Induksi Matematika pada Pengecekan Terminasi Pengulangan Program

Friska 13514042[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]friska.nataniel@itb.ac.id

Abstrak*—Pengulangan dalam program mempunyai suatu kondisi awal dan kondisi akhir. Suatu program dapat berjalan dengan baik bila badan pengulangan yang dijalankan menemui kondisi berhenti / menemui keadaan terminasi. Pengulangan menjadi baik bila tujuan dari dijalankannya tercapai. Diperlukan fasilitas untuk mengecek pengulangan pada suatu program. Fasilitas tersebut berupa loop invariant / pengulangan tidak berubah yang menjadi patokan penyesuaian keadaan variabel dan decrementing function / fungsi berkurang yang mengecek terminasi pengulangan.

***Keywords*—fungsi berkurang, induksi matematika, pengulangan, pengulangan tidak berubah**

I. PENGANTAR

Dunia semakin maju dengan berbagai teknologi yang semakin berkembang. Tidak bisa dihindari bahwa manusia yang hidup pada zaman ini harus mengerti teknologi. Teknologi diciptakan untuk membuat kemajuan bagi manusia. Beberapa manusia lebih maju dari manusia lainnya dalam hal pengetahuan. Dari sekian kecanggihan teknologi yang ada, beruntunglah manusia menciptakan komputer.

Suatu komputer dibekali dengan mesin yang akan menjalankan instruksi sesuai bahasa yang dimengertinya. Manusia umumnya menggunakan komputer yang sudah dipermudah dan disesuaikan untuknya. Di balik itu, harus ada tenaga yang dikeluarkan untuk membuat mesin mengerti keinginan manusia. Usaha tersebut dituliskan dalam program dengan bentuk kode.

Pemrograman adalah proses penuangan pikiran manusia dalam bentuk kode. Kode tersebut disusun sesuai jenis bahasa yang digunakan. Pemrograman dilakukan pada komputer, dengan tujuan menyampaikan keinginan manusia untuk dimengerti oleh mesin lewat perantara bahasa. Sama seperti manusia yang satu hendak menyampaikan pendapat kepada manusia yang lain, mereka membutuhkan media berkomunikasi yakni bahasa; manusia juga berkomunikasi dengan mesin menggunakan bahasa. Kenyataannya, bahasa yang digunakan sangat beragam, sehingga sebaiknya manusia memelajari cara berpikir bukan cara menggunakan masing – masing bahasa pemrograman. Adapun cara berpikir tersebut disebut sebagai algoritma. Lebih lengkapnya, algoritma adalah jalan berpikir dengan setiap langkahnya menuju kepada penyelesaian masalah. Salah satu struktur pada algoritma adalah pengulangan.

Pengulangan memfasilitasi program untuk terus melakukan aksi yang sama berulang kali dengan kondisi awal (kondisi yang diperlukan untuk memasuki badan pengulangan) dan berhenti pada kondisi tertentu. Membahas kembali pada gambaran program secara utuh, program akan berjalan baik bila ia berjalan sesuai keinginan kita. Salah satu hal yang harus dicek adalah badan pengulangan yang ada dalam program tersebut. Salah satu cara mengecek pengulangan adalah dengan memanfaatkan induksi matematika.

