KOMPRESI STRING
MENGGUNAKAN ALGORITMA LZW DAN HUFFMAN

Muhammad Maulana Abdullah / 13508053

Program Studi Teknik Informatika 2008
Bandung
e-mail: if18053@students.if.itb.ac.id

ABSTRAK

Saat ini kompresi file dalam bentuk zip sudah tidak asing lagi di telinga pengguna komputer sehari-hari. Proses penyimpanan, pengiriman data maupun kebutuhan bandwidth sangat penting karena informasi terkini sebagian besar berasal dari dunia maya yaitu internet. Terutama jika berkaitan dengan attachment file dalam pengiriman email, menyalin file dari komputer ke media storage yang lain, baik mengunduh atau mengunggah, dan lain sebagainya. Kompresi data merupakan suatu cara yang dapat dimanfaatkan untuk mengatasi ukuran suatu file yang besar sehingga masalah penyimpanan, pengiriman data maupun kebutuhan bandwidth yang digunakan dapat diatasi. Berbagai algoritma sederhana merupakan konsep dari kompresi .zip ini. Oleh karena itu, pada kesempatan kali ini dibahas perbandingan konsep Huffmann dan LZW. Dengan kata lain, Pemilihan algoritma kompresi sangat menentukan rasio kompresi. Oleh karena itu, diperlukan perbandingan rasio kompresi terbaik dari beberapa algoritma kompresi agar mengurangi pemborosan memori serta efisiensi waktu kompresi.

Kata kunci: Zip, LZW, Huffmann.

1. PENDAHULUAN

Dalam komunikasi data, pesan yang dikirim seringkali ukurannya sangat besar sehingga waktu pengirimannya lama. Begitu juga dalam penyimpanan data, arsip atau file yang berukuran besar memakan ruang penyimpanan yang besar. Kedua masalah tersebut dapat diatasi dengan mengkodekan pesan atau isi arsip sesingkat mungkin, sehingga waktu pengiriman pesan juga relatif cepat, dan ruang penyimpanan yang dibutuhkan juga sedikit. Cara pengkodean seperti ini disebut pemampatan atau kompresi data.

Proses penghematan data bit berdasarkan kebutuhan tempat penyimpanan dalam transmisi data disebut kompresi.

Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (isi file input) menjadi sekumpulan codeword, metode kompresi terbagi menjadi dua kelompok, yaitu :

· Metode statik : menggunakan peta kode yang selalu sama. Metode ini membutuhkan dua fase (two-pass): fase pertama untuk menghitung probabilitas kemunculan tiap simbol/karakter dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan ditransmisikan.
Contoh: algoritma Huffman statik.

· Metode dinamik (adaptif) : menggunakan peta kode yang dapat berubah dari waktu ke waktu. Metode ini disebut adaptif karena peta kode mampu beradaptasi terhadap perubahan karakteristik isi file selama proses kompresi berlangsung. Metode ini bersifat onepass, karena hanya diperlukan satu kali pembacaan terhadap isi file. Contoh: algoritma LZW.

Berdasarkan teknik pengkodean/pengubahan simbol yang digunakan, metode kompresi dapat dibagi ke dalam tiga kategori, yaitu :

· Metode symbolwise : menghitung peluang kemunculan dari tiap simbol dalam file input, lalu mengkodekan satu simbol dalam satu waktu, dimana simbol yang lebih sering muncul diberi kode lebih pendek dibandingkan simbol yang lebih jarang muncul, contoh: algoritma Huffman.

· Metode dictionary : menggantikan karakter/fragmen dalam file input dengan indeks lokasi dari karakter/fragmen tersebut dalam sebuah kamus (dictionary), contoh: algoritma LZW.

· Metode predictive : menggunakan model finite-context atau finite-state untuk memprediksi distribusi probabilitas dari simbol-simbol selanjutnya; contoh: algoritma DMC.

