Implementasi Logika Penurunan Persamaan Aritmatika pada Program Komputer

Cendhika Imantoro - 13514037
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
cendhikaimantoro@s.itb.ac.id

***Abstract*—Di berbagai bidang disiplin, melakukan penghitungan suatu nilai tentunya diperlukan. Penghitungan ini selalu didasari dari beberapa persamaan dasar yang berhubungan. Dari beberapa persamaan dasar yang ada, berbagai macam besaran dapat dihitung. Karena penghitungan dibutuhkan, tentunya ada dorongan untuk melakukan otomatisasi terhadap hal ini dalam bentuk sebuah program komputer. Produk otomatisasi yang diharapkan tentunya harus bisa menurunkan persamaan dasar yang sudah ada untuk memperoleh nilai suatu variabel. Makalah ini membahas implementasi metode penurunan persamaan pada program komputer.**

***Keywords*— persamaan, penurunan, BFS, DFS, ekspresi matematika**

I. Pendahuluan

Kebutuhan manusia terus bertambah. Untuk memenuhi kebutuhan tersebut, manusia terus mengembangkan ilmu pengetahuan dan teknologi. Dari penelitian untuk pengembangan teknologi, berbagai pengetahuan baru muncul. Salah satu wujud pengetahuan yang dihasilkan adalah persamaan matematis.

Persamaan yang dihasilkan dari penelitian menyatakan hubungan antara satu variabel dengan variabel lain. Variabel yang muncul dalam sebuah persamaan merepresentasikan aspek yang benar-benar ada di sekitar kehidupan manusia. Dengan mengetahui hubungan antar variabel tersebut, teknologi terus berkembang.

Beberapa persamaan tersebut diaplikasikan pada berbagai alat yang ada saat ini. Di antara alat-alat tersebut, ada beberapa alat yang pengaruhnya sangat penting untuk kehidupan manusia. Kesalahan penghitungan dalam perancangan suatu alat yang penting tentu dapat merugikan manusia.

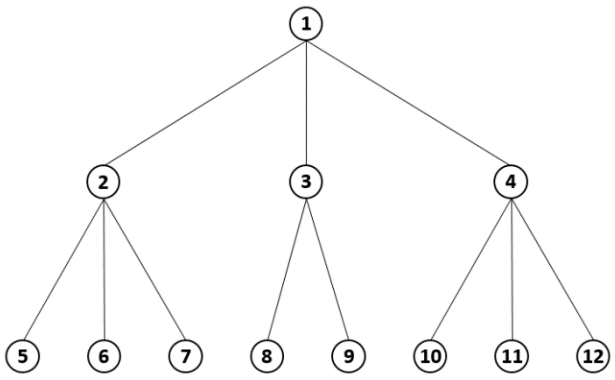
Namun, manusia tentu tidak terhindar dari kesalahan. Untuk meminimalkan kesalahan yang terjadi dalam penggunaan persamaan, otomatisasi penghitungan perlu dilakukan.

Untuk melakukan penghitungan, computer tentu harus tahu persamaan yang ada. Namun tentunya computer hanya menyimpan persamaan dasar. Tidak mungkin computer menyimpan smeua kemungkinan persamaan yang tak hingga jumlahnya. Untuk menyelesaikan persamaan yang tidak tersimpan, sebuah program computer harus menurunkan persamaan baru dari persamaan yang ada. Metode yang biasa digunakan manusia untuk menurunkan persamaan bermacam-macam. Dan implementasinya pada program computer lebih banyak lagi macamnya. Namun kelebihan computer yang diharapkan dari otomatisasi penghitungan tidak hanya akurasi dan konsistensi, kecepatan juga termasuk. Oleh karena itu, program yang ideal harus mampu memberikan jawaban dalam waktu singkat.

II. Breadth First Search dan Depth First Search

Algoritma Breadth First Search (BFS) dan Depth First Search (DFS) merupakan algoritma pemrograman yang berorientasi terhadap urutan pencarian solusi. Algoritma ini tidak mengarah ke perbaikan terhadap kompleksitas program untuk kasus ekstrim, namun dapat memperbaiki kompleksitas untuk kasus umum.

