BAB 2

LANDASAN TEORI

2.1 Algoritma

Kata algoritma berasal dari kata *Algoris* dan *Ritmis*, pertama kali diungkapkan oleh Abu Ja'far Mohammed ibn Musa al Khowarizmi dalam buku Al-jabr W'almulqabala (Horowitz, Ellis dan Sartaj Sahni, 1978, p1).

Beberapa pengertian algoritma, antara lain adalah sebagai berikut :

1. Menurut D.R. Stinson

   Algoritma adalah sebuah kumpulan aturan-aturan untuk menyelesaikan suatu masalah dalam tahapan-tahapan yang terstruktur dan terbatas.

2. Menurut Abu Ja'far Mohammad Ibn Musa Al Khowarizmi

   Algoritma adalah suatu fungsi khusus untuk menyelesaikan suatu persoalan.

3. Menurut Goodman Hedetniemi

   Algoritma adalah urut-urutan terbatas dari operasi-operasi yang terdefinisi dengan baik, yang masing-masing membutuhkan memori dan waktu yang terbatas untuk menyelesaikan suatu masalah.

4. Menurut Gamedev.net (http://www.gamedev.net)

   Algoritma adalah sekumpulan instruksi untuk melaksanakan suatu tugas.

Berdasarkan definisi di atas maka pengertian algoritma bila dipandang dari ilmu komputer adalah suatu fungsi yang terdiri dari serangkaian langkah-langkah

yang terstruktur dan dituliskan secara sistematis yang akan dikerjakan untuk menyelesaikan masalah dengan bantuan kamputer.

2.2 Intelegensia Semu

Intelegensia semu atau kecerdasan buatan merupakan salah satu bagian ilmu computer yang membuat agar mesin (komputer) dapat melakajukan pekerjaan seperti dan sebaik yang dilakukan manusia. Pada awal diciptaknnya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kehidupan umat manusia. Komputer tidak lagi hanya digunakan sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia.

Manusia bisa menjadi pandai dalam menyelesaikan segala permasalahn di dunia ini karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki oleh seseorang tentu saja diharapkan akan lebih mampu 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 untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik, namun tanpa bekal pengetahuan dan pengalaman yang memdai, manusia juga tidak akan bisa menyelsaikan masalah dengan baik.

Agar komputer bisa bertindak seperti dan sebaik manusia, maka komputer juga harus diberi bekalpengetahuan, dan kemampuan untuk menlar. Untuk itu pada intlegensia semu, akan mencoba untuk memberikan beberapa metoda untuk membekali komputer dengan kedua komponen tersebut agar komputer bisa menjadi mesin yang pintar.

Lebih detilnya, pengertian kecerdasan buatan dapat dipandang dari berbagai sudut pandang, antara lain:

1. Sudut pandang kecerdasan.

   Kecerdasan buatan akan membuat mesin menjadi ‘cerdas’ (mampu berbuat seperti apa yang dilakukan oleh manusia).

2. Sudut pandang peneltian.

   Kecerdasan buatan 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.

   - Persepsi (*vision* & *speech*).
   - Bahasa alami (understanding, generation & translastion).
   - Pemikiran yang bersifat commonsense.
   - Robot control.

   b. Formal task.

   - Permainan /*games*.

- Matematika (geometri, logika, kalkulus integrasi, pembuktian).

c. Expert task.

- Analisis finansial.
- Analisis medikal.
- Analisis ilmu pengetahuan.
- Rekayasa (desain, pencarian kegagalan, perencanaan manufaktur).

3. Sudut pandang bisnis.

Kecredasan buatan adalah kumpulan peralatan yang sangat powerful dan metodologis dalam menyelesaikan masalah-masalah bisnis.

4. Sudut pandang pemrograman.

Kecerdasan buatan meliputi studi tentang pemrograman simbolik, penyelesaian masalah (problem solving) dan pencarian (searching).

Untuk melakukan aplikasi kecerdasan buatan ada 2 bagian utama yang sangat dibutuhkan, yaitu:

a. **Basis Pengetahuan (*Knowledge base*)**, berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
b. **Motor Inferensi (*Inference Engine*)**, yaitu kemampuan menarik kesimpulan berdasarkan pengalaman.

