BAB II
TINJAU PUSTAKA

2.1 *Artificial Intelligence (AI)*

2.1.1 Definisi AI

Kecerdasan buatan sering disebut juga dengan *Artificial Intelligence* (AI). AI merupakan salah satu bagian ilmu komputer yang mempelajari tentang bagaimana caranya agar komputer dapat melakukan pekerjaan seperti yang dapat dilakukan oleh manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan zaman, teknologi komputer semakin ditingkatkan dan peran komputer semakin mendominasi kehidupan umat manusia. Komputer tidak lagi hanya digunakan sebagai alat hitung, melainkan komputer diharapkan dapat diberdayakan untuk mengerjakan hal-hal yang dapat memudahkan pekerjaan manusia.

Manusia bisa menjadi pandai dalam menyelesaikan segala permasalahan di dunia ini karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan dapat diperoleh dari proses belajar. Semakin banyak bekal pengetahuan yang dimiliki oleh seseorang tentu saja diharapkan akan lebih baik dalam menyelesaikan permasalahan. Namun bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran, mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang mereka miliki. Tanpa memiliki kemampuan menalar yang baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik, demikian pula dengan kemampuan menalar yang sangat baik, namun tanpa bekal pengetahuan dan pengalaman yang memadai, manusia juga tidak akan bisa menyelesaikan masalah dengan baik.

Agar komputer bisa bertindak seperti apa yang dapat dilakukan manusia, maka komputer juga harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Untuk itu AI akan mencoba untuk memberikan beberapa metode untuk membekali komputer dengan kedua komponen tersebut agar komputer bisa

menjadi mesin yang pintar. (Kusumadewi, 2003, 1-2)

Beberapa definisi atau pengertian dari AI dapat dilihat pada tabel 2.1 berikut.

Tabel 2.1 Beberapa definisi AI

No.	Definisi AI
1.	"*The study of mental faculties through the use of computasional models*" (Chaniak and mcDermott, 1985)
2.	"*The study of the computation that make it possible to perceive, reason and act*" (Winston, 1992)
3.	"*The art of creating machine that performs functions that require intelligence when performed by people*" (Kurzweil, 1990)
4.	"*The study of how to make computers to do things at which, at the moment, people are better*" (Rich and Knight, 1991)
5.	"*A field of study that seeks to explain and emulate intelligent behaviour in term of computational processes*" (Schalkoff, 1990)
6.	"*The branch of computer science that is concerned with the automation of intelligent behaviour*" (Luger and Stubblefield, 1993)
7.	*System that thinks like humans*.
8.	*System that acts like humans*.
9.	*System that thinks rationally*.
10.	*System that acts rationally*.

(Sumber : Desiani dan Arhami, 2005, 1-2)

Dari beberapa definisi pada tabel 2.1, definisi AI dapat dibagi dalam empat kategori, yaitu:

1. Sistem yang dapat berpikir seperti manusia "*Thinking humanly*"
2. Sistem yang dapat bertingkah laku seperti manusia "*Acting humanly*"
3. Sistem yang dapat berpikir secara rasional "*Thinking rationally*"
4. Sistem yang dapat bertingkah laku secara rasional "*Acting rationally*"

Secara historis, keempat pendekatan tersebut telah dilakukan. Pendekatan manusia haruslah merupakan suatu ilmu empiris, termasuk hipotesa dan konfirmasi percobaan dan pendekatan rasional meliputi kombinasi dari matematika dan rekayasa. (Desiani dan Arhami, 2005, 3)

2.1.2 Sejarah AI

AI atau kecerdasan buatan termasuk bidang ilmu yang relatif muda. Pada tahun 1950-an para ilmuan dan peneliti mulai memikirkan bagaimana caranya agar mesin dapat melakukan pekerjaannya seperti yang bisa dikerjakan oleh manusia. Alan Turing, seorang matematikawan dari Inggris pertama kali mengusulkan adanya tes untuk melihat bisa tidaknya sebuah mesin dikatakan cerdas. Hasil tes tersebut kemudian dikenal dengan Turing Test, dimana si mesin tersebut menyamar seolah-oleh sebagai seseorang di dalam suatu permainan yang mampu memberikan respon terhadap serangkaian pertanyaan yang diajukan. Turing beranggapan bahwa, jika mesin dapat membuat seseorang percaya bahwa dirinya mampu berkomunikasi dengan orang lain, maka dapat dikatakan bahwa mesin tersebut cerdas, layaknya seorang manusia.

AI itu sendiri dimunculkan oleh seorang profesor dari *Massachusetts Institute of Technology* yang bernama John McCarthy pada tahun 1956 pada *Darmouth Conference* yang dihadiri oleh para peneliti AI. Pada konferensi tersebut juga didefinisikan tujuan utama dari kecerdasan buatan, yaitu mengetahui dan memodelkan proses-proses berpikir manusia dan mendesain mesin agar dapat menirukan kelakuan mesin tersebut.

Beberapa program AI yang mulai dibuat tahun 1956-1966, antara lain:

1. *Logic Theorist*, diperkenalkan pada *Dartmouth Conference*, program ini dapat membuktikan teorema-teorema matematika.
2. *Sad Sam*, diprogram oleh Robert K. Lindsay (1960). Program ini dapat mengetahui kalimat-kalimat sederhana yang ditulis dalam bahasa Inggris dan mampu memberikan jawaban dari fakta-fakta yang didengar dalam sebuah percakapan.
3. ELIZA, diprogram oleh Joseph Weizenbaum (1967). Program ini mampu melakukan terapi terhadap pasien dengan memberikan beberapa pertanyaan dan jawaban. (Kusumadewi, 2003, 5-6)

2.1.3 AI dari Berbagai Sudut Pandang

AI dapat dilihat dari berbagai sudut pandang, antara lain:

1. Sudut pandang kecerdasan.

   AI akan membuat mesin menjadi "cerdas" (mampu berbuat seperti apa yang dilakukan oleh manusia).

2. Sudut pandang penelitian.

   AI adalah suatu studi bagaimana membuat agar komputer dapat melakukan sesuatu sebaik yang dikerjakan oleh manusia. *Domain* yang sering dibahas oleh para peneliti meliputi:

a. Mundane task

   1) Persepsi (*vision & speech*).
   2) Bahasa alami (*understanding*, *generation* & *translation*).
   3) Pemikiran yang bersifat *commonsense*.
   4) Robot *control*.

b. Formal task

   1) Matematika (geometri, logika, kalkulus integral, pembuktian).
   2) Permainan/*games*.

c. Expert task

   1) Analisis finansial.
   2) Analisis medikal.
   3) Analisis ilmu pengetahuan.
   4) Rekayasa (desain, pencarian kegagalan, perencanaan manufaktur)

3. Sudut pandang bisnis.

   AI adalah kumpulan peralatan yang sangat *powerful* dan metodologis dalam menyelesaikan masalah-masalah bisnis.

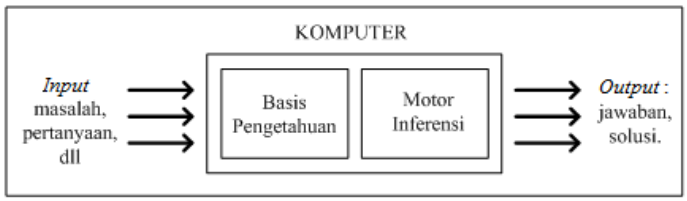
4. Sudut pandang pemograman.

   AI meliputi studi tentang pemograman simbolik, penyelesaian masalah (*problem solving*) dan pencarian (*searching*). (Kusumadewi, 2003, 1-2)

Untuk menciptakan aplikasi AI ada 2 bagian utama yang sangat dibutuhkan, yaitu:

1. Basis Pengetahuan (*Knowledge Base*), berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan yang lainnya.

2. Motor Inferensi (*Inference Engine*), yaitu kemampuan menarik kesimpulan berdasarkan pengalaman.



Gambar 2.1 Penerapan konsep AI di dalam komputer

(Sumber : Kusumadewi, 2003, 3)

2.1.4 Ruang Lingkup AI pada Aplikasi Komersial

Makin pesatnya perkembangan teknologi menyebabkan adanya perkembangan dan perluasan lingkup yang membutuhkan kehadiran AI. Karakteristik ‘cerdas’ sudah mulai dibutuhkan di berbagai disiplin ilmu dan teknologi. AI tidak hanya dominan di bidang ilmu komputer (informatika), namun juga sudah merambah di berbagai disiplin ilmu yang lain. Irisan antara psikologi dan AI melahirkan sebuah area yang dikenal dengan nama *cognition* & *psycolinguistics*. Irisan antara teknik elektro dengan AI melahirkan berbagai ilmu, seperti: pengolahan citra, teori kendali, pengenalan pola dan robotika. Dewasa ini, AI juga memberikan konstribusi yang cukup besar di bidang manajemen. Adanya sistem pendukung keputusan dan Sistem Informasi Manajemen juga tidak terlepas dari andil AI.

