BAB II

LANDASAN TEORI

2.1 *Game* atau Permainan Komputer

2.1.1 Sejarah *Game* atau Permainan Komputer

Sejarah *Game* menurut situs www.wordpress.com yang diunduh pada tanggal 15 April 2012 sebagai berikut, generasi *game* pertama muncul dari ATARI 2600 merupakan konsole *game* pertama yang sukses di masanya atari 2600 ini dirilis pada Oktober 1977, setelah itu dikenal dengan nama VCS (*Video Computer System*)

Setelah sukses pada generasi pertama atari tidak menyerah sampai disitu saja untuk memajukan dunia *game* elektronik pada generasi kedua muncullah atari 7800 konsol ini dirilis pada juni 1986 pada atari 7800 ini ada sedikit kemajuan dengan menambahkan joystik sehingga user dapat lebih mudah untuk memainkan konsole *game* ini namun pada masanya harga dari konsole *game* ini selangit yaitu $140.

Setelah Atari 7800 ada NES Nintendo *entertaintment System*inilah konsole permainan pertama kali yang menggunakan 8 bit nah nintendo ini menghasilkan produk produk game yang lain daripada yang lain.

Setelah NES mencapai ekistensinya, lalu muncul lagi konsole *game* Sega Mega Drive sega mega drive ini merupakan generasi ke tiga dari dunia *game* sega menggunakan 16 bit dan dirilis pada tahun 1988.

Pada generasi ke empat dari sejarah *game* produksi *game* semakin menjamur dan berkembang pesat salah satu contoh pada generasi ke empat ini lahirlah playstation merupakan salah satu game konsole terlaris

Generasi sekarang, pada tahun ini sudah banyak sekali konsole game menjamur di dunia PS juga masih terus mengembangkan sistemnya dengan mengeluarkan PS3, Xbox juga mengeluarkan XBOX 360, lalu nintendo juga mengeluarkan WII.

2.1.2 Pengertian *Game* atau Permainan

Menurut Hans Daeng (dalam Andang Ismail, 2009: 17) permainan adalah bagian mutlak dari kehidupan anak maupun dewasa dan permainan merupakan bagian integral dari proses pembentukan kepribadian seseorang. Selanjutnya, Andang Ismail (2009: 26) menuturkan bahwa permainan ada dua pengertian, antara lain :

1. Permainan adalah sebuah aktifitas bermain yang murni mencari kesenangan tanpa mencari menang atau kalah.
2. Permainan diartikan sebagai aktifitas bermain yang dilakukan dalam rangka mencari kesenangan dan kepuasan, namun ditandai pencarian menang-kalah.

2.2 *Game Puzzle*

2.2.1 Pengertian *Game Puzzle*

Dalam dunia anak-anak terdapat berbagai jenis permainan, salah satu jenis permainan yang bermanfaat bagi anak dan bersifat edukatif adalah *puzzle*. *Puzzle* merupakan permainan yang membutuhkan kesabaran dan ketekunan anak dalam merangkainya. Dengan terbiasa bermain *puzzle*, lambat laun mental anak juga akan terbiasa untuk bersikap tenang, tekun, dan sabar dalam menyelesaikan sesuatu. Kepuasan yang didapat saat ia menyelesaikan *puzzle* pun merupakan salah satu pembangkit motifasi untuk mencoba hal-hal yang baru baginya.

Puzzle sudah bisa dimainkan oleh anak berusia 5 tahun, tentunya dengan kepingan gambar (*puzzle*) yang sedikit dan tingkat kesulitannya lebih mudah. Untuk awal, kenalkan anak anda dengan *puzzle* sederhana yang terdiri dari sebuah keping saja, misalnya angka dari 1-15. Makin tinggi usia anak, biasanya tingkat kesulitan lebih rumit.

2.2.2 Manfaat Bermain *Puzzle* sebagai Game Edukasi

permainan *puzzle* memiliki banyak manfaat diantaranya :

- **Meningkatkan kemampuan berpikir dan membuat anak belajar berkonsentrasi.**

  Saat bermain *puzzle*, anak akan melatih sel-sel otaknya untuk mengembangkan kemampuan berpikirnya dan berkonsentrasi untuk menyelesaikan potongan-potongan kepingan gambar tersebut.

- **Melatih koordinasi tangan dan mata.**

Puzzle dapat melatih koordinasi tangan dan mata anak untuk mencocokkan keping-keping *puzzle* dan menyusunnya menjadi satu gambar. *Puzzle* juga membantu anak mengenal dan menghapal bentuk.

- **Meningkatkan Keterampilan Kognitif.**

  Keterampilan kognitif (*cognitive skill*) berkaitan dengan kemampuan untuk belajar dan memecahkan masalah. *Puzzle* adalah permainan yang menarik bagi anak balita karena anak balita pada dasarnya menyukai bentuk gambar dan warna yang menarik. Dengan bermain *puzzle* anak akan mencoba memecahkan masalah yaitu menyusun gambar.

- **Belajar bersosialisasi.**

  Dua anak yang bermain bersama-sama tentunya butuh diskusi untuk merancang kepingan-kepingan gambar dari *puzzle* tersebut. Anak yang lebih besar akan merasa senang jika dapat membantu anak yang lebih kecil, sebaliknya pun begitu, jadi akan tercipta suasana yang nyaman dan terciptanya interaksi ketika bermain.