Dalam BFS, pertama harus ada *state* awal. State ini disebut sebagai *state* level 0. Langkah pertama untuk menyelesaikan masalah adalah memeriksa apakah *state* awal sudah mencapa solusi atau belum. Jika sudah, maka pencarain dihentikan. Jika belum, dibuat *state-state* baru yang merupakan *state* hasil eksekusi suatu langkah penyelesaian dari *state* awal. *State* yang dihasilkan di tahap ini disebut *state* level 1. Lalu pencarian dilanjutkan dengan melakukan pemeriksaan yang sama terhadap semua *state* level 1. Pemeriksaan semua *state* level 1 menghasilkan *state* level 2. Lalu pemeriksaan dilakukan terhadap *state* level 2. Pemeriksaan ini diulang hingga solusi ditemukan.

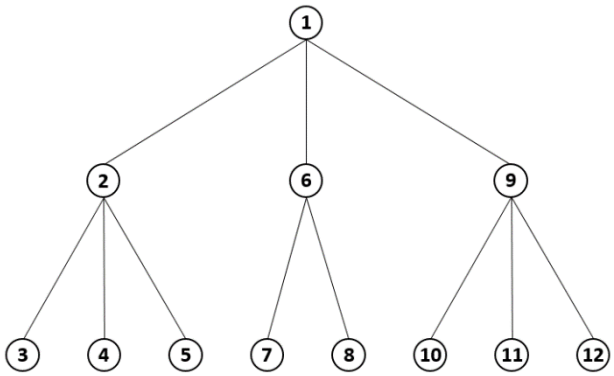


Gambar 1 urutan pemrosesan state algoritma BFS

Algoritma BFS memiliki urutan pemeriksaan state First In First Out. Oleh karena itu, dalam program biasanya algoritma BFS menggunakan struktur data queue. Dalam implementasi pada queue, algoritma BFS dapat dinyatakan dalam beberapa langkah, yaitu :

1. Masukkan state awal ke queue kosong.
2. Ambil elemen queue paling depan.
3. Jika elemen tersebut adalah solusi, akhiri pencarian. Jika bukan, buat semua state yang merupakan turunan langsung dari elemen tersebut, lalu masukkan ke dalam queue.
4. Ulangi langkah 2 dan 3 hingga solusi ditemukan.

Dalam DFS, perlakuan terhadap semua state mirip dengan BFS, namun berbeda di urutannya. Jika state yang diproses bukan solusi, maka state turunannya yang pertama diperiksa. Jika pemeriksaan state turunan tersebut tidak menghasilkan solusi, state turunan yang kedua diperiksa dan seterusnya. Jika semua state turunan tidak menghasilkan solusi, dapat disimpulkan state yang sedang diperiksa tidak menghasilkan solusi.



Gambar 2 urutan pemrosesan state algoritma DFS

Algoritma DFS memiliki urutan pemrosesan Last In First Out. Oleh karena itu, implementasinya bisa dilakukan menggunakan stack. Langkah pemrosesan yang dilakukan pada DFS dengan struktur data stack adalah :

1. Masukkan state awal ke stack.
2. Ambil elemen stack paling atas.
3. Jika elemen tersebut adalah solusi, pencarian dihentikan. Jika bukan, buat semua state yang merupakan turunan dari elemen yang sedang diproses, lalu masukkan ke stack.
4. Ulangi langkah 2 dan 3 hingga solusi ditemukan.

Selain stack, karena tiap state yang diproses merupakan turunan dari state yang sebelumnya diproses, algoritma DFS juga bisa diimplementasikan secara rekursif. Dalam implementasi secara rekursif, langkah pemrosesan pada DFS adalah :

1. Buat state awal.
2. Jika state awal merupakan solusi, pencarian dihentikan. Jika bukan, periksa state turunan pertama dengan algoritma DFS.
3. Jika state turunan pertama bukan solusi, periksa state turunan kedua dengan DFS dan seterusnya. Ulangi hingga solusi ditemukan atau semua state turunan tidak menghasilkan solusi.
4. Jika semua state turunan tidak menghasilkan solusi, dapat disimpulkan state yang sedang diproses tidak menghasilkan solusi.

Kelebihan algoritma BFS adalah selama masalah memiliki solusi dan pilihan langkah tidak tak hingga, solusi pasti ditemukan. Selain tu, jika level dari state ekuivalen dengan banyak lagkah untuk mencapai state tersebut, maka solusi yang diperoleh melalui BFS dapat dipastikan merupakan solusi dengan langkah minimum.

