



**MODUL ALGORITMA  
(CCC-110)**

**MODUL 01  
KONSEP ALGORITMA**

**DISUSUN OLEH  
MALABAY,S.KOM,M.KOM**

**UNIVERSITAS ESA UNGGUL  
2020**

## Konsep Algoritma

### A. Kemampuan Akhir Yang Diharapkan

Setelah mempelajari modul ini, diharapkan mahasiswa mampu memahami konsep algoritma dan memahami Gerbang Logika.

### B. Uraian dan Contoh

Untuk menyelesaikan suatu masalah harus menggunakan komputer untuk memudahkannya, namun masalah yang ingin diselesaikan tidak dapat langsung diterima oleh komputer.

Agar komputer dapat menyelesaikan masalah, maka Anda perlu merumuskan langkah-langkah penyelesaian tersebut dalam suatu rangkaian instruksi. Komputerlah yang akan mengerjakan rangkaian instruksi tersebut, karena komputer dapat mengerjakannya dengan cepat, akurat, bahkan berulang-ulang tanpa kenal lelah dan bosan.

Sekumpulan instruksi yang merupakan penyelesaian masalah itu dinamakan program. Program “dimasukkan” ke dalam komputer, komputer mengerjakan instruksi di dalam program tersebut, lalu memberikan hasil atau keluaran yang diinginkan.

Tapi untuk memberikan instruksi anda harus membuat suatu program menggunakan bahasa yang dimengerti oleh komputer Sebagaimana dalam kehidupan manusia, hanya dapat memberikan perintah kepada orang lain dalam bahasa yang dimengerti olehnya.

Karena komputer adalah mesin, maka program harus ditulis dalam bahasa yang khusus dibuat untuk “berkomunikasi” dengan komputer. Bahasa komputer yang digunakan dalam menulis program dinamakan bahasa pemrograman.

Saat ini, dengan berkembangnya teknik pemrograman terstruktur, orang tidak lagi memecahkan masalah dengan langsung menulis programnya dalam bahasa pemrograman. Orang mulai memikirkan suatu cara penyelesaian masalah yang

akan diprogram dengan menekankan pada desain atau rancangan yang mewakili pemecahan masalah tersebut.

Desain ini independen dari bahasa pemrograman yang digunakan dari komputer yang menjalankan program. Desain menyajikan cara berpikir si pemrogram dalam menyelesaikan masalah. Desain berisi urutan langkah-langkah pencapaian solusi yang ditulis dalam notasi-notasi deskriptif. Urutan langkah-langkah yang sistematis untuk menyelesaikan sebuah masalah dinamakan algoritma. Di Artikel kali ini akan memberikan informasi tentang pengenalan Algoritma: Definisi, Ciri – ciri, dan contohnya.

**Mengenal Apa Itu Algoritma ?** Algoritma adalah deretan instruksi yang jelas dalam memecahkan masalah, yaitu untuk memperoleh keluaran yang diinginkan dari suatu masukan dalam jumlah waktu yang terbatas.

Algoritma yang ditulis dalam bahasa komputer dinamakan program. Bahasa komputer yang digunakan untuk menulis program dinamakan bahasa pemrograman. Orang yang membuat program komputer disebut pemrogram, dan kegiatan merancang dan menulis program disebut pemrograman, serta ada aktivitas menulis kode program dinamakan coding.

Pada pemrograman ditekankan pada pemecahan masalah, rancangan pemecahan masalah berisi urutan langkah-langkah pencapaian solusi ditulis dalam notasi deskriptif (notasi algoritmik)

Kata algorism berasal dari nama penulis buku Arab yang terkenal, yaitu Abu Ja'far Muhammad Ibnu Musa alKhuwarizmi (al-Khuwarizmi dibaca orang Barat menjadi algorism). Seorang ilmuwan Persia yang menulis al jabr w'al muqabala yang artinya "*Buku pemugaran dan pengurangan*" (The Book of Restoration and Reduction).

Perubahan kata dari kata algorism menjadi algorithm muncul karena kata algorism sering dikelirukan dengan arithmetic, sehingga akhiran -sm berubah menjadi -thm. Karena perhitungan dengan angka Arab sudah menjadi hal yang biasa/lumrah, maka lambat laun kata algorithm berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna

aslinya. Dalam bahasa Indonesia, kata algorithm diserap menjadi algoritma. Berikut definisi dari Algoritma :

Langkah – langkah yang dilakukan agar solusi masalah dapat diperoleh.

