BAB 2

TINJAUAN PUSTAKA

2.1 AI *Games*

Dengan semakin cepatnya perkembangan *hardware* dan *software* manusia berusaha menciptakan suatu komputer yang memungkinkan untuk memahami, menalar dan bertindak sehingga dapat mengerjakan sesuatu yang pada saat ini orang dapat mengerjakan secara baik dalam kehidupan sehari-hari melalui *Artificial Intellegence* (AI). AI merupakan suatu cara yang sederhana untuk membuat *computer* dapat berfikir secara *intelligent*, yaitu suatu *system* yang memperlihatkan karakteristik yang ada pada tingkah laku manusia seperti mengerti suatu bahasa mempelajari, mempertimbangkan dan memecahkan suatu masalah. Adapun yang menjadi tujuan dasar dari AI adalah Untuk mengembangkan metode dan sistem untuk menyelesaikan masalah,masalah yang biasa diselesaikan melalui aktifivitas intelektual manusia, meningkatkan kinerja sistem informasi yang berbasis *computer*.

Kecerdasan buatan dalam sebuah permainan menjadi salah satu kajian ilmu artificial intelligence yang menarik dalam ilmu komputer. Akan tetapi tidak semua teknik AI bisa diterapkan dalam permainan yang sama. Oleh karena itu supaya tidak terjadi kesalahan dalam penerapan teknik diperlukan pemahaman yang baik dalam karakteristik permainan. AI games merupakan aplikasi untuk memodelkan karakter yang terlibat, baik sebagai lawan ataupun karakter pendukung yang merupakan bagian dari permainan yang tidak bermain.

AI games memiliki karakter tersendiri dalam pengklasifikasiannya. Adapun yang menjadi karakter dalam AI games yaitu :

1. Karakter antagonis (*opponent* AI)

   Karakter antagonis adalah karakter yang dalam permainan memiliki tujuan yang sama dengan pemain yaitu memenangkan permainan. Untuk mewujudkan tujuan ini, karakter tersebut dapat melakukan aksi kepada

pemain dan sebaliknya pemain dapat melakukan aksi kepada pemain lawan yang sesuai dengan aturan permainan. Karakter lain adalah lawan yang harus dikalahkan oleh pemain dengan cara melakukan aksi (penyerangan). Fungsi AI disini adalah pada karakter antagonis yang dapat melakukan aksi-aksi yang dapat memperbesar karakter tersebut untuk menang.

2. Karakter pendukung

Karakter pendukung merupakan karakter yang terlibat dalam permainan tetapi tidak memiliki tujuan untuk memenangkan permainan tetapi melakukan peran yang mendukung pemain atau lawan untuk memenangkan permainan. Sebagai contohnya, pada game tertentu umumnya memberikan informasi kepada karakter mengenai hal-hal yang berkaitan dengan cerita dalam permainan untuk menentukan dialog dengan pemain.

Teori dalam sebuah pertandingan merupakan suatu kajian tentang bagaimana para pemain harus bermain dengan rasional. Setiap pemain berusaha mendapatkan point sebanyak mungkin. Yang menjadi dasar ide dari teori permainan adalah perilaku strategis dari pemain atau pengambil keputusan.

Setiap pemain dianggap mempunyai suatu perencanaan dimana dia bisa memilih kalau kita memiliki suatu set strategi untuk setiap situasi yang muncul dalam suatu permainan yang digunakan untuk memutuskan tindakan yang akan diambil selanjutnya.

Ada beberapa hal yang menentukan permainan, yaitu :

1. *The number of player* (jumlah pemain). Suatu permainan dibangun oleh 2 orang. Bila ada sejumlah *n* pemain yang bermain, maka disebut permainan sejumlah n orang.
2. *The net winning of the game* (keuntungan bersih dari permainan). Suatu permainan *zero-sum* adalah ketika keuntungan bersih sama dengan nol.

