Penggunaan Algoritma Dijkstra untuk Mencari Rute Terpendek Melalui Jalur Tol yang Harus Dilalui

Rehan Adi Satrya – 13518061 (*Author*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: rehanadi457@gmail.com

***Abstract*—Dalam berpindah dari suatu tempat ke tempat lain, jalan tol lazim digunakan. Namun seiringan dengan banyaknya opsi jalan tol di sepanjang perjalanan, jarak yang ditempuh menjadi semakin variatif. Pencarian jarak terdekat bisa kita lakukan dengan pendekatan metode *greedy* melalui algoritma Dijkstra dengan hanya mempersoalkan masalah jarak semata.**

***Keywords—Jarak, Tol, Dijkstra, Greedy*, Variasi, Terpendek**

I. PENDAHULUAN

Mudik sudah seakan menjadi agenda tahunan rakyat Indonesia, yaitu pulang ke kampung halaman dengan tujuan merayakan idulfitri bersama sanak saudara tercinta. Di luar musim mudik pun para perantau juga acap kali pulang ke kampung halaman untuk mengurus berbagai hal. Dalam bepergian, banyak sekali opsi yang tersedia untuk berpindah dari suatu tempat ke tempat lain. Namun jika waktu bukanlah masalah atau memang ada niatan membawa kendaraan pribadi ke kampung halaman, maka menaiki mobil di jalan tol bisa menjadi opsi yang menarik.

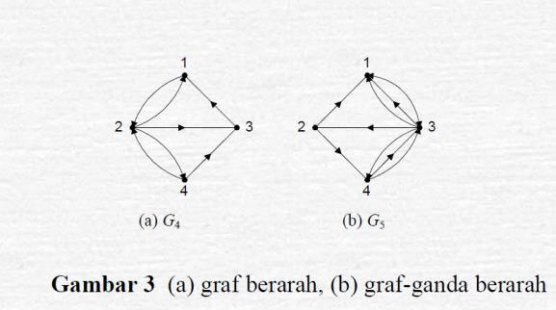
Jalan tol (di Indonesia sering digunakan secara bergantian dengan jalan bebas hambatan) adalah suatu jalan yang dikhususkan untuk kendaraan bersumbu dua atau lebih (mobil, bus, truk) dan bertujuan untuk mempersingkat jarak dan waktu tempuh dari satu tempat ke tempat lain. Namun dalam penerapannya, pengguna jalan tol harus membayar suatu biaya yang dibayarkan setiap melewati gerbang tol. Jika perjalanan jauh, maka biaya yang perlu dikeluarkan relatif menjadi lebih besar.

Namun layaknya pepatah, ada banyak jalur yang bisa digunakan untuk mencapai suatu tujuan dari titik tertentu sehingga untuk tujuan yang sama, bisa didapatkan jarak tempuh yang berbeda. Jarak yang berbeda-beda tersebut bisa kita dapatkan yang terdekatnya tanpa memperhatikan faktor lainnya dengan menggunakan pendekatan metode *Dijkstra*.

II. LANDASAN TEORI

A. *Teori Graf*

Graf G = (V, E), yang dalam hal ini: V = himpunan tidak-kosong dari simpul-simpul (vertices) = { v1 , v2 , ... , vn }
E = himpunan sisi (edges) yang menghubungkan sepasang simpul = {e1 , e2 , ... , en }



Gambar 3 (a) graf berarah, (b) graf-ganda berarah

Sumber:
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf

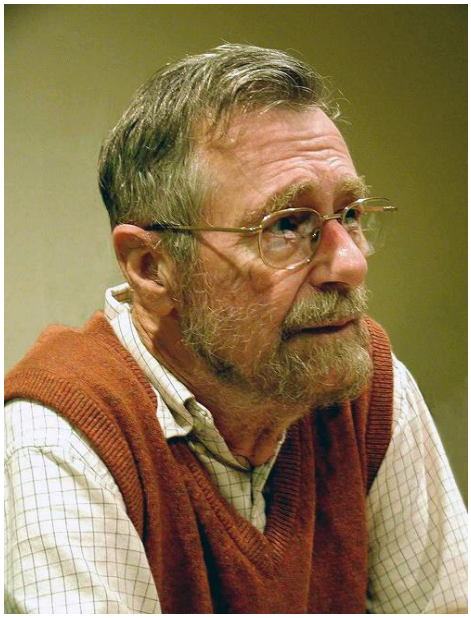
Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas 2 jenis:

1. graf berarah (directed graph atau digraph) ialah Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah
2. graf tak-berarah (undirected graph) ialah Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah.

