Aplikasi Decrease and Conquer dalam Pencegahan Kekaosan Tugas Besar Informatika Semester 4

Muhammad Hanif Adzkiya / 13517120
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517120@std.stei.itb.ac.id

***Abstrak*—Decrease and Conquer adalah metode pemecahan masalah yang dapat meminimalkan jumlah langkah eksekusi yang diperlukan untuk menyelesaikan masalah. Tugas besar Informatika dapat diselesaikan secara efektif jika pengerjaan sesuai dengan prioritas. Bagaimana *Decrease Conquer* dapat menyelesaikan permasalahan kekaosan tugas besar informatika semester 4 ?**

***Kata Kunci*—Decrease and Conquer, Priority Queue, Binary Search, Tree**

I. Pendahuluan

Kehidupan perkuliahan tidak terlepas dari tugas, termasuk di Teknik Informatika Institut Teknologi Bandung. Pada pelaksanaan perkuliahan di Teknik Informatika, untuk menguji pemahaman dan kreativitas mahasiswa dalam menerapkan materi kuliah yang diajarkan, setiap mata kuliah memberikan tugas besar.

Pada semester 4, mata kuliah yang diajarkan adalah Strategi Algoritma, Basis Data, Dasar Rekayasa Perangkat Lunak, Pemrograman Berbasis Objek, Probabilitas dan Statistika, dan Sistem Operasi. Setiap mata kuliah tersebut memberikan tugas besar dengan frekuensi masing - masing.

Di akhir semester, hampir semua tugas besar memiliki batas waktu pengumpulan yang berdekatan. Hal ini membuat banyak mahasiswa terpaksa bergadang di malam sebelum pengumpulan. Di sisi lain, sebenarnya waktu yang diberikan untuk mengerjakan tugas besar cukup lama.

Sebagai seorang mahasiswa yang sudah sedikit banyak menghadapi dinamika kehidupan kampus, penulis menyadari ada suatu cacat besar dalam otak penulis yang selalu menganggap semua masalah adalah hal sepele dan cenderung untuk menunda-nunda bahkan lari dari masalah — akademik dan/atau kemahasiswaan — tersebut. Istilah dari gangguan psikologis ini sering disebut dengan procrastination. Procrastination ini sendiri, secara arti, adalah sebuah kecenderungan untuk menunda suatu pekerjaan hingga kesempatan untuk bertindak itu habis. Orang dengan kecenderungan seperti ini biasa disebut procrastinator atau kadang deadliner.

Makalah ini akan membahas solusi terhadap kesulitan mahasiswa memberikan prioritas pengerjaan tugas besar melalui paradigma *decrease and conquer*.

II. DASAR TEORI

2.1 Tugas Besar

Tugas besar (Tubes) adalah salah satu istilah yang digunakan pada perkuliahan di Teknik Informatika ITB. Istilah ini merujuk pada tugas yang harus diselaikan mahasiswa diakhir bab / semester untuk menguji pemahaman mahasiswa, sekaligus sebagai sarana pelatihan mahasiswa untuk menerapkan materi yang diajarkan dalam bentuk yang lebih nyata.

2.1.1 Atribut Tugas Besar

Setiap tugas besar di Informatika ITB memiliki 3 atribut yaitu :

1. Nama mata kuliah

   Nama mata kuliah adalah nama mata kuliah yang memberikan tugas besar kepada mahasiswa. Nama mata kuliah yang diadakan pada semester 4 adalah Strategi Algoritma (Stima), Basis Data (Basdat), Dasar Rekayasa Perangkat Lunak (DRPL), Pemrograman Berbasis Objek (OOP), Probabilitas dan Statistika (Probstat), dan Sistem Operasi (OS).

2. Nama tugas besar

   Nama tugas besar adalah identifikator tugas besar mata kuliah yang bersangkutan. Berikut nama tugas besar setiap mata kuliah yang diajarkan pada semester 4 :

   A. Strategi Algoritma : Tubes 1, Tubes 2, dan Tubes 3

B. Basis Data : Milestone 1
C. Dasar Rekayasa Perangkat Lunak : SKPL, Use Case, Kelas Diagram, PDHUPL, dan Revisi
D. Pemrograman Berbasis Objek : Tubes 1A, Tubes 1B, Tubes 2
E. Probabilitas dan statistika : Tubes 1
F. Sistem Operasi : Milestone 1, Milestone 2, Milestone 3

3. Deadline pengumpulan

Deadline pengumpulan adalah tenggat batas waktu maksimal pengumpulan tugas besar.

