Pemanfaatan Algoritma Greedy dalam Scheduling Kegiatan Sehari-hari

William Rukmansa - 13516066
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika ITB
Bandung, Indonesia
13516066@std.stei.itb.ac.id

***Abstrak*—Dalam kehidupan yang dinamis dan semakin kompleks, manusia seringkali dihadapkan dengan lebih dari satu kegiatan yang merupakan kewajiban ataupun keinginan orang yang bersangkutan. Dalam hal tersebut, tidak semua orang memiliki kemampuan manajemen waktu yang baik untuk melakukan semua kegiatan tersebut. Selain itu, ada orang yang juga kesulitan untuk mengingat kegiatan-kegiatan yang hendak dilakukan. Salah satu bentuk penyelesaian terhadap masalah tersebut adalah penggunaan sebuah aplikasi *to-do-list* atau sejenisnya, misal aplikasi kalender pada *smartphone*. Ada banyak cara untuk menyelesaikan masalah penjadwalan kegiatan yang hendak dilakukan, salah satunya adalah dengan menggunakan algoritma *greedy* dalam menjadwalkan kegiatan dengan menggunakan konsep dasar sebuah aplikasi penjadwalan. Pada umumnya, algoritma *greedy* tidak selalu memberikan hasil optimal, namun algoritma tersebut dapat memudahkan orang dalam membuat jadwal.**

Kata Kunci—penjadwalan, scheduling, greedy

I. PENDAHULUAN

Kita hidup di era modern. Era tersebut memperkenalkan kita kepada tantangan dan masalah kehidupan yang berbeda dibandingkan era dahulu. Masalah dan tantangan tersebut cenderung lebih banyak, lebih kompleks, dan dapat berubah di waktu yang akan datang. Kita dituntut agar mampu beradaptasi dengan perkembangan zaman dan melakukan berbagai hal dalam kehidupan kita. Tidak semua orang dapat melakukan semua hal yang ingin dilakukannya dalam suatu jangka waktu tertentu, dan tidak semua orang pula dapat memanajemen waktu yang ada untuk melakukan semua kegiatan yang diinginkan. Terlebih lagi, rencana manusia tidak selalu berhasil dilaksanakan, sehingga manusia sering perlu menyusun ulang rencananya untuk waktu ke depan.

Pada era-era sebelumnya, kertas dan pena adalah alat yang efektif untuk mengingatkan kita akan kegiatan-kegiatan yang ingin dilakukan di tengah kesibukkan hidup. Di era saat ini, teknologi telah berkembang pesat sehingga kita dapat menggunakan ponsel pintar atau laptop kita seperti sebuah kertas pengingat, dilengkapi dengan notifikasi berupa *reminder* dan berbagai fitur lain. Konsep lain yang juga membantu kita melakukan aktivitas yang kita inginkan adalah *scheduler*, dimana kita memberikan suatu jangka waktu untuk setiap aktivitas yang kita catat. Biasanya kegiatan-kegiatan yang dicatat dengan aplikasi tersebut merupakan kegiatan-kegiatan penting, sebab kegiatan yang kurang penting biasanya tak menentu dan dapat berubah waktu pelaksaannya.

Jika dalam sehari atau suatu jangka waktu tertentu ada berbagai kegiatan yang ingin dilakukan, tetapi tidak semua dapat dilakukan dalam jangka waktu tersebut, kita sering menggunakan algoritma *greedy* dalam memilih kegiatan-kegiatan yang ingin dilakukan. Biasanya, orang akan mempertimbangkan kegiatan yang ingin dilakukan berdasarkan prioritasnya, yaitu seberapa penting atau mendesak pelaksaan suatu kegiatan. Dengan mengimplementasikan prioritas kepada setiap kegiatan, aplikasi *scheduler* dapat bekerja dengan lebih efektif.