Graf juga memiliki beberapa terminologi, diantaranya adalah:

1. ketetanggaan (*Adjacent*)
   dua buah simpul dikatakan bertetangga bila keduanya terhubung langsung
2. bersisian (*Insidency*)
   untuk sembarang sisi e = (v_j, v_k) dikatakan e bersisian dengan simpul v_j, atau e bersisian dengan simpul v_k
3. simpul terpencil (*Isolated Vertex*)
   simpul terpencil ialah simpul yang tidak mempunyai sisi yang bersisian dengannya
4. graf kosong (null graph/empty graph)
   graf yang himpunan sisinya merupakan himpunan kosong (N_n)
5. derajat (*degree*)
   derajat suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut dengan notasi: d(v).

B. Algoritma Dijkstra



Edsgar Wybe Dijkstra, penemu Algoritma Dijkstra

Sumber: https://id.wikipedia.org/wiki/Berkas:Edsger_Wybe_Dijkstra.jpg

Algoritma Dijkstra, (dinamai sesuai penemunya, seorang *computer scientist*, Edsger Dijkstra), adalah sebuah teknik yang menggunakan pendekatan *greedy* yang dipakai dalam memecahkan permasalahan jarak terpendek (shortest path problem) untuk sebuah graf berarah dengan bobot-bobot yang bernilai nonnegatif, [0, ∞). Masukan algoritma ini adalah sebuah graf berarah yang berbobot ***G*** dan sebuah titik asal ***S*** dalam himpunan garis ***V***.

Contohnya, jika titik dari sebuah graf melambangkan gerbang tol dan bobot garis melambangkan jarak antar gerbang tol tersebut, algoritma Dijkstra dapat digunakan untuk menemukan jarak terpendek antara dua kota yang melalui jalan tol.

Biaya dari sebuah garis dapat dianggap sebagai jarak antara dua simpul, yaitu jumlah jarak semua garis dalam jalur tersebut. Untuk sepasang titik ***X*** dan ***Y*** dalam ***V***, algoritma ini menghitung jarak terpendek dari ***X*** ke ***Y***.

Pseudocode:

```
function Dijkstra(Graf, asal):
        Q adalah sets of titik

        foreach titik v dalam Graf:
            jarak[v] ← tak hingga
            sebelum[v] ← kosong
            tambahkan v ke dalam Q
        jarak[asal] ← 0;

        while Q not empty:
            u ← titik dalam Q dengan jarak[u] minimum
            pop u dari Q

            foreach tetangga v dari u:
  // hanya v yang masih dalam Q
alt ← jarak[u] + jarak_antara(u, v)
if alt < jarak[v]:
                    jarak[v] ← alt
                    sebelum[v] ← u

    return jarak[], sebelum[]
```

Kompleksitas = O(E log V)
Dengan E adalah *edges* dan V adalah *vertices*.
Algoritma Dijkstra hanya bisa digunakan untuk graf berarah yang memiliki bobot non-negatif. Jika graf memiliki bobot negatif, maka harus digunakan algoritma lain. Contohnya adalah Bellman—Ford Algorithm.

III. PEMANFAATAN ALGORITMA DIJKSTRA PADA PENCARIAN OPSI PEMILIHAN JALUR TOL TERMURAH

Pada makalah ini, algoritma yang digunakan menggunakan bahasa python 2.7 dan bersumber dari Bogo to Bogo dengan modifikasi pada driver sesuai kebutuhan.