Jenis Kompresi Data Berdasarkan Mode Penerimaan Data oleh manusia :

• Dialoque Mode: yaitu proses penerimaan data dimana pengirim dan penerima seakan berdialog (real time), seperti pada contoh video conference.

o Dimana kompresi data harus berada dalam batas penglihatan dan pendengaran manusia. Waktu tunda (delay) tidak boleh lebih dari 150 ms, dimana 50 ms untukproses kompresi dan dekompresi, 100 ms mentransmisikan data dalam jaringan

• Retrieval Mode: yaitu proses penerimaan data tidak dilakukan secara real time.
o Dapat dilakukan fast forward dan fast rewind di client
o Dapat dilakukan random access terhadap data dan dapat bersifat interaktif.

Jenis Kompresi Data Berdasarkan Output
• Lossy Compression
o Teknik kompresi dimana data hasil dekompresi tidak sama dengan data sebelum kompresi namun sudah "cukup" untuk digunakan. Contoh: Mp3, streaming media, JPEG, MPEG, dan WMA.
o Kelebihan: ukuran file lebih kecil dibanding loseless namun masih tetap memenuhi syarat untuk digunakan.
o Biasanya teknik ini membuang bagian-bagian data yang sebenarnya tidak begitu berguna, tidak begitu dirasakan, tidak begitu dilihat oleh manusia sehingga manusia masih beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi.
o Misal terdapat image asli berukuran 12,249 bytes, kemudian dilakukan kompresi dengan JPEG kualitas 30 dan berukuran 1,869 bytes berarti image tersebut 85% lebih kecil dan ratio kompresi 15%.
• Loseless
o Teknik kompresi dimana data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi.
Contoh aplikasi: ZIP, RAR, GZIP, 7-Zip
o Teknik ini digunakan jika dibutuhkan data setelah dikompresi harus dapat diekstrak/dekompres lagi tepat sama.
Contoh pada data teks, data program/biner, beberapa image seperti GIF dan PNG.
o Kadangkala ada data-data yang setelah dikompresi dengan teknik ini ukurannya menjadi lebih besar atau sama.

2. KONSEP DASAR KOMPRESI STRING

Dalam pengkompresian string ini akan di bahas 2 konsep dasar yaitu :

1. Algoritma Huffman
2. Algoritma LZW atau *Lempel-Ziv-Welch*

2.1 Algoritma Huffman

Algoritma Huffman merupakan penemuan dari mahasiswa MIT, David Huffman.

Ide dari algoritma Huffman adalah meminimkan jumlah bit yang memiliki kekerapan kemunculan hurufnya dalam sebuah string itu tinggi. Kode dari setiap huruf tidak boleh merepresentasikan huruf yang lain atau dengan kata lain setiap huruf memiliki kode peminimisasian yang berbeda-beda. Karena akan menyebabkan keraguan dalam proses pemulihanny atau decoding.

Langkah pengerjaan algoritma tersebut antara lain :

1. Mentraversal sebuah string atau masukan dari user dan menghitung kemunculan atau kekerapan kemunculan setiap huruf berbeda dalam sebuah string. Setelah itu buat daftar dari huruf-huruf tersebut beserta peluang kemunculannya karena huruf-huruf tersebut akan menjadi daun dalam poho Huffman.
2. Mentraversal kembali daftar yang telah dibuat untuk kemudian membedakan daun (berupahuruf dengan peluang terkecil) dan penjumlahan 2 daun yang akan menjadi akar dari dua daun sebelumnya.

Sebagai contoh untuk string ABACCDA.

