Penerapan Algoritma Greedy pada pengelolaan reservasi di restoran

Muhammad Gerald Akbar Giffera - 13520143
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520143@std.stei.itb.ac.id

***Abstrakt*—Algoritma greedy merupakan algoritma yang menerapkan heuristic untuk mengoptimalaan solusinya. Pada dasarnya tujuan dari algoritma greedy adalah untuk memaksimalkan atau meminimalisasi suatu solusi dari sebuah permasalahan. Algoritma ini akan dicoba untuk diimplementasikan untuk mengoptimalisasi penempatan tempat duduk dari reservasi sebuah restoran. Walaupun persoalan ini tampaknya tidak rumit, namun nyatanya seteleah berbincang dengan beberapa pemilik restoran, permasalahan ini seringkali dihadapi sehingga kapasitas pelanggan yang bisa ditampung tidak mencapai jumlah yang maksimal. Algoritma yang dikembangkan pada makalah ini bertujuan untuk mengoptimalkan penempatan pelanggan, sehingga restoran bisa terpenuhi sepenuh mungkin.**

Kata kunci—greedyt; manajemen restoran; algoritma

I. PENDAHULUAN

Sebuah restoran tentunya tidak hanya menerima pelanggan secara *walk-in,* terutama pada restoran-restoran ternama tentunya akan ada penerimaan pelanggan melalui reservasi dimana pelanggan bisa memesan tempat untuk memastikan dirinya mendapatkan tempat di restoran tersebut. Tentunya hal ini sangat menguntungkan bagi pelanggan, sehingga pelanggan tidak perlu cemas bahwa mereka tidak akan mendapatkan tempat di restoran yang mereka mau. Akan tetapi, hal ini dapat menimbulkan masalah bagi pihak restoran.

Apabila demand reservasi dari pelanggan berjumlah banyak, seperti pada bulan puasa ataupun jika restoran tersebut memang sanagat terkenal, maka pengaturan penempatan duduk bagi para pelanggan yang melakukan reservasi tidak akan semudah biasanya. Akan ada banyak factor yang mempengaruhi pengaturan tersebut seperti jumlah satu reservasi yang mungkin tidak sesuai dengan kapasitas meja, ataupun request penempatan tempat duduk yang harus bersebelahan dan banyak lainnya. Sehingga, jika mengandalkan kemampuan logika dan pikiran saja, maka akan ada kemungkinan dimana penempatan tempat duduk tersebut bisa jadi tidak optimal dan banyak meyisakan kursi kosong. Jika hal tersebut terjadi maka baik pihak pelanggan maupun pengelola restoran pun akan menaglami kerugian mereka masing-masing.

Untuk meminimalisasi kesalahan pada penempatan ini, maka pada makalah ini, akan dicoba untuk dikembangkan algoritma yang meminimalisasi jumlah tempat yang masih kosong pada dari sebuah rangkaian reservasi yang diterima. Jenis algoritma yang digunakan adalah algortima greedy, karena algoritma ini seringkali dimanfaatkan untuk penyelesaian masalah optimasi, sehingga penulis merasa bahwa algoritma ini merupakan salah satu solusi yang tepat untuk penyelesaian permasalahan ini.

Karena algoritma greedy adalah salah satu algoritma yang sering dimanfaatkan untuk penyelesaian persoalan optimasi, sehingga pada makalah in

II. LANDASAN TEORI

A. Algoritma Greedy

Algoritma greedy merupakan metode paling umum dalam penyelesaian persoalan optimasi. Persoalan optimasi sendiri berarti sebuah persoalan yang ditujukan untuk mencari solusi optimal. Persoalan optimasi dibagi menjadi dua yaitu maksimasi dan minimalisasi. Secara definisi, algoritma greedy adalah algoritma yang memecahkan persoalan secara Langkah demi Langkah (*step by step)* sedemikian sehingga pada setiap Langkah dilakukan:

1. Pengambilan solusi terbaik yang dapat diperoleh saat itu tapa memperhatikan konsekuensi ke masa depan.
2. Pengambilan solusi yang optimum local pada setiap Langkah diharapkan dapat menghasilkan solusi yang optimum global.

