Aplikasi Struktur Data Graf, *Queue* dan Algoritma *Breadth First Search(BFS)* pada Pencarian Solusi Labirin

Ridho Pratama / 13516032[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13516032@std.stei.itb.ac.id, p.ridho@students.itb.ac.id

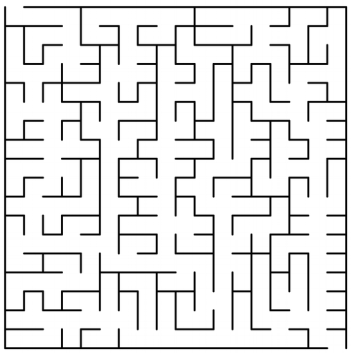
***Abstrak*—Labirin (*maze*) adalah salah satu contoh permainan teka-teki dimana ada sebuah sistem jalur yang rumit, berliku-liku, dan dapat memiliki jalan buntu. Labirin ini dapat berbentuk berbagai macam, dan dapat menggunakan berbagai macam media, seperti kertas, ataupun benda fisik. Labirin dapat di modelkan menjadi model graf sehingga algoritma pencarian jalur seperti BFS, Dijkstra, atau A*. Pada makalah ini akan dibahasan penyelesaian maze yang bermodel kisi-kisi (*grid*) sehingga dapat dimodelkan menjadi graf tak-berbobot sehingga dapat digunakan algoritma BFS untuk menentukan penyelesaiannya.**

***Keywords*—Labirin, graf, BFS, shortest path.**

I. Pendahuluan

Teori graf adalah salah satu topik dalam matematika diskrit yang mempunyai sangat banyak aplikasinya pada dunia nyata. Diawali dengan pertanyaan Euler tentang tujuh jembatan Königsberg, bidang ilmu ini berkembang sangat pesat. Orang-orang menemukan bahwa banyak masalah sehari-hari, dari berbagai macam bidang, dapat dimodelkan dan diselesaikan dengan teori graf. Salah satunya adalah pertanyaan jalan apa yang harus ditempuh untuk mencapai suatu titik dari titik asal, dengan optimal dan sependek mungkin. Pertanyaan ini adalah pertanyaan yang dihadapi oleh fitur penunjuk arah pada GPS ataupun Google Maps, atau pada perancangan pabrik papan sirkuit (*circuit boards)* dimana kita harus menentukan posisi lajur-lajur konduktif antar komponen agar hanya menggunakan material sesedikit mungkin.

Pada contoh yang paling mudah, kita dapat melihat permasalahan ini pada permainan teka teki labirin, dimana kita harus mencari jalan dari titik awal ke titik akhir. Labirin adalah sebuah sistem jalur yang rumit, berliku-liku dan mungkin memiliki jalan buntu. Penulis memilih penggunaan labirin sebagai contoh pada makalan ini karena labirin adalah contoh yang cukup simpel untuk dipahami, namun cukup kuat untuk memodelkan permasalahan lain yang mirip.



Gambar 1a. Contoh jenis labirin yang dibahas di makalah ini.
(Sumber :
https://upload.wikimedia.org/wikipedia/commons/thumb/2/28/Prim_Maze.svg/1200px-Prim_Maze.svg.png)

Melalui studi pustaka, kita dapat menemukan bahwa permasalahan seperti ini disebut permasalahan jalan terpendek satu asal (*single-source shortest path*), dan ada beberapa algoritma yang dapat diaplikasikan untuk menyelesaikannya seperti BFS, Dijkstra, Bellman-Ford, dan A*. Pada makalah ini penulis memilih untuk menggunakan algoritma BFS karena algoritma tersebut adalah algoritma dasar yang ada pada teori graf, dan permasalahan labirin sebelumnya dapat dimodelkan sebagai graf yang dimana BFS dapat diaplikasikan. Selain permasalahan jalan terpendek, BFS juga dapat digunakan untuk menyelesaikan permasalahan lain seperti *Garbage Collection* pada bahasa pemrograman modern seperti Python dan Java.

Untuk contoh, penulis juga membuat sebuah implementasi algoritma BFS dengan bahasa pemrograman Python. Implementasi tersebut selain dapat menyelesaikan masalah jalan terpendek, juga menghasilkan animasi pencari BFS.