Tabel Kode ASCII

Simbol	Kode ASCII
A	01000001
B	01000000
C	01000011
D	01000100

Jika tidak dikompresi akan membentuk rangkaian bit seperti berikut :

01000001 01000000 01000001 01000011 01000011 01000100 01000001

Jika terdapat 7 karakter dalam sebuah string maka memori yang dibutuhkan adalah 7 x 8 bit. Atau dengan kata lain akan dibutuhkan :

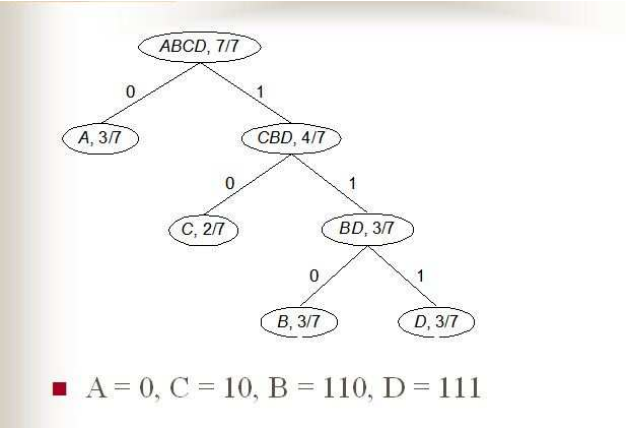
Memori = n x 8 bit (1)

n = jumlah karakter dalam sebuah string

Dari langkah pengerjaan algoritma Huffman langkah 1 akan dihasilkan tabel seperti berikut :

Tabel Kode Huffman

Simbol	Kekerapan	Peluang	Kodi Huffman
A	3	3/7	0
B	1	1/7	110
C	2	2/7	10
D	1	1/7	111



Gambar Pohon Huffman

Apabila terdapat tabel ditulis seperti contoh berikut:

Perhatikan bahwa simbol paling sering muncul memiliki kode dengan jumlah bit paling sedikit. Perhatikan pula bahwa tidak ada simbol yang kodenya merupakan kode awalan untuk simbol yang lain. Dengan cara ini, kode yang diawali 0 dapat dipastikan adalah A, sedangkan kode yang diawali 1 mungkin B atau C atau D yang harus diperika lagi dari bit-bit selanjutnya.

ABACCDA = 0 110 0 10 10 111 0

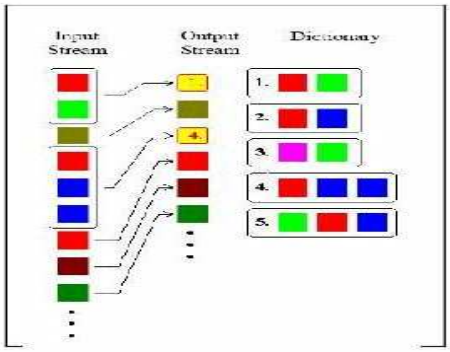
Dapat terlihat jelas bahwa sekarang untuk string tersebut hanya membutuhkan memori 13 bit.

2.2. Algoritma LZW

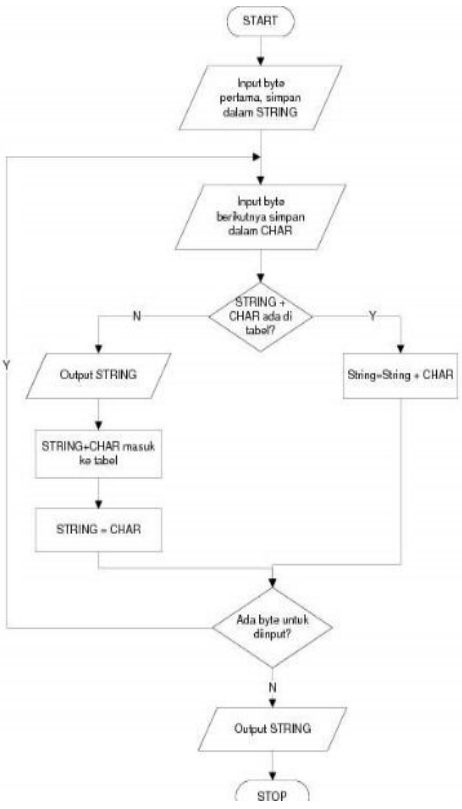
LZW merupakan kependekan kata dari Lempel-Ziv-Welch. Abraham Lempel, Jacob Ziv, dan Terry Welch adalah pencipta algoritma kompresi lossless universal ini. Kelebihan algoritma ini yaitu cepat dalam implementasi dan kekurangannya kurang optimal karena hanya melakukan analisis terbatas pada data. Algoritma ini melakukan kompresi dengan menggunakan kamus, di mana fragmen-fragmen teks digantikan dengan indeks yang diperoleh dari sebuah "kamus".

