# IMPLEMENTASI ALGORITMA KOMPRESI LZ77 PADA *SMARTPHONE BLACKBERRY*

Tommy Gunardi / 13507109[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]tommy_gunardi@hotmail.com*

***Abstract*—Dewasa ini, seluruh aspek teknologi berkembang dengan sangat pesat. Hal ini juga dipicu oleh banyaknya *smartphone* seperti *BlackBerry* yang terus dikeluarkan. Kegunaan dari *smartphone* ini tidak hanya untuk menelpon atau sms saja, namun digunakan sebagai *email*, media hiburan (gambar, video, lagu), *games*, *chatting*, *social network* dan lain-lain. Tentu dengan kegunaannya yang semakin beragam dan lengkap, *smartphone* seperti *BlackBerry* membutuhkan *storage* atau tempat penyimpanan data yang cukup besar. Namun, aplikasi khusus untuk mengompresi/mendekompresi *file* pada *storage BlackBerry* belum ada.**

**Oleh karena itu, pada Tugas Akhir ini dibuatlah suatu aplikasi untuk mengompresi dan mendekompresi data pada *storage BlackBerry*. Aplikasi ini bernama *BB-Compress*. Aplikasi ini dibuat sebagai suatu aplikasi khusus yang bertujuan mengompresi dan mendekompresi berbagai macam *file* yang ada pada *storage BlackBerry* dengan menggunakan algoritma kamus *LZ77*.**

***BB-Compress* diimplementasikan pada *BlackBerry Bold* 9000. *BlackBerry* memiliki kemampuan perangkat keras dan lunak yang cukup memadai untuk melakukan proses kompresi dan dekompresi. *BB-Compress* ini diimplementasikan dengan menggunakan kakas *Eclipse Ganymede* dan *Netbeans*.**

***BB-Compress* berhasil mengimplementasikan algoritma *LZ77* pada *smartphone BlackBerry* dan dapat melakukan fungsinya untuk mengompresi dan mendekompresi berbagai macam *file* pada *BlackBerry*. Untuk beberapa *file*, aplikasi ini menunjukkan hasil yang sangat baik dalam kompresinya. Namun waktu kompresi pada *BB-Compress* menjadi masalah karena proses kompresi berjalan cukup lambat.**

**Kata kunci: *BB-Compress*, *LZ77*, *BlackBerry*, kompresi dan dekompresi**

## I. PENDAHULUAN

Dewasa ini, seluruh aspek teknologi berkembang dengan sangat pesat. Hal ini juga dipicu oleh banyaknya alat komunikasi *mobile* seperti *BlackBerry* dan *iPhone* tipe baru yang terus dikeluarkan. Alat komunikasi tersebut saat ini sudah menjadi *smartphone*, yaitu sebuah *mobile phone* yang menawarkan kemampuan komputasi dan konektivitas lebih dari sekedar basic *phone* kontemporer. *Smartphone* mungkin dianggap sebagai komputer genggam yang terintegrasi dalam telepon seluler, sementara kebanyakan *mobile phone* menjalankan aplikasi berbasis *platform Java ME*, *smartphone* memungkinkan pengguna untuk meng*instal* dan menjalankan aplikasi lebih maju berdasarkan *platform* yang lebih spesifik. *Smartphone* menjalankan suatu sistem operasi lengkap dan menyediakan *platform* untuk pengembang aplikasi. Jadi saat ini kita dapat mempertimbangkan sebuah *smartphone* sebagai *Pocket Personal Computer* (PPC) dengan fungsi ponsel, karena sesungguhnya *device-device* ini adalah komputer, meskipun dengan ukuran lebih kecil daripada *desktop* komputer.

Kegunaan dari *smartphone* ini tidak hanya untuk menelpon atau sms saja, namun digunakan sebagai *email*, media hiburan (gambar, video, lagu), *games*, *chatting*, *social network* dan lain-lain. Tentu dengan kegunaannya yang semakin beragam dan lengkap, *smartphone* seperti *BlackBerry* dan i*Phone* membutuhkan *storage* atau tempat penyimpanan data yang cukup besar. Hal ini dikarenakan volume data semakin lama semakin besar. Padahal, tempat penyimpanan data pada *smartphone-smartphone* tersebut sangat terbatas. Maka diperlukan suatu metoda untuk memanipulasi data-data tersebut supaya bisa menghemat tempat penyimpanan. Salah satu bentuk manipulasi itu adalah dengan mengompresi data sehingga ukurannya menjadi lebih kecil dari ukuran *file* asli.

Pada tugas akhir ini, *smartphone* yang dipilih adalah *BlackBerry*. Mengapa *BlackBerry*? Karena jumlah pemakai *BlackBerry* sudah sangat banyak di Indonesia. Selain kebutuhan utamanya untuk *email*, *BlackBerry* juga memiliki fitur *chatting* antar pengguna *BlackBerry* yang bernama *BlackBerry* Messenger. *BlackBerry* merupakan suatu *smartphone* yang dirancang dan didesain oleh perusahaan *Research In Motion (RIM)* sejak 1996. Keunggulan *BlackBerry* ada pada sistem *email*nya yang sangat mendukung. Selain itu fitur Blackberry untuk mengunduh dan mengunggah berbagai jenis file melalui internet. Semakin banyak data-data tersebut maka semakin besarlah ukuran *storage memory* yang dibutuhkan *BlackBerry*. Kapasitas storage pada *BlackBerry* untuk penyimpanan *file* sangat terbatas. Untuk itu kompresi data pada *BlackBerry* sangat diperlukan.

Kompresi data adalah suatu proses perubahan suatu *input* data sumber menjadi suatu data *stream* lain dengan ukuran lebih kecil. Apabila kita melakukan proses kompresi data terhadap suatu *file*, maka kita harus dapat melakukan dekompresi terhadap *file* tersebut. Metoda kompresi dan dekompresi data memiliki dua tipe yakni *lossless* dan *lossy* compression. Algoritma kompresi *lossless* dan *lossy* adalah istilah yang menggambarkan apakah setelah mengalami proses kompresi data, data hasil dekompresi dapat dikembalikan seperti data asli sebelum kompresi. Dengan kompresi *lossless*, setiap bit data yang pada awalnya ada dalam *file* akan tetap ada setelah data selesai didekompresi. Seluruh informasi akan dikembalikan seluruhnya. Berlawanan dengan kompresi *lossless*, algoritma kompresi *lossy* menghilangkan informasi tertentu secara permanen, terutama informasi-informasi yang dianggap berlebihan. Bila *file* terkompresi, hanya sebagian dari informasi *file* yang masih ada, meskipun ini biasanya tidak disadari *user*.