Adanya irisan penggunaan AI di berbagai disiplin ilmu tersebut menyebabkan cukup rumitnya untuk mengklasifikasikan AI menurut disiplin ilmu yang menggunakannya. Untuk memudahkan hal tersebut, maka pengklasifikasian lingkup AI didasarkan pada *output* yang diberikan, yaitu pada aplikasi komersial (meskipun sebenarnya AI itu sendiri bukan merupakan medan komersial).

Lingkup utama dalam kecerdasan buatan adalah:

1. Sistem Pakar (*Expert System*). Disini, komputer digunakan sebagai sarana untuk menyimpan pengetahuan para pakar. Dengan demikian, komputer akan memiliki keahlian untuk menyelesaikan permasalahan dengan meniru keahlian yang dimiliki oleh pakar.

2. Pengolahan Bahasa Alami (*Natural Language Processing*). Dengan pengolahan bahasa alami ini diharapkan *user* dapat berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari.
3. Pengenalan Ucapan (*Speech Recognition*). Melalui pengenalan ucapan diharapkan manusia dapat berkomunikasi dengan komputer menggunakan suara.
4. Robotika & Sistem Sensor (*Robotics & Sensory Systems*).
5. *Computer Vision*, mencoba untuk dapat menginterpretasikan gambar atau objek-objek tampak melalui komputer.
6. *Intelligent Computer-aided Instruction*. Komputer dapat digunakan sebagai *tutor* yang dapat melatih dan mengajar.
7. *Game Playing*. Permainan dapat menghasilkan sejumlah besar pencarian ruang. Hal ini cukup besar dan kompleks dan membutuhkan teknik pencarian untuk menentukan alternatif pengeksplorasian ruang permasalahan. Untungnya, setiap permainan memiliki aturan main. Hal ini mempermudah upaya menghasilkan ruang pencarian. (Kusumadewi, 2003, 6-7)

Beberapa karakteristik yang ada pada sistem yang menggunakan AI adalah pemogramannya yang cenderung bersifat simbolik ketimbang algoritmik, bisa mengakomodasi *input* yang tidak lengkap, bisa melakukan inferensi dan adanya pemisahan antara kontrol dengan pengetahuan. Namun, seiring dengan perkembangan teknologi, muncul beberapa teknologi yang juga bertujuan untuk membuat agar komputer menjadi cerdas sehingga dapat menirukan kerja manusia sehari-hari.

Teknologi ini juga mampu mengakomodasi adanya ketidakpastian dan ketidaktepatan data *input*. Dengan didasari pada teori himpunan, maka pada tahun 1965 muncul Logika Fuzzy. Kemudian pada tahun 1975, John Holland mengatakan bahwa setiap *problem* berbentuk adaptasi (alami maupun buatan) secara umum dapat diformulasikan dalam terminologi genetika. Algoritma genetika ini merupakan simulasi proses evolusi Darwin dan operasi genetika atas kromosom. (Sumber : Kusumadewi, 2003, 7)

2.1.5 Perbedaan antara AI dan Kecerdasan Alami

Dibandingkan dengan kecerdasan yang dimiliki oleh manusia (kecerdasan alami), AI memiliki beberapa keuntungan antara lain:

1. AI lebih bersifat permanen.

   Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. AI tidak akan berubah sepanjang sistem komputer dan program tidak diubah.

2. AI lebih mudah diduplikasi dan disebarkan.

   Mentransfer pengetahuan manusia dari satu orang ke orang lain membutuhkan proses yang sangat lama, dan keahlian itu juga tidak akan pernah dapat diduplikasi dengan lengkap. Oleh karena itu, jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebut dapat disalin dari komputer tersebut dan dapat dipindahkan dengan mudah ke komputer yang lain.

3. AI lebih murah dibanding dengan kecerdasan alami.

   Menyediakan layanan komputer akan lebih mudah dan lebih murah dibandingkan dengan harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.

4. AI bersifat konsisten.

   Hal ini disebabkan karena AI adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami akan senantiasa berubah-ubah.

5. AI dapat didokumentasi.

   Keputusan yang dibuat oleh komputer dapat didokumentasi dengan mudah dengan cara melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.

6. AI lebih cepat.

   AI dapat mengerjakan pekerjaan jauh lebih cepat dibanding dengan kecerdasan alami.

7. AI lebih baik.

   AI dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami. (Kusumadewi, 2003, 3-4)

Sedangkan keuntungan dari kecerdasan alami adalah:

1. Kreatif.

   Kemampuan untuk menambah ataupun memenuhi pengetahuan itu sangat melekat pada jiwa manusia. Pada kecerdasan buatan, untuk menambah pengetahuan harus dilakukan melalui sistem yang dibangun.
2. Kecerdasan alami memungkinkan orang untuk menggunakan pengalaman secara langsung. Sedangkan AI buatan harus berkerja dengan input-input simbolik.
3. Pemikiran manusia dapat digunakan secara luas, sedangkan AI sangat terbatas. (Kusumadewi, 2003, 4)

2.1.6 Perbedaan antara AI dan Komputasi Konvensional

Data yang diproses oleh komputer konvensional dapat dilihat pada tabel 2.2 berikut ini.

Tabel 2.2 Pemrosesan data komputer konvensional

Proses	Tugas
Kalkulasi	Mengerjakan operasi-operasi matematis seperti: +, -, x, : atau mencari akar persamaan, menyelesaikan rumus/persamaan.
Logika	Mengerjakan operasi logika seperti: and, or, invert.
Penyimpanan	Menyimpan data dan gambar pada *file*.
Retrieve	Mengakses data yang disimpan pada *file*.
Translate	Mengonversi data dari satu bentuk ke bentuk yang lain.
Sort	Memeriksa data dan menampilkan dalam urutan yang diinginkan.
Edit	Melakukan perubahan, penambahan dan penghapusan pada data.
Monitor	Mengamati event ekseternal dan internal serta melakukan tindakan jika kondisi tertentu tercapai.
Kontrol	Memberikan perintah atau mengendalikan peralatan luar.

(Sumber : Desiani dan Arhami, 2005, 19)

Perbandingan antara AI dengan pemograman konvensional dapat dilihat pada tabel 2.3 berikut ini.

Tabel 2.3 Perbandingan AI dengan Pemograman Konvensional

Dimensi	AI	Pemograman Konvensional
Processing	Simbolik	Algoritmik
Input	Tidak harus lengkap	Harus lengkap
Search	Heuristik	Algoritmik
Explanation	Tersedia	Tidak Tersedia
Major Interest	Pengetahuan	Data dan Informasi
Struktur	Terpisah antara kontrol dan pengetahuan	Kontrol terintegrasi dengan data
Output	Tidak harus lengkap	Harus lengkap
Maintenance dan Update	Mudah karena menggunakan modul-modul	Umumnya susah dilakukan
Hardware	*Workstation* dan PC	Semua tipe
Kemampuan pemikiran	Terbatas tetapi dapat ditingkatkan	Tidak ada

(Sumber : Desiani dan Arhami, 2005, 19)

2.2 Masalah Ruang Keadaan (*State and Space*)

Ruang keadaan (*state space*) merupakan suatu ruang yang berisi semua keadaan yang mungkin dalam suatu kasus AI. Kondisi dalam ruang keadaan meliputi:

1. Keadaan sekarang (keadaan awal / *start state*)
2. Keadaan tujuan (*goal state*), merupakan solusi yang dijangkau dan perlu diperiksa apakah telah mencapai sasaran.
3. Kaidah atau aturan yang memberikan batasan-batasan bagaimana mengubah suatu keadaan menjadi keadaan lain.