2.2.1 Kecerdasan Buatan dan Kecerdasan Alami

Jika dibandingkan dengan kecerdasan alami (kecerdasan yang dimiliki oleh manusia), kecerdasan buatan memiliki beberapa keuntungan secara komersial antara lain:

a. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. Kecerdasan buatan tidak akan berubah sepanjang sisitem komputer & program tidak mengubahnya.

b. Kecerdasan buatan lebih mudah diduplikasi & disebarkan. Mentransfer pengethuan manusia dari satu orang ke orang lain membutuhkan proses yang sangat lama; dan juga dsuatu keahlian itu tidak akan pernah dapat diduplikasi degan 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.

c. Kecerdasan buatan lebih murah dibanding kecerdasan alami. Menyediakan layanan komputer akan lebih mudah & lebih murah dibandingkan dengan harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.

d. Kecerdaan buatan bersifat konsisten. Hal ini disebabkan karena kecerdasan buatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami akan senantiasa berubah-ubah.

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

f. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibanding dengan kecerdasan alami.

g. Kecerdasan buatan dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami.

Sedangkan keuntungan dari kecerdasan alami adalah:

a. Kreatif. Kemampuan untuk menambah ataupun memnuhi pengetahuan itu sanagat melekat pada jiwa manusia. Pada kecerdasan buatan, untuk menambah pengethuan harus dilakukan melalui sistem yang dibangun.

b. Kecerdasan alami memungkinkan orang untuk menggunakan pengalaman secara langsung. Sedangkan pada kecerdasan buatan harus bekerja secara input-input simbolik.

c. Pemikiran manusia dapat digunaka secara luas, sedangkan kecerdasan buatan sangat terbatas.

2.2.2 Sejarah Kecerdasan Buatan

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-olah 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 (seperti layaknya manusia).

Kecredasan Buatan atau "Artificial Intelligence" itu sendiri dimunculkan oleh seorang profesor dari Massachusetts Institute of Technology yang bernama John McCarthy pada tahun 1956 pada Dartmouth Conference yang dihadiri oleh para peneliti AI. Pada koferensi tersebut juga didefinisikan tujuan utama dari kecerdasan buatan, yaitu: mengetahui dan memodelkan proses-proses berfikir manusia dan mendesain mesin agar dapat menirukan kelakuan manusia tersebut.

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

- Logic Theorist, diperkenalkan pada Dartmouth Conference, program ini dapat membuktikan teorema-teorema matematika.
- 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.
- ELIZA, diprogram oleh Joseph Weizenbaum (1967). Program ini mampu melakukan terapi terhadap pasien dengan memberikan beberapa pertanyaan.

2.2.3 Lingkup Kecerdasan Buatan Pada Aplikasi Komersial

Makin pesatnya perkembangan teknologi menyebabkan adanya perkembangan dan perluasan lingkup yang membutuhkan kehadiran kecerdasan buatan. Karakteristik ‘cerdas’ sudah mulai dibutuhkan di berbagai disiplin ilmu dan 5teknologi. Kecerdasan buatan tidak hanya dominan di bidang ilmu komputer (informatika), namun juga sudah merambah di berbagai disiplin ilmu yang lain. Irisan antara psikologi dan kecerdasan buatan melahirkan sebuah area yang dikenal dengan nama cognition & psycolinguistics. I4risan antara teknik elektro dengan kecerdasan buatan melahirkan berbagai ilmu seperti: pengolahan citra, teori kendali, pengenalan pola dan robotika.

Dewasa ini, kecerdasan buatan juga memberikan kontribusi yang cukup besar di bidang manajemen. Adanya sistem pendukung keputusan,

dan Sistem Informasi Manajemen juga tidak terlepas dari andil kecerdasan buatan.

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

Lingkup utama kecerdasan buatan adalah:

1. Sistem Pakar (Expert System). Di sini 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 dengan menggunakan suara.
4. Robotika & Sistem Sensor (Robotics & Sensory Systems.
5. Computer Vision, mencoba untuk dapat menginterpretasikan gambar atau obyek-obyek tampak melalui komputer.

6. Intelligent Computer-aided Instruction. Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar.
7. Game Playing.

Beberapa karakteristik yang ada pada sistem yang enggunakan artificial intelligence adalah pemrogramannya yang cenderung bersifat simbolik ketimbang algoritmik, bisa mengakomodasi input yang tidak lengkap, bisa melakukan inferensi, dan adanya pemisahan antara komtrol 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.

