BAB II

LANDASAN TEORI

2.1 Konsep Dasar Algoritma

2.1.1 Sejarah Algoritma

Dilihat dari asal usul kata, kata algoritma sendiri mempunyai sejarah yang aneh. Kata ini tidak muncul dalam kamus *Webster* sampai akhir tahun 1957. Orang hanya menemukan kata *algorism* yang berarti proses menghitung dengan angka Arab [KNU73]. Seseorang dikatakan *algorist* jika menggunakan angka Arab. Para ahli berusaha menemukan asal kata *algorism* ini namun hasilnya kurang memuaskan. Akhirnya para ahli sejarah matematika menemukan asal mula kata tersebut. Kata *algorism* 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 wal-muqabala*, yang artinya "Buku pemugaran dan pengurangan" (*The book of restoration and reduction*). Dari judul buku itu diperoleh juga akar kata "aljabar" (*algebra*). Perubahan dari kata *algorism* menjadi *alghorithm* 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 aslinya. Dalam bahasa Indonesia, kata *alghorism* diserap menjadi algoritma.

2.1.2 Definisi Algoritma

Menurut Rinaldi Munir (2005 : 176) *"Algoritma adalah urutan logis langkah-langkah penyeleseian masalah yang disusun secara sistematis"*. Alur pemikiran dalam menyelesaikan suatu pekerjaan yang dituangkan secara tertulis. Yang ditekankan pertama adalah alur pikiran, sehingga algoritma seseorang dapat juga berbeda dari algoritma orang lain. Sedangkan penekanan kedua adalah tertulis, yang artinya dapat berupa kalimat, gambar, atau tabel tertentu.

Algoritma dapat dituliskan dalam berbagai notasi, misalnya dalam notasi kalimat-kalimat deskriptif. Dengan notasi kalimat deskriptif, deskripsi setiap langkah dijelaskan dengan bahasa sehari-hari secara jelas. Setiap langkah biasanya diawali dengan kata kerja seperti 'baca', 'hitung', 'masukan', 'bagi', 'ganti', dan sebagainya. Sedangkan pernyataan bersyarat dinyaakan dengan 'jika','maka', dan sebagainya.

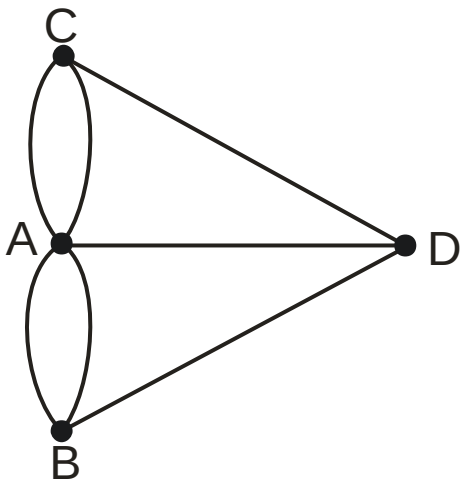
2.2 Konsep Dasar Graf

2.2.1. Sejarah Graf

Menurut catatan sejarah, masalah jembatan *Konigsberg* adalah masalah yang pertama kali menggunakan graf (tahun 1736). Di kota *konigsberg* (sebelah timur negara bagian Prussia, Jerman), sekarang bernama kota *Kaliningrad*, terdapat sungai Pregal yang mengalir mengintari pulau *Kneiphof* lalu bercabang menjadi dua buah anak sungai.

Ada tujuh buah jembatan yang menghubungkan daratan yang dibelah oleh sungai tersebut. Masalah jembatan *Konigsberg* adalah : apakah mungkin melalui ketujuh buah jembatan itu masing-masing tepat satu kali, dan kembali lagi ke

tempat semula. Sebagian penduduk kota tersebut sepakat bahwa memang tidak mungkin melalui setiap jembatan itu hanya sekali dan kembali lagi ke tempat asal mula keberangkatan, tetapi mereka tidak dapat menjelaskan mengapa demikian jawabannya, kecuali dengan cara coba-coba. Tahun 1736, seorang matematikawan Swiss, L.Euler, adalah orang pertama yang berhasil menemukan jawaban masalah itu dengan pembuktian yang sederhana. Ia memodelkan masalah ini kedalam graf. Daratan (titik-titik yang dihubungkan oleh jembatan) dinyatakannya sebagai titik (noktah) yang disebut simpul (vertex) dan jembatan dinyatakan sebagai garis yang disebut sisi (edge). Setiap titik diberi label huruf A, B, C, dan D. Graf yang dibuat oleh Euler diperlihatkan pada gambar berikut :



Gambar 2.1 Graf Yang Merepresentasikan Jembatan Konigsberg

(Sumber : Rinaldi Munir.2009)

Jawaban yang dikemukakan oleh Euler adalah : orang tidak mungkin melalui ketujuh jembatan itu masing-masing satu kali dan kembali lagi ke tempat asal keberangkatan jika derajat setiap simpul tidak seluruhnya genap. Yang

dimaksud dengan derajat adalah banyaknya garis yang bersisian dengan noktah. Sebagai contoh, simpul C memiliki 3 derajat karena ada tiga buah garis yang bersisian dengannya, simpul B dan D juga berderajat dua, sedangkan simpul A berderajat 5. Karena tidak semua simpul berderajat genap, maka tidak mungkin dilakukan perjalanan berupa sirkuit (yang dinamakan sirkuit Euler) pada graf tersebut.