1. Suatu prosedur yang merupakan urutan langkah-langkah yg berintegrasi.
2. Algoritma Suatu metode khusus yang digunakan untuk menyelesaikan suatu masalah yang nyata.

Ciri – Ciri dari Algoritma,

Tidak semua urutan langkah penyelesaian masalah yang logis dapat disebut sebagai algoritma. Menurut Donald E. Knuth di dalam bukunya yang berjudul The Art of Computer Programming, algoritma mempunyai lima ciri penting yang meliputi:

Finiteness (keterbatasan), algoritma harus berakhir setelah mengerjakan sejumlah langkah proses. Algoritma harus berhenti setelah mengerjakan sejumlah langkah terbatas (berhingga). Barisan instruksi yang dibuat dalam suatu urutan tertentu, dimaksudkan agar masalah yang dihadapi dapat diselesaikan. Banyaknya instruksi atau langkah itu haruslah berhingga.

Jika tidak demikian, proses yang dilakukan akan memerlukan waktu yang relatif lebih lama dan diperoleh hasil yang tidak diperlukan atau tidak berhubungan dengan masalah yang ada, bahkan mungkin proses akan terus berlangsung walaupun solusi yang diharapkan telah diperoleh. Hasil akhir yang didapat merupakan solusinya atau informasi tidak ditemukannya solusi.

Program yang tidak pernah berhenti adalah program yang berisi algoritma yang salah. Suatu prosedur yang hanya akan berhenti jika mempunyai atau menghasilkan solusi disebut semi algoritma. Definiteness (kepastian), setiap langkah harus didefinisikan secara tepat dan tidak berarti ganda.

Algoritma harus efektif dan efisien. Setiap langkah harus sederhana sehingga dapat dikerjakan dalam sejumlah waktu yang masuk akal. Suatu algoritma dikatakan efektif jika algoritma tersebut dapat menghasilkan suatu solusi yang

sesuai dengan masalah yang diselesaikan. Dengan kata lain suatu algoritma harus tepat guna.

Suatu algoritma dikatakan efisien jika waktu proses dari algoritma relatif lebih singkat dan penggunaan memorinya lebih sedikit.

Pada beberapa sumber lain, ada tambahan ciri dari algoritma: Algoritma harus terstruktur. Urutan baris langkah-langkahnya yang digunakan harus disusun sedemikian rupa agar proses penyelesaiannya tidak berbelit-belit, sehingga memungkinkan waktu prosesnya akan menjadi relatif lebih singkat.

Hal ini akan memperlihatkan bahwa bagian-bagian dari proses tersebut dapat dibedakan secara jelas (bagian input, proses dan output). Dengan demikian memudahkan didalam melakukan pemeriksaan ulang. Suatu algoritma harus menghasilkan output yang tepat guna (efektif) dalam waktu yang relatif singkat dan penggunaan memori yang relatif sedikit (efisien) dengan langkah yang berhingga dan prosesnya berakhir baik dalam keadaan diperoleh suatu solusi maupun tidak adanya solusi. Sifat algoritma adalah:

1. Tidak menggunakan simbol atau sintaks dari suatu bahasa pemrograman tertentu.
2. Tidak tergantung pada suatu bahasa pemrograman tertentu.
3. Notasi-notasinya dapat digunakan untuk seluruh bahasa manapun.
4. Algoritma dapat digunakan untuk merepresentasikan suatu urutan kejadian secara logis dan dapat diterapkan di semua kejadian sehari-hari
5. Langkah-langkah pada algoritma haruslah logis. Secara umum, pihak (benda) yang mengerjakan proses disebut pemroses (processor). Pemroses tersebut dapat berupa manusia, komputer, robot, atau alat-alat mekanik/elektronik lainnya. Pemroses melakukan suatu proses dengan melaksanakan atau mengeksekusi algoritma yang menjabarkan proses tersebut. Melaksanakan algoritma berarti mengerjakan langkah-langkah di dalam algoritma tersebut.