2.1.2 Tips Pengerjaan Tugas Besar

Dalam tips pengerjaan tugas seperti yang dilansir himasif.ilkom.unej.ac.id, cara pengerjaan tugas yang baik adalah :

1. Niat
2. Melist tugas
3. Kerjakan secara urut dari yang mendekati *deadline*
4. Mencari referensi pengerjaan tugas
5. Mencoret tugas yang sudah dikerjakan
6. Bersemangat dalam mengerjakan tugas baru

2.2 Decrease and Conquer

2.2.1 Pengertian

Decrease and conquer adalah metode desain algoritma dengan mereduksi persoalan menjadi beberapa sub-persoalan yang lebih kecil, tetapi selanjutnya hanya memproses satu sub-persoalan saja.

2.2.2 Langkah - Langkah

Langkah - langkah dalam penyelesaian dengan paradigma *decrease and conquer* umumnya terbagi menjadi 3 langkah :

1. Mengurangi permasalahan ke sub permasalahan yang lebih kecil dari masalah yang sama dan memperluas solusi.
2. Menyelesaikan permasalahan dengan memecahkan sub permasalahan yang lebih kecil.
3. Perluas solusi sub permasalahan yang lebih kecil untuk mendapatkan solusi untuk permasalahan asli.

Ide dasar teknik *decrease and conquer* didasarkan pada mencari hubungan antara solusi untuk permasalahan tertentu dan solusi untuk sub permasalahan yang lebih kecil. Pendekatan ini juga dikenal sebagai pendekatan inkremental atau induktif.

2.2.3 Variasi

Dilihat dari cara pembagian permasalahan menuju sub permasalahan yang lebih kecil, *decrease and conquer* dapat dibagi menjadi 3 variasi :

A. Decrease by Constant

Dalam variasi ini, ukuran permasalahan dikurangi dengan konstanta yang sama pada setiap iterasi algoritma. Biasanya, konstanta ini sama dengan satu, meskipun pengurangan ukuran konstan lainnya dapat terjadi sesekali. Cara penyelesaian yang menggunakan *decrease by constant* adalah *Insertion Sort, Depth First Search, Breadth First Search,* dan *Topological Sorting*

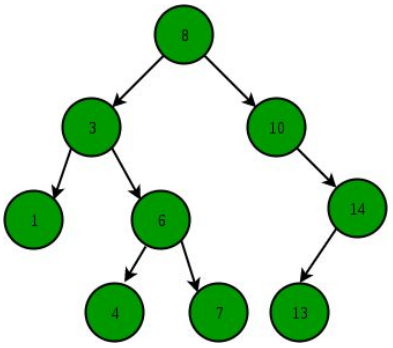
B. *Decrease by Constant Factor*

Teknik ini menyarankan pengurangan *instance* masalah dengan faktor konstan yang sama pada setiap iterasi algoritma. Dalam sebagian besar aplikasi, faktor konstan ini sama dengan dua. Pengurangan dengan faktor selain dua sangat jarang terjadi.

Penurunan oleh algoritma faktor konstan sangat efisien terutama ketika faktor tersebut lebih besar dari 2

2.3 *Binary Search Tree*

2.3.1 Pengertian



Gambar 1. Binary Search Tree

Binary Search Tree adalah pohon biner yang masing-masing simpul internal menyimpan kunci (dan secara opsional, nilai yang terkait) dan masing-masing memiliki dua sub-pohon yang berbeda, dilambangkan kiri dan kanan. Pohon memenuhi properti pencarian biner, yang menyatakan bahwa kunci di setiap simpul harus lebih besar atau sama dengan kunci apa pun yang disimpan di sub-pohon kiri, dan kurang dari atau sama dengan kunci apa pun yang disimpan di sub-pohon kanan. Daun (simpul akhir) dari pohon tidak mengandung sub-poboh berikutnya dan tidak memiliki struktur untuk membedakannya satu sama lain.