Algoritma kompresi *LZ77* adalah suatu metoda kompresi data *lossless* yang akan dibuat aplikasinya pada pengerjaan tugas akhir ini. Aplikasi ini bertujuan untuk melakukan pengompresian dan pendekompresian data secara khusus pada *BlackBerry*. Algoritma *LZ77* diperkenalkan untuk pertama kalinya pada tahun 1977 oleh Abraham Lempel dan Jacob Ziv. Algoritma *LZ77* merupakan suatu algoritma dasar yang banyak dikembangkan oleh orang-orang. Misalnya saja *LZW*, *LZSS*, dan *LZMA*. Algoritma *LZ77* sendiri sering disebut dengan *LZ1* (Salomon, 2006). Algoritma ini juga disebut algoritma "*sliding window*" karena melakukan kompresi data dengan cara menggerakkan *buffer* tempat simbol-simbol berada setiap kali satu atau lebih simbol terkompresi. *Buffer* yang dibutuhkan algoritma kompresi ini ada dua, yaitu *search buffer* dan *look-ahead buffer*. Kedua *buffer* ini memegang peran penting dalam melakukan kompresi terhadap suatu *file*. Alasan pemilihan algoritma ini adalah bahwa algoritma ini memberikan performa terbaik untuk beberapa jenis file dibandingkan algoritma-algoritma modifikasinya (Zeeh, 2003).

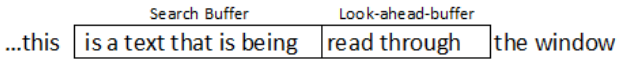
Menurut beberapa aplikasi terkait kompresi data pada *BlackBerry* (AppWorld, 2011), kebanyakan aplikasi tersebut menambahkan kompresi dan dekompresi sebagai fungsi tambahan dari fungsi utamanya. Aplikasi lainnya hanya dapat melakukan dekompresi terhadap suatu file yang terkompresi. Pihak *RIM* (Re*search in Motion*) yakni perusahaan yang mendevelop *BlackBerry* di Kanada, lebih mengkhususkan kompresi data terhadap data-data yang ada di internet. Hal ini dikarenakan *wireless* data merupakan prioritas RIM. Jumlah pemakaian *wireless* data antara semua alat komunikasi *mobile* naik sebesar 463 persen antara akhir tahun 2009 sampai pertengahan tahun 2010 (Finocchiaro 2010). Pengguna *BlackBerry* mengonsumsi sekitar 79.5 *megabyte* data per bulan untuk setiap *user*, dibandingkan dengan 375 *megabyte* untuk *iPhone user*. Namun karena terlalu mengkhususkan kepada *wireless* data yang diunduh dan diunggah, data yang ada di *storage* tidak dikompresi.

## II. ALGORITMA YANG DIGUNAKAN

Metoda ini dikenalkan sebagai suatu algoritma kompresi data lossless yang dikeluarkan sebagai paper oleh Abraham Lempel dan Jacob Ziv pada tahun 1977. Algoritma ini juga dikenal sebagai algoritma LZ1. Algoritma ini menjadi dasar dari beberapa algoritma lainnya seperti LZW, LZSS, LZMA dan lainnya.

Metoda ini merupakan suatu metoda kamus (dictionary method) yang tidak menggunakan model statistik, dan tidak menggunakan kode berukuran variabel (Salomon 2006). Metoda ini merupakan suatu algoritma kompresi data yang beroperasi dengan mencari kesamaan antara data untuk dikompresi dan suatu struktur data (kamus) yang berisi sejumlah string yang diatur oleh encoder. Ketika encoder menemukan kesamaan tersebut, ia mengganti referensi terhadap suatu posisi string yang terdapat pada struktur data.

Algoritma kompresi LZ77 melakukan kompresinya dengan mengganti porsi data dengan suatu referensi untuk menyamakan data yang sudah dilewati oleh encoder dan decoder. Algoritma ini menggunakan suatu "sliding window" (Muller 2008) yang terdiri dari search buffer dan look-ahead buffer. Search buffer digunakan sebagai kamus, sedangkan look-ahead buffer merupakan buffer yang berisi string yang akan dikompresi. Search buffer merupakan buffer yang sudah melalui tahap kompresi, namun digunakan sebagai kamus. Kedua buffer tersebut dapat dilihat pada contoh berikut :




**Gambar 1 Search buffer dan Look-ahead buffer (Salomon 2006)**

Metode ini melakukan pencarian satu (atau lebih) simbol pada look-ahead buffer yang sama dengan simbol pada search buffer. Pencarian dilakukan dari kanan ke kiri. Prakteknya, perbandingan ukuran search buffer dengan look-ahead buffer kira-kira adalah beberapa ribu banding 10.

Algoritma ini melakukan pencarian terhadap kumpulan simbol terpanjang yang sama pada awal look-ahead buffer terhadap search buffer dan mengeluarkan sebuah pointer terhadap kesamaan tersebut. Secara umum, sebuah token LZ77 memiliki tiga buah bagian yakni : offset, panjang, dan simbol selanjutnya pada look-ahead buffer. Offset dan panjang merupakan pointer yang menunjukkan posisi dan jumlah kecocokan. Apabila ditemukan kecocokan, LZ77 menambahkan suatu pointer dengan suatu simbol di belakangnya. Apabila tidak ada yang cocok, maka dihasilkan null pointer dan simbol unik. Selanjutnya sliding window digeser sebanyak simbol yang terkompresi berdasarkan pointernya.

Berikut ilustrasi metoda kompresi LZ77 :

Misalkan kata yang ingin dikompresi adalah “abracadabrad”

Encoding String :
"abracadabrad"

output tuple : (offset, length, symbol)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | a | b | r | a | c | ada... | (0,0,a) |
| | | | | | | | a | b | r | a | c | a | dab... | (0,0,b) |
| | | | | | | a | b | r | a | c | a | d | abr... | (0,0,r) |
| | | | | | a | b | r | a | c | a | d | a | bra... | (3,1,c) |
| | | | a | b | r | a | c | a | d | a | b | r | ad | (2,1,d) |
| | a | b | r | a | c | a | d | a | b | r | a | d | | (7,4,d) |
| ...ac | a | d | a | b | r | a | d | | | | | | | |

Search Buffer | Look-ahead buffer

**Gambar 2 Tabel Simulasi LZ77 terhadap string "abracadabrad" (Muller 2008)**

Tabel di atas merupakan simulasi terhadap kompresi untuk string “abracadabrad” dengan ukuran search buffer 7 dan look ahead buffer 5. Berikut langkah-langkahnya:

