JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 2, April 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i2.2828
Hal 50−54

Implementasi Algoritma Adaptive Arithmetic Coding Pada Aplikasi Juz' Amma Berbasis Android

Ari Maulana Asri, Abdul Sani Sembiring

Teknik Informatika, STMIK Budi Darma, Medan, Indonesia
Email: arieatom97@gmail.com
Submitted **12-02-2021**; Accepted **10-04-2021**; Published **25-04-2021**

Abstrak

Ukuran file text relatif besar dimana semakin banyaknya informasi yang terkandung pada file tersebut, maka besarnya ukuran file text yang dibutuhkan untuk menyimpan informasi tersebut juga akan semakin besar, sehingga berdampak pada ukuran file yang harus disimpan pada media penyimpanan harus semakin besar pula. Ukuran memori yang dibutuhkan untuk menyimpan sebuah file berbanding lurus dengan ukuran file text tersebut. Permasalahan yang terjadi adalah terbatasnya ruang penyimpanan yang tersedia khususnya pada aplikasi android. Jika ukuran file text lebih besar dari ruang penyimpanan atau pun batasan maksimum ukuran sebuah file, maka pengiriman file text akan gagal ataupun informasi yang terkandung dialam file text tersebut tidak akan tersampaikan dengan baik. Penelitian ini mempelajari teknik kompresi file teks menggunakan pengkodean aritmetika, serta mengimplementasikannya untuk kasus kompresi dan dekompresi. Hal yang dikaji antara lain: melihat pengaruh variasi karakter di dalam suatu teks terhadap rasio kompresi, waktu kompresi yang dihasilkan serta waktu dekompresinya. Hasil percobaan menunjukkan bahwa rasio kompresi yang tertinggi diperoleh pada saat suatu teks hanya memiliki satu jenis karakter saja. Adapun pengaruh variasi karakter terhadap rasio kompresi adalah semakin banyak variasi karakter yang dimiliki oleh suatu teks maka rasio kompresi akan semakin rendah.

Kata Kunci: Kompresi; File Teks; Juz' amma; Adaptive Arithmetic Coding

Abstract

The size of the text file is relatively large, where the more information contained in the file, the larger the size of the text file needed to store the information, so that it has an impact on the file size that must be stored on the storage media. The memory size required to store a file is directly proportional to the size of the text file. The problem that occurs is the limited storage space available, especially for Android applications. If the text file size is larger than the storage space or the maximum size limit for a file, sending the text file will fail or the information contained in the text file will not be conveyed properly. This research studies text file compression techniques using arithmetic coding, and implements it for compression and decompression cases. Things to be studied include: looking at the effect of character variations in a text on the compression ratio, the resulting compression time and the decompression time. The experimental results show that the highest compression ratio is obtained when a text has only one type of character. The effect of character variations on the compression ratio is that the more character variations a text has, the lower the compression ratio.

Keywords: Compression; Text Files; Juz 'amma; Adaptive Arithmetic Coding

1. PENDAHULUAN

Juz' Amma adalah juz terakhir dari tiga puluh juz Al Quran. Ciri utama surah-surahnya adalah singkat-singkat, dengan bahasa yang indah memesona, menyentuh hati atau menghardikannya disertai dengan argumentasi-argumentasi rasional yang mampu menyakinkan nalar yang belum dikeruhkan oleh kerancuan berpikiratau subjektivitas pandangan. Sehingga perlu dibangun sebuah aplikasi Juz' Amma tersebut terutama berbasis android yang memudahkan untuk seluruh masyarakat mengakses atau pun membaca Juz' Amma dimana pun dan kapan pun. Salah satu informasi *file* yang disimpan pada aplikasi Juz' Amma adalah *File Text* dengan format *file text* antaranya .doc, .txt, pdf dan sebagainya. Ukuran*file text* relatif besar dimana semakin banyaknya informasi yang terkandung pada *file* tersebut, maka besarnya ukuran *file text* yang dibutuhkan untuk menyimpan informasi tersebut juga akan semakin besar, sehingga berdampak pada ukuran *file* yang harus disimpan pada media penyimpanan harus semakin besar pula. Ukuran memori yang dibutuhkan untuk menyimpan sebuah *file* berbanding lurus dengan ukuran *file text* tersebut. Permasalahan yang terjadi adalah terbatasnya ruang penyimpanan yang tersedia khususnya pada aplikasi android. Jika ukuran *file text* lebih besar dari ruang penyimpanan atau pun batasan maksimum ukuran sebuah *file*, maka pengiriman *file text* akan gagal atau pun informasi yang terkandung didalam *file text* tersebut tidak akan tersampaikan dengan baik. Salah satu teknik sederhana yang sangat dibutuhkan untuk masalah diatas adalah kompresi. Kompresi merupakan proses untuk mengecilkan ukuran *file* dari ukuran aslinya. Penyimpanan *file text* yang memiliki ukuran lebih besar dapat dikompres menjadi ukuran yang lebih kecil dan mengurangi redundansi dari data – data yang tedapat dalam suatu *file* sehingga dapat disimpan dan ditransmisikan secara efisien. Salah satu format *file text* yang paling sering digunakan untuk menyimpan informasi pada *file* dan sering digunakan pada aplikasi berbasis android berformat .doc.