Dalam pemilihan solusi, algoritma greedy mempunyai 6 elemen yang dijadikan dasar pemilihan solusi, yaitu :

1. Himpunan Kandidat (C): himpunan yang berisi kandidat yang akan dipillih pada setiap langkah (cth. simpul/sisi dalam graf, job, koin dll.)
2. Himpunan Solusi (S): himpunan yang berisi kandidat yang akan dipilih.
3. Fungsi Solusi: fungsi yang digunakan untuk menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
4. Fungsi Seleksi: Fungsi yang digunakan untuk memilih kandidat berdasarkan strategi greedy yang diimplementasikan. Strategi ini bersifat heuristic.
5. Fungsi Kelayakan: Fungsi yang digunakan untuk memeriksa apakah sebuah kandidat layak dimasukkan kedalam himpunan solusi.
6. Fungsi Objektif: Fungsi tujuan, antara minimalisasi atau maksimalisasi.

Dengan menggunakan keenam elemen diatas, maka dapat dikatakan bahwa algoritma greedy melibatkan pencarian sebuah bagian S, dari himpunan kandidat C; yang dalam hal ini S harus memenuhi beberapa kondisi sehingga bisa mencapai fungsi objektif. Skema umum dari algoritma greedu dapat dilihat dibawah ini:

```
function greedy(C: himpunan_kandidat) → himpunan_solusi {Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy}
Deklarasi
x: kandidat
S: himpunan_solusi
Algoritma:
S ← {} {inisialisasi S dengan kosong}
while (not SOLUSI(S)) and (C ≠ {}) do
        x ← SELEKSI(C) { pilih sebuah kandidat dari C}
        C ← C – {x} {buang x dari C karena sudah dipilih}
        if LAYAK(S ∪ {x}) then
        {x memenuhi kelayakan untuk dimasukkan ke dalam himpunan solusi} S ← S ∪ {x}
        {masukkan x ke dalam himpunan solusi}
        endif
endwhile
{SOLUSI(S) or C = {}}
if SOLUSI(S) then {solusi sudah lengkap}
        return S
else
        write('tidak ada solusi')
endif
```

Dari skema diatas, dapat disimpulkan bahwa pada akhir setiap iterasi didapatkan sebuah solusi yang merupakan solusi optimum local. Pada akhir bagian *while-do* diperoleh optimum global (jika ada). Hal ini dikarenakan sebuah solusi optimum global belum tentu solusi yang optimum, bisa jadi pseudo-optimum ataupun sub-optimum, dikarenakan sifat dari algoritma greedy yang tidak memeriksa semua kemungkinan yang ada. Sehigga tidak selalu menghasilkan solusi yang optimum.

Contoh persoalan yang selalu ditemukan hasil optimal oleh algoritma greedy adalah Coin exchange problem, sedangkan yang tidak selalu menghasilkan hasil optimal diantaranya Integer Knapsack Problem.

B. Coin Exhange Problem

Coin Exchange Problem adalah persoalan optimasi untuk mencari jumlah pecahan koin paling minimum untuk menukar sejumlah uang. Untuk beberapa mata uang, pendekatan algoritma greedy dapat menghasilkan hasil yang optimal setiap saat. Hal ini tergantung kepada pecahan-pecahan uang yang tersedia dari sebuah mata uang.

Berikut merupakan pemetaan elemen-elemen greedy pada persoalan ini:

1. Himpunan Kandidat: Semua pecahan uang yang nilainya lebih kecil dari sisa uang yang harus ditukar
2. Himpunan Solusi: Pecahan uang yang nilainya paling besar yang lebih kecil dari sisa uang yang harus ditukar
3. Fungsi Solusi: Memeriksa apakah nilai uang dari kandidat sudah yang paling besar
4. Fungsi Seleksi: Memilih nilai uang yang paling besar ari pecahan yang tersedai
5. Fungsi Kelayakan: Memeriksa apakah pecahan uang mempunyai nilai yang lebih kecil dari sisa uang yang harus ditukar
6. Fungsi Objektif: Dihasillkan jumlah pecahan uang paling minim yang diperlukan untuk menukar sejumlah uang.