2.3.2 Operasi

Binary Search Tree memiliki 3 operasi yaitu :

A. Searching

Mencari nilai simpul *Binary Serach Tree* untuk nilai tertentu dapat diprogram secara rekursif atau berulang.

Dimulai dengan memeriksa simpul akar. Jika pohon adalah nol, kunci yang dicari tidak ada di pohon. Jika tidak, jika kunci sama dengan akar, pencarian berhasil dan dikembalikan nilai simpul. Jika kunci kurang dari akar, dicari subtree kiri. Demikian pula, jika kunci lebih besar dari pada akar, dicari subtree kanan. Proses ini diulang sampai kunci ditemukan atau subtree yang tersisa adalah nol. Jika kunci yang dicari tidak ditemukan setelah subtree nol tercapai, maka kunci tersebut tidak ada di pohon. Ini mudah dinyatakan sebagai algoritma rekursif .

B. *Insertion*

Penyisipan dimulai dari simpul akar. Jika kuncinya tidak sama dengan akar, dicari pada subpohon kiri atau kanan seperti sebelumnya. Akhirnya, kita akan mencapai simpul eksternal dan menambahkan pasangan nilai kunci baru sebagai anak kanan atau kirinya, tergantung pada kunci simpul tersebut. Dengan kata lain, diperiksa dimulai dari simpul akar dan secara rekursif menyisipkan simpul baru ke subtree kiri jika kuncinya kurang dari akar, atau *subtree* kanan jika kuncinya lebih besar dari atau sama dengan akar.

C. *Deletion*

Operasi *deletion* dimulai dari simpul akar. Operasi dilakukan secara rekursif dengan mengunjungi simpul anak dari simpul akar. Jika simpul kanan lebih besar dari simpul akar operasi berlanjut ke arah *subtree* kanan dari akar, sebaliknya operasi berlanjut ke arah simpul kiri dari akar. Jika nilai sama dengan kunci yang akan dihapus, simpul akan dihapus dari *Binary Serach Tree*.

2.3.3 Kelebihan

Binary Search Tree banyak digunakan dalam mengimplementasikan suatu struktur data pada pemrograman.

Berikut berapa keuntungan memiliki struktur data *Binary Search Tree* :

A. BST menampilkan data secara struktur

B. BST memberikan proses *Insertion, Deletion, dan Searching* secara optimal

C. Pohon yang dibentuk bersifat fleksibel

2.3.5 Kompleksitas

Kompleksitas terbaik dari pencarian *binary search tree* adalah o(1). Kompleksitas ini diperoleh ketika akar dari binary search tree merupakan elemen yang dicari. Kompleksitas terburuk dari pencarian binary search tree adalah ketika seluruh simpul berada di kiri atau kanan. Kompleksitas pada kasus ini adalah pencarian diharuskan menelusuri semua simpul sehingga kompleksitas berjumlah o(n). Kompleksitas rata – rata adalah ketika simpul di sisi kiri dan kanan seimbang. Kompleksitas rata – rata memiliki waktu sebesar o(log n).

2.4 *Priority Queue*

2.4.1 Pengertian

Dalam ilmu komputer, *priority queue* adalah tipe data abstrak seperti struktur data *queue*, tetapi setiap elemen memiliki "prioritas" yang telah ditentukan. Dalam *priority queue*, elemen dengan prioritas lebih tinggi disajikan sebelum elemen dengan prioritas lebih rendah. Dalam beberapa implementasi, jika dua elemen memiliki prioritas yang sama, mereka dilayani sesuai dengan urutan di mana mereka masuk, sedangkan dalam implementasi lainnya, penempatan elemen dengan prioritas yang sama tidak diperbolehkan.

2.4.2 Operasi

Priority Queue memiliki 3 operasi yaitu *top, pop,* dan *push*. Operasi *top* adalah mengetahui nilai yang memiliki prioritas tertinggi pada *priority queue*. Operasi *pop* adalah mengeluarkan nilai yang memiliki prioritas tertinggi pada *priority queue*. Operasi *push* adalah memasukkan elemen baru pada *priority queue*.