2.2.2. Definisi Graf

Menurut Indra Yatini. B. dan Erliansyah Nasution, 2005 : 379 Graf penting sebagai model untuk berbagai jenis proses atau struktur. Kota-kota dan jalan-jalan yang menghubungkan membentuk graf, seperti juga komponen pada papan sirkuit yang berhubungan di antara mereka.

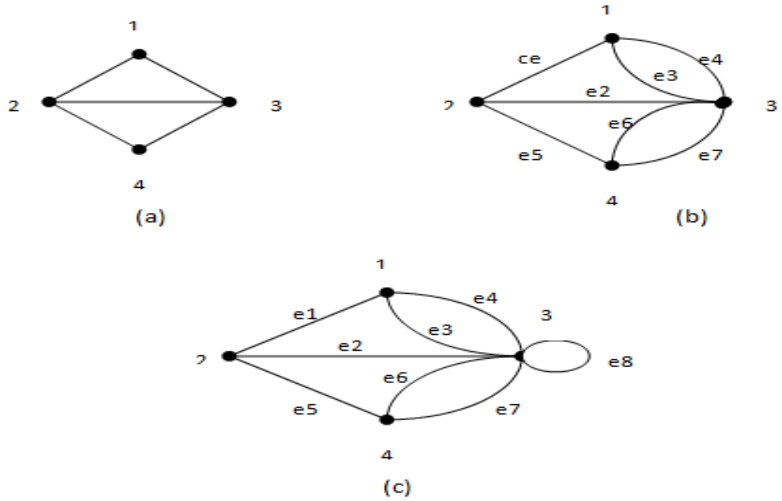
"Graf secara umum bisa didefinisikan sebagai kumpulan titik (nodes atau vertices) dan garis (arcs atau edges)" (P.Insap Santosa, 2004 : 497). Karena garis selalu diawali dari suatu titik dan diakhiri pada titik yang lain, maka garis bisa dituliskan sebagai pasangan antara dua titik.

Graf *G* didefinisikan sebagai pasangan himpunan *(V,E)*, yang dalam hal ini : *V* = himpunan berhingga dan tidak kosong dari simpul-simpul (*vertices* atau *node*) = { $v_1, v_2, ..., v_n$ dan *E* = himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul = { $e_1, e_2, ..., e_n$ } atau dapat ditulis singkat notasi *G = (V, E .)*. Simpul pada graf dapat dinomori dengan huruf, seperti *v, w,…,*dengan bilangan asli 1, 2, 3,…, atau gabungan keduanya. Sedangkan sisi yang menghubungkan simpul v_i dengan simpul v_j dinyatakan dengan pasangan

(v_i, v_j) atau dengan lambang $e_1, e_2, ...$ Dengan kata lain, jika e adalah sisi yang menghubungkan simpul v_i dengan simpul v_j, maka e dapat ditulis sebagai :

$$e = (v_i, v_j)$$

Berikut adalah contoh gambar graf :



Gambar 2.2 Graf sederhana (a), Graf Ganda (b), Graf semu (c)

(Sumber : Rinaldi Munir.2009)

2.2.3. Jenis Graf

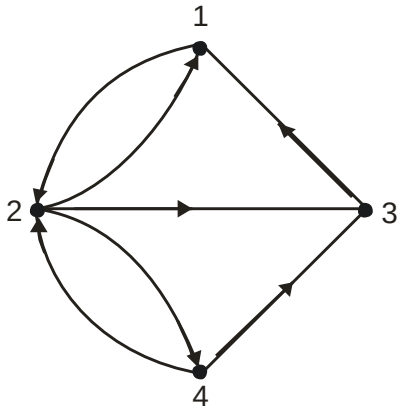
Sisi pada graf dapat mempunyai orientasi arah, berdasarkan orientasi arah pada sisi, maka secara umum dapat dibedakan menjadi dua jenis, yaitu :

2.2.3.1 Graf Tak-berarah

Graf tak berarah adalah graf yang sisinya tidak mempunyai orientasi arah. Pada graf tak-berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. Jadi, (u, v) = (v, u) adalah sisi yang sama. Contoh gambar graf tak berarah bisa dilihat pada gambar 2.1 di atas.

