BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Program permainan (*game*) merupakan salah satu implementasi dari bidang ilmu komputer. Perkembangan permainan pada masa kini sudah sangat pesat dan telah menjadi mode tersendiri di dunia karena mayoritas pengguna komputer menghabiskan sebagian besar waktu mereka di depan komputer dalam program permainan [HAK03]. Salah satu algoritma yang digunakan untuk mengembangkan program permainan adalah algoritma berbasis pohon ruang pencarian *(searching algorithm).* Salah satu *game* yang menggunakan algoritma berbasis pohon ruang pencarian dalam menyelesaikan permainannya yaitu *slide puzzle*.

Slide puzzle merupakan permainan menyusun potongan gambar dengan aturan sebuah potongan hanya dapat dipindahkan dengan menggesernya ke ruang kosong *(blank tile). Puzzle* ini merupakan jenis *puzzle* yang memiliki tingkat kesulitan dalam menyelesaikan masalahnya sangat tinggi dibandingkan jenis *puzzle* lain. Umumnya orang yang memainkan *puzzle* ini butuh waktu lama dalam menyelesaikan permainannya. Hal ini disebabkan karena pada *slide puzzle* tidak ada informasi tambahan yang dimiliki untuk membantu melakukan pencarian solusi, sehingga saat proses penyusunan potongan-potongan *puzzle* terjadi susunan *puzzle* semula. Untuk menyelesaikan persoalan pada permainan ini dibutuhkan suatu algoritma pencarian efektif yang dapat diterapkan.

Dilihat dari karakteristik persoalan *slide puzzle*, *puzzle* ini membentuk ruang solusi yang diorganisasikan ke dalam struktur pohon dinamis. Struktur pohon dinamis sendiri dibangun dengan 2 metode traversal yaitu *Breadth First Search* (BFS) dan *Depth First Search* (DFS) [MUN04]. Untuk itu penulis menerapkan algoritma *Breadth First Search* dan *Depth First Search* dalam menyelesaikan permainan *slide puzzle*.

1.2 RUMUSAN MASALAH

Berdasarkan latar belakang permasalahan di atas, maka penulis menerapkan pencarian solusi *slide puzzle* dengan menggunakan algoritma *Breadth First Search* dan *Depth First Search*.

1.3 TUJUAN DAN MANFAAT

Tujuan dari tugas akhir ini adalah mengimplementasikan penerapan algoritma *Breadth First Search* dan *Depth First Search* sehingga dapat digunakan untuk mengoptimalkan waktu dalam menyelesaikan permainan *slide puzzle* yang umumnya tidak dapat digunakan jika penyelesaian permainan dilakukan secara manual (menggunakan orang sebagai pemain)

Manfaat dari tugas akhir ini adalah menambah ragam permainan *puzzle* yang telah ada sehingga dapat digunakan sebagai salah satu media alternatif untuk mengisi waktu senggang. Selain itu, permainan *puzzle* juga termasuk salah satu jenis permainan edukasi sehingga dapat digunakan untuk melatih kemampuan nalar dan logika seseorang

1.4 PEMBATASAN MASALAH

Penulis membatasi masalah pada tugas akhir ini sebagai berikut:

1) *Slide puzzle* dibuat dengan menggunakan sistem *Graphical User Interface* (GUI).
2) Bahasa pemrograman yang dipakai dalam pembuatan *slide puzzle* adalah Visual Basic 6.0 .
3) Data yang dimasukkan ke dalam *slide puzzle* berupa gambar dengan dimensi 200 x 250 pixel.
4) *Puzzle* berukuran 4 x 5 yang berisi 20 kotak potongan gambar.
5) Data gambar masukkan bertipe *.jpeg, *.gif, dan *.bmp.
6) Algoritma yang digunakan dalam pencarian solusi *slide puzzle* adalah BFS dan DFS.

1.5 METODOLOGI PENULISAN

Metodologi yang digunakan dalam penulisan tugas akhir ini adalah:

1) Tinjauan pustaka, mempelajari buku, artikel, dan situs yang terkait dengan pemrograman *game*.
2) Desain, tahapan ini dimulai dari perancangan arsitektur sistem, proses, *user interface*, dan interaksi sistem dengan pengguna.
3) Implementasi, desain yang telah dibuat kemudian diterapkan ke dalam kode program yang digunakan dengan berpedoman pada teori-teori dan data-data yang berkenaan dengan pemrograman *game*.
4) Tes, setelah selesai maka dilakukan tes untuk mengetahui bahwa sistem berjalan dengan baik.

1.6 SISTEMATIKA PENULISAN

Penulisan tugas akhir ini tersusun dalam 5 (lima) bab dengan sistematika penulisan sebagai berikut:

BAB I. PENDAHULUAN

Pendahuluan berisi latar belakang masalah, rumusan masalah, tujuan dan manfaat, metodologi penulisan, dan sistematika penulisan tugas akhir.

