Kompresi File dengan Huffman Coding

Rita Sarah 13512009
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512009@std.stei.itb.ac.id

***Abstrak*— Makalah ini membahas tentang Huffman coding serta perannya dalam pengompresian file yang berguna untuk membuat ukuran data lebih kecil dari seharusnya.**

***Kata kunci* : Huffman coding, encoding ,decoding, *lossless files, lossy files* ,kompresi data.**

I. PENDAHULUAN

Pada era perkembangan teknologi informasi , banyak sekali hal yang dapat dilakukan dengan bantuan perangkat lunak pada komputer. Teknologi yang dikenal dengan teknologi informasi itu berperan dengan memberi informasi dari suatu tempat ke tempat lainnya sehingga dapat diakses. Hal ini memudahkan banyak hal dalam pengiriman data.

Karena itu sangat dibutuhkan kompresi data yang baik untuk meningkatkan efisiensi pengiriman informasi. Dengan kompresi data yang baik, kita dapat menggunakan sebuah file dengan ukuran *drive* yang tidak terlalu besar, sehingga dapat mempercepat sampainya pengiriman informasi. Selain itu, tentu saja file yang berukuran kecil akan lebih mudah diakses, sehingga dapat digunakan dan diberdayakan oleh banyak orang.

Masalah lain muncul ketika kita ingin untuk mengunggah file. Terlebih lagi jika file yang ingin kita ungga berukuran besar sehingga membutuhkan kecepatan internet yang lebih. Melihat bahwa akses file di Indonesia belum cukup memadai, hal ini tentu cukup merepotkan untuk sebagian orang.

Masalah berlanjut ketika kita ingin mendapat preview dari sebuah file berukuran besar yang sudah diunggah di internet. Tentunya file dengan ukuran yang lebih besar akan membutuhkan waktu yang lebih lama supaya kita bisa mendapat previewnya.

Jika demikian, bagaimana masalah – masalah di atas dapat diatasi? Tentu saja ada cara untuk mengatasi masalah tersebut,yaitu dengan cara mengkompresi data tersebut menjadi lebih kecil. Salah satu cara kompresi itu adalah dengan menggunakan Huffman coding. Huffman coding ini digunakan pada banyak file, seperti JPEG,MP3,GZip. File tersebut ada yang bersifat *lossless* dan *lossy*.

Walau ukuran kecil dapat berpengaruh baik bagi pengaksesan data yang cepat, atau *storage use* yang minimal, namun dapat berefek pada data tersebut.

Dalam kompresi file, sering digunakan Huffman coding untuk membuat ukuran file menjadi lebih efisien pada storage.

Huffman coding dilakukan dengan mengubah karakter – karakter yang terdapat dalam suatu file menjadi sekumpulan kode, kemudian memisahkan kode mana yang sering digunakan dan yang tidak sering digunakan pada file tersebut.

Proses mengubah karakter – karakter menjadi sekumpulan kode disebut proses encoding. Pada proses ini, ukuran file sekaligus terkompres sehingga menjadi lebih kecil. Jika kita ingin membaca file tersebut, maka kita harus melakukan proses decoding, yaitu mengubah kembali kode – kode menjadi sekumpulan karakter. Proses decoding sekaligus menghilangkan kompresi file.

.

II. HUFFMAN CODING

Pada Huffman coding digunakan pohon . Pohon adalah graf tidak berarah terhubung yang tidak mengandung sirkuit. Misalkan G=(V,E) adalah graf tidak berarah sederhana dan jumlah simpulnya n . Maka,

- G adalah pohon
- Setiap pasang smpul di dalam G terhubung lintasan tunggal
- G terhubung dan memiliki m=n-1 sisi
- G tidak mengandung sirkuit dan memiliki m=n-1 sisi
- G tidak mengandung sirkuit dan penambahan satu pada graf akan membuat hanya satu sirkuit
- G terhubung dan semua sisinya adalah jembatan (jembatan adalah sisi yang bila dihapus menyebabkan graf terpecah menjadi dua komponen)

Pada aplikasi pohon, simpul tertentu diperlakukan sebagai akar (*root*). Simpul lainnya dapat dicapai dari akar dengan member arah pada sisi-sisi pohon yang mengikutinya. Pohon yang sebuah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah adalah pohon berakar.