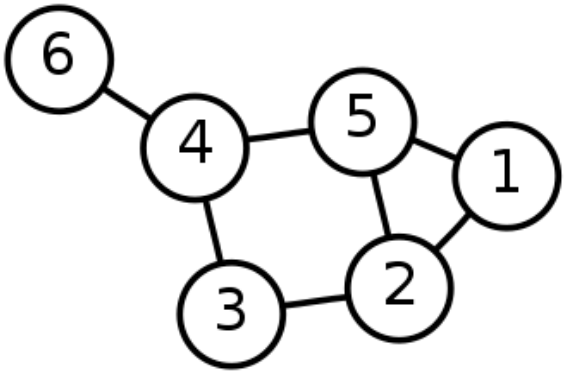
II. Graf

A. Definisi Graf

Graf (*Graph*) adalah himpunan objek-objek berupa simpul

(*vertex, node*), yang tiap simpulnya dihubungkan oleh sisi (*edge*). Dalam representasi visualnya graf dapat digambarkan dengan titik/bulat sebagai simpul dan garis sebagai sisinya.

Dua buah simpul dikatakan bertetangga apabila ada sekurangnya satu sisi yang menghubungkan antar dua simpul tersebut.



Gambar 2a. Contoh representasi visual sebuah graf (sumber: https://upload.wikimedia.org/wikipedia/commons/thumb/5/5b/6n-graf.svg/333px-6n-graf.svg.png)

B. Jenis-Jenis Graf

Menurut kearahan sisinya, graf dapat dibagi menjadi graf berarah dan graf tak-berarah. Graf berarah (*Directed graph*) adalah graf yang sisinya memiliki arah, sehingga sebuah sisi yang menghubungkan A→B tidak sama dengan sisi yang menghubungkan B→A. Graf berarah berguna salah satunya dalam pemodelan jalan kota dimana ada beberapa bagian jalan yang hanya satu arah. Graf tak berarah (*Undirected graph*) adalah graf yang sisinya tidak memiliki arah sehingga sisi yang menghubungkan A→B sama dengan sisi yang menghubungkan B→A. Graf tak-berarah berguna salah satunya pada pemodelan hubungan antar benda, atau hubungan pertemanan pada media sosial seperti *Facebook*.

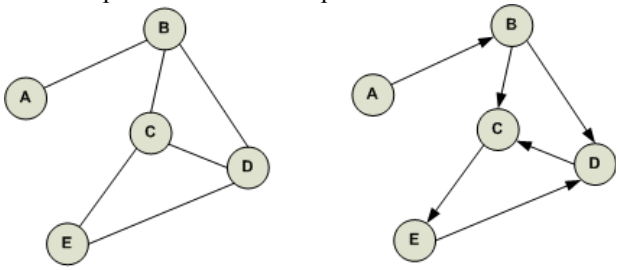
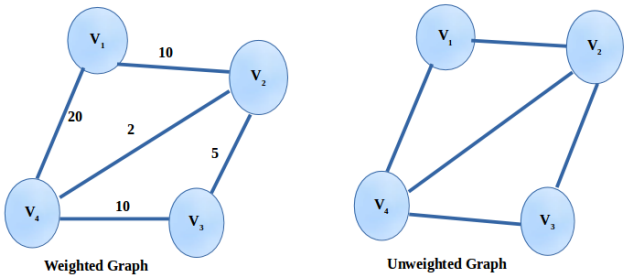


Fig 1. Undirected Graph Fig 2. Directed Graph

Gambar 2b. Contoh graf tak-berarah (kiri) dan graf berarah (kanan). (sumber : http://www.codediesel.com/wp-content/uploads/2012/02/d-graph1.gif)

Menurut bobot sisinya, graf terbagi menjadi graf berbobot dan graf tak-berbobot. Graf berbobot (*Weighted graph*) adalah graf yang sisinya memiliki bobot. Graf berbobot berguna salah satunya pada pemodelan peta dimana sebuah jalan memiliki jarak yang dapat dianggap sebagai berat sisi tersebut. Graf tak-berbobot (*Unweighted graph*) adalah graf yang sisinya tidak memiliki bobot. Dalam beberapa implementasi, sisi-sisi pada graf tak-berbobot dapat dianggap memiliki bobot 1 dan bobotnya tidak digambarkan. Graf tak-berbobot beguna pada contohnya pemodelan pertemanan pada media sosial dimana kita hanya mempedulikan hubungan antar orang.