6. Pemroses mengerjakan proses sesuai dengan algoritma yang diberikan kepadanya. Karena itu suatu algoritma harus dinyatakan dalam bentuk yang dapat dimengerti oleh pemroses. Jadi suatu pemroses harus :
7. Mengerti setiap langkah dalam algoritma.
8. Mengerjakan operasi yang bersesuaian dengan langkah tersebut.

Definisi lain Algoritma berarti "proses atau sekumpulan aturan yang harus diikuti dalam perhitungan atau operasi pemecahan masalah lainnya". Oleh karena itu, Algoritma mengacu pada seperangkat aturan / instruksi yang langkah demi langkah menentukan bagaimana suatu pekerjaan akan dieksekusi untuk mendapatkan hasil yang diharapkan.

Hal itu bisa dipahami dengan mengambil contoh memasak resep baru. Untuk memasak resep baru, seseorang membaca instruksi dan langkah-langkahnya dan menjalankannya satu per satu, dalam urutan yang diberikan. Hasil yang diperoleh adalah hidangan baru yang dimasak dengan sempurna. Demikian pula, algoritma membantu melakukan tugas dalam pemrograman untuk mendapatkan keluaran yang diharapkan.

Algoritma yang dirancang tidak bergantung pada bahasa, yaitu hanya instruksi biasa yang dapat diimplementasikan dalam bahasa apa pun, namun hasilnya akan sama, seperti yang diharapkan.

Karena seseorang tidak akan mengikuti instruksi tertulis untuk memasak resep, tetapi hanya yang standar. Demikian pula, tidak semua instruksi tertulis untuk pemrograman adalah algoritma. Agar beberapa instruksi menjadi Algoritma, ia harus memiliki karakteristik sebagai berikut:

Jelas dan Tidak Rancu: Algoritma harus jelas dan tidak ambigu. Setiap langkahnya harus jelas dalam semua aspek dan hanya mengarah pada satu makna.

Input yang Ditentukan Dengan Baik: Jika suatu Algoritma mengatakan untuk mengambil input, itu harus merupakan input yang terdefinisi dengan baik.

Output yang Terdefinisi dengan Baik: Algoritma harus dengan jelas mendefinisikan output apa yang akan dihasilkan dan juga harus didefinisikan dengan baik.

Finite-ness: Algoritma harus terbatas, yaitu tidak boleh berakhir dalam loop tak terbatas atau serupa.

Layak: Algoritma harus sederhana, umum, dan praktis, sehingga dapat dijalankan sesuai keinginan sumber daya yang tersedia. Itu tidak boleh mengandung beberapa teknologi masa depan, atau apapun.

Bahasa Independen: Algoritma yang dirancang harus bahasa-independen, yaitu harus hanya instruksi biasa yang dapat diimplementasikan dalam bahasa apapun, namun hasilnya akan sama, seperti yang diharapkan.

Bagaimana Mendesain Algoritma?

Untuk menulis Algoritma, hal-hal berikut diperlukan sebagai prasyarat:

Masalah yang harus diselesaikan dengan algoritma ini.

Kendala masalah yang harus diperhatikan saat menyelesaikan masalah.

Masukan yang akan diambil untuk menyelesaikan masalah tersebut.

Output yang diharapkan ketika masalah terpecahkan.

Solusi untuk masalah ini, dalam batasan yang diberikan.

Kemudian Algoritma ditulis dengan bantuan parameter di atas sehingga dapat menyelesaikan masalah.

Contoh: Pertimbangkan contoh untuk menambahkan tiga angka dan mencetak jumlahnya.

Langkah 1: Memenuhi prasyarat

Seperti dibahas di atas, untuk menulis sebuah algoritma, prasyaratnya harus dipenuhi.

Masalah yang harus diselesaikan dengan algoritma ini: Tambahkan 3 angka dan cetak jumlahnya.

Batasan soal yang harus diperhatikan saat menyelesaikan soal: Angka hanya boleh berisi angka dan tidak boleh ada karakter lain.

Masukan yang akan diambil untuk menyelesaikan masalah: Tiga angka yang akan ditambahkan.

Output yang diharapkan ketika masalah terpecahkan: Jumlah dari tiga angka yang diambil sebagai input.

Solusi untuk masalah ini, dalam batasan yang diberikan: Solusinya terdiri dari penjumlahan 3 angka. Ini dapat dilakukan dengan bantuan operator '+', atau sedikit bijak, atau metode lainnya.