1. Pertama-tama *search buffer* kosong sedangkan *look-ahead buffer* berisi 5 simbol yaitu “abrac”. Simbol ‘a’ berada pada bagian paling kiri *look ahead buffer*. Karena ‘a’ unik, maka dihasilkan *output* (0,0,a).
2. *Sliding window* digeser ke kanan sebanyak 1, simbol a berada di *search buffer*. Kumpulan simbol “braca” berada di *look-ahead buffer*. Simbol ‘b’ berada pada bagian paling kiri *look ahead buffer*. Karena ‘b’ unik, maka dihasilkan *output* (0,0,b).
3. *Sliding window* digeser ke kanan sebanyak 1, kumpulan simbol “ab” berada di *search buffer*. Kumpulan simbol “racad” berada di *look-ahead buffer*. Simbol ‘r’ berada pada bagian paling kiri *look ahead buffer*. Karena ‘r’ unik, maka dihasilkan *output* (0,0,r).
4. *Sliding window* digeser ke kanan sebanyak 1, kumpulan simbol “abr” berada di *search buffer*. Kumpulan simbol “acada” berada di *look-ahead buffer*. Simbol ‘a’ berada pada bagian paling kiri *look ahead buffer* dan ada di dalam *search buffer*. Ada satu simbol ‘a’ pada *search buffer* dengan *offset* 3 sepanjang 1 dan simbol setelahnya adalah ‘c’. maka dihasilkan *output* (3,1,c).
5. *Sliding window* digeser ke kanan sebanyak 2, kumpulan simbol “abrac” berada di *search buffer*. Kumpulan simbol “adabr” berada di *look-ahead buffer*. Simbol ‘a’ berada pada bagian paling kiri *look ahead buffer* dan ada di dalam *search buffer*. Ada dua simbol ‘a’ pada *search buffer* yakni pada *offset* 2 dengan panjang 1 dan *offset* 5 dengan panjang 1. Karena panjang keduanya sama, diambil *offset* 2 dengan panjang 1 dan simbol setelahnya adalah ‘d’. maka dihasilkan *output* (2,1,d).
6. *Sliding window* digeser ke kanan sebanyak 2, kumpulan simbol “abracad” berada di *search buffer*. Kumpulan simbol “abrad” berada di *look-ahead buffer*. Simbol ‘a’ berada pada bagian paling kiri *look ahead buffer* dan ada di dalam *search buffer*. Ada tiga simbol ‘a’ pada *search buffer* yakni pada *offset* 2 dengan panjang 1, *offset* 4 dengan panjang 1, dan *offset* 7 dengan panjang 4. Maka dipilih *offset* 7 dengan panjang 4 dan simbol setelahnya adalah ‘d’. maka dihasilkan *output* (7,4,d).
7. Ukuran *file* awal adalah 12 * 8 bit = 96 bit. Perhitungan hasil kompresi menghasilkan 6 buah tuple yaitu (0,0,a), (0,0,b), (0,0,r), (3,1,c), (2,1,d), (7,4,d). Offset dan panjang yang bernilai 0 tidak perlu dimasukkan. Karena panjang search buffer adalah 7 byte, maka ukuran untuk offset dan panjang adalah masing-masing 3 bit. Untuk membedakan bit mana yang menunjukkan posisi dan panjang dengan simbol, diberikan tanda dengan menambahkan 1 bit. Bit 0 menunjukkan bahwa 8 bit selanjutnya dari program adalah simbol, sedangkan bit 1 menunjukkan bahwa 6 bit selanjutnya berisi panjang dan offset simbol yang diikuti 8 bit isi simbolnya. Maka ditemukanlah hasil kompresinya sebagai berikut :

   (0,0,a), (0,0,b), (0,0,r), (3,1,c), (2,1,d), (7,4,d)

   (1+8) + (1+8) + (1+8) + (1+6+8) + (1+6+8) + (1+6+8)

   9 + 9 + 9 + 15 + 15 + 15 = 72 bit
8. Maka kita dapat menghitung performa kompresinya sebagai berikut :
   a. Rasio Kompresi
      Nilai rasio kompresi adalah ukuran output / input : 72/96 = 75%
   b. Faktor Kompresi
      Nilai faktor kompresi adalah ukuran input / output : 96/72 = 133%
   c. Entropi = rdrd
      Untuk menghitung entropi kita harus mengetahui probabilitas kemunculan masing-simbol : a = 5, b = 2, c = 1, d = 2, r = 2. Lalu dihitung dengan rumus entropi menghasilkan entropi = -17.6094

Proses dekompresi algoritma *LZ77* sangat cepat karena tidak perlu melakukan proses pencarian terhadap kesamaan di dalam *buffer*. Berikut cara dekompresi dengan menggunakan tabel simbol kompresi:

| input | | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|
| (0,0,a) | | | | | | | | a |
| (0,0,b) | | | | | | | a | b |
| (0,0,r) | | | | | | a | b | r |
| (3,1,c) | | | | a | b | r | a | c |
| (2,1,d) | | a | b | r | a | c | a | d |
| (7,4,d) | abrac | a | d | a | b | r | a | d |

**Gambar 1 Tabel dekompresi *LZ77* (Muller, 2008)**

Langkah-langkah dekompresi :

1. Dimulai dengan (0,0,a), *string* yang ada adalah "a".
2. Selanjutnya menambahkan simbol (0,0,b) ke akhir *string* sehingga *string* yang dihasilkan adalah "ab".
3. Selanjutnya menambahkan simbol (0,0,r) ke akhir *string* sehingga *string* yang dihasilkan adalah "abr".
4. Melakukan pencarian terhadap simbol (3,1,c) dengan bergerak 3 langkah dari kanan ke kiri sehingga ditemukan simbol 'a' sepanjang 1. Selanjutnya menambahkan simbol 'c' ke simbol 'a'. Kumpulan simbol 'ac' ditambahkan ke akhir *string* sehingga *string* yang dihasilkan adalah "abrac".
5. Melakukan pencarian terhadap simbol (2,1,d) dengan bergerak 2 langkah dari kanan ke kiri sehingga ditemukan simbol 'a' sepanjang 1. Selanjutnya menambahkan simbol 'd' ke simbol 'a'. Kumpulan simbol 'ad' ditambahkan ke akhir *string* sehingga *string* yang dihasilkan adalah "abracad".

Melakukan pencarian terhadap simbol (7,4,d) dengan bergerak 7 langkah dari kanan ke kiri sehingga ditemukan kumpulan simbol 'abra' sepanjang 4. Selanjutnya menambahkan simbol 'd' ke kumpulan simbol 'abra'. Kumpulan simbol 'abrad' ditambahkan ke akhir *string* sehingga *string* yang dihasilkan adalah "abracadabrad".

## III. ANALISIS

### III.1 Analisis Existing System

Pada bab ini akan dijelaskan mengenai analisis existing system, yakni sistem dimana aplikasi akan dikembangkan, yaitu deskripsi *BlackBerry Bold* 9000. Hal-hal yang akan dijelaskan mengenai struktur *file* dan *encoding* pada *BlackBerry*, kapasitas dan kompresi data, dan kapasitas processor dan *memory*.

