BAB 2
TINJAUAN PUSTAKA

Dalam penulisan tugas akhir ini, penulis mengambil beberapa materi dan memaparkan teori-teori ilmiah yang didapat dari metode pencarian fakta yang digunakan untuk mendukung penyusunan tugas akhir ini, tinjauan pustaka yang dimaksud adalah sebagai berikut.

2.1. Kompresi Data

Kompresi berarti memampatkan/mengecilkan ukuran. Kompresi data adalah proses mengkodekan informasi menggunakan *bit* atau *information-bearing unit* yang lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem *encoding* tertentu. Contoh kompresi sederhana yang biasa kita lakukan misalnya adalah menyingkat kata-kata yang sering digunakan tapi sudah memiliki konvensi umum. Misalnya: kata "yang" dikompres menjadi kata "yg". Pengiriman data hasil kompresi dapat dilakukan jika pihak pengirim/yang melakukan kompresi dan pihak penerima memiliki aturan yang sama dalam hal kompresi data. Pihak pengirim harus menggunakan algoritma kompresi data yang sudah baku dan pihak penerima juga menggunakan teknik dekompresi data yang sama dengan pengirim sehingga data yang diterima dapat dibaca/di-dekode kembali dengan benar. Kompresi data menjadi sangat penting karena memperkecil kebutuhan penyimpanan data, mempercepat pengiriman data, memperkecil kebutuhan *bandwidth*.

Proses kompresi merupakan proses mereduksi ukuran suatu data untuk menghasilkan representasi digital yang padat atau mampat (*compact*) namun tetap dapat mewakili kuantitas informasi yang terkandung pada data tersebut. Pada citra, video, dan audio, kompresi mengarah pada minimalisasi jumlah *bit rate* untuk representasi digital. Pada beberapa literatur, istilah kompresi sering disebut juga *source coding*, *data compression*, *bandwidth compression*, dan *signal compression*. (Putra, D. 2010)

Dengan merujuk beberapa definisi diatas, dapat ditarik kesimpulan bahwa kompresi adalah sebuah usaha untuk memperkecil ukuran data pada suatu *file*, dimana informasi yang terdapat di dalam *file* tersebut tidak berubah.

Sering terjadi kesalahan pemahaman mengenai data dan informasi, pada data dan informasi adalah dua hal yang berbeda. Pada data terkandung suatu informasi. Namun tidak semua bagian data terkait dengan informasi tersebut atau pada suatu data terdapat bagian-bagian data yang berulang untuk mewakili informasi yang sama. Bagian data yang tidak terkait atau bagian data yang berulang tersebut disebut dengan data yang berlebihan (*redundancy data*). Tujuan daripada kompresi data adalah untuk mengurangi data berlebihan tersebut sehingga ukuran data menjadi lebih kecil dan lebih ringan dalam proses transmisi. (Putra, D. 2010)

Di dalam penelitian ini, file yang akan diteliti berfokus pada jenis *file* teks, dengan berekstensi txt. *File* teks (*flat file*) adalah salah satu jenis *file* computer yang tersusun dalam suatu urutan baris data teks biasanya diwakili oleh 8 bit kode *ASCII*. (Solihin, M. 2013)

2.2. Jenis Kompresi Data

Adapun diketahui beberapa jenis kompresi data yaitu :

1. Jenis Kompresi Data Berdasarkan Mode Penerimaan Data oleh Manusia :

-*Dialogue Mode*: yaitu proses penerimaan data dimana pengirim dan penerima seakan berdialog (*real time*), seperti pada contoh *video conference*. Dimana kompresi data harus berada dalam batas penglihatan dan pendengaran manusia. Waktu tunda (*delay*) tidak boleh lebih dari 150 ms, dimana 50 ms untuk proses kompresi dan dekompresi, 100 ms mentransmisikan data dalam jaringan.