2.3 Kecerdasan Buatan (*Artificial Intelegence*)

2.3.1 Pengertian Kecerdasan Buatan (*Artificial Intelegence*)

Artificial Intelligence (AI) atau kecerdasan buatan merupakan cabang dari ilmu komputer yang berhubungan dengan pengautomatisasi tingkah laku cerdas. Pernyataan tersebut juga dapat dijadikan definisi dari AI. Definisi ini menunjukkan bahwa AI adalah bagian dari komputer sehingga harus didasarkan pada *sound theoretical* (teori suara) dan prinsip-prinsip aplikasi dari bidangnya.

Prinsip-prinsip ini meliputi struktur data yang digunakan dalam representasi pengetahuan, algoritma yang diperlukan untuk mengaplikasikan pengetahuan tersebut, serta bahasa dan teknik pemrograman yang digunakan dalam mengimplementasikannya. Dari beberapa perspektif, AI dapat dipandang sebagai:

1. Dari perspektif kecerdasan, AI adalah bagaimana membuat mesin yang cerdas dan dapat melakukan hal-hal yang sebelumnya hanya dapat dilakukan manusia.
2. Dari perspektif bisnis, AI adalah sekelompok alat bantu *(tools)* yang berdayaguna dan metodologi yang menggunakan alat-alat bantu tersebut untuk menyelesaikan masalah-masalah bisnis.
3. Dari perspektif pemrograman, AI meliputi studi tentang pemrograman simbolik, pemecahan masalah, dan proses pencarian *(search)*.
4. Dari perspektif penelitian:

a. Riset tentang AI dimulai pada awal tahun 1960-an, percobaan pertama adalah membuat program permainan catur, membuktikan teori, dan *general problem solving*.

b. *Artificial intelligence* adalah nama pada akar dari studi area.

Ada dua hal yang sangat mendasar mengenai penelitian-penelitian AI, yaitu *knowledge representation* (representasi pengetahuan) dan *search* (pelacakan). Para peneliti AI terus mengembangkan berbagai jenis teknik baru dalam menangani sejumlah permasalahan yang tergolong ke dalam AI seperti *vision* dan percakapan, pemrosesan bahasa alami, dan permasalahan khusus seperti diagnosa medis.

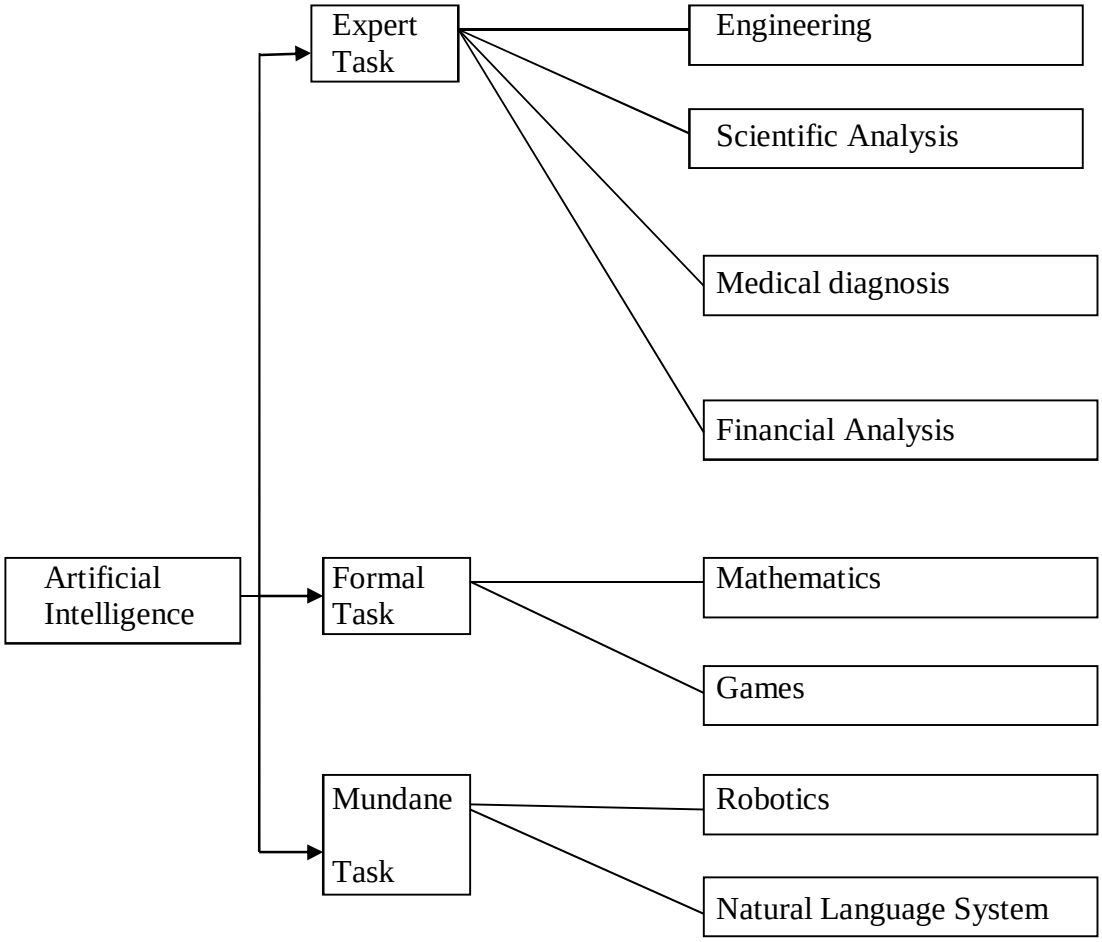
AI seperti bidang ilmu lainnya juga memiliki sejumlah sub-disiplin ilmu yang sering digunakan untuk pendekatan yang esensial bagi penyelesaian suatu masalah dan dengan aplikasi bidang AI yang berbeda. Gambar 2.1 merupakan sejumlah bidang-bidang tugas *(task domains)* dari AI.