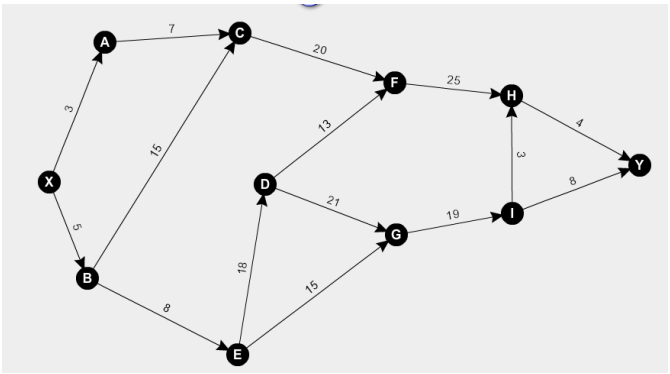
Dalam penerapannya, jarak antar satu gerbang tol ke gerbang tol lainnya akan menjadi bobot graf sementara gerbang tol itu sendiri akan menjadi simpul.

Untuk memanfaatkan Algoritma Dijkstra program tersebut, dilakukan langkah-langkah berikut:

A. Memetakan jalur

Jalur yang akan kita gunakan sebagai contoh bukanlah jalur asli di dunia nyata. Dalam kasus kita, kita akan menjadikan titik asal kita adalah X dan titik tujuan kita adalah Y. Jarak antar gerbang tol akan kita jadikan sebagai bobot graf.

Misalkan kita akan memiliki 9 gerbang tol di antara X dan Y yang terletak sedemikian rupa, maka didapat sebuah graph berarah dan berbobot non-negatif sebagai berikut:



Sumber: Dokumentasi Pribadi

B. Penerapan Algoritma Dijkstra

1. $X_X = 0$
   Isi PrioQueue = [X]
   Simpul dikunjungi = []
2. $A_X = 3$
   $B_X = 5$
   Isi PrioQueue = [A_3, B_5]
   Simpul dikunjungi = [X]
3. $C_A = 10$
   Isi PrioQueue = [B_5, C_{10}]
   Simpul dikunjungi = [X, A]
4. $E_B = 13$
   Isi PrioQueue = [C_{10}, E_{13}]
   Simpul dikunjungi = [X, A, B]
5. $F_C = 30$
   Isi PrioQueue = [E_{13}, F_{30}]
   Simpul dikunjungi = [X, A, B, C]
6. $G_E = 28$
   $D_E = 31$
   Isi PrioQueue = [G_{28}, $F_{30,}$ D_{31}]
   Simpul dikunjungi = [X, A, B, C, E]
7. $I_G = 47$
   Isi PrioQueue = [F_{30}, D_{31}, I_{47}]
   Simpul dikunjungi = [X, A, B, C, E, G]
8. $H_F = 55$
   Isi PrioQueue = [D_{31}, I_{47}, H_{55}]
   Simpul dikunjungi = [X, A, B, C, E, G, F]
9. $F_D = 44$
   $G_D = 52$
   Keduanya tidak memenuhi karena F dan G sudah pernah dipop dari PrioQueue
   Isi PrioQueue = [I_{47}, H_{55}]
   Simpul dikunjungi = [X, A, B, C, D, E, F, G]
10. $H_I = 50$
    $Y_I = 58$
    Isi PrioQueue = [H_{50}, Y_{58}]
    Simpul dikunjungi = [X, A, B, C, D, E, F, G, I]
11. $Y_H = 54$
    Isi PrioQueue = [Y_{54}]
    Simpul dikunjungi = [X, A, B, C, D, E, F, G, H, I]
12. Isi PrioQueue = []
    Simpul dikunjungi = [X, A, B, C, D, E, F, G, H, I, Y]
13. Karena PrioQueue sudah kosong, maka, kita cukup telusuri kembali jalur dengan cost termurah yang sudah kita lalui secara mundur dari Y ke X.