2.3 Data

Menurut Prahasta (2002) data adalah representasi dari kenyataan apa adanya di lapangan, konsep-konsep atau instruksi-instruksi yang diformalkan dan sesuai untuk komunikasi, interpretasi atau pemrosesan, baik yang dilakukan oleh manusia maupun secara otomatis dengan bantuan mesin dan alat-alatnya. Maka dari itu data merupakan bentuk yang masih mentah yang belum dapat bercerita banyak sehingga perlu diolah lebih lanjut.

2.4 Informasi

Menurut Prahasta (2002, p30) informasi adalah makna atau pengertian yang dapat diambil dari suatu data dengan menggunakan konversi-konversi yang umum digunakan di dalam representasinya. Dengan demikian informasi dapat menjadi masukkan yang berguna dalam pengambilan keputusan.

2.5 Database

2.5.1 Definisi Database

Menurut Prahasta (2002, p190) *database* adalah himpunan kelompok data (file/arsip) yang saling berhubungan dan terorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah. Dengan demikian *database* merupakan salah satu komponen yang penting dalam sistem karena dapat menyediakan informasi bagi pemakai.

2.5.2 List Database

List Database adalah database yang datanya berupa deretan kata yang masing-masing datanya memiliki keunikan sendiri.

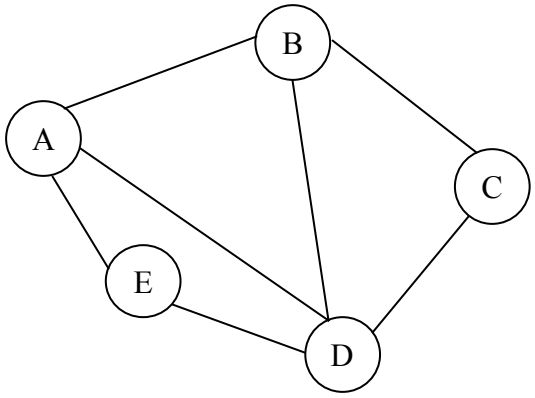
2.6 Representasi Data

Representasi data adalah suatu cara yang digunakan agar data-data yang ada dapat dideskripsikan sehingga dapat dimengerti oleh sistem komputer dan lebih mudah diolah.

2.6.1 *Graph*

Graph adalah suatu struktur data yang direpresentasikan dalam bentuk *network*/jaringan dimana hubungan antar elemen-elemennya adalah *many-to-many relations*.

Menurut Wiitala (1987, p178), *graph* adalah sebuah pasangan yang berurutan dari (v,e) dimana v adalah sekumpulan *vertex*/node dan e adalah kumpulan dari *edge* atau kumpulan dari garis yang menghubungkan antara *vertex* yang satu dengan *vertex* yang lain.

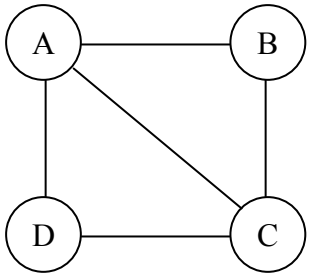


Gambar 2.1 *Graph*

Graph dapat dibedakan menjadi 2 tipe, yaitu :

- *Undirected Graph*

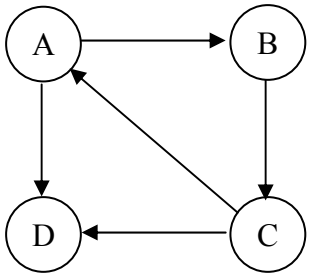
  Beberapa node yang dihubungkan oleh *edge* sehingga membentuk *graph* dan tidak mempunyai arah.



Gambar 2.2 *Undirected Graph*

- *Directed Graph*

  Beberapa node yang dihubungkan oleh *edge* sehingga membentuk *graph* dan mempunyai arah.