Setiap permainan memiliki aturan main. Hal ini mempermudah upaya menghasilkan ruang pencarian dan memberikan kebebasan pada para peneliti dari bermacam-macam ambisi dan kompleksitas sifat serta kurangnya struktur permasalahan. Papan konfigurasi yang digunakan untuk memainkan permainan ini mudah direpresentasikan pada komputer dan tidak memerlukan bentuk yang kompleks.

Permainan dapat menghasilkan sejumlah besar pencarian ruang. Hal ini cukup besar dan kompleks sehingga membutuhkan suatu teknik yang tangguh untuk menentukan alternatif pengeksplorasian ruang permasalahan.

Teknik ini dikenal dengan nama *heuristik* dan merupakan area utama dari penelitian tentang AI. Banyak hal yang biasanya dikenal sebagai kecerdasan tampaknya berada dalam *heuristik* yang digunakan oleh manusia untuk menyelesaikan permasalahannya.

Seperti yang telah dibahas sebelumnya bahwa pada AI terdapat berbagai bidang ilmu yang merupakan sejumlah bidang-bidang tugas *(task domains)* dari AI, seperti yang tergambar pada bagan 2.1, seperti dibawah ini :



Gambar 2.1. Bidang-bidang tugas *(task domains)* dari AI

2.4 Algoritma

2.4.1 Sejarah Algoritma

Menurut situs yang diunduh pada tanggal 11 April 2012 (www.internetsehat.org) Kata algoritma berasal dari latinisasi nama seorang ahli matematika dari UzbekistanAl Khawārizmi (hidup sekitar abad ke-9), sebagaimana tercantum pada terjemahan karyanya dalam bahasa latin dari abad ke-12 "*Algorithmi de numero Indorum*". Pada awalnya kata algoritma adalah

istilah yang merujuk kepada aturan-aturan aritmetis untuk menyelesaikan persoalan dengan menggunakan bilangan numerik arab (sebenarnya dari India, seperti tertulis pada judul di atas). Pada abad ke-18, istilah ini berkembang menjadi algoritma, yang mencakup semua prosedur atau urutan langkah yang jelas dan diperlukan untuk menyelesaikan suatu permasalahan.

2.4.2 Pengertian Algoritma

Menurut situs yang diunduh pada tanggal 11 April 2012 (www.internetsehat.org) 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 algoritmaadalah :

1. Algoritma haruslah benar. Artinya algoritma akan memberikan keluaran yang dikehendaki dari sejumlah masukan yang diberikan. Tidak peduli sebagus apapun algoritma, kalau memberikan keluaran yang salah, pastilah algoritma tersebut bukanlah algoritma yang baik.
2. Pertimbangan kedua yang harus diperhatikan adalah kita harus mengetahui seberapa baik hasil yang dicapai oleh algoritma tersebut. Hal ini penting terutama pada algoritma untuk menyelesaikan masalah yang memerlukan aproksimasi hasil (hasil yang hanya berupa

pendekatan). Algoritma yang baik harus mampu memberikan hasil yang sedekat mungkin dengan nilai yang sebenarnya.

3. Ketiga adalah efisiensi algoritma. Efisiensi algoritma dapat ditinjau dari 2 hal yaitu efisiensi waktu dan memori. Meskipun algoritma memberikan keluaran yang benar (paling mendekati), tetapi jika kita harus menunggu berjam-jam untuk mendapatkan keluarannya, algoritma tersebut biasanya tidak akan dipakai, setiap orang menginginkan keluaran yang cepat. Begitu juga dengan memori, semakin besar memori yang terpakai maka semakin buruklah algoritma tersebut. Dalam kenyataannya, setiap orang bisa membuat algoritma yang berbeda untuk menyelesaikan suatu permasalahan, walaupun terjadi perbedaan dalam menyusun algoritma, tentunya kita mengharapkan keluaran yang sama. Jika terjadi demikian, carilah algoritma yang paling efisien dan cepat.

2.4.3 Jenis-Jenis Algoritma

Menurut situs yang diunduh pada tanggal 11 April 2012 (www.internetsehat.org) Terdapat beragam klasifikasi algoritma dan setiap klasifikasi mempunyai alasan tersendiri. Salah satu cara untuk melakukan klasifikasi jenis-jenis algoritma adalah dengan memperhatikan paradigma dan metode yang digunakan untuk mendesain algoritma tersebut. Beberapa paradigma yang digunakan dalam menyusun suatu algoritma akan dipaparkan dibagian ini.

Masing-masing paradigma dapat digunakan dalam banyak algoritma yang berbeda.

1. *Divide and Conquer*, paradigma untuk membagi suatu permasalahan besar menjadi permasalahan-permasalahan yang lebih kecil. Pembagian masalah ini dilakukan terus menerus sampai ditemukan bagian masalah kecil yang mudah untuk dipecahkan. Singkatnya menyelesaikan keseluruhan masalah dengan membagi masalah besar dan kemudian memecahkan permasalahan-permasalahan kecil yang terbentuk.
2. *Dynamic programming*, paradigma pemrograman dinamik akan sesuai jika digunakan pada suatu masalah yang mengandung sub struktur yang optimal, dan mengandung beberapa bagian permasalahan yang tumpang tindih . Paradigma ini sekilas terlihat mirip dengan paradigma *Divide and Conquer*, sama-sama mencoba untuk membagi permasalahan menjadi sub permasalahan yang lebih kecil, tapi secara intrinsik ada perbedaan dari karakter permasalahan yang dihadapi.
3. Metode serakah, Sebuah algoritma serakah mirip dengan sebuah Pemrograman dinamik, bedanya jawaban dari submasalah tidak perlu diketahui dalam setiap tahap; dan menggunakan pilihan "serakah" apa yang dilihat terbaik pada saat itu.