BAB II. DASAR TEORI

Dasar teori berisi beberapa teori yang mendasari penyusunan tugas akhir ini. Adapun yang dibahas dalam bab ini adalah teori yang berkaitan dengan penerapan *Breadth First Search* dan *Depth First Search* pada permainan *slide puzzle.*

BAB III. ANALISIS DAN DESAIN SISTEM

Pada bab ini diuraikan analisis dan desain perangkat lunak yang sedang dikerjakan.

BAB IV. IMPLEMENTASI DAN PEMBAHASAN

Berisi implementasi dan evaluasi terhadap algoritma *Breadth First Search* dan *Depth First Search* yang telah diterapkan dalam menyelesaikan permainan *slide puzzle*.

BAB V. PENUTUP

Bab penutup berisi kesimpulan dan saran.

BAB II

DASAR TEORI

2.1 KECERDASAN BUATAN

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.

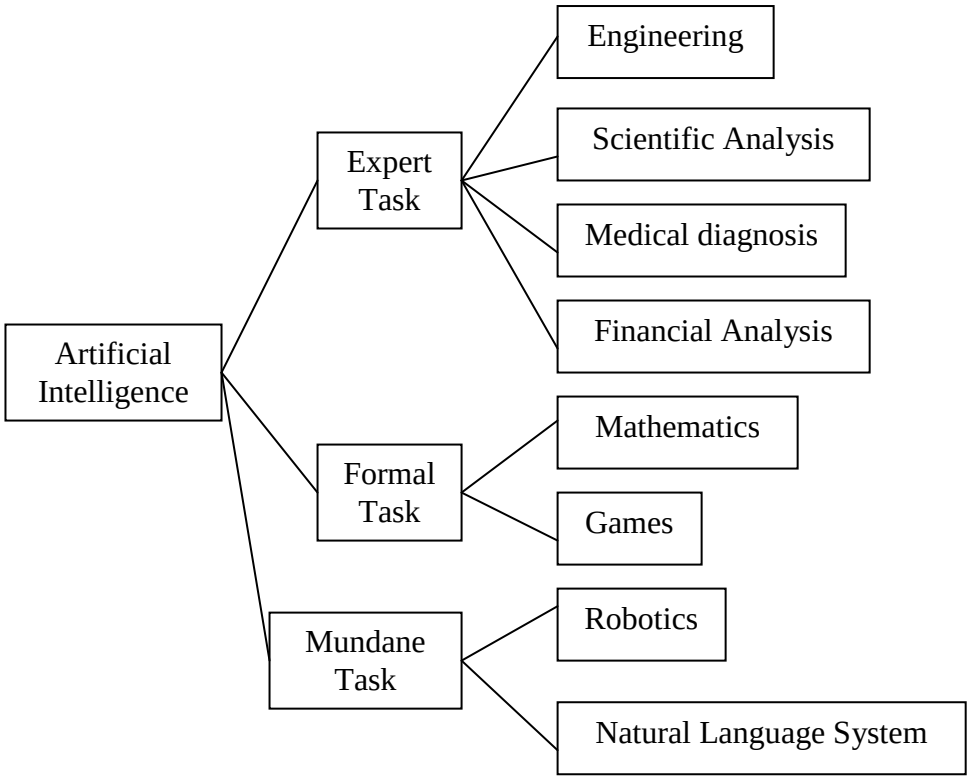
Ada dua hal yang sangat mendasar mengenai penelitian-penelitian AI, yaitu *knowledge representation* (representasi pengetahuan) dan *search* (pelacakan). Para peneliti AI terus mengembangkan berbagai jenis teknik baru dalam menangani sejumlah permasalahan yang tergolong ke dalam AI seperti *vision* dan 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 [DES06].



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

2.2 TEKNIK-TEKNIK DASAR PENCARIAN

Pencarian atau pelacakan merupakan salah satu teknik untuk menyelesaikan permasalahan AI. Keberhasilan suatu sistem salah satunya ditentukan oleh kesuksesan dalam pencarian dan pencocokan. Teknik dasar pencarian memberikan suatu kunci bagi banyak sejarah penyelesaian yang penting dalam bidang AI. Ada beberapa aplikasi yang menggunakan teknik pencarian ini, yaitu [KUS03]:

1) Papan *game* dan *puzzle* (tic-tac-toe, catur, menara hanoi).
2) Penjadwalan dan masalah *routing (travelling salesman problem).*
3) *Parsing* bahasa dan inteprestasinya (pencarian struktur dan arti).
4) Logika pemrograman (pencarian fakta dan implikasinya).
5) *Computer vision* dan pengenalan pola.
6) Sistem pakar bebasis kaidah *(rule based expert system).*

Pencarian adalah proses mencari solusi dari suatu permasalahan melalui sekumpulan kemungkinan ruang keadaan *(state space).* Ruang keadaan merupakan suatu ruang yang berisi semua keadaan yang mungkin. Kondisi suatu pencarian meliputi [MUN03]:

1) Keadaan sekarang atau awal.
2) Keadaan tujuan-solusi yang dijangkau dan perlu diperiksa apakah telah mencapai sasaran.
3) Biaya atau nilai yang diperoleh dari solusi.