Gambar 2c. Contoh graf berbobot (kiri) dan graf tidak berbobot (kanan). (sumber : http://www.codingeek.com/wp-content/uploads/2016/11/weighted.png)

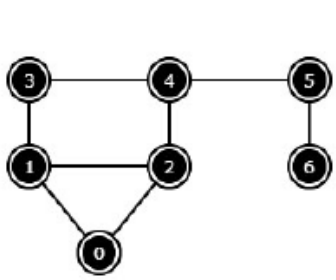
C. Representasi Graf di Komputer

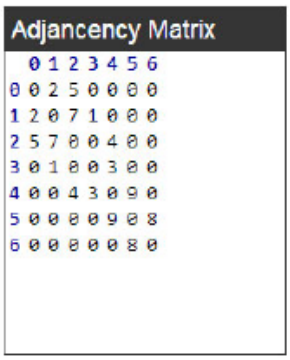
Pada umumnya, ada 3 macam jenis representasi graf di komputer. Pembagian ini ada karena beberapa jenis algoritma akan berjalan lebih cepat dengan sebuah representasi tertentu dibandingkan representasi lainnya.

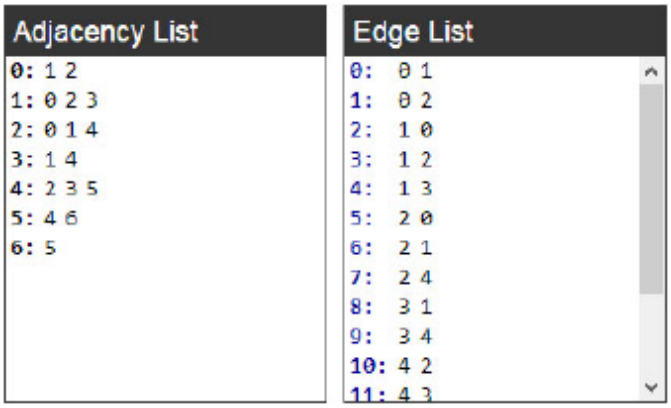
Pertama, representasi Matriks Ketetanggaan (*Adjacency Matrix*). Pada representasi ini, ada sebuah matriks dengan dimensi *n*n* dimana *n* adalah banyaknya simpul dalam graf. Untuk graf tak-berbobot, elemen matriks A_{ij} akan bernilai 1 jika ada sisi yang menghubungkan antara simpul *i dan j*, dan bernilai 0 jika tidak ada. Untuk graf berbobot, nilainya dapat berupa bobot sisi yang menghubungkan simpulnya. Kearahan sisi graf juga dapat direpresentasikan dengan menentukan persetujuan dimana A_{ij} merepresentasikan sisi antara simpul *i* dan *j*, sedangkan A_{ji} merepresentasikan sisi antara simpul *j* dan *i*, dan keduanya adalah sisi yang berbeda.

Kedua, representasi Larik Ketetanggaan (*Adjacency List*). Pada representasi ini, untu setiap simpul pada graf, ada larik (*list*) yang berisi simpul-simpul yang bertetanggaan. Representasi ini adalah representasi yang umunya digunakan pada algoritma pencarian jalur.

Ketiga, representasi Larik Sisi (*Egde List*). Pada representasi ini, ada sebuah larik yang berisi sisi-sisi suatu graf. Representasi ini berguna jika dibutuhkan daftar seluruh sisi graf, seperti pada algoritma kruskal.







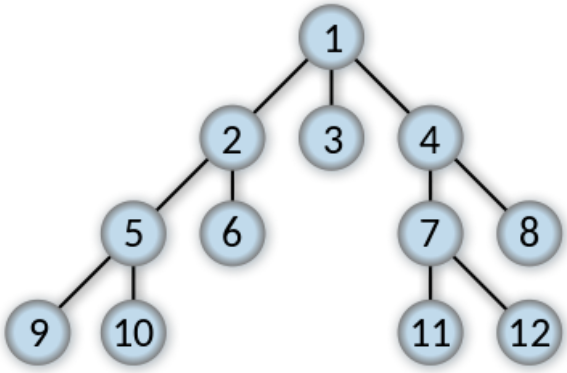
Gambar 2d. Contoh representasi-representasi sebuah graf. (sumber : S. Halim and F. Halim, *Competitive Programming* 3, Singapore, Lulu. 2013.)

