BAB II

LANDASAN TEORI

2.1. Konsep Dasar Algoritma

2.1.1. Sejarah Algoritma

Ditinjau dari asal-usul katanya, kata Algoritma mempunyai sejarah yang unik. Orang hanya menemukan kata *algorism* yang berarti proses menghitung dengan angka arab. Seseorang dikatakan *algorist* jika orang tersebut menghitung menggunakan angka arab. Para ahli bahasa berusaha menemukan asal kata ini, namun hasilnya kurang memuaskan.

Akhirnya para ahli sejarah matematika menemukan asal kata tersebut yang berasal dari nama penulis buku arab yang terkenal yaitu Abu Ja'far Muhammad Ibnu Musa Al-Khuwarizmi. Al-Khuwarizmi dibaca orang barat menjadi Algorism. Al-Khuwarizmi menulis buku yang berjudul Kitab Al Jabar WalMuqabala yang artinya "Buku pemugaran dan pengurangan" (*The book of restoration and reduction*). Dari judul buku tersebut diperoleh akar kata "*Aljabar*" (*Algebra*).

Perubahan kata dari *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, maka lambat laun kata *algorithm* berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna kata aslinya. Dalam bahasa Indonesia, kata *algorithm* diserap menjadi algoritma.

2.1.2. Definisi Algoritma

Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis dan logis. Kata logis merupakan kata kunci dalam algoritma. Langkah-langkah dalam algoritma harus logis dan harus dapat ditentukan bernilai salah atau benar. Dalam beberapa konteks, algoritma adalah spesifikasi urutan langkah untuk melakukan pekerjaan tertentu.

Pertimbangan dalam pemilihan algoritma adalah algoritma haruslah benar. Artinya algoritma akan memberikan keluaran yang dikehendaki dari sejumlah masukan yang diberikan. Tidak peduli serumit apapun algoritma, jika memberikan keluaran yang salah, pastilah algoritma tersebut bukanlah algoritma yang baik.

Pertimbangan lain yang harus diperhatikan adalah seberapa baik hasil yang dicapai oleh algoritma tersebut. Hal ini penting terutama pada algoritma untuk menyelesaikan masalah yang memerlukan aproksimasi hasil. Algoritma yang baik harus mampu memberikan hasil yang sedekat mungkin dengan nilai yang sebenarnya.

Selanjutnya adalah efisiensi algoritma. Efisiensi algoritma dapat ditinjau dari 2 hal yaitu efisiensi waktu dan memori. Meskipun algoritma memberikan keluaran yang benar, tetapi jika harus menunggu lama untuk mendapatkan keluarannya, algoritma tersebut bukanlah algoritma yang baik, setiap orang menginginkan keluaran yang cepat.

Dalam kenyataannya, setiap orang bisa membuat algoritma yang berbeda untuk menyelesaikan suatu permasalahan, walaupun terjadi perbedaan dalam menyusun algoritma, tentunya diharapkan keluaran yang sama.

2.2. Pengantar Pascal

2.2.1. Sejarah Pascal

Pascal merupakan pengembangan dari bahasa Algol 60, bahasa pemrograman untuk sains komputasi. Tahun 1960, beberapa ahli komputer bekerja untuk mengembangkan bahasa Algol, salah satunya adalah Dr. Niklaus Wirth dari Swiss Federal Institute of Technology (ETH-Zurich), yang merupakan anggota grup yang membuat Algol.

Tahun 1971, diterbitkan suatu spesifikasi untuk *highly-structured language* (bahasa tinggi yang terstruktur) yang menyerupai Algol yaitu Pascal. Pascal bersifat data oriented, yaitu programmer diberi keleluasaan untuk mendefinisikan data sendiri. Pascal juga merupakan *teaching language* (banyak dipakai untuk pengajaran tentang konsep pemrograman). Kelebihan yang lain adalah penulisan kode Pascal yang luwes.

2.2.2. Ciri – Ciri Pascal

Susunan dari kode-kode dalam teks Pascal harus ditulis secara terurut, pernyataan-pernyataan yang ditulis lebih awal akan dieksekusi lebih dahulu. Oleh karena itu, suatu pernyataan yang menyangkut suatu variabel di dalam program,

maka variable itu harus terdefinisi dahulu sebelumnya. Hal ini terutama menyangkut pada pemanggilan sub-program oleh sub-program yang lain.

Blok dengan batas-batas yang jelas.Pascal memberikan pembatas yang jelas pada tiap-tiap blok, seperti pada blok program utama, sub-program, struktur kontrol (pengulangan/ pemilihan), dll. Pemakaian kata kunci “begin” untuk mengawali operasi pada blok dan “end” untuk menutupnya memudahkan programmer menyusun programnya dengan mudah.