    Didapat: X->B->E->G->F->H->I->Y

C. Hasil Eksekusi

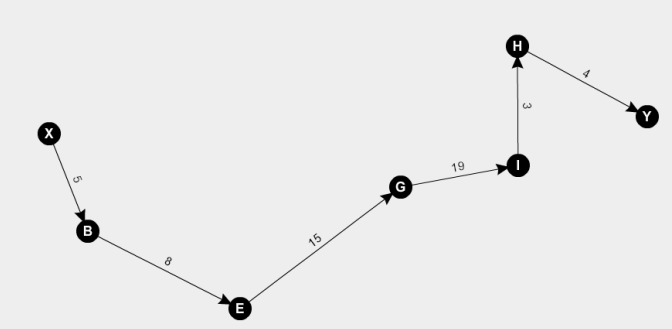
```
Graph data:
( a , x,   3)
( a , c,   7)
( c , f,  20)
( c , b,  15)
( c , a,   7)
( b , e,   8)
( b , x,   5)
( b , c,  15)
( e , d,  18)
( e , b,   8)
( e , g,  15)
( d , f,  13)
( d , e,  18)
( d , g,  21)
( g , e,  15)
( g , d,  21)
( g , i,  19)
( f , d,  13)
( f , h,  25)
( f , c,  20)
( i , y,   8)
( i , h,   3)
( i , g,  19)
( h , f,  25)
( h , y,   4)
( h , i,   3)
( y , h,   4)
( y , i,   8)
( x , b,   5)
( x , a,   3)
Dijkstra's shortest path
updated : current = x next = b new_dist = 5
updated : current = x next = a new_dist = 3
updated : current = a next = c new_dist = 10
updated : current = b next = e new_dist = 13
not updated : current = b next = c new_dist = 10
updated : current = c next = f new_dist = 30
updated : current = e next = d new_dist = 31
updated : current = e next = g new_dist = 28
not updated : current = g next = d new_dist = 31
updated : current = g next = i new_dist = 47
not updated : current = f next = d new_dist = 31
```

```
    updated : current = f next = h new_dist = 55
    updated : current = i next = y new_dist = 55
    updated : current = i next = h new_dist = 50
    updated : current = h next = y new_dist = 54
The shortest path : ['x', 'b', 'e', 'g', 'i', 'h', 'y']
```

Dari hasil eksekusi program tersebut, didapat bahwa jalur terpendek yang dapat kita tempuh adalah:

x->b->e->g->i->h->y

seperti visualisasi berikut:



Sumber: Dokumentasi Pribadi

Algoritma ini dapat digunakan dengan peta yang berbeda asalkan kita bisa mengubah peta tersebut menjadi graf berbobot.

IV. Penerapan di Dunia Nyata

Walaupun tidak diketahui algoritma apa yang digunakan, beberapa aplikasi telah mempermudah pencarian jalur dengan mencari jalur tercepat yang bisa dilalui secara *real-time*.

Satuan yang ditampilkan adalah satuan waktu karena dengan hal tersebut, diharapkan pengguna bisa memperkirakan waktu tiba di titik tujuan. Sederhananya, waktu tempuh bisa didapat dengan rumus:

t = s/v

dengan:

t adalah waktu tempuh

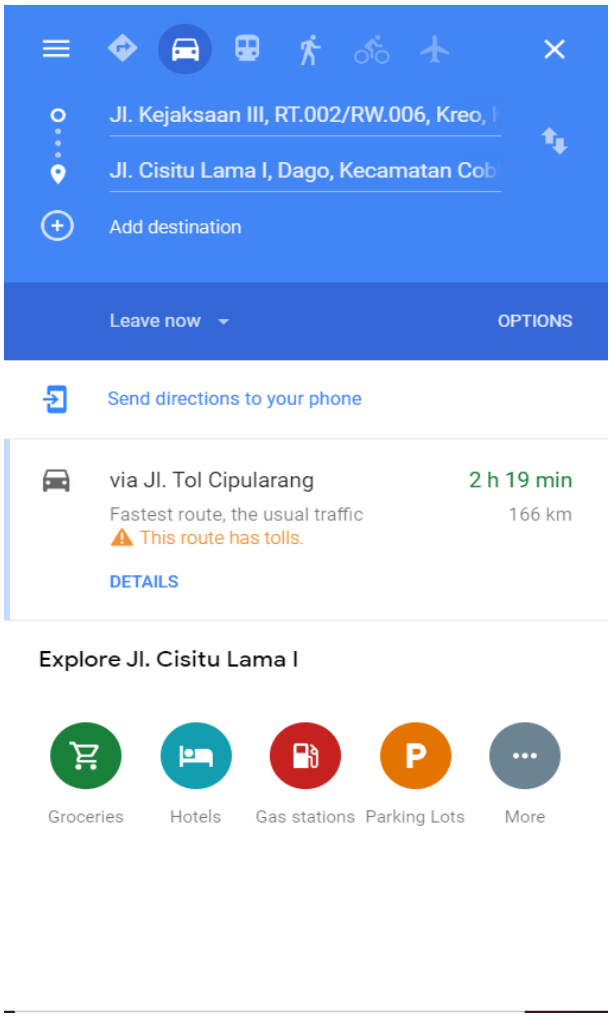
s adalah jarak terpendek yang harus ditempuh