III. Struktur Data Antrian

Struktur data antrian (*queue*) adalah sebuah struktur data mirip larik, dimana antrian juga dapat menyimpan banyak objek. Namun, kita hanya dapat memasukkan objek ke belakang antrian (*push*) dan mengeluarkan objek dari depan antrian (*pop*). Urutan objek didalam *queue* adalah FIFO (*First In First Out*), yang berarti objek yang lebih dahulu dimasukkan, akan lebih dahulu dikeluarkan seperti antrian pada umunya.

IV. Algoritma BFS

Algoritma BFS adalah algoritma yang melakukan pencarian jalur terpendek satu asal (*single-souce shortest path*) pada graf, dari suatu simpul asal. Algoritma ini akan menelusuri semua simpul yang bertetangga dengan simpul awal, lalu akan menelusuri semua simpul yang bertetangga dengan simpul yang sudah ditelusuri sebelumnya.



Gambar 4a. Urutan penelusuran simpul pada algoritma BFS, dimulai dari simpul 1 (sumber : https://upload.wikimedia.org/wikipedia/commons/3/33/Breadth-first-tree.svg)

Algoritma BFS membutuhkan graf yang akan ditelusuri berupa graf tak-berbobot. Algoritma BFS dijamin akan menemukan sebuah penyelesaian(jika ada) jalur dari simpul awal ke simpul akhir yang terpendek, namun dapat memakan waktu lama jika jarak simpul awal ke simpul akhir jauh, dengan kompleksitas algoritma $O(|V| + |E|)$ dimana $|V|$ adalah banyak simpul dan $|E|$ adalah banyak sisi.

Algoritma BFS selain membutuhkan graf tak-berbobot dan struktur data antrian untuk mengatur urutan penelusuran graf. BFS juga membutuhkan graf yang direpresentasikan dalam bentuk larik ketetanggaan agar dapat menentukan tetangga sebuah simpul dengan cepat.

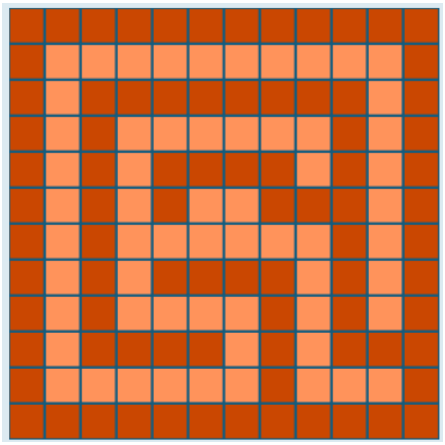
Setelah algoritma pencarian jalan selesai, dapar dilakukan pelacakan mundur(backtracking) untuk menentukan urutan sisi yang ditemukan.

```
queue defined
graph defined
start_node, end_node defined and is in
graph.nodes
queue.push(start_node)
start_node.visited = true
start_node.distance = 0
while queue is not empty:
        current_node = queue.pop()
        for nodes in current_node.adjacent_nodes:
                queue.push(node)
                node.visited = true
                node.distance =
current_node.distance + 1
# check if end_node is visited
if end_node.visited:
        A possible path found, we can use
backtracking to find the edges of the path
else:
        No possible path found
```