Keadaan direpresentasikan sebagai *node* (simpul), sedangkan langkah yang diperbolehkan atau aksi direpresentasikan dengan *arc* (busur). (Kusumadewi, 2003, 11-12)

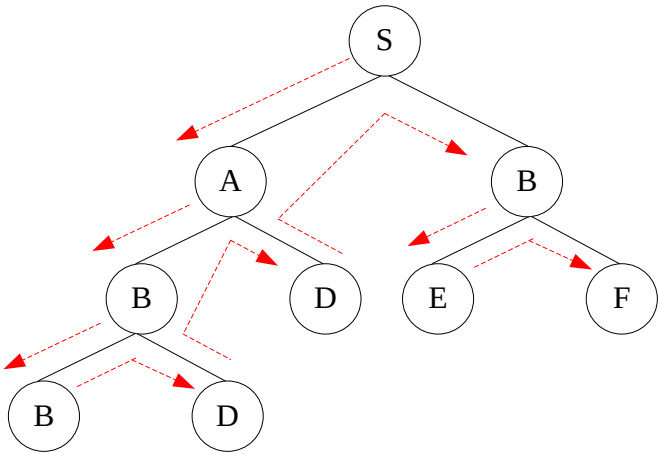
2.3 Teknik Pencarian

Hal terpenting dalam menentukan keberhasilan sistem yang berdasarkan AI adalah kesuksesan dalam pencarian dan pencocokan. Pencarian dan pelacakan

dilakukan dengan membangkitkan keadaan baru dari keadaan awal (*start state*) hingga didapatkan keadaan tujuan (*goal state*). Tetapi, ada kalanya keadaan tujuan (*goal state*) tidak akan pernah dicapai. Hal ini tergantung dari keadaan awal yang diberikan dan aturan-aturan yang harus dipatuhi untuk mencapai keadaan tujuan. (Kusumadewi, 2003, 23)

2.3.1 Pencarian Mendalam Pertama (*Depth First Search*)

Pencarian dengan metode *Depth First Search* (DFS) 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 pada gambar 2.3. (Desiani dan Arhami, 2005, 30)



Gambar 2.3 Teknik *Depth First Search*

(Sumber : Desiani dan Arhami, 2005, 30)

Berdasarkan gambar 2.3, 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 (*goal*). Operasi semacam ini dikenal dengan sebutan *backtracking*.

DFS juga memiliki kelebihan di antaranya adalah cepat mencapai kedalaman ruang pencarian. Jika diketahui bahwa lintasan solusi permasalahan akan panjang maka DFS tidak akan memboros waktu untuk melakukan sejumlah

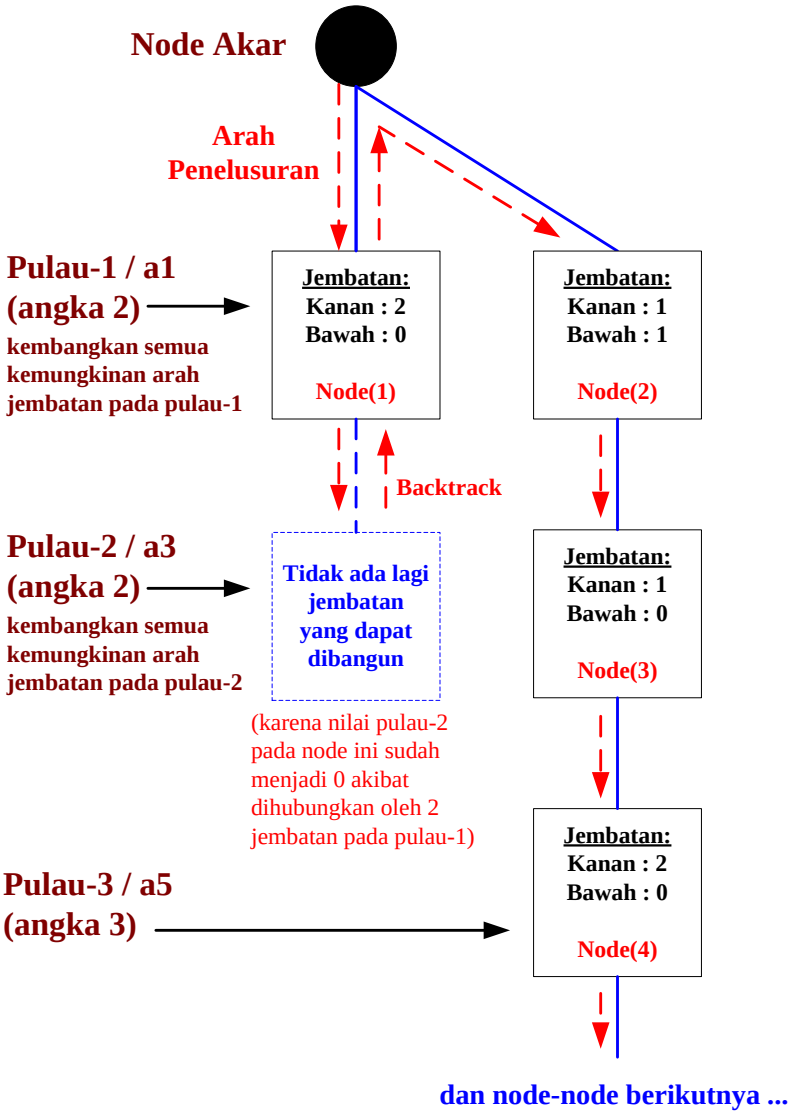
besar keadaan ‘dangkal’ dalam permasalahan *graph*/pohon. DFS jauh lebih efisien untuk ruang pencarian dengan banyak cabang karena tak perlu mengevaluasi semua simpul pada suatu level tertentu pada daftar *open*. Selain itu, DFS memerlukan memori yang relatif kecil karena hanya *node-node* pada lintasan yang aktif saja yang disimpan.

Selain kelebihan, DFS juga memiliki kelemahan, di antaranya adalah memungkinkan tidak ditemukannya tujuan yang diharapkan dan hanya akan mendapatkan satu solusi pada setiap pencarian. (Desiani dan Arhami, 2005, 31)

Secara ringkas, algoritma proses DFS dapat dijabarkan sebagai berikut:

1. Penelusuran dimulai pada pulau-1.
2. Kembangkan satu kemungkinan arah jembatan ke kanan atau ke bawah dari pulau-1. Tempatkan kemungkinan arah jembatan *valid* ini ke dalam satu *node* baru pada level-1, misalkan Node(1).
3. Penelusuran dilanjutkan pada Node(1). Kembangkan lagi satu kemungkinan arah jembatan ke kanan atau ke bawah dari pulau-2. Tempatkan kemungkinan arah jembatan *valid* ini ke dalam *node* baru pada level-2, misalkan Node(2).
4. Teruskan penelusuran yang sama pada *node* berikutnya, secara mendalam pada level-level berikutnya.
5. Apabila terdapat suatu *node* yang tidak dapat membangun jembatan karena menyalahi aturan Hashi, maka penelusuran kembali (*backtrack*) ke *node* induk dan bangun lagi jembatan lain yang berbeda dengan *node* anak yang sudah ada. Teruskan penelusuran ke *node* anak yang baru.
6. Penelusuran yang sama dilanjutkan pada *node-node* berikutnya, hingga didapatkan solusi.

Sebagai contoh, penelusuran solusi dengan pohon DFS dapat dilihat pada gambar 3.16 berikut.



Gambar 2.4 Penerapan DFS dalam Mencari Solusi *Hashi*

2.3.2 Teknik Pencarian BFS

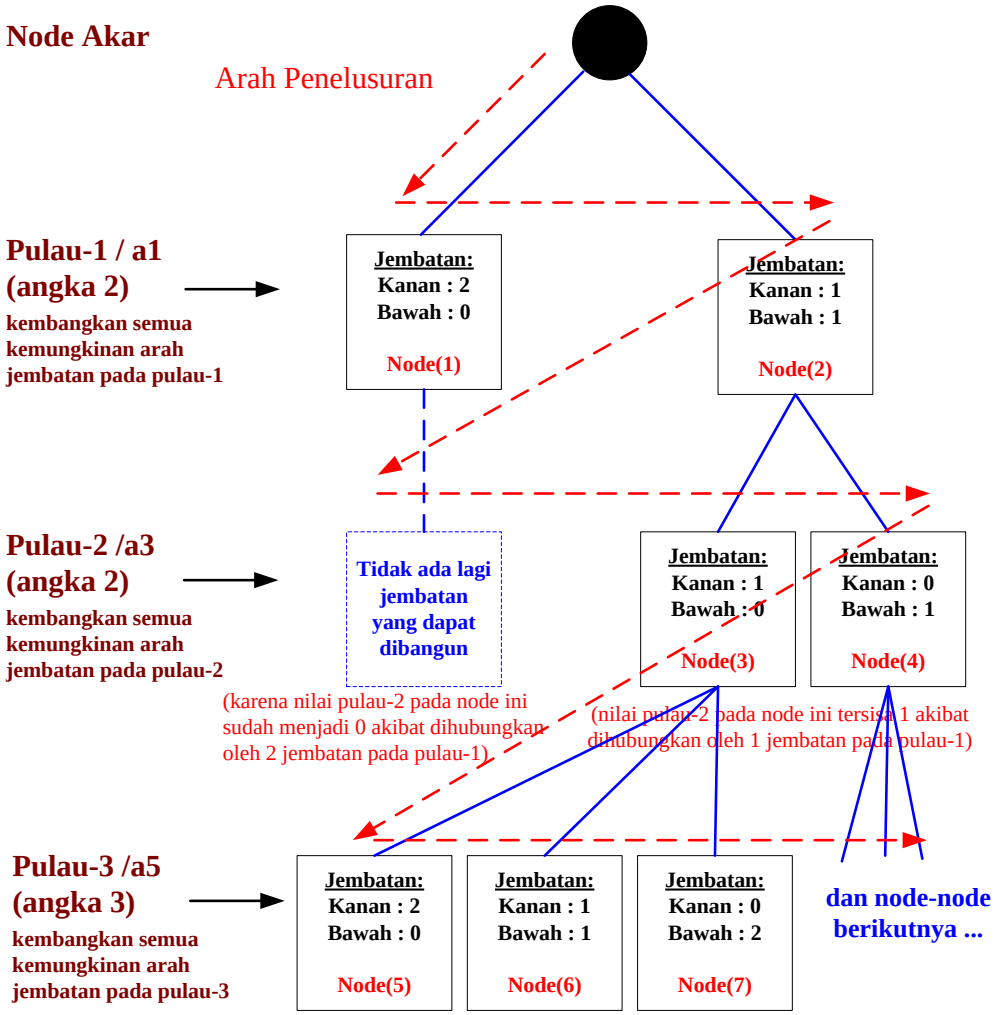
Metode pencarian BFS akan mengembangkan ruang dan keadaan secara melebar. Dari keadaan awal *puzzle Hashi*, algoritma BFS akan mengembangkan semua kemungkinan arah jembatan yang dapat dibangun pada suatu pulau. Oleh karena, penelusuran dimulai dari pulau paling kiri atas, maka penelusuran kemungkinan arah jembatan bisa dibatasi hanya pada arah kanan dan bawah, untuk setiap pulau. Dari pulau paling kiri atas, semua kemungkinan jembatan dikembangkan. Kemudian berlanjut ke pulau-pulau atau *node-node* berikutnya.