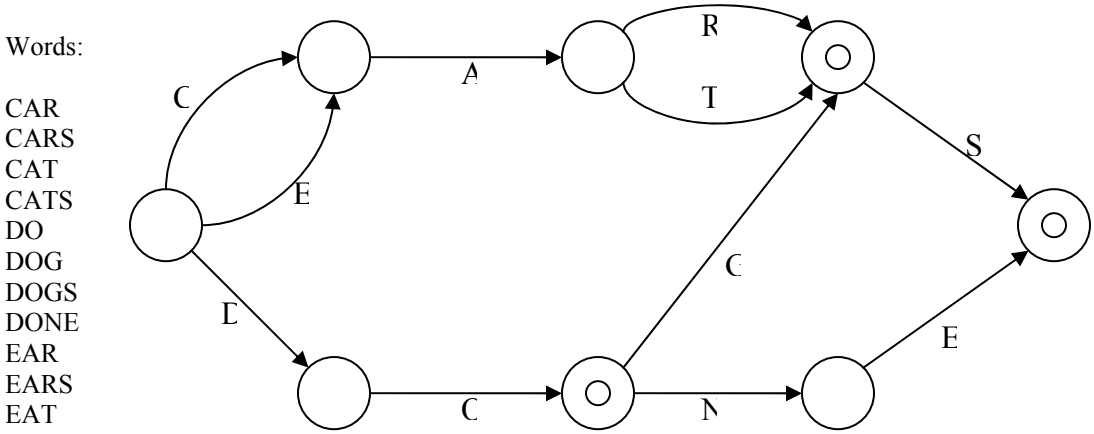
Gambar 2.3 *Directed Graph*

Special Graph adalah *graph* yang memiliki ciri khas/tertentu yang khusus antara *edge* dan *vertex*nya, beberapa diantaranya adalah :

2.6.2 *Directed Acyclic Graph*

Dalam teknologi informatika dan matematika, directed acyclic graph, atau yang disebut juga DAG, adalah suatu directed graph tanpa directed cycles (Graph untuk setiap vertek v dimana tidak ada directed path yang tidak kosong mulai dan berakhir di v). Contoh, jika edge u-v menyatakan bahwa v adalah bagian dari u, maka jalan seperti itu menyatakan bahwa u adalah bagian dari dirinya sendiri, dimana hal itu tidak mungkin.

Setiap DAG berhubungan dengan order parsial pada verteknya, dimana u <= v dalam order parsial tepat pada saat dimana ada directed path dari u ke v pada graph. Tapi, banyak DAG yang berbeda merepresentasikan order parsial yang sama. Pada graph-graph tersebut, graph yang jumlah edgenya paling sedikit adalah pengurangan transitif dan graph yang jumlah edgenya paling banyak adalah closure transitif. DAG yang isinya berupa huruf dapat kita sebut *Directed Acyclic Word Graph* (DAWG).



Gambar 2.4 DAWG

2.7 Algoritma *Sorting*

Dalam dunia Teknolgi Informatika dan Matematika, dikenal suatu metode pengurutan yang disebut *sorting*. Sorting adalah suatu metode mengurutkan sederetan bilangan atau karakter secara teratur. Ada 2 jenis urutan dalam sorting yaitu diurutkan secara ascending atau descending. Secara Ascending berarti sederetan bilangan atau karakter tersebut diurutkan dari yang kecil terlebih dahulu hingga yang paling besar. Sedangkan secara descending berarti diurutkan dari yang paling besar hingga yang paling kecil.

Algoritma sorting sederhana ada 5 jenis yaitu *Selection Sort*, *Bubble Sort*, *Insertion Sort*, *Merge Sort*, dan *Quick Sort*. Algoritma-algoritma sorting ini masing-masing memiliki karakteristik dan sifatnya masing-masing dilihat dari jumlah iterasi dan waktu yang digunakan untuk mencapai keadaan urutan teratur yang diinginkan.

2.7.1 *Selection Sort*

Selection sort adalah algoritma sorting yang paling sederhana. Algoritma *selection sort* adalah dengan membandingkan satu persatu bilangan atau karakter urutan pertama dengan kedua dan seterusnya. Algoritma dari *selection sort* adalah:

1. Cari nilai minimum dari list.
2. Tukar nilai terkecil tersebut ke posisi pertama.
3. Ulangi langkah 1 dan 2 hingga list sudah terurut (mulai dari posisi ke-2).

