Penerapan Algoritma Greedy dalam Algoritma Penjadwalan Prosesor Tunggal *Shortest Job First*

Girvandi Ilyas, 13515051
Program Studi Teknik Informatika
Sekolah Teknik Elekro dan Informatika
Insitut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
girvandip@gmail.com, 13515051@std.stei.itb.ac.id

***Abstract*—Dalam bidang studi Sistem Operasi, dipelajari tentang beberapa algoritma untuk menjadwalkan pekerjaan (job) yang harus diselesaikan oleh suatu prosesor tunggal. Salah satu algoritma penjadwalan prosesor tunggal adalah algoritma *shortest job first*. Algoritma ini mengutamakan pekerjaan yang bisa diselesaikan dengan waktu paling cepat. Dalam algoritma penjadwalan *shortest job first* terdapat penerapan algoritma *greedy* yang akan dijelaskan lebih lanjut dalam makalah ini.**

***Keywords*—SJF(*shortest job first*), algoritma, *greedy*, FCFS**

I. PENDAHULUAN

Di zaman modern ini, perkembangan komputer sudah sangat pesat dan digunakan dalam berbagai bidang kehidupan. Komputer digunakan untuk melakukan pekerjaan seperti mengetik dokumen, mengolah data, mendesain, ataupun digunakan untuk hiburan seperti menonton video atau bermain game. Namun, seringkali pengguna komputer hanya menggunakan komputer untuk menyelesaikan pekerjaannya tanpa mengetahui bagaimana komputer itu sendiri bekerja. Antarmuka yang menghubungkan antara pengguna dengan perangkat keras komputer adalah sistem operasi.

Gambar 1. Macam-macam sistem operasi

Sumber : http://3.bp.blogspot.com/--bHh2C5f9ag/VmVHtBNL85I/AAAAAAAAAYw/jbOpVk0RwZk/s1600/so1-copy.jpg

Sistem operasi adalah sebuah perangkat lunak yang digunakan untuk mengatur semua operasi-operasi dasar sistem seperti menjalankan aplikasi-aplikasi yang biasa digunakan pengguna untuk menyelesaikan masalah. Selain itu, sistem operasi juga mengatur semua umber daya yang dimiliki oleh komputer. Sumber daya tersebut berupa memori dan data-data yang ada didalam perangkat keras komputer. Sistem operasi merupakan kumpulan dari beberapa perintah atau program yang mengendalikan suatu komputer yang sudah disesuaikan dengan perangkat keras yang dimiliki. Ketika suatu komputer dinyalakan, maka sistem operasi akan dijalankan terlebih dahulu sebelum menjalankan perangkat lunak yang lainnya.

Salah satu tugas utama dari sistem operasi adalah melakukan manajemen proses, yaitu mengelola semua proses-proses yang datang untuk dikerjakan. Penjadwalan proses akan menentukan proses mana yang akan terlebih dahulu dikerjakan dan berapa lama proses tersebut akan dikerjakan oleh CPU. Penjadwalan proses yang dibahas dalam makalah ini adalah penjadwalan proses dalam prosesor tunggal. Ada beberapa algoritma yang dapat digunakan untuk melakukan penjadwalan proses dalam prosesor tunggal. Beberapa algoritma tersebut antara lain *first come first* serve (FCFS), *shortest job first* (SJF), *round robin*, dan penjadwalan dengan prioritas. Setiap algoritma mempunyai keunggulannya masing-masing. Namun, dalam makalah ini hanya akan dibahas algoritma SJF karena algoritma ini merupakan salah satu contoh penerapan algoritma *greedy*.