Akar mempunyai derajat masuk 0 dan simpul lainnya berderajat masuk 1. Simpul yang mempunyai derajat keluar sama dengan nol disebut simpul dalam atau simpul cabang. Setiap simpul di pohon berakar dengan a adalah

simpul akarnya. Arah sisi didalam poho dapat dibuang dan lintasan didalam pohon berakar dari atas ke bawah.

Pada pohon dikenal beberapa terminology seperti :

- Anak (*child* atau *children*) dan orangtua (*parent*) . Simpul y dikatakan anak simpul x jika ada sisi dari simpul x ke y. x disebut orangtua (*parent*) y .
- Lintasan (*path*) dari simpul awal ke akhir adalah runtutan simpul-simpul sedemikian sehingga vi adalah orangtua dari vi+1 .
- Keturunan (*descendant*) dan leluhur (*ancestor*) jika terdapat simpul x ke y maka x adalah leluhur y dan y adalah keturunan simpul x.
- Saudara kandung (*sibling*) adalah simpul yang berorang tua sama. Contoh simpul x memiliki orangtua y yang memiliki anak x dan z , maka z adalah saudara kandung x.
- Upapohon (*subtree*) jika x adalah simpul dari pohon T, upapohon dari T dengan x sebagai akar ialah T'=(V',E') sedemikian sehingga V' mengandung x dan semua keturunan dan E' mengandung sisi-sisi dalam semua lintasan yang berasal dari x. Jika pohon berukuran lumayan besar terdapat banyak upapohon.
- Derajat (*degree*) , derajat sebuah simpul pada pohon berakar adalah jumlah upapohon (atau jumlah anak) pada simpul tersebut. Derajat yang dimaksudkan adalah derajat keluar. Derajat maksimum dari semua simpul merupakan derajat pohon itu sendiri.
- Daun (*leaf*) adalah simpul yang berderajat 0 , berarti tidak mempunyai anak ,hanya mempunyai orang tua.
- Simpul dalam (*internal nodes*) adalah simpul yang mempunyai anak . Semua simpul kecuali daun dan akar adalah simpul dalam.
- Aras (*level*) atau tingkat . Akar memiliki aras 0. Aras simpul lain adalah 1+ panjang lintasan ke simpul tersebut.
- Tinggi (*height*) atau kedalaman (*depth*) adalah aras maksimum dari sebuah pohon. Untuk mendapat aras maksimum dapat dengan menghitung panjang lintasan dari akar ke daun.

Sebuah aplikasi dari pohon biner adalah Huffman coding, sebuah metode yang meminimalkan redundansi dari kode-kode. Pohon biner memiliki 2 *branch*, kiri dan kanan. Pohon biner sering digunakan dalam bidang Teknologi Informasi.

Huffman coding termasuk *greedy algorithm*. *Greedy algorithm* adalah proses matematika yang membuat set objek secara rekursif dari bagian-bagian terkecil. *Greedy algorithm* mencari solusi-solusi yang mudah terlebih dahulu sampai yang kompleks.

Huffman coding merupakan algoritma yang digunakan umumnya pada kompresi data *lossless*. Huffman coding dibuat oleh David A. Huffman di makalah buatannya pada tahun 1952 "A Method for the Construction of Minimum-Redundancy Codes.".

Waktu yang dibutuhkan dalam penjalanan algoritma ini termasuk cepat dengan kompleksitas waktu O(n log n).

Selain itu dikenal juga Adaptive Huffman coding , yang didasarkan Huffman coding , membuat kode menjadi simbol yang dikirimkan, tanpa pengetahuan tentang sumber distribusi dan memungkinan one-pass encoding dan adaptasi untuk mengganti kondisi di data.

Ini bermanfaat agar sumber dapat langsung di encode secara real time, namun lebih sensitif pada error sehingga kehilangan simbol dapat merusak seluruh kode.

Pada Huffman coding, digunakan representasi bit untuk merepresentasikan sebuah kode, jika kode tersebut memiliki peluang kemunculan sangat tinggi , maka representasi bit lebih pendek dibandingkan sebuah kode yang memiliki peluang kemunculan rendah.