2.4.3 Implementasi

Priority Queue dapat diimplementasikan melalui :

A. Implementasi Naif

Ini merupakan implementasi dengan cara yang paling simpel. Implementasi naif memiliki kompleksitas yang tidak efisien dalam melakukan operasi *pop* elemen. Setiap elemen yang baru ditambahkan disimpan dalam array yang tidak terurut. Setiap perintah pop dipanggil, proses yang dilakukan adalah mencari nilai terbesar dalam array tersebut lalu mengembalikan nilai tersebut.

Kompleksitas dalam melakukan pemasukan elemen adalah o(1) namun kompleksitas mengembalikan nilai pop adalah o(n) karena harus dilakukan iterasi terhadap keseluruhan elemen.

B. Implementasi *Binary Search Tree*

Implementasi yang lebih baik dilakukan dengan menggunakan *binary serach tree*. Implementasi yang dilakukan adalah :

1. *Push*

Elemen yang akan disisipkan akan ditaruh pada simpul yang tepat menggunakan penyisipan pada *Binary Search Tree*

2. *Pop*

Pengambilan elemen dilakukan dengan cara mengambil elemen paling kanan menggunakan modifikasi fungsi pencarian pada *Binary Search Tree*.

Kompleksitas dalam implementasi Binary Search Tree yaitu mirip dengan teori Binary Search Tree yaitu o(log n).

III. Isi dan Pembahasan

3.1 Ide Penyelesaian

Permasalaan Tugas Besar dapat diselesaikan dengan *priority queue* dengan menetapkan sistem prioritas sebagai tugas dengan *deadline* terdekat.

3.2 Operasi

Dalam proses penyelesaian tugas besar, terdapat 3 operasi yang dapat dikerjakan, yaitu :

A. Menambah Tubes baru

Operasi tambah tubes baru dapat diselesaikan dengan cara melakukan perintah *push* pada *priority queue* pada daftar tugas besar

Operasi ini dapat diimplementasikan dalam bentuk *binary search tree*. Tugas baru yang akan ditambahkan dibandingkan dengan tugas pada simpul akar. Jika tugas baru memiliki *deadline* pengumpulan lebih dekat dari pada simpul akar, maka secara rekursif akan menelurusi bagian subtree kiri dari simpul akar, sebaliknya jika tugas baru memiliki .deadline pengumpulan lebih jauh dari simpul akar, maka secara rekursif akan menelurusi *subtree* kanan dari simpul akar. Penelusuran dilakukan secara rekursif hingga simpul mencapai simpul daun dari *binary search tree*. tubes baru akan ditempatkan sesuai dengan posisi yang tepat sebagai simpul anak dari simpul daun terakhir.

B. Melihat Tubes yang harus diselesaikan

Karena tubes yang harus diselesaikan adalah tubes yang memiliki *deadline* terdekat, maka ketika operasi ini dijalankan akan mengeluarkan tubes yang memiliki *deadline* terdekat.

Operasi ini dapat diimplementasikan menggunakan operasi searching pada *Binary Search Tree*. Pencarian dimulai dari simpul akar. Dikarenakan, operasi ini haruslah mencari tubes dengan deadline tanggal terdekat, pencarian akan dilakukan mengunjungi subtree dari simpul akar hingga mencapai simpul daun. Jika sudah mencapai simpul daun, kembalikan tubes yang berada di simpul daun dan inisialisasikan sebagai tubes yang harus diselesaikan.

C. Menyelesaikan Tubes

Operasi menyelesaikan tubes akan menghapus tubes yang memiliki deadline terdekat dalam *priorty queue* tugas besar.

Operasi ini dapat diimplementasikan menggunakan operasi *delete* pada *Binary Search Tree*. Pencarian dimulai dari simpul akar. Dikarenakan, operasi ini haruslah mencari tubes dengan *deadline* tanggal terdekat, pencarian akan dilakukan mengunjungi subtree dari simpul akar hingga mencapai simpul daun. Jika sudah mencapai simpul daun, kembalikan tubes yang berada di simpul daun dan *delete* tubes tersebut.

3.3 Kompleksitas

Ide penyelesaian dari permasalahan tugas besar adalah menggunakan priority queue yang diimplementasikan dalam bentuk *binary search tree* .