-*Retrieval Mode*: yaitu proses penerimaan data tidak dilakukan secara *real time.*

Dapat dilakukan *fast forward* dan *fast rewind* di client. Dapat dilakukan *random access* terhadap data dan dapat bersifat interaktif.

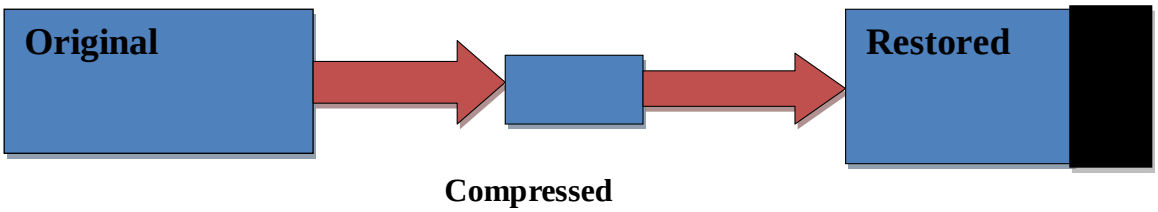
2. Jenis Kompresi Data Berdasarkan Output :

-*Lossy Compression*

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. Kelebihan: ukuran *file* lebih kecil dibanding *loseless* namun masih tetap memenuhi syarat untuk digunakan. 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. Misal terdapat *image* asli berukuran 12,249 bytes, kemudian dilakukan kompresi dengan JPEG kualitas 30 dan berukuran1,869 bytes berarti image tersebut 85% lebih kecil dan ratio kompresi 15%. Contoh beberapa algoritma *lossy compression* yang umum digunakan antara lain algoritma ADPCM, LPC, MPEG, JPEG, *Fractal* dan sebagainya.

LOSSY



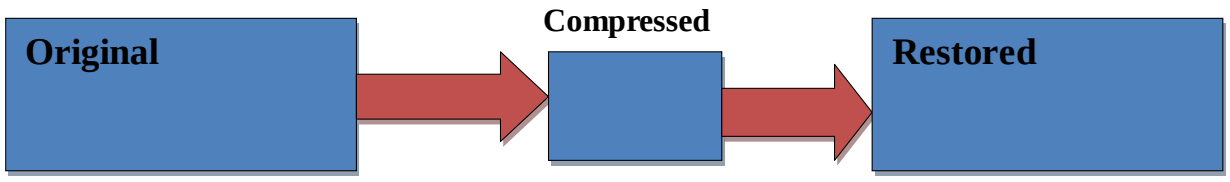
Gambar 2.1 Ilustrasi Kompresi Lossy (Sayood, K. 2005)

- *Lossless Compression*

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. 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. Kadangkala ada data-data yang setelah dikompresi dengan teknik ini ukurannya menjadi lebih besar atau sama. (Antonopoulos, C.P. & Voros, N.S. 2015) Contoh beberapa algoritma *lossless compression* yang umum digunakan antara lain algoritma *Ternary Comma Code, Huffman, Shannon-Fano* dan sebagainya.

Jenis kompresi yang akan dibahas di penelitian ini adalah *lossless*, mengingat sifat kompresi jenis ini yang tidak mengubah isi data ketika akan di dekompresi. *Ternary Comma Code* adalah algoritma yang akan dibahas dalam penelitian ini.