Pada umumnya algoritma kompresi data melakukan teknik pengkodean dengan penggantian satu atau beberapa simbol yang sama dengan kode tertentu, berbeda dengan sebuah angka *floating point* dengan interval [0,1) atau angka yang lebih besar atau sama dengan 0 dan lebih kecil dari 1 ($0 \leq x < 1$). Salah satu algoritma kompresi yang dapat digunakan untuk mengecilkan ukuran *file* dari *file* aslinya adalah *Adaptive Arithmetic Coding*. *Adaptive Arithmetic Coding* memiliki sejarah yang penting karena pada saat itu algoritma ini berhasil menggantikan algoritma *Huffman* selama 25 tahun, *Arithmetic Coding* memiliki perfomansi yang lebih unggul dari pada *Huffman Coding* khususnya apabila diaplikasikan pada kumpulan alphabet yang ukurannya relatif kecil, awalnya *Arithmetic Coding* diperkenalkan oleh Shannon Fano dan Elias, kemudiandikembangkan secara luas oleh Pasco, Rissanene dan Langdon, yaitu sebagai ide alternatif yang menggantikan

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 2, April 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i2.2828
Hal 50−54

setiap *input* simboldengansebuah *codeword*, yaitu dengan mengkodekan seluruh *input stream* dengan sebuah *single floating point* sebagai *output* proses kompresi. Untuk melakukan proses kompresi maupun dekompresi, *AdaptiveArithmetic Coding* membutuhkan dua fase, yaitu dengan terlebih dahulu menentukan probalitas dan *range* setiap simbol lalu melakukan proses *encoding* maupun *decoding*. Pada hasil penelitians ebelumnya yang telah dilakukan oleh Syahfitri Kartika Lidya, Mohammad Andri Budiman dan Romi Fadillah Rahmat pada tahun 2013 telah berhasil melakukan kompresi dan juga dekompresi serta mengembalikan informasi yang telah dikompresi dengan tingkat rasio rata – rata sebesar 62,88% [1] .Penelitian yang sebelumnya dilakukan untuk mengkompresi Citra BMP, berdasarkan referensi tersebut dan juga dengan algoritma yang sama penulis ingin melakukan kompresi pada sebuah *file text* padaaplikasi android.

2. METODOLOGI PENELITIAN

2.1 Kompresi

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 dan audio, kompresi ini mengarah pada minimalisasi jumlah bit rate untuk representasi digital [2].

2.2 Aplikasi Juz'Amma

Juz' Amma adalah juz terakhir dari tiga puluh juz Al Quran. Ciri utama surah-surahnya adalah singkat-singkat, dengan bahasa yang indah memesona, menyentuh hati atau menghardikannya disertai dengan argumentasi-argumentasi rasional yang mampu menyakinkan nalar yang belum dikeruhkan oleh kerancuan berpikiratau subjektivitas pandangan. Sehingga perlu dibangun sebuah aplikasi Juz' Amma tersebut terutama berbasis android yang memudahkan untuk seluruh masyarakat mengakses atau pun membaca Juz' Amma dimana pun dan kapan pun. Salah satu informasi *file* yang disimpan pada aplikasi Juz' Amma adalah *File Text* dengan format *file text* antaranya .doc, .txt, pdf dan sebagainya [3].