3. *Fairness of the game* (sikap adil dalam permainan). Sebuah permainan yang tidak condong ke salah satu pemain dalam memberikan suatu kemenangan.

2.2 *Game* Congklak

Congklak merupakan suatu permainan tradional yang yang dimainkan oleh 2 orang dengan menggunakan media papan. Biasanya dalam permainan ini menggunakan batu atau sejenis kulit kerang yang digunakan sebagai alat permainan [8].

Papan congklak terdiri dari sejumlah lubang kecil dan dual bang besar yang terdapat di masingmasing tiap ujung papan. Setiap pemain memiliki daerahnya sendiri. Pada awal permainan, semua lubang yang ada dipapan diisi dengan jumlah yang lubang yang sama (batu, biji atau kerang). Jumlah isi di setiap lubang biasanya sama dengan jumlah lubang di salah satu sisi. Tujuan dari permainan ini adalah untuk mengumpulkan isi lubang sendiri dan mengosongkan isi lubang lawan yang ada di papan permainan. Pemain yang memenangkan permainan adalah pemain yang mengumpulkan batu / kulit kerang terbanyak.

2.2.1 Sejarah Perkembangan Permainan Congklak

Congklak memiliki asal-usul dalam baik Afrika atau dunia Arab, tergantung pada teori sarjana Anda memilih untuk percaya. Beberapa bukti tertua ditemukan di National Geographic disponsori penggalian arkeologi dating kembali ke 7.000 menjadi 5.000 SM di zaman sekarang Yordania. Penggalian sebuah rumah menemukan sebuah lempengan batu kapur dengan dua baris paralel depresi melingkar. Tata letak adalah mudah dikenali oleh seorang arkeolog pada penggalian sebagai dewan bermain Congklak. Murray, seorang sarjana mencatat, menelusuri asal-usul untuk Usia Kekaisaran Mesir kuno (sekitar 15 sampai abad 11 SM). Banyak ahli menduga bahwa Congklak mungkin sebenarnya papan permainan tertua yang pernah ada.[8]

Tulisan-tulisan paling awal tercatat menggambarkan permainan yang ditemukan di referensi dalam teks-teks agama Mancala Arab ke Abad Pertengahan. Beberapa sarjana percaya bahwa permainan berasal dari Timur Tengah dan menyebar dari sana ke Afrika. Kemudian, permainan menyebar ke Asia dengan pedagang Arab dan datang ke Karibia sekitar 1.640 melalui perdagangan budak Afrika. Ahli lain menempatkan asal-usul di Afrika Tengah.[8]

Saat ini, *game* ini dikenal dengan berbagai nama di seluruh dunia. Nama-nama yang diambil dari budaya lokal dengan menggunakan kata-kata yang mencerminkan di mana permainan ini dimainkan, cara untuk menang, modus dari bermain dan papan atau counter digunakan. Hal ini disebut dalam bahasa Inggris saat *Count* dan *Capture*.[8]

Di negara-negara Arab, nama yang paling umum adalah Mancala (makna kata Arab dalam bahasa Inggris "bergerak"). Di beberapa negara Afrika Barat yang depresi di papan disebut sebagai Warri atau Awari, yang berarti rumah, sehingga memberikan yang Wari nama. Di Nigeria permainan dikenal sebagai Adi, yang juga merupakan nama dari bibit yang digunakan untuk bermain game. Karena popularitas yang luasnya, para penggemar telah mengembangkan situs web banyak mendokumentasikan berbagai versi dari permainan, untuk daftar link yang baik untuk banyak situs-situs, lihat http://www.cs.uu.nl/~hansb/d.gam/mancala.html. Lain situs web yang baik, http://www.myriad-online.com/en/resources/awalink.html, daftar 279 nama dari permainan yang ditemukan melalui penelitian mereka. Ketika Anda surfing melalui situs-situs web, itu menarik untuk melihat bagaimana masing-masing negara atau budaya memiliki adat istiadat yang berkaitan dengan pembuatan papan, yang bisa bermain, serta bagaimana dan kapan permainan ini dimainkan.[8]