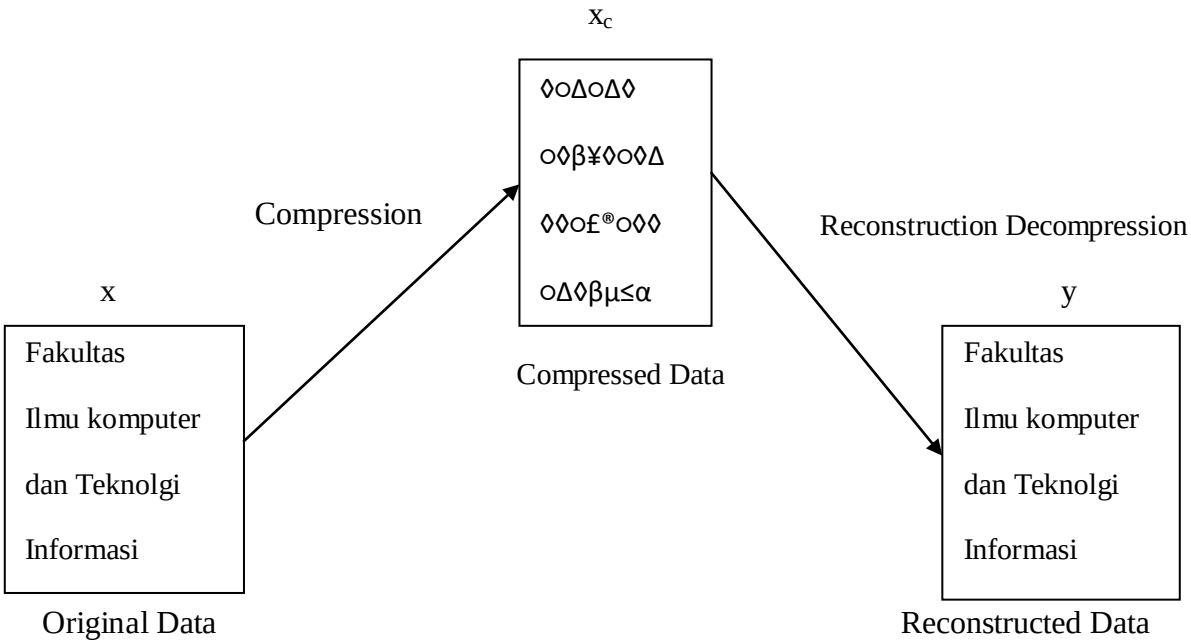
LOSSLESS



Gambar 2.2 Ilustrasi Kompresi Lossless (Sayood, K. 2005)

2.3. Teknik Kompresi

Ketika kita berbicara tentang teknik kompresi atau algoritma kompresi, kita sebenarnya mengacu pada dua algoritma. Ada algoritma kompresi yang mengambil sebuah input x dan menghasilkan x_c sebagai representasi yang memerlukan bit yang lebih sedikit, dan ada algoritma rekonstruksi yang beroperasi pada representasi kompresi x_c untuk menghasilkan rekonstruksi y operasi ini secara skematis diperlihatkan pada Gambar 2.3 (Sayood, K. 2005)



Gambar 2.3 Kompresi dan Rekonstruksi (Sayood, K. 2005)

Berdasarkan persyaratan rekonstruksi, skema kompresi data dapat dibagi menjadi dua kelas besar : skema kompresi *loseless*, dimana *x* identik dengan *y*, dan skema kompresi *lossy*, yang umumnya menyediakan kompresi jauh lebih tinggi daripada kompresi *loseless* tetapi memungkinkan *y* menjadi berbeda dari *x*. (Sayood, K. 2005).

2.4. Parameter Analisis Kompresi

Pada suatu teknik yang digunakan dalam proses kompresi data terdapat beberapa faktor atau variabel yang biasa digunakan untuk menganalisa kualitas dari suatu teknik kompresi data tersebut, yaitu :

1. *Ratio of Compression (R_C)* (Salomon, D. & Motta, G. 2010)

   Ratio of Compression (R_C) adaalah nilai perbandingan antara ukuran bit data sebelum dikompresi dengan ukuran bit data yang telah dikompresi. Secara sistematis dapat dituliskan sebagai berikut :

   $$R_C = \frac{\text{ukuran bit data sebelum dikompresi}}{\text{ukuran bit data setelah dikompresi}} \text{.....(1)}$$

   Persamaan 1. Misalkan di dapat sebuah nilai *Ratio of Compression (R_C)* sebesar 2,75. Itu berarti besar data sebelum kompresi adalah 2,75 kali lipat dari besar data setelah dikompresi.

