Penerapan algoritma BFS pada penentuan jalur mudik

Azis Adi Kuncoro – 13515120
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
azisadikuncoro@gmail.com
13515120@std.stei.itb.ac.id

***Abstract*—Dewasa ini perjalanan ke kampung halaman atau yang biasa disebut mudik sudah menjadi tradisi sejak nenek moyang. Banyak rute – rute alternatif menuju kota yang akan kita kunjungi. Namun kita belum bisa menerka rute mana yang sebaiknya kita lalui maupun rute mana yang seharusnya kita hindari. Dengan memanfaatkan algoritma BFS yang menelusuri kemungkinan menuju kota Tujuan dengan cara menelusuri secara meluas diharapkan dapat membantu menyelesaikan permasalahan ini. Persoalan ini representasinya menggunakan graf tidak berarah untuk jarak antar kotanya, dan menggunakan graf berarah untuk level kemacetan antar kota. Hal ini dikarenakan volume kendaraan yang berada di sisi kiri jalan belum tentu sama dengan volume kendaraan yang berada di sisi kanan.**

Keywords—Mudik, BFS, Rute, Yogyakarta

I. PENDAHULUAN

Gambar 1. Ilustrasi Mudik

Mudik adalah kegiatan perantau/pekerja migran untuk kembali ke kampung halamannya. Mudik di Indonesia identik dengan tradisi tahunan yang terjadi menjelang hari raya besar keagamaan misalnya menjelang Lebaran. Pada saat itulah ada kesempatan untuk berkumpul dengan sanak saudara yang tersebar di perantauan, selain tentunya juga sowan dengan orang tua. Transportasi yang digunakan antara lain : pesawat terbang, kereta api, kapal laut, bus, dan kendaraan pribadi seperti mobil dan sepeda motor, bahkan truk dapat digunakan untuk mudik. Tradisi mudik muncul pada beberapa negara berkembang dengan mayoritas penduduk Muslim, seperti Indonesia dan Bangladesh.

Jumlah mudik lebaran yang terbesar dari Jakarta adalah menuju Jawa Tengah. Secara rinci prediksi jumlah pemudik tahun 2014 ke Jawa Tengah mencapai 7.893.681 orang. Dari jumlah itu didasarkan beberapa kategori, yakni 2.023.451 orang pemudik sepeda motor, 2.136.138 orang naik mobil, 3.426.702 orang naik bus, 192.219 orang naik kereta api, 26.836 orang naik kapal laut, dan 88.335 orang naik pesawat. Bahkan menurut data Kementerian Perhubungan Indonesia menunjukkan tujuan pemudik dari Jakarta adalah 61% Jateng dan 39% Jatim. Ditinjau dari profesinya, 28% pemudik adalah karyawan swasta, 27% wiraswasta, 17% PNS/TNI/POLRI, 10% pelajar/mahasiswa, 9% ibu rumah tangga dan 9% profesi lainnya. Diperinci menurut pendapatan pemudik, 44% berpendapatan Rp. 3-5 Juta, 42% berpendapatan Rp. 1-3 Juta, 10% berpendapatan Rp. 5-10 Juta, 3% berpendapatan dibawah Rp. 1 Juta dan 1% berpendapatan di atas Rp. 10 Juta.

Mudik merupakan istilah untuk kegiatan eventual biasanya terjadi pada saat menjelang hari raya Idul Fitri yang melibatkan masyarakat indonesia yang statusnya sedang berada di luar kampung halaman (biasanya merantau) dan berencana melakukan perjalanan pulang ke kampung halamannya. Mudik merupakan kepanjangan dari “Mulih Udik” yang berarti pulang ke kampung.

Dewasa ini, budaya mudik sudah menjadi rutinitas masyarakat indonesia. Dengan bertambahnya penduduk yang melakukan mudik pada tiap tahunnya, memicu kemacetan di beberapa daerah. Salah satu penyebabnya adalah kurang dimanfaatkannya rute alternatif, sehingga pengguna jalan kebanyakan berada pada rute yang sudah macet namun tidak

dapat melihat rute alternatif dikarenakan sudah mengambil rute yang sedang dilalui.