Contoh:

31 25 12 22 11

11 25 12 22 31

11 12 25 22 31

11 12 22 25 31

2.7.2 *Insertion Sort*

Insertion sort adalah algoritm *sorting* yang menyediakan satu tempat kosong. *Sorting* jenis ini kurang efisien pada list dengan jumlah bilangan atau karakter dalam jumlah besar dibandingkan dengan algoritma yang lebih maju seperti *heap sort*, *quick sort*, atau *merge sort*. Tapi *insertion sort* memiliki kelebihannya sendiri:

- Mudah diimplementasikan.
- Efisien terhadap jumlah data yang sedikit.
- Efisien terhadap data yang sebagian sudah terurut.
- Lebih efisien dalam latihan daripada algoritma lainseperti selection atau *bubble sort*.
- Stabil (Tidak mengubah urutan relatif elemen-elemen dengan nilai sama).
- *In-place* (hanya membutuhkan satu tempat kosong dalam memory).
- Merupakan algoritma online, dapat langsung dilakukan *sorting* pada saat *list* diterima.

Algoritma *Insertion sort*:

1. Array dari nilai-nilai yang akan diurutkan dibagi menjadi 2 bagian : bagian yang sudah terurut dan yang belum terurut.
2. Pada setiap iterasi, elemen pertama pada bagian yang belum terurut diambil dan ditaruh pada posisi yang benar dari bagian yang sudah terurut.
3. Proses sorting berlangsung hingga elemen-elemen dalam bagian yang belum terurut habis.

Contoh *insertion sort*:

	5	2	4	3	1
2	5		4	3	1
	2	5	4	3	1
4	2	5		3	1
	2	4	5	3	1
3	2	4	5		1
	2	3	4	5	1
1	2	3	4	5	
	1	2	3	4	5

Gambar 2.5 *Insertion Sort*

2.8 Algoritma *Searching*

Dalam Teknologi Informatika, algoritma *searching* adalah algoritma yang mengambil suatu masalah sebagai input dan mengembalikannya sebagai solusi untuk masalah tersebut, umumnya setelah memeriksa sekumpulan kemungkinan solusi. Kebanyakan dari algoritma yang dipelajari oleh para ahli yang menyelesaikan masalah algoritma *search*. Kumpulan dari semua kemungkinan solusi untuk masalah disebut ruang pencarian (*search space*). Algoritma pencarian tanpa informasi (*uninformed search algorithm*) menggunakan metode pencarian paling sederhana pada ruang pencarian, sedangkan algoritma pencarian dengan informasi menggunakan heuristik untuk mendapatkan pengetahuan tentang struktur dari ruang pencarian untuk mencoba mengurangi waktu pencarian.

Secara umum algoritma pencarian jalan dibagi menjadi 2 bagian, yaitu :

a. Uninformed Search Algorithm

Uninformed Search Algorithm adalah algoritma yang tidak memiliki keterangan tentang jarak atau biaya dari path dan tidak memiliki pertimbangan akan path mana yang lebih baik. Algoritma ini hanya dapat membedakan yang mana goal dan yang mana bukan goal. *Algoritma Uninformed Search* juga disebut sebagai algoritma pencarian buta (*blind search*).

b. Informed Search Algorithm

Informed Search Algorithm adalah algoritma yang memiliki keterangan tentang jarak atau biaya dari path dan menggunakan pertimbangan berdasarkan pengetahuan dan path mana yang lebih baik.

2.9 Simulasi

Simulasi adalah pemalsuan dari suatu benda, keadaan, atau proses asli dari suatu hal. Dalam teknologi informatika, simulasi memiliki arti khusus: Alan Turing menggunakan istilah "simulasi" untuk menjelaskan yang terjadi saat komputer digital mengeksekusi suatu program yang menjelaskan keadaan transisi, input dan output. Dalam pemrograman komputer, simulasi sering digunakan untuk mengeksekusi program yang hanya dapat dijalankan dalam komputer tertentu atau dalam keadaan tertentu.

2.9.1 MINIMAX

Minimax (terkadang disebut minmax) adalah metode pada tori keputusan untuk meminimalisasi kemungkinan kerugian maksimal. Atau dengan kata lain, minimax memaksimalkan keuntungan minimum (maximin).