2.4.4 Perbedaan Algoritma dan Program

Adapun perbedaan antara algoritma dengan program menurut situs yang diunduh pada tanggal 11 April 2012 (www.internetsehat.org), Program adalah

kumpulan pernyataan komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ditulis dengan menggunakan bahasa pemrograman. Jadi bisa disebut bahwa program adalah suatu implementasi dari bahasa pemrograman. Beberapa pakar memberi formula bahwa :

Program = Algoritma + Bahasa (Struktur Data)

Bagaimanapun juga struktur data dan algoritma berhubungan sangat erat pada sebuah program. Algoritma yang baik tanpa pemilihan struktur data yang tepat akan membuat program menjadi kurang baik, demikian juga sebaliknya.

Pembuatan algoritma mempunyai banyak keuntungan di antaranya :

1. Pembuatan atau penulisan algoritma tidak tergantung pada bahasa pemrograman manapun, artinya penulisan algoritma independen dari bahasa pemrograman dan komputer yang melaksanakannya.
2. Notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
3. Apapun bahasa pemrogramannya, *output* yang akan dikeluarkan sama karena algoritmanya sama.

Beberapa hal yang perlu diperhatikan dalam membuat algoritma :

a. Teks algoritma berisi deskripsi langkah-langkah penyelesaian masalah. Deskripsi tersebut dapat ditulis dalam notasi apapun asalkan mudah dimengerti dan dipahami.
b. Tidak ada notasi yang baku dalam penulisan teks algoritma seperti notasi bahasa pemrograman. Notasi yang digunakan dalam menulis algoritma disebut notasi algoritmik.

c. Setiap orang dapat membuat aturan penulisan dan notasi algoritmik sendiri. Hal ini dikarenakan teks algoritma tidak sama dengan teks program. Namun, supaya notasi algoritmik mudah ditranslasikan ke dalam notasi bahasa pemrograman tertentu, maka sebaiknya notasi algoritmik tersebut berkorespondensi dengan notasi bahasa pemrograman secara umum.
d. Notasi algoritmik bukan notasi bahasa pemrograman, karena itu *pseudocode* dalam notasi algoritmik tidak dapat dijalankan oleh komputer. Agar dapat dijalankan oleh komputer, *pseudocode* dalam notasi algoritmik harus ditranslasikan atau diterjemahkan ke dalam notasi bahasa pemrograman yang dipilih. Perlu diingat bahwa orang yang menulis program sangat terikat dalam aturan tata bahasanya dan spesifikasi mesin yang menjalannya.
e. Algoritma sebenarnya digunakan untuk membantu kita dalam mengkonversikan suatu permasalahan ke dalam bahasa pemrograman.
f. Algoritma merupakan hasil pemikiran konseptual, supaya dapat dilaksanakan oleh komputer, algoritma harus ditranslasikan ke dalam notasi bahasa pemrograman. Ada beberapa hal yang harus diperhatikan pada translasi tersebut, yaitu :

1. Pendeklarasian variabel, Untuk mengetahui dibutuhkannya pendeklarasian variabel dalam penggunaan bahasa pemrograman apabila tidak semua bahasa pemrograman membutuhkannya.

2. Pemilihan tipe data, Apabila bahasa pemrograman yang akan digunakan membutuhkan pendeklarasian variabel maka perlu hal ini dipertimbangkan pada saat pemilihan tipe data.
3. Pemakaian instruksi-instruksi, beberapa instruksi mempunyai kegunaan yang sama tetapi masing-masing memiliki kelebihan dan kekurangan yang berbeda.
4. Aturan sintaksi, pada saat menuliskan program kita terikat dengan aturan sintaksis dalam bahasa pemrograman yang akan digunakan.
5. Tampilan hasil, pada saat membuat algoritma kita tidak memikirkan tampilan hasil yang akan disajikan. Hal-hal teknis ini diperhatikan ketika mengkonversikannya menjadi program.
6. Cara pengoperasian *compiler* atau *interpreter*, bahasa pemrograman yang digunakan termasuk dalam kelompok *compiler* atau *interpreter*.

2.4.5 Struktur Dasar Algoritma

Menurut situs yang diunduh pada tanggal 12 April 2012 www.wikipedia.com yang termasuk struktur dasar algoritma adalah sebagai berikut, Algoritma berisi langkah-langkah penyelesaian suatu masalah. Langkah-langkah tersebut dapat berupa runtunan aksi (*sequence*), pemilihan aksi (*selection*), pengulangan aksi (*iteration*) atau kombinasi dari ketiganya. Jadi struktur dasar pembangunan algoritma ada tiga, yaitu:

1. Struktur Runtunan
2. Digunakan untuk program yang pernyataannya *sequential* atau urutan.

3. Struktur Pemilihan
4. Digunakan untuk program yang menggunakan pemilihan atau penyeleksian kondisi.
5. Struktur Perulangan
6. Digunakan untuk program yang pernyataannya akan dieksekusi berulang-ulang.