Solusi merupakan suatu lintasan dari keadaan awal sampai keadaan tujuan. Secara umum, proses pencarian dapat dilakukan seperti berikut [RIY06].

1) Memeriksa keadaan sekarang atau awal.
2) Mengeksekusi aksi yang dibolehkan untuk memindahkan ke keadaan berikutnya.
3) Memeriksa jika keadaan baru merupakan solusinya. Jika tidak, keadaan baru tersebut menjadi keadaan sekarang dan proses ini diulangi sampai solusi ditemukan atau ruang keadaan habis terpakai.

2.2.1 Masalah Pencarian

Masalah pencarian merupakan proses pencarian solusi yang direncanakan, yang mencari lintasan dari keadaan sekarang sampai keadaan tujuan. Suatu masalah pencarian direpresentasikan sebagai graf berarah. Keadaan direpresentasikan sebagai simpul *(node),* sedangkan langkah yang dibolehkan atau aksi direpresentasikan dengan busur *(arc).* Dengan demikian, secara khusus masalah pencarian didefinisikan sebagai [MUN03]:

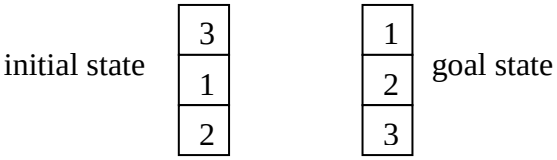
1) *State space* (ruang keadaan).
2) *Start node* (permukaan simpul).
3) Kondisi tujuan dan uji untuk mengecek apakah kondisi tujuan ditemukan atau tidak.
4) Kaidah yang memberikan bagaimana mengubah keadaan.

2.2.2 Contoh Pencarian

Misalkan ada tiga kotak 1, 2, 3 pada sebuah papan. Sebuah kotak dapat dipindahkan jika tidak ada kotak lain di atasnya dan hanya ada satu kotak yang boleh dipindahkan. Ada dua kemungkinan pemindahannya, yaitu:

1) Pindahkan sebuah kotak ke atas papan.
2) Pindahkan sebuah kotak ke atas kotak lainnya.

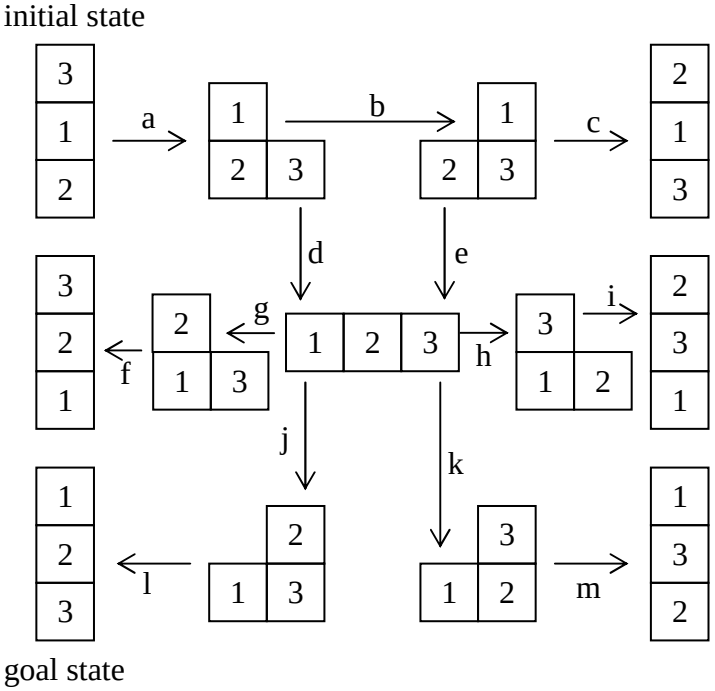
Masalah muncul jika diketahui keadaan awalnya (*initial state* atau *current state*) dan tujuan akhirnya (*goal state* atau *final state*) seperti pada gambar 2.2.



Gambar 2.2. Contoh permainan yang menerapkan pencarian.