Kelebihan algoritma DFS adalah urutan pemrosesan memeriksa turunan terlebih dahulu sehingga dapat segera mengeliminasi state yang tidak menghasilkan solusi. Kekurangann DFS adalah untuk masalah tertentu, pohon dapat diekspan terus-menerus. Dalam kasus ini, DFS bisa tidak menghasilkan solusi.

III. Logika Penurunan Persamaan Aritmatka

Dalam penurunan persamaan, pertama ada tiga hal yang harus terdefinisi. Tiga hal ini berfungsi untuk menentukan titik awal, langkah penurunan persamaan yang mungkin, dan batasan pencarian. Tiga hal yang dimaksud adalah variabel yang dicari, variabel yang diketahui nilainya, dan persamaan dasar yang berlaku.

Persamaan	Diketahui
F = m x a	Ek = 12
a = dv / dt	a = 2
v = dx / dt	v = 2
a = v x v / r	
P = F / A	Dicari
W = m x g	F = ?
Ek = ½ x m x v x v	

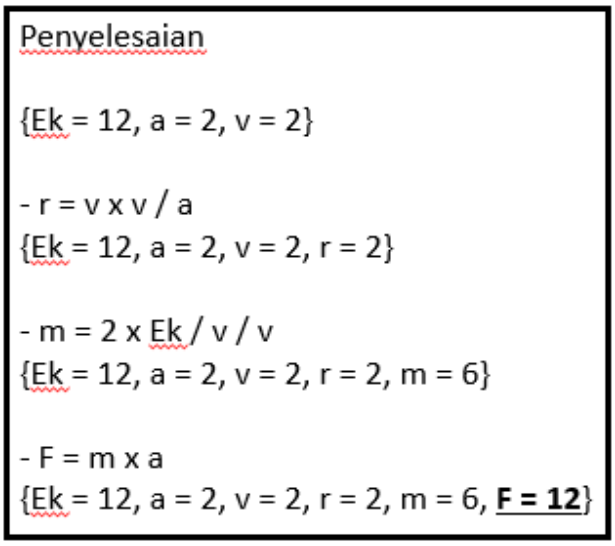
Gambar 3 informasi yang dibutuhkan untuk menurunkan persamaan

Setelah ketiga hal tersebut terdefinisi, ada dua cara untuk menemukan nilai dari variabel yang dicari. Kedua cara ini menggunakan titik awal yang berbeda, namun memberikan hasil yang sama.

Cara yang pertama adalah memulai penurunan persamaan dari variabel yang diketahui. Metode ini bertujuan mencari variabel apa yang nilainya dapat ditemukan dari variabel yang diketahui dan memasukkan variabel tersebut ke himpunan variabel yang diketahui. Lalu, dengan bertambahnya variabel yang diketahui nilainya, variabel lain juga dapat diketahui sehingga

himpunan ini akan terus membesar. Hal ini diulang hingga variabel yang dicari masuk ke dalam himpunan variabel yang diketahui nilainya.

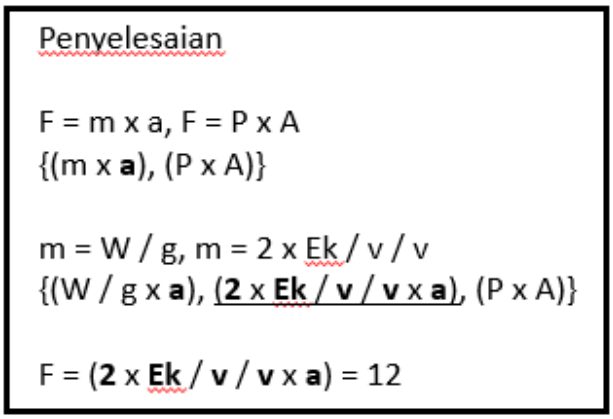
Sebagai contoh, diambil soal pada gambar 3. Pertama yang dilakukan adalah membuat himpunan {Ek, a, v}. Lalu, dari himpunan yang ada dan persamaan ke-4, nilai r dapat diketahui, sehingga r juga dimasukkan ke himpunan. Saat ini himpunan menjadi {Ek, a, v, r}. Dari himpunan baru dan persamaan ke-7, nilai m dapat diperoleh, sehingga himpunan menjadi {Ek, a, v, r, m}. Dari persamaan ke-1 dan himpunan baru, nilai F dapat diperoleh, sehingga himpunan menjadi {Ek, a, v, r, m, F}. Karena variabel yang dicari sudah ada dalam himpunan, pencarian dihentikan.