2. *Compression Ratio (C_R)* (Salomon, D. & Motta, G. 2010)

   Compression Ratio (C_R) adalah persentase perbandingan antara data yang sudah dikompresi dengan data yang belum dikompresi. Secara matematis dapat dituliskan sebagai berikut :

   $$C_R = \frac{\text{ukuran bit data setelah dikompresi}}{\text{ukuran bit data sebelum dikompresi}} \times 100\% \text{.....(2)}$$

   Persamaan 2. Misalkan di dapat sebuah nilai *Compression Ratio(C_R)* sebesar 35%. Itu berarti setelah dikompresi ukuran data adalah 35% dari data sebelum dikompresi.

3. *Space Savings (S_S)* (Siagian, P. 2015)

   Space Savings (S_S) adalah persentase selisih antara data yang belum dikompresi dengan besar data yang dikompresi.

   $$S_S = 1 - C_R \text{.....(4)}$$

   Persamaan 4. Misalkan telah didapat hasil dari nilai C_R, maka 1 dikurangkan dengan hasil dari nilai C_R tersebut. Sehingga didapatkan nilai S_s.

4. Waktu kompresi dan dekompresi (Siagian, P. 2015)

   Waktu kompresi dan dekompresi adalah waktu yang dibutuhkan untuk melakukan proses kompresi dan dekompresi. Semakin kecil waktu yang diperoleh maka semakin efisien metode yang digunakan dalam proses kompresi dan dekompresi itu.

2.5. *File Text*

Teks adalah kumpulan dari karakter-karakter atau string yang menjadi satu kesatuan. Teks yang memuat banyak karakter didalamnya selalu menimbulkan masalah pada media penyimpanan dan kecepatan waktu pada saat transmisi data. *File* teks merupakan *file* yang berisi informasi-informasi dalam bentuk teks. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh masukan data teks yang terdiri dari karakter, angka dan tanda baca. (Pramilo, C. 2008)

Format *file* teks yang akan digunakan dalam penelitian ini adalah format data teks *(*.txt)* .

Masukan dan keluaran data teks dipresentasikan sebagai set karakter atau sistem kode yang dikenal oleh sistem komputer. Ada tiga macam set karakter yang umum digunakan untuk masukan dan keluaran pada komputer, yaitu *ASCII, Unicode,* dan *EBCDIC, ASCII* (*American Standard Code For Information Interchange*) merupakan suatu standar internasional dalam kode huruf dan simbol seperti *Hex,* dan *Unicode,* tetapi *ASCII* bersifat lebih universal. *ASCII* memiliki komposisi bilangan biner sebanyak 8 bit, dimulai dari 00000000 dan 11111111. Total kombinasi yang dihasilkan sebanyak 256, dimulai dari kode 0 hingga 255 dalam sistem bilangan desimal. *Unicode* adalah suatu standar industri yang dirancang untuk mengizinkan teks dan simbol dari semua sistem tulisan di dunia untuk ditampilkan dan dimanipulasi secara konsisten oleh komputer.*EBCDIC (Extended Binary Code Decimal Interchange Code)* merupakan set karakter yang diciptakan oleh komputer merk *IBM*. *EBCDIC* terdiri dari 256 karakter yang masing-masing berukuran 8 *bit*. (Sudewa, I. B. A. 2003)

2.5.1. Format Teks

Tipe teks merupakan tipe dasar yang sudah sangat dikenal dalam kehidupan sehari-hari. Tipe teks terdiri dari tipe karakter (char) dan tipe string. Tipe karakter (char) terdiri atas satu huruf, angka, tanda baca, atau karakter khusus seperti "c", "1", "#" dan sebagainya. Tipe string terdiri atas nol atau lebih karakter seperti "teks" dan "algoritma" dan sebagainya.