2.3 Algoritma Adaptive Arithmetic Coding

Arithmetic Coding diperkenalkan pada tahun 1970-an, metode ini memiliki efisiensi yang baik dan implementasinya pada perangkat keras sangat fleksibel. Topik tentang algoritma ini pertama kali diberikan oleh Abramson dan Peter Elias pada tahun 1960, namun pada saat itu metode ini belum memenuhi solusi yang pantas untuk masalah yang akan dihadapi, yaitu keakurasian Arithmetic Coding harus ditingkatkan dengan panjang dari pesan yang dimasukkan. Untungnya, pada tahun 1976 Pasco dan Rissanen membuktikan bahwa panjang angka yang terbatas sebenarnya memadai untuk encoding, tanpa mengurangi akurasinya.Pada tahun 1979 – 1980, Rubin, Guazzo, Rissanen, dan Langdon mempublikasikan algoritma dasar encoding yang masih digunakan sampai sekarang.Algoritma ini berdasarkan ketelitian aritmatik yang terbatas. Arithmetic Coding menggantikan satu deretan simbol input dengan sebuah bilangan floating point. Semakin panjang dan semakin kompleks pesan yang dikodekan, semakin banyak bit yang diperlukan untuk keperluan tersebut. Output dari metode ini adalah satu angka yang lebih kecil dari 1 dan lebih besar atau sama dengan 0. Angka ini secara unik dapat di-encode sehingga menghasilkan deretan simbol yang dipakai untuk menghasilkan angka tersebut[4].

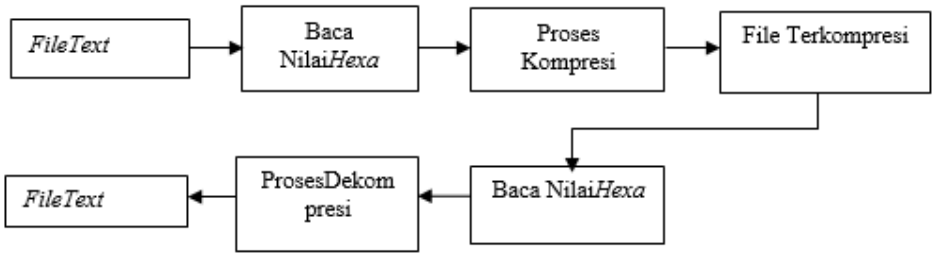
Gagasan utama pada arithmetic coding adalah memberikan setiap simbol sebuah range atau interval. Dimulai dengan range [0..1), setiap range dibagi dalam beberapa subrange, dimana ukurannya sebanding dengan probabilitas (kemungkinan yang muncul dari simbol-simbol yang sama). Semakin tinggi probabilitas yang dimiliki oleh suatu simbol, semakin tinggi pula range yang diberikan terhadap simbol tersebut. Tanda"[..)" berarti bahwa angka yang didalamnya diikutsertakan kecuali angka terakhir. Setelah menentukan range dan probabilitasnya, mulai dapat melakukan proses encoding simbol-simbol, setiap simbol menentukan dimana output angka floating point itu berada. Sebagai contoh, jika terdapat suatu message "tawa", maka distribusi probabilitasnya yaitu : Bagian yang paling penting pada proses encoding yaitu simbol pertama. Ketika melakukan encoding dari message baca, simbol yang pertama adalah "t". Agar simbol pertama dapat di-decode dengan tepat, hasil dari encode message harus sebuah angka lebih besar atau sama dengan 0.5 dan kurang dari 0.75. Apa yang dilakukan untuk encoding angka ini yaitu tetap menjaga agar angka ini jatuh diantara range tersebut. Jadi setelah simbol pertama di-encode, range terendah adalah 0.5 dan range tertinggi adalah 0.75. Setelah simbol pertama di-encode, sudah dapat diketahui range untuk output number yang dibatasi oleh low number dan high number. Apa yang terjadi selama proses encoding yaitu bahwa setiap simbol baru untuk di-encode lebih lanjut akan membatasi range dari output angka. Simbol selanjutnya untuk di-encode yaitu "a", memiliki range 0.0 hingga 0.5. Jika itu merupakan angka pertama dalam message, maka akan diset lownumber dan high number langsung pada nilainya. Tetapi "a" adalah simbol kedua. Jadi simbol "a" memiliki range yang sama dengan 0.0-0.5 dalam subrange yang baru 0.5- 0.75. Ini berarti angka hasil encode yang baru akan jatuh pada bagian yang ke-0 hingga yang ke-50 dari range sebelumnya. Dengan logika ini didapatkan angka range 0.5 hingga 0.625.