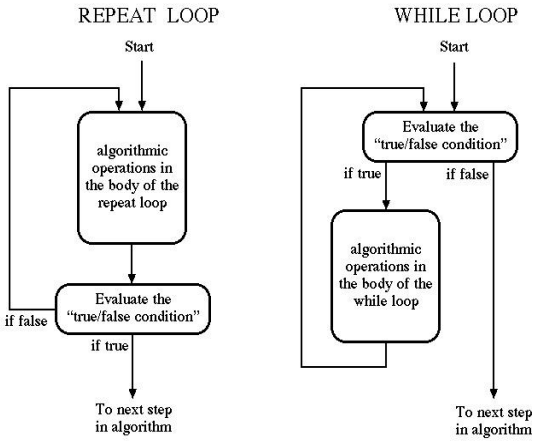
II. TEORI YANG BERKAITAN

A. Pengulangan

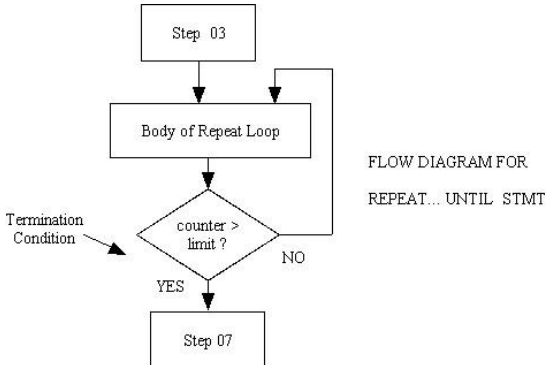
Manusia meletakkan pengulangan pada kode programnya untuk suatu tujuan. Salah satu jenisnya adalah untuk merubah nilai variabel. Nilai tersebut akan diubah secara bertahap sesuai dengan kondisi ulang yang diinginkan. Sebuah pengulangan dikatakan berhasil bila ia menemui keadaan akhir (terminasi) dan mencapai tujuan / *goal* yang diinginkan. Untuk itu dibuat pengulangan yang tidak berubah / *loop invariant* dan fungsi berkurang / *decrementing function*. Pengulangan yang tidak berubah akan menunjukkan tercapainya tujuan dari pengulangan tersebut, fungsi berkurang akan menunjukkan terminasi dari fungsi (saat memiliki nilai minimum maka seharusnya program keluar dari badan pengulangan). Keduanya bergantung pada interasi dari badan pengulangan.

Ada beberapa jenis pengulangan: struktur *while do*, struktur *repeat until*, dan struktur *for*. *While do* adalah struktur pengulangan yang akan menjalankan pengulangannya berdasarkan kondisi benar. Dapat dimengerti: selama keadaan masih berupa sekian, perintah dijalankan. Program menyocokkan dahulu kondisi dengan

syarat ulang, bila masih sama maka program akan masuk ke badan pengulangan. Dengan demikian struktur ini akan berhenti bila kondisi sudah tidak cocok dengan kondisi ulang / syarat untuk ulang. *Repeat until* adalah struktur pengulangan berdasarkan kondisi berhentinya. Struktur ini membuat program otomatis berjalan satu kali dahulu (masuk ke dalam badan pengulangan), kemudian baru mengecek keadaan. Bila keadaan sudah sesuai dengan syarat berhenti (pada *until*) maka program akan keluar dari badan pengulangan. Dapat dimengerti: lakukan suatu perintah sampai keadaan berhenti terpenuhi. *For* adalah struktur pengulangan berdasarkan iterasi. Pengulangan dilakukan dari suatu nilai awal sampai suatu nilai akhir secara bertahap / traversal.



Gambar 1. Algoritma pengulangan *repeat* dan *while*

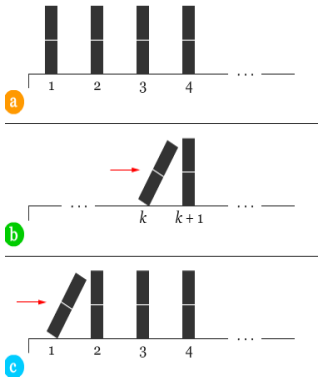


Gambar 2. Algoritma pengulangan *for*

B.Induksi Matematika

Induksi matematika adalah suatu alat yang memberikan bukti bagi suatu pernyataan bila ia bernilai benar. Gagasan dari induksi matematika adalah memberikan bukti kebenaran untuk suatu nilai, kemudian secara induksi membuat nilai yang lain memiliki nilai benar pada pernyataan tersebut. Menurut sejarahnya, induksi matematik berawal pada akhir abad ke-19. R. Dedekind mengembangkan sekumpulan aksioma yang menggambarkan bilangan bulat positif. G. Peano memerbaiki aksioma tersebut dan memberikannya interpretasi logis.

- *Prinsip Induksi Sederhana*
  (Munir, 2015: IV – 2)
  Misalkan p(n) adalah pernyataan perihal bilangan bulat positif dan kita ingin membuktikan bahwa p(n) benar untuk semua bilangan bulat positif n. Untuk membuktikan pernyataan ini, kita hanya perlu menunjukkan bahwa:
  1. p(1) benar
  Basis. Setelah dibuktikan, maka hasil ini dapat digunakan untuk step ke-2
  2. untuk semua bilangan bulat positif n>=1, jika p(n) benar maka p(n+1) juga benar.
  Dengan memanfaatkan pembuktian pada langkah pertama, pernyataan tersebut dijadikan hipotesis induksi. Hipotesis tersebut menjadi alat pembuktian untuk langkah ke-2.
  Bila kedua langkah sudah benar, maka terbukti kebenaran untuk nilai dan pernyataan tersebut.