Berdasarkan data statistik dari *Voice+* terhadap penggunaan alat transportasi untuk mudik pada tahun 2013. Didapatkan bahwa masyarakat menggunakan alat transportasi Mobil (59%), Motor(14%), Pesawat (13%), Bus (7.4%), Kereta (3.9%), Kapal (2.9 %). Lebih dari separuh masyarakat indonesia yang mudik pada tahun 2013 menggunakan jalur darat dalam mudik ke kampung halamannya. Sehingga besar kemungkinan terjadinya kemacetan dikarenakan padatnya volume kendaraan yang melalui rute tersebut.

Dengan adanya algoritma BFS kemacetan yang tidak dapat diprediksi dapat di bantu, membantu mempermudah proses pencarian jalur yang lebih efisien untuk dilalui pengguna jalan

Melalui matakuliah IF2211 Strategi Algoritma, saya mendapat titik cerah untuk membantu meminimalisir terjadinya kemacetan yang bertumpu di suatu rute. Pendekatan yang saya gagas pada makalah ini dengan algoritma Breadth First Search dengan sedikit modifikasi.

Gambar 2. Suasana Macet saat mudik

II. DASAR TEORI

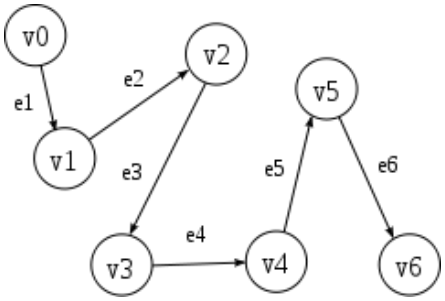
A. *Graf*

Dalam dunia Ilmu Komputer istilah graf sering dikaitkan dengan himpunan benda – benda yang disebut simpul, yang terhubung oleh sisi.

Graf terbagi menjadi 2 jenis, yaitu :

- Graf berarah
- Graf tidak berarah

Graf berarah, dan Graf tidak berarah. Graf berarah menandakan adanya hubungan dari simpul asal ke simpul tujuan, namun belum tentu sebaliknya. Sedangkan graf tidak berarah menunjukkan ada hubungan timbal balik antara simpul asal dengan simpul tujuan.



Gambar 3. Graf Berarah

B. *Algoritma Penelusuran Graf*

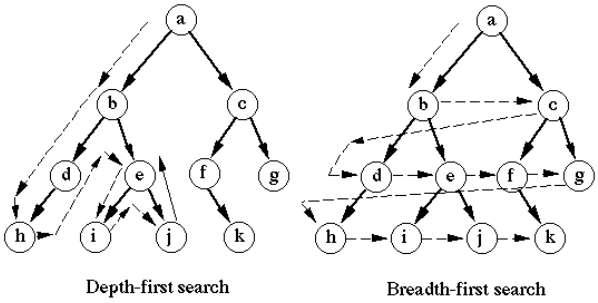
Ada banyak pendekatan untuk menelusuri graf, sehingga sampai pada simpul yang diinginkan (*Goal node)*, diantaranya : Breadth First Search, Depth First Search, Uniform Cost Search, A*, Greedy Best First Search, dan banyak lagi variasi pencariannya. Tiap – tiap metoda pencarian memiliki karakter tersendiri.

Algoritma A * (diucapkan sebagai "bintang A" (listen)) adalah algoritma komputer yang banyak digunakan dalam pathfinding dan graph traversal, proses merencanakan jalur yang diarahkan secara efisien antara beberapa titik, disebut node. Ia menikmati penggunaan yang meluas karena kinerjanya dan keakuratannya. Namun, dalam sistem perutean perjalanan praktis, umumnya mengungguli algoritma yang dapat melakukan pra-proses grafik untuk mencapai kinerja yang lebih baik, walaupun karya lain telah menemukan A * lebih unggul dari pendekatan lainnya.

DFS (Depth-First-Search) adalah salah satu algoritma penelusuran struktur graf / pohon berdasarkan kedalaman. Simpul ditelusuri dari root kemudian ke salah satu simpul anaknya (misalnya prioritas penelusuran berdasarkan anak pertama [simpul sebelah kiri]), maka penelusuran dilakukan terus melalui simpul anak pertama dari simpul anak pertama level sebelumnya hingga mencapai level terdalam. Setelah sampai di level terdalam, penelusuran akan kembali ke 1 level sebelumnya untuk menelusuri simpul anak kedua pada pohon biner [simpul sebelah kanan] lalu kembali ke langkah sebelumnya dengan menelusuri simpul anak pertama lagi sampai level terdalam dan seterusnya.