Dalam Algoritma, tidak dipakai simbol-simbol/sintaks dari suatu bahasa pemrograman tertentu, melainkan bersifat umum dan tidak tergantung pada suatu bahasa pemrograman apapun juga. Notasi-notasi algoritma dapat digunakan untuk seluruh bahasa pemrograman manapun.

2.5 Algoritma *Depth-First Search*

2.5.1 Pengertian Algoritma *Depth-First Search*

Menurut situs blogspot.com yang diunduh pada tanggal 03 April 2012 *Depth-frist search* adalah suatu jenis dan metode pencarian *tree/graph.* Proses pencarian didalam g*raph* digunakan untuk mencari di mana simpul bisa di jangkau dari root dengan mengikuti jalan (*path*) yang dicari. metode pencarian solusi dimana Proses pencarian dilakukan pada semua anaknya sebelum dilakukan pencarian ke node-node yang selevel. Pencarian dimulai dari node akar ke level yang lebih tinggi. Proses ini diulangi terus hingga ditemukannya solusi.

2.5.2 Kegunaan Algoritma *Depth-Frist search*

Adapun kegunaan dari algoritma *Depth-Frist search*, antara lain sebagai berikut :

1. Mencari komponen yang terhubung
2. Pengurutan topologi
3. Mencari 2 komponen yang terhubung
4. Mencari *strongly connected component*
5. Menyelesaikan *puzzle* yang mempunyai hanya 1 solusi
6. Mencari *biconnectivity* di dalam graph.

2.5.3 Cara Kerja Algoritma *Depth-First Search*

Dalam algoritma *Depth-First Search*, simpul yang telah dikunjungi disimpan dalam suatu tumpukan (*stack*). Antrian ini digunakan untuk mengacu simpul-simpul yang akan dikunjungi sesuai urutan tumpukan (masuk terakhir, keluar pertama) dan mempermudah proses runut-balik jika simpul sudah tidak mempunyai anak (simpul pada kedalaman maksimal). Untuk memperjelas cara kerja algoritma *Depth-Frist search* beserta tumpukan yang digunakannya, berikut langkah-langkah algoritma *Depth-Frist search*:

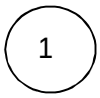
1. Masukkan simpul ujung (akar) ke dalam tumpukan
2. Ambil simpul dari tumpukan teratas, lalu cek apakah simpul merupakan solusi.
3. Jika simpul merupakan solusi, pencarian selesai dan hasil dikembalikan.

4. Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) ke dalam tumpukan
5. Jika tumpukan kosong dan setiap simpul sudah dicek, pencarian selesai dan mengembalikan hasil solusi tidak ditemukan
6. Ulangi pencarian dari langkah kedua.

Pencarian dengan metode ini dilakukan dari simpulawal secara mendalam hingga yang paling akhiratau sampai ditemukannya solusi. Sebagai ilustrasinya dapat dilihat pada langkah-langkah di bawah ini.

Langkah Awal:

Dimulai dari simpul akar, atau simpul tujuan yang kita pilih, yaitu :



Gambar 2.2: Langkah Awal

Kita akan menggunakan dua list info untuk menyimpan langkah yang kita lakukan, yaitu Open dan Close. Open akan menyimpan langkah yang perlu kita lakukan, dan Close akan menyimpan langkah yang telah kita lakukan. Saat ini, kita hanya punya

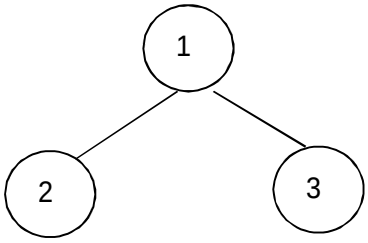
titik awal yang kita pilih, yaitu simpul 1. Kita belum melakukan apapun ke simpul tersebut, sehingga sekarang kita tambahkan ke Open.

Open : 1

Close : < kosong >

Langkah 1

Sekarang, kita memeriksa tetangga- tetangga dari simpul 1. Dengan kata lain, mengambil simpul awal dari Open kita dan kemudian menelusuri tetangga-tetangganya.



Gambar 2.3 :Langkah 1

Tetangga-tetangga dari simpul 1 adalah simpul 2 dan 3. Karena sekarang kita telah selesai dengan simpul 1, kita dapat menghapusnya dari Open dan menambahkannya ke Close. Kita belum selesai dengan langkah ini tentunya. Tambahkan dua simpul

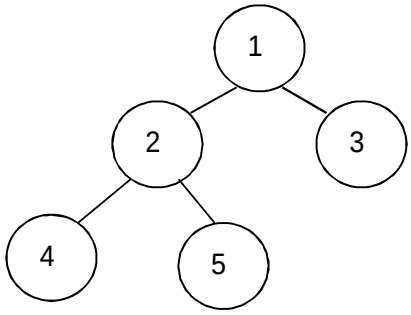
2 dan 3 tadi ke Open. Sekarang, Open dan Close berisi data sebagai berikut :

Open : 2, 3

Close: 1

Langkah 2

Open kini berisi dua simpul. Untuk *depth first search*, selalu kita telusuri simpulpertama dari Open. Simpul pertama pada opensekarang adalah simpul 2. 2 bukan tujuan kita,sehingga kita telusuri tetangga-tetangganya.



Gambar 2.4 : Langkah 2

Karena kini 2 telah diluaskan, kita dapat menghapusnya dari Open dan menambahkannya ke Close. Kini, simpul baru kita adalah 4 dan 5, dan kita tambahkan simpul ini ke awal dari Open (menggantikan 2 yang telah dihapus).