Gambar 3. Ilustrasi induksi matematika

- Prinsip Induksi yang dirampatkan
  Prinsip ini berguna bila pembuktian diharapkan memenuhi semua bilangan bulat.≥ bilangan basis yang dipilih. Prinsip induksi yang dirampatkan adalah prinsip induksi sederhana yang dimodifikasi sehingga bisa memenuhi semua bilangan bulat.
  Misal ada pernyataan mengenai bilangan bulat yang dituliskan sebagai p(n). Kita ingin membuktikan bahwa semua bilangan bulat yang .≥ bilangan basis (yakni n.≥ n_0) memenuhi pernyataan tersebut, maka perlu ditunjukkan
  1. Langkah Basis
  Dilakukan pembuktian terhadap nilai basis yakni n_0 pada pernyataan tersebut (n_0) memenuhi pernyataan.
  2. Langkah Induksi

Dilakukan pembuktian untuk semua bilangan bulat n $\geq n_0$ harus memenuhi pernyataan. Jika p(n) benar, p(n+1) benar.

Contoh induksi: (Munir, 2015: IV-4)

Untuk semua n ≥1, buktikan dengan induksi matematik bahwa $n^3 + 2n$adalah kelipatan 3

Penyelesaian:

(i) Basis. Untuk n = 1, diperoleh
$1^3 + 2(n) = 3$ dengan 3 adalah kelipatan 3 sehingga basis terbukti benar.

(ii) Langkah induksi. Pembuktian dari langkah pertama dijadikan hipotesis induksi. Kita harus menunjukkan bahwa

$$(n+1)^3 + 2(n+1)$$

adalah kelipatan 3. Dapat ditunjukkan sebagai berikut:

$$\begin{aligned}&(n+1)^3 + 2(n+1)\\ &= (n^3 + 3n^2 + 3n + 1) + (2n + 2)\\ &= (n^3 + 2n) + (3n^2 + 3n + 3)\\ &= (n^3 + 2n) + 3(n^2 + n + 1)\end{aligned}$$

Karena $n^3 + 2n$ adalah keliptan 3 (dari hipotesis induksi) dan $3(n^2 + n + 1)$ juga kelipatan 3, maka $(n^3 + 2n) + 3(n^2 + n + 1)$ adalah jumlah dua buah bilangan kelipatan 3. Dengan demikian $(n^3 + 2n) + 3(n^2 + n + 1)$juga kelipatan 3. Terbukti bahwa untuk $n \geq 1$, $n^3 + 2n$ adalah kelipatan 3.

III. PEMBAHASAN

Manusia yang membuat program (baik dalam notasi algoritmik maupun sudah dalam bahasa pemrograman tertentu) – selanjutnya akan disebut pemrogram memiliki jalan pikir tertentu yang menurutnya akan menyelesaikan masalah. Contoh sederhana dari pengulangan adalah mengubah bilangan 0 menjadi bilangan lain yang memiliki nilai di atas 0. Pemrogram membuat suatu struktur pengulangan. Hal yang sangat utama dibahas dalam makalah ini adalah pengecekan terhadap pengulangan tersebut.

Suatu badan pengulangan disebut berhasil bila memiliki akhir / terminasi, dan tujuan dari masuknya variabel dalam badan pengulangan tersebut tercapai

Dengan demikian, perlu dibuat fasilitas untuk mengecek apakah badan pengulangan tersebut bertemu pada terminasi dan tujuannya tercapai. Sesungguhnya, syarat minimal untuk menjalankan program adalah lulus sintaks (memiliki sintaks yang tepat sesuai bahasa yang digunakan) dan lulus semantik. Akibatnya, program masih bisa berjalan biarpun ada kesalahan pada badan pengulangan yang terdapat dalam algoritmanya.

Manusia terus berkembang dan semakin pandai memanfaatkan teknologi. Akibatnya, manusia ingin membuat program yang lebih baik. Kedua fasilitas di bawah dapat mengecek badan pengulangan yang ada pada program.