Pengembangan arah jembatan tidak boleh menyalahi aturan Hashi. Dengan demikian, semua kemungkinan arah jembatan di dalam *node-node* BFS adalah jembatan yang *valid* dan tidak menyalahi aturan. Proses pengembangan ini berlanjut hingga ditemukan solusi *puzzle*.

Secara ringkas, algoritma proses BFS dapat dijabarkan sebagai berikut:

1. Penelusuran dimulai pada pulau-1.
2. Kembangkan semua kemungkinan arah jembatan ke kanan dan ke bawah dari pulau-1. Tempatkan semua kemungkinan arah jembatan *valid* ini ke dalam *node-node* baru pada level-1, misalkan Node(1) dan Node(2).
3. Penelusuran dilanjutkan pada Node(1). Kembangkan lagi semua kemungkinan arah jembatan ke kanan dan ke bawah dari pulau-2. Tempatkan semua kemungkinan arah jembatan *valid* ini ke dalam *node-node* baru pada level-2, misalkan Node(3) dan Node(4).
4. Penelusuran yang sama dilanjutkan pada Node(2) dan seterusnya, hingga didapatkan solusi.

Sebagai contoh, penelusuran solusi dengan pohon BFS dapat dilihat pada gambar 2.5 berikut.



Gambar 2.5 Penerapan BFS dalam Mencari Solusi *Hashi*

2.3.3 Pencarian Heuristik (*Heuristic Searching*)

Pencarian heuristik biasanya diistilahkan dengan *informed search* atau pencarian yang berdasarkan panduan. George Poyla mendefinisikan heuristik sebagai studi metode dan kaidah penemuan. Dalam pencarian ruang keadaan, heuristik dinyatakan sebagai aturan untuk melakukan pemilihan cabang-cabang dalam ruang keadaan yang paling tepat untuk mencapai solusi permasalahan yang dapat diterima.

Pemecahan masalah AI menggunakan heuristik dalam dua situasi dasar, yaitu:

1. Permasalahan yang mungkin tidak mempunyai solusi yang pasti disebabkan oleh ambiguitas (keraguan/ketidakpastian) mendasar dalam pernyataan permasalahan atau data yang tersedia. Diagnosa kedokteran merupakan salah satu contohnya, dimana sejumlah gejala mungkin dapat ditimbulkan oleh berbagai macam penyebab yang mungkin. Dokter menggunakan heuristik untuk memilih atau menentukan diagnosa yang paling dapat diharapkan dan merumuskan rencana penanganannya. *Vision* merupakan masalah lainnya untuk permasalahan ketidakpastian mendasar.
2. Permasalahan yang boleh jadi memiliki solusi pasti, tetapi biaya komputasi untuk mendapatkan solusi tersebut mungkin sangat tinggi. Dalam banyak masalah seperti catur, ruang keadaan bertambah luar biasa seiring dengan pertambahan jumlah keadaan yang dimungkinkan.

Sayangnya, seperti semua kaidah penemuan lainnya, heuristik juga dapat salah. Heuristik hanyalah panduan informasi untuk menebak langkah berikutnya yang harus diambil dalam menyelesaikan suatu permasalahan, dan sering dilakukan berdasarkan eksperimen/percobaan atau secara intuisi. Oleh karena menggunakan informasi yang terbatas, heuristik jarang dapat memprediksi tingkah laku yang eksak dari ruang keadaan saat dilakukan pencarian. Heuristik dapat membimbing solusi suboptimal atau gagal menemukan solusi apapun.

Heuristik dan perancangan algoritma untuk mengimplementasikan pencarian heuristik telah menjadi inti permasalahan penelitian AI. *Game playing* dan pemecahan teorema (*theorem solving*) adalah dua aplikasi yang paling tua dari AI, yang menggunakan heuristik untuk memangkas ruang dari solusi yang mungkin.

Ada 4 metode pencarian yang terdapat dalam teknik pencarian heuristik, yaitu:

1. *Generate and Test*
2. *Hill Climbing*
3. *Best First Search*
4. *Simulated Annealing*

(Desiani dan Arhami, 2005, 33-35)

2.3.4 *Generate-and-Test*

Metode *Generate-and-Test* (pembangkit dan pengujian) merupakan metode yang paling sederhana dalam pencarian heuristik. Jika pembangkitan *possible solution* dikerjakan secara sistematis maka prosedur akan mencari solusinya, jika memang ada. Namun jika ruang masalahnya sangat luas, mungkin akan memerlukan waktu yang sangat lama.

Metode *Generate-and-Test* pada umumnya merupakan gabungan antara teknik pencarian DFS dan BFS, yaitu bergerak ke belakang menuju suatu keadaan awal. Nilai pengujiannya berupa jawaban “ya” atau “tidak”.

Jika pembuatan solusi yang dimungkinkan dapat dilakukan secara sistematis maka prosedur ini dapat menemukan solusinya bila ada. Akan tetapi, ruang permasalahannya sangat besar maka proses ini akan membutuhkan waktu yang lama. Metode ini juga dapat dilakukan dengan pembangkitan solusi secara acak, namun tidak ada jaminan solusi akan ditemukan. (Desiani dan Arhami, 2005, 35)

2.3.5 *Hill Climbing*

Metode ini hampir sama dengan metode *Generate-and-Test* (pembangkit dan pengujian) dan merupakan salah satu variasi dari metode tersebut. Yang membedakan kedua metode ini adalah umpan balik (*feedback*) yang berasal dari prosedur pengujian digunakan untuk memutuskan arah gerak dalam pencarian. Selain itu, dalam *Generate-and-Test* respons fungsi pengujian hanya ya atau tidak. Namun, jika fungsi pengujian dikombinasikan dengan fungsi heuristik yang menyediakan pengukur kedekatan suatu keadaan yang diberikan dengan keadaan tujuan maka prosedur *generate* dapat mengungkapkannya.

Hill Climbing sering digunakan jika terdapat fungsi heuristik yang baik untuk mengevaluasi *state*. Sebagai contoh, ada seseorang yang berada di sebuah kota yang tidak dikenal, tanpa peta dan ingin ke pusat kota. Cara sederhananya adalah mencari gedung yang tinggi. Fungsi heuristiknya adalah jarak antara lokasi sekarang dengan gedung yang tinggi dan *state* (jalan) yang dibutuhkan adalah

jarak yang terpendek. (Desiani dan Arhami, 2005, 38-39)

2.3.6 *Simulated Annealing*

Simulated Annealing atau Simulasi *Annealing* adalah teknik optimalisasi numerik dengan prinsip *thermo-dynamic*. *Annealing* adalah proses dimana material solid dilebur dan didinginkan secara perlahan-lahan dengan mengurangi temperatur. Partikel dari material berusaha menyusun dirinya selama proses pendinginan. Metropolis menemukan bahwa proses *equilibrium* dapat disimulasikan untuk temperatur tetap menggunakan metode Monte Carlo untuk membangkitkan deretan *state* energi.