#### III.1.1 Struktur *File* dan *Encoding BlackBerry*

*Storage* pada *BlackBerry* terbagi ke dalam dua bagian yakni *device memory* yang merupakan *memory* standar yang sudah tersedia pada *BlackBerry* dan media card. Tipe media card yang digunakan oleh *BlackBerry* adalah microSD card (Secure Digital).
Struktur *file* pada *BlackBerry* berupa folder yang di dalamnya dapat berisi berbagai macam *file* dan folder lain. Bagian terluar dari struktur *file BlackBerry* adalah *device memory* dan media card. Struktur *file* dari *device memory* adalah sebagai berikut :

1. *appdata* (*hidden*)
   folder *appdata* berisi folder rim yang berisi folder bda dan media.
2. *applications* (*hidden*)
   berisi data-data aplikasi yang ada pada *BlackBerry*.
3. *dev* (*hidden*)
   folder developer *BlackBerry*
4. *system* (*hidden*)
   folder system berisi folder accessibility dan fonts.
5. *tmp* (*hidden*)
   folder temporary *BlackBerry* berisi data-data sementara *device BlackBerry*.
6. *BlackBerry*
   Folder *BlackBerry* berisi data aplikasi pihak ketiga yang memiliki ekstensi '.cod.rem' atau '.app.rem'
7. *home*
   *Folder home* berisi data pribadi milik pengguna seperti dokumen, lagu, video dan gambar. Di direktori ini tempat *file* yang nantinya akan dikompresi dan didekompresi.

Pada media card, hanya terdapat *folder BlackBerry* yang berisi data-data pribadi milik pengguna. *File-file* yang berada di media card juga dapat dikompresi dan didekompresi.

IANA (*Internet Assigned Numbers Authority*) (IANA, 2011) adalah suatu organisasi yang bertanggungjawab terhadap alokasi dari nama dan angka yang digunakan dalam internet protocols yang dipublikasikan oleh dokumen RFC (*Request for Comments*). IANA mengelola banyak parameter dalam protokol internet. Beberapa contohnya adalah nama skema *encoding* karakter yang direkomendasikan untuk digunakan pada internet.

*Encoding* adalah pemetaan karakter-karakter dalam format tertentu yang bertujuan untuk mengefisienkan transmisi atau *storage*. Beberapa karakter *encoding* yang disuport oleh *BlackBerry* antara lain :

1. "ISO-8859-1"
   ISO-8859-1 merupakan *encoding* default *BlackBerry*. ISO 8859-1 mengencode "Latin alphabet no.1" yang berisi 191 karakter dari skrip latin, masing-masing dikodekan dengan sebuah nilai kode berukuran 8 bit (Wordiq, 2011). *Encoding* ini digunakan di hampir semua bahasa Eropa Barat.
2. *UTF-8*
   *UTF-8* (*UCS Transformation Format-8bit*), dikatakan *backward compatible* dengan ASCII karena dapat membaca standar input *ASCII*. *Encoding* ini digunakan karena *BlackBerry* merupakan *device smartphone* yang mendukung fasilitas email dan internet. UTF-8 merupakan *encoding* dominan untuk *world-wide web* yang digunakan lebih dari setengah total web yang ada.
3. "UTF-16BE"
   UTF-16BE (Unicode Transformation Format – 16bit Big Endian) merupakan *encoding* karakter yang memetakan kode poin dari karakter set unicode ke rangkaian 2 *byte*s. *Encoding* ini digunakan pada internet protocol dan beberapa bahasa pemrograman bahkan aplikasi-aplikasi

software, namun bukan merupakan *encoding* standar pada internet protocol.

4. "US-ASCII"
   American Standard Code for Information Interchange meruoajan skema *encoding* karakter berbasis urutan dari alfabet bahasa inggris. Kode ASCII merepresentasikan teks dalam komputerm peralatan komunikasi dan *device* lain yang menggunakan teks. Kebanyakan skema *encoding* karakter modern berbasis ASCII, meskipun kadang mereka mensupport lebih banyak karakter dibandingkan ASCII.

Banyaknya *encoding* yang dapat disupport *BlackBerry* tidak membuat proses kompresi menjadi merepotkan karena format string pada Java adalah Unicode. Oleh karena itu tidak perlu dilakukan konversi terhadap data dengan menggunakan *encoding* seperti ISO-8859-1 atau UTF-8 (Strange, 2010).

### III.1.2 Kapasitas dan Kompresi Data

Seperti yang sudah dijelaskan pada subbab sebelumnya, *BlackBerry* memiliki dua macam *storage* yakni *Device Memory* dan Media Card. *Bold* 9000 memiliki kapasitas *device memory* sebesar 1 GB. Segala macam *file-file* sistem operasi *BlackBerry* disimpan pada *device memory*. Media card yang dapat digunakan *BlackBerry* sangat beragam. Besarnya kapasitas *microSD card* yang dapat digunakan *device BlackBerry* bergantung pada versi perangkat lunak sistem operasi pada *BlackBerry*. Dengan adanya kompresi data pada *storage*, penggunaaan memori tentu dapat dioptimalkan secara maksimal. Berikut data kapasitas maksimum microSD card yang bisa ditampung *BlackBerry* :

1. *BlackBerry Device Software* 4.2.0 - 2 GB ke bawah
2. *BlackBerry Device Software* 4.2.1 - 4 GB ke bawah
3. *BlackBerry Device Software* 4.2.2 - 4 GB ke bawah
4. *BlackBerry Device Software* 4.3.0 - 8 GB ke bawah
5. *BlackBerry Device Software* 4.5.0 - 8 GB ke bawah
6. *BlackBerry Device Software* 4.5.0.81 - 16 GB ke bawah
7. *BlackBerry Device Software* 4.6.0 dan seterusnya - 32 GB ke bawah

Kapasitas maksimum *microSD* yang bisa ditampung *BlackBerry* adalah 32 *GigaByte* dengan dukungan operating system 4.6.0 ke atas. Namun harga dari microSD card ini cukup mahal yakni sekitar $69.71 (data didapat dari amazon.com). Pada kasus *BlackBerry Bold* 9000, software sistem operasi yang bisa digunakan adalah sampai *Device* Software 5.0. Oleh karena itu, *BlackBerry Bold* dapat menggunakan microSD card sebesar 32 Giga*Byte*.

### III.1.3 Kapasitas Processor dan *Memory*

Processor pada *BlackBerry* adalah Intel Xscale 624MHz CPU. Termasuk salah satu yang terbaik di kelasnya saat itu. *BlackBerry Bold* dapat melakukan 624.000.000 instruksi per detik. *Memory* pada *BlackBerry Bold* 9000 adalah 128 MB.