Pendekatan ini bersifat adaptif dan efektif karena banyak karakter dapat dikodekan dengan mengacu pada string yang telah muncul sebelumnya dalam teks. Prinsip kompresi tercapai jika referensi dalam bentuk pointer dapat disimpan dalam jumlah bit yang lebih sedikit dibandingkan string aslinya. Algoritma LZW secara lengkap:

1. Kamus diinisialisasi dengan semua karakter dasar yang ada : {'A'..'Z','a'..'z','0'..'9'}.
2. *P* <- karakter pertama dalam *stream* karakter.
3. C <- karakter berikutnya dalam stream karakter.
4. Apakah string (P + C) terdapat dalam dictionary ?
   - Jika ya, maka P <- P + C (gabungkan P dan C menjadi string baru).
   - Jika tidak, maka :
     i. Output sebuah kode untuk menggantikan string P.
     ii. Tambahkan string (P + C) ke dalam dictionary dan berikan nomor/kode berikutnya yang belum digunakan dalam dictionary untuk string tersebut.
     iii. P <- C.
5. Apakah masih ada karakter berikutnya dalam stream karakter ?
   i. Jika ya, maka kembali ke langkah 2.
   ii. Jika tidak, maka output kode yang menggantikan string P, lalu terminasi proses (stop).



Prinsip Algoritma LZW



Prinsip Algoritma Kompresi LZW

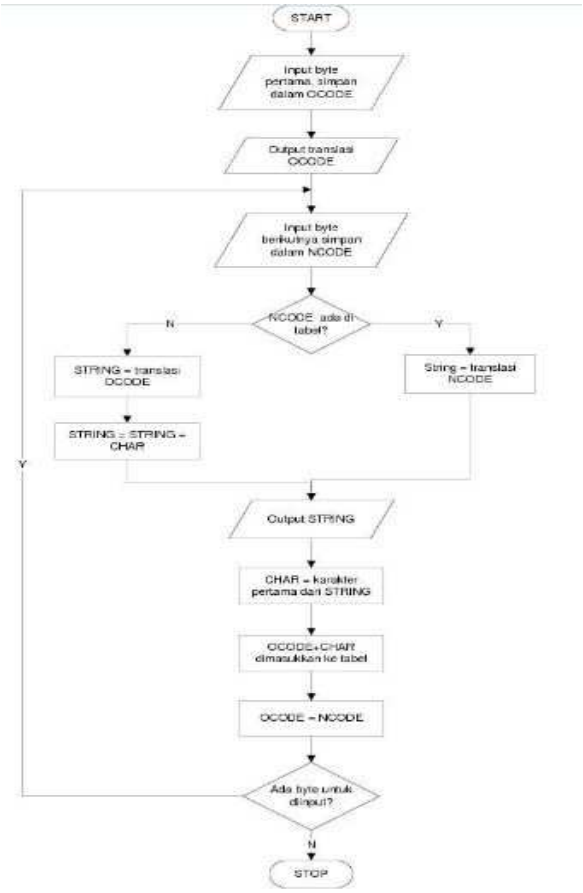
Sebagai contoh, string "ABACCDA" akan dikompresi dengan LZW. Isi dictionary pada awal proses diset dengan tiga karakter dasar yang ada: "A", "B", "C", "D".