Secara umum, format data teks dibagi menjadi dua bagian, yaitu (Purnomo, H. & Zacharias, T. 2005) :

1. Teks sederhana (*plain text*)

   Format data teks (*.txt) merupakan contoh format teks jenis ini yang paling populer.

2. Teks terformat (*formatted text*)

   Merupakan teks yang terformat dan mengandung *styles*. Format data dokumen *Microsoft Word* (*.doc) merupakan contoh format teks jenis ini yang paling populer.

Contoh format data teks di atas beserta perangkat lunak yang bisa digunakan di antaranya adalah (Purnomo, H. & Zacharias, T. 2005) :

1. Format data teks (*.txt)

   Format data teks merupakan format teks yang digunakan untuk menyimpan huruf, angka, karakter kontrol (tabulasi, pindah baris, dan sebagainya) atau simbol-simbol lain yang biasa digunakan dalam tulisan seperti titik, koma, tanda petik dan sebagainya. Satu huruf, angka, karakter kontrol atau simbol pada arsip teks memakan tempat satu *byte*. Berbeda dengan jenis teks terformat yang satu huruf saja dapat memakan tempat beberapa *byte* untuk menyimpan format dari huruf tersebut seperti *font*, ukuran, tebal atau tidak dan sebagainya. Kelebihan dari format data teks ini adalah ukuran datanya yang kecil karena tiadanya fitur untuk memformat tampilan teks. Saat ini perangkat lunak yang paling banyak digunakan untuk memanipulasi format data ini adalah *Notepad*.

2. Format data dokumen (*.doc)

   Doc merupakan ekstensi arsip dokumen perangkat lunak *Microsoft Word* yang paling banyak digunakan dalam menulis laporan, makalah dan sebagainya. *Doc* merupakan jenis teks terformat yang tidak hanya dapat mengatur tampilan

teks seperti *styles* (*font*, ukuran huruf, dan sebagainya), namun juga dapat menyisipkan gambar. Kekurangan format teks dokumen ini terletak pada ukuran datanya yang besar.

3. *Hyper text markup language* (*.htm atau *.html)
   Merupakan format teks standard untuk tampilan dokumen web.
4. *Rich text format* (*.rtf)
   Format teks ini dikembangkan oleh Microsoft yang dapat dibaca oleh berbagai macam *platform*, seperti *Windows*, *Linux*, *Android*, *Mac OS* dan sebagainya.

2.6. Algoritma *Ternary Comma Code*

Algoritma merupakan urutan langkah-langkah untuk menyelesaikan masalah yang disusun secara sistematis, algoritma dibuat dengan tanpa memperhatikan bentuk yang akan digunakan sebagai implementasinya, sehingga suatu algoritma dapat menjelaskan "*bagaimana"* cara melaksanakan fungsi dapat diekspresikan dengan suatu program atau suatu komponen fisik. (Hartono, T. B. 2007)

Bilangan biner (basis 2) adalah didasarkan pada dua bit 0 dan 1. Demikian juga seperti bilangan *ternary* (basis 3) yang didasarkan pada tiga bilangan (*trit*) 0, 1 dan 2. Setiap *trit* dapat dituliskan dalam dua bit, tetapi dua bit dapat memiliki empat nilai. Sehingga, ini memungkinkan untuk bekerja dengan sistem bilangan *ternary* dimana masing-masing trit di wakili oleh dua bit dan dalam penjumlahan ketiga trit ada simbol ke empat yaitu comma (c). Ketika kita memasukkan (c), ini akan menjadi lebih mudah untuk membuat kode *ternary comma* untuk bilangan bulat. Kode *comma* dari n secara sederhana mewakili ternary dari n-1 di ikuti oleh sebuah *comma* (c). Sehingga kode *comma* dari 8 adalah 21c (karena 7= 2.3+1) dan kode *comma* dari 18 adalah 122c (karena 17=1.9+2.3+2).