Pada gambar 2.3 dapat dilihat bahwa ruang keadaan tersebut memiliki 13 elemen atau *node*, dengan perpindahan sebagai berikut:

1) Perpindahan kotak 1 ke atas papan untuk lintasan d dan e.
2) Perpindahan kotak 3 ke atas papan untuk lintasan a.
3) Perpindahan kotak 1 ke atas kotak 2 untuk lintasan l.
4) Perpindahan kotak 1 ke atas kotak 3 untuk lintasan b dan m.
5) Perpindahan kotak 2 ke atas kotak 1 untuk lintasan c dan g.
6) Perpindahan kotak 2 ke atas kotak 3 untuk lintasan i dan j.
7) Perpindahan kotak 3 ke atas kotak 1 untuk lintasan h.
8) Perpindahan kotak 3 ke atas kotak 2 untuk lintasan f dan k.



Gambar 2.3. Ruang keadaan

Penyelesaian untuk masalah permainan pada gambar 2.3 adalah anggota kumpulan semua lintasan dari keadaan awal hingga tujuan yang lintasannya ditandai dengan huruf a, d, j, dan l

Secara umum, algoritma pencarian dapat dituliskan seperti berikut [DES06]:

```
Function Gsearch (Problem, QueueingFn): Solution |
Failure
 Var nodes : structure;
 Begin
  While
    Begin
     Node := remove_front_nodes(nodes);
     IF Goal_test(problem,STATE(node)) succeeds then
      Solution := Node
     Else
      Nodes := QueueningFn(node, OPERATOR(problem));
    End;
  EndWhile
 End;
```

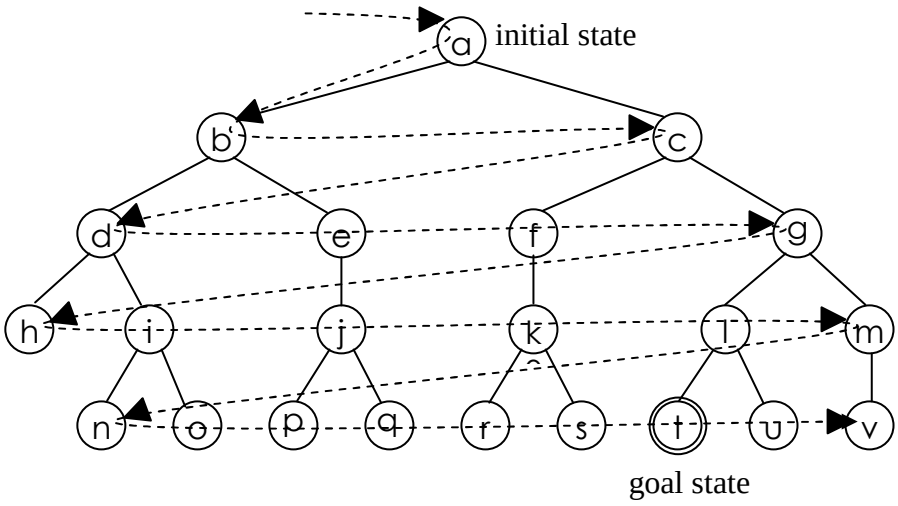
2.3 STRATEGI PENCARIAN MENDALAM

Pencarian boleh jadi merupakan hasil dari suatu solusi ruang keadaan yang mungkin telah terkunjungi semua, tetapi tanpa penyelesaian. Pencarian yang mendalam *(Exchausting Search Strategy)* mungkin dilakukan dengan menggunakan strategi *Breadth First Search* atau *Depth First Search (Iterative Deepening).* Kedua pencarian ini merupakan pencarian buta *(blind search).*

2.3.1 *Breadth First Search*

Prosedur *Breadth First Search* merupakan pencarian yang dilakukan dengan mengunjungi tiap-tiap *node* secara sistematis pada setiap level hingga keadaan tujuan *(goal state)* ditemukan. Atau

dengan kata lain, penulusuran yang dilakukan adalah dengan mengunjungi tiap-tiap *node* pada level yang sama hingga ditemukan *goal state*-nya [MUN03]. Untuk lebih jelasnya, perhatikan ilustrasi dari *Breadth First Search* pada gambar 2.4.