Open: 4, 5, 3

Close: 1, 2

Langkah 3 :

Kita akan melihat bahwa sebuah pola mulai terbentuk.Karena 4 ada di awal dari Open, maka kita meluaskannya. 4 bukan tujuan kita, dan tidak berisi satu pun tetangga. Selanjutnya, kita dapat menghapus 4 dari Open dan menambahkanya ke Close.

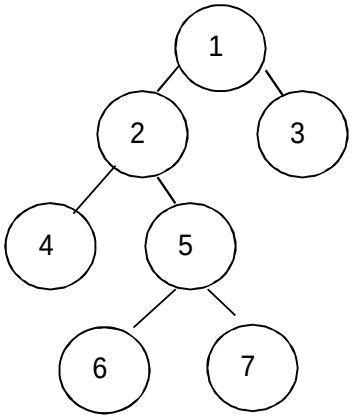
Open: 5, 3

Close: 1, 2, 4

Langkah 4:

Sekarang kita meluaskan simpul 5 dari Open. 5 bukan tujuan kita, sehingga kita telusuri tetangga-tetangganya dan menemukan bahwa ia memiliki tetangga 6 dan 7. Karena 6 adalah tujuan kita, kita masih belum berhenti di sini tentunya. Di samping 6

berada pada lintasan kita, kita hanya akan berhenti begitu kita akan meluaskan simpul tujuan kita, dalam kasus ini yaitu 6 :



Gambar 2.5 : Langkah 4

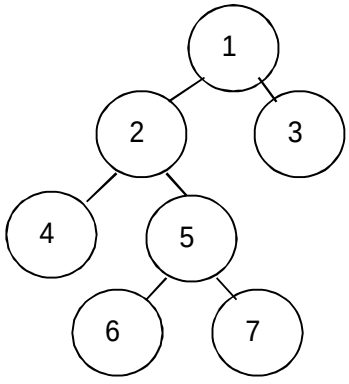
Kita akan menghapus simpul 5 dari Open dan menambahkan simpul 6 dan 7 ke dalamnya. Kemudian, simpul 5 ditambahkan ke Close:

Open: 6, 7, 3

Close: 1, 2, 4, 5

Langkah 5

Sekarang kita meluaskan simpul 6. Karena 6 adalah tujuan kita, maka kita berhenti :



Gambar 2.6 : Langkah 5

Kita menghapus 6 dari Open dan menambahkannya ke Close. Karena kita telah sampai di tujuan, maka tidak perlu lagi meluaskan 6 dan mencari tetangganya. Open dan Close akhir kita akan berisi data sebagai berikut :

Open: 7, 3

Close: 1, 2, 4, 5, 6

Lintasan akhir yang akan diambil oleh metode *Depth First Search* kita adalah nilai terakhir dari Close, yaitu: 1, 2,4, 5, 6.

2.5.4 Kelebihan dan kekurangan Algoritma *Depth-frist search*

2.5.4.1 Kelebihan Algoritma *Depth-frist search*

a. Membutuhkan memori yang relative kecil, karena hanya node-node pada lintasan yang aktif saja yang disimpan.
b. Secara kebetulan, metode depth-first search akan menemukan solusi tanpa harus menguji lebih banyak lagi.

2.5.4.2 Kekurangan Algoritma *Depth-frist search*

a. Memungkinkan tidak ditemukannya tujuan yang diharapakan.
b. Hanya akan menemukan 1 solusi pada setiap pencarian.

2.6 NetBeans

2.6.1 Sejarah NetBeans

Menurut situs yang diunduh pada tanggal 09 Juni 2012 www.wikipedia.org Pengembangan NetBeans diawali dari Xelfi, sebuah proyek

mahasiswa tahun 1997 di bawah bimbingan Fakultas Matematika dan Fisika Universitas Charles, Praha. Sebuah perusahaan kemudian dibentuk untuk proyek tersebut dan menghasilkan versi komersial NetBeans IDE hingga kemudian dibeli oleh Sun Microsystem pada tahun 1999. Sun kemudian menjadikan NetBeans open source pada bulan Juni tahun 2000. Sejak itu komunitas NetBeans terus berkembang.

NetBeans mengacu pada dua hal, yakni platform untuk pengembangan aplikasi desktop java, dan sebuah *Integrated Development Environment* (IDE) yang dibangun menggunakan platform NetBeans.

Platform NetBeans memungkinkan aplikasi dibangun dari sekumpulan komponen perangkat lunak moduler yang disebut 'modul'. Sebuah modul adalah suatu arsip Java (Java *archive*) yang memuat kelas-kelas Java untuk berinetraksi dengan *NetBeans Open API* dan file manifestasi yang mengidentifikasinya sebagai modul. Aplikasi yang dibangun dengan modul-modul dapat dikembangkan dengan menambahkan modul-modul baru. Karena modul dapat dikembangkan secara independen, aplikasi berbasis platform NetBeans dapat dengan mudah dikembangkan oleh pihak ketiga secara mudah dan *powerful*.

2.6.2 Pengertian NetBeans

NetBeans adalah Integrated Development Environment (IDE) berbasiskan Java dari Sun Microsystems yang berjalan di atas Swing. Swing sebuah teknologi Java untuk pengembangan aplikasi Desktop yang dapat bejalan di berbagai macam platforms seperti Windows, Linux, Mac OS X and Solaris.

Suatu IDE adalah lingkup pemrograman yang diintegrasikan kedalam suatu aplikasi perangkat lunak yang menyediakan pembangun Graphic User Interface (GUI), suatu text atau kode editor, suatu compiler atau interpreter dan suatu debugger.

