = 1.0 – 0.8 = 0.2
- ES = 0. 86537424 – *low_range*
- ES = 0.06537424
- ES = 0.06537424 / CR
- ES = 0.3268712

Proses *decoding* akan terus dilakukan selama ES masih memiliki nilai. Untuk lebih ringkas, proses *decoding* akan dilanjutkan dalam bentuk tabel sebagai berikut.

Tabel 2.5 Proses Decoding untuk Kata ‘TELEMATIKA’

ES	S	Low	High	CR
0.86537424	T	0.8	1.0	0.2
0.3268712	E	0.2	0.4	0.2
0.634356	L	0.6	0.7	0.1
0.34356	E	0.2	0.4	0.2
0.7178	M	0.7	0.8	0.1
0.178	A	0.0	0.2	0.2
0.89	T	0.8	1.0	0.2
0.45	I	0.4	0.5	0.1
0.5	K	0.5	0.6	0.1
0	A	0.0	0.2	0.2

Pada proses *decoding* di atas, didapatkan hasil berupa kata ‘TELEMATIKA’. Proses *decoding* akan berakhir ketika ES sudah tidak memiliki nilai lagi, seperti pada tabel di atas di mana nilai ES = 0.