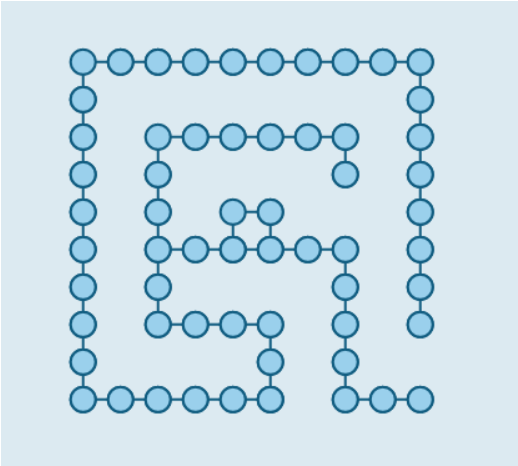
Psudeocode algoritma BFS

V. Pemodelan Labirin Dengan Graf

Dari pembahasan sebelumnya, kita dapat memperhatikan bahwa untuk menyelesaikan sebuah labirin seperti pada bab I dengan graf, kita harus memodelkan labirin tersebut dengan graf tak-berbobot. Tak berbobot dikarenakan kita menganggap titik tengah sebuah kotak labirin yang kosong sebagai simpul grafnya, dan semua sisi yang bersentuhan dengan kotak tersebut sebagai sisinya. Tetapi bisa dalam beberapa implementasi dimana objek labirin pasti berbentuk kotak-kotak, kita dapat mengakalinya dengan langsung mengaplikasikan algoritma BFS ke labirin tersebut sehingga tidak harus membuat grafnya terlebih dahulu, dengan begitu kita dapat menghemat waktu dan memori.



Gambar 5a. Contoh labirin sebelum pemodelan



Gambar 5b. Hasil pemodelan labirin pada gambar 5a. (sumber : http://bryukh.com/labyrinth-algorithms/)

Pada beberapa jenis labirin yang memungkinkan kita hanya bisa berjalan pada satu arah, kita dapat memodelkan sisinya sebagai sisi berarah. Selama kita menggunakan larik ketetanggaan sebagai representasi graf, kita tetap dapat mengaplikasikan algoritma BFS.

VI. Implementasi

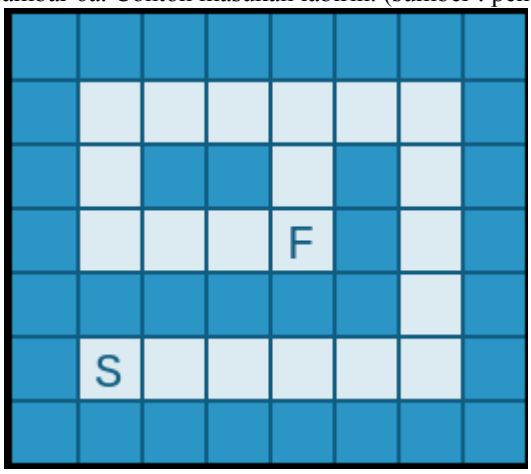
Sebagai contoh untuk makalah ini, penulis memilih menggunakan bahasa pemrograman Python dengan library Pillow untuk menghasilkan gambar visualisasi.

Pada awalnya file yang berisi data bentuk labirin. Karakter `#` digunakan untuk menandakan dinding dan karakter `.` (titik) digunakan untuk menandakan kotak kosong. Karakter `S` dan `F` digunakan untuk menandakan kotak awal dan kotak tujuan di labirin, kedua kotak tersebut dianggap kosong. Program juga akan membuat sebuah *queue* sebagain penyimpanan urutan kotak.

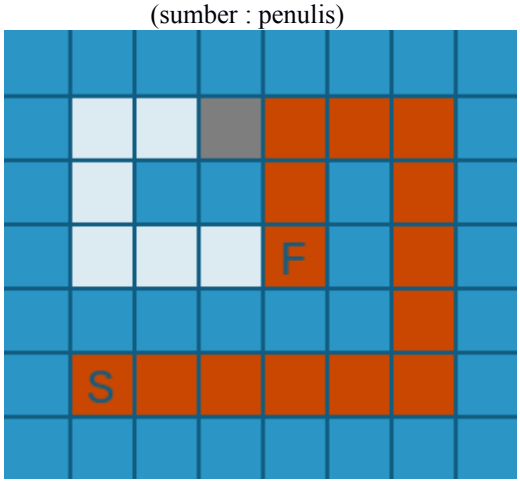
Kemudian program akan melakukan algoritma dimulai dari kotak S hingga algoritma sampai ke kotak F. Apabila tidak ada jalan dari S ke F ditemukan, maka queue akan menjadi kosong dan tidak akan bertambah karena semua kotak yang mungkin sudah ditelusuri. Setiap penelusuran sebuah kita juga mencatat jarak kotak sekarang dari kotak S.