Netbeans merupakan software development yang Open Source, dengan kata lain software ini di bawah pengembangan bersama, bebas biaya

NetBeans merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra. Sun Microsystems mendirikan proyek kode terbuka NetBeans pada bulan Juni 2000 dan terus menjadi sponsor utama.Saat ini terdapat dua produk : NetBeans IDE dan NetBeans Platform.

2.6.3 Produk NetBeans

Menurut Blog yang diunduh pada tanggal 09 Juni 2012 bahwa terdapat dua produk NetBeans, Antara lain : NetBeans IDE dan NetBeans Platform.

1. NetBeans IDE (*Integrated Development Environment*)

   The NetBeans IDE adalah sebuah lingkungan pengembangan - sebuah kakas untuk pemrogram menulis, mengompilasi, mencari kesalahan dan menyebarkan program. Netbeans IDE ditulis dalam Java - namun dapat mendukung bahasa pemrograman lain. Terdapat banyak modul untuk memperluas Netbeans IDE. Netbeans IDE adalah sebuah produk bebas dengan tanpa batasan bagaimana digunakan. NetBeans IDE adalah IDE open source yang ditulis sepenuhnya dengan bahasa pemrograman Java

menggunakan platform NetBeans. NetBeans IDE mendukung pengembangan semua tipe aplikasi Java (J2SE, web, EJB, dan aplikasi *mobile*). Fitur lainnya adalah sistem proyek berbasis Ant, kontrol versi, dan *refactoring*.

Versi terbaru saat ini adalah NetBeans IDE 5.5.1 yang dirilis Mei 2007 mengembangkan fitur-fitur Java EE yang sudah ada (termasuk *Java Persistence support*, EJB-3 dan JAX-WS). Sementara paket tambahannya, NetBeans Enterprise Pack mendukung pengembangan aplikasi perusahaan Java EE 5, meliputi alat desain visual SOA, skema XML, *web service* dan pemodelan UML. NetBeans C/C++ Pack mendukung proyek C/C++.

Modularitas: Semua fungsi IDE disediakan oleh modul-modul. Tiap modul menyediakan fungsi yang didefinisikan dengan baik, seperti dukungan untuk bahasa pemrograman Java, editing, atau dukungan bagi CVS. NetBeans memuat semua modul yang diperlukan dalam pengembangan Java dalam sekali *download*, memungkinkan pengguna untuk mulai bekerja sesegera mungkin. Modul-modul juga mengijinkan NetBeans untuk bisa dikembangkan. Fitur-fitur baru, seperti dukungan untuk bahasa pemrograman lain, dapat ditambahkan dengan menginstal modul tambahan. Sebagai contoh, Sun Studio, Sun Java Studio Enterprise, dan Sun Java Studio Creator dari Sun Microsystem semuanya berbasis NetBeans IDE.

2. NetBeans Platform

NetBeans Platform sebuah fondasi yang modular dan dapat diperluas yang dapat digunakan sebagai perangkat lunak dasar untuk membuat aplikasi desktop yang besar. Mitra ISV menyediakan plug-in bernilai tambah yang dapat dengan mudah diintegrasikan ke dalam Platform dan dapat juga digunakan untuk membuat kakas dan solusi sendiri.

Platform NetBeans adalah framework yang dapat digunakan kembali (*reusable*) untuk menyederhanakan pengembangan aplikasi desktop. Ketika aplikasi berbasis platform NetBeans dijalankan, kelas Main dari platform dieksekusi. Modul-modul yang tersedia ditempatkan di sebuah *registry* di dalam memori, dan tugas *startup* modul dijalankan. Secara umum, kode modul dimuatkan ke dalam memori hanya ketika ia diperlukan.

Aplikasi dapat menginstal modul secara dinamis. Aplikasi dapat memasukkan modul *Update Center* untuk mengijinkan pengguna aplikasi men-*downloaddigitally-signed upgrade* dan fitur-fitur baru secara langsung ke dalam aplikasi yang berjalan. Penginstalan kembali sebuah upgrade atau rilis baru tidak memaksa pengguna untuk men-*download* keseluruhan aplikasi lagi.

Platform NetBeans menawarkan layanan-layanan yang umum bagi aplikasi desktop, mengijinkan pengembang untuk fokus ke logika yang spesifik terhadap aplikasi. Fitur-fitur yang disediakan oleh platform NetBeans:

- Manajemen antarmuka (misal: menu & toolbar)
- Manajemen pengaturan pengguna
- Manajemen penyimpanan (menyimpan dan membuka berbagai macam data)
- Manajemen jendela
- *Wizard framework* (mendukung dialog langkah demi langkah).

2.6.4 Lisensi NetBeans

Sejak Juli 2006, NetBeans IDE dilisensikan di bawah *Common Development and Distribution License* (CDDL), yaitu lisensi yang berbasis *Mozilla Public License* (MPL).

2.6.5 Paket-Paket Tambahan NetBeans IDE

1. ***NetBeans Mobility Pack***

   NetBeans Mobility Pack adalah alat untuk mengembangkan aplikasi yang berjalan pada perangkat bergerak (*mobile*), umumnya telepon seluler, tetapi juga mencakup PDA, dan lain-lain.