Langkah 2: Mendesain Algoritma



Merancang Algoritma dengan bantuan prasyarat di atas:

Algoritma untuk menambahkan 3 angka dan mencetak jumlahnya:

Mulailah

Deklarasikan 3 variabel integer num1, num2 dan num3.

Ambil ketiga angka tersebut, yang akan ditambahkan, sebagai input masing-masing dalam variabel num1, num2, dan num3.

Deklarasikan jumlah variabel integer untuk menyimpan jumlah resultan dari 3 angka.

Tambahkan 3 angka dan simpan hasilnya dalam jumlah variabel.

Cetak nilai jumlah variabel

AKHIR

Langkah 3: Menguji algoritma dengan mengimplementasikannya.

Untuk menguji Algoritma, mari terapkan dalam bahasa C.

Satu masalah, banyak solusi: Solusi untuk suatu algoritma bisa atau tidak bisa lebih dari satu. Artinya saat mengimplementasikan algoritma, bisa ada lebih dari satu metode untuk mengimplementasikannya. Misalnya pada soal di atas untuk menjumlahkan 3 angka maka penjumlahannya bisa dihitung dengan berbagai cara seperti:

+ operator

Operator yang bijak

.. dll

Bagaimana Menganalisis Algoritma?

Agar Algoritma standar bagus, Algoritma standar harus efisien. Oleh karena itu, efisiensi suatu algoritma harus diperiksa dan dipertahankan. Ini bisa dalam dua tahap:

Analisis Priori: "Priori" berarti "sebelum". Oleh karena itu analisis Priori berarti memeriksa algoritma sebelum diimplementasikan. Dalam hal ini, Algoritma diperiksa ketika ditulis dalam bentuk langkah-langkah teoretis. Efisiensi Algoritma ini diukur dengan mengasumsikan bahwa semua faktor lain, misalnya, kecepatan prosesor, adalah konstan dan tidak berpengaruh pada implementasi. Ini biasanya dilakukan oleh perancang algoritma. Dalam metode ini, Kompleksitas Algoritma ditentukan.



Analisis Posterior: “Posterior” berarti “setelah”. Oleh karena itu analisis Posterior berarti memeriksa algoritma setelah diimplementasikan. Dalam hal ini, Algoritma diperiksa dengan mengimplementasikannya dalam bahasa pemrograman apa pun dan menjalankannya. Analisis ini membantu mendapatkan laporan analisis aktual dan nyata tentang kebenaran, ruang yang dibutuhkan, waktu yang dihabiskan, dll.

Apa itu Algoritma Complexity dan Bagaimana menemukannya?

Algoritma didefinisikan sebagai kompleks berdasarkan jumlah Ruang dan Waktu yang dikonsumsi. Oleh karena itu, Kompleksitas Algoritma mengacu pada ukuran Waktu yang diperlukan untuk mengeksekusi dan mendapatkan output yang diharapkan, dan Space yang diperlukan untuk menyimpan semua data (input, data sementara, dan output). Oleh karena itu, kedua faktor ini menentukan efisiensi suatu algoritma.

Dua faktor Kompleksitas Algoritma adalah:

Faktor Waktu: Waktu diukur dengan menghitung jumlah operasi kunci seperti perbandingan dalam algoritma pengurutan.

Faktor Ruang: Ruang diukur dengan menghitung ruang memori maksimum yang dibutuhkan oleh algoritma.

Oleh karena itu kompleksitas suatu algoritma dapat dibagi menjadi dua jenis:

Kompleksitas Ruang: Kompleksitas ruang suatu Algoritma mengacu pada jumlah memori yang diperlukan Algoritma ini untuk mengeksekusi dan mendapatkan hasilnya. Ini bisa untuk input, operasi sementara, atau output.

Bagaimana cara menghitung Kompleksitas Ruang?

Kompleksitas ruang dari suatu algoritma dihitung dengan menentukan 2 komponen berikut:

Secara informal, Algoritma adalah prosedur komputasi yang terdefinisi dengan baik yang mengambil beberapa nilai, atau kumpulan nilai, sebagai input dan menghasilkan beberapa nilai, atau kumpulan nilai, sebagai output. Algoritma dengan demikian adalah urutan langkah komputasi yang mengubah input menjadi output.

