RINGKASAN

Pada prinsipnya kompresi data dapat dicapai dengan mereduksi *redundancy*. Metoda kompresi data secara keseluruhan dapat dibagi ke dalam dua kelompok besar yakni pertama, kelompok metoda kompresi data boleh-hilang (*lossy compression)*, dan kedua, metoda kompresi *lossless*. Algoritma kompresi *lossless* bisa diklasifikasikan sebagai berikut: *RLE, Statistical Compression, Dictionary Based Compression,* dan metoda transformasi.

Riset ini menjelaskan tentang beberapa metoda kompresi *lossless* yang populer dan banyak dipakai orang untuk diimplementasikan dalam program, lalu digabungkan menjadi sebuah file *library* dengan memakai salah satu fitur dari pemrograman berorientasi objek, yakni konsep *class*. *Class* adalah konsep OO yang mengenkapsulasi data dan abstraksi prosedural yang diperlukan untuk menggambarkan isi serta tingkah laku berbagai entitas dunia nyata. Pengkapsulan dan penyembunyian data akan mengikat secara kuat data dan prosedur serta membatasi ruang lingkup dan keterlihatan prosedur dan fungsi yang mampu memanipulasi data pada bagian program yang terlokalisasikan pada perangkat lunak.

Tujuan dari riset ini adalah menciptakan atau menyediakan *class library* mengenai berbagai algoritma kompresi *lossless*, serta sebuah program aplikasi *toolbox* kompresi *lossless* untuk menguji kinerja *library* (*proof of concept*) dan untuk penelitian lebih lanjut (kompresi bertingkat). Dengan adanya *library* ini diharapkan para peneliti tentang kompresi data atau para programmer dapat menggunakan *class-class* yang ada dalam *library* tersebut untuk mendukung programnya

Penelitian ini diawali dengan tahapan studi literatur dan referensi tentang berbagai macam kategori algoritma kompresi lossless yang ada. Lalu dilanjutkan dengan proses analisis dan pengkajian dari hasil tahap tersebut. Dari hasil analisis dan pengkajian dapat ditetapkan algoritma kompresi *lossless* yang paling banyak digunakan dan diteliti terbagi dalam dua kategori besar yakni *statistical* dan *dictionary based compression.* Kategori *statistical compression* yang paling terkenal, sering digunakan, dan diteliti adalah algoritma kompresi *Huffman*, dan *Arithmetic Coding,* sedang untuk kategori *dictionary-based compression* yang paling banyak diteliti dan diimplementasikan adalah varian algoritma *LZ77* dan *LZ78* .

Untuk menciptakan suatu file pustaka *(library)* caranya adalah pertama, meng-enkapsulasi, yakni satu rutin metoda kompresi *lossless* tertentu (dalam satu file) dijadikan sebuah *class* tertentu. Lalu menggabungkan berbagai *class* dari algoritma kompresi tersebut dalam satu file pustaka dengan bantuan *project manager (file project).* File pustaka yang dibuat adalah *static library* dan *dynamic link library.*

Setelah *class library* terbentuk, langkah selanjutnya adalah membuat program aplikasi *toolbox* kompresi *lossless* dengan tulang punggung *library* tersebut. Dari program aplikasi ini dapat dilakukan penelitian kompresi tunggal dan kompresi bertingkat terhadap file-file sampel dengan semua algoritma kompresi yang ada dalam *library*. Mekanismenya adalah dengan melakukan proses kompresi dan

dekompresi terhadap berbagai macam tipe dan ukuran file yang umum digunakan, dalam hal ini : *bmp, cpp, doc, exe, xml, html,* dan *txt*. Lalu diamati dan dicatat besarnya rasio kompresi dan waktu kompresi (dan dekompresi) yang diperlukan oleh masing-masing algoritma kompresi. Pada penelitian teknik kompresi bertingkat, kombinasi algoritma yang dipakai adalah dipilih dari algoritma terbaik berdasarkan hasil pengamatan dari penelitian kompresi tunggal.

Untuk mengetahui apakah rutin-rutin kompresi dan dekompresi dari suatu algoritma kompresi tertentu yang sudah dibuat dan digunakan tersebut sudah benar atau belum maka harus dilakukan proses verifikasi. Proses verifikasi dilakukan dengan cara membandingkan *byte* per *byte* dari file asli (file sumber) dengan file hasil dekompresi. Jika ada ketidak sesuaian antara file asli dengan file hasil dekompresi, maka program akan memberikan pesan tidak valid. Hal ini berarti ada kesalahan dalam program entah itu pada proses kompresi atau pada proses dekompresinya.

Algoritma-algoritma kompresi *lossless* yang berhasil diimplementasikan dengan pendekatan konsep kelas dan dijadikan dalam satu file pustaka *(static/dynamic library)* adalah sebagai berikut: *Run Length Encoding* orde-0, 1, 2, dan 3 (class rle_n), *Static Huffman* orde-0 (class shuff), *Adaptive Huffman* orde-0 (class ahuff_0), *Adaptive Huffman* orde-1(class ahuff_n), *Static Arithmetic Coding* orde-0 (class s_Arith0), *Adaptive Arithmetic Coding* orde-0,1,2, dan 3 (class aarith_n), *LZSS* (class lzss0) dengan ukuran indek *dictionary* (jendela teks) berkisar antara 12-16 bits, dan panjang frase antara 4 – 8 bits sehingga *LZSS* seluruhnya ada 25 kombinasi model indek/panjang, *LZW* (class lzw0) dengan ukuran *dictionary* 9 – 16 bits. Jadi seluruhnya ada 45 alternatif pilihan algoritma kompresi *lossless* yang dapat digunakan untuk mengkompresi sembarang tipe file.