Tabel dibawah mencatat beberapa bilangan kode *ternary comma* (kolom L adalah panjang dari kode dalam *bit*). Kode ini bertambah panjang (lebih panjang dari kebanyakan kode yang digambarkan disini) tetapi tumbuh secara perlahan sehingga kode ini cocok digunakan untuk aplikasi yang biasanya menggunakan bilangan bulat yang besar. Kode ini juga mudah untuk dibuat dan diuraikan, dan kekurangan yang mendasar adalah simbol *comma* (yang menandakan akhir dari kode) membutuhkan dua bit. Ketidak efisienan ini tidak serius, tetapi menjadi lebih untuk kode comma yang berdasar pada bilangan yang lebih besar. Pada kode comma yang berbasis 15,

contohnya masing-masing dari 15 bilangan membutuhkan 4 bit dan comma juga berpola 4 bit. Setiap kode diakhiri dengan comma 4 bit, bukan 1 bit dan fitur ini membuat kode ini tidak efisien. (Namun, keseluruhan redudansi setiap simbol mengurangi bilangan yang berbasis besar. Di sistem basis 7, 1 dari 8 simbol dikurangi untuk comma, sementara dalam basis 15 adalah 1 dari 16 simbol). (Salomon, D. 2007)

Tabel 2.1 Algortima *Ternary Comma Code* dan Panjangnya. (Salomon, D. 2007)

Value	Code	L	Value	Code	L
0	C	2	11	101c	8
1	0c	4	12	102c	8
2	1c	4	13	110c	8
3	2c	4	14	111c	8
4	10c	6	15	112c	8
5	11c	6	16	120c	8
6	12c	6	17	121c	8
7	20c	6	18	122c	8
8	21c	6	19	200c	8
9	22c	6	20	201c	8
…			…		
64	2100c	10	1,000	1101000c	16
128	11201c	12	3,000	11010002c	18
256	100110c	14	10,000	111201100c	20
512	200221c	14	65,536	10022220020c	24

2.7. Kompleksitas Algoritma

Kompleksitas waktu dari algoritma berisi ekspresi bilangan dan jumlah langkah yang dibutuhkan sebagai fungsi dari ukuran permasalahan. Analisa asimtotik menghasilkan notasi O atau *Big Oh*, dan dua notasi untuk komputer sain yaitu Θ (*Big Theta*) dan Ω (*Big Omega*). (Purwanto, E. B. 2008)

Kinerja algoritma dibuktikan dengan menjumlahkan bilangan bulat dari masing-masing operasi ketika algoritma di jalankan. Kinerja sebuah algoritma dievaluasi sebagai fungsi ukuran masukan n dan konstanta modulo pengali yang digunakan. Pada penelitian ini kompleksitas yang digunakan adalah *Big Theta (Θ).*

1. Big-O (O)

Secara informal, *0(g(n))* adalah himpunan semua fungsi yang lebih kecil atau dengan urutan yang sama dengan *g(n)* (hingga beberapa konstanta, sampai n ke tak terhingga). Sebuah fungsi *t(n)* dikatakan bagian dari *0((g(n))* yang dilambangkan dengan *t(n)* $\in$ *0(g(n))*, jika *t(n)* batas atasnya adalah beberapa konstanta *g(n)* untuk semua *n* besar, jika terdapat konstanta *c* positif dan beberapa bilangan bulat non negatif n_0 seperti $t(n) \leq cg(n)$ untuk semua $n \geq n_0$. (Levitin, A. 2011)

2. Big Theta (Θ)

Θ(g(n)) adalah himpunan semua fungsi yang memiliki tingkat pertumbuhan yang sama dengan *g(n)* (hingga beberapa konstanta, sampai *n* ke tak terhingga). Sebuah fungsi *t(n)* dikatakan bagian dari *Θ(g(n))*, dilambangkan dengan *t(n)* $\in$ *Θ(g(n))*, jika *t(n)* batas atas dan bawahnya adalah beberapa konstanta positif *g(n)* untuk semua *n* yang besar, yaitu jika ada beberapa konstanta positif c_1 dan c_2 serta beberapa bilangan bulat non negatif n_0 seperti $c_2 g(n) \leq t(n) \leq c_1 g(n)$ untuk semua $n \geq n_0$. (Levitin, A. 2011)