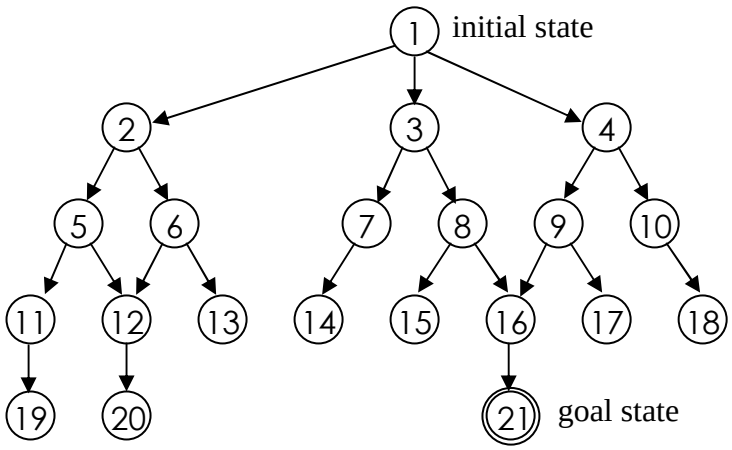


Gambar 2.4. Teknik pencarian *Breadth First Search*

Pengimplementasian *Breadth First Search* dapat ditelusuri dengan menggunakan daftar *(list)*, *open*, dan *closed*, untuk menelusuri gerakan pencarian di dalam ruang keadaan. Prosedur untuk *Breadth First Search* dapat dituliskan sebagai berikut:

```
Procedure breadth first search
 Begin
  Open := [start]
  Closed := []
  While open = [] do
   Begin
    Remove leftmost state from open, call it x
    If x is goal then return SUCCESS
    Else
     Begin
      Generate children of x;
      Put x on closed;
      discard children of x is already on open or
      closed;
      put remaining children on right end of open;
     End;
    End
    return FAIL
 End.
```

Untuk lebih jelasnya, simak contoh graf pada gambar 2.5



Gambar 2.5. Contoh graf yang akan ditelusuri.

Pada gambar 2.5, *state* 21 merupakan tujuannya *(goal)* sehingga bila ditelusuri menggunakan prosedur *Breadth First Search*, diperoleh:

1) Open = [1]; closed = [].
2) Open = [2, 3, 4]; closed = [1].
3) Open = [3, 4, 5, 6]; closd = [2, 1].
4) Open = [4, 5, 6, 7, 8]; closed = [3, 2, 1].
5) Open = [5, 6, 7, 8, 9, 10]; closed = [4, 3, 2, 1].
6) Open = [6, 7, 8, 9, 10, 11, 12]; closed = [5, 4, 3, 2, 1].
7) Open = [7, 8, 9, 10, 11, 12, 13] (karena 12 telah di-open); closed = [6, 5, 4, 3, 2, 1].
8) Open = [8, 9, 10, 11, 12, 13, 14]; closed = [7, 6, 5, 4, 3, 2, 1].
9) Dan seterusnya sampai *state* 21 diperoleh atau open = [].

Ada beberapa keuntungan menggunakan algoritma *Breadth First Search* ini, di antaranya adalah tidak akan menemui jalan buntu dan jika ada satu solusi maka *Breadth First Search* akan

menemukannya, dan jika ada lebih dari satu solusi maka solusi minimum akan ditemukan.

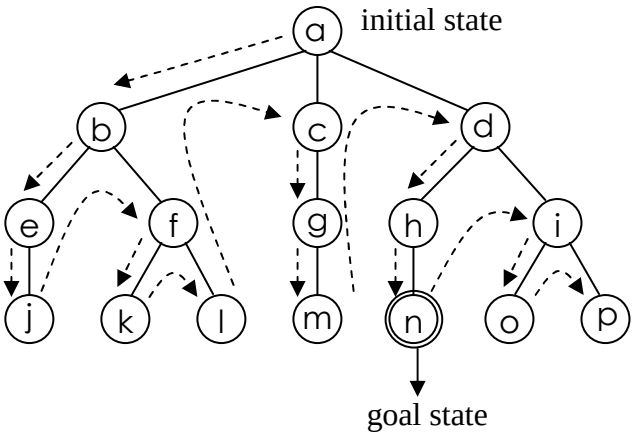
Namun ada tiga persoalan utama berkenaan dengan *Breadth First Search* ini yaitu:

1) Membutuhkan memori yang lebih besar, karena menyimpan semua *node* dalam satu pohon.
2) Membutuhkan sejumlah besar pekerjaan, khususnya jika lintasan solusi terpendek cukup panjang, karena jumlah *node* yang perlu diperiksa bertambah secara eksponensial terhadap panjang lintasan.
3) Tidak relevannya operator akan menambah jumlah *node* yang harus diperiksa.

Oleh karena proses *Breadth First Search* mengamati *node* di setiap level graf sebelum bergerak menuju ruang yang lebih dalam maka mula-mula semua keadaan akan dicapai lewat lintasan yang terpendek dari keadaan awal. Oleh sebab itu, proses ini menjamin ditemukannya lintasan terpendek dari keadaan awal ke keadaan tujuan (akhir). Lebih jauh karena mula-mula semua keadaan ditemukan melalui lintasan terpendek sehingga setiap keadaan yang ditemui pada kali kedua didapati pada sepanjang sebuah lintasan yang sama atau lebih panjang. Kemudian, jika tidak ada kesempatan ditemukannya keadaan yang identik pada sepanjang lintasan yang lebih baik maka algoritma akan menghapusnya [DES06].