Hasil penelitian, dapat dianalisa bahwa secara umum algoritma yang berunjuk kerja rasio kompresi lebih tinggi, akan membawa konsekuensi waktu kompresi yang relatif lebih lama (kecepatan kompresi rendah). Dalam penelitian ini diperoleh informasi algoritma yang menghasilkan rata-rata rasio kompresi paling tinggi adalah *LZSS* 16/7, dengan kecepatan rata-rata 4.080,65 bps (nomor 3 dari bawah). Algoritma *LZSS* dapat dipastikan selalu ada dalam peringkat 10 besar untuk kompresi semua tipe file sampel dalam penelitian ini. Bahkan untuk rata-rata rasio kompresi dari keseluruhan kompresi terhadap seluruh file sampel, peringkat 20 besar didominasi oleh *LZSS* dalam berbagai kombinasi indek jendela dan panjang *LA*-nya.

Algoritma yang paling cepat waktu kompresi dan dekompresi adalah *RLE* orde-2, lalu *RLE* orde-3, dan *RLE* orde-0. Hal ini bisa dimengerti karena algoritma ini relatif lebih sederhana dalam proses komputasi. Namun konsekuensinya rata-rata rasio kompresi yang dihasilkan adalah paling rendah. Algoritma ini paling tepat jika digunakan untuk mengawali proses kompresi pada teknik kompresi bertingkat.

Hal lain yang perlu dicermati dari hasil penelitian ini adalah pada hasil kompresi file-file *bmp* dan *html*. Di sini algoritma *adaptive Arithmetic Coding* orde-3 dan orde-2 dapat menduduki peringkat pertama, bahkan ada sebagian hasil kompresi yang dapat mengungguli hasil dari algoritma *Winzip 8.0*. Sementara itu, untuk algoritma *LZW* menduduki urutan (mulai) ke-28 setelah ke 25 kombinasi

indek/panjang algoritma *LZSS*, lalu *adaptive Arithmetic Coding* orde-3 dan orde-2 dalam unjuk kerja rasio kompresi. Namun berdasarkan waktu kompresinya *LZW* masuk dalam kelompok 10 besar tercepat jauh di atas algoritma-algoritma *LZSS*.

Hasil pengamatan dalam penelitian teknik kompresi bertingkat, dapat diperoleh beberapa informasi antara lain, algoritma *RLE* relatif 'aman' untuk digunakan sebagai 'alat' kompresi awal pada teknik kompresi bertingkat, dalam arti tidak akan terjadi proses ekspansi pada tahap kompresi berikutnya. Kombinasi algoritma pada kompresi bertingkat tingkat-3 yang paling aman adalah *RLE-0 + LZSS12/8 + Adaptive Huffman orde-1*. Meskipun hasil yang diperoleh tidak begitu memuaskan, namun tidak akan terjadi proses ekspansi pada tahap kompresi yang terakhir.

Berdasarkan rasio kompresi yang dihasilkan, kombinasi algoritma kompresi *lossless* yang terbaik untuk mengkompres file *bmp (filebmp3.bmp)* adalah *LZW 16bits + Adaptive Huffman* orde-1 yakni 76,94%, untuk *file cpp (filecpp2.cpp)* adalah : *LZSS 16/8 + Adaptive Huffman* orde-1 yakni 91,27%, untuk file *doc (filedoc3.doc)* adalah: *RLE orde-0+LZSS 16/7+Adaptif Huffman* orde-0 yakni 76,91%, untuk file *exe (fileexe3.exe)* adalah: *RLE* orde-0 + *LZSS* 12/8 + *Adaptive Huffman* orde-1 yakni 51,73%, untuk file *htm (html4.htm)* adalah: *RLE* orde-0+*LZSS* 16/7+*Adaptive Huffman* orde-0 yakni 84,22%, untuk file teks *(teks2.txt)* adalah: *RLE* orde-0 + *Adaptive Arithmetic Coding* orde-3 yakni 89,98%, dan untuk file tipe *xml (filexml4.xml)* kombinasi yang terbaik adalah: *LZSS* 16/8 + *Adaptive Huffman* orde-1 yakni 95,49%.

Ketujuh tipe file sampel yang digunakan, hanya 2 tipe file saja (*exe* dan *html*) yang menghasilkan rasio kompresi tertinggi lebih rendah dibandingkan dengan hasil terbaik pada teknik kompresi tunggal. Sedangkan untuk 4 tipe file yang lain dapat dicapai rasio kompresi yang lebih tinggi dari pada hasil terbaik dari teknik kompresi tunggal. Jadi secara keseluruhan teknik kompresi bertingkat ini dapat dikatakan berhasil dalam meningkatkan unjuk kerja terutama pada peningkatan rasio kompresi.

Kesimpulan dari seluruh penelitian dalam riset ini adalah bahwa pengembangan pustaka algoritma kompresi *lossless* baik yang statik *(static library)* ataupun yang dinamis *(dynamic link library)* dapat dilakukan dengan pendekatan konsep *class*. Lalu, pengembangan sebuah program aplikasi *toolbox* kompresi data *lossless* dapat dilakukan, dan relatif akan lebih efisien dan mudah apabila memanfaatkan *class library* dari algoritma kompresi *lossless* tersebut.

Akhirnya, dari hasil penelitian ini diharapkan dapat bermanfaat dan memperluas khasanah kegiatan penelitian dan pemrograman, khususnya pada penelitian dan pemrograman tentang kompresi data.