BFS (Breadth First Search) juga merupakan salah satu algoritma penelusuran struktur graf / pohon seperti DFS, namun bedanya BFS melakukan pencarian secara melebar atau per level pohon. Simpul ditelusuri dari root kemudian menelusuri semua simpul pada setiap level di bawahnya (misalnya prioritas penelusuran dari kiri ke kanan), maka penelusuran dilakukan terus dari simpul paling kiri ke simpul anak – anak tetangganya yang selevel.

Pencarian pendalaman iteratif atau lebih spesifik iterative deepening depth-first search (IDS atau IDDFS) adalah strategi pencarian ruang / grafik negara di mana kedalaman kedalaman pencarian kedalaman pertama dijalankan berulang kali dengan kedalaman yang meningkat. Batas sampai tujuan ditemukan. IDDFS setara dengan pencarian pertama, tapi menggunakan lebih sedikit memori; Pada setiap iterasi, ia mengunjungi simpul-simpul di pohon pencarian dengan urutan yang sama seperti penelusuran pertama-tama, namun urutan kumulatif dimana simpul pertama kali dikunjungi secara efektif adalah keluasan-pertama.



Gambar 4. Ilustrasi DFS dan BFS

C. Representasi Graf

Graf mudah dipandang oleh mata manusia, namun dalam representasinya sebagai data dalam komputer, graf memiliki beberapa representasi, diantaranya :

- Matriks ketetanggaan (*Adjacency Matrix)*
- Matriks bersisian (Incidence Matrix)
- Senarai Ketetanggan (Adjacency List)

Adjacency Matrix adalah array 2D dengan ukuran V x V dimana V adalah jumlah simpul dalam grafik. Misalkan array 2D adalah adj [] [], sebuah slot adj [i] [j] = 1 menunjukkan bahwa ada edge dari vertex i ke vertex j. Matriks adjacency untuk grafik yang tidak berarah selalu simetris. Adjacency Matrix juga digunakan untuk merepresentasikan grafik tertimbang. Jika adj [i] [j] = w, maka ada edge dari vertex i ke vertex j dengan bobot w.

Incidence Matrix adalah matriks yang menunjukkan hubungan antara dua kelas objek. Jika kelas pertama adalah X dan yang kedua adalah Y, matriks memiliki satu baris untuk setiap elemen X dan satu kolom untuk setiap elemen Y. Entri pada baris x dan kolom y adalah 1 jika x dan y terkait (disebut kejadian Dalam konteks ini) dan 0 jika tidak.

Adjacency list. Ukuran array sama dengan jumlah simpul. Biarkan array menjadi array []. Sebuah array entri [i] mewakili daftar verteks yang terhubung yang berdekatan dengan simpul ke-i. Representasi ini juga dapat digunakan untuk merepresentasikan grafik tertimbang. Bobot tepi dapat disimpan di node linked list. Berikut ini adalah representasi daftar kedekatan dari grafik di atas.

D. Travelling Salesperson Problem

Secara umum, masalah salesman keliling (TSP) adalah masalah menemukan cara terbaik untuk mengunjungi semua simpul dalam satu graf dari sebuah titik sekali dan kemudian kembali ke titik asal. Cara terbaik itu sendiri bisa diukur dengan berbagai parameter, baik itu jumlah simpul (biasanya di graf tak berbobot), jarak, biaya, waktu, dll. Masalah salesman perjalanan biasanya dimodelkan dengan graf yang berarah, dan diselesaikan dengan menggunakan berbagai algoritma, Seperti brute force, Greedy, dan Branch and Bound.

Gambar 5. Ilustrasi TSP

Di luar Teknologi Informasi, ada yang berbeda Pendekatan terhadap masalah ini Sebagai contoh, industri Penelitian operasional teknik terkadang menggunakan linier Optimasi karena lebih mudah beradaptasi dengan yang lain masalah terkait. ada variasi masalah TSP yang disebut set TSP masalah. Set TSP Problem adalah variasi (atau lebih tepatnya, Generalisasi) masalah Traveling Salesman, di
Yang setiap simpul harus dikunjungi setidaknya sekali, sebagai lawan ke TSP biasa dimana setiap titik harus dikunjungi hanya sekali.