## III.2 Kesesuaian Algoritma LZ77 pada device BlackBerry

Pengimplementasian Algoritma *LZ77* pada *BlackBerry* dengan memperhitungkan *encoding* default *BlackBerry* (ISO-8859-1), yakni per 8 bit atau 1 *byte*, sehingga pengompresian dilakukan per *byte*. Pengompresian dilakukan dalam beberapa tahap, tergantung besarnya *file*. Algoritma dilakukan per tahap supaya tidak boros memori.

### III.2.1 Kompresi

Berikut Pseudocode standar kompresi *LZ77* :

```
while( lookAheadBuffer not empty )
          {
          get pointer ( position, match ) to the
longest match in the window
               for the lookahead buffer;

          if( length > MINIMUM_MATCH_LENGTH )
               {
               output a ( position, length )
pair;
               shift the window length characters
along;
               }
          else
               {
               output the first character in the
lookahead buffer;
               shift the window 1 character
along;
               }
          }
```

Pertama-tama program membaca *file* sebesar y, lalu dimasukkan ke dalam suatu array of *byte* dengan ukuran x. Apabila ukuran y lebih besar dari x maka isi *file* yang diambil hanya sebesar x terlebih dahulu. Sebelumnya *file* output diberi header yang menunjukkan bahwa *file* ini benar telah terkompresi oleh algoritma *LZ77*. Pengompresian dilakukan terhadap buffer tersebut. Bagian yang disimpan adalah isi dari *byte* tersebut dan posisi *byte*. Setelah seluruh isi buffer tersebut terkompresi, isi buffer dimasukkan ke *file* output. Buffer lalu diisi kembali sisa dari *file* dengan ukuran y-x (karena sudah diambil sebanyak x pada kompresi sebelumnya) sebanyak x lalu dilakukan kompresi terhadap buffer tersebut. Hal ini dilakukan terus-menerus sampai *file* dengan ukuran y sudah selesai terkompresi seluruhnya.

### III.2.2 Dekompresi

Pertama-tama program membaca header *file* apakah benar *file* ini terkompresi oleh program kompresi *LZ77*.

Lalu program membaca isi *file* terkompresi dengan format yang diletakkannya adalah isi, posisi, dan panjang *byte*(-*byte*) tersebut. Setelah terbaca, program mengekstraksi *file* seperti keadaan semula.

## IV. PERANCANGAN

### IV.1 Deskripsi Umum Perangkat Lunak

Perangkat lunak yang akan diimplementasikan pada tugas akhir ini adalah suatu implementasi algoritma *LZ77* pada *smartphone BlackBerry* yang diberi nama *BB-Compress*. Program *BB-Compress* dapat mengompresi *file-file* yang ada di *storage BlackBerry* dan mendekompresi *file* yang telah terkompresi kembali menjadi *file* aslinya. Karena algoritma yang digunakan di sini adalah *LZ77* yakni suatu algoritma kompresi lossless, maka *file* yang terkompresi harus dikembalikan seperti keadaan semula.

Program dapat menerima inputan *file* teks dan binary. *File* yang terkompresi akan memiliki format tertentu. Hanya pengguna yang sudah menginstal aplikasi yang dapat menggunakan aplikasi untuk mengompresi dan mendekompresi data. *File* yang telah terkompresi hanya dapat didekompresi dengan menggunakan *BB-Compress*.

### IV.2 Analisis Kebutuhan

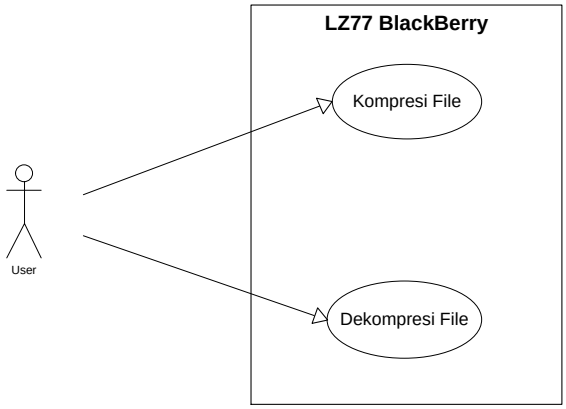
Setelah menentukan deskripsi umum perangkat lunak yang akan dibuat, selanjutnya kita dapat menentukan kebutuhan fungsional yang mampu dilakukan oleh aplikasi, antara lain :

1. Mengompresi *file*
   Program dapat melakukan kompresi *LZ77* terhadap semua jenis *file* yang berada di *storage BlackBerry* baik di *device memory* ataupun media card.
2. Mendekompresi *file*
   Program dapat melakukan dekompresi terhadap semua jenis *file* terkompresi *LZ77* yang berada di *storage BlackBerry* baik di *device memory* ataupun media card.
3. Menyimpan *file* kompresi
   Program dapat menyimpan hasil *file* yang sudah dikompresi.
4. Menyimpan *file* dekompresi
   Program dapat menyimpan hasil *file* yang sudah didekompresi.
5. Memilih *file*
   Pengguna dapat memilih *file* yang ada di direktori untuk dikompresi ataupun didekompresi.

Pengguna yang berinteraksi dengan perangkat lunak ini dapat dijalankan pada *smartphone BlackBerry* dengan sistem operasi 4.60 ke atas sebagai kebutuhan utama perangkat keras. Pengguna dapat berinteraksi dengan aplikasi dengan menggunakan Track ball. Input ini digunakan sebagai masukan perintah yang ada pada aplikasi seperti button. Dapat memilih suatu *file* tertentu pada direktori.

### IV.2 Analisis Use Case

Diagram Use Case pada tugas akhir ini mengandung dua buah komponen utama yakni Kompresi *File* dan Dekompresi *File*. Skema use case yang ada adalah sebagai berikut:




**Gambar 4 Use case *BB-Compress***

Pada diagram use case ini dapat dilihat dua komponen utama yang dapat dilakukan oleh program *BB-Compress*. Kedua komponen ini adalah

1. Kompresi *file*

Program dapat melakukan proses kompresi *LZ77* terhadap *file-file* yang ada pada *smartphone BlackBerry*. Use case ini termasuk membuka *file*, mengambil data sebanyak ukuran tertentu untuk dikompresi (pengulangan sampai semua data awal terbaca), mengompresi, menghitung rasio kompresi yang dihasilkan dan menghasilkan *file* kompresi.

2. Dekompresi *file*.

Program dapat melakukan proses dekompresi *LZ77* terhadap *file-file* yang sudah terkompresi pada *smartphone BlackBerry* kembali ke bentuk semula. Proses-proses yang termasuk dalam use case ini antara lain membuka *file* terkompresi, membaca hasil terkompresi, melakukan dekompresi, menyimpan *file* hasil dekompresi.