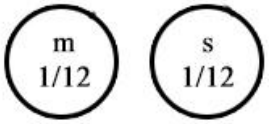
Pada algoritma ini, hal –hal yang perlu dilakukan adalah :

1. Mengelompokkan kode yang akan diproses, sehingga kode tersebut diketahui peluang-peluang kemunculannya.
2. Memilih 2 simbol yang memiliki peluang paling kecil . Kedua simbol tersebut digabungkan , misalnya A dan B menjadi AB .
3. Peluang A dan B dijumlahkan menjadi peluang AB .
4. Simbol baru ini diperlakukan sebagai simpul orangtua baru dan diperhitungkan untuk mencari simbol lain.
5. Pilih 2 simbol lagi dari tabel Huffman termasuk simbol yang sebelumnya dibuat , dan membuat lagi simbol baru
6. Lakukan step diatas sampai tinggal 1 simbol.
7. Pohon Huffman sudah terbentuk, namun belum selesai, tentukan bit untuk kiri dan kanan pohon.
8. Memasang deretan bit pada pohon biner dengan nilai 0 dan 1 , nilai ini harus diletakkan di tempat yang sama sehingga konsisten.
9. Pad peluang kemunculan terendah memiliki deret bit paling panjang, dan selanjutnya sampai semua kode sudah diassign dengan deretan bits.
10. Membuat tabel Huffman yang akan dipakai beserta representasi bit baru .
11. Menggantikan representasi bit yang lama dengan representasi bit yang baru.

Untuk contoh Huffman encoding, pada string "ooooosmiifff". Jika dibuat tabel kemunculannya akan seperti:

Simbol	m	i	f	o	s
Frekuensi	1/12	2/12	3/12	5/12	1/12

Tabel 2.1 Tabel Huffman 1

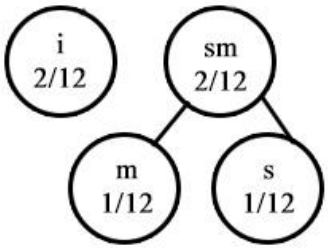


Grafik 2.1 Pembuatan Pohon Huffman 1

Pertama-tama memilih yang frekuensinya terendah, pada kasus ini karakter m dan s. m dan s diletakkan di bagian bawah pohon Huffman dan diassign dengan bit 1 dan 0 . Karena m dan s sudah bergabung , maka tabel akan menjadi seperti:

Simbol	Ms	i	f	o
Frekuensi	2/12	2/12	3/12	5/12

Tabel 2.2 Tabel Huffman 2

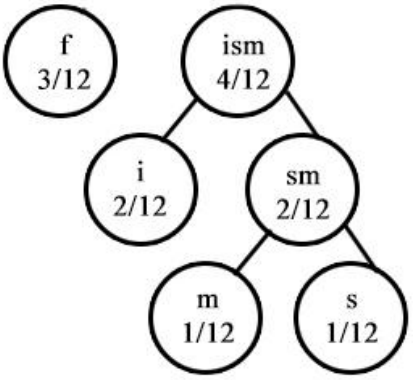


Grafik 2.2 Pembuatan Pohon Huffman 2

Karena 2 node yang memiliki frekuensi terendah sekarang adalah ms dan i , maka kita menggunakan ms dan i sebagai node lalu digabungkan menjadi 1 node lagi . Maka tabel akan menjadi seperti :

Simbol	Ism	F	o
Frekuensi	4/12	3/12	5/12

Tabel 2.3 Tabel Huffman 3



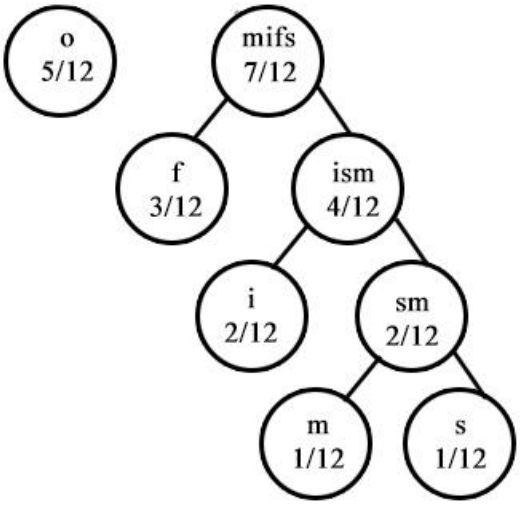
Grafik 2.3 Pembuatan Pohon Huffman 3

Setelah itu , pilih lagi 2 node yang memiliki frekuensi terendah dari tabel. ism dan f memiliki frekuensi terendah , maka ism dan f digabung menjadi 1 node.