2.2.3.2 Graf Berarah

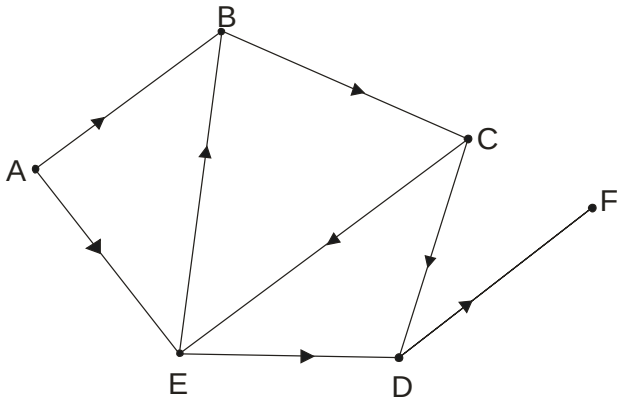
Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah. Kita lebih suka menyebut sisi berarah dengan sebutan **busur** *(arc)*. Pada graf berarah, (u, v) dan (v, u) menyatakan dua buah busur yang berbeda, dengan kata lain (u, v) ≠ (v, u). untuk busur (u, v), simpul u dinamakan **simpul asal,** dan simpul v dinamakan **simpul terminal.**



Gambar 2.3 Graf Berarah

(Sumber : Rinaldi Munir.2009)

Menurut Insap Santoso 2004 : 499 dalam bukunya yang berjudul Struktur Data Menggunakan Turbo Pascal 6.0 Menjelaskan bahwa Graph berarah (*directed graph*, atau *digraph*) adalah merupakan bentuk yang lebih khusus dari graph seperti yg dijelaskan di atas, karena kedalam graph berarah terkandung suatu aliran (*flow*), misalnya aliran beban, dari satu titik ketitik lain; dalam gambar biasanya disajikan menggunakan anak panah. Dengan demikian, pasangan titik yang menyatakan suatu garis harus disusun secara berurutan (*ordered pair*). Hal ini bisa dipahami dengan membayangkan suatu jalan yang lalulintasnya hanya bisa dalam satu arah (*one-way-traffic*). Dalam pasangan berurutan ini titik pertama menunjukan titik asal aliran (*source*), dan titik kedua menunjukan titik tujuan (*sink*).



Gambar 2.4 Implementasi Graph Berarah

(Sumber : P.Insap Santosa. 2004)

Dengan kenyataan diatas maka gambar 2.3 harus dituliskan sebagai :

N = {A,B,C,D,E,F}

E = {[A,B],[A,E],[B,C],[E,B],[C,E],[C,D],[E,D],[D,F]}

Selain sebutan titik asal dan titik tujuan, titik pertasa juga sering disebut dengan predesesor dari titik kedua, dan titik kedua adalah suksesor dari titik pertama. Dengan demikian titik A adalah predesesor dari titik B dan E, dan titik B adalah suksesor dari titik A.

2.3 Definisi Lintasan

Lintasan adalah hubungan antara titik dalam sebuah graf. Lintasan yang berawal dan berakhir pada simpul yang sama disebut lintasan tertutup (*closed path*), sedangkan lintasan yang tidak berawal dan berakhir pada simpul yang sama disebut lintasan terbuka (*open path*).

2.3.1 Lintasan Terpendek *(Shortest Path)*

Persoalan mencari lintasan terpendek di dalam graf merupakan salah satu persoalan optimasi. Graf yang digunakan dalam pencarian lintasan terpendek adalah graf berbobot (*weighted graph*), yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot. Bobot pada sisi graf dapat menyatakan jarak antar kota, waktu pengiriman pesan, ongkos pembangunan, dan sebagainya. Asumsi yang digunakan disini adalah bahwa semua bobot bernilai positif. Kata "terpendek" jangan selalu diartikan secara fisik sebagai panjang minimum, sebab kata "terpendek" berbeda-beda maknanya tergantung pada tipikal persoalan yang akan diseleseikan. Namun secara umum "terpendek" berarti meminimisasi bobot pada suatu lintasan di dalam graf.

Ada beberapa macam persoalan lintasan terpendek, antara lain :

a. Lintasan terpendek antara dua buah simpul tertentu.
b. Lintasan terpendek antara semua pasangan simpul.
c. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain.
d. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu.

2.4 Algoritma *Floyd Warshall*

Algoritma Floyd-Warshall ditemukan oleh Stephen Warshall dan Robert W. Floyd. Stephen Warshall lahir di New York pada tahun 1935 dan meninggal pada tanggal 11 desember 2006. Robert W. Floyd, lahir di New York pada tanggal 8 juni 1936 dan meninggal pada tangggal 25 September 2001.