Kami juga dapat melihat algoritma sebagai alat untuk memecahkan masalah komputasi yang ditentukan dengan baik. Pernyataan masalah secara umum

menentukan hubungan input / output yang diinginkan. Algoritma menjelaskan prosedur komputasi khusus untuk mencapai hubungan input / output tersebut.

Meskipun saat ini Algoritma terutama dikaitkan dengan perangkat lunak dan komputer, asal-usulnya jauh di masa lalu. Mereka telah digunakan secara intuitif selama berabad-abad, misalnya dalam bentuk sistem pengaturan, instruksi, aturan permainan, denah arsitektur, dan partitur musik.

Bahkan beberapa buku seni bergambar pada periode Renaisans, misalnya “Empat Buku Pengukuran” Albrecht Dürer pada tahun 1525, sebenarnya merupakan instruksi terstruktur untuk memproduksi lukisan, patung, dan bangunan. Dalam sejarah musik, dari Bach dan Mozart hingga Schönberg dan Schillinger, melihat metode matematika dan bahkan perangkat mekanis kecil digunakan untuk membuat proses komposisi musik lebih mudah.

Petunjuk langkah demi langkah

Tapi apa sebenarnya yang dimaksud dengan istilah algoritma? Seiring waktu, deskripsi berikut telah muncul sebagai definisi: "Algoritma adalah prosedur untuk pengambilan keputusan atau instruksi tentang bagaimana bertindak yang terdiri dari sejumlah aturan terbatas" atau "... urutan terbatas dari instruksi dasar yang tidak ambigu yang persis dan mendeskripsikan secara lengkap cara untuk memecahkan masalah tertentu. " Ini berlaku terlepas dari apakah itu berkaitan dengan matematika, seni rupa atau musik. Pertimbangkan bagaimana Anda menggunakan komputer di hari-hari biasa. Misalnya, Anda mulai mengerjakan laporan, dan setelah Anda menyelesaikan satu paragraf, Anda melakukan pemeriksaan ejaan. Anda membuka aplikasi spreadsheet untuk melakukan beberapa proyeksi keuangan untuk melihat apakah Anda mampu membeli pinjaman mobil baru. Anda menggunakan browser web untuk mencari online untuk jenis mobil yang ingin Anda beli.

Anda mungkin tidak memikirkan hal ini dengan sangat sadar, tetapi semua operasi yang dilakukan oleh komputer Anda ini terdiri dari Algoritma. Algoritma adalah prosedur yang didefinisikan dengan baik yang memungkinkan komputer untuk memecahkan masalah. Cara lain untuk mendeskripsikan Algoritma adalah urutan instruksi yang tidak ambigu. Penggunaan istilah 'tidak ambigu' menunjukkan bahwa tidak ada ruang untuk interpretasi subjektif. Setiap kali Anda meminta

komputer Anda untuk menjalankan algoritma yang sama, itu akan melakukannya dengan cara yang persis sama dengan hasil yang sama persis.

Pertimbangkan kembali contoh sebelumnya. Pengecekan ejaan menggunakan algoritma. Perhitungan keuangan menggunakan algoritma. Mesin pencari menggunakan algoritma. Nyatanya, sulit untuk memikirkan tugas yang dilakukan oleh komputer Anda yang tidak menggunakan algoritma.

Definisi lainnya perihal Algoritma adalah :

Algoritma dapat didefinisikan sebagai "urutan langkah-langkah yang harus dilakukan untuk keluaran yang dibutuhkan dari masukan tertentu". Ada tiga fitur utama algoritma dari definisinya:

Tujuan penting dari suatu Algoritma adalah mendapatkan keluaran tertentu,

Algoritma melibatkan beberapa langkah berkelanjutan,

Outputnya muncul setelah Algoritma menyelesaikan seluruh proses.

Jadi pada dasarnya, semua Algoritma bekerja secara logis sambil mengikuti langkah-langkah untuk mendapatkan keluaran untuk masukan yang diberikan.

Jenis Algoritma

Algoritma dapat diklasifikasikan menjadi 3 jenis berdasarkan strukturnya:

**Urutan:** jenis Algoritma ini ditandai dengan serangkaian langkah, dan setiap langkah akan dijalankan satu demi satu.

**Pencabangan:** jenis algoritma ini diwakili oleh masalah "jika-maka". Jika suatu kondisi benar, outputnya adalah A, jika kondisinya salah, outputnya adalah B. Jenis algoritma ini juga dikenal sebagai "tipe pilihan".