3. HASIL DAN PEMBAHASAN

Dalam hal ini penerapan algoritma *adaptive arithmetic coding* dilakukan terhadap *file text* yang memiliki ukuran yang relatif besar. Dengan begitu proses kompresi data sangat diperlukan. Kompresi data merupakan teknik pemampatan data, sehingga diperoleh ukuran *file* yang lebih kecil dari aslinya. *File text* memiliki ukuran relatif besar, semakin besar informasi yang tersimpan pada sebuah *file* maka membutuhkan alokasi penyimpanan yang besar. Dalam melakukan kompresi *file text*

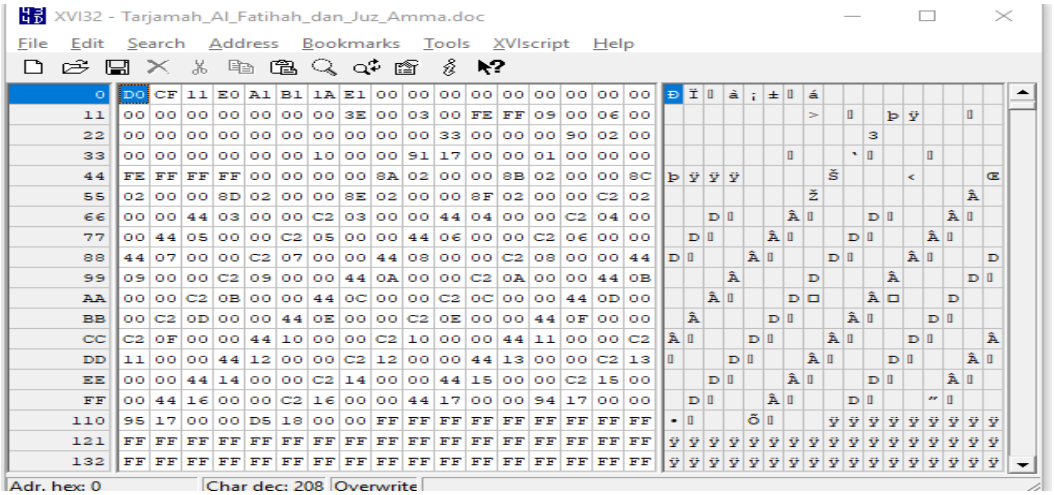
JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 2, April 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i2.2828
Hal 50−54

sebelumnya harus dilakukan analisa terhadap *filetext*. *Filetext* yang akan dianalisa yaitu *file text* berformat *doc*. Format *file doc* merupakan format yang minim kompresi, sehingga ukurannya cukup besar, sehingga diperlukan pengkompresian *file*. Adapun tujuan dari analisa terhadap sistem yang akan dibangun yaitu untuk mengetahui dan merumuskan kebutuhan dari sistem serta membantu meminimalisir sumber daya yang berlebih. Berikut merupakan prosedur kompresi dan dekompresi *file* gambar 1.



Gambar 1. Prosedur Kompresi Dekompresi *File Text*.

Berdasarkan analisa, *file text* memiliki ukuran lebih besar dibanding ukuran *file* lainnya. Dengan melakukan kompresi data, data yang berukuran besar akan dikompresi menjadi ukuran yang kecil dan akan mengurangi alokasi penyimpan. Dalam menganalisa *file text* harus dilakukan pengambilan *sample file* untuk mendapatkan nilai dari *pixel*-nya. Berikut adalah langkah untuk mengkompresi dan mendekompresi *file text*. Struktur data pada *file text* berbeda-beda tergantung format *file*, pada contoh berikut akan dikompresi *file text* dengan ukuran 3,1 MB. Dari *sample file* didapat nilai *hexadecimal* menggunakan bantuan *software hex editor* seperti pada gambar 1.