C. Integer Knapsack Problem

Integer knapsack problem adalah persoalan optimasi untuk mendapatkan keuntungan terbesar dari beberapa rangkaian barang yang masing-masing mempunyai keuntungan dan berat tersendiri. Persoalan ini bertujuan untuk memaksimalkan keuntungan dengan membawa serangkaian barang sedemikian rupa sehingga barang-barang tersebut akan menghasilkan keuntungan maksimal untuk kapasitas knapsack tertentu.

Berikut merupakan pemetaan elemen-elemen greedy pada persoalan ini:

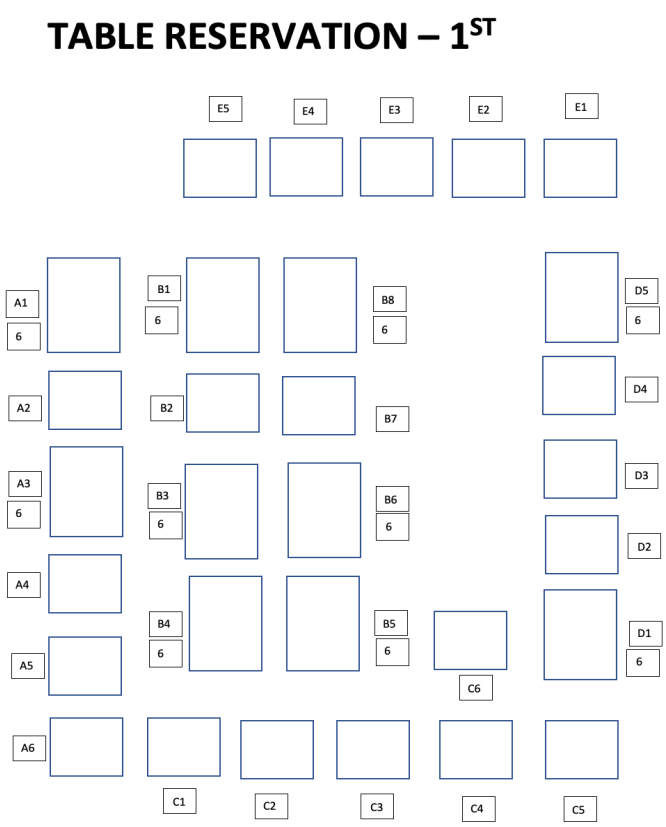
1. Himpunan Kandidat: Semua barang yang beratnya lebih kecil dari sisa kapasitas di dalam knapsack
2. Himpunan Solusi: Barang yang menghasilkan keuntungan paling besar, atau mempunyai berat paling kecil (tergantung strategi yang digunakan)
3. Fungsi Solusi: Memilih barang yang punya keuntungan paling besar ataupun berat paling kecil.
4. Fungsi Seleksi:
5. Fungsi Kelayakan: Memeriksa apakah sebuah barang masih bisa dimasukkan kedalam knapsack (berat < kapasitas)

6. Fungsi Objektif: Memaksimalakan keuntungan yang diperoleh dari pembawaaan serangkaian barang

III. Deskripsi Masalah

Pada sebuah restoran ataupun tempat makan lainnya, terkadang kita dapat melakukan reservasi untuk memesan tempat terlebih dahulu, sehinggga memastikan kita untuk mendapat tempat pada restoran tersebut. Namun tentunya, restoran tidak akan menerima 1, 2 ataupun 5 reservasi sekaligus, melainkan bisa saja menerima puluhan reservasi sekaligus masing-masing dengan jumlah kursi yang ingin dipesan oleh pelanggan. Dengan jumlah yang banyak ini, maka kombinasi penempatan pelanggan dari hasil penerimaan reservasi pun akan sangat banyak, sehingga meningkatkan peluang untuk timbulnya kesalahan dimana penempatan yang tidak tepat akan menghasilkan kapasitas yang dapat diduduki tidak maksimal.