## V. IMPLEMENTASI

### V.1 Lingkungan Implementasi

Implementasi Aplikasi *BB-Compress* mencakup dua macam lingkungan implementasi yakni lingkungan implementasi peranti keras dan peranti lunak. Lingkungan implementasi peranti keras adalah sebagai berikut :

***BlackBerry Bold* 9000**

- Display Type 65k colors
- Display Monitor Size 480 x 320 pixels, 2.6 inches

- Memory Internal 1 GB *storage*, 128 MB RAM
- Memory Card microSD 1 GB
- CPU Intel Xscale 624MHz
- OS v5.0.0.822 (Bundle 1385, Platform 5.2.0.76)

Lingkungan implementasi peranti lunaknya menggunakan bahasa pemrograman Java for *BlackBerry*. Beberapa kakas pemrograman dan peranti lunak lain yang digunakan adalah sebagai berikut :
- *Netbeans 6.8*
- *Eclipse Ganymede*
- *BlackBerry Simulator*
- *Adobe Photoshop*

### V.2 Batasan Implementasi

Berikut merupakan batasan implementasi yang ada pada Tugas Akhir ini :
1. Tidak menangani kompresi folder
2. Kompresi/dekompresi dilakukan hanya jika tempat untuk menampung hasil kompresi/dekompresi tersedia (lebih atau sama dengan hasil).
3. Diimplementasikan pada *BlackBerry Bold* 9000
4. Hasil Kompresi/dekompresi diletakkan pada direktori *file* yang sama dengan sumber.

### V.3 Implementasi Kelas

Implementasi kelas tidak berbeda jauh dari hasil perancangan kelas. Berikut Diagram kelas yang ada dapat dilihat pada tabel V-1:

**Tabel 1 Implementasi Kelas**

| Kelas Rancangan | Implementasi |
|---|---|
| **BitIO** | `BitIO.java` |
| **LZ77Compress** | `LZ77Compress.java` |
| **LZ77Decompress** | `LZ77Decompress.java` |

Penjelasan dari fungsi-fungsi pada diagram kelas tersebut antara lain :

1. `BitIO.java`
   a. `void append(BitSet mainbit, int pos, BitSet addbit, int addbitlength)`
      Prosedur untuk menambahkan BitSet addbit ke BitSet mainbit
   b. `byte[] bitToByteArray(BitSet bitset, int outl)`
      Fungsi untuk mengonversi BitSet menjadi array of byte
   c. `BitSet byteToBitset(byte b)`
      Fungsi untuk mengonversi byte ke BitSet
   d. `BitSet byteArrayToBitset(byte[] byt)`
      Fungsi untuk mengonversi array of byte ke BitSet
   e. `BitSet intToBitset(int b, int length)`
      Fungsi untuk mengonversi integer ke BitSet
   f. `int bitToInt(BitSet bitset, int length)`
      Fungsi untuk mengonversi BitSet ke Integer
   g. `BitSet getBitSet(BitSet main, int start, int length)`
      Fungsi untuk mengambil nilai BitSet
   h. `OffPos Search(byte[] isi)`
      Fungsi pencarian kesamaan pola byte

2. `LZ77Compress.java`
   a. `void Compress(String in, String out)`
      Prosedur untuk mengompresi sebuah *file* in dan menuliskan hasil kompresinya pada out
   b. `byte[] CompressOut(String in)`
      Fungsi untuk mengompresi *file* in dan mengeluarkan output berupa array byte hasil kompresi
   c. `void CompressAll(String[] arr`*`File`*`)`
      Fungsi untuk mengompresi semua *file* yang ada pada array String arr*file*
   d. `void Trans(byte[] mainbyte, byte[] addit)`
      Prosedur untuk menambahkan array byte addit pada akhir array byte mainbyte
   e. `byte[] MoveArray(byte[] b, int sequence)`
      Fungsi untuk mengambil sebagian array byte b

3. `LZ77Decompress.java`
   a. `void Decompress(String in, String out)`
      Prosedur untuk mendekompresi *file* in dan menuliskan hasil dekompresi ke *file* out
   b. `void DecompressOut(String in, String out)`
      Prosedur untuk mendekompresi *file* in dan menuliskan hasil dekompresi ke *file* out
   c. `void DecompressAll(String in)`
      Prosedur untuk mendekompresi *file* in dan menuliskan hasil dekompresi ke *file* out. Prosedur ini dapat mendekompresi beberapa *file* sekaligus
   d. `byte[] ArrayByteToArraybyte(Byte[] l)`
      Fungsi untuk merubah array Byte menjadi array byte

### V.4 Implementasi Antarmuka

Implementasi antarmuka agak berbeda dari perancangan antarmuka karena terjadi beberapa penambahan fitur seperti kompresi terhadap banyak *file* sekaligus dan penambahan progress bar. Antarmuka program adalah sebagai berikut :


**Gambar 0-1 Antarmuka utama aplikasi BB-Compress**

User dapat memilih *file* yang ingin dikompresi atau didekompresi. Kompresi dapat dilakukan terhadap beberapa *file* sekaligus. Dekompresi dilakukan terhadap sebuah input *file*. Dekompresi dilakukan sekaligus terhadap seluruh *file* yang ada pada *file* hasil kompresi. Berikut antarmuka pemilihan *file*:


**Gambar 0-3 Antarmuka pemilihan *file***


**Gambar 0-2 Antarmuka file selesai dipilih**

Setelah pemilihan *file* dieksekusi, *file* akan ditambahkan ke list *file* yang ingin dikompresi.
Lalu kompresi dilakukan. Berikut Antarmuka kompresi:


**Gambar 0-4 Kompresi terhadap dua buah file**

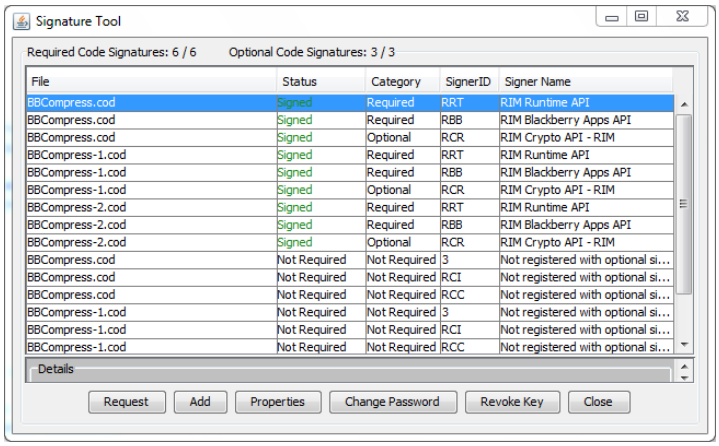
Proses Dekompresi juga sama seperti proses kompresi. Dilakukan pemilihan terhadap *file* yang ingin didekompresi dan lakukan dekompresi.