Contoh sederhana dari algoritma minimax dapat dilihat dari permainan tic-tac-toe, dimana setiap pemain dapat menang, kalah atau seri. Jika pemain A dapat menang dengan 1 langkah lagi, maka langkah terbaiknya adalah langkah untuk menang tersebut. Jika pemain B tahu bahwa suatu langkah tertentu dapat menyebabkan situasi dimana pemain A dapat menang dengan satu langkah, sedangkan langkah lain menyebabkan situasi dimana pemain A paling baik hanya seri, maka langkah terbaik pemain B adalah yang mengarah pada hasil seri. Pada akhir permainan, mudah untuk mencari langkah terbaik. Algoritma Minimax membantu mencari langkah terbaik tersebut dengan mengerjakan secara terbalik (backward) dari akhir permainan. Setiap langkah diasumsikan pemain A berusaha

memaksimalkan kesempatan pemain A untuk menang, sementara pada giliran berikutnya, pemain B berusaha meminimalkan kemungkinan pemain A untuk menang (juga untuk memaksimalkan kesempatan pemain B untuk menang).

2.9.2 Nega-Max *Function*

Nega-max adalah optimisasi dari MINIMAX. Setelah nilai hasil evaluasi pertama dihasilkan, nilai tersebut dinegasikan dan diteruskan ke node yang lebih bawah. Negasi dilakukan karena nilai tersebut akan dilihat dari sisi pandang yang berbeda (musuh). Jika suatu nilai evaluasi dinegasikan yang kemudian dilakukan fungsi evaluasi dan kemudian dinegasikan kembali, maka nilai yang didapat adalah nilai dari MINIMAX 2 ply. Fungsi ini memiliki langkah penelusuran node yang sama seperti MINIMAX dan memiliki prosedur dan hasil yang sama. Optimisasi yang dilakukan adalah optimisasi dari coding yang ditulis, yang berarti mengoptimisasi kinerja proses, memperkecil *bug* dari program yang akan terjadi, dan mempermudah pemeliharaannya (*maintenance*).

2.9.3 Alpha Beta Cutoff

Pada MINIMAX, terdapat istilah *pruning* yaitu untuk mengurangi jumlah keadaan yang harus diuji untuk menentukan nilai dari suatu pohon pelacakan. Kita dapat menyimpan nilai batas bawah pada node yang melakukan maksimasi, dan kita tidak perlu menghiraukan cabang-cabang yang tidak akan memperbaiki batas tersebut (lebih tinggi). Demikian pula kita dapat menyimpan batas atas dari node yang melakukan minimasi, dan kita juga tidak perlu menghiraukan cabang-cabang

yang tidak akan memperbaiki batas tersebut (lebih rendah). Variabel alpha (α) digunakan sebagai batas bawah node yang melakukan maksimasi, sedangkan variable (β) digunakan sebagai batas atas bagi node yang melakukan minimasi. Pada node-node yang melakukan minimasi, evaluasi akan dihentikan jika sudah didapat node anak yang memiliki nilai lebih kecil disbanding dengan batas bawah (α), sedangkan pada node-node yang melakukan maksimasi, evaluasi akan dihentikan jika sudah didapat node anak yang memiliki nilai lebih besar dibanding dengan batas atas (β).

Pada akar pohon pencarian, nilai α diberikan nilai sebesar mungkin sedangkan nilai β diberikan nilai sekecil mungkin. Node-node yang melakukan maksimasi dan minimasi akan memperbaiki nilai α dan β nya masing-masing.

2.10 *Backtracking*

Algoritma *backtracking* mencoba setiap kemungkinan hingga menemukan tujuannya. Algoritma *backtracking* menggunakan *Depth-First-Search* dari serangkaian solusi-solusi yang memungkinkan. Saat pencarian, jika suatu alternatif tidak berhasil, pencarian kembali (*backtrack*) ke titik pilih (choice *point*), titik yang memiliki alternatif lain, dan mencoba alternatif berikutnya. Saat semua alternatif sudah dicoba semua, pencaran kembali ke titik pilih sebelumnya dan mencoba alternatif dari titik tersebut. Jika tidak ada lagi titik pilik yang dapat dicoba, maka pencarian gagal (*fail*).

Algortima seperti ini biasanya dilakukan dalam fungsi rekursif dimana setiap instance mengambil 1 variabel lebih dan secara alternatif mengatur semua

nilai yang tersedia padanya, menjaga agar variabel tersebut konsisten dengan pemanggilan recursive yang subsequent. Backtracking mirip dengan *Depth-first-Serach* tapi menggunakan lebih sedikit space, menyimpan 1 state solusi pada saat itu dan meng-update-nya.