2.3. Pengantar C

2.3.1. Sejarah C

Berasal dari bahasa BCPL (*Basic Combined Programming Language*) oleh Martin Richard, Cambridge tahun 1967, Ken Thompson membuat bahasa B untuk dipakai pada komputer DEC PDP-7 dibawah sistem operasi UNIX pada Bell laboratory, Murray Hill, New Jersey tahun 1970.

Bahasa B merupakan suatu bahasa pemrograman yang tidak memiliki jenis suatu data seperti halnya PL/M. Berdasarkan gambaran bahasa B, Dennis Ritchie menulis bahasa C. Nama C diambil berdasarkan urutan sesudah B dari bahasa BCPL. Tujuan bahasa C pada mulanya untuk membentuk suatu sistem operasi yang akan digunakan pada mesin komputer DEC PDP-11 yang baru.

Pada tahun 1975, sistem operasi UNIX versi 6 dan bahasa C mulai diberikan kepada Universitas maupun Akademi. Dan pada tahun 1979, sistem operasi UNIX versi 7 dikeluarkan dengan bahasa C. Sistem operasi ini (versi 7) seluruhnya ditulis dalam bahasa C.

Pada 1978 Dennis Ritchie dan Brian Kernighan kemudian mempublikasikan buku *The C Programming Language* yang semakin memperluas pemakaiannya dan dijadikan standar oleh ANSI *(American National Standard Institute)* pada tahun 1989. C kemudian dikembangkan lagi oleh Bjarne Stroustrup menjadi C++ (1986). C dan/atau C++ banyak digunakan sebagai bahasa pemrograman untuk membuat sistem operasi.

2.3.2. Ciri-ciri C

Program C pada hakekatnya tersusun atas sejumlah blok fungsi. Sebuah program minimal mengandung sebuah fungsi. Fungsi pertama yang harus ada dalam program C dan sudah ditentukan namanya adalah main(). Setiap fungsi terdiri atas satu atau beberapa pernyataan, yang secara keseluruhan dimaksudkan untuk melaksanakan tugas khusus. Bagian pernyataan fungsi (sering disebut tubuh fungsi) diawali dengan tanda kurung kurawal buka ({) dan diakhiri dengan tanda kurung kurawal tutup (}). Di antara kurung kurawal itu dapat dituliskan statemen-statemen program C.

Namun pada kenyataannya, suatu fungsi bisa saja tidak mengandung pernyataan sama sekali. Walaupun fungsi tidak memiliki pernyataan, kurung kurawal haruslah tetap ada. Sebab kurung kurawal mengisyaratkan awal dan akhir definisi fungsi. Bahasa C dikatakan sebagai bahasa pemrograman terstruktur karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagiannya (subroutine). Fungsi-fungsi yang ada selain fungsi utama (main()) merupakan program-program bagian. Fungsi-fungsi ini dapat ditulis setelah fungsi

utama atau diletakkan di file pustaka (library). Jika fungsi-fungsi diletakkan di file pustaka dan akan dipakai di suatu program, maka nama file judulnya (header file) harus dilibatkan dalam program yang menggunakannya dengan preprocessor directive berupa #include.

2.4. Konsep Dasar String Matching

2.4.1. String Matching

String matching adalah pencarian sebuah *pattern* pada sebuah teks. Prinsip kerja algoritma string matching adalah sebagai berikut:

1. Memindai teks dengan bantuan sebuah window yang ukurannya sama dengan panjang *pattern*.
2. Menempatkan window pada awal teks.
3. Membandingkan karakter pada window dengan karakter dari *pattern*. Setelah pencocokan (baik hasilnya cocok atau tidak cocok), dilakukan shft ke kanan pada window. Prosedur ini dilakukan berulang-ulang sampai window berada pada akhir teks. Mekanisme ini disebut mekanisme sliding-window.

Algoritma string matching mempunyai tiga komponen utama, yaitu:

1. *Pattern*, yaitu deretan karakter yang akan dicocokkan dengan teks, dinyatakan dengan x[0..m-1], panjang *pattern* dinyatakan dengan m.
2. Teks, yaitu tempat pencocokan *pattern* dilakukan, dinyatakan dengan y[0..n-1], panjang teks dinyatakan dengan n.

3. Alfabet, yang berisi semua simbol yang digunakan oleh bahasa pada teks dan *pattern*, dinyatakan dengan $\sum$ dengan ukuran dinyatakan dengan ASIZE.

2.4.2. Algoritma Boyer-Moore

Algoritma Boyer-Moore termasuk algoritma string matching yang paling efisien dibandingkan algoritma-algoritma string matching lainnya. Karena sifatnya yang efisien, banyak dikembangkan algoritma string matching dengan bertumpu pada konsep algoritma Boyer-Moore, beberapa di antaranya adalah algoritma Turbo BM dan algoritma Quick Search.