Setiap operasi pada penyelesaian tugas besar sama dengan operasi pada *Binary Search Tree*, yaitu membagi permasalahan menjadi 2 sub permasalahan yang lebih kecil dengan ukuran pembagian selalu sama dengan konstan faktor sama dengan 2. Kompleksitas *Decrease and Conquer* dengan konstan faktor sama dengan 2 adalah o(log n)

3.4 Implementasi dalam Bahasa Pemrograman

IV. Hasil Eksperimen

4.1 Data Tugas Besar Teknik Informatika Semester 4

Dalam penyusunan makalah ini, penulis mengambil data tugas besar teknik informatika semester 4 tahun ajaran 2018/2019. Tugas besar yang dijadikan sampel data adalah tugas besar yang memiliki tenggat waktu deadline pengumpulan pada bulan April 2019.

Berikut daftar tugas besar teknik informatika semester 4 tahun ajaran 2018/2019 :

Mata Kuliah	Nama Tubes	Deadline
Stima	Tubes 3	22 April 2019 jam 12
Basdat	Milestone 4	26 April 2019 jam 18
OS	Milestone 3	24 April 2019 jam 23
DRPL	PDHUPL	27 April 2019 jam 22

Tabel 1. Daftar Tubes Informatika Semester 4

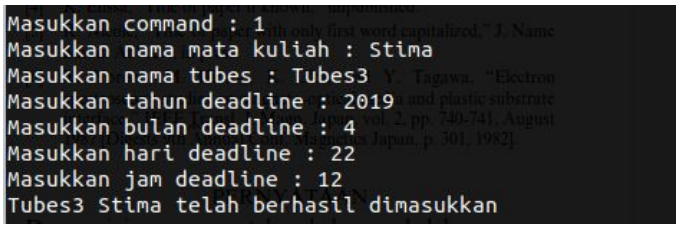
4.2 Simulasi Pengerjaan Tugas Besar

Penulis melakukan simulasi pengerjaan tugas besar melalui program yang telah dibuat. Daftarperintah yang dapat dilakukan adalah :

1 : Tambah Tubes Baru ke dalam daftar
2 : Lihat Tubes yang harus diselesaikan sekarang
3 : Konfirmasi tubes sekarang selesai

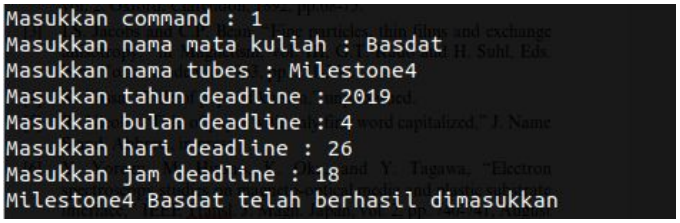
Skenario yang dibuat adalah :