Dalam Wikipedia dikutip bahwa Algoritma Floyd-Warshall memiliki input graf berarah dan berbobot (*V*,*E*), yang berupa daftar titik (node/vertex *V*) dan daftar sisi (edge *E*). Jumlah bobot sisi-sisi pada sebuah jalur adalah bobot jalur tersebut. Sisi pada *E* diperbolehkan memiliki bobot negatif, akan tetapi tidak diperbolehkan bagi graf ini untuk memiliki siklus dengan bobot negatif. Algoritma ini menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah pasangan titik, dan melakukannya sekaligus untuk semua pasangan titik. Algoritma ini berjalan dengan waktu $\Theta(|V|^3)$.

Dalam blog zafnatpaneyah.blogspot.com dikutip bahwa Pengertian Umum Algoritma Floyd Warshall: Merupakan salah satu varian dari pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu.

Adapun pembahasan yang lebih detil yang di kutip oleh Raja Untung pada web *id.shvoong.com* yaitu Jarak terpendek merupakan bagian dari teori graph. Jika diberikan sebuahgraph berbobot, masalah jarak terpendek adalah bagaimana kita mencari sebuah jalur pada graph yang meminimalkan jumlah bobot sisi pembentuk jalur tersebut. Salah satu algoritma untuk mengatasi hal ini adalah dengan menggunakanalgoritma Floyd- Warshall. Algoritma Floyd-Warshall memiliki input graph berarah dan berbobot (V,E),yang berupa daftar titik (node / vertex V) dan daftar sisi (edge E). Jumlah bobot sisi-sisi pada sebuah jalur adalah bobot jalur tersebut. Sisi pada E diperbolehkan memiliki bobot negatif, akan tetapi

tidak diperbolehkan bagi graph ini untuk memiliki siklus dengan bobot negatif. Algoritma ini menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah pasangan titik, dan melakukannya sekaligus untuk semua pasangan titik. Algoritma ini berjalan dengan waktuO(|V|3).Jadi pada intinya Algoritma Floyd Warshall membandingkan semua kemungkinan lintasan pada graph untuk setiap sisi dari semua simpul. Menariknya, algoritma ini mampu mengerjakan proses perbandingkan ini sebanyak V3 kali (bandingkan dengan kemungkinan jumlah sisi sebanyak V2(kuadrat jumlah simpul) pada graph, dan setiap kombinasi sisi diujikan).

Haltersebut bisa terjadi karena adanya perkiraan pengambilkan keputusan (pemilihanjalur terpendek) pada setiap tahap antara dua simpul, hingga perkiraan tersebut diketahui sebagai nilai optimal.

2.5 Algoritma Greedy

Algoritma *Greedy* adalah algoritma yang memecahkan masalah langkah demi langkah dan merupakan salah satu metode dalam masalah optimasi. Prinsip dari algoritma greedy adalah "*take what you can get now"* yaitu mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan

konsekuensi kedepan. Algoritma *greedy* membentuk solusi langkah per langkah sebagai berikut:

1. Terdapat banyak pilihan yang perlu diekspolarasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilhan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya.
2. Pendekatan yang digunakan di dalam algoritma greedy adalah membuat pilihan yang terlihat memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal pada setiap langkah dan diharapkan akan mendapatkan solusi optimum global.

Algoritma greedy didasarkan pada pemindahan *edge* per *edge* dan pada setiap langkah yang diambil tidak memikirkan konsekuensi ke depan, *greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada serta sebagian masalah *greedy* tidak selalu berhasil memberikan solusi yang benar-benar oprimum tapi pasti memberikan solusi yang mendekati nilai optimum.

Algoritma greedy disusun oleh elemen-elemen sebagai berikut :

1. Himpunan Kandidat

   Himpunan ini berisi elemen-elemen yang memiliki peluang pembentuk solusi.

2. Himpunan Solusi

   Himpunan ini berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Elemennya terdiri dari elemen dalam himpunan kandidat,

namun tidak semuanya dengan kata lain himpunan solusi ini adalah bagian dari himpunan kandidat.

3. Fungsi seleksi

   Fungsi yang pada setiap langkah memilih kandidat yang paling mungkin untuk menghasilkan solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan

   Fungsi yang memeriksa apakah suatu kandidat yang telah dipilih (diseleksi) dapat memberikan solusi yang layak.

5. Fungsi obyektif

   Fungsi yang memaksimumkan atau meminimumkan nilai solusi. Tujuannya adalah memilih satu saja solusi terbaik dari masing-masing anggota himpunan solusi.

2.6 Definisi Taksi

Taksi adalah angkutan umum yang menggunakan mobil untuk mengangkut penumpangnya. Taksi umumnya menggunakan mobil jenis sedan, namun di beberapa negara ada pula taksi jenis van yang dapat mengangkut lebih banyak penumpang atau muatan. Istilah "taksi" juga dapat merujuk kepada angkutan umum lain selain mobil yang mengangkut penumpang dalam kapasitas kecil, misalnya "taksi air", yang sebenarnya mungkin hanya berupa sampan (Wikipedia).