Simulated Annealing (SA) dikembangkan pada tahun 1983. Abramson (1991) mengemukakan bahwa aplikasi SA pada masalah penjadwalan relatif berbanding lurus. Atom-atom diganti dengan *elemen*. Elemen adalah kombinasi tertentu dari dosen, matakuliah, ruang, dan kelas. Suatu elemen dapat diperlukan lebih dari satu kali dalam seminggu. Energi sistem diganti dengan biaya penjadwalan. Alokasi awal dibuat dalam kondisi dimana elemen diganti dengan sesi (satuan waktu perkuliahan) yang dipilih secara acak. Temperatur awal ditentukna. Biaya digunakan untuk mencerminkan kualitas penjadwalan, sebagaimana energi sistem yang mencerminkan kualitas substansi yang diinginkan (*annealed*). Temperatur digunakan untuk mengontrol probabilitas kenaikan biaya. (Desiani dan Arhami, 2005, 55)

2.4 Struktur Data

Struktur berarti susunan/jenjang, dan data berarti sesuatu simbol/huruf/lambang angka yang menyatakan sesuatu. Struktur data berarti susunan dari simbol/huruf/lambang angka untuk menyatakan sesuatu hal. Sebagai contoh, struktur program Pascal dapat didefenisikan seperti berikut,

1. Judul Program
2. Bagian Deklarasi / Blok Program, yang terdiri dari,
   a. Deklarasi Label
   b. Deklarasi Konstanta

c. Deklarasi Tipe Data
d. Deklarasi Variabel
e. Deklarasi Nama Fungsi / Prosedur

Gabungan dari algoritma dan struktur data akan membentuk suatu program. Adapun manfaat dari struktur data adalah sebagai berikut,

1. Mengefisiensikan program

   Program yang dibuat dengan menerapkan konsep–konsep yang berlaku pada struktur data akan lebih efisien dibandingkan dengan program yang dibuat dengan mengabaikan konsep struktur data.

2. Modifikasi

   Sesuatu program harus dapat dimodifikasi apabila diperlukan, hal ini dapat dilakukan jika fasilitas yang diperlukan dibuat (disertakan) walaupun pada tahap awal belum dipakai.

3. Memilih metode yang tepat

   Misalkan suatu *plaza* pada hari–hari tertentu mengalami antrian yang panjang pada kasir, hal ini dapat diatasi dengan metode,

a. Pemasukan data tidak melalui *keyboard* lagi, melainkan melalui *barcode*.
b. Membuat pemberitahuan pada kasir–kasir.
c. Berdasarkan hal tersebut, maka di setiap ATM selalu ada pemberitahuan misalkan “ATM ini hanya mengeluarkan uang pecahan Rp. 100.000,-“.

2.4.1 *Array* (Larik)

Array (larik) adalah suatu tipe data terstruktur yang dapat menampung (berisikan) data-data yang sejenis. Komponen–komponen dari *Array* antara lain,

1. Nama *Array*
2. Nilai *Array*
3. Index *Array*
4. Jenis (tipe) *Array*

Deklarasi *Array* dapat dibuat pada,

1. Bagian Tipe

   Contoh,

Type A = Array [1..5] of Char;

2. Bagian Variabel

   Contoh,

   Var A = Array [1..5] of Char;

Array terdiri dari beberapa jenis antara lain,

1. *Array* 1 dimensi.

   Merupakan sebuah variabel yang menyimpan sekumpulan data yang memiliki tipe sama dan elemen yang akan diakses hanya melalui 1 indeks atau subskrip.

   Bentuk umum pendeklarasian:

   nama_array[jumlah_elemen];

   Contoh misalkan diketahui sebuah *Array* A dengan jenis data *Char*,

A =	A	B	C	D	E	F	G	H
	1	2	3	4	5	6	7	8

   A[1] = ‘A’ A[5] = ‘E’

   A[2] = ‘B’ A[6] = ‘F’

   A[3] = ‘C’ A[7] = ‘G’

   A[4] = ‘D’ A[8] = ‘H’

2. *Array* Multi dimensi

   Merupakan sebuah variabel yang menyimpan sekumpulan data yang memiliki tipe sama dan elemen yang akan diakses melalui 2 indeks atau subskrip yaitu indeks baris dan indeks kolom.

   Bentuk umum pendeklarasian:

   nama_array[jumlah_elemen_baris][jumlah_elemen_kolom];

   Misalkan diketahui sebuah *Array* dua dimensi B dengan jenis data *Integer*, maka contoh penulisan *array* 2 dimensi adalah: B[1,1] = 1, B[1,2] = 2 dan B[2,1] = 3.

2.4.2 *Record*

Record (rekaman) adalah suatu tipe data terstruktur yang berisikan data yang sejenis atau berbeda. *Record* adalah kumpulan dari beberapa *field*. Panjang

satu *record* sama dengan jumlah panjang *item*-nya (*field*-nya). Contoh, satu *record* Mahasiswa terdiri dari *field* Nama, NIM, Alamat dan Nilai. Operasi-operasi yang dapat dilakukan pada *record* adalah,

1. Operasi menyimpan data.
2. Operasi mengubah data.
3. Operasi menghapus data.
4. Operasi mencari data.
5. Operasi mengurutkan data.

2.4.3 *File*

Untuk menyimpan *record*, dibutuhkan *file*. Kumpulan *record* akan membentuk sebuah *data file* (biasa disebut dengan *file* saja). Dalam satuan *file* inilah, data bisa disimpan di media penyimpan elektronik *external memory* seperti *disket*, *hard disk*, *flash disk*, dan sebagainya. Jadi, bila sebuah *file* digambarkan sebagai sebuah tabel, maka *record* adalah barisnya (*row*), dan *field* adalah kolomnya (*column*). Selain *data file*, ada ribuan jenis *file* lain yang ada di sebuah *hard disk* misalnya. *File* biasanya dibedakan dari jenis dan formatnya.

2.4.4 Basis Data (*Database*)

Basis data dapat dibayangkan sebagai sebuah lemari arsip. Tetapi basis data terdiri atas dua kata yaitu basis dan data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek.

Beberapa definisi basis data adalah sebagai berikut:

1. Himpunan kelompok data (arsip) yang saling berhubungan dan diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan mudah dan cepat.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan yang tidak perlu, untuk memenuhi berbagai kebutuhan.
3. Kumpulan *file*/table/arsip yang saling berhubungan yang disimpan pada media

penyimpanan elektronis.

4. Meliputi spesifikasi tipe data struktur dan pembatasan (*constraints*) dari data yang disimpan.

Basis data dan lemari arsip sesungguhnya memiliki prinsip kerja dan tujuan yang sama. Prinsip utamanya adalah pengaturan data atau arsip. Dan tujuan utamanya adalah kemudahan dan kecepatan. yang sangat ditonjolkan dalam basis data adalah pengaturan, pemilihan, pengelompokan dan pengorganisasian data yang akan disimpan sesuai dengan fungsi dan jenisnya.

Elemen-elemen dalam suatu basis data adalah sebagai berikut :

1. *Record* merupakan kumpulan data yang terdiri dari satu atau lebih *field*.
2. *Field* merupakan kumpulan data yang mempunyai atau menyimpan fakta yang sama (sejenis) untuk setiap baris pada tabel.
3. Tabel merupakan bentuk matriks dari item-item data yang diorganisir menjadi baris dan kolom.

2.5 Permainan *Hashi*

2.5.1 Pengenalan

Hashi (*Hashiwokakero*) adalah salah satu permainan *puzzle* yang membutuhkan logika berpikir. *Hashi* diciptakan oleh Nikoli, sebuah perusahaan *game* di Jepang yang memiliki spesialisasi dalam pembuatan *game* logika. *Puzzle* logika lain yang juga diciptakan oleh Nikoli adalah *Sudoku* dan *Kakuro*. Selain mengasyikkan, *Hashi* juga menguji logika dan kecerdasan orang yang memainkannya.

Menurut Wikipedia, *puzzle Hashi* muncul di Majalah *Times* dan dipublikasikan dalam bahasa Inggris dengan nama *Bridges* atau *Chopsticks*. Di Perancis, Denmark, Belanda dan Belgia, *puzzle Hashi* dipublikasikan dengan nama Ai-Ki-Ai. *Puzzle Hashi* pertama sekali muncul pada bulan Desember 1989 di majalah *Puzzle* Nikoli pada edisi ke-28.

2.5.2 Aturan Main

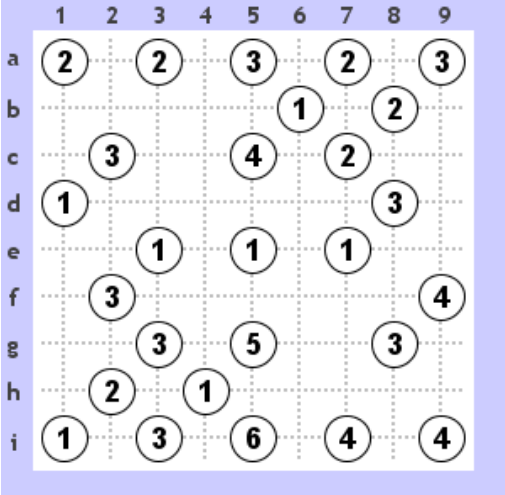
Hashi dimainkan pada sebuah kotak. Di dalam kotak ini, terdapat beberapa

lingkaran (sel) yang terletak secara horizontal dan vertikal. Sel ini dikenal juga dengan sebutan pulau. Di dalam setiap pulau, terdapat angka dari 1 sampai 8. Tujuan dari permainan *Hashi* adalah menghubungkan semua pulau dengan membuat jembatan antar pulau dan tidak ada pulau yang terisolir (atau tidak terhubung satu sama lain).