Tabel Kompresi LZW

Langkah	Posisi	Karakter	Kamus	Output
1	1	A	[5]AB	1
2	2	B	[6]BA	2
3	3	A	[7]AC	3
4	4	C	[8]CC	4
5	5	C	[9]CD	5
6	6	D	[10]DA	6
7	7	A	--	--

Stream Karakter -> Kode Output
A = [1]
B = [2]
A = [1]
C = [3]
C = [3]
D = [4]
A = [1]

Frasa baru yang di tambahkan ke dalam kamus
5 = AB
6 = BA
7 = AC
8 = CC
9 = CD



10 = DA

Prinsip Algoritma Dekompresi LZW

Proses dekompresi pada LZW dilakukan dengan prinsip yang sama seperti proses kompresi. Algoritma diberikan sebagai berikut:

1. *Dictionary* diinisialisasi dengan semua karakter dasar yang ada : {'A'..'Z','a'..'z','0'..'9'}.
2. *CW* <- kode pertama dari stream kode (menunjuk ke salah satu karakter dasar).
3. Lihat dictionary dan output string dari kode tersebut (string.CW) ke stream karakter.
4. PW <- CW; CW <- kode berikutnya dari stream kode.
5. Apakah string.CW terdapat dalam dictionary ?
   - Jika ada, maka :
     1. output string.CW ke stream karakter
     2. P <- string.PW
     3. C <- karakter pertama dari string.CW
     4. tambahkan string (P+C) ke dalam kamus

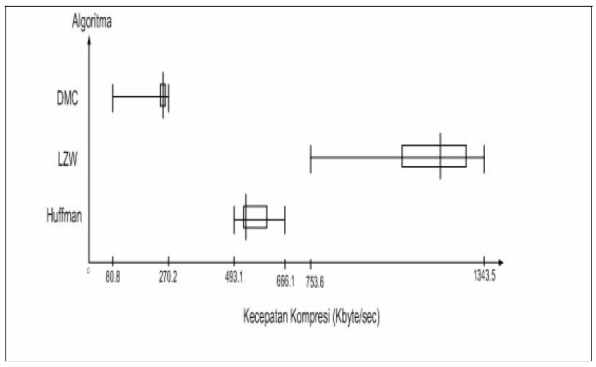
Tabel Dekompresi LZW

Langkah	Posisi	Output	Kamus
1	1	A	--
2	2	B	AB
3	3	A	BA
4	4	C	AC
5	5	C	CC
6	6	D	CD
7	7	A	A

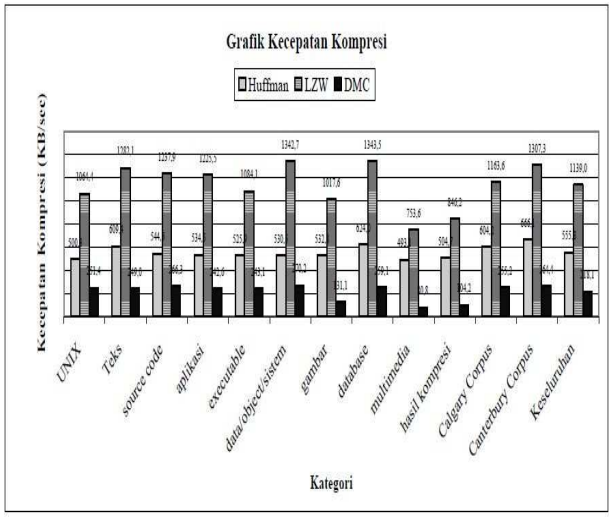
3. Perbandingan Komparasi String Algoritma Huffman dan Algoritma LZW

Dari data yang saya peroleh dari internet. Berikut merupakan perbandingan antara rasio kompresi Huffman dan LZW dalam sebuah percobaan yang dilakukan oleh Linawati dan Henry P. Panggabean, Jurusan Ilmu Komputer, FMIPA Universitas Katolik Parahyangan Bandung 40141.