Gambar 4 penurunan persamaan dimulai dari variabel yang diketahui untuk deskripsi

Cara kedua adalah memulai penurunan persamaan dari variabel yang dicari nilainya. Dari persamaan yang ada, dapat diperiksa apa saja variabel yang dibutuhkan untuk memperoleh nilai dari variabel yang dicari. Dari variabel yang dibutuhkan tersebut, diperiksa variabel mana yang belum diketahui nilainya. Pemeriksaan yang sama dilakukan terhadap variabel yang belum diketahui nilainya tersebut. Hal ini diulang hingga himpunan variabel yang dibutuhkan merupakan subset dari variabel yang diketahui.

Sebagai contoh, digunakan soal pada gambar 3. Pertama dicari persamaan yang mengandung variabel F. Dari proses ini, diperoleh dua persamaan, yaitu persamaan 1 dan persamaan 5. Dari kedua persamaan tersebut, diperoleh dua ekspresi matematika yang menyatakan nilai F, yaitu (m*a) dan (P*A). Dari semua ekspresi yang muncul, variabel yang nilainya diketahui hanya a. Untuk itu, diambil satu variabel yang akan disubtitusi. Variabel yang diambil misalnya adalah m. Diantara persamaan yang ada, persamaan yang mengandung variabel m dan belum digunakan adalah persamaan 6 dan 7. Dari himpunan ekspresi F yang lama, semua ekspresi yang mengandung m dihilangkan dan semua ekspresi hasil subtitusi m dimasukkan. Dari proses ini, ekspresi F yang mungkin adalah (W/G*a), (2*Ek/v/v*a), dan (P*A). Dari ekspresi yang sudah terbentuk, ada ekspresi yang nilai semua variabelnya telah diketahui, yaitu ekspresi (2*Ek/v/v*a). Oleh karena itu,pencarian solusi dihentikan.



Gambar 5 penurunan persamaan dimulai dari variabel yang dicari

Metode pertama merupakan metode yang cenderung menggunakan pendekatan iterative. Yang dilakukan adalah memeriksa satu per satu anggota himpunan persamaan dasar. Kelebihannya adalah ketika variabel yang dicari tidak dapat diturunkan dari variabel yang diketahui, cirinya akan sangat mudah diidentifikasi, yaitu dengan memeriksa kondisi akhir iterasi. Jika iterasi telah memeriksa semua persamaan yang ada hingga himpunan tidak membesar lagi namun variabel yang dicari masih belum masuk himpunan, itu berarti variabel yang dicari tidak dapat diturunkan dari variabel yang diketahui. Kekurangan dari metode ini adalah himpunan juga harus mampu menyimpan ekspresi matematika (seperti perkalian variabel), dan elemen himpunan yang merupakan ekspresi matematika ini kemungkinan harus dikelola lebih lanjut seiring membesarnya himpunan karena dapat memperlambat pemrosesan lebih lanjut.

Metode kedua merupakan metode yang dapat diimplementasikan secara rekursif maupun iterative. Hal ini disebabkan oleh proses yang sama dilakukan untuk tiap state yang ada. Kelebihan dari metode ini adalah hanya menyimpan variabel yang dibutuhkan sehingga mencari persamaan dasar yang terlibat dalam penurunan persamaan cenderung lebih mudah. Selain itu, implementasi dari metode ini bisa dalam bentuk tree, sehingga history pembangunan solusi dapat tersimpan. Kekurangan dari metode ini adalah ketika variabel yang dicari tidak dapat diturunkan dari variabel yang diketahui, identifikasinya lebih rumit. Hal ini disebabkan tree yang dibentuk dapat diperbesar terus-menerus.

IV. Implementasi Penurunan Persamaan pada Program Komputer

A. Implementasi Metode Penurunan dari Variabel yang Diketahui

Konsep dasar dari metode ini adalah memperbesar himpunan variabel yang diketahui hingga variabel yang

dicari masuk ke himpunan atau himpunan tidak bisa diperbesar lagi.

Langkah pertama implementasi adalah membuat himpunan kosong dan mengisinya dengan nama variabel serta nilai dari variabel yang diketahui. Untuk mempermudah penjelasan, himpunan ini disebut sebagai himpunan K pada subbab ini. Selain itu, perlu dibuat sebuah iterator untuk melakukan iterasi pada list yang berisi persamaan dasar yang berlaku.