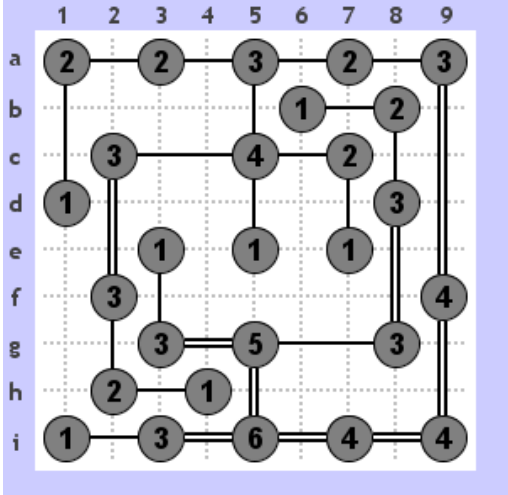
Aturan pembuatan jembatan pada kotak *Hashi* adalah sebagai berikut:

1. Satu jembatan dapat menghubungkan dua pulau yang terletak pada koordinat *x* yang sama (horizontal) atau koordinat *y* yang sama (vertikal),
2. Jembatan tidak boleh saling bertimpaan dengan jembatan lainnya,
3. Maksimum hanya boleh terdapat 2 jembatan di antara 2 pulau yang sama,
4. Banyaknya jembatan yang terhubung ke pulau harus sama dengan angka yang tertulis pada pulau.

Gambar soal *Hashi* dapat dilihat pada gambar 2.6 dan gambar jawaban soal *Hashi* dapat dilihat pada gambar 2.7.



Gambar 2.6 Soal *Hashi*



Gambar 2.7 Jawaban Soal *Hashi*

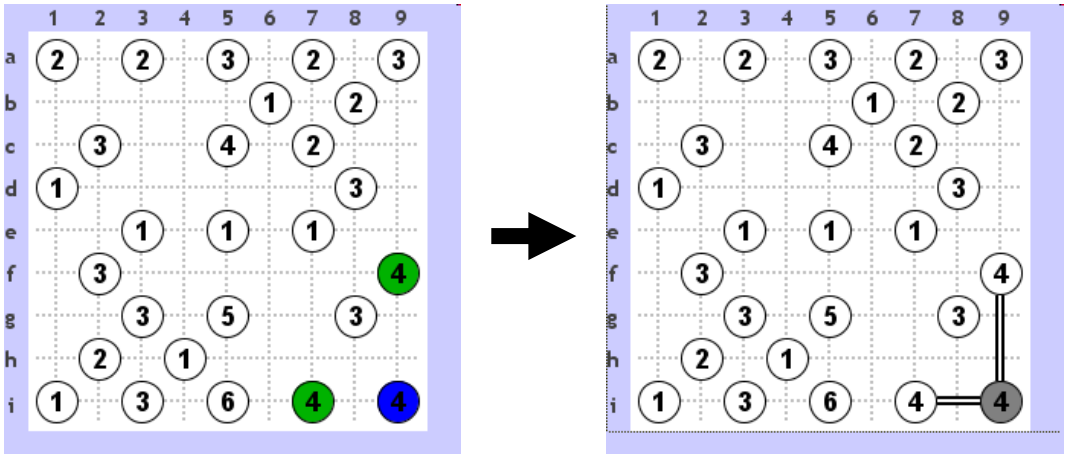
2.5.3 Cara Penyelesaian *Hashi*

Menurut *Indigopuzzles*, cara terbaik untuk mencari jawaban *Hashi* adalah dengan berusaha menyelesaikan satu per satu masalah dalam setiap soalnya. Beberapa teknik *solving Hashi* yang dipublikasikan oleh *Indigopuzzles* (http://www.indigopuzzles.com/ipuz/help.action?helpId=hashi/howToSolve_01) dan dapat digunakan untuk mencari penyelesaian *Hashi*, yaitu:

1. Teknik *Just Enough Neighbours*
2. Teknik *One Unsolved Neighbour*
3. Teknik *Few Neighbours*
4. Teknik *Leftovers*
5. Teknik *Isolation*

2.5.3.1 Teknik "*Just Enough Neighbours*"

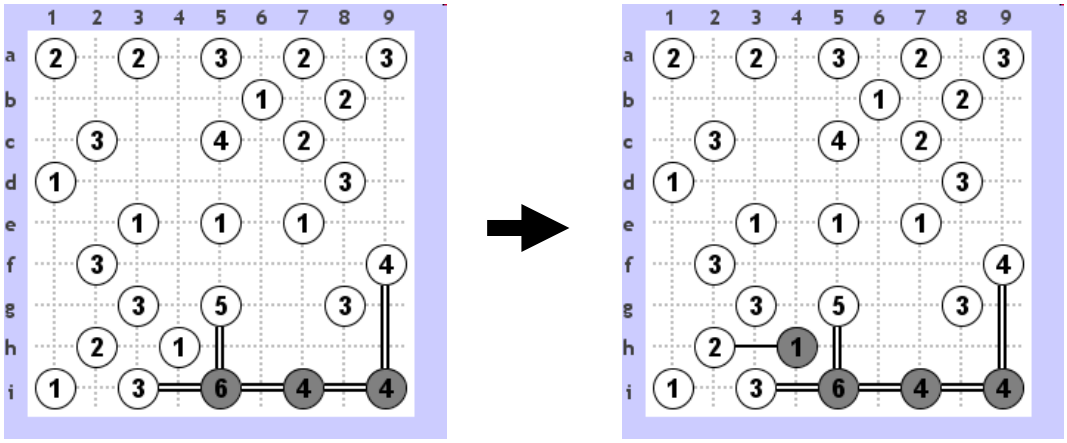
Perhatikan pada gambar 2.6 bahwa pulau (4) pada i9 membutuhkan 4 jembatan. Dan oleh karena berada di pojok, pulau tersebut tidak mempunyai pilihan jembatan lain selain menghubungkan diri dengan pulau f9 dan i7, dengan masing-masing 2 jembatan.



Gambar 2.8 Teknik Penyelesaian "*Just Enough Neighbours*"

2.5.3.2 Teknik "*One Unsolved Neighbours*"

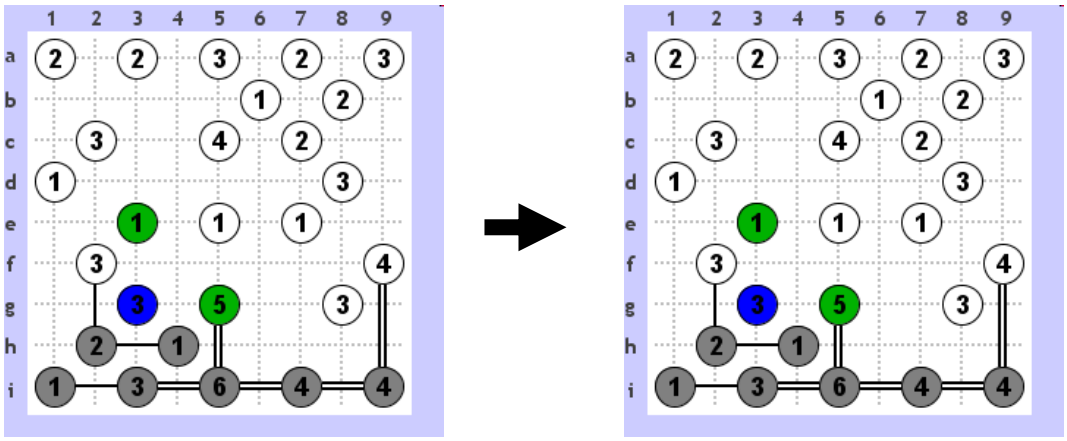
Perhatikan pada gambar 2.7 bahwa pulau (1) pada h4 hanya membutuhkan 1 jembatan. Dan oleh karena tidak ada pulau lain yang dapat diraih, maka pulau h4 hanya bisa menghubungkan dirinya dengan pulau (2) pada h2.