**Gambar 0-5 Hasil Dekompresi**

### V.5 Implementasi SIgning

Setelah melakukan implementasi ke simulator, kita harus melakukan *signing* terhadap aplikasi tersebut agar dapat dideploy ke *BlackBerry*. Signing tersebut adalah suatu tanda tangan digital yang menunjukkan programmer mana yang mendevelop aplikasi tersebut. Apabila tidak melakukan signing, aplikasi tidak dapat dideploy ke Aplikasi yang sebenarnya. Untuk melakukan signing, harus dilakukan pendaftaran ke BlackBerry untuk mendapatkan kunci privat untuk melakukan signing tersebut. Diperlukan biaya sekitar 200.000 Rupiah untuk mendapatkan kunci privat tersebut.

Setelah mendapatkan kunci privat, dilakukan signing terhadap aplikasi tersebut. Dari kakas *Eclipse Ganymede*, pilih menu *BlackBerry*, lalu pilih *Request Signatures…* Selanjutnya akan muncul popup menu lalu tekan tombol *Request.* PopUp menu akan meminta pengguna memasukkan kunci privatnya. Apabila berhasil, maka akan muncul layar berikut :




**Gambar 0-6 Antarmuka selesai *Signing***

## VI. PENGUJIAN

### VI.1 Tujuan Pengujian

Tujuan pengujian tugas akhir ini adalah memeriksa apakah kebutuhan fungsional sudah terpenuhi yakni dapat mengompresi dan mendekompresi *file* yang ada pada *smartphone BlackBerry*. Tujuan lain pengujian ini adalah untuk mengukur performa aplikasi yang mencakup rasio kompresi, entropi, dan waktu kompresi.

### VI.2 Rancangan Kasus Uji

Beberapa rancangan kasus uji yang dibuat dalam Tugas Akhir ini mencakup dua macam kasus uji. Pengujian pada Tugas Akhir ini mencakup beberapa kasus uji sebagai berikut :

1. Pengujian terhadap beberapa ukuran *Search Buffer*
   Pengujian dilakukan terhadap ukuran search buffer yang berbeda-beda. Pada pengujian kali ini, ukuran search buffer yang diuji adalah 4095 dan 1023.
2. Pengujian terhadap berbagai macam tipe *file*
   Pengujian dilakukan terhadap berbagai jenis *file* seperti *file* gambar (jpg, gif, png), audio(mp3 dan wav) dan teks(txt).

3. Pengujian terhadap beberapa *file* sekaligus
   Pengujian dilakukan terhadap banyak *file* sekaligus dalam *storage BlackBerry*. Pada kasus ini dua buah sekaligus.

**VI.3 Batasan Pengujian**

Beberapa batasan yang ada pada pengujian Tugas Akhir ini adalah

1. Pengujian dilakukan hanya di *BlackBerry Bold* 9000
2. Perhitungan performa kompresi dihitung otomatis oleh program

Perhitungan waktu kompresi diasumsikan *BlackBerry* hanya menjalankan aplikasi ini dan tidak menyalakan aplikasi lain yang diperkirakan akan menggunakan prosesor secara besar.

**VI.4 Hasil Pengujian**

Hasil pengujian akan ditampilkan pada lampiran.

**VI.4 Analisis Pengujian**

Pada bagian ini akan dijelaskan mengenai analisis pengujian berdasarkan pelaksanaan pengujian untuk beberapa kasus *file* pada sub bab sebelumnya. Kompresi yang terjadi dapat dibedakan menjadi dua jenis file yaitu file teks dan file binary. Kompresi terhadap *file* teks dapat mengompresi hampir 50% terhadap ukuran semula. Waktu kompresi file teks cukup lambat. Pada kasus nomor 1, file teks berukuran 111 KB membutuhkan waktu 210 detik untuk mengompresinya menjadi 58.2 KB. Kasus nomor 2 membutuhkan waktu 72.42 detik untuk mengompresinya menjadi 20 KB. Dibutuhkan waktu 795.29 detik untuk file teks berukuran 368 KB pada kasus nomor 3 dan 97.054 detik untuk mengompresi 53 KB file nomor 4. Sehingga dapat diketahui kecepatan kompresi terhadap file teks berkisar antara 0.5- 1 byte per detiknya. Rasio kompresi file teks berkisar antara 50-60%. Entropi dari file teks berkisar sekitar 7.93 – 7.94.

Lain halnya dengan file binary. Beberapa *file* binary yang dikompresi pada pengujian sudah dalam format terkompresi. Apabila kita melakukan kompresi *LZ77* terhadap *file* yang sudah terkompresi kemungkinan besar yang terjadi adalah ekspansi yakni ukuran *file* hasil kompresi lebih besar dari ukuran awal. Lima dari tujuh kasus menunjukkan bahwa setelah mengimplementasikan algoritma LZ77, terjadi ekspansi. Ada dua kasus dimana hasil kompresi terhadap file tersebut berhasil mengurangi ukuran file. Waktu kompresi sangat lambat. Untuk binary *file* proses kompresi berjalan lebih lambat daripada teks karena variasi byte yang jauh lebih besar. Entropi dari file binary berkisar antara 7.86 – 7.92. Rata-rata terjadi ekspansi terhadap jenis file ini. Hal ini dikarenakan untuk file *binary*, byte-byte yang merepresentasikannya ada 256 byte, sedangkan untuk file teks kebanyakan kurang dari itu. Karena kesamaan terhadap file teks relatif lebih banyak dibandingkan file *binary*, otomatis rasio kompresi untuk jenis file ini lebih terjamin.

Pengubahan terhadap ukuran search buffer mengakibatkan perubahan rasio kompresi dan kecepatan kompresi dan dekompresi. Semakin kecil ukuran search buffer, maka kecepatan kompresi semakin kecil, dengan konsekuensi rasio kompresi yang semakin besar. Karena file kompresinya besar, kecepatan dekompresinya pun lebih lambat. Semakin besar ukuran search buffer, maka kecepatan kompresi semakin lambat, dengan keuntungan rasio kompresi yang semakin kecil. Karena hasil kompresinya makin kecil, maka kecepatan dekompresinya makin cepat.

Dengan ini, kebutuhan fungsional untuk melakukan kompresi dan dekompresi dapat terpenuhi. Pengukuran performa kompresi juga berhasil diketahui dan dapat dibandingkan terhadap file yang berbeda-beda.