II. DASAR TEORI

A. *Algoritma Greedy*

Algoritma *greedy* merupakan salah satu algoritma yang paling banyak digunakan untuk memecahkan permasalahan optimasi. Kata *greedy* berasal dari bahasa inggris yang berarti tamak, rakus, atau serakah. Dalam menyelesaikan masalah optimasi, suatu algoritma dituntut untuk menghasilkan solusi yang paling optimal dari beberapa kemungkinan solusi. Ada dua macam persoalan optimasi, yaitu maksimasi dan minimisasi. Beberapa contoh masalah optimasi adalah pencarian jalur terdekat dari suatu tempat ke tempat yang lain (*shortest path*), persoalan penukaran uang, dan juga persoalan

memilih barang bernilai maksimum dengan batasan berat barang (*integer knapsack).*

Algoritma greedy memecahkan masalah langkah demi langkah. Pada setiap langkah diambil pilihan terbaik yang dapat diambil pada langkah tersebut. Prinsip algoritma *greedy* adalah "*take what you can get now*" yang berarti mengambil keputusan terbaik saat ini tanpa memikirkan konsekuensi kedepannya. Keputusan tersebut tidak dapat diubah lagi pada langkah berikutnya. Setiap langkah yang diambil membentuk solusi optimum lokal dengan harapan langkah-langkah berikutnya menuju ke solusi optimum global.

Algoritma *greedy* tidak selalu menghasilkan solusi optimum pada suatu persoalan karena pengambilan keputusan pada setiap langkah hanya memperhatikan kondisi pada langkah tersebut. Namun, algoritma greedy tetap merupakan algoritma yang baik untuk digunakan karena kecepatan dan ketepatannya dalam menentukan nilai optimum lokal. Algoritma ini memiliki kemungkinan sukses yang cukup baik untuk setiap persoalan optimasi.

Persoalan optimasi dalam konteks algoritma greedy disusun oleh elemen-elemen sebagai berikut :

1. Himpunan kandidat, C

   Himpunan ini berisi elemen-elemen pembentuk solusi. Contohnya adalah himpunan koin, himpunan *job* yang akan dikerjakan, himpunan simpul dalam graf, dll. Pada setiap langkah, satu buah kandidat diambil dari himpunannya.

2. Himpunan solusi, S

   Berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Dengan kata lain, himpunan solusi merupakan himpunan bagian dari himpunan kandidat

3. Fungsi seleksi

   Fungsi yang pada setiap langkah memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan

   Memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yaitu kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (*constraints*) yang ada. Kandidat yang layak dimasukkan kedalam himpunan solusi, sementara kandidat yang tidak layak dibuang dan tidak akan dipertimbangkan lagi.

5. Fungsi obyektif

   Fungsi yang memaksimumkan atau meminimumkan nilai solusi (misalnya panjang lintasan dan keuntungan).

Dengan elemen-elemen tersebut algoritma memiliki skema umum yang dapat dirumuskan sebagai berikut :

1. Inisialisasi S dengan kosong
2. Pilih sebuah kandidat (dengan fungsi seleksi) dari C
3. Kurangi C dengan kandidat yang sudah dipilih dari langkah (2) diatas
4. Periksa apakah kandidat yang dipilih tersebut bersama-sama dengan himpunan solusi membentuk solusi yang layak (diperiksa menggunakan fungsi layak). Jika layak, maka masukkan kandidat tersebut ke dalam himpunan solusi. Jika tidak layak, maka buang kandidat tersebut dan tidak perlu dipertimbankan lagi
5. Periksa apakah himpunan solusi sudah memberikan solusi yang lengkap (diperiksa menggunakan fungsi solusi). Jika sudah lengkap, maka berhenti. Jika belum lengkap, maka ulangi lagi dari langkah (2).

Berikut adalah beberapa contoh penggunaan algoritma greedy :

1. Minimisasi Waktu di dalam Sistem (penjadwalan)

   Sebuah server (dapat berupa processor, pompa, kasir di bank, dll) mempunyai n pelanggan (customer, client) yang harus dilayani. Waktu pelayanan untuk setiap pelanggan i adalah ti. Minimumkan total waktu didalam sistem.