Tabel setelah sim dan f digabung :

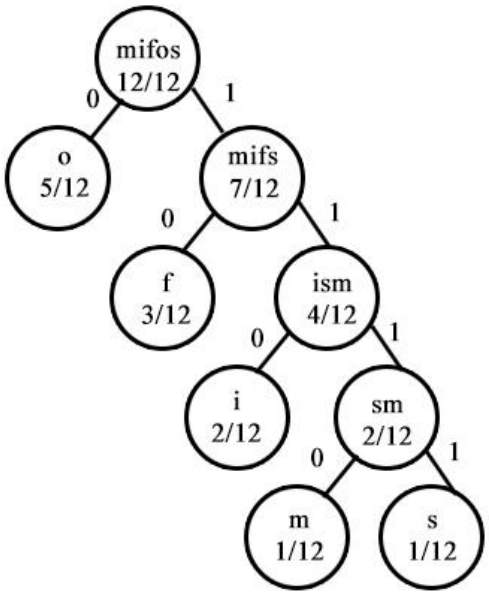
Simbol	Miffs	o
Frekuensi	7/12	5/12

Tabel 2.4 Tabel Huffman 4



Grafik 2.4 Pembuatan Pohon Huffman 4

Sisa 2 node terakhir digabung juga menjadi 1 node. Setelah itu assign bit pada pohon, sehingga pohon Huffman sudah selesai , dengan pohon sebagai berikut :



Grafik 2.5 Pohon Huffman yang sudah diberi bit 1 dan 0 pada pohon.

Dengan adanya pohon biner tersebut dapat dilihat representasi bit baru .

Dari Grafik 2.5 didapatkan data deret bit untuk setiap karakter , masukkan ke tabel Huffman .

Simbol	Representasi Bit
m	1110
s	1111
f	10
i	110
o	0

Tabel 2.5 Tabel Huffman baru ,representasi bit baru

Jika dimasukkan ke string yang dimiliki sebelumnya "ooooosmiifff" , maka representasi bit barunya menjadi : 00000111111101101101010 yang merupakan hasil kompresi dari Huffman coding.

Untuk melihat efek dari representasi bit baru , bandingkan dengan representasi bit lama , pada tabel :

Simbol	Representasi Bit ASCII
m	01101101
s	01110011
f	01100110
i	01101001
o	01101111

Tabel 2.6 Representasi bit lama

Jika digunakan untuk string yang sama "ooooosmiifff" maka representasi bit lama yang menggunakan ASCII menjadi:
01101111011011110110111101101111011011110111001101101101011010010110100101100110011001100110 0110.

Pada deretan bit yang baru digunakan 23 bit sedangkan pada deretan bit yang lama digunakan 96 bit. Pada kasus ini, rasio kompresi = ((Jumlah bit lama- jumlah bit baru)/jumlah bit lama)*100% =76.04% . Pada umumnya Huffman encoding mengkompresi sekitar 20-30% dari file input.

Pada Huffman decoding, dapat menggunakan pohon biner atau tabel (array).Jika sudah ada pohon biner Huffman, decoding dilakukan dengan melihat bit pertama hasil encoding, lalu melihat bit selanjutnya. Jika deret bit yang bersangkutan merupakan daun dari pohon biner, maka sebuah karakter telah di *decode* , maka karakter tersebut dapat diletakkan di sebuah output . Selanjutnya bit setelah bit tersebut dianggap seperti bit pertama lagi dan dilakukan pencarian apakah bit tersebut merupakan daun dari pohon biner. Jika belum ada pohon biner, dapat menggunakan array untuk menjadi tabel lookup dan operasi shift dan mask.