A. Loop Invariant (Pengulangan yang tidak berubah)

Loop invariant ditujukan untuk mengecek tujuan dari suatu badan pengulangan. Setelah mencari definisi dari beberapa sumber, dapat dikatakan bahwa pengulangan yang tidak berubah adalah suatu pernyataan yang selalu tetap dan bernilai benar sebelum, selama, dan sesudah badan pengulangan. Pernyataan ini melibatkan variabel – variabel yang akan diproses pada badan pengulangan. Pernyataan ini menjadi pedoman yang selalu dicocokkan dengan keadaan variabel – variabel tersebut. Pernyataan tersebut perlu dicocokkan pada setiap keadaan: sebelum masuk badan pengulangan, selama, dan sesudah badan pengulangan.

Hal yang sulit dilakukan dalam pengulangan adalah pengecekan satu – persatu keadaan variabel selama berjalannya badan pengulangan. Maka itu, dapat diterapkan induksi matematika untuk melakukan pengecekan tersebut. Untuk alasan tersebut, dapat dikatakan bahwa pengulangan yang tidak berubah adalah pernyataan yang dijadikan fungsi dalam pengecekan secara induksi matematika. Inti dari penggunaan pengulangan yang tidak berubah adalah mengecek apakah pengulangan tersebut sudah tepat/tidak. Bila ditemui ketidakcocokkan pada saat dibandingkan antara pernyataan dengan keadaan sebelum atau selama atau sesudah badan pengulangan, ada dua kemungkinan: pernyataan yang dipilih salah atau badan pengulangan tersebut yang salah. Dalam makalah ini diasumsikan (setelah melewati rangkaian proses terpilihlah) pernyataan yang selalu benar, maka ditunjukkan bahwa badan pengulangan tersebut salah. (Perkara memilih pernyataan yang selalu benar tidak dibahas dalam makalah ini)

B. Decrementing Function (Fungsi berkurang)

Tujuan fungsi berkurang adalah mengecek apakah suatu program menemui terminasinya. Fungsi berkurang ditentukan sendiri oleh pengamat. Kebanyakan, fungsi berkurang berlangsung secara rekursif. Fungsi berkurang awalnya memiliki nilai tertentu dan berkurang seiring dijalankannya badan pengulangan. Fungsi akan berhenti bila menemui nilai minimum. Pengecekan terminasi dilakukan saat fungsi bernilai minimum. Bila fungsi bernilai minimum dan pengulangan yang tidak berubah (loop invariant) bernilai benar maka ditemui terminasi.

Penjelasan ini akan lebih mudah dimengerti dengan contoh. Contoh dari pengecekan ini adalah perpangakatan. Contoh algoritma adalah

```
p = 1
i = 1
while (i!=N) do
        p = p*a
        i =  i+1
end while
//diyakini bahwa p = a^i
```

1. Penentuan pernyataan tidak berubah

Cara menentukan pernyatan tidak berubah adalah:

- Mencoba mencari pola dengan cara menjalankan pengulangan beberapa kali

  i=1, $p_i = 1*a = a$
  i=2, $p_i = a*a = a^2$

  i=3, $p_i = a^2*a = a^3$
  i=4, $p_i = a^3*a = a^4$
  Didapatkan kandidat pernyataan tidak berubah adalah: a^i
- Mengetes pernyataan tidak berubah
  $p_i = a^i$
  - Tes pada sebelum masuk badan pengulangan
    Menurut hasil inisialisasi program i = 1
    $p_1 = a^1 = a$
    Diperoleh hasil yang benar
  - Tes pada masuk badan pengulangan
    Pada langkah ini dilakukan induksi matematika
    Basis:
    i = 1 {asumsi N lebih besar dari 1}
    $p_1 = a^1 = a$
    Langkah induksi:
    Misal untuk i>=1, pernyataan $p_i = a^i$ adalah benar (hipotesis induksi). Akan diperlihatkan bahwa
    $p_{i+1} = a^{i+1}$
    $p_{i+1} = a^i * a = a^{i+1}$
    Karena langkah basis dan langkah induksi sudah diperlihatkan benar, terbutki bahwa untuk semua i>=1, $p_i = a^i$
  - Tes pada setelah keluar badan pengulangan
    Badan pengulangan berhenti saat i = N. Pada saat itu, maka
    $p_i = a^i = a^N$