**Loop:** untuk tipe ini, proses dapat dilakukan berulang kali dalam kondisi tertentu. Ini diwakili oleh masalah "sementara" dan "untuk". Namun pastikan prosesnya akan berakhir setelah sejumlah loop dalam kondisi tersebut. Jenis algoritma ini juga dikenal sebagai "jenis pengulangan".

Kualitas algoritma yang bagus

Input dan output harus didefinisikan secara tepat.

Setiap langkah dalam Algoritma harus jelas dan tidak ambigu.

Algoritma harus paling efektif di antara banyak cara berbeda untuk memecahkan masalah.

Algoritma tidak boleh menyertakan kode komputer. Alih-alih, Algoritma harus ditulis sedemikian rupa sehingga dapat digunakan dalam bahasa pemrograman yang berbeda.

Algoritma menentukan serangkaian langkah yang melakukan komputasi atau tugas tertentu. Algoritma awalnya lahir sebagai bagian dari matematika - kata "algoritma" berasal dari penulis Arab Muḥammad ibn Mūsā al-Khwārizmī, - tetapi saat ini kata tersebut sangat terkait dengan ilmu komputer. Di sepanjang buku ini, kami akan memeriksa sejumlah Algoritma yang berbeda untuk melakukan berbagai tugas.

Algoritma menyerupai resep. Resep memberi tahu Anda cara menyelesaikan tugas dengan melakukan sejumlah langkah. Misalnya untuk memanggang kue langkah-langkahnya adalah: panaskan oven terlebih dahulu; campur tepung, gula, dan telur sampai merata; tuangkan ke dalam loyang; Dan seterusnya.

Namun, "Algoritma" adalah istilah teknis dengan arti yang lebih spesifik daripada "resep", dan menyebut sesuatu sebagai Algoritma berarti semua properti berikut benar:

Algoritma adalah deskripsi yang tidak ambigu yang menjelaskan apa yang harus diterapkan. Dalam resep, langkah seperti "Panggang sampai matang" bersifat ambigu karena tidak menjelaskan arti "selesai". Deskripsi yang lebih eksplisit seperti "Panggang sampai keju mulai menggelembung" lebih baik. Dalam Algoritma komputasi, langkah seperti "Pilih bilangan besar" tidak jelas: apa yang besar? 1 juta, 1 miliar, atau 100? Apakah nomor tersebut harus berbeda setiap kali, atau dapatkah nomor yang sama digunakan pada setiap kali?

Algoritma mengharapkan serangkaian input yang ditentukan. Misalnya, mungkin memerlukan dua angka di mana kedua angka lebih besar dari nol. Atau mungkin memerlukan satu kata, atau daftar nol atau lebih banyak angka.

Algoritma menghasilkan serangkaian keluaran yang ditentukan. Ini mungkin menampilkan yang lebih besar dari dua angka, versi kata semua huruf besar, atau versi daftar angka yang diurutkan.

Algoritma dijamin akan berhenti dan menghasilkan hasil, selalu berhenti setelah waktu tertentu. Jika Algoritma berpotensi berjalan selamanya, itu tidak akan terlalu berguna karena Anda mungkin tidak akan pernah mendapatkan jawaban.

Sebagian besar Algoritma dijamin menghasilkan hasil yang benar. Ini jarang berguna jika Algoritma mengembalikan angka terbesar 99% dari waktu, tetapi 1% dari waktu Algoritma gagal dan mengembalikan angka terkecil. Jika suatu algoritma memaksakan persyaratan pada inputnya (disebut prasyarat), persyaratan itu harus dipenuhi. Misalnya, prasyaratnya mungkin bahwa Algoritma hanya akan menerima bilangan positif sebagai input. Jika prasyarat tidak terpenuhi, Algoritma dibiarkan gagal dengan menghasilkan jawaban yang salah atau tidak pernah dihentikan.

Mempelajari Algoritma adalah bagian fundamental dari ilmu komputer. Ada beberapa karakteristik berbeda dari suatu algoritma yang berguna untuk diketahui: Apakah Algoritma benar-benar ada untuk melakukan tugas tertentu?