3. Big Omega (Ω)

Ω(g(n)) merupakan himpunan semua fungsi dengan tingkat pertumbuhan lebih besar atau sama dengan *g(n)* (hingga beberapa konstanta, sampai *n* ke tak terhingga). Sebuah fungsi *t(n)* dikatakan bagian dari *Ω(g(n)),* dilambangkan dengan *t(n)* $\in$ *Ω(g(n)),* jika *t(n)* batas bawahnya adalah beberapa konstanta positif dari *g(n)* untuk semua *n* besar. Terdapat konstanta *c* positif dan beberapa bilangan bulat non negatif n_0 seperti $t(n) \geq cg(n)$, (untuk setiap $n \geq n_0$). (Levitin, A. 2011)

2.8. *Platform Android*

Android sudah menjadi platform sistem operasi yang makin popular. Padahal sebenarnya, *Android* termasuk sistem operasi '*yunior*' dibandingkan lainnya yang lebih '*senior*', seperti *symbian*, *iOS*, atau sistem operasi *blackberry*. *Android* secara

sederhana bisa diartikan sebagai sebuah *software* yang digunakan pada perangkat *mobile* yang mencakup sistem operasi, *middleware*, dan aplikasi kunci yang dirilis oleh *Google*. Sehingga android mencakup keseluruhan sebuah aplikasi, dimulai dari sistem operasi sampai pada pengembangan aplikasi itu sendiri. (EMS, Tim. 2015)

Pengembangan aplikasi pada *platform Android* ini menggunakan dasar bahasa pemrograman *Java*. Tapi secara sempit, *Android* biasanya mengacu pada sistem operasinya saja. *Platform* pengembangan aplikasi *Android* ini bersifat *open source* atau terbuka, sehingga dapat mengembangkan kemampuan untuk membangun aplikasi yang kaya dan inovatif. (EMS, Tim. 2015)

Dasar pemrograman *Android* adalah *Java*, karena aplikasi android ditulis dalam bahasa *java*. *Android* menyediakan lingkungan atau *run time environment* yang dikenal sebagai *Dalvik Virtual Machine* ini merupakan *Java runtime environment* yang telah dioptimasi untuk *device* dengan sistem memori yang kecil. (EMS, Tim. 2015)

2.9. Penelitian Relevan

Berikut ini beberapa penelitian relevan yang terkait dengan penelitian ini :

1. Umri Erdiansyah (2014) dalam skripsi yang berjudul *Perbandingan Algoritma Elias Delta Code Dengan Levenstein Untuk Kompresi File Teks*. Dalam skripsi dapat disimpulkan bahwa perbedaan cara kerja kompresi antara kedua algoritma tersebut dalam mengompresi file teks. Proses kompresi dengan menggunakan *Elias Delta* mempunyai keterbatasan karena hanya dapat mempresentasikan 7 karakter pertama dengan jumlah bit dibawah 8 bit. Waktu yang dibutuhkan untuk kompresi tidak bergantung pada besarnya ukuran file dan begitu pula waktu dekompresi tidak tergantung pada waktu kompresi.

2. Pahara Siagian (2015) dalam skripsi yang berjudul *Analisis Perbandingan Kinerja Algoritma Fixed Length Binary Encoding (FLBE) Dengan Algoritma Sequitur Dalam Kompresi File Text*. Dalam skripsi dapat disimpulkan bagaimana perbandingan kedua algoritma tersebut dalam kinerja mengkompresi file teks. Sehingga dapat menghasilkan pemampatan nilai kompresi yang baik.