III. APLIKASI HUFFMAN CODING

A. Lossless files

Lossless file merupakan jenis file yang melewati jenis kompresi yang menghasilkan file output yang bersifat sama dengan file input. Algoritma Huffman umumnya digunakan untuk kompresi file-file bersifat *lossless*. Seperti pada contoh di bab II, string yang digunakan tetaplah sama namun dengan representasi bit yang berbeda sehingga file tersebut memang terkompres, namun tidak merubah isi file tersebut. Ide dari *lossless compression* adalah untuk memindah-mindahkan file menjadi bentuk yang lebih kecil , dan digabungkan pada akhirnya untuk bisa digunakan kembali.

Pada kompresi ini kualitas file output sama dengan input sehingga tidak perlu mengkhawatirkan turunnya kualitas file setelah diencode.

Lossless file penting karena ada beberapa file yang butuh keakuratan sama persis dengan file input sehingga tidak boleh diganti bitnya.

Contoh file lossless misalnya pada sebuah file JPEG-LS, JPEG 2000. File JPEG (Joint Photographic Experts Group) merupakan skema kompresi file bitmap.

File extension lainnya yang termasuk kompresi lossless untuk gambar adalah PNG dan TIFF. Perbedaan pada file JPEG dan PNG adalah file JPEG tidak bisa menyimpan gambar transparan, sedangkan file PNG bisa. Sedangkan yang termasuk kompresi lossless untuk audio adalah FLAC, APE, TTA, dan WMA. Selain itu, HuffYUV adalah salah satu contoh codec video yang menggunakan Huffman coding untuk lossless files.

B. Lossy files

Lossy files merupakan istilah yang digunakan untuk menyebut kompresi file yang bersifat *lossy*.Pada bagian A dibahas tentang kompresi *lossless*.Kompresi jenis ini sangat berbeda dengan kompresi yang sebelumnya dibahas. *Lossy* file merupakan file yang melewati jenis kompresi yang menghasilkan file output yang agak berbeda dengan file input. Pada proses kompresi itu, algoritma yang digunakan mengeliminasi bit-bit yang dianggap tidak penting pada file. Ini dapat berdampak dengan berkurangnya kualitas file. Karena file tersebut bersifat *lossy*, maka file outputnya adalah hasil file yang terkompres lebih banyak, sehingga ukuran file *lossy* lebih kecil daripada *lossless.*

Kompresi *lossy* juga tidak merubah langsung jenis file dan spesifikasi file secara umum , hanya mengurangi bit bit yang dianggap kurang penting.

Walaupun mengeliminasi beberapa bit yang kurang penting , tetap encoding ini memiliki banyak hal positif. Encoding ini berguna untuk banyak hal . Misalnya seorang kolektor file MP3 hanya menyiapkan ukuran drive yang lebih kecil daripada kolektor file FLAC. Walaupun termasuk *lossy*, file MP3 dengan rate 320kbps termasuk lumayan baik. Namun jika file yang dihasilkan terlalu lossy, mengakibatkan file output menjadi kurang

baik karena terlalu banyak mengeliminasi bit-bit.Misalnya pada kompresi gambar , dalam sebuah gambar langit yang berwarna biru dengan gradasi abu-abu atau putih akan kurang terlihat gradasinya pada langit ketika sudah menjadi file yang *lossy* , dan pengguna data tidak dapat mengembalikan file tersebut menjadi file *lossless*, maka pengguna data harus memperhitungkan akibat dari kompresi file yang akan dilakukan.

C. Contoh

Pada akhir pembuatan JPEG digunakan Huffman coding ,JPEG merupakan *file extension* yang termasuk *lossy* pada umumnya , dan merupakan file biasanya digunakan pada gambar , namun tidak menutup kemungkinan digunakan untuk teks juga, walau teks tersebut akan berfungsi sebagai gambar ketika sudah menjadi JPEG.

Sebuah file JPEG berisi 4 tabel Huffman yang mendefinisikan mapping dari panjang kode-kode tersebut dan nilai kode itu sendiri. Dalam pembuatan tabel ini dimulai dengan menghitung frekuensi simbol yang ada pada gambar dan mengalokasi bit string, namun pada umumnya encoder JPEG menggunakan tabel Huffman yang sudah ada sebagai standar JPEG.