Beberapa orang memiliki keterampilan kurang dalam memanajemen waktu. Hal ini dapat menjadi masalah bagi orang-orang yang ingin memaksimalkan waktu yang ada. Algoritma *greedy* dapat membantu banyak dalam pemanfaatan waktu bebas yang ada dengan menyusun urutan-urutan kegiatan secara otomatis bagi pengguna, berdasarkan urutan prioritas suatu kegiatan. Keunggulan menggunakan strategi tersebut adalah penerapannya sederhana dan pengguna akan dapat melakukan kegiatan-kegiatan yang penting dan tetap dapat melakukan kegiatan lain yang kurang penting dengan waktu bebasnya. Selain itu, apabila terjadi perubahan rencana kegiatan sehari-hari, penyusunan ulang jadwal tidaklah rumit. Apabila terdapat kegiatan-kegiatan yang tidak dapat diselesaikan dalam jangka waktu tertentu, penyusunan jadwal dapat diibaratkan seperti sebuah *knapsack problem*, dimana yang diinginkan adalah banyak kegiatan berprioritas tinggi yang diselesaikan dahulu dalam jangka waktu tersebut.

II. DASAR TEORI

A. Algoritma Greedy

Algoritma *greedy*, juga dikenal sebagai algoritma “rakus”, adalah algoritma yang memilih suatu solusi atau bagian solusi yang terbaik berdasarkan kriteria yang dipilih atau fungsi optimasi yang digunakan. Algoritma greedy memiliki prinsip “*take what you can get now*”. Pemilihan solusi terbatas hanya pada ruang lingkupnya, artinya solusi yang terbaik untuk tahap penyelesaian suatu masalah belum tentu akan menghasilkan solusi yang terbaik untuk penyelesaian permasalahan secara menyeluruh.

Algoritma *greedy* adalah algoritma yang sering dipakai dalam persoalan-persoalan optimasi, yaitu persoalan-persoalan yang membutuhkan suatu hasil paling efektif dari suatu himpunan solusi yang ada. Ada dua macam persoalan optimasi, yaitu:

1. Maksimasi, misalnya memasukkan benda sebanyak mungkin ke dalam sebuah tas, atau mengerjakan sebanyak mungkin tugas sebelum tenggat waktu.
2. Minimasi, misalnya mencari rata-rata waktu tunggu paling sedikit dalam melayani tamu, atau jumlah lembaran uang paling sedikit dalam memberi uang kembalian.

Dalam persoalan optimasi dikenal elemen berupa:

- himpunan kandidat, yaitu himpunan seluruh kemungkinan yang dipilih,
- himpunan solusi, yaitu himpunan penyelesaian masalah,
- fungsi seleksi, yaitu fungsi yang digunakan untuk pemilihan suatu solusi,
- kendala, atau lebih dikenal sebagai *constraint*, yaitu pembatas atau syarat suatu solusi diterima atau ditolak, biasanya digunakan dalam fungsi kelayakan,
- fungsi objektif, yaitu fungsi yang mengoptimasikan suatu solusi, biasanya menyatakan tujuan algoritma.

B. Contoh Persoalan: Knapsack Problem

Gambar 1. Ilustrasi Knapsack Problem

Persoalan *knapsack problem* adalah persoalan yang digambarkan sebagai suatu permasalahan mendapat keuntungan terbesar dari benda-benda yang akan dimasukkan ke dalam sebuah tas. Benda memiliki bobot berat dan nilai harga. Tas memiliki suatu kapasitas berat maksimum. Solusi yang diinginkan adalah himpunan benda-benda yang muat ke dalam tas dan menghasilkan keuntungan tertinggi. Walaupun digambar demikian, *knapsack problem* dapat digunakan untuk mengibaratkan permasalahan lain dengan parameter berbeda. Misalnya, dengan sejumlah uang, seseorang ingin membeli barang-barang dengan total berat maksimum.

Persoalan tersebut merupakan salah satu persoalan yang tidak dapat ditemukan algoritma yang menyelesaikannya dalam waktu polinomial, atau waktu yang sewajarnya. Oleh karena itu, penyelesaian persoalan tersebut menggunakan algoritma yang dikatakan hampir mencapai solusi terbaik yang diinginkan. Salah satu bentuk penyelesaiannya adalah dengan menggunakan algoritma *greedy.*

Algoritma *greedy* memberikan tiga jenis fungsi seleksi untuk persoalan *knapsack*, yaitu berdasarkan keuntungan, bobot, atau densitas. Fungsi seleksi pertama memilih benda dengan keuntungan tertinggi dahulu, jika dimasukkan ke dalam tas tidak melebihi kapasitas tas, benda tersebut dimasukkan ke dalam himpunan solusi. Fungsi seleksi kedua memilih benda berdasarkan berat paling ringan dahulu. Fungsi seleksi ketiga memilih benda dengan rasio keuntungan dengan bobot yang tertinggi.