Beberapa fitur situs on-line versi dari permainan juga, jika Anda ingin bermain dengan penggemar lain di seluruh dunia. Pada topik congklak online, telah menyarankan bahwa game congklak terinspirasi mungkin mulai muncul di

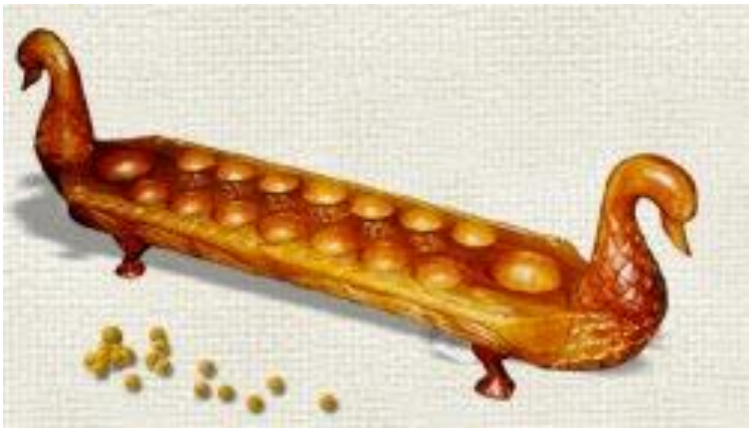
kasino online, yang sudah menawarkan permainan tradisional seperti sic bo dan Baccarat.

Permainan ini sangat populer sehingga Museum British Museum Manusia menampilkan sebuah pameran Mancala, Wari dan papan Congklak lain bermain pada tahun 1997.[8]

2.2.2 Congklak Di Indonesia

Bahkan di Indonesia, Congklak dikenal dengan nama yang berbeda dari daerah ke daerah. Nama yang paling umum, Congklak, diambil dari shell cowrie, yang biasa digunakan untuk bermain game. Di Malaysia, permainan ini dikenal sebagai Congkak, nama yang digunakan dalam provinsi Sumatera juga banyak. Di Jawa, permainan ini dikenal sebagai Congklak, dakon, dhakon atau dhakonan.Di Lampung, permainan ini disebut, dentuman lamban. Di Sulawesi, permainan ini disebut sebagai Mokaotan, Maggaleceng, Aggalacang dan Nogarata.[8]

Referensi historis untuk congklak merujuk pada permainan yang dimainkan oleh gadis-gadis muda dari bangsawan Jawa.Hal ini paling mungkin bahwa pedagang asing, karena kontak dekat mereka dengan kelas atas, diperkenalkan Congklak kepada mereka. Dengan berlalunya waktu, popularitas congklak tumbuh sampai sekarang yang banyak dimainkan oleh rakyat biasa juga. Di sebagian besar wilayah, Congklak bermain terbatas pada gadis-gadis muda, remaja dan wanita di waktu luang mereka dan dilihat sebagai 'permainan wanita'. Dalam beberapa daerah adalah Congklak dimainkan oleh laki-laki dan anak laki-laki juga. Berikut gambar dari papan Congklak.



Gambar 2.1. Papan *Game* Congklak

Di Sulawesi, historis, permainan disediakan untuk bermain hanya selama periode berduka, setelah kematian orang yang dicintai. Ini dianggap tabu untuk bermain game pada waktu lainnya. Di Jawa Tengah, pada masa pra-sejarah kali, Congklak digunakan oleh petani untuk menghitung musim, untuk mengetahui kapan harus menanam dan panen, serta untuk memprediksi masa depan.[8]