Gambar 2. Nilai *Hexadecimal File Text* .doc

Berdasarkan pada gambar diatas didapat nilai *hexadecimal file text* doc. Untuk keperluan hitungan manual hanya diambil sampel nilai sebanyak 34 karakter nilai *hexadecimal file text*. Nilai *hexadecimal* diambil dari sisi kiri atas sampai dengan bilangan ke32. Adapun bilangan *hexadecimal file text sample* tersebut adalah D0, CF, 11, E0, A1, B1, 1A, E1, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 3E, 00, 03, 00 FE, FF, 09, 00, 06, 00. Langkah awal dari proses kompresi terlebih dahulu yaitu mencari probabilitas masing-masing data dengan cara membagi jumlah frekuensi bilangan *hexadecimal* dengan jumlah total frekuensi bilangan *hexadecimal*. Dapat dilihat pada tabel 3.1.

Tabel 1. Tabel Probabilitas

No	Nilai	Frekuensi	Probabilitas
1	D0	1	1/34=0,03
2	CF	1	1/34=0,03
3	11	1	1/34=0,03
4	E0	1	1/34=0,03
5	A1	1	1/34=0,03
6	B1	1	1/34=0,03
7	1A	1	1/34=0,03
8	E1	1	1/34=0,03
9	00	20	20/34=0,58
10	3E	1	1/34=0,03
11	03	1	1/34=0,03
12	FE	1	1/34=0,03

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 2, April 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i2.2828
Hal 50−54

13	FF	1	1/34=0,03
14	09	1	1/34=0,03
15	06	1	1/34=0,03

Selanjutnya akan diperoleh *table range* probabilitas dengan menjumlahkan *range* terendah data dan probabilitas data. Dapat dilihat pada tabel 2.

Tabel 2. *Range* Probabilitas

No	Nilai	Frekuensi	Probabilitas	Range
1	D0	1	1/34=0,03	0,00 < D0 > 0,03
2	CF	1	1/34=0,03	0,03 < CF > 0,06
3	11	1	1/34=0,03	0,06 < 11 > 0,09
4	E0	1	1/34=0,03	0,09 < E0 > 0,12
5	A1	1	1/34=0,03	0,12 < A1 > 0,15
6	B1	1	1/34=0,03	0,15 < B1 > 0,18
7	1A	1	1/34=0,03	0,18 < 1A > 0,21
8	E1	1	1/34=0,03	0,21 < E1 > 0,24
9	00	20	20/34=0,58	0,24 < 00 > 0,82
10	3E	1	1/34=0,03	0,82 < 3E > 0,85
11	03	1	1/34=0,03	0,85 < 03 > 0,88
12	FE	1	1/34=0,03	0,88 < FE > 0,91
13	FF	1	1/34=0,03	0,91 < FF > 0,94
14	09	1	1/34=0,03	0,94 < 09 > 0,97
15	06	1	1/34=0,03	0,97 < 06 > 1,00

Selanjutnya mencari nilai Cr (Xi) atau nilai jarak interval karakter.Data yang diproses adalah data yang *range* awalnya adalah 0. Data yang pertama diolah adalah D0 dengan nilai *low* (X1) = 0,0dan *high* (X1) =0,03. Setelah menentukan Cr (X2) = *high* (X2) – *low* (X2) =0,03 kemudian menentukan nilai *high* (X2) dan *low* (X2), yaitu :

Low(X2) = low(X1) + cr(X2) * low(X2)
= 0,0 + 0,03 * 0,03 = 0,0009
Low(X3) = low(X2) + cr(X3) * low(X3)
= 0,03 + 0,03 * 0,06 = 0,0318
Low(X4) = low(X3) + cr(X4) * low(X4)
= 0,06 + 0,03 * 0,09 = 0,0627
Low(X5) = low(X4) + cr(X5) * low(X5)
= 0,09 + 0,03 * 0,12 = 0,0936
Low(X6) = low(X5) + cr(X6) * low(X6)
= 0,12 + 0,03 * 0,15 = 0,1245
Low(X7) = low(X6) + cr(X7) * low(X7)
= 0,15 + 0,03 * 0,18 = 0,1554
Low(X8) = low(X7) + cr(X8) * low(X8)
= 0,18 + 0,03 * 0,21 = 0,1863