## VII. KESIMPULAN

Berdasarkan hasil pengujian terhadap kompresi data LZ77 pada smartphone BlackBerry, didapat beberapa kesimpulan sebagai berikut :

1. Aplikasi BB-Compress berhasil mengimplementasikan algoritma kompresi LZ77 pada BlackBerry Bold 9000
2. Algoritma LZ77 dapat diimplementasikan terhadap semua jenis file, namun tidak menjamin terjadinya kompresi. Bahkan dapat terjadi ekspansi.
3. Algoritma LZ77 menunjukkan performasi terbaiknya terhadap file teks karena jumlah simbol yang muncul lebih sedikit. Sedangkan byte-byte dalam file binary lebih bervariasi, maka kemungkinan kemunculan pola byte yang sama jarang terjadi.
4. Kurang efektifnya waktu kompresi dikarenakan prosesor BlackBerry yang kurang memadai. Semakin cepat prosesor, maka proses kompresi dan dekompresi akan semakin cepat.
5. Pemrosesan terhadap file teks lebih cepat daripada file binary karena variasi byte file teks lebih kecil daripada file binary.
6. Ukuran Search buffer mempengaruhi performa kompresi. Apabila diinginkan rasio kompresi makin baik bila ukuran search buffer diperbesar, sebaliknya untuk kecepatan kompresi yang terbaik didapat dengan memperkecil ukuran search buffer.

## REFERENCES


(Salomon, 2006) Salomon, David. *Data Compression The Complete Reference Fourth Edition.* Springer., 2006.

(Muller, 2008) Muller, Jens. *Data Compression LZ77.* Universitat Stuttgart., 2008.

(Strange, 2010) Strange, Peter. http://supportforums.*BlackBerry*.com/t5/Java-Development/Default-encoding/td-p/656701 dibuka pada tanggal 14 April 2011 pukul 07.18

(Finocchiaro,2010) Finocchiaro, Peter. *Mobile data consumption continues to accelerate: study*. http://www.mobilemarketer.com/cms/news/research/7319.html., 2010. Diakses pada tanggal 29 November 2010 pukul 22.05.

(Wordiq, 2011) Wordiq, http://www.wordiq.com/definition/ISO_8859-1 dibuka pada tanggal 13 April 2011 pukul 16.51.

(AppWorld, 2011) AppWorld. http://appworld.*BlackBerry*.com/webstore/category/76?lang=en dibuka pada tanggal 19 April 2011 pukul 00.09.

(Krzysztow, 2011) Krzysztow, http://krzysztow.com/BBNotePad/ dibuka pada tanggal 19 April 2011 pukul 11.06.

(IANA, 2011) IANA, http://www.iana.org/protocols/ dibuka pada tanggal 19 April 2011 pukul 12.01

(FileScout, 2011) *File*Scout, http://www.emacberry.com/bb*file*scout.html dibuka pada tanggal 19 April 2011 pukul 11.57.

(UnRAR, 2010) UnRAR, http://www.berryindo.com/*BlackBerry*-unrar/ dibuka pada tanggal 19 April 2011 pukul 11.58.

(Zeeh, 2003) Zeeh, Christina, http://tuxtina.de/files/seminar/LempelZiv.pdf dibuka pada tanggal 16 Juni 2011 pukul 21.33

# Lampiran

Data hasil pengujian dengan buffer 4095

| No | File Awal | | | | File Hasil Kompresi | | | | Dekompresi |
|---|---|---|---|---|---|---|---|---|---|
| | Nama File | Tipe | Detail | Ukuran (KB) | Ukuran (KB) | Rasio (%) | Waktu(s) | Entropi | Waktu (s) |
| 1. | bil | teks | *File* Bibliografi | 111.3 | 58.2 | 53.64 | 332.69 | 7.938 | 206.62 |
| 2. | progc | teks | Source Code bahasa C | 38.6 | 20.3 | 52.39 | 117.35 | 7.942 | 100.63 |
| 3. | news | teks | *USENET batch file* | 368 | 217 | 59.0 | 1181.3 | 7.938 | 637.68 |
| 4. | paper1 | teks | *Technical Paper* | 53.1 | 28 | 54.3 | 158.25 | 7.940 | 91.23 |
| 5. | ERP.docx | *binary* | *Microsoft Word Document* | 24 | 24.2 | 100.17 | 299.74 | 7.923 | 106.79 |
| 6. | imk.pdf | *binary* | *Portable Document Format* | 96 | 101 | 105.1 | 1329.6 | 7.957 | 310.47 |
| 7. | bellring .wav | *audio* | *Wave Sound* | 15 | 13.6 | 89.57 | 44.092 | 7.881 | 25.211 |
| 8. | teddy.gif | *image* | *GIF* | 12 | 13.2 | 110.26 | 90.796 | 7.860 | 19.231 |
| 9. | roar.mp3 | *audio* | *MP3* | 7 | 7.06 | 107.43 | 50.807 | 7.900 | 15.923 |
| 10 | IU.jpg | *image* | *JPEG* | 16 | 140 | 72.85 | 102.71 | 7.911 | 23.653 |
| | Scandal7 .jpg | *image* | *Joint Photographic Expert Group* | 120 | | 107.68 | 990.99 | 7.956 | 320.786 |

Data hasil pengujian dengan buffer 1023

| No | File Awal | | | | File Hasil Kompresi | | | | Dekompresi |
|---|---|---|---|---|---|---|---|---|---|
| | Nama File | Tipe | Detail | Ukuran (KB) | Ukuran (KB) | Rasio (%) | Waktu(s) | Entropi | Waktu (s) |
| 1. | bib | teks | *File* Bibliografi | 111.3 | 68 | 66.31 | 102.76 | 7.907 | 236.23 |
| 2. | progc | teks | Source Code bahasa C | 38.6 | 22 | 56.85 | 48.037 | 7.903 | 120.21 |
| 3. | news | teks | *USENET batch file* | 368 | 250 | 67.78 | 584.1 | 7.887 | 628.21 |
| 4. | paper1 | teks | *Technical Paper* | 53.1 | 32 | 61.37 | 64.175 | 7.901 | 121.23 |
| 5. | ERP.docx | *binary* | *Microsoft Word Document* | 24 | 24 | 100.89 | 92.122 | 7.916 | 126.79 |
| 6. | imk.pdf | *binary* | *Portable Document Format* | 96 | 100 | 105.32 | 392.65 | 7.955 | 359.47 |
| 7. | bellring .wav | *audio* | *Wave Sound* | 15 | 13.6 | 88.92 | 26.905 | 7.832 | 32.211 |
| 8. | teddy.gif | *image* | *GIF* | 12 | 13.3 | 111.41 | 51.632 | 7.807 | 26.231 |
| 9. | roar.mp3 | *audio* | *MP3* | 7 | 8.21 | 107.27 | 20.3 | 7.894 | 31.923 |
| 10 | IU.jpg | *image* | *JPEG* | 16 | 142 | 72.35 | 50.96 | 7.883 | 28.653 |
| | Scandal7 .jpg | *image* | *Joint Photographic Expert Group* | 120 | | 108.61 | 518.23 | 7.945 | 320.78 |