E. Algoritma Branch and Bound

Memecahkan TSP dengan cepat dan akurat biasanya membutuhkan sebuah algoritma pemecahan masalah. Salah satu yang paling umum untuk memecahkan masalah TSP adalah algoritma Branch and Bound. Branch and Bound (sering disingkat menjadi BnB) adalah sebuah metode sistematis untuk memecahkan masalah optimasi. ini Diperkenalkan oleh A.H. Land dan A.G. Doig pada tahun 1960. Algoritma BnB optimal, karena pencarian keseluruhan Ruang solusi untuk masalah yang diberikan untuk menemukan yang terbaik. Selanjutnya, ini adalah optimasi yang agak umum teknik itu berlaku dalam banyak kasus, bahkan di

tempat yang dimana algoritma greedy dan program dinamis gagal. Namun, terlepas dari kemungkinan penggunaan luas, cabang dan algoritma BnB umumnya lebih lambat dari greedy dan program dinamis. Algoritma ini memang lebih cepat daripada algoritma brute force, namun membutuhkan implementasi yang cermat agar benar-benar berjalan cepat.

Gagasan umum tentang BnB adalah algoritma pencarian mirip dengan BFS untuk solusi optimal, tapi Tanpa memperluas semua simpul (yaitu, anak-anak mereka). Sebaliknya, kriteria yang dipilih secara hati-hati menentukan simpul mana yang akan diperluas dan kapan untuk mengembangkannya, dan yang lainnya kriteria memberitahu algoritma bila solusi optimalnya telah ditemukan. Ada dua langkah utama di algoritma branch and bound, yang merupakan bagian "cabang" dan "bagian terikat". Bagian cabang adalah tempat pohon itu berkembang secara rekursif, dimana terikat adalah bagian dimana algoritma mencari lower / upper bound dan trim semua node yang tidak lagi layak sebagai jawaban Terikat itu sendiri berbeda untuk masing-masing masalah, di mana masalah unik mungkin ada pada mereka sendiri, heuristik untuk menentukan pasti terikat. Bahkan satu masalah akan memiliki beberapa cara yang sama valid untuk menentukan
batas.

Heuristik itu sendiri biasanya ditulis sebagai C (i) = f (i) + g (i) dimana c (i) adalah biaya untuk titik tertentu i, F (i) adalah biaya untuk mencapai titik awal i dari titik asal, dan g (i) adalah biaya untuk mencapai target vertex dari simpul i. Langkah umum dalam menggunakan cabang dan ikatan adalah:

1) Pilih titik asal, dan tentukan yang terikat nilai menggunakan heuristik jika dibutuhkan.

2) Secara rekursif memperluas grafik sambil terus menerus Menyimpan nilai terikat.

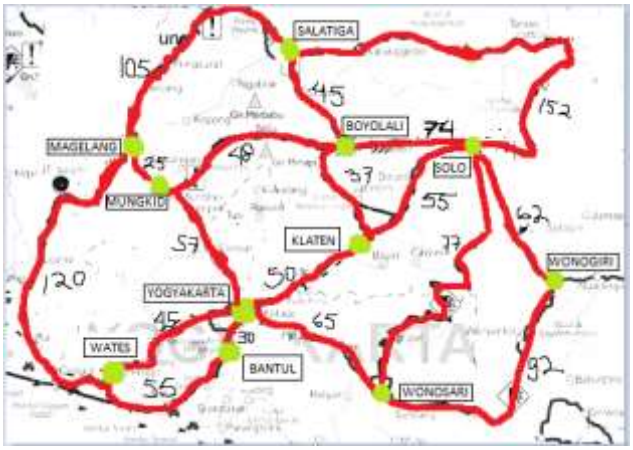
3) Ketika sebuah solusi ditemukan, bunuh semua simpul yang harganya terjangkau sudah lebih tinggi dari hasil (sehingga tidak mungkin untuk menjadi solusinya).

4) Lanjutkan dari vertex biaya terkecil secara rekursif sampai Tidak ada lagi kemungkinan perluasan.

III. STUDI KASUS : RUTE MUDIK YOGYAKARTA

Daerah Istimewa Yogyakarta dan sekitarnya merupakan salah satu destinasi mudik yang kerap dikunjungi pemudik, biasanya banyak perantau yang pulang ke kampung halamannya yang berada di sekitar yogyakarta.