Algoritma Boyer-Moore menggunakan metode pencocokan string dari kanan ke kiri yaitu memindai karakter *pattern* dari kanan ke kiri dimulai dari karakter paling kanan. Algoritma Boyer-Moore menggunakan dua fungsi shift yaitu good-suffix shift dan bad-character shift untuk mengambil langkah berikutnya setelah terjadi ketidakcocokan antara karakter *pattern* dan karakter teks yang dicocokkan.

2.5. Konsep Dasar Stack

Stack adalah suatu bentuk khusus dari linear list (suatu struktur data yang merupakan himpunan terurut yang dapat berkurang atau bertambah setiap saat) di mana operasi penyisipan dan penghapusan atas elemen-elemennya hanya dapat dilakukan pada satu sisi saja yang disebut sebagai “TOP”. Ada empat operasi dasar yang didefinisikan pada stack, yaitu:

1. Create, operator ini berfungsi untuk membuat sebuah stack kosong.
2. IsEmpty, operator ini berfungsi untuk menentukan apakah suatu stack adalah stack kosong.
3. Push, operator ini berfungsi untuk menambahkan satu elemen ke dalam stack.
4. Pop, operator ini berfungsi untuk mengeluarkan satu elemen dari dalam stack.

Dalam penggunaannya, untuk menempatkan stack biasanya digunakan sebuah array. Tetapi perlu diingat di sini bahwa stack dan array adalah dua hal yang berbeda. Selain itu penggunaan stack dengan array dirasakan kurang tepat. Penggunaan stack pada pembangunan Translator ini menggunakan record sebagai implementasi stack.

2.6. Konsep Dasar Pohon Biner

Tree/pohon merupakan struktur data yang tidak linear/non linear yang digunakan terutama untuk merepresentasikan hubungan data yang bersifat hierarkis antara elemen-elemennya. Kumpulan elemen yang salah satu elemennya disebut dengan *root* (akar) dan sisa elemen yang lain disebut sebagai simpul (*node/vertex*) yang terpecah menjadi sejumlah himpunan yang tidak saling berhubungan satu sama lain, yang disebut *subtree*/cabang.

Sebuah pohon biner T dapat didefinisikan sebagai sekumpulan terbatas dari elemen-elemen yang disebut nodes/simpul dimana :

a. T dikatakan kosong (disebut null *tree*/pohon null atau *empty tree*/pohon kosong)
b. T terdiri dari sebuah node khusus yang dipanggil R, disebut root dari T dan node-node T lainnya membentuk sebuah pasangan terurut dari *binary tree* T1 dan T2 yang tidak berhubungan yang kemudian dipanggil *subtree* kiri dan *subtree* kanan.

Jika T1 tidak kosong maka rootnya disebut *successor* kiri dari R dan jika T2 tidak kosong, maka rootnya disebut *successor* dari R.

Jenis-jenis *binary tree* :

1. *Complete Binary Tree*

   Suatu *binary tree* T akan disebut *complete*/lengkap jika semua levelnya memiliki *child* 2 buah kecuali untuk level paling akhir. Tetapi pada akhir level setiap *leaf*/daun muncul terurut dari sebelah kiri.

2. *Extended Binary Tree : 2-Tree*

   Sebuah *binary tree* dikatakan 2-*tree* atau *extended binary tree* jika tiap simpul N memiliki 0 atau 2 anak. Simpul dengan 2 anak disebut dengan simpul internal (internal node), dan simpul dengan 0 anak disebut dengan external node. Kadang-kadang dalam diagram node-node tersebut dibedakan dengan menggunakan tanda lingkaran untuk internal node dan kotak untuk eksternal node.

2.7. Konsep Dasar Flowchart

Flowchart adalah representasi grafis peta atau dari suatu proses. Langkah-langkah dalam suatu proses yang ditampilkan dengan bentuk simbolik, dan aliran proses ini ditunjukkan dengan panah menghubungkan simbol. Pemrograman komputer flowchart dipopulerkan pada 1960-an yang digunakan untuk memetakan logika program. Dalam pekerjaan peningkatan kualitas, flowchart sangat berguna untuk menampilkan bagaimana proses saat ini berfungsi atau bisa idealnya fungsi.

Flowchart dapat membantu melihat apakah langkah-langkah proses telah logis, mengungkap masalah atau miskomunikasi, menentukan batas-batas proses, dan mengembangkan dasar umum pengetahuan tentang proses. Flowcharting proses seringkali membawa kepada redudansi cahaya, penundaan, buntu, dan jalur tidak langsung yang dinyatakan akan tetap diperhatikan atau diabaikan. Tapi flowchart tidak akan bekerja jika alur tidak akurat.