Induksi yang dibuktikan pada langkah pengulangan menggunakan prinsip induksi sederhana. Dapat pula digunakan prinsip induksi yang dirampatkan, dengan tujuan mengetahui semua bilangan bulat $\geq$ bilangan basis. Pembuktian akan menjadi sebagai berikut:

1. Basis
   i = 1 {asumsi N lebih besar dari 1}
   $p_1 = a^1 = a$
2. Langkah induksi
   Andaikan $p_i = a_i$ benar (hipotesis induksi), maka harus ditunjukkan bahwa
   $p_n+1 = a_n+1$
   Dengan langkah sebagai berikut
   I = 1, I =2, … I = n
   $p_n+1 = a_n * a$
   Dari sebelumnya telah dibuktikan
   $p_n+1 = p_n * a$

Berdasarkan kedua langkah tersebut maka terbukti benar.

Setelah dilakukan ketiga tes tersebut, terbukti bahwa $p_i = a^i$ adalah pernyataan tidak berulang yang sesuai dengan keadaan pengulangan seperti ini. Sesungguhnya, bila diamati, keadaan yang sedang kita lakukan adalah menganalisis pengulangan dan membuat pernyataan tidak berulangnya. Pernyataan tidak berulang ini kemudian dijadikan basis untuk mengetes bila program menemui badan pengulangan serupa. Pada penerapan sesungguhnya dalam kehidupan nyata, program seharusnya sudah memiliki cara untuk memilih pernyatan tidak berubah ini secara otomatis, lalu pernyataan ini ditujukan untuk mengecek apakah tujuan / *goal* dari badan pengulangan tercapai. Pada keadaan sekarang ini, program hanya mengecek sintaks dan semantik dari kode. Sehingga yang dibutuhkan untuk menjalankan program adalah bila terminasi tercapai. Bila kondisi variabel dalam badan pengulangan berubah sedemikian hingga terminasi tercapai, program dinyatakan berjalan.

2. Penentuan Fungsi berkurang

Sama halnya dengan pernyataan tidak berubah, saat ini kita menganalisis badan pengulangan yang sudah ada dan menebak fungsi berkurangnya. Untuk sementara kita akan menebak badan fungsi berkurangnya adalah:

$$D(m, i) = m - i$$

Akan dilakukan pengecekan terhadap fungsi berkurang tersebut.

- Pengecekan terhadap nilai dari fungsi berkurang terhadap keberjalanan badan pengulangan

  $$D(m, i) = m - i$$
  $$i = i + 1$$

  //D ($m_{baru} - i_{baru}$) < D ($m_{lama} - i_{lama}$)
  Dari pernyataan ini, terbukti bahwa loop berkurang
- Perbandingan nilai minimum fungsi dan keberhentian pengulangan

  $$D(m, i) = m - i$$

  Nilai minimum adalah = 0, yakni saat m = i
  Kondisi ulang adalah selama nilai i tidak sama dengan m, maka saat i = m program keluar dari badan pengulangan.

  Menurut kedua langkah pengecekan di atas, fungsi berkurang yang dipilih sudah tepat. Sama halnya dengan pernyataan tidak berubah, fungsi berkurang ini dijadikan dasar bila ke depannya program menemui badan pengulangan yang sama lagi.

Kedua langkah ini diperlukan untuk mengecek setiap kondisi ulang dalam kode program. Bila hanya ada satu pengulangan ,cukup dibutuhkan pengecekan terhadap badan pengulangan tersebut. Pada kenyataannya, manusia tidak hanya menggunakan satu pengulangan pada satu program. Bisa saja ia melakukan banyak pengulangan, akibatnya setiap pengulangan memerlukan pengecekannya masing – masing. Bila ada lima pengulangan, perlu dilakukan pengecekan lima kali, bila ada sepuluh maka sepuluh kali pengecekan, dan seterusnya.