2.3.2 *Depth First Search*

Pencarian dengan metode ini dilakukan dari *node* awal secara mendalam hingga yang paling akhir *(dead-end)* atau sampai ditemukan. Dengan kata lain, simpul cabang atau anak yang terlebih dahulu dikunjungi. Sebagai ilustrasinya dapat dilihat gambar 2.6.



Gambar 2.6. Teknik pencarian *Depth First Search*

Berdasarkan gambar 2.6, proses pencarian dilakukan dengan mengunjungi cabang terlebih dahulu hingga tiba di simpul terakhir. Jika tujuan yang diinginkan belum tercapai maka pencarian dilanjutkan ke cabang sebelumnya, turun ke bawah jika memang masih ada cabangnya. Begitu seterusnya hingga diperoleh tujuan akhir *(goal)*.

Depth First Search, seperti halnya *Breadth First Search*, juga memiliki kelebihan di antaranya adalah cepat mencapai kedalaman ruang pencarian. Jika diketahui bahwa lintasan solusi permasalahan akan panjang maka *Depth First Search* tidak akan memboroskan waktu untuk melakukan sejumlah besar keadaan dangkal dalam permasalahan graf. *Depth First Search* jauh lebih efisien untuk ruang

pencarian dengan banyak cabang karena tidak perlu mengeksekusi semua simpul pada suatu level tertentu pada daftar *open.* Selain itu, *Depth First Search* memerlukan memori yang relatif kecil karena banyak *node* pada lintasan yang aktif saja yang disimpan [MUN03].

Selain kelebihan, *Depth First Search* juga memiliki kelemahan di antaranya adalah memungkinkan tidak ditemukannya tujuan yang diharapkan dan hanya akan mendapatkan satu solusi pada setiap pencarian.

Prosedur *Depth First Search* dapat diimplementasikan dengan melakukan modifikasi proses *Breadth First Search* menjadi:

```
Procedure depth first search
 Begin
  Open := [start]
  Closed := []
  While open = [] do
   Begin
    Remove leftmost state from open, call it x
    If x is goal then return SUCCESS
    Else
     Begin
      Generate children of x;
      Put x on closed;
      discard children of x is already on open or
      closed;
      put remaining children on left end of open;
     End;
    End
    return FAIL
   End.
```

Untuk lebih jelasnya, dapat dilihat aplikasi algoritma tersebut untuk gambar 2.5 dengan *state* 21 diasumsikan sebagai tujuannya. Langkah-langkah penelusuran tersebut adalah:

1) Open = [1]; closed = [].

2) Open = [2, 3, 4]; closed = [1].

3) Open = [5, 6, 3, 4]; closed = [2, 1].

4) Open = [11, 12, 6, 3, 4]; closed = [5, 2, 1].

5) Open = [19, 12, 6, 3, 4]; closed = [11, 5, 2, 1].

6) Open = [12, 6, 3, 4]; closed = [19, 11, 5, 2, 1].

7) Open = [20, 6, 3, 4]; closed = [12, 19, 11, 5, 2, 1].

8) Open = [6, 3, 4]; closed = [20, 12, 19, 11, 5, 2, 1].

9) Open = [13, 3, 4] (karena 12 telah di-closed); closed = [6, 20, 12, 19, 11, 5, 2, 1].

10) Open = [3, 4]; closed = [13, 6, 20, 12, 19, 11, 5, 2, 1].

11) Open = [7, 8, 4]; closed = [3, 13, 6, 20, 12, 19, 11, 5, 2, 1].

12) Dan seterusnya sampai *state* 21 diperoleh atau open = [].

Penentuan teknik pencarian yang sesuai atau yang tepat untuk sebuah kasus khusus penganalisaan suatu ruang permasalahan, menjadi sangat penting dan biasanya dengan melakukan konsultasi dengan para pakar di bidangnya untuk mendapatkan dan mengetahui tingkah laku ruang permasalahan tersebut [DES06].

2.4 PERMAINAN KOMPUTER *(COMPUTER GAME)*

Permainan komputer pertama kali dibuat pada tahun 1966 oleh Ralph Baer bersama 500 insinyur dan teknisi. Permainan yang diproduksinya hanya dapat dimainkan dengan komputer seharga US$ 40.000.

Pada tahun 1965 militer datang kepada Baer dan meminta simulasi komputer yang dapat membantu pasukan untuk belajar strategi dan mengukur kemampuan refleks. Proyek ini dikerjakan dengan tingkat pengamanan yang tinggi di tengah situasi perang dingin.