Contoh soal *knapsack probem* :

Seseorang ingin menambahkan benda-benda ke dalam tas berkapasitas 19. Berikut adalah data benda-benda yang ada.

Karakteristik benda			
Jenis	Berat	Keuntungan	Densitas
A	3	12	4
B	10	20	2
C	6	15	2.5
D	7	21	3

Tabel 1. Ciri-ciri benda untuk soal

Penyelesaian dengan algoritma *greedy*:

Misalkan kita menggunakan fungsi seleksi berdasarkan keuntungan. Maka, masukkan dulu benda dengan keuntungan tertinggi, yaitu D. Total berat benda di dalam tas adalah 7. Selanjutnya, dipilih benda B, sehingga total berat benda menjadi 17. Selanjutnya, dipilih benda C, namun total berat melebihi kapasitas tas. Sama halnya ketika memilih benda A. Sehingga himpunan solusinya adalah benda D dan benda B, dengan total keuntungan 41.

Solusi tersebut cukup bagus, walaupun lumayan jauh dari solusi terbaik, yaitu A, C, dan D, dengan total keuntungan 48. Jika kita memilih fungsi seleksi berdasarkan densitas, maka kita akan mendapatkan solusi optimum.

Secara lebih terperinci, cara kerja algoritma *greedy* untuk persoalan *knapsack* adalah:

1. Pilih kandidat dari himpunan kandidat berdasarkan fungsi seleksi.

2. Periksa apakah kandidat tersebut jika ditambahkan ke dalam himpunan solusi melanggar kendala-kendala yang berlaku. Jika tidak melanggar, kandidat ditambahkan ke dalam himpunan solusi. Jika melanggar, kandidat dikeluarkan dari himpunan kandidat (tidak dianggap).
3. Ulangi langkah 1 dan 2 sampai tas sudah muat penuh atau himpunan kandidat sudah kosong.

Jika benda-benda sudah terurut berdasarkan ciri-ciri yang dibutuhkan fungsi seleksi, maka kompleksitas algoritmanya adalah *O*(n).

C. *Kelebihan dan Kekurangan Algoritma Greedy*

Kelebihan dari algoritma *greedy* adalah:

- Mudah diimplementasikan dan tidak terlalu rumit
- Umumnya mudah dimengerti banyak orang
- Solusi dapat mendekati solusi paling optimal

Kekurangan dari algoritma *greedy* adalah:

- Algoritma tersebut mungkin tidak bekerja dengan benar jika memilih fungsi seleksi yang tidak tepat
- Tidak selalu memberikan solusi optimal atau solusi yang mendekati solusi optimal

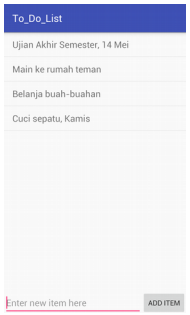
III. Penerapan Algoritma Greedy Dalam Scheduling

Penjadwalan kegiatan adalah sesuatu tugas yang tak selalu mudah dilakukan seseorang. Terkadang, kehidupan yang dijalani seseorang cukup rumit sehingga menjadwalkan kegiatan-kegiatan sehari-hari seseorang secara manual cukup memakan banyak waktu. Berikut adalah beberapa metode *scheduling* yang dapat diterapkan untuk membantu menjalani hidup.

A. *To-do-list*

Sebuah *to-do-list* adalah suatu daftar kegiatan-kegiatan yang harus dilakukan atau ingin dilakukan. Daftar tersebut berguna bagi banyak orang sebab di tengah kesibukkan, seseorang cenderung lupa kegiatan apa yang hendak dilakukan selanjutnya. Selain itu, beberapa orang memiliki daya ingat yang terbatas sehingga suatu daftar kegiatan membantunya memutuskan kegiatan apa yang hendak dilakukan. Secara tersirat, mereka yang memakai *to-do-list* mencatat hal-hal yang penting namun mungkin terlupakan atau terlewatkan.

Suatu to-do-list merupakan bentuk penjadwalan kegiatan yang paling sederhana. Beberapa variasinya menambahkan suatu waktu tenggat atau waktu mulai di sebelah kegiatan yang bersangkutan. Kelemahan daftar tersebut adalah segala sesuatu dipertimbangkan secara manual, seperti kegiatan apa dulu yang harus dilakukan, jadwal yang saling tumpang tindih, dan perubahan-perubahan rencana pada masa yang akan datang.