1. Masukan Tubes baru berupa Tubes 3 Stima dengan deadline pengumpulan 22 April 2019 jam 12.00



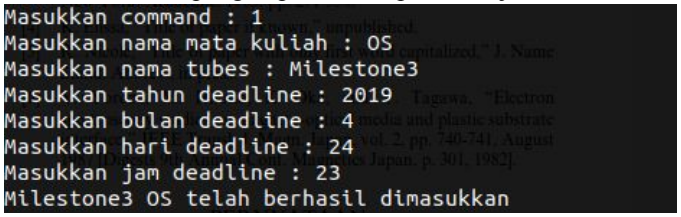
Gambar 2. Simulasi 1

2. Masukan Tubes baru berupa Milestone 4 Basdat dengan deadline pengumpulan 26 April 2019 jam 18.00



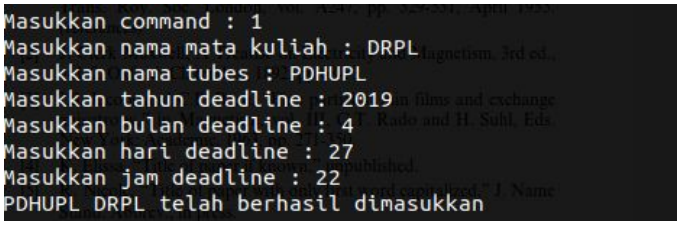
Gambar 3. Simulasi 2

3. Masukan Tubes baru Milestone 3 OS dengan deadline pengumpulan 24 April 2019 jam 23.00



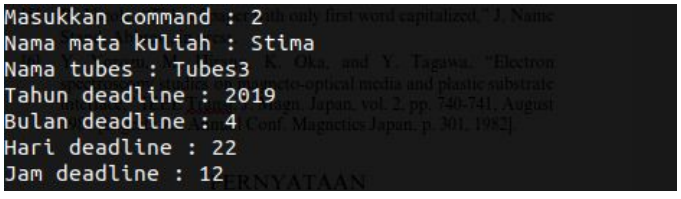
Gambar 4. Simulasi 3

4. Masukan Tubes baru PDHUPL DRPL dengan deadline pengumpulan 27 April 2019 jam 22.00



Gambar 5. Simulasi 4

5. Lihat Tubes yang harus diselesaikan

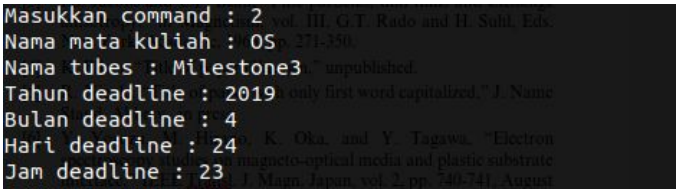


Gambar 6. Simulasi 5

6. Konfirmasi Tubes sekarang selesai

Gambar 7. Simulasi 6

7. Lihat Tubes yang harus diselesaikan

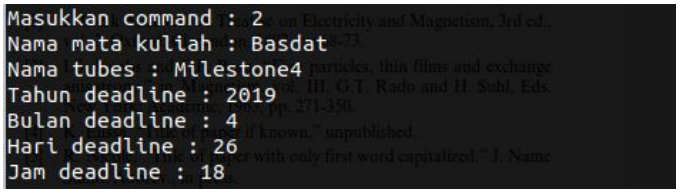


Gambar 8. Simulasi 7

8. Konfirmasi Tubes sekarang selesai

Gambar 9. Simulasi 8

9. Lihat Tubes yang harus diselesaikan



Gambar 10. Simulasi 9

10. Konfirmasi Tubes sekarang selesai

Gambar 11. Simulasi 10

V. Simpulan

Kekaosan tugas besar mahasiswa Teknik Informatika 4 dapat diatasi dengan menyusun skala prioritas pengerjaan. Salah satu parameter yang dapat dipakai adalah prioritas bedasarkan *deadline* pengumpulan yang terdekat.

Dalam ilmu komputer, hal ini dapat diterapkan dalam bentuk struktur data *priority queue*. Untuk mengoptimalkan pemrosesan *query*, dapat diimplementasikan dalam bentuk *binary search tree* yang merupakan salah satu teknik penyelesaian masalah dengan paradigma *decrease and conquer*.

VI. Ucapan Terimakasih

Pertama, penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena berkat rahmat-Nya penulis dapat menyelesaikan makalah dengan judul "Aplikasi Decrease and Conquer dalam Pencegahan Kekaosan Tugas Besar Informatika Semester 4" ini. Tak lupa, penulis juga ingin berterima kasih kepada semua pihak yang telah mendukung penulisan makalah ini, baik secara langsung maupun secara tidak langsung. Penulis berterimakasih kepada Pak Rinaldi Munir sebagai pengampu mata kuliah strategi algoritma kelas K-3.

Referensi

[1] S. Halim and F. Halim, "Competitive Programming, 3rd Edition" Singapore, April 2013.

[2] S. Robert and W. Kevin, "Algorithms (4th Edition)", Addison-Wesley Professional, Maret 2011

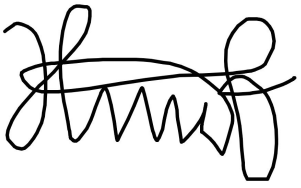
[3] Himasif, Bingung Tugas Kuliah yang Menumpuk ? Ini nih kita kasih tips yang bisa jadi kiat – kiat kalian dalam mengerjakan tugas kuliah, http://himasif.ilkom.unej.ac.id/2018/04/01/bingung-tugas-kuliah-yang-menumpuk-ini-nih-kita-kasih-tips-yang-bisa-jadi-kiat-kiat-kalian-dalam-mengerjakan-tugas-kuliah

[4] M. Rinaldi, Algoritma dan Pemrograman Ed. 6, Penerbit Informatika, Februari 2016

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, 26 April 2019



Muhammad Hanif Adzkiya 13517120