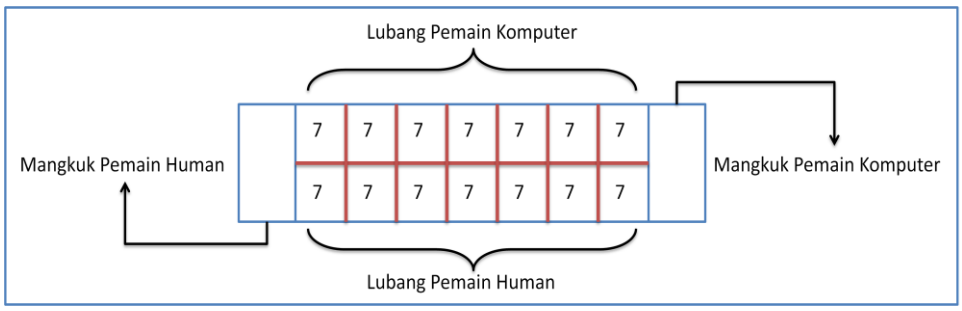
2.2.3 Aturan Permainan Congklak

Permainan dimulai dengan menyetujui siapa yang akan mulai. Para pemain memilih salah satu lubang di sisi sendiri dengan yang berisi setiap lubang. Pemain mengambil biji dari lubang dan menempatkan mereka satu-per-satu di lubang sebelah lubang awal, dengan cara berlawanan arah. Penuhi setiap lubang dengan biji yang dipilih (dengan diasumsikan bahwa setiap pemain sepakat untuk memilih tiga sampai tujuh biji dan tiga sampai tujuh lubang untuk dimainkan). Jangan menempatkan biji-biji tersebut di mangkuk (lubang yang besar), karena mangkuk adalah lubang induk besar yang terletak diujung paling kiri dan kanan di lubang papan pemain. Jika counter (biji dalam benggaman) terakhir dimasukkan ke dalam mangkok pemain, pemain bisa memilih lubang baru dari sisi sendiri dan bergerak lagi. Ketika lubang terakhir dari langkah tersebut dimasukan ke dalam lubang kosong di sisi pemain, maka lubang papan yang berlawanan ditangkap dan dimasukkan ke dalam mangkok dan giliran pemain berakhir. Tetapi ketika *counter*

terakhir dimasukkan ke dalam lubang kosong di sisi lawan, maka giliran permain berakhir tanpa mendapatkan nilai karena berakhir tanpa penangkapan.[8]