v adalah kecepatan rata-rata mobil/kendaraan

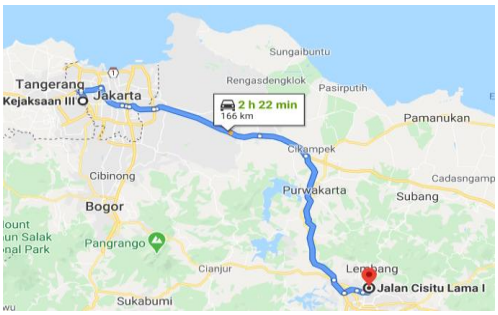
dengan asumsi khusus untuk mobil, kecepatan diatur di kisaran angka 70 km/jam.

Berikut adalah contoh aplikasi yang menawarkan rute tercepat untuk mencapai suatu tujuan:

A. *Google Maps*

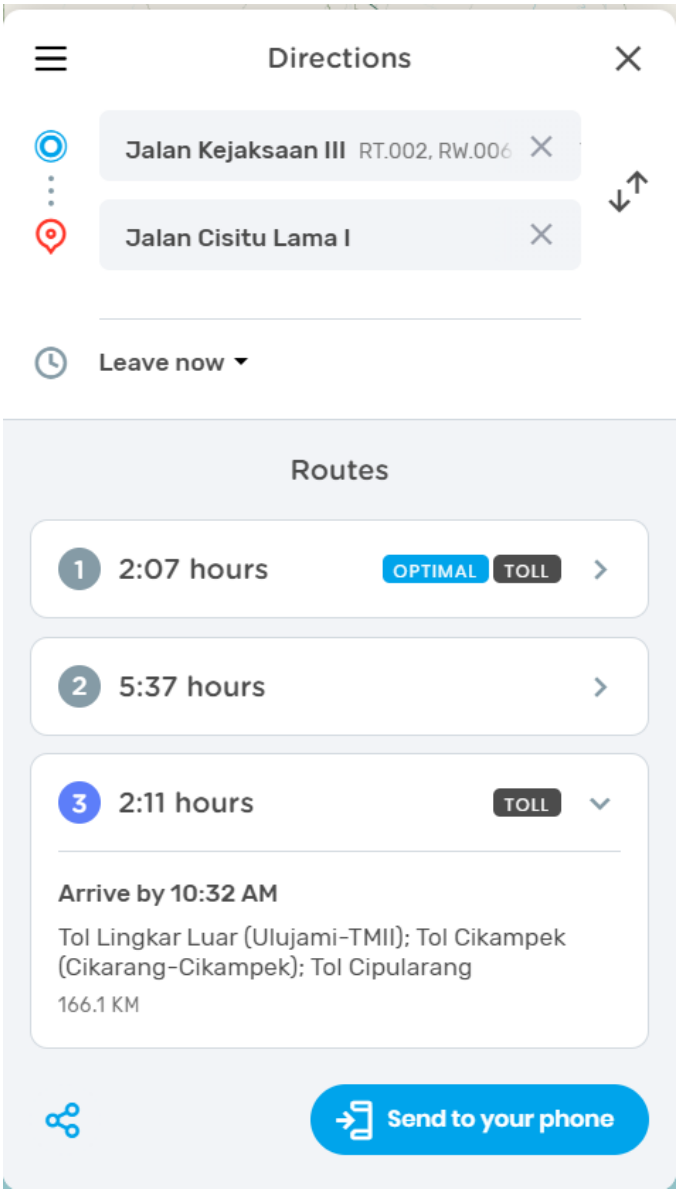


Sumber: Google Maps (Dokumentasi Pribadi)