Gambar 2. Contoh aplikasi to-do-list

B. *Priority Scheduling*

Penjadwalan dengan cara tersebut memanfaatkan skala prioritas untuk mengatur kegiatan apa saja yang hendak dilakukan. Prioritas yang dimaksud dapat diartikan berbeda oleh orang berbeda. Pada umumnya, prioritas suatu kegiatan akan lebih tinggi apabila:

- Ada tenggat waktu
- Suatu kewajiban atau keharusan
- Sangat diinginkan oleh penyusun jadwal

Contoh kasus penjadwalan:

Seseorang hari ini memiliki waktu kosong dari jam 13.00 sampai jam 18.00. Dia ingin melakukan beberapa kegiatan, namun beberapa dari kegiatan tersebut harus dilakukan sebelum jam tertentu. Berikut adalah tabel kegiatan dan waktu dimana kegiatan tersebut masih dapat dilakukan. Asumsi semua kegiatan membutuhkan waktu paling lama 1 jam untuk diselesaikan.

	13.00	14.00	15.00	16.00	17.00	18.00
A						
B						
C						
D						
E						

Tabel 2. Tenggat waktu tiap kegiatan untuk soal

Maka dengan algoritma *greedy*, gunakan tenggat waktu sebagai fungsi seleksi, dimana dipilih kegiatan dengan tenggat waktu terdekat terlebih dahulu. Setelah itu, apabila waktu saat itu belum melewati tenggat waktu kegiatan tersebut, penyusun jadwal menambahkan kegiatan tersebut ke dalam daftar kegiatan pada jangka waktu tersebut. Jika hanya salah satu dari dua kegiatan yang dapat dilakukan (misal, kegiatan B dan D pada tabel), maka penyusun jadwal akan memilih kegiatan yang prioritas yang lebih tinggi (prioritas tersebut merupakan prioritas yang ditetapkan oleh pengguna).

Contoh kasus tersebut adalah kasus yang telah disederhanakan. Pada kenyataannya, waktu yang diperlukan untuk melakukan suatu kegiatan bervariasi, pula jadwal waktu kosong seseorang tidak selalu berderetan, misalnya seseorang memiliki waktu bebas dari jam 7 sampai jam 8, lalu kuliah pada jam 8 dan pulang pada jam 5, lalu memiliki waktu kosong lagi dari jam 5 sampai jam 9.

Oleh karena itu, penyusun jadwal harus menyusunnya sesuai dengan keinginan pengguna: apakah yang ada tenggat waktu harus selesai duluan, atau pengguna ingin melakukan sebanyak mungkin kegiatan tanpa menghiraukan prioritas suatu kegiatan.

Apabila tujuan pengguna adalah melakukan sebanyak mungkin kegiatan dalam waktu tertentu, maka kasus tersebut menjadi seperti kasus *knapsack problem*, hanya saja "tas" yang digunakan lebih banyak daripada sebelumnya.

Contoh kasus penjadwalan:

Seseorang memiliki sejumlah kegiatan yang jarang ia lakukan dalam kehidupan sehari-hari. Berikut adalah tabel data kegiatan dan waktu kosong yang ia miliki.

Kegiatan	Lama (jam)		Waktu Kosong	
A	2		08.00	
B	1		09.00	
C	3		10.00	
D	2		11.00	
E	4		12.00	
F	1		13.00	
G	1		14.00	
H	3		15.00	
			16.00	
			17.00	
			18.00	

Tabel 3. Lama kegiatan dan waktu kosong untuk soal

Algoritma untuk menentukan penjadwalakan adalah menempatkan kegiatan dengan waktu pelaksanaan paling pendek ke dalam kelompok waktu kosong yang paling kecil dahulu, apabila muat. Dalam contoh tersebut, kegiatan B akan mengisi jam 18.00 sampai jam 19.00, kegiatan F dan G akan mengisi jam 08.00 sampai jam 10.00, kegiatan A dan D akan mengisi jam 13.00 hingga jam 17.00.