   Tiga pelanggan dengan

   t1 = 5, t2 = 10, t3 = 3,

   Enam urutan pelayanan yang mungkin :

   ==========

   Urutan *T*

   ==========

   1, 2, 3: 5 + (5 + 10) + (5 + 10 + 3) = 38

   1, 3, 2: 5 + (5 + 3) + (5 + 3 + 10) = 31

   2, 1, 3: 10 + (10 + 5) + (10 + 5 + 3) = 43

   2, 3, 1: 10 + (10 + 3) + (10 + 3 + 5) = 41

   3, 1, 2: 3 + (3 + 5) + (3 + 5 + 10) = 29 ← (optimal)

   3, 2, 1: 3 + (3 + 10) + (3 + 10 + 5) = 34

   ==========

   Algoritma :

   - Urutkan pelanggan berdasarkan waktu pelayanannya, mulai dari yang terkecil hingga yang terbesar
   - Pada setiap langkah pilihlah pelanggan yang memiliki waktu pelayanan terkecil diantara pelanggan lain yang belum dilayani

   Elemen-elemen *greedy* dalam persoalan tersebut adalah :

   - C = {t1, t2, t3}
   - S = {}
   - Fungsi Seleksi = Pelanggan dengan waktu pelayanan terkecil

- Funsgi kelayakan = tidak ada
- Fungsi Objektif = Total waktu melayani semua pelanggan minimum

Pada masalah minimisasi waktu dalam sistem, algoritma greedy menghasilkan solusi optimum global.

2. Integer Knapsack

Terdapat sebuah beberapa barang, setiap barang memiliki berat dan nilai. Terdapat sebuah wadah dengan berat maksimal yang dapat dimuat. Tentukan barang mana saja yang dapat dimasukkan kedalam wadah sehingga mendapatkan nilai sebanyak-banyaknya

Adapun elemen-elemen *greedy* pada persoalan tersebut adalah :

- C = {barang$_1$, barang$_2$, …, barang$_n$}
- S = {}
- Fungsi seleksi bergantung pada algoritma greedy yang digunakan :
    - *Greedy by Profit* = barang yang mempunyai nilai terbesar
    - *Greedy by Weight* = barang yang mempunyai berat paling ringan
    - *Greedy by Density* = barang yang memiliki p_i/w_i terbesar
- Fungsi kelayakan = jumlah berat lebih kecil atau sama dengan kapasitas wadah
- Fungsi Objektif = mendapatkan nilai sebanyak-banyaknya

Contoh :

$w_1 = 100$; $p_1 = 40$; $w_2 = 50$; $p_2 = 35$; $w_3 = 45$; $p_3 = 18$;

$w_4 = 20$; $p_4 = 4$; $w_5 = 10$; $p_5 = 10$; $w_6 = 5$; $p_6 = 5$;

Properti Objek				Greedy by			
i	wi	pi	pi/wi	profit	weight	density	Solusi Optimal
1	100	40	0,4	1	0	0	0
2	50	35	0,7	0	0	1	1
3	45	18	0,4	0	1	0	1
4	20	4	0,2	0	1	1	0
5	10	10	1	0	1	1	0
6	5	2	0,4	0	1	1	0
Total Bobot				100	80	85	100
Total Keuntungan				40	34	51	55

Dari hasil penggunaan 3 teknik *greedy* tidak satupun dapat menghasilkan solusi yang optimal

Dari beberapa contoh diatas, dapat terlihat bahwa kekurangan dari algoritma *greedy* adalah algoritma ini dapat tidak berhasil untuk setiap kasus. Selain itu solusi optimum lokal terkadang tidak mengarah ke solusi optimum global. Sementara kelebihan algoritma *greedy* adalah penyelesaian masalahnya berjalan dengan cepat karena tidak memikirkan konsekuensi kedepannya. Kemudian, pengimplementasian algoritma *greedy* dapat dikatakan lebih mudah dibandingkan algoritma lain.

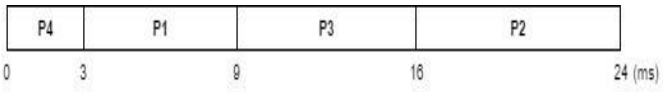
B. Algoritma Penjadwalan Prosesor Tunggal "Shortest Job First"

Algoritma *shortest job first* merupakan salah satu algoritma penjadwalan prosesor tunggal. Algoritma ini menyesuaikan dengan waktu penyelesaian setiap proses yang diterima. Saat CPU sudah tersedia, maka CPU akan mengerjakan proses yang memiliki waktu penyelesaian yang lebih cepat. Jika ada dua proses yang memiliki waktu penyelesaian yang sama, maka digunakan algoritma FCFS yang akan mengerjakan proses yang terlebih dahulu diterima.