Beberapa encoder juga mengoptimasikan tabel tersebut sehingga dapat membuat pohon biner yang menghasilkan tabel Huffman yang lebih efisien. Karena itu ada file yang bersifat *lossless* dan *lossy* tergantung algoritma encoding.

Huffman coding digunakan pada bagian terakhir pembuatan file MP3. File MP3 termasuk pada file-file yang berjenis lossy. Format MP3 menggunakan frame 1152 nilai sampel..Pada MP3 ada rate sampel (yang meruapakan frekuensi) dan dari data tersebut dapat dikira-kira waktu yang digunakan perframe. Spektrum dari frame sampel ini dianalisis secara spektral dan frekuensinya dikelompokkan di 32 channel.

Model-model ini menentukan jenis suara di setiap channel. Informasi semua channel digabungkan menjadi sinyal yang berbeda-beda waktunya. Signal dari frame tersebut melalui Huffman coding dan menghasilkan sebuah file MP3.

Huffman coding untuk menghasilkan kompresi file yang lossy juga bisa dikarenakan encoder menggunakan tabel Huffman sendiri seperti kasus yang dibahas pada bagian sebelumnya.

Membuat sebuah MP3 termasuk hal yang susah dilakukan , dan proses encoding lebih sulit daripada decoding. Decoding yang dimaksud pada MP3 adalah memutar file MP3.

Dapat disimpulkan jika memiliki bit yang sama dan banyak maka akan membuat file size menjadi lebih efisien. Misalnya 2 buah gambar , gambar A hanya berisi warna biru solid , dan gambar B berisi gradasi warna biru merah dan putih . Gambar A akan memiliki *file size* yang lebih kecil dibandingkan dengan gambar B karena dalam pemanggilan representasi bit yang baru , gambar B memanggil banyak warna sedangkan gambar A hanya 1 warna.

V. KESIMPULAN

Huffman coding digunakan sebagai salah satu cara dalam kompresi file. Huffman encoding merupakan algoritma yang mengganti representasi bit menjadi baru berdasarkan pohon Huffman. Aplikasi Huffman coding ditemukan pada banyak *file extension* dan tetap digunakan hingga sekarang sebagai metode kompresi oleh beberapa encoder. Kompresi file ada 2 yaitu *lossless* dan *lossy*. Keduanya memiliki kekuatan dan kelemahan masing-masing.

VII. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan yang Maha Esa, karena oleh karena-Nya makalah ini dapat diselesaikan. Penulis juga mengucapkan terima kasih kepada Bapak Rinaldi Munir dan Ibu Harlili selaku guru dari mata kuliah Matematika Diskrit IF 2120 yang telah mengajari mata kuliah ini sehingga lebih mudah dipahami dan memudahkan dalam pembuatan paper .

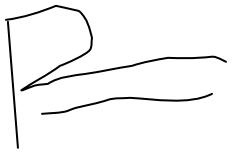
. DAFTAR PUSTAKA

[1]Rinaldi, Munir. *Diktat Kuliah IF2120 Matematika Diskrit*. Bandung: Institut Teknologi Bandung. 2006.
[2]http://scanftree.com/Data_Structure/huffman-code http://www.phy.davidson.edu/fachome/dmb/py115/huffman_coding.htm Waktu Akses : 16.34 14 Desember 2013
[3]http://www.mp3-converter.com/mp3codec/huffman_coding.htm Waktu Akses : 16.45 14 Desember 2013
[4]http://computer.howstuffworks.com/file-compression3.htm Waktu Akses : 17.01 14 Desember 2013
[5]http://www.impulseadventure.com/photo/jpeg-huffman-coding.html Waktu Akses :17.22 14 Desember 2013
[6]http://people.cs.pitt.edu/~kirk/cs1501/animations/Huffman.html Waktu Akses : 19.35 14 Desember 2013
[7]http://stackoverflow.com/questions/2235208/how-to-decode-huffman-code-quickly Waktu Akses :21.12 14 Desember 2013
[8]http://whatis.techtarget.com/definition/greedy-algorithm Waktu Akses :22.34 14 Desember 2013
[9] http://www.maximumcompression.com/algoritms.php Waktu Akses : 23.09 14 Desember 2013

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 27 November 2013



Rita Sarah
13512009