Gambar 2.9 Teknik Penyelesaian "*One Unsolved Neighbours*"

2.5.3.3 Teknik "*Few Neighbours*"

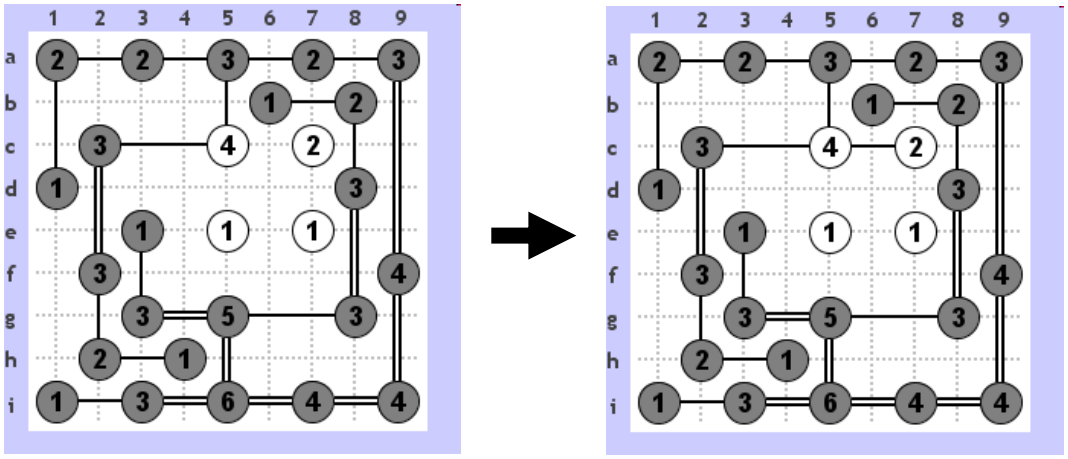
Teknik ini berdasarkan pada peraturan bahwa "hanya boleh ada 2 jembatan antar pulau". Perhatikan pada gambar 2.8 bahwa pulau (3) pada g3 hanya bisa meraih 2 pulau lainnya. Dan oleh karena pulau e3 hanya mempunyai angka 1, maka pulau g3 menghubungkan 1 jembatan ke pulau e3 dan 2 jembatan ke pulau g5.



Gambar 2.10 Teknik Penyelesaian "*Few Neighbours*"

2.5.3.4 Teknik "*Leftovers*"

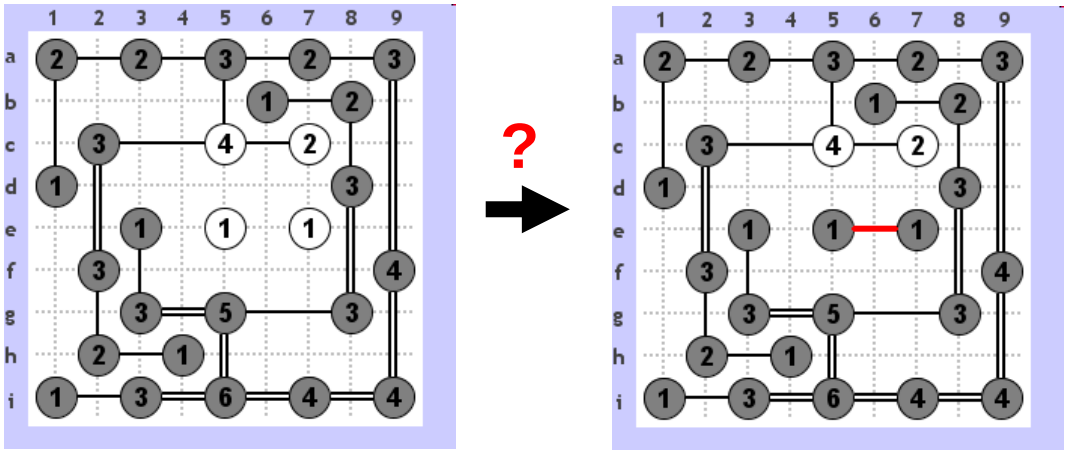
Perhatikan pada gambar 2.9 bahwa pulau (4) pada c5, membutuhkan 2 jembatan lainnya. Salah satu tetangganya adalah pulau (1) pada e5 yang hanya dapat menerima 1 jembatan. Oleh karena itu, setidaknya satu jembatan pada pulau c5 harus mengarah ke pulau lainnya, yaitu pulau (2) pada c7.



Gambar 2.11 Teknik Penyelesaian "*Leftovers*"

2.5.3.5 Teknik "*Isolation*"

Teknik ini adalah teknik yang paling penting untuk menyelesaikan *puzzle Hashi*. Teknik ini menggunakan aturan bahwa setiap pulau harus bisa menjangkau pulau lainnya. Perhatikan pulau (1) di e5 pada gambar 2.12 yang hanya akan memiliki 1 jembatan berikut.



Gambar 2.12 Teknik Penyelesaian "*Isolation*"

Pada gambar 2.12, jika pulau e5 dihubungkan ke pulau di samping kanannya, maka kedua pulau tersebut akan menjadi ter-isolasi. Oleh karenanya, pulau e5 harus dihubungkan dengan pulau di atasnya, c5.

2.6 Microsoft Visual Basic 6.0

Microsoft Visual Basic (sering disingkat sebagai VB) merupakan sebuah bahasa pemrograman yang bersifat *even driven* dan menawarkan *Integrated Development Environment* (IDE) visual untuk membuat program aplikasi berbasis sistem operasi Microsoft Windows (COM). Visual Basic merupakan turunan bahasa BASIC dan menawarkan pengembangan aplikasi komputer berbasis grafik dengan cepat, akses ke basis data menggunakan *Data Access Object* (DAO), *Remote Data Objects* (RDO), atau *ActiveX Data Object* (ADO), serta menawarkan pembuatan kontrol *ActiveX* dan objek *ActiveX*. Beberapa bahasa skrip seperti *Visual Basic for Applications* (VBA) dan *Visual Basic Scripting Editio*n (VBScript), mirip seperti halnya Visual Basic, tetapi cara kerjanya yang berbeda.

Para programer dapat membangun aplikasi dengan menggunakan komponen-komponen yang disediakan oleh Microsoft Visual Basic Program-program yang ditulis dengan Visual Basic juga dapat menggunakan Windows API, tapi membutuhkan deklarasi fungsi eksternal tambahan.

Dalm pemrograman untuk bisnis, Visual Basic memiliki pangsa pasar yang sangat luas. Dalam sebuah survey yang dilakukan pada tahun 2005, 62% pengembang perangkat lunak dilaporkan menggunakan berbagai bentuk Visual Basic, yang diikuti oleh C++, JavaScript, C# dan Java.

Bill Gates, pendiri Microsoft, memulai bisnis softwarenya dengan mengembangkan interpreter bahasa basic untuk Altair 8800, untuk kemudian di ubah agar berjalan di atas IBM PC dengan sistem operasi DOS. Perkembangan berikutnya ialah diluncurkannya BASICA (*basic-advanced*) utuk DOS. Setelah BASICA, Microsoft meluncurkan Microsoft QuickBasic dan Microsoft Basic (dikenal juga sebagai Basic Compiler).

Sejarah BASIC di tangan Microsoft sebagai bahasa yang diinterpretasi (BASICA) dan juga bahasa yang dikompilasi (BASCOM) membuat *Visual Basic* diimplementasikan sebagai gabungan keduanya.

Programer yang menggunakan *Visual Basic* bisa memilih kode terkompilasi atau kode yang harus diinterpretasi sebagai hasil *executable* dari

kode VB. Sayangnya, meskipun sudah terkompilasi jadi bahasa mesin, DLL bernama MSVBVMxx.DLL tetap dibutuhkan. Namun karakteristiknya bahasa terkompilasi tetap muncul (ini lebih cepat jika di bandingkan dengan menggunakan mode terinterpretasi).

Visual Basic merupakan bahasa yang mendukung OOP, namun tidak sepenuhnya. Beberapa karakteristik objek tidak dapat dilakukan pada Visual Basic, seperti *Inhertance* tidak dapat dilakukan pada *class module*. *Polymorphism* secara terbatas bisa dilakukan dengan mendeklarasikan *class module* yang memiliki *Interface* tertentu. Visual Basic (VB) tidak bersifat case sensitif.

Visual Basic menjadi pupoler karena kemudahan desain form secara visual dan adanya kemampuan untuk menggunakan komponen-komponen *ActiveX* yang dibuat oleh pihak lain. Namun komponen *ActiveX* memiliki masalahnya tersendiri yang dikenal sebagai DLL *hell*. Pada Visual Basic.Net, Microsoft mencoba mengatasi masalah DLL *hell* dengan mengubah cara penggunaan komponen (menjadi independen terhadap *registry*).