Tarif layanan jasa angkutan taksi dihitung melalui dua cara:

a. menggunakan argometer - dihitung secara otomatis tergantung jumlah jarak yang ditempuh.
b. berdasarkan kesepakatan - penumpang dan pengemudi menyepakati tarif sebelum (kadang bisa juga sesudah) perjalanan.

Perbedaan utama antara taksi dengan angkutan umum darat yang modern lainnya seperti bus terletak pada jumlah penumpangnya. Menggunakan mobil, taksi hanya dapat memuat sekitar 4 penumpang di dalamnya, dan penumpangnya tersebut biasanya berada dalam satu kelompok.

2.7 Definisi Jarak

Menurut Wikipedia Definisi jarak adalah angka yang menunjukkan seberapa jauh suatu benda berubah posisi melalui suatu lintasan tertentu. Dalam fisika atau dalam pengertian sehari-hari, jarak dapat berupa estimasi jarak fisik dari dua buah posisi berdasarkan kriteria tertentu (misalnya jarak tempuh antara Jakarta-Bandung). Dalam bidang matematika, jarak haruslah memenuhi kriteria tertentu.

2.8 Definisi Node (Simpul)

Menurut ajiexku dalam blognya http://ajiexku.wordpress.com Node atau titik simpul adalah titik pertemuan dari dua atau lebih elemen rangkaian. Junction atau titik simpul utama atau titik percabangan adalah titik pertemuan dari tiga atau lebih elemen rangkaian.

2.9 Flowchart

Menurut Wikipedia Flowchart atau diagram alir merupakan sebuah diagram dengan simbol-simbol grafis yang menyatakan aliran algoritma atau proses yang menampilkan langkah-langkah yang disimbolkan dalam bentuk kotak, beserta urutannya dengan menghubungkan masing masing langkah tersebut menggunakan tanda panah. Diagram ini bisa memberi solusi selangkah demi selangkah untuk penyelesaian masalah yang ada di dalam proses atau algoritma tersebut.

2.9.1. Pedoman Membuat Flowchart

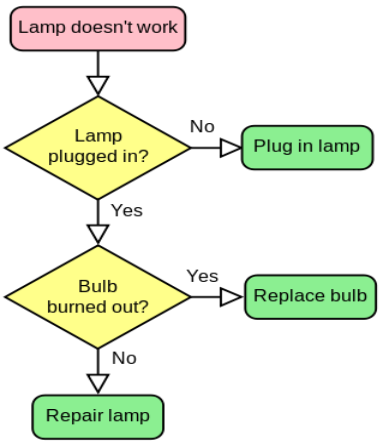
Bila seorang analis dan programmer akan membuat flowchart, ada beberapa petunjuk yang harus diperhatikan, seperti:

1. Flowchart digambarkan dari halaman atas ke bawah dan dari kiri kekanan.
2. Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini harus dapat dimengerti oleh pembacanya.
3. Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas.
4. Setiap langkah dari aktivitas harus diuraikan dengan menggunakan deskripsi kata kerja
5. Setiap langkah dari aktivitas harus berada pada urutan yang benar.
6. Lingkup dan range dari aktifitas yang sedang digambarkan harusditelusuri dengan hati-hati. Percabangan-percabangan yang memotong aktivitas yang sedang digambarkan tidak perlu digambarkan pada flowchart yang sama. Simbol konektor harus digunakan dan percabangannya diletakan pada

halaman yang terpisah atau hilangkan seluruhnya bila percabangannya tidak berkaitan dengan sistem.

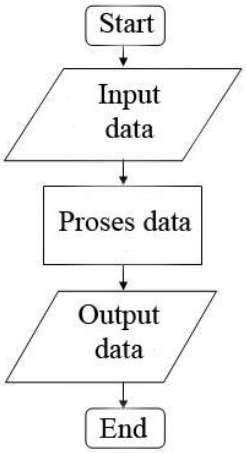
7. Gunakan simbol-simbol flowchart yang standar.

Berikut gambar dari contoh flowchart :



Gambar 2.5. Contoh diagram alir proses yang berhubungan dengan kerusakan lampu

(Sumber : http://id.wikipedia.org/wiki/Diagram_alir)



Gambar 2.6. Contoh diagram alir proses

(Sumber : http://id.wikipedia.org/wiki/Diagram_alir)