Namun beberapa kendala dalam menuju kampung halaman ialah terjadinya macet di beberapa titik yang harus dihindari pengunjung. Berikut merupakan ilustrasi sebagian daerah di sekitar Yogyakarta beserta jarak antar kota.



Gambar 6. Rute mudik di sekitar Yogyakarta

Dengan mengubah informasi jarak yang berada di gambar menjadi representasi *adjacency matrix* sehingga dalam hal pemrosesan data oleh program menjadi jauh lebih mudah, hasil representasi *adjacency matrix* dapat dilihat pada tabel dibawah ini :

	A	B	C	D	E	F	G	H	I	J	K
A	-1	-1	-1	-1	-1	-1	-1	55	-1	-1	30
B	-1	-1	37	-1	48	45	74	-1	-1	-1	-1
C	-1	37	-1	-1	-1	-1	55	-1	-1	-1	50
D	-1	-1	-1	-1	25	105	-1	120	-1	-1	-1
E	-1	48	-1	25	-1	-1	-1	-1	-1	-1	57
F	-1	45	-1	105	-1	-1	152	-1	-1	-1	-1
G	-1	74	55	-1	-1	-1	-1	-1	62	77	-1
H	55	-1	-1	120	-1	-1	-1	-1	-1	-1	45
I	-1	-1	-1	-1	-1	-1	62	-1	-1	92	-1
J	-1	-1	-1	-1	-1	-1	77	-1	92	-1	65
K	30	-1	50	-1	57	-1	-1	45	-1	65	-1

Tabel 1. Tabel jarak

Pada representasi matriks tabel jarak “-1” berarti tidak terdapat jalur / rute dari node asal (kode pada baris paling kiri) ke node tujuan (kode pada kolom atas). Contoh : A ke C bernilai “-1” yang berarti tidak terdapat rute / jalur yang dapat ditempuh dari kota A menuju kota C.

Diberikan pula level kemacetan antar kota yang representasinya dalam matrix sebagai berikut.

	A	B	C	D	E	F	G	H	I	J	K
A	-1	-1	-1	-1	-1	-1	-1	1.1	-1	-1	1.2
B	-1	-1	1.3	-1	1.4	1.5	1.1	-1	-1	-1	-1
C	-1	1.7	-1	-1	-1	-1	1.3	-1	-1	-1	1.1
D	-1	-1	-1	-1	1.4	1.8	-1	1.9	-1	-1	-1
E	-1	1.3	-1	1.2	-1	-1	-1	-1	-1	-1	1.8
F	-1	1	-1	1.6	-1	-1	1.7	-1	-1	-1	-1
G	-1	1.5	1.4	-1	-1	-1	-1	-1	1.9	1.2	-1
H	1.1	-1	-1	1.3	-1	-1	-1	-1	-1	-1	1.3
I	-1	-1	-1	-1	-1	-1	1.4	-1	-1	1	-1
J	-1	-1	-1	-1	-1	-1	1.3	-1	1	-1	1
K	1.5	-1	1.1	-1	1.2	-1	-1	1	-1	1	-1

Tabel 2. Tabel level kemacetan

Dengan diketahuinya jarak dan level kemacetan, dapat dibentuk matriks cost antar kota, yang tiap – tiap elemennya adalah **jarak * level kemacetan.** Level kemacetan bervariasi dari rentang 1 hingga 2, 1 berarti rute normal, tidak ada faktor penghambat, namun jika lebih dari satu terdapat faktor tambahan yang menghambat rute tersebut, jika levelnya 2, menandakan rute tersebut benar – benar tidak efektif untuk dilalui, karena jarak tempuhnya akan terasa 2 kali lipat perjalanan yang lancar.

Berikut merupakan keterangan kode huruf pada kedua tabel diatas :

BANTUL	A
BOYOLALI	B
KLATEN	C
MAGELANG	D
MUNGKID	E
SALATIGA	F
SOLO	G
WATES	H
WONOGIRI	I
WONOSARI	J
YOGYAKARTA	K

Tabel 3. Kode huruf yang merepresentasikan kota

Dengan menggunakan kode alfabet diatas, memudahkan program dalam melakukan pemetaan terhadap kota – kota yang ada di dalam kasus ini.