Optimasi terus dikembangkan oleh manusia. Hal yang dibutuhkan dalam pengembangan adalah kepastian bahwa setiap pengulangan memiliki terminasi. Bila tidak dijumpai terminasi, program akan terus menjalankan pengulangan tersebut karena ia tidak menemui keadaan akhir untuk berhenti / syarat untuk berhenti. Akibatnya saat dijalankan, dapat terjadi kesalahan / *error* yang dapat berupa pengulangan tak berhingga / *infinite loop.*

Dalam upaya mewujudkan itu, ada sebuah teori yang bernama teori himpunan / *set theory*. Dalam teori himpunan,dijabarkan bilangan ordinal, bilangan kardinal. Ada juga prinsip *transfinite induction. Transfinite* dapat diterjemahkan menjadi hal yang meyangkut ketidakberhinggaan. Bilangan *transfinite* adalah bilangan yang lebih besar dari bilangan *finite /* terhingga.

Induksi *transfinite* ditujukan untuk membuktikan bahwa sebuah program menjumpai keadaan berhentinya dari badan pengulangan / terminasi. Induksi *transfinite* dapat digunakan untuk memenuhi tujuan yang praktikal dalam pembuktian terminasi pengulangan ini. Induksi *transfinite* masih termasuk dalam induksi matematika. Induksi yang telah dibahas sebelumnya adalah induksi sederhana dan induksi yang dirampatkan. Keduanya membahas bilangan bulat dan berhingga, yang pada kasus tertentu akan terbatasi. Induksi *transfinite* bermain pada bilangan ordinal dan bilangan cardinal.

Secara umum, dapat dikatakan bahwa induksi *transfinite* digunakan untuk membuktikan pernayataan p(n) bernilai benar untuk semua n. Perbedaan muncul menyangkut bilangan yang dibahas. Bilangan yang dibahas pada induksi *transfinite* adalah ordinal, berbeda dengan bilangan bulat berhingga. Bilangan bulat berhingga pada suatu saat akan menemui batasnya. Induksi *transfinite* memilki hubungan erat dengan *well-ordered set* atau himpunan yang tertata baik yang tidak akan dibahas pada makalah ini. Bilangan ordinal sendiri digunakan untuk melambangkan cara menata koleksi objek. Bilangan ordinal akan menunjukkan kompleksitas dari pembuktian dan pengecekan terminasi program. Kompleksitas program dalam konteks pengulangan dan keadaan berhentinya berbanding lurus dengan bilangan ordinal.

IV. KESIMPULAN

Algoritma dalam program memiliki struktur pengulangan. Pada struktur pengulangan, ada perintah khusus yang menandai pengulangan tersebut dimulai dan menandai diakhiri. Ada kondisi khusus yang harus dipenuhi untuk terus menjalankan pengulangan.Keadaan mulai dan berhenti dari badan pengulangan perlu dicek agar program tidak salah. Pengecekan dapat dilakukan melalui *loop invariant /* pengulangan tidak berubah yang mengecek tujuan / *goal* dari badan pengulangan dan *decrementing function /* fungsi berkurang yang mengecek terminasi pengulangan.

VII. UCAPAN TERIMA KASIH

Ucapan terima kasih penulis sampaikan kepada Tuhan YME, atas penyertaan-Nya makalah ini bisa terselesaikan. Terima kasih kepada Pak Rinaldi Munir dan Ibu Harlili selaku dosen matematika diskrit untuk waktu dan ilmu yang diberikan. Terima kasih juga kepada dukungan moral dari orang tua dan teman.

REFERENSI

http://mathoverflow.net/questions/10334/what-practical-applications-does-set-theory-have
http://www-inst.cs.berkeley.edu/~cs170/fa14/tutorials/tutorial1.pdf
https://courses.cs.washington.edu/courses/cse331/12sp/lectures/lect03-reasoning-loops.pdf
http://www.cs.miami.edu/home/burt/learning/Math120.1/Notes/LoopInvar.html
Real Analysis, Lecture 26: Ordinal Numbers and Transfinite Induction: https://www.youtube.com/watch?v=TpnXBhaTERA
Munir, Rinaldi. *Matematika Diskrit*. Bandung:2003
Gambar 1: http://www.cs.kent.edu/~batcher/CS10051/c2.html
Gambar2:http://www.cs.kent.edu/~durand/CS0/Notes/Chapter02/algorithm_notes02.html
Gambar 3: http://askyourdaddy.blog.uns.ac.id/files/2014/09/efek-domino.png

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, 11 Desember 2015

Friska
13514042