Proses	Waktu Penyelesaian (ms)
P1	6
P2	8
P3	7
P4	3

Tabel 1. Contoh kumpulan proses dan waktu penyelesaiannya

Contoh diatas merupakan kumpulan dari beberapa proses dan waktu penyelesaiannya. Pertama-tama proses yang akan dikerjakan adalah proses P4 karena memiliki waktu penyelesaian yang paling kecil diantara yang lainnya. Kemudian proses P1 akan dilakukan karena memiliki waktu penyelesaian yang paling kecil diantara yang lainnya. Demikian seterusnya sampai semua proses telah dikerjakan. Waktu tunggu untuk masing-masing proses adalah 3 milisekon untuk proses P1, 16 milisekon untuk proses P2, 9 milisekon untuk proses P3, dan 0 milisekon untuk proses P4. Ini berarti rata-rata waktu tunggu proses-proses tersebut adalah (3 + 16 + 9 + 0) / 4 = 7 milisekon. Waktu tunggu rata-rata yang dihasilkan oleh algoritma ini lebih cepat apabila dibandingkan dengan waktu tunggu rata-rata algoritma lain. Sebagai contoh jika diterapkan algoritma FCFS, maka waktu tunggu rata-rata yang dihasilkan adalah (0 + 6 + 14 + 21) / 4 = 10.5



Gambar 2. Gant-Chart proses non-preemptive

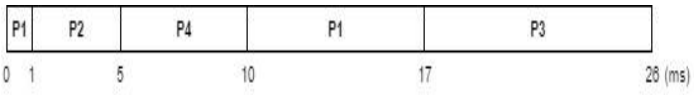
Implementasi dari algoritma *shortest job first* dapat digunakan untuk dua situasi yaitu *preemptive* dan *non-preemptive*. Pada situasi *preemptive* proses-proses bisa dikerjakan bergantian tanpa menyelesaikan proses yang sedang

dikerjakan terlebih dahulu. Proses dikerjakan sesuai waktu kedatangan, Misal jika ada suatu proses yang datang dan memiliki waktu penyelesaian yang lebih cepat dibandingkan dengan proses yang sedang dikerjakan, maka CPU akan beralih untuk mengerjakan proses tersebut terlebih dahulu dan meninggalkan proses yang sebelumnya.

Proses	Waktu kedatangan (ms)	Waktu penyelesaian (ms)
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Tabel 2. Contoh kumpulan proses, waktu kedatangan proses, dan waktu penyelesaiannya

Pertama-tama proses yang akan dikerjakan adalah proses P1, karena hanya ada proses 1 dalam antrian pekerjaan. Satu detik setelah itu datang proses P2 dengan waktu penyelesaian 4 detik, maka CPU beralih menyelesaikan proses P2. Pada detik ke dua dan ketiga datang proses P3 dan P4. Namun, proses-proses tersebut lebih lama waktu penyelesaiannya daripada proses P2, maka P2 terus diselesaikan. Setelah menyelesaikan proses P2 CPU beralih ke proses P4 dengan waktu penyelesaian 5 detik. Terakhir, CPU menyelesaikan proses P1 kemudian P3.



Gambar 3. Gant-Chart proses *preemptive*

Algoritma penjadwalan *shortest job first* terbukti cukup optimal karena dapat menghasilkan waktu tunggu rata-rata yang minimum untuk sekian banyak proses. Mengerjakan proses yang lebih cepat waktu penyelesaiannya terlebih dahulu mengurangi waktu menunggu dari setiap proses yang lainnya. Oleh karena itu, waktu tunggu rata-rata dapat berkurang.

III. PENERAPAN ALGORITMA GREEDY DALAM PENJADWALAN *SHORTEST JOB FIRST*

Algoritma *greedy* dalam algoritma penjadwalan *shortest job first* dilakukan pada setiap detik/milisekon (bergantung pada waktu yang digunakan). Suatu proses dapat datang untuk meminta dikerjakan oleh CPU dalam waktu yang acak. Prosesor harus bisa menentukan proses mana yang dikerjakan pada setiap saat untuk menghasilkan solusi yang optimal. Oleh karena itu, untuk memilih proses mana yang harus dikerjakan, algoritma greedy akan mengevaluasi proses mana yang memiliki waktu penyelesaian paling minimum saat ini.