Setelah sebulan bekerja keras, Baer berhasil menampilkan dua titik putih yang berkejar-kejaran di layar. Hal ini membuat militer kagum dan memberikan dana yang jauh lebih besar lagi sehingga ia dapat menyewa asisten lebih banyak. Tim ini berhasil membuat permainan antara papan dan bola yang pada akhir tahun 1966 dipresentasikan di depan pejabat Pentagon. Ternyata hal ini tidak membuat Pentagon tertarik sehingga Baer kemudian berusaha memperoleh izin agar dapat memproduksi mesin permainan secara komersial.

Tahun 1970, Bill Endens yang tergabung dalam Magnavox mencoba meyakinkan eksekutif Magnavox untuk memberikan kesempatan pada Baer dan mesin permainannya. Hasilnya adalah munculnya video *game* komersial pertama, yaitu *Magnavox Odyssey*, yang terjual lebih dari 100.000 unit dengan harga US$ 100 per unit.

Nolan Bushnell, pada tanggal 27 Juni 1972, mendirikan perusahaan Atari dan membuat *game Arcade Pong*. Mesin *Arcade Pong* pertama kali ditempatkan dalam bar Andy Capp’s di Sunnyvale. Hari pertama, suara *pong* menarik perhatian pengunjung bar dan hampir tiap orang di sana memainkannya. Hari kedua, orang-orang telah berbaris di depan Andy Capp’s pada jam 10 pagi untuk bermain *pong*. Sekitar jam 10 malam, *game* ini tiba-tiba mati disebabkan kontainer koin di mesin kelebihan muatan dan mengenai sistem elektronik. Nolan Bushnell membuatnya dengan modal US$ 500 dan empat tahun kemudian perusahaan itu dijualnya seharga US$ 28 juta.

Pada tahun 1988, *Nintendo* (sebuah perusahaan Jepang yang awalnya memproduksi mesin fotokopi) dengan *video game system*-nya telah mencapai omset kira-kira US$ 1,7 milyar dan menjadi nomor satu di antara perusahaan yang memproduksi mainan di Amerika (TIME edisi 19 Desember 1988).

Kini permainan komputer telah berkembang sedemikian pesat seiring perkembangan *hardware* komputer. Diawali dengan kehadiran penyerbu ruang angkasa *Space Invader, penggali Digger, PacMan,* dan *Alley Cat,* program permainan telah berkembang menjadi lebih kompleks dengan tampilan grafik tiga dimensi (3D) yang luar biasa. Program permainan saat ini mampu membuat pemainnya merasakan bagaimana berperang menghadapi teroris dalam *Counter Strike* atau menjadikan pemainnya sebagai manajer sebuah klub sepak bola terkemuka dalam *Championship Manager*. Para perancang program permainan berlomba-lomba mewujudkan fantasi untuk memuaskan para pemain. *Programmer* permainan canggih saat ini telah memiliki kemampuan setara dengan *programmer* pengolah kata seperti *word* ataupun *spreadsheet* [HAK03].

2.5 MICROSOFT VISUAL BASIC 6.0

Microsoft Visual Basic 6.0 adalah bahasa pemrograman yang bekerja dalam lingkup MS-*Windows*. Visual Basic berasal dari bahasa pemrograman yang populer yang disebut Basic *(Beginner's All Purpose Symbol Instruction Code)*. Visual Basic 6.0 memiliki kelebihan-kelebihan antara lain kompiler (proses *compile*) dapat dilakukan dengan cepat, mendukung

kontrol data objek yang baru, mendukung berbagai macam *database*, pembuatan program permainan yang lebih mudah, dan mendukung pengaksesan terhadap internet. Visual Basic 6.0 menyediakan tiga macam *interface* yang bisa digunakan untuk merancang aplikasi sesuai dengan kebutuhan. *Interface* tersebut berupa *Multi Document Interfac* (MDI), *Single Document Interface* (SDI), dan *Explorer Document Interface* (EDI) [HAS09].

Fasilitas yang disediakan juga lebih lengkap sehingga bisa memenuhi selera *programmer* yang pada akhirnya akan meningkatkan produktivitas kerja. Fasilitas-fasilitas dalam Microsoft Visual Basic 6.0 antara lain:

a) *Menu* merupakan daftar perintah-perintah yang dikelompokkan dalam kriteria tertentu yang berfungsi untuk melaksanakan sebuah perintah.
b) *Toolbar* merupakan kumpulan tombol yang dapat melakukan sebuah perintah dengan cepat.
c) *Form* adalah tempat untuk meletakkan objek-objek yang digunakan untuk melaksanakan perintah yang diberikan.
d) *Window code* adalah jendela tempat menuliskan kode program.
e) *Toolbox* adalah kumpulan objek yang digunakan untuk kontrol pada sebuah program.
f) *Project explorer* digunakan untuk melihat bagian-bagian proyek pembuatan aplikasi.
g) *Window properties* menampilkan semua properti dari objek yang digunakan.

h) *Windows form layout* digunakan untuk mengatur letak *form* pada layar monitor [DAR03].