Jika seseorang mengusulkan Algoritma untuk menyelesaikan tugas, apakah kami yakin Algoritma tersebut berfungsi untuk semua masukan yang memungkinkan?

Berapa lama waktu yang dibutuhkan Algoritma untuk berjalan? Berapa banyak ruang memori yang dibutuhkan?

Setelah kami mengetahui bahwa masalah dapat diselesaikan dengan suatu Algoritma, pertanyaan yang wajar adalah apakah Algoritma tersebut adalah yang terbaik. Bisakah masalah diselesaikan lebih cepat?

Gerbang Logika / *Logic Gate* adalah dasar pembentuk Sistem Elektronika Digital yang berfungsi untuk mengubah satu atau beberapa Input (masukan) menjadi sebuah sinyal Output (Keluaran) Logis. Gerbang Logika beroperasi berdasarkan sistem bilangan biner yaitu bilangan yang hanya memiliki 2 kode simbol yakni **0** dan **1** dengan menggunakan Teori Aljabar Boolean. Aljabar Boolean Atau Dalam Bahasa Inggris Disebut Dengan Boolean Algebra Adalah Matematika Yang Digunakan Untuk Menganalisis Dan Menyederhanakan Gerbang Logika Pada Rangkaian-Rangkaian Digital Elektronika. Boolean Pada Dasarnya Merupakan Tipe Data Yang Hanya Terdiri Dari Dua Nilai Yaitu “True” Dan “False” Atau “Tinggi” Dan “Rendah” Yang Biasanya Dilambangkan Dengan Angka “1” Dan “0” Pada Gerbang Logika Ataupun Bahasa Pemrograman Komputer. Aljabar Boolean Ini Pertama Kali Diperkenalkan Oleh Seorang Matematikawan Yang Berasal Dari Inggris Pada Tahun 1854. Nama Boolean Sendiri Diambil Dari Nama Penemunya Yaitu George Boole.

Beberapa jenis Gerbang Logika Dasar yang membentuk sebuah Sistem Elektronika Digital, yaitu :

1. Gerbang AND, Gerbang AND memerlukan 2 atau lebih Masukan (Input) untuk menghasilkan hanya 1 Keluaran (Output). Gerbang AND akan menghasilkan Keluaran (Output) Logika 1 jika semua masukan (Input) bernilai Logika 1 dan akan menghasilkan Keluaran (Output) Logika 0 jika salah satu dari masukan (Input) bernilai Logika 0.
2. Gerbang OR, Gerbang OR memerlukan 2 atau lebih Masukan (Input) untuk menghasilkan hanya 1 Keluaran (Output). Gerbang OR akan menghasilkan Keluaran (Output) 1 jika salah satu dari Masukan (Input) bernilai Logika 1 dan jika ingin menghasilkan Keluaran (Output) Logika 0, maka semua Masukan (Input) harus bernilai Logika 0.
3. Gerbang NOT, Gerbang NOT hanya memerlukan sebuah Masukan (Input) untuk menghasilkan hanya 1 Keluaran (Output). Gerbang NOT disebut juga dengan Inverter (Pembalik)
4. Gerbang NAND, NAND adalah NOT AND atau BUKAN AND, Gerbang NAND merupakan kombinasi dari Gerbang AND dan Gerbang NOT yang menghasilkan kebalikan dari Keluaran (Output) Gerbang AND. Gerbang NAND akan menghasilkan Keluaran Logika 0 apabila semua Masukan (Input) pada Logika 1 dan jika terdapat sebuah Input yang bernilai Logika 0 maka akan menghasilkan Keluaran (Output) Logika 1.
5. Gerbang NOR, NOR adalah NOT OR atau BUKAN OR, Gerbang NOR merupakan kombinasi dari Gerbang OR dan Gerbang NOT yang menghasilkan kebalikan dari Keluaran (Output) Gerbang OR. Gerbang NOR akan menghasilkan Keluaran Logika 0 jika salah satu dari Masukan (Input) bernilai Logika 1 dan jika ingin mendapatkan Keluaran Logika 1, maka semua Masukan (Input) harus bernilai Logika 0.
6. Gerbang X-OR (Exclusive OR), singkatan dari Exclusive OR yang terdiri dari 2 Masukan (Input) dan 1 Keluaran (Output) Logika. Gerbang X-OR akan menghasilkan Keluaran (Output) Logika 1 jika semua Masukan-masukannya (Input) mempunyai nilai Logika yang berbeda. Jika nilai Logika Inputnya sama, maka akan memberikan hasil Keluaran Logika 0.