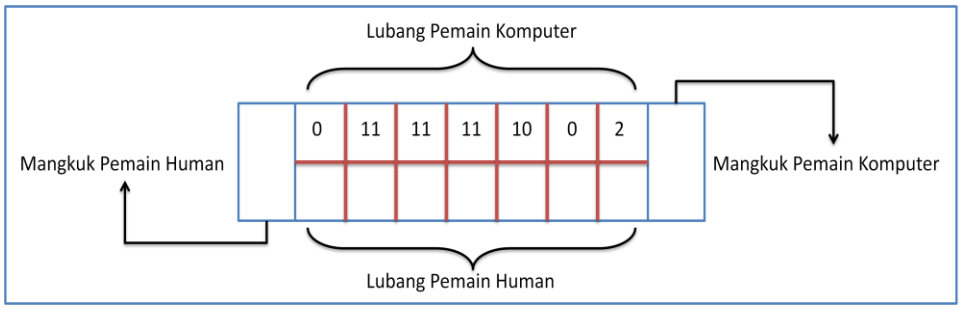
Memulai permainan

1. Permainan di mulai dengan menentukan siapa yang bermain.



Gambar 2.2 Papan Congklak pada posisi awal

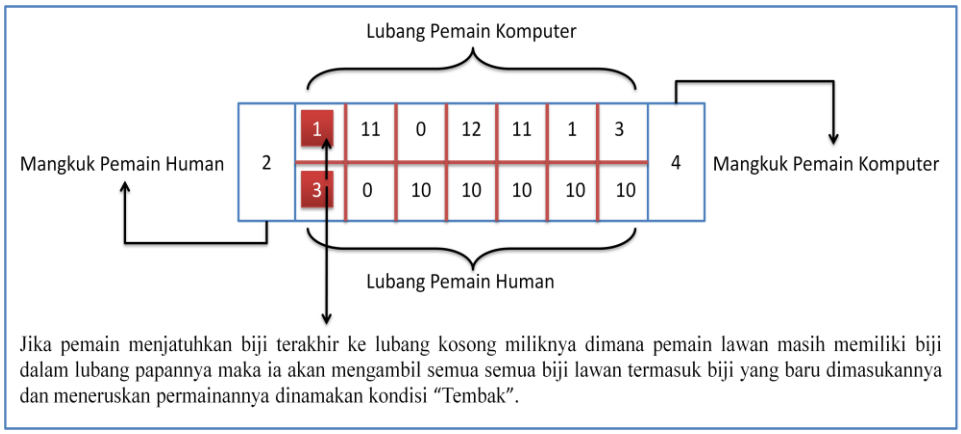
2. Semua pemain harus mengambil semua biji yang terdapat pada salah satu lubang papan miliknya dan menjatuhkannya satu demi satu biji ke setiap lubang papan pemain yang dilaluinya
3. Pada setiap putaran, masing-masing pemain harus menjatuhkan satu biji ke mangkuk miliknya tetapi ia tidak boleh menjatuhkan biji ke mangkuk (lubang yang besar pada papan) lawannya.
4. Apabila pemain menjatuhkan biji terakhir di mangkuk maka pemain diijinkan untuk mengambil biji kembali di lubang milik pemain.



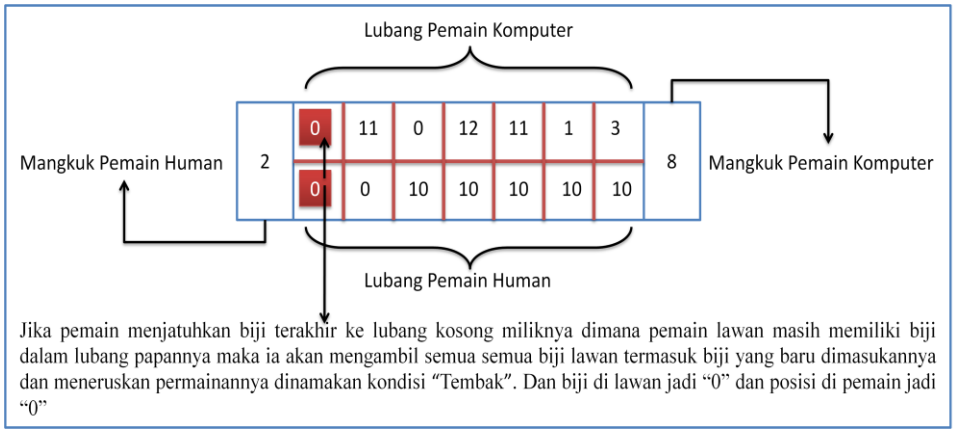
Gambar 2.3 Papan Congklak Setelah Biji Dimainkan

Awal dari giliran silih berganti

1. Setiap pemain memulai gilirannya dengan mengambil seluruh biji yang ada pada salah satu dari salah satu lubang miliknya dan dan menjatuhkannya satu demi satu ke lubang papan yang dilaluinya.
2. Jika pemain menjatuhkan biji terakhir ke lubang kosong miliknya dimana pemain lawan masih memiliki biji dalam lubang papannya maka ia akan mengambil semua semua biji lawan termasuk biji yang baru dimasukannya dan meneruskan permainannya dinamakan kondisi "Tembak".