Saat algoritma BFS sampai ke kotak F, maka algoritma BFS akan berhenti. Dari sini kita bisa melakukan pelacakan balik dimulai dari kotak F. Kita harus mengecek untuk seluruh kotak yang bertetanggaan dengan F, kotak mana saja yang nilai jaraknya kurang satu dari nilai jarak kotak F, apabila ada beberapa kemungkinan, kita boleh memilih sembarang dari salah satu kemungkinan tersebut. Lalu untuk kotak yang sudah dipilih tadi, kita kembali mencari kotak sekelilingnya yang nilai jaraknya kurang satu dari nilai jarak kotak sekarang, dan memilihnya sebagai kotak selanjutnya. Hal ini kita ulang terus menerus hingga kita sampai ke kotak S. Seluruh kotak yang telah kita pilih sebelumnya, termasuk kotak S dan F, adalah kotak-kotak yang membentuk jalan terpendek dari S ke F. Tidak menutup kemungkinan ada beberapa jalan terpendek yang bisa ditemukan.

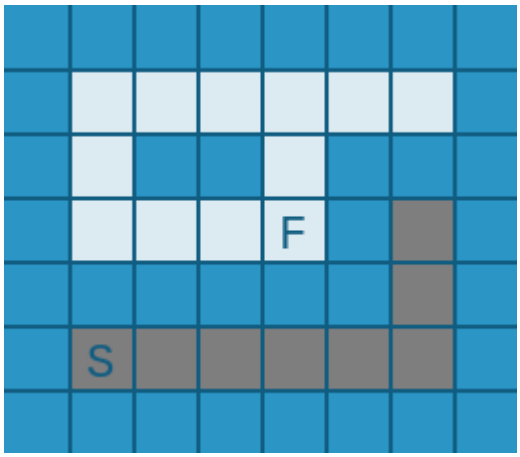
Gambar 6a. Contoh masukan labirin. (sumber : penulis)



Gambar 6b. Hasil labirin dengan masukan pada gambar 6a. (sumber : penulis)



Gambar 6c. Hasil pengaplikasian algoritma BFS pada labirin di gambar 6b. Kotak berwarna jingga tua menandakan jalan terpendek dari S ke F. (sumber : penulis)



Gambar 6d. Hasil pengaplikasian algoritma BFS pada labirin berbeda. Pada labirin ini tidak ada jalan ditemukan dari S ke F. Kotak abu-abu menandakan kotak-kotak yang sudah ditelusuri oleh algoritma BFS. (Sumber : penulis).

Source code contoh aplikasi dapat dilihat di https://github.com/ridho9/bfs-animation.

VII. Kesimpulan

Dari makalah ini dapat kita lihat bahwa algoritma BFS dapat digunakan untuk menemukan jarak terpendek dari suatu tempat ke tempat lainnya, selama kita dapat memodelkan pertanyaan sebagai graf tak-berbobot atau berbobot 1. Kita juga dapat mengaplikasikan algoritma BFS ke graf berbobot dengan memecah sisinya menjadi sisi-sisi yang berbobot satu. Kelemahan algoritma BFS adalah untuk graf yang besar dengan jarak dari simpul awal dengan simpul akhir yang jauh dibutuhkan waktu yang lama, karena BFS akan menelusuri seluruh simpul yang jaraknya kurang dari jarak simpul awal ke simpul akhir, namun selama ada jalan antara simpul awal dan akhir, BFS dijamin akan bisa menemukan jalan terpendek.

VIII. Ucapan Terima Kasih

Adapun makalah ini disusun dengan semaksimal mungkin dan mendapat dukungan dari berbagai pihak, sehingga memperlancar proses pembuatan makalah ini. Oleh karena itu, saya mengucapkan terimakasih kepada semua pihak yang telah membantu saya dalam pembuatan makalah ini. Saya juga mengucapkan terima kasih kepada ibu Dra. Harlili selaku dosen mata kuliah IF2120 Matematika Diskrit kelas K02.

Daftar Pustaka

[1] Halim, Steven and Halim, Felix. 2013. *Competitive Programming 3*, Singapore : Lulu.
[2] Munir, Rinaldi. 2006. *Diktat Kuliah IF2120 Matematika Diskrit*. Bandung : Institut Teknolgi Bandung.
[3] Rosen, Kenneth H.,2012. *Discrete Mathematics and Its Application. 7th Edition*. New York: McGraw-Hill.

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, 3 Desember 2017

Ridho Pratama, 13516032