Jika tujuannya bukan mengerjakan sebanyak mungkin kegiatan, melainkan menyelesaikan kegiatan dengan semakin lama suatu kegiatan, semakin tinggi prioritasnya, maka susunan kegiatannya akan berbeda. Dengan fungsi tersebut, E akan mengisi waktu jam 13.00 sampai 17.00, C atau H akan mengisi waktu jam 08.00 sampai jam 11.00, B, F, atau G akan mengisi jam 18.00 sampai jam 19.00.

Seringkali, seseorang menjadwalkan kegiatan-kegiatannya dengan prioritas saja tidak cukup, sehingga ada *to-do-list* yang berkelompok, dimana tiap kelompok memiliki prioritas tertentu dan penjadwalan suatu kegiatan didasarkan oleh prioritas kelompok yang ia tempati. Penjadwalan dengan pengelompokkan tersebut biasanya adalah solusi praktis untuk memisahkan kegiatan-kegiatan yang penting dengan kegiatan-kegiatan yang kurang penting.

Dengan pengelompokkan, penyusunan jadwal dibagi menjadi dua tahap, masing-masing menerapkan algoritma *greedy* dengan berbeda. Tahap pertama: mengisi jadwal dengan kegiatan-kegiatan yang dikelompokkan ke kelompok kegiatan berprioritas tinggi atau wajib, dengan fungsi seleksi berupa tenggat waktu terdekat. Tahap kedua: dengan waktu sisa yang ada, isi jadwal dengan kegiatan-kegiatan dari kelompok yang berprioritas rendah dengan tujuan melakukan sebanyak mungkin kegiatan yang ada.

Kelebihan menggunakan metode penyusunan kegiatan tersebut adalah jadwal akan terlihat lebih teratur dan terencana, kelemahannya adalah pengguna harus menetapkan batas-batas waktu bebas dan tak bebasnya dalam membuat jadwal, serta ada lebih banyak pertimbangan dari pengguna mengenai prioritas suatu kegiatan dan tujuan pengguna membuat penjadwalan tersebut.

C. Priority Scheduling Improvements

Sejauh ini, banyak aplikasi perangkat lunak yang telah berhasil menerapkan berbagai variasi dari konsep *to-do-list* untuk membantu kehidupan kita sehari-hari, namun penyusunan dan pertimbangan prioritas suatu kegiatan diserahkan kepada pengguna. *Priority scheduling* yang telah dijelaskan sebelumnya cukup membantu pengguna mengambil keputusan, namun keefektifannya masih dapat ditingkatkan lagi dengan cara pengguna menentukan waktu-waktu kosong yang ia biasa miliki setiap harinya dalam seminggu, lalu perangkat lunak mencatat data tersebut, kemudian menyusun jadwal berdasarkan kendala normal tersebut secara otomatis bagi pengguna. Apabila ada kendala khusus, pengguna dapat membuat suatu kegiatan berprioritas tertinggi untuk mengisi jadwal waktu kosong yang ia miliki pada hari dan waktu yang bersangkutan.

Contoh kasus yang telah disederhanakan:

Seseorang ingin menghabiskan Sabtu dan Minggu dengan melakukan berbagai kegiatan. Berikut data waktu kosongnya pada hari-hari tersebut (asumsi pada waktu normal, ia bangun dan sudah siap pada jam 08.00 dan bersiap untuk tidur jam 20.00).

	Sabtu	Minggu
08.00		
09.00		
10.00		
11.00		
12.00		
13.00		
14.00		
15.00		
16.00		
17.00		
18.00		
19.00		
20.00		

Tabel 4. Waktu kosong untuk soal (tanda hitam artinya tidak kosong)

Berikut adalah data kegiatan-kegiatan yang ingin dia lakukan untuk Sabtu Minggu ini.

Kegiatan	Prioritas	Tenggat	Durasi (jam)
A	Tinggi	Sabtu, 15.00	2
B	Tinggi	-	3
C	Rendah	Minggu, 10.00	3
D	Rendah	Minggu, 13.00	1
E	Sedang	-	2
F	Tinggi	Sabtu, 18.00	3
G	Rendah	Minggu, 16.00	2
H	Sedang	-	4
I	Sedang	Sabtu, 17.00	1

Tabel 5. Data kegiatan untuk soal

Maka dengan menggunakan algoritma *greedy*, pada tahap pertama akan dilihat hanya kegiatan-kegiatan berprioritas tinggi dan memiliki waktu tenggat terlebih dahulu. Dengan fungsi seleksi berupa tenggat waktu terdekat, maka penyusunan jadwal pada tahap pertama akan menampilkan hasil sebagai berikut.