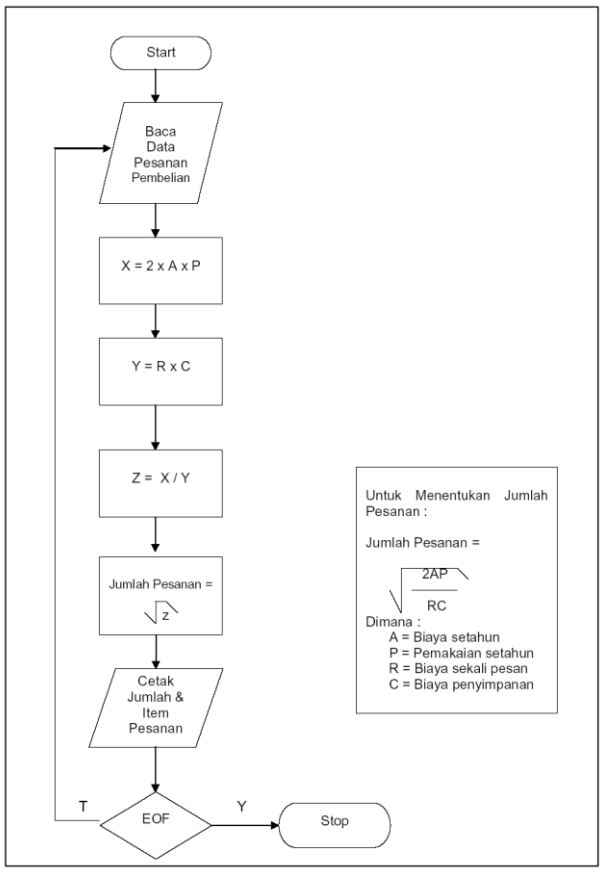
2.9.2. Jenis Flowchart

Flowchart terbagi atas lima jenis, yaitu :

- Flowchart Sistem (*System Flowchart*)
- Flowchart Paperwork / Flowchart Dokumen (*Document Flowchart*)
- Flowchart Skematik (*Schematic Flowchart*)
- Flowchart Program (*Program Flowchart*)
- Flowchart Proses (*Process Flowchart*)

Dalam hal ini penulis menggunakan Flowchart Program (*Program Flowchart*) untuk mengetahui alur simulasi jalur terpendek. Flowchart Program dihasilkan dari Flowchart Sistem. Flowchart Program merupakan keterangan yang lebih rinci tentang bagaimana setiap langkah program atau prosedur sesungguhnya dilaksanakan. Flowchart ini menunjukkan setiap langkah program atau prosedur

dalam urutan yang tepat saat terjadi. Programmer menggunakan flowchart program untuk menggambarkan urutan instruksi dari program komputer. Analis Sistem menggunakan flowchart program untuk menggambarkan urutan tugas-tugas pekerjaan dalam suatu prosedur atau operasi.



Gambar 2.7 Contoh *Flowchart* Program.

(Sumber : http://id.wikipedia.org/wiki/Diagram_alir)

2.10 JavaScript

JavaScript adalah bahasa pemrograman berbasis prototipe yang berjalan di sisi klien. Jika berbicara dalam konteks web, sederhananya kita dapat memahami *JavaScript* sebagai bahasa pemrogrman yang berjalan khusus untuk di browser

atau halaman web agar halaman web menjadi lebih hidup. Jika dilihat dari suku katanya, *JavaScript* terdiri dari dua kata, yaitu *Java* dan *Script*. *Java* adalah bahasa pemrograman yang berorientasi objek, sedangkan *Script* adalah serangkaian instruksi program. *JavaScript* adalah bahasa *script* (bahasa pemrograman yang dapat memegang kontrol aplikasi) yang berbasis pada bahasa pemrograman Java. Namun *JavaScript* bukanlah bagian dari teknologi *Java* dari Sun. *Java* merupakan bahasa pemrograman berorientasi objek murni, sedangkan *JavaScript* digunakan secara prosedural.

Cara kerja JavaScript adalah mengakses elemen pada HTML dan membuat aksi jika elemen-elemen HTML itu mengalami perubahan,misalnya berubahnya warna halaman web begitu sebuah tombol di klik. *JavaScript* membuat sebuah halaman web menjadi lebih dinamis. *JavaScript* yang digunakan pada halaman web merupakan client side scripting yang berarti bahwa web browser mengidentifikasi dan menjalankan skrip program yang disisipkan dalam dokumen web (yang diterima dari server),dan mungkin memperbarui tampilan halaman di komputer pemakai (*user*) tanpa mengirimkan permintaan (*request*) baru kepada web server. Karena *JavaScript* pada halaman web adalah client side scripting, maka kode *JavaScript* yang ditulis satu file dengan dokumen HTML akan ditampilkan ketika sebuah halaman web dilihat source code-nya. Secara fungsional, *JavaScript* digunakan untuk menyediakan akses skrip pada objek yang dibenamkan (*embedded*).