Pemanfaatan algoritma Greedy pada persoalan ini bertujuan untuk meminimalisasi kesalahan dan memaksimalisasi keuntungan dari restoran dengan cara memaksimalkan kapasitas tempat duduk yang tersedia sepenuh mungkin. Penyelesaian permalsahan ini memiliki konsep yang serupa dengan 2 permasalahan algoritma greedy yaitu *Coin Exchange Problem* dan *Integer Knapsack Problem.* Pada persoalan ini, seperti Coin Exchange Problem, penyelesaian persoalan tergantung kepada kapasitas tiap meja pada restoran tersebut, karena pada restoran bisa saja mempunyai kapasitas 4 kursi, 2 kursi, 5 kursi dan sebagainya. Sehingga penempatan tiap reservasi bisa jadi berbeda jika kapasitas kursi yang tersedia berbeda. Selain itu ada konsep lain pada integer knapsack problem, dimana pada strategi *greedy by weight* akan dihasilkan solusi dimana akan dimasukkan barang sebanyak mungkin. Pada persoalan ini hal ini bisa diimplementasikan dimana reservasi akan ditempatkan di meja yang akan menghasilkan kursi kosong paling sedikit.



Gambar 3.1 Contoh Denah Restoran

Gambar diatas adalah salah satu contoh denah restoran dimana terdapat meja yang mempunyai kapasitas 4 kursi dan 6 kursi. Salah satu factor yang jadi penentu pada penyelesaian persoalan ini adalah meja dan kursi pada kondisi ini statis dan tidak bisa digeser. Oleh karena itu 2 konsep yang mirip yang disebutkan diatas dapat diimplementasikan dengan baik.

IV. Metodologi

A. *Pemetaan Denah Restoran*

Pada makalah ini, hanya akan diimplementasikan persoalan yang menggunakan denah seperti diatas, dimana setiap meja mempunyai kode yang terdiri dari huruf dan angka, jika disebelah kode meja tidak terdapat angka, maka meja tersebut mempunyai kapasistas 4 kursi. Perlu dipertimbangkan juga bahwa pada umumnya jika sebuah pelanggan memesan kursi dalam jumlah yang banyak, pelanggan akan ingin bahwa kursi-kursi tersebut terdapat di tempat yang berdekatan. Pada gambar 3.1 Contoh meja berdekatan bisa dilihat pada meja B1 dan B2

B. *Penentuan Strategi Greedy*

Tentunya, pada algoritma greedy terdapat beberapa strategi yang bisa digunakan untuk dijadikan pendekatan. Pada persoalan ini akan diterapkan strategi greedy yaitu *greedy by chair remainder,* dimana strategi ini akan berusaha untuk meminimalkan jumlah kursi yang kosong dari setiap penanganan reservasi. Hal ini dilakukan karena jika suatu meja sudah ditempati oleh sebuah pelanggan dengan jumlah

reservasi tertentu dan masih ada kursi kosong, kursi kosong tersebut tidak dapat digunakan/dijual ke pelanggan lainnya.

C. *Pemetaan Elemen Greedy*

Seperti yang dijelaskan pada landasan teori, sebuah algoritma greedy dapat didefinisikan menjadi enam elemen berbeda yaitu himpunan kandidat, himpunan solusi, fungsi solusi, fungsi seleksi, fungsi kelayakan dan fungsi objektif. Berikut adalah pemetaan elemen-elemen tersebut dalam penyelesaian persoalan ini:

1. Himpunan Kandidat (C): Seluruh meja ataupun serangkaian meja yang ada di dalam restoran
2. Himpunan Solusi (S): Meja ataupun serangkaian meja yang sudah dipilih sedemikian rupa sehingga dapat mencukupi kapasitas dari sebuah reservasi.
3. Fungsi Solusi: Memeriksa apakah meja atau serangkaian meja yang dipilih mempunyai sisa kursi kosong yang paling minim.
4. Fungsi Seleksi: Memilih meja atau serangkaian meja yang mempunyai sisa kursi kosong paling minim.
5. Fungsi Kelayakan: Memeriksa apakah meja ataupun serangkaian meja dapat menampung seluruh pelanggan pada suatu reservasi.
6. Fungsi Objektif: Memaksimalkan jumlah pelanggan yang ada di dalam restoran.