Langkah selanjutnya, yang dilakukan adalah membuat persamaan yang ekuivalen dengan persamaan dasar yang ditunjuk iterator, namun himpunan variabel pada ruas kanan persamaan harus merupakan subset dari himpunan K dan himpunan variabel pada ruas kiri persamaan tidak boleh beririsan dengan K. Misal himpunan variabel pada ruas kanan persamaan disebut sebagai R dan himpunan variabel pada ruas kiri persamaan disebut sebagai L. Maka persamaan yang dibuat harus memenuhi syarat

$$L \cap K = \emptyset$$
$$dan$$
$$R \cap K = R$$

Setelah persamaan baru diperoleh, ruas kiri beserta nilainya (ruas kanan) dari persamaan baru dimasukkan ke dalam K. Lalu, iterator maju dan dengan himpunan K yang baru langkah pembuatan persamaan diulang. Jika iterator mencapai akhir list, iterator kembali ke awal.

Jika variabel yang dicari telah masuk ke dalam himpunan K, pencarian dihentikan. Jika ada iterasi list penuh yang tidak menimbulkan perubahan terhadap K sama sekali, disimpulkan variabel yang dicari tidak dapat diturunkan dari variabel yang diketahui dan pencarian dihentikan.

B. Implementasi Metode Penurunan dari Variabel yang Dicari dengan BFS

Untuk implementasi metode penurunan dari variabel yang dicari menggunakan BFS, pertama perlu dibuat sebuah queue kosong. Lalu masukkan variabel yang dicari sebagai sebuah ekspresi matematika ke dalam queue.

Langkah yang perlu diulang adalah mengambil bagian queue paling depan dan memeriksanya. Jika semua variabel pada elemen yang diambil diketahui, pencarian selesai dan nilai variabel yang dicari sama dengan nilai dari ekspresi matematika yang diambil. Jika ada variabel yang tidak diketahui nilainya, ada beberapa langkah pemrosesan yaitu :

1. Ambil salah satu variabel yang tidak diketahui nilainya (misal X).
2. Cari semua persamaan yang mengandung X.
3. Buat persamaan ekuivalen yang ruas kirinya X dari persamaan yang diperoleh sebelumnya.
4. Untuk tiap ekspresi matematika di ruas kanan persamaan yang dihasilkan, buat sebuah ekspresi matematika yang merupakan hasil subtitusi X pada elemen yang diproses dengan ruas kanan persamaan.
5. Masukkan semua ekspresi matematika yang dihasilkan ke queue.

Pengulangan langkah tersebut dihentikan ketika solusi ditemukan atau Queue kosong. Jika berhenti saat queue kosong, itu berarti variabel yang dicari tidak dapat diturunkan dari variabel yang diketahui.

Ada kasus khusus, dimana queue tidak akan kosong meskipun variabel yang dicari tidak dapat diturunkan dari yang diketahui. Kejadian ini disebabkan munculnya kembali variabel yang sebelumnya sudah dihilangkan dengan subtitusi. Untuk mengatasi hal ini, selain ekspresi matematika, elemen queue juga harus mengandung daftar variabel yang pernah disubtitusi. Jika variable yang pernah disubtitusi muncul kembali, ada perlakuan khusus, yaitu tidak dimasukkan ke dalam queue lagi. Dengan begini, kasus infinite loop bisa terhindari.

C. Implementasi Metode Penurunan dari Variabel yang Dicari dengan DFS

Untuk implementasi metode penurunan dari variabel yang dicari dengan menggunakan DFS, ada dua cara. Yaitu dengan cara menggunakan stack dan dengan cara rekursif.

Untuk cara menggunakan stack, pertama harus ada stack kosong. Lalu variabel yang dicari dimasukkan ke dalam stack sebagai sebuah ekspresi matematika.

Untuk tiap loop, yang dilakukan adalah mengambil elemen teratas stack. Jika semua variabel pada ekspresi matematika yang terambil sudah diketahui, maka solusi ditemukan. Jika ada variabel yang belum diketahui, langkah yang dilakukan sama dengan BFS, yaitu :

1. Ambil salah satu variabel yang tidak diketahui nilainya (misal X).
2. Cari semua persamaan yang mengandung X.
3. Buat persamaan ekuivalen yang ruas kirinya X dari persamaan yang diperoleh sebelumnya.
4. Untuk tiap ekspresi matematika di ruas kanan persamaan yang dihasilkan, buat sebuah ekspresi matematika yang merupakan hasil subtitusi X pada elemen yang diproses dengan ruas kanan persamaan.
5. Masukkan semua ekspresi matematika yang dihasilkan ke stack.