high(X2) = low(X1) + cr(X2) * high(X2)
= 0,0 + 0,03 * 0,06 = 0,0018
high(X3) = low(X2) + cr(X3) * high(X3)
= 0,03 + 0,03 * 0,09 = 0,0327
high(X4) = low(X3) + cr(X4) * high(X4)
= 0,06 + 0,03 * 0,12 = 0,0636
high(X5) = low(X4) + cr(X5) * high(X5)
= 0,09 + 0,03 * 0,15 = 0,0945
high(X6) = low(X5) + cr(X6) * high(X6)
= 0,12 + 0,03 * 0,18 = 0,1254
high(X7) = low(X6) + cr(X7) * high(X7)
= 0,15 + 0,03 * 0,21 = 0,1563
high(X8) = low(X7) + cr(X8) * high(X8)
= 0,18 + 0,03 * 0,24 = 0,1872

Tabel 3. Hasil *encoding* data *sample*

No	Nilai	Low	High
1	D0	0	0,03
2	CF	0.0009	0,0018
3	11	0,0318	0,0327
4	E0	0,0627	0,0636
5	A1	0,0936	0,0945
6	B1	0,1245	0,1254
7	1A	0,1554	0,1563
8	E1	0,1863	0,1872
9	00	0,3492	0,6856
10	3E	0,2646	0,2655
11	03	0,8455	0,8464
12	FE	0,8764	0,8733
13	FF	0,9073	0,9082
14	09	0,9382	0,9391
15	06	0,9691	0,97

JURIKOM (Jurnal Riset Komputer), Vol. 8 No. 2, April 2021
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v8i2.2828
Hal 50−54

Tabel 3. menunjukkan bahwa nilai *low* dan untuk data terakhir adalah nilai batas bawah yang akan digunakan untuk menggantikan *sample file* yang telah di *encoding* yaitu nilai *sample* audio D0, CF, 11, E0, A1, B1, 1A, E1, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 3E, 00, 03, 00 FE, FF, 09, 00, 06, 00diubah menjadi 0,9691.

4. KESIMPULAN

Adapun kesimpulan dari hasil penelitian diatas untuk proses kompresi pada penelitian ini, terlebih dahulu harus mengetahui nilai hexa desimal suatu file. Kemudian dapat dilakukan proses kompresi menggunakan teknik kompresi. Algoritma adaptive arithmetic coding dapat digunakan untuk mengurangi ukuran file pada aplikasi juz' amma.

REFERENCES

[1] S. K. Lidya, M. A. Budiman and R. F. Rahmat, "Implementasi dan Analisis Kinerja Algoritma Arithmetic Coding dan Shannon-Fano Pada Kompresi Citra BMP," ResearchGate, pp. 1-5, 2013.
[2] I. M. Pu, Fundamentals Data Compression, Butterworth-Heinemann is an imprint of Elsevier, 2006.
[3] G. M. David Salomon and D. Bryant, Handbook of Data Compression Fifth Edition, 2010.
[4] R. S. Brar and B. singh, "A Survey on Different Compression Techniques and Bit Reduction Algorithm for Compression of Text/Lossless Data," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, 2013.
[5] M. Drs. Lamhot Sitorus, Algoritma dan Pemrograman, Yogyakarta: Penerbit Andi, 2015.
[6] Sukiman and T. Chandra, "Aplikasi Kompresi File dengan Algoritma Elias Gamma," 2013.
[7] H. Purnomo and T. Zacharias, Pengenalan Informatika Perspektif Teknik dan Lingkungan, Yogyakarta: Andi, 2005.
[8] B. S. D. Oetomo, Perencanaa dan Pengembangan Sistem Informasi, Yogyakarta: CV. Andi Offset, 2002.
[9] R. A. Sukmato and M. Shalahuddin, Rekayasa Perangkat Lunak (Arsitektur Dan Berorientasi Objek), Bandung: Bandung : Informatika, 2014.
[10] H. N. Lengkong, S. M. Alicia A.E. Sinsuw and S. M. Arie S.M Lumenta, "Perancangan Penunjuk Rute Pada Kendaraan Pribadi Menggunakan Aplikasi Mobile GIS Berbasis Android Yang Terintegrasi Pada Google Maps," E-journal Teknik Elektro dan Komputer, p. 21, 2015.