D. *Implementasi Algoritma Greedy*

Pada implementasi ini, dimanfaatkan dua konsep pada algortima greedy yang disebutkan diatas yaitu pada *Coin Exchange Problem* dan *Integer Knapsack Problem.* Sehingga, secara garis besar algoritma ini dibagi menjadi dua tahap yaitu:

1. Memparitsi sebuah reservasi menjadi beberapa reservasi berdasarkan kapasitas meja (sejenis *coin exchange problem).*
2. Memasukkan partisi tersebut kedalam meja-meja yang tersedia selama belum memenuhi kapasitas meja dan belum dipesan oleh orang lain (sejenis *integer knapsack problem)*

Tentunya, pada implementasi ini akan ditambahkan fungsi-fungsi heuristic tertentu pada proses penentuan himpunan solusi, karena seperti yang kita ketahui bahwa pada *coin exchange problem* tidak akan selalu didapatkan jumlah pecahan koin yang paling minimal, karena tidak semua rangkaian pecahan dapat merepresentasikan nilai uang dengan baik. Sama halnya dengan pada kasus ini dengan meja 4 dan 6 kursi, tidak selalu dapat dihasilkan hasil yang mempunyai kursi kosong paling minimal. Berikut beberapa contohnya:

Contoh.

Reservasi: 8 orang

*jika menggunakan konsep coin exchange problem maka, akan didapatkan

8 = 6 + 2 (2 ini akan dimasukkan ke meja berkapasitas 4 sehingga ada sisa 2 kursi)

Sedangkan solusi optimalnya adalah

8 = 4 + 4

*dengan menggunakan jumlah meja yang sama(2), bisa dihasilkan solusi dimana tidak ada kursi kosong yang tersisa.

Selain itu ada beberapa kasus dimana sebuah reservasi yang bisa dipartisi tidak perlu dipartisikan, berikut contohnya:

Contoh.

Reservasi: 5 orang

*jika dipartisikan maka akan dihasilkan

5 = 4 + 1
*dengan ini akan diperlukan 2 meja berkapasitas 4, sehingga akan dihasilkan 3 kursi kosong

Namun jika kelima orang ini dimasukkan kedalam meja berkapasitas 6, maka akan dihasilkan hanya 1 kursi kosong (paling optimal).

Dengan mempertimbangkan beberapa kasus yang disebutkan diatas maka dapat dibentuk formulasi dari persoalan pada gambar 3.1 sebagai berikut:

1. Total Reservasi: N
2. Himpunan Meja: {M1, M2, M3, …}, Mi merupakan truple dengan (Kode,kapasitas,kursiKosong)
3. Himpunan Solusi: {X1, X2 ,X3, …}, Xi = 1 jika dipilih dan = 0 jika tidak.
4. Meminimalkan jumlah kursi kosong yang ada di dalam restoran

Berikut merupakan contoh implementasi algoritma dalam bentuk pseudocode:

```
{Akan ada variable global seperti}

ReservasiMeja: Himpunan Meja yang
berisi nama pelanggan yang sudah
memesan

function ChooseTable(M: Himpunan Meja
dalam restoran, R: Reservasi ) → 
mengembalikan meja-meja yang dapat
ditempati untuk sebuah reservasi.

Deklarasi

      i: integer
      jumlahReservasi: integer


Algoritma
function ChooseTable(M: Himpunan Meja
dalam restoran, R: Reservasi ) → 
mengembalikan meja-meja yang dapat
ditempati untuk sebuah reservasi.

Deklarasi

    i: integer
    jumlahReservasi: integer
    S: array[1..N] of Meja
    Rn: array [1..N] of reservasi

   jumlahReservasi <--
R.jumlahReservasi

    if (jumlahReservasi <= 4 ) then
        i transversal [1..N]
            if (M[i].kapasitas == 4
and not terisi(M[i])) then
                S <-- M[i]
                break
            endif
    endif

    else if (jumlahReservasi <=6 )
then
        i transversal [1..N]
            if (M[i].kapasitas = 6 and
not terisi(M[i])) then
                S <-- M[i]
                break
            endif
    endif

    {jika reservasi > 6}
    else
        {lakukan partisi terhadap
      if (jumlahReservasi mod 6 = 0) then
            count ← jumlahReservasi/6
            i transversal [1..count]
                Rn[i] = new Reservasi(6)
            {cari meja bertetangga sejauh
count}
            {dengan kapasitas masing-
masing 6}
{cari meja bertetangga sejauh count}
            {dengan kapasitas masing-
masing 6}
        else if (jumlahReservasi mod 4 =
0) then
            count ←  jumlahReservasi/4
            i transversal [1..count]
                Rn[i] ←  new Reservasi(4)
            {cari meja bertetangga sejauh
count}
            {dengan kapasitas masing-
masing 4}
        endif
        {kasus jika tidak habis dibagi 4
atau 6}
        else
            {bandingkan sisa kursi yang
paling sedikit}
            {dengan mendahulukan 4}
      i transversal [1..N]
            if (M[i].kapasitas == 4 and
not terisi(M[i])) then
                S <-- M[i]
                break
            endif
      S ← S + ChooseTable(M,
      Reservasi(jumlahReservasi-4)
```