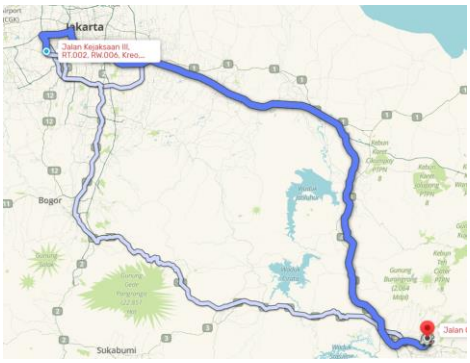


Sumber: Google Maps (Dokumentasi Pribadi)

B. *Waze*



Sumber: Live Map Waze (Dokumentasi Pribadi)



Sumber: Live Map Waze (Dokumentasi Pribadi)

Namun, pada aplikasi penyedia jalur daring, rute yang dipaparkan juga dipengaruhi beberapa variabel. Diantaranya adalah:

1. kemacetan
2. ganjil-genap
3. penutupan ruas jalan
4. penutupan jalan karena bencana alam (banjir)
5. pemberlakuan *contra flow*

dan lain sebagainya.

Algoritma ini juga bisa dipakai untuk pemetaan hal-hal lain, seperti contohnya:

1. Mencari rute stasiun kereta api
2. Mencari rute pesawat dengan transit

V. Kesimpulan

Algoritma Dijkstra dapat digunakan untuk menentukan jalur terpendek yang harus dilalui untuk menempuh sebuah perjalanan dari suatu titik ke titik lainnya. Kasus ini dapat diterapkan dalam aplikasi peta yang harus menunjukkan jalur terbaik bagi pengguna secara *real-time* sehingga pengguna aplikasi bisa sampai pada tujuan secepat mungkin walau pada kenyataannya, pemilihan rute terbaik tidak hanya berdasarkan jarak semata, namun harus memperhatikan variabel-variabel yang memengaruhi waktu tempuh perjalanan pengguna.

VI. Video Link at Youtube

https://youtu.be/K9kVdHTBEK4

VII. Ucapan Terima Kasih

Pertama-tama saya ucapkan terima kasih untuk Tuhan Yang Maha Esa karena berkat kuasanyalah makalah ini dapat terselesaikan di kondisi pandemi sehingga penulis masih diberikan kesehatan dalam mengerjakan tugas-tugas serta Ujian Akhir Semester. Semoga ujian berupa pandemi ini bisa cepat berlalu.

Tak lupa pula saya ucapkan terima kasih untuk Ibu Masayu Leylia Khodra, S.T., M.T. sebagai dosen saya untuk mata kuliah Strategi Algoritma (IF2211) yang telah sabar dan ikhlas dalam mendidik saya dan kawan-kawan Decrypt seperjuangan saya, terutama dari K1. Dan juga terima kasih untuk Bapak Dr. Ir. Rinaldi Munir karena berkat *website* beliau saya bisa mengakses materi pembelajaran yang dapat saya manfaatkan untuk menyelesaikan makalah, maupun menimba ilmu terutama karena ketiadaan kuliah tatap muka yang menjadikan semua kegiatan perkuliahan menjadi daring.

VIII. Referensi

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf

[2] https://www.bogotobogo.com/python/files/Dijkstra/Dijkstra_shortest_path.py

[3] E. W. Dijkstra: A note on two problems in connexion with graphs. In: Numerische Mathematik. 1 (1959), S. 269–271

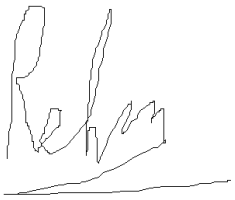
[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 24.3: Dijkstra's algorithm, pp.595–601.

[5] https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

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, 29 April 2020



Rehan Adi Satrya
13518061