Setelah solusi ditemukan atau stack kosong, pencarian solusi dihentikan. Jika berhenti karena stack kosong, dapat disimpulkan variabel yang dicari tidak dapat diturunkan dari yang diketahui.

Untuk implementasi DFS secara rekursif, yang dibutuhkan adalah sebuah fungsi rekursif. Fungsi yang perlu dibuat adalah sebuah fungsi dengan parameter ekspresi matematika, iterator list persamaan, dan variabel yang akan disubtitusi dari parameter ekspresi matematika. Return value dari fungsi ini bertipe ekspresi matematika.

Misal fungsi yang digunakan adalah F(x,y,z) dengan x

ekspresi matematika, y iterator, dan z nama variabel. Basis pertama dari fungsi rekursif adalah ketika persamaan yang ditunjuk y tidak mengandung z. Setelah basis pertama dilewati, variabel z pada x disubtitusi sesuai dengan persamaan yang ditunjuk oleh y. Misal hasil subtitusi adalah ekspresi matematika x2. Basis kedua adalah ketika semua variabel pada x2 sudah diketahui. Pada kasus ini, fungsi mengembalikan x2. Pada kasus lainnya (masih ada variael yang belum diketahui), dilakukan proses rekursif. Proses ini pertama memilih variabel yang akan disubtitusi selanjutnya, misal z2. Setelah itu membuat iterator persamaan y2 dan meletakkannya di bagian awal list. Lalu, memanggil fungsi F(x2,y2, z2) dan memeriksa return valuenya. Jika mengembalikan ekspresi tidak kosong, ekspresi inilah yang dikembalikan fungsi F(x,y,z). Jika yang diperoleh adalah ekspresi kosong, yang dilakukan adalah menggeser y2 ke persamaan selanjutnya dan memanggil kembali fungsi F(x2,y2,z2). Jika semua persamaan sudah diproses dan semua fungsi F(x2,y2,z2) mengembalikan ekspresi kosong, fungsi F(x,y,z) mengembalikan ekspresi kosong).

Sama dengan BFS, algoritma DFS akan mengalami infinite loop pada kasus dimana variabel yang telah disubtitusi muncul kembali. Penanganan kasus ini pada implementasi dengan stack sama dengan BFS, yaitu menambahkan daftar variabel yang telah disubtitusi pada elemen stack. Untuk implementasi secara rekursif, penanganannya adalah dengan menambah daftar variabel yang telah disubtitusi sebagai parameter dan menambahkan basis untuk kasus variabel yang telah disubtitusi muncul kembali, yaitu fungsi mengembalikan ekspresi kosong. Perlakuan khusus ini menyebabkan pencarian secara DFS menghilangkan node yang tidak akan menuju solusi. Oleh karena itu, setelah menambahkan perlakuan ini, algoritmanya berubah menjadi algoritma backtracking.

V. Simpulan

Secara umum, metode menurunkan persamaan ada dua, yaitu dimulai dari variabel yang dicari dan dimulai variabel yang diketahui. Untuk metode pertama, implementasinya dapat dilakukan dengan menggunakan struktur data set. Untuk metode kedua, dapat diimplementasikan dengan algoritma BFS dan DFS. Implementasi dengan algoritma BFS menggunakan Struktur data queue. Implementasi dengan DFS bisa dengan menggunakan struktur data stack dan menggunakan fungsi rekursif. Implementasi BFS dengan queue mirip dengan implementasi DFS dengan stack.

VI. Terima Kasih

Puji Syukur penulis ucapkan kepada Tuhan Yang Maha Esa karena hanya atas rahmat-Nya makalah ini dapat terselesaikan. Ucapan terima kasih juga penulis ucapkan kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen pembimbing mata kuliah IF2211 Strategi Algoritma yang telah memberikan ilmu kepada penulis yang merupakan dasar dari penulisan makalah ini.

References

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/stima15-16.htm

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, 9 Mei 2016

Cendhika Imantoro - 13514037