2.7 Microsoft Access 2007

Microsoft Access (atau Microsoft Office Access) adalah sebuah program aplikasi basis data komputer relasional yang ditujukan untuk kalangan rumahan dan perusahaan kecil hingga menengah. Aplikasi ini merupakan anggota dari beberapa aplikasi Microsoft Office, selain tentunya Microsoft Word, Microsoft Excel, dan Microsoft PowerPoint. Aplikasi ini menggunakan mesin basis data Microsoft jet Database Engine, dan juga menggunakan tampilan grafis yang intuitif sehingga memudahkan pengguna. Versi terakhir adalah Microsoft Office Access 2010 yang termasuk ke dalam Microsoft Office System 2010.

Microsoft Access dapat menggunakan data yang disimpan di dalam format *Microsoft Access, Microsoft Jet Database Engine, Microsoft SQL Server, Oracle Database*, atau semua kontainer basis daya yang mendukung standar ODBC. Para pengguna/*programmer* yang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang kompleks, sementara para *programmer* yang kurang mahir dapat menggunakannya untuk mengembangkan

perangkat lunak aplikasi yang sederhana. Access juga mendukung teknik-teknik pemrograman beorientasi objek, tetapi tidak dapat digolongkan ke dalam perangkat bantu pemrograman berorientasi objek.

Microsoft merilis Microsoft Access 1.0 pada bukan November 1992 dan di lanjutkan dengan merilis versi 2.0 pada tahun 1993. Microsoft menentukan spesifikasi minimum untuk menjalankan Microsoft Access 2.0 adalah sebuah komputer dengan sistem operasi Microsoft Windows 3.0, RAM berkapasitas 4 *megabyte* (6 *megabyte* lebih disarankan) dan ruangan kosong *hard disk* yang dibutuhkan 8 *megabyte* (14 *megabyte* lebih disarankan). Versi 2.0 dari Microsoft Access ini datang dengan tujuh buah disket *floppy* 3½ inci berukuran 1.44 *megabyte.*

Perangkat lunak tersebut bekerja dengan sangat baik pada sebuah basis data dengan banyak *record* tapi terdapat beberapa kasus dimana *data* mengalami kerusakan. Sebagai contoh, pada ukuran basis data melebihi 700 *megabyte* sering mengalami masalah seperti ini (pada saat itu, memang *hard disk* yang beredar masih di bawah 700 *megabyte*). Buku manual yang dibawanya memperingatkan bahwa beberapa kasus tersebut disebabkan oleh *driver* perangkat yang kuno atau konfigurasi yang tidak benar.

Nama kode (*codename*) yang digunakan oleh Access pertama kali adalah Cirrus yang dikembangkan sebelum Microsoft mengembangkan Visual Basic, sementara mesin pembuat *form* antarmuka yang digunakan dinamakan dengan Ruby. Bill Gates melihat purwarupa (*prototype*) tersebut dan memutuskan bahwa komponen bahasa pemprograman BASIC harus dikembangkan secara bersama-sama sebagai sebuah aplikasi terpisah tapi dapat diperluas. Proyek ini dinamakan degan Thunder. Kedua proyek tersebut dikembangkan secara terpisah, dan mesin pembuatan *form* yang digunakan oleh keduanya tidak saling cocok satu sama lainya. Hal tersebut berakhir saat microsoft merilis Visual Basic for Application (VBA).

Microsoft Access digunakan kebanyakan oleh bisnis-bisnis kecil dan menengah, di dalam sebuah organisasi yang kecil bahkan mungkin juga digunakan oleh perusahaan yang cukup besar dan juga para programer untuk

membuat sebuah sistem buatan sendiri untuk menangani pembuatan dan manipulasi data. Access juga dapat digunakan sebagai basis data untuk aplikasi Web dasar yang disimpan didalam *server* yang menjalankan *Microsoft Internet Services* (IIS) dan menggunakan *Microsoft Active Server Pages* (ASP). Meskipun demikian, penggunaan Access kurang disarankan, mengingat telah ada *Microsoft SQL Server* yang memiliki kemampuan yang lebih tinggi.

Beberapa pengembang aplikasi profesional menggunakan Microsoft Access untuk mengembangkan aplikasi secara cepat (digunakan sebagai *Rapid Application Development/RAD tool*), khususnya untuk pembuatan purwarupa untuk sebuah program yang lebih besar dan aplikasi yang berdiri sendiri untuk para *salesman*.

Microsoft Access kurang begitu bagus jika diakses melalui jaringan sehingga aplikasi-aplikasi yang digunakan oleh banyak pengguna cenderung menggunakan solusi sistem manajemen basis data yang bersifat klien/server. Meskipun demikian, tampilan muka Access (*form, report, query,* dan kode Visual Basic) yang dimilikinya dapat digunakan untuk menangani basis data yang sebenarnya diproses oleh sistem manajemen basis data lainnya, seperti halnya *Microsoft Jet Database Engine* (yang secara *default* digunakan oleh Microsoft Access), *Microsoft SQL Server*, *Oracle Database*, dan beberapa produk lainnya yang mendukung ODBC.

Salah satu keunggulan Microsoft Access dilihat dari perspektif *Programmer* adalah kompatibilitasnya dengan bahasa pemprograman *Structured Query Language* (SQL) query dapat dilihat dan disunting sebagai statemen-statemen SQL dan statemen SQL dapat digunakan secara langsung didalam macro dan VBA Module untuk secara langsung memamipulasi tabel data dalam Access. Para pengguna dapat mencampurkan dan menggunakan kedua jenis bahasa tersebut (VBA dan Macro) untuk memprogram *form* dan logika dan juga untuk mengaplikasikan konsep berorientasi objek.

Microsoft SQL Server Deskop Engine (MSDE) 2000, yang merupakan sebuah versi mini dari Microsoft SQL Server 2000, dimasukkan kedalam Office

XP Developer Edition dan dapat digunakan oleh Microsoft Access sebagai alternatif dari *Microsoft Jet Database Engine*.

Tidak seperti sebuah sistem manajemen basis data relational yang komplit, *Microsoft JET Database Engine* tidak memiliki fitur *trigger* dan *standart procedure.* Dimulai dari Microsoft Access 2000 yang menggunakan Microsoft Jet Database Engine versi 4.0, ada sebuah sintaksis yang mengizinkan pembuatan kueri dengan beberapa parameter, dengan sebuah cara seperti halnya sebuah *stored procedure*, meskipun prosedur tersebut dibatasi hanya untuk sebuah petanyataan tiap prosedurnya. Access juga mengizinkan *form* untuk mengandung kode yang dapat dieksekusi ketika terjadi sebuah perubahan terhadap tabel basis data, seperti halnya *trigger,* selama modifikasi dilakukan hanya dengan menggunakan *form* tersebut, dan merupakan sesuatu hal yang umum untuk menggunakan kueri yang akan diteruskan (*pass-through)* dan teknik lainnya didalam Access untuk menjalankan *stored procedure* didalam RDBMS yang mendukungnya.

Dalam berkas *Access Database Project* (ADP) yang didukung oleh Microsoft Access 200 dan yang selanjutnya, fitur-fitur yang berkaitan dengan basis data berbeda dari versi format/struktur data yang digunakan Access(*.MDB), karena jenis berkas ini dapat membuat koneksi kesebuah basis data MSDE atau Microsoft SQL Server, ketimbang menggunakan Microsoft JET Database Engine. Sehingga, dengan menggunakan ADP adalah mungkin untuk membuat hampir semua objek didalam server yang menajalankan mesin basis data tersebut (tabel basis data dengan *constraints* dan *trigger, view, stored procedure* dan *UDF*). Meskipun demikian, yang disimpan didalam berkas ADP hanyalah *form, report, macro* dan modul, sementara untuk tabel dan objek lainnya disimpan didalam server basis data yang membelakangi program tersebut.

Access mengizinkan pengembangan yang relatif cepat karena semua tabel basis data, *kueri, form,* dan *report* disimpan didalm berkas basis data miliknya (*.MDB). untuk membuat Query menggunakan Query Design Grid, sebuah program berbasis grafis yang mengizinkan para penggunanya untuk membuat query tanpa harus mengetahui pemprograman SQL didalam Query Design Grid,

para pengguna dapat memperlihatkan tabel basis data sumber dari *query* dan memilih *field-field* mana yang hendak dikembalikan oleh proses dengan mengklik dan menyeretnya kedalam *grid. Join* juga dapat dibuat dengan cara mengklik dan menyeret *field-field* dalam tabel kedalam *field* dalam tabel lainnya. Access juga mengizinkan pengguna untuk melihat dan memanipulasinya ke kode SQL jika memang diperlukan.

Bahasa pemrograman yang tersedia didalam Access adalah Microsoft Visual Basic for Applications (VBA), seperti halnya dalam beberapa aplikasi Microsoft Office. Dua buah pustaka komponen *Component Object Model* (COM) utuk mengakses basis data pun disediakan, yakni *Data Access Object* (DAO), yang hanya terdapat didalam Access 97, dan *ActiveX Data Objects* (ADO) yang tersedia dalam versi-versi Access terbaru.