A. Cuplikan kode program

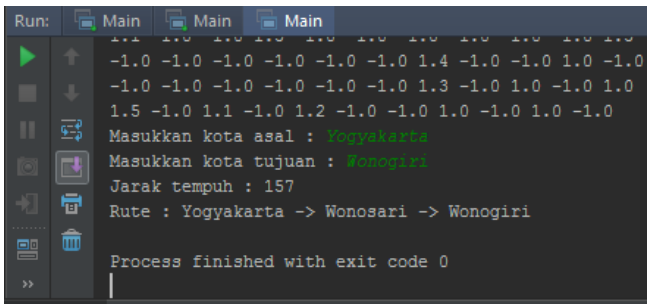
```
Function BFS(Node currentNode, Node goal) : Integer
{ Melakukan penelusuran BFS dengan cara mencari minimum
dari semua kemungkinan child node. Sehingga dihasilkan
cost untuk menuju kota tujuan }

If (!currentNode.equals(goal)) {
  Int min = 99999;
  Foreach (Node child : currentNode.child()) {
    Int val = BFS(child,goal));
    If (val < min) {
      min = val;
    }
  }
  Return currentNode.cost() + min;
} else {
  Return 0;
}
```

Tabel 2. Pseudocode algoritma BFS

IV. UJI COBA PROGRAM

Dalam pengujian program ini, penulis menggunakan salah satu testcase guna menguji kevalid'an program yang dibuat.

Contoh testcasenya ialah sebagai berikut, diberikan kota asal ialah kota Yogyakarta, dan diberikan pula kota tujuan yaitu kota Wonogiri, program akan melakukan pemrosesan menggunakan algoritma BFS untuk sampai ke kota wonogiri. Selanjutnya akan ditampilkan Jarak tempuh dari kota yogyakarta hingga kota wonogiri, dan juga rute mana saja yang harus dilalui pengguna jalan.



Gambar 7. Hasil uji program

Dari hasil eksekusi program tersebut, dapat diketahui jarak tempuh yang dilalui oleh pemudik, dan rute yang efisien untuk melakukan perjalanan dari Kota Yogyakarta ke Kota Wonogiri.

Ide dari program ini ialah, melakukan penelusuran ke simpul – simpul yang menjadi anak dari simpul sekarang secara BFS, hingga di temukan simpul tujuan. Jangan lupa simpan jalur – jalur yang dilalui untuk dapat menampilkan jarak tempuh dan rute yang dilalui dari kota asal hingga kota tujuan.

V. KESIMPULAN

Mudik merupakan fenomena tahunan yang perlu diwaspadai, dikarenakan memungkinkan terjadinya kemacetan, terutama pada jalur darat. Untuk meminimalisir kejadian tersebut, salah satunya adalah dengan memilih rute yang benar, sehingga rute – rute alternatif dapat dimanfaatkan dengan baik. Karena hal ini termasuk persoalan optimasi, sehingga salah satu pendekatan yang bisa dilakukan adalah dengan metode BFS.

Dari hasi percobaan program, algoritma ini berhasil menunjukkan kesuksesan dalam menunjukkan pengguna jarak tempuh beserta rute yang perlu dilalui untuk mencapai suatu tujuan. Dengan hal ini, dapat mengantisipasi kemacetan yang ada di jalur darat dengan mempertimbangkan level kemacetan di suatu rute.

VI. UCAPAN TERIMAKASIH

Saya mengucapkan terimakasih kepada Allah SWT atas berkat dan karunianya sehingga saya dapat mendapatkan inspirasi untuk menulis makalh ini, Bapak Rinaldi Munir yang telah memberikan tugas ini sebagai pemacu saya untuk dapat memberikan publikasi publikasi bermanfaat mengenai keilmuan informatika.

REFERENCES

[1] http://stackoverflow.com

[2] http://houseofinfographics.com/infografis-lebaran-dan-mudik-indonesia-2013/?doing_wp_cron=1495120174.7674119472503662109375

[3] https://id.wikipedia.org/wiki/Mudik

[4] https://saungkode.wordpress.com/2014/04/16/penelusuran-pohon-biner-berdasarkan-kedalaman-dengan-algoritma-dfs-stack-dan-secara-melebar-level-order-dengan-algoritma-bfs-queue-dan-implementasinya-dalam-bahasa-c/

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, 19 Mei 2017

Azis Adi Kuncoro - 13515120