Elemen-elemen greedy dalam implementasi algoritma *shortest job first* adalah sebagai berikut :

1. Himpunan kandidat

   Himpunan kandidat dari persoalan ini adalah semua proses-proses yang harus dikerjakan oleh CPU

2. Himpunan solusi

   Himpunan yang seluruh prosesnya telah dikerjakan

3. Fungsi seleksi

   Memilih proses yang memiliki waktu penyelesaian paling minimum

4. Fungsi layak

   Memeriksa apakah proses yang dikerjakan melebihi jumlah proses yang ada

5. Fungsi obyektif

   Waktu menunggu rata-rata dari setiap proses minimum

Salah satu contoh permasalahan penjadwalan proses yang dapat diselesaikan dengan algoritma greedy adalah masalah penjadwalan *preemptive.* Diberikan sekumpulan proses dengan waktu kedatangan dan waktu penyelesaiannya sebagai berikut :

Proses	Waktu kedatangan (ms)	Waktu penyelesaian (ms)
P1	0	5
P2	2	8
P3	3	1
P4	6	4
P5	10	6

Tabel 3. Kumpulan proses, waktu kedatangan proses, dan waktu penyelesaiannya

Persoalan tersebut dapat dikerjakan dengan algoritma *shortest job first preemptive* yang bisa dimodelkan menjadi algoritma *greedy*. Langkah pertama adalah pada detik 0-1 hanya ada proses P1 sehingga proses tersebut dikerjakan. Pada detik kedua datang proses baru

Langkah 1 (0 ms) :

Proses	Waktu kedatangan (ms)	Sisa Waktu penyelesaian (ms)
P1	0	5

Pada waktu 0-1ms hanya ada proses P1 yang telah datang untuk dikerjakan sehingga tidak perlu melakukan seleksi untuk memilih proses yang dikerjakan.

Langkah 2 (2 ms) :

Proses	Waktu kedatangan (ms)	Sisa Waktu penyelesaian (ms)
P1	0	3

P2	2	8

Pada waktu 2ms datang proses P2 yang melakukan pemintaan CPU. Namun, proses P1 masih memiliki waktu penyelesaian yang lebih kecil dibandingkan proses P2 sehingga proses P1 tetap dikerjakan.

Langkah 3 (3 ms) :

Proses	Waktu kedatangan (ms)	Sisa Waktu penyelesaian (ms)
P1	0	2
P2	2	8
P3	3	1

Pada waktu 3ms datang proses P3 dengan waktu penyelesaian 1 ms. CPU akan berpindah melakukan proses P3 terlebih dahulu karena waktu penyelesaiannya yang lebih kecil.

Langkah 4 (4 ms) :

Proses	Waktu kedatangan (ms)	Sisa Waktu penyelesaian (ms)
P1	0	2
P2	2	8

Proses P3 telah selesai dikerjakan, maka CPU beralih mengerjakan proses 1 kembali hingga selesai.

Langkah 5 (6 ms) :

Proses	Waktu kedatangan (ms)	Sisa Waktu penyelesaian (ms)
P2	2	8
P4	6	4

Proses P1 telah selesai dikerjakan. Pada saat P1 selesai dikerjakan datang lagi sebuah proses P4 dengan waktu penyelesaian 4ms. Waktu penyelesaian ini lebih kecil dari yang dimiliki oleh P2 sehingga P4 lah yang dikerjakan terlebih dahulu dan P2 harus menunggu lagi.

Langkah 6 (10 ms) :

Proses	Waktu kedatangan (ms)	Sisa Waktu penyelesaian (ms)
P2	2	8
P5	10	6

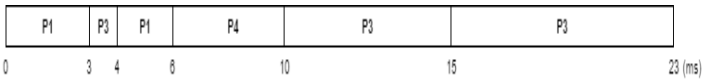
Proses P4 telah selesai dikerjakan. Pada saat P4 selesai dikerjakan datang lagi sebuah proses P5 dengan waktu penyelesaian 6ms. Waktu penyelesaian ini lebih kecil dari yang dimiliki oleh P2 sehingga P5 dikerjakan terlebih dahulu dan P2 harus menunggu lagi.