Contoh sederhana dari penggunaan *JavaScript* adalah membuka halaman pop up, fungsi validasi pada form sebelum data dikirimkan ke server, merubah

gambar kursor ketika melewati objek tertentu, dan lain-lain. Yang harus diperhatikan dalam pengelolaan pemrograman *JavaScript* diantaranya *JavaScript* adalah bahasa pemrograman yang *case sensitive*, yang artinya *JavaScript* membedakan huruf kecil dan huruf besar. Hal ini sama seperti bahasa pemrograman Turbo C atau C++ dimana huruf "A" tidak sama dengan huruf "a".

JavaScript bekerja pada sisi browser, artinya untuk menampilkan halaman web, user menuliskan alamat web di *address bar url*. Setelah itu, *browser* mengambil file HTML (dengan file *JavaScript* yang melekat padanya jika memang ada) ke server yang beralamat di URL yang diketikkan oleh user. Selesai file diambil, file ditambilkan pada browser. Setelah file *Java Script* berada pada *browser*, barulah skrip *JavaScript* tersebut bekerja. Efek dari *JavaScript* yang bekerja pada sisi browser ini, *Java Script* dapat merespon perintah *user* dengan cepat, dan membuat halaman web menjadi lebih responsif. *Java Script* melakukan apa yang tidak bisa dilakukan oleh HTML, PHP, dan CSS dalam menangani hal-hal yang membutuhkan respon cepat terhadap aksi dari *user*. Misalnya, fungsi validasi pada *form*. Ketika *user* mengisi sebuah *form* yang divalidasi menggunakan *JavaScript*, *user* mengetikkan data lalu mengklik *submit*. Sebelum data dikirimkan ke server, data akan dicek terlebih dahulu pada *browser* menggunakan fungsi *JavaScript* yang ada pada halaman web. Sehingga jika memang data yang diisikan tidak valid, daripada membuang-buang waktu dengan mengirim data ke *server* baru divalidasi di *server*, lalu server mengirimkan respon balik mengenai ketidak validan input data *user*, lebih baik pengecekan *validasi* data *form* dilakukan secara lokal di browser menggunakan fungsi *JavaScript*.

2.10.1 Kelebihan JavaScript

Kelebihan dari bahasa pemrograman JavaScript adalah sebagai berikut :

- Ukuran file kecil Skrip dari *JavaScript* memiliki ukuran yang kecil sehingga ketika web yang memiliki *JavaScript* ditampilkan di browser, maka akses tampilannya akan lebih cepat dibandingkan ketika browser membuka suatu web yang memiliki skrip *Java*. Hal ini juga sangat berkaitan dengan daya kerja server. Semakin kecil space suatu web yang disimpan dalam suatu server, maka daya kerja server ketika di-browsing oleh user di internet akan tidak terlalu berat. Selain itu sifat *JavaScript* Client Side yang tidak perlu lagi ditangani oleh server ketika browser memanggil web dari sebuah server.
- Mudah untuk dipelajari *JavaScript* merupakan bahasa semi pemrograman yang merupakan gabungan antara bahasa pemrograman *Java* dengan bahasa kode *HTML* sehingga disebut juga bahasa *hybrid*. Walaupun *JavaScript* merupakan turunan dari *Java*, namun *JavaScript* tidak memiliki aturan serumit *Java*.
- Terbuka, *JavaScript* tidak terikat oleh hardware maupun software tertentu, bahkan sistem operasi seperti *windows* maupun *unix*. Karena ia bersifat terbuka, maka ia dapat dibuat maupun dibaca di semua jenis komputer.

2.10.2 Kekurangan *JavaScript*

Pada setiap bahasa pemrograman yang digunakan untuk membangun suatu aplikasi tentunya ada kekurangan dan kelebihan yang didapat. Begitupun juga

dengan bahasa pemrograman *JavaScript.* Berikut kekurangan yang dimiliki oleh javaScript :

- Skrip tidak terenkripsi

  Karena JavaScript bersifat *client side* , maka skrip yang kita buat di *text* editor dan telah dijadikan web di *server*, ketika user me-*request* web dari *server* tersebut, maka sintaks *JavaScript* akan langsung ditampilkan di browser. *User* bisa melihatnya dan menirunya dari *source* -nya.

- Kemampuan terbatas

  Walaupun *JavaScript* mampu membuat bentuk web menjadi interaktif dan dinamis, namun *JavaScript* tidak mampu membuat program aplikasi sendiri seperti *Java*.

- Keterbatasan objek

JavaScript tidak mampu membuat kelas-kelas yang bisa menampung objek-objek tambahan seperti *Java*, karena *JavaScript* telah memiliki objek yang built in pada struktur bahasanya.

2.11 Macromedia Dreamweaver 8

Macromedia Dreamweaver adalah program untuk membuat dan mengedit dokumen HTML secara visual dan mengelola halaman sebuah situs. Dreamweaver menyediakan banyak perangkat yang berkaitan dengan pengkodean dan fitur seperti HTML, CSS, JavaScript, PHP, ASP, ColdFusion, dan XML.