7. Gerbang X-NOR (Exclusive NOR), Gerbang X-NOR juga terdiri dari 2 Masukan (Input) dan 1 Keluaran (Output). X-NOR adalah singkatan dari Exclusive NOR dan merupakan kombinasi dari Gerbang X-OR dan Gerbang NOT. Gerbang X-NOR akan menghasilkan Keluaran (Output) Logika 1 jika semua Masukan atau Inputnya bernilai Logika yang sama dan akan menghasilkan Keluaran (Output) Logika 0 jika semua Masukan atau Inputnya bernilai Logika yang berbeda. Hal ini merupakan kebalikan dari Gerbang X-OR (Exclusive OR)


Input dan Output pada Gerbang Logika hanya memiliki 2 level. Kedua Level tersebut pada umumnya dapat dilambangkan dengan :

- High (Tinggi) - Low (Rendah)
- True (Benar) - False (Salah)
- On (Hidup) - Off (Mati)
- 1 - 0

Di simpulkan melalui tabel kebenaran Gerbang Logika yang berguna untuk proses pengkondisian .

<p>AND</p>  <table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th rowspan="2">OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	INPUT		OUTPUT	A	B	0	0	0	1	0	0	0	1	0	1	1	1	<p>OR</p>  <table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th rowspan="2">OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	INPUT		OUTPUT	A	B	0	0	0	1	0	1	0	1	1	1	1	1	<p>XOR</p>  <table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th rowspan="2">OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	INPUT		OUTPUT	A	B	0	0	0	1	0	1	0	1	1	1	1	0
INPUT		OUTPUT																																																			
A	B																																																				
0	0	0																																																			
1	0	0																																																			
0	1	0																																																			
1	1	1																																																			
INPUT		OUTPUT																																																			
A	B																																																				
0	0	0																																																			
1	0	1																																																			
0	1	1																																																			
1	1	1																																																			
INPUT		OUTPUT																																																			
A	B																																																				
0	0	0																																																			
1	0	1																																																			
0	1	1																																																			
1	1	0																																																			
<p>NAND</p>  <table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th rowspan="2">OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	INPUT		OUTPUT	A	B	0	0	1	1	0	1	0	1	1	1	1	0	<p>NOR</p>  <table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th rowspan="2">OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	INPUT		OUTPUT	A	B	0	0	1	1	0	0	0	1	0	1	1	0	<p>XNOR</p>  <table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th rowspan="2">OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	INPUT		OUTPUT	A	B	0	0	1	1	0	0	0	1	0	1	1	1
INPUT		OUTPUT																																																			
A	B																																																				
0	0	1																																																			
1	0	1																																																			
0	1	1																																																			
1	1	0																																																			
INPUT		OUTPUT																																																			
A	B																																																				
0	0	1																																																			
1	0	0																																																			
0	1	0																																																			
1	1	0																																																			
INPUT		OUTPUT																																																			
A	B																																																				
0	0	1																																																			
1	0	0																																																			
0	1	0																																																			
1	1	1																																																			

NOT	
	
INPUT	OUTPUT
A	
0	1
1	0



### **C. Latihan**

1. Sebutkan 3 jenis Algoritma berdasarkan strukturnya ?
2. Apa yang dimaksudn Gerbang Logika AND ?

### **D. Kunci Jawaban**

1. Jawaban : Urutan, Pencabangan, Pengulangan.
2. Jawaban : Gerbang AND akan berlogika 1 apabila semua inputnya berlogika 1, namun bila salah satu atau semua keluarannya berlogika 0 maka keluarannya berlogika 0.

### **E. Daftar Pustaka**

1. Algorithms, 4th Edition; Author(s) Robert Sedgewick and Kevin Wayne, Publisher: Addison-Wesley Professional; 2011
2. Introduction to Algorithms 3rd Ed, hal 1002-1012. London. The MIT. Press, 2009
3. <https://idcloudhost.com/mengenal-apa-itu-algoritma-definisi-ciri-ciri-dan-contohnya/>
4. <https://teknikelektronika.com/pengertian-gerbang-logika-dasar-simbol/>

Universitas  
**Esa Unggul**