2.5.1 Grafik

Grafik merupakan salah satu bagian yang sangat penting dalam program permainan. Visual Basic menangani grafik secara baik sehingga memungkinkan untuk membuat program permainan yang memiliki gambar grafis yang menarik.

Sistem koordinat pada Visual Basic dikenal dengan sistem koordinat dua dimensi.yang sepenuhnya didukung oleh *Graphics Device Interface* (GDI) dimana GDI merupakan kumpulan fungsi yang digunakan untuk mengakses atau menggambar ke piranti tertentu seperti monitor, *printer*, dan sebagainya. Sistem koordinat pada pemrograman *Windows* terdiri dari sistem koordinat fisik.

Sistem koordinat fisik *(Device Coordinate System)* merupakan koordinat yang dipakai oleh peralatan fisik. Layar monitor memiliki titik pusat koordinat fisik di kiri atas dengan sumbu x positif berasal dari pusat menuju ke kanan dan sumbu y positif berasal dari pusat menuju ke bawah. Mengingat seluruh objek dan komponen pada Visual Basic ditempatkan pada *form*, untuk menentukan koordinatnya digunakan penulisan (x, y). Standar awal dari koordinat adalah (0, 0) yaitu pada titik paling kiri-atas dari *form* atau objek yang akan didefinisikan koordinatnya. Misalnya, jika menjalankan

Windows dengan resolusi 200 x 250, maka koordinat kiri-atas adalah (0, 0) dan kanan-bawah adalah (199, 249).

Satuan yang dipakai dalam koordinat Visual Basic secara standar adalah *twips*, dimana 1 *twip* = 1/1440 *inci*. Keuntungan satuan ini adalah dalam resolusi apapun, suatu garis dengan panjang 1440 *twip* akan tetap ditampilkan 1 *inci*. Satuan ini menyulitkan dalam perhitungan, sehingga digunakan satuan *pixel* yang lebih umum dikenal. *Pixel* merupakan titik dalam monitor, misal pada resolusi 1024x768 *pixel*, maka pada koordinat x akan terdapat 1024 titik. Satuan *pixel* sangat tergantung pada monitor. Sebuah garis dengan panjang 800 *pixel* akan terlihat panjang pada pada resolusi 800x600 *pixel*, tetapi akan terlihat pendek (setengah panjang monitor) pada resolusi 1024x768 *pixel.* Satuan ini dapat diubah pada *properties form* di bagian *scalemode*: 3-*pixel* [HAK03].

Visual Basic memiliki beberapa kontrol yang dapat digunakan secara khusus untuk menangani grafik. Salah satunya adalah kontrol *picturebox* yang dapat ditambahkan dengan mengklik ikon *picturebox* pada jendela *toolbox*. Dibandingkan kontrol *image* yang juga digunakan untuk menangani grafik, kontrol *picturebox* memiliki properti yang lebih banyak. Perbedaan antara *image* dengan *picturebox* yaitu [HAS09]:

1) Bingkai dari objek *picturebox* yang terpasang di *form* tidak transparan (garis putus-putus) seperti *image.*

2) Properti *stretch* seperti pada kontrol *image* tidak ada tetapi pada kontrol *picturebox* diganti dengan a*utosize.*

3) Kontrol *picturebox* dapat menangkap jejak *mouse* sehingga bisa menulis dan menggambar dengan *mouse* pada objek ini. Seluruh kontrol yang ada di dalam kontrol *picturebox* dapat dengan mudah dipindahkan ke bagian lain dari *form*.

4) Kontrol *picturebox* dapat menampung objek-objek lain dan menjadikannya satu kelompok.

2.5.2 Kontrol Permainan

Keyboard merupakan peralatan *input* utama yang harus dimiliki oleh setiap *personal computer*. Selain itu *keyboard* mempunyai banyak tombol yang dapat dikombinasikan, misalnya CTRL+V. Visual Basic telah menyediakan *event* standar untuk mendeteksi adanya masukan dari *keyboard*, yaitu *keypress, keydown,* dan *keyup.*

Mouse berperan sama pentingnya dengan *keyboard* karena kemudahan penggunaannya. Hampir semua program permainan saat ini menggunakan *mouse* untuk kemudahan dan keleluasaan pengendalian. Seperti halnya *keyboard*, Visual Basic juga telah memiliki *event* standar untuk mendeteksi masukan melewati *mouse*, misalnya dengan *event mouseover*, *click*, *dblclick*, dan lain-lain [HAK03].