   NetBeans Mobility Pack dapat digunakan untuk menulis, menguji, dan *debugging* aplikasi untuk perangkat bergerak yang menggunakan teknologi berplatform Java Micro Edition (platform Java ME). Paket ini mengintegrasikan dukungan terhadap *Mobile Information Device Profile* (MIDP) 2.0, *Connected Limited Device Configuration* (CLDC) 1.1, dan *Connected Device Configuration (CDC). Emulator dari pihak ketiga*

dapat diintegrasikan dengan mudah untuk lingkungan pengujian yang lebih kokoh. NetBeans Mobility Pack *saat ini tersedia dalam dua klaster yang berbeda, yang satu memuat CDC dan yang lainnya CLDC.*

2. ***NetBeans ProfilerNetBeans Profiler*** adalah alat untuk mengoptimalkan aplikasi Java, membantu menemukan kebocoran memori dan mengoptimalkan kecepatan.

   Profiler ini berdasarkan sebuah proyek riset *Sun Laboratories* yang dahulu bernama Jfluid. Riset tersebut mengungkap teknik tertentu yang dapat digunakan untuk menurunkan *overhead* proses *profiling* aplikasi Java. Salah satu dari teknik tersebut adalah instrumentas i kode byte dinamis, yang berguna untuk *profiling* aplikasi Java yang besar. Dengan menggunakan instrumentasi kode byte dinamis dan algoritma-algoritma tambahan, *Netbeans Profiler* mampu mendapatkan informasi *runtime* aplikasi yang terlalu besar atau kompleks bagi *profiler* lain. NetBeans IDE 6.0 akan mendukung *Profiling Point* yang memungkinkan kita memprofilkan titik yang tepat dari eksekusi dan mengukur waktu eksekusi.

3. ***NetBeans C/C++ Pack***

   NetBeans C/C++ Pack menambahkan dukungan terhadap pengembang C/C++ ke NetBeans IDE 5.5. Paket ini memperbolehkan pengembang menggunakan sekumpulan kompiler dan alat sendiri bersama dengan NetBeans IDE untuk membangun aplikasi *native* untuk MS Windows, Linux, dan Solaris. Paket ini membuat editor mengenali bahasa C/C++ dan menyediakan *project template,* browser kelas yang dinamis, dukungan

pembuatan file dan fungsionalitas *debugger*. Para pengembang juga dapat mengembangkan paket tersebut dengan fungsionalitas tambahan mereka sendiri.

4. *NetBeans Enterprise Pack*

NetBeans Enterprise Pack memperluas dukungan terhadap ngembangan aplikasi perusahaan dan *web service* di NetBeans IDE 5.5.

Enterprise Pack ini mengembangkan kemampuan untuk menulis, menguji, dan *debug* aplikasi dengan arsitektur berorientasi layanan (*Service-Oriented Architecture*) menggunakan XML, BPEL, dan Java *web service*. Paket ini menambahkan alat desain visual untuk pemodelan UML, skema XML, dan *web service orchestration*, juga dukungan untuk *web service* dengan menggunakan identitas yang aman. Paket ini juga menginstal dan mengkonfigurasi *runtime* yang diperlukan, termasuk mesin BPEL dan server manajemen identitas yang terintegrasi dengan *Sun Java System Application Server*.

5. *NetBeans Ruby Pack*

Versi NetBeans 6.0 mendatang akan mengijinkan pengembangan IDE menggunakan Ruby dan Jruby, sebagaimana Rails untuk dua implementasi Ruby yang lain. *Preview NetBeans Ruby Pack* tersedia sejak rilis *Milestone 7* NetBeans 6.

Ruby Pack memasukkan fungsionalitas editor seperti:

a. pengeditan dasar

b. pewarnaan sintaks untuk Ruby

c. pelengkapan kode

d. *occurence highlighting*

e. pop-up dokumentasi yang terintegrasi untuk pemanggilan Ruby API

f. analisis semantik dengan *highlighting* parameter dan variabel lokal yang tidak terpakai

6. *NetBeans JavaScript Editor*

NetBeans JavaScript Editor menyediakan perluasan dukungan terhadap JavaScript dan CSS. Fitur-fiturnya antara lain:

1. Editor JavaScript
    a. *syntax highlighting*
    b. pelengkapan kode untuk objek dan fungsi *native*
    c. semua fitur dalam editor NetBeans
    d. pembuatan kerangka kelas JavaScript secara otomatis
    e. pembuatan pemanggilan AJAX dari *template*
2. Ekstensi editor CSS
    a. pelengkapan kode untuk nama-nama *style*
    b. navigasi cepat melalui panel navigator
    c. penampilan deklarasi aturan CSS di *List View*
    d. penampilan struktur file di *Tree View*
    e. mengurutkan *outline view* berdasarkan nama, tipe, atau urutan deklarasi (*List & Tree*)
    f. pembuatan deklarasi aturan (hanya *Tree*)

g. pemfaktoran kembali sebagian nama rule (hanya *Tree*)

2.6.6 Kelebihan dan Kekurang NetBeans

1. Kelebihan NetBeans GUI Builder :

   Salah satu yang menjadi kelebihan NetBeans GUI Builder adalah yang telah disebutkan diatas, yaitu GRATIS. Selain itu NetBeans GUI Builder sangat kompetebel dengan Swing karena memang langsung dikembangkan oleh Sun yang notabenenya sebagai pengembang Swing.

2. Kekurangan NetBeans GUI Builder :

   NetBeans hanya mensupport 1 pengembangan Java GUI, yaitu Swing, yang padahal ada Java GUI yang dikembangkan oleh eclipse yang bernama SWT dan JFace yang sudah cukup populer.

NetBeans mempatenkan source untuk Java GUI yang sedang dikerjakan dalam sebuah Generated Code, sehingga programmer tak dapat mengeditnya secara manual.