Untuk mempercepat pencarian, saat suatu nilai dipilih, sebelum membuat pemanggilan recursive, algoritmanya menghapus nilai dari domain-domain yang belum diatur yang saling berlawanan (*forward checking*) atau mengecek semua halangan (*constraint*) untuk melihat apakah nilai-nilai lain di luar nilai yang baru diatur (*constraint propagation*). Ini merupakan teknik paling efisien untuk beberapa masalah seperti 0/1 *knapsack* dan n-queen problem. Teknik ini memberikan hasil yang lebih baik dari dynamic proggraming untuk masalah-masalah seperti ini.

2.11 Anagram

Kata ‘anagram’ berasal dari bahasa Yunani *ana* yang berarti kembali atau lagi dan *graphien* yang berarti menulis. Anagram adalah suatu tipe permainan kata, hasil dari perombakkan huruf-huruf dari kata atau frase untuk menghasilkan kata-kata lain, dengan menggunakan semua huruf-huruf asalnya sekaligus. Seseorang yang menciptakan anagram disebut anagramist.

Anagram seringkali diekspresikan dalam bentuk persamaan, dengan simbol ekual (=) memisahkan subjek awal dan hasil anagramnya. ‘Earth = Heart’ adalah salah satu contoh ekspresi anagram menggunakan simbol ‘=’.

Dalam permainan *scrabble*, pemain harus membuat kata-kata dengan menaruh tiles pada board untuk menghasilkan poin lebih banyak dari lawan pada akhir permainan. Seorang pemain *scrabble* harus melakukan anagraming terhadap tiles yang dimilikinya. Versi lain dari *scrabble* yang bernama *Clabbers*, bahkan dari namanya saja merupakan anagram dari *scrabble*, memperbolehkan tiles diletakkan dimanapun pada board selama mereka merupakan anagram dari kata yang valid.

2.12 Probabilitas

Probabilitas adalah besarnya kemungkinan suatu hal untuk terjadi. Dalam matematika, probabilitas memiliki range nilai antara 0 hingga 1. Suatu kejadian yang tidak mungkin terjadi probabilitasnya 0 sedangkan untuk kejadian yang sudah pasti terjadi probabilitasnya 1.

Untuk menghitung besarnya probabilitas secara matematis dapat dibagi menjdai 2 faktor yaitu Ruang Sampel (Himpunan yang memuat semua kemungkinan kejadian) serta Kejadian (himpunan bagian dari ruang contoh). Jadi, probabilitas dapat dirumuskan menjadi:

Prob = E / R

E → *Event* (Kejadian)

R → Ruang Sampel

2.12.1 Permutasi

Permutasi sejumlah objek adalah penyusunan obyek dalam suatu urutan tertentu. Dalam permutasi urutan diperhatikan.

Dalil 1 permutasi adalah banyaknya permutasi n benda yang berbeda adalah n! (faktorial, contoh: 5! = 5x4x3x2x1).

Dalil 2 permutasi adalah banyaknya permutasi r benda dari n benda yang berbeda adalah

$$ {}_nP_r = \frac{n!}{(n-r)!} $$

Dalil 3 permutasi adalah banyaknya permutasi n benda yang berbeda yang diletakkan secara melingkar adalah (n-1)!.

2.12.2 Kombinasi

Kombinasi r objek yang dipilih dari n objek adalah susunan r objek tanpa memperhatikan urutan.

Dalil kombinasi =

$$ C_r^n = \frac{n!}{r!(n-r)!} $$

Perbedaan paling mencolok antara permutasi dan kombinasi dapat dilihat dari contoh berikut:

Ada 3 buah huruf yaitu A, B dan C. Akan dipilih 2 huruf, berapa besar a) Permutasi b) Kombinasi

i. P = 3! / (2-1)!

= 3x2x1 / 1

= 6 / 1

= 6 (AB, AC, BA, BC, CA, CB)

ii. C = 3! / 2! (3-2)!

= 3x2x1 / 2x1 (1)

= 6 / 2

= 3 (AB, AC, BC; AB dan BA dianggap sama)