Langkah 7 (16 ms) :

Proses	Waktu kedatangan (ms)	Sisa Waktu penyelesaian (ms)
P2	2	8

Pada waktu 16ms hanya tersisa proses P2 sehingga P2 dikerjakan hingga selesai oleh CPU.

Gant-Chart yang dihasilkan dari persoalan tersebut :



Gambar 3. Gant-Chart penerapan algoritma greedy pada contoh persoalan

IV. ANALISIS

Implementasi algoritma *greedy* dalam persoalan penjadwalan tersebut menghasilkan waktu tunggu rata-rata setiap proses (1 + 13 + 0 + 0 + 0) / 5 = 2.8. Hasil ini dapat dikatakan sangat baik dalam mengelola 5 buah proses yang datang dengan waktu yang berbeda-beda. Jika dibandingkan dengan algoritma penjadwalan *first come first serve* (FCFS) yang memiliki waktu tunggu rata-rata setiap proses (0 + 5 + 10 + 8 + 8) / 5 = 6.2, maka algoritma *shortest job first* yang merupakan implementasi dari algoritma *greedy* dapat dikatakan lebih baik. Untuk setiap persoalan penjadwalan *preemptive* algoritma *greedy* selalu memberikan hasil yang lebih baik daripada algoritma FCFS.

Namun, dalam pengimplementasian algoritma *greedy* ada suatu fenomena yang terjadi yaitu *starvation*. *Starvation* adalah suatu fenomena dimana suatu proses harus menunggu dalam waktu yang lama untuk mendapatkan jatah pengerjaan dari CPU. Dalam contoh yang sudah diberikan diatas, proses P2 mengalami *starvation* karenan harus menunggu selama 13 ms sebelum dikerjakan oleh CPU. Padahal, proses P2 sudah meminta pengerjaan sejak dari waktu 2 ms. Ini merupakan salah satu kekurangan dari implementasi algoritma *greedy* pada penjadwalan proses.

V. KESIMPULAN

Salah satu strategi algoritma yaitu algoritma *greedy* ternyata telah banyak sekali diimplementasikan dalam persoalan optimasi. Bahkan dalam satu bidang ilmu yaitu informatika, algoritma *greedy* digunakan untuk mengimplementasikan penjadwalan prosesor yang merupakan fokus dari bidang studi Sistem Operasi.

Algoritma penjadwalan prosesor tunggal *shortest job first* pada dasarnya adalah algoritma *greedy* yang digunakan untuk melakukan penjadwalan proses-proses dalam CPU. Algoritma *shortest job first* dapat dikatakan sebagai algoritma yang baik untuk penjadwalan proses dalam prosesor tunggal. Namun, algoritma ini terkadang tidak memberikan solusi yang optimal karena sifat dari solusi algoritma *greedy* yang belum tentu optimal.

VI. UCAPAN TERIMA KASIH

Pertama penulis mengucapkan terima kasih kepada Tuhan yang Maha Esa karena atas rahmat dan bimbingan-Nya penulis dapat menyelesaikan makalah ini. Kemudian Penulis berterima kasih kepada kedua orangtua penulis dan teman-

teman penulis yang senantiasa mendukung dan memberikan masukan dalam mengerjakan makalah ini.

Penulis juga berterima kasih kepada dosen IF 2211 yaitu Pak Rinaldi Munir, Ibu Masayu Leylia Khodra, dan Ibu Nur Ulfa Maulidevi yang telah memberikan materi Strategi Algoritma yang diterapkan dalam penulisan makalah ini. Tidak lupa juga penulis berterima kasih kepada pihak-pihak yang membantu penulis dalam menyelesaikan makalah ini.

REFERENSI

[1] Munir, Rinaldi, 2017. Diktat Kuliah Strategi Algoritma. Bandung : Program Studi Teknik Informatika Institiut Teknologi Bandung

[2] http://www.zonasiswa.com/2014/10/sistem-operasi-komputer-pengertian.html

[3] Silberchatz, Abraham, 2012. "Operating System Concepts, 9th Edition"

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, 18 Mei 2017

Girvandi Ilyas
13515051