- Area Kerja Dreamweaver 8

Halaman Awal



Gambar 2.8 Halaman Awal Dreamweaver

(Sumber : http://www.elearning.atmaluhur.ac.id/.../Macromedia-Dreamweaver8)

Menu Di Dreamweaver

Pada halaman awal Dreamweaver 8 terdapat beberapa menu yang dapat dipilih :

- Open a Recent Item

  Pada menu ini akan ditampilkan beberapa file yang sebelumnya pernah kita buka dengan menggunakan Dreamweaver 8. Atau di paling bawah ada Open yang dapat digunakan untuk membuka file yang lain.

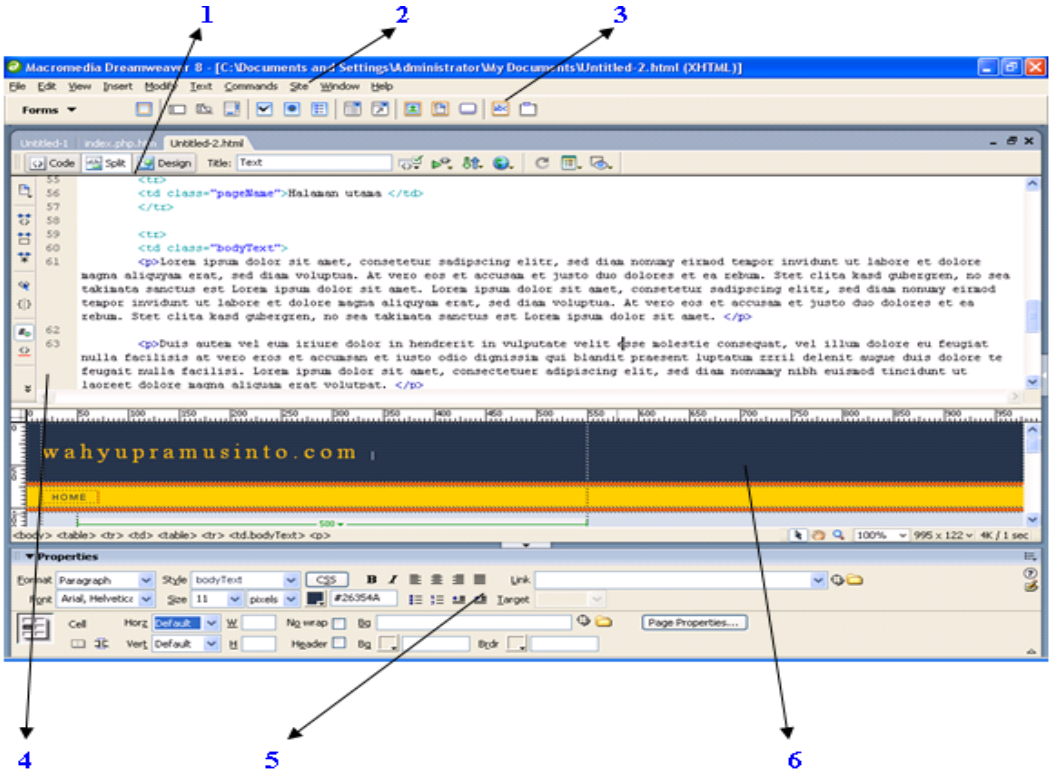
- Create New

  Pada menu ini kita dapat memilih dokumen baru apa yang akan kita buat dengan menggunakan Dreamweaver 8. Ada banyak pilihan, diantaranya HTML, ColdFusion, PHP, ASP, JavaScript, CSS.

- Create From Samples

  Pada menu ini kita dapat membuat file berdasarkan contoh yang sudah diberikan oleh Dreamweaver.

Halaman Utama Dreamweaver 8



Gambar 2.9 Halaman Utama Dreamweaver

(Sumber : http://www.elearning.atmaluhur.ac.id/.../Macromedia-Dreamweaver8)

Keterangan :

1. **Toolbar Dokumen**

   Toolbar dokumen digunakan untuk mengubah tampilan dan mengakses fungsi-fungsi penting secara cepat dan mudah. Pada toolbar dokumen

terdapat menu untuk berpindah antar dokumen kerja window dan mengatur tampilan area kerja. Untuk mengatur tampilan kita bisa memilih Code, Split dan Design.

2. **Menu Utama**

   Menu Utama berisi semua perintah yang dapat digunakan untuk bekerja pada Dreamweaver.

3. **Insert Bar**



Gambar 2.10 Inset Bar

(Sumber : http://www.elearning.atmaluhur.ac.id/.../Macromedia-Dreamweaver8)

Insert bar merupakan tempat semua perangkat kerja (tombol) tang digunakan untuk membuat halaman web. Insert bar mempunyai dua jenis tampilan, yaitu tampilan sebagai menu dan tampilan sebagai tab.