E. Hasil Pengujian Algoritma

Berikut merupakan contoh penggunaaan algoritma yang telah diajukna dalam pengangan reservasi berjumlah 13

Langkah 1

Jumlah dipesan: 13
S = {}

Langkah 2
Jumlah dipesan: 9 (sisa)
S = {4}
Pilih meja 4, karena 13 bukan kelipatan 4 ataupun 6

Langkah 3
Jumlah dipesan: 5 (sisa)
S = {4,4}

Langkah 4
Jumlah dipesan: 0
S = {4,4,6}

Pilih meja 6, karena dengan meja 6 hanya perlu 1 meja untuk diduduki 5 orang sedangkan dengan meja 4, perlu dua meja yang menyebabkan ada 3 kursi kosong

Tentunya algoritma yang diusulkan masih bisa banyak dilakukan pengembangan, akan tetapi pengembangan lebih lanjut akan menghasilkan algoritma yang mengimplementasikan konsep *Program Dinamis,* sehingga tidak bisa diklasifikasikan sebagai greedy. Dengan algoritma yang diusulkan ini, diharapkan pada kebanyakan kasus akan menemukan solusi yang optimal. Akan tetapi karena jumlah meja terbatas, sehingga bisa jadi aka nada beberapa kasus selama penangaanan serangkaian reservasi yang menyisakan banyak kursi kosong. Oleh karena alasan ini pula, algoritma ini mengutamakan pemakaian kursi yang berkapasitas 4, karena selain menyisakan kursi kosong lebih sedikit, jumlahnya dalam permaslahan ini juga lebih banyak. Selain itu hal yang perlu diperhatikan yaitu bahwa tidak semua restoran bisa memanfaatkan algoritma ini, karena jumlah kapasitas tiap meja yang berbeda-beda.

V. KESIMPULAN

Algoritma Greedy adalah salah satu pendekatan paling umum dalam menyelesaikan persoalan optimasi, akan tetapi walaupun mudah untuk digunakan, terdapat kekurangan yaitu terkadang tidak bisa menghasilkan solusi yang paling optimal, oleh karena itu Jika ingin menyelesaikan persoalan ini secara optimal, maka perlu dikembangkan algoritma *program dinamis,* dimana ia akan mempertimbangkan segala kemungkinan yang ada, dan akan memilih solusi yang paling optimal.

.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih serta puji dan syukur kepada Allah SWT Esa karena berkat rahmatnya penulis dapat menyelesaikan makalah ini dengan tepat waktu. Penulis juga mengucapkan terima kasih kepada seluruh tim dosen mata kuliah IF2211 Strategi Algoritma yang telah dengan sabar membimbing penulis selama satu semester. Tanpa bimbingan dan ajaran dari tim dosen IF2211 makalah ini tidak mungkin terwujud. Penulis juga berterima kasih kepada seluruh tim asisten mata kuliah IF2211 yang telah dengan sabar memberi ilmu selama mata kuliah IF2211 semester ini baik secara langsung saat demo maupun tidak langsung.

REFERENCES

[1] Levitin, Anany. 2011. Introduction to the Design and Analysis of Algorithms. Pearson: New York.

[2] Munir, Rinaldi. (2021, Januari 27). Greedy Bagian 1,2, dan 3. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/20202021/stima20 - 21.htm, diakses pada 20 Mei 2022

[3] https://www.geeksforgeeks.org/greedy-algorithm-to-find-minimum-number-of-coins/

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, 20 Mei 2022

Ttd

Muhammad Gerald Akbar Giffera

13520143