	Sabtu	Minggu
08.00	A	
09.00	A	
10.00	F	
11.00	F	
12.00	F	
13.00		
14.00		
15.00		
16.00		
17.00		
18.00		
19.00		
20.00		

Tabel 6. Tabel jadwal tahap pertama

Lalu pada tahap kedua dan seterusnya, fungsi objektif dan fungsi seleksi tergantung keinginan dan tujuan pengguna. Misalkan untuk kegiatan berprioritas sedang dan rendah, pengguna ingin menyelesaikan kegiatan yang memiliki tenggat waktu sama seperti tahap pertama. Maka hasil proses tahap kedua dan ketiga adalah sebagai berikut.

	Sabtu	Minggu
08.00	A	D
09.00	A	G
10.00	F	G
11.00	F	
12.00	F	
13.00	I	
14.00		
15.00		
16.00		
17.00		
18.00	C	
19.00	C	
20.00	C	

Tabel 7. Tabel jadwal tahap kedua dan ketiga

Kemudian, untuk kegiatan yang tidak memiliki tenggat waktu, pengguna ingin memasukkan sebanyak mungkin kegiatan untuk mengisi waktu kosong, terlepas dari prioritasnya. Dalam kasus tersebut, perangkat lunak gagal menempatkan kegiatan H pada jadwal, dan sebaiknya memberikan pesan peringatan kepada pengguna apakah ingin merubah jadwal secara manual atau tidak.

	Sabtu	Minggu
08.00	A	D
09.00	A	G
10.00	F	G
11.00	F	
12.00	F	
13.00	I	
14.00	E	
15.00	E	
16.00		
17.00		
18.00	C	B
19.00	C	B
20.00	C	B

Tabel 8. Tabel jadwal tahap terakhir

Cara penyelesaian tersebut hanyalah salah satu cara dari berbagai macam cara penjadwalan yang dapat dilakukan, tergantung fungsi seleksi yang dipakai dan pemisahan tahap-

tahap penyusunan jadwal. Oleh karena itu, kelemahan metode ini adalah bagaimana pengguna menentukan fungsi seleksi untuk tiap kelompok akan mempengaruhi jadwal yang dihasilkan, pula pengisian waktu kosong yang dimiliki pengguna yang berbeda juga dapat menghasilkan hasil yang berbeda. Namun, keunggulan utama dari menggunakan metode penjadwalan tersebut adalah kemudahan dalam mengatur kegiatan-kegiatan yang ada, waktu yang dihematkan dalam menyusun jadwal serta membuat keputusan menjadi lebih mudah.

IV. KESIMPULAN

Sebagaimana algoritma *greedy* pada umumnya, penerapan algoritma tersebut dalam *scheduling* tidak selalu menghasilkan jadwal yang memanfaatkan waktu secara maksimal. Namun, algoritma tersebut seringkali berhasil mempermudah seseorang dalam menyusun jadwal serta memilih kegiatan yang perlu dilakukan berdasarkan kategori prioritas kegiatan. Dengan kekuatan teknologi, jika terjadi perubahan pada jadwal, data kegiatan, atau data waktu kosong, perangkat lunak mampu menampilkan hasil yang baru dengan cukup cepat. Karena kesederhanaannya, kompleksitas algoritma tersebut untuk penjadwalan pada umumnya adalah $O(n)$.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan yang memampukan manusia untuk dapat menyelesaikan makalah tersebut. Penulis juga mengucapkan banyak terima kasih kepada dosen pengampu mata kuliah IF2211 Strategi Algoritma di kelas K-03, Bapak Rinaldi Munir, atas pengajarannya dan bimbingannya sehingga penulis pada akhirnya dapat menulis makalah tersebut. Penulis juga mengucapkan terima kasih kepada orang tua, saudara dan teman-teman yang senantiasa mendukung penulis dalam perkuliahan maupun pembuatan makalah tersebut. Selain itu, penulis juga memohon maaf apabila makalah masih jauh dari sempurna, penulis terbuka untuk kritik dan saran agar dapat membuat makalah yang lebih baik ke depannya.

REFERENSI

[1] R. Munir, Diktat Bahan Kuliah, bab 3-4, 2004.

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, 14 Mei 2018

William Rukmansa - 13516066