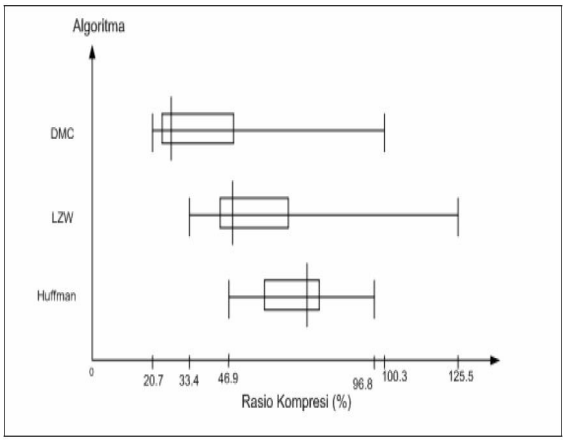
Percobaan ini dilakukan di atas mesin lingkungan perangkat keras sbb.: prosesor Intel Pentium IV 1.5 GHz, memori 128MB RAM, *harddisk* Maxtor 20 GB, dan *motherboard* Intel D850GB.



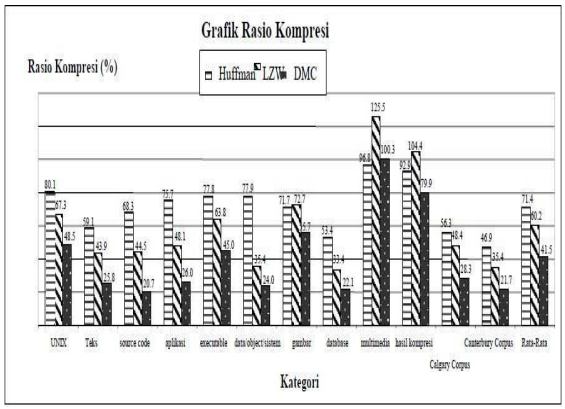
***Box Plot* Kecepatan Kompresi Algotritma Huffman dan LZW**



Grafik Perbandingan Kecepatan Algotritma Huffman dan LZW



Rasio Kompresi Algotritma Huffman dan LZW



Grafik Perbandingan Kompresi Algotritma Huffman dan LZW

4. KESIMPULAN

- Untuk mengompresi string terdapat berbagai macam algoritma, di antaranya adalah algoritma Huffman dan algoritma LZW.
- Konsep dasar dari kompresi .zip menggunakan algoritma Huffman dan algoritma LZW.
- Dari kedua algoritma ini, terdapat kelebihan atau keunggulan dan kelemahan nya masing-masing.
- Hasil kompresi Huffman lebih baik dibandingkan LZW hanya pada kasus file biner, file multimedia, file gambar, dan file hasil kompresi.
- Algoritma Huffman memberikan hasil yang relatif hampir sama untuk setiap kasus uji, sedangkan LZW memberikan hasil kompresi yang buruk (dapat > 100%) untuk file multimedia dan file hasil kompresi.
- Secara rata-rata algoritma LZW membutuhkan waktu kompresi yang tersingkat (kecepatan kompresinya = 1139 KByte/sec ± 192,5), diikuti oleh algoritma Huffman (555,8 KByte/sec ± 55,8).
- Kecepatan kompresi algoritma LZW secara signifikan berkurang pada file UNIX, file *executable*, file gambar, file multimedia, dan file hasil kompresi.
- Kecepatan kompresi algoritma Huffman hampir merata untuk semua kategori file.

REFERENSI

[1] Linawati dan Henry P. Panggabean, "Perbandingan Kinerja Algoritma Kompresi Huffman, LZW, DAN DMC pada Berbagai Tipe File", Integral, Vol. 9, No. 1, 2004, hal 14-15.
[2] Retno Wulandari, "Perbandingan Rasio Kompresi Pada Teknik Lossless", Vol. 1, No. 1, 2006, hal 1.
[3] http://jiunkpe/s1/info/2006/jiunkpe-ns-s1-2006-26402120-7900-kompresi-chapter1.pdf. 19 September 2010. 21.00.