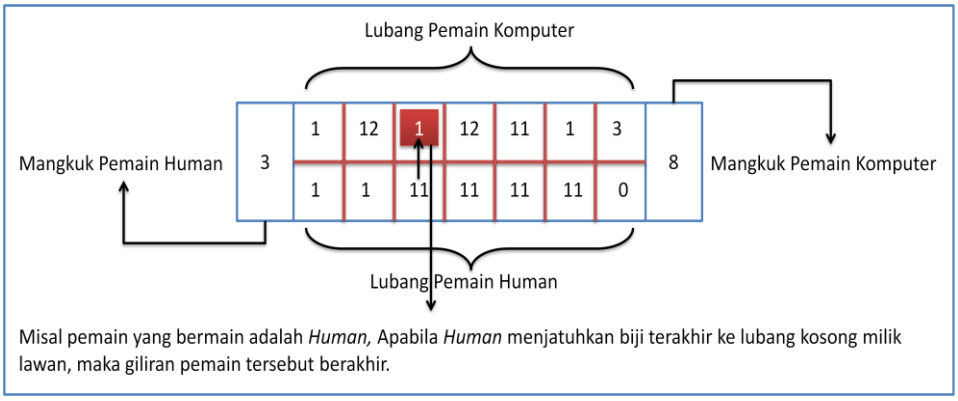


Gambar 2.4 Papan Congklak Ketika saat Terjadi Tembak



Gambar 2.5 Papan Congklak Setelah Terjadi Tembak

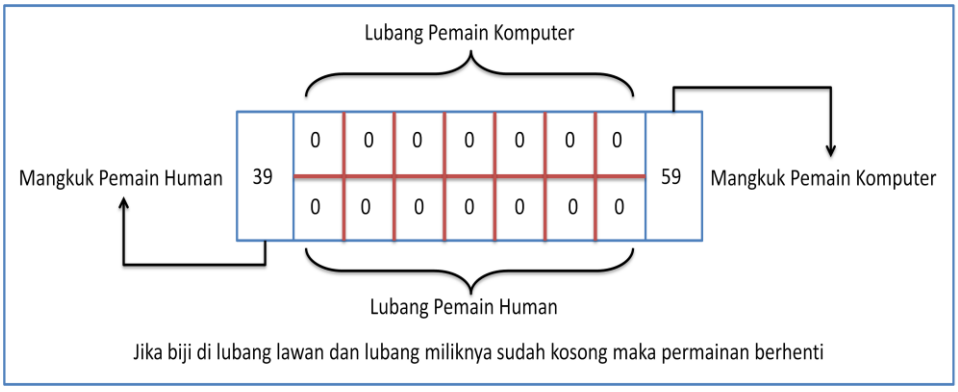
3. Apabila salah seorang pemain menjatuhkan biji terakhir ke lubang kosong milik lawan, maka giliran pemain tersebut berakhir.



Gambar 2.6 Papan Congklak ketika biji terakhir di lubang lawan

Akhir Permainan :

1. Jika biji di lubang lawan dan lubang miliknya sudah kosong maka permainan berhenti.



Gambar 2.7 Papan Congklak untuk Akhir Permainan

2. Pemain yang berhasil mendapatkan biji paling banyak di mangkuknya berarti pemain tersebut menang biji.

2.3 Konsep Dasar Algoritma

Sebuah algoritma merupakan deskripsi-deskripsi langkah-langkah pelaksanaan suatu proses. Sebuah proses dikerjakan oleh pemrosesan berdasarkan algoritma yang diberikan. Setiap langkah penyelesaian dinyatakan dengan sebuah pernyataan yang menggambarkan suatu aksi. Bila suatu aksi dieksekusi, maka sejumlah operasi yang sesuai akan dikerjakan oleh pemroses dimana efek dari pengerjaan suatu aksi dapat diamati dengan melihat perbandingan keadaan pada saat aksi tersebut dimulai dan selesai dikerjakan.

Algoritma berasal dari kata algorism, menurut para ahli bahwa kata algorism berasal dari nama penulis buku arab yang terkenal, yaitu Abu Ja'far Muhammad ibnu musa al-Khuwarizmi (al-Khuwarizmi dibaca orang barat menjadi algorism) Al-Khuwarizmi menulis buku yang berjudul kitab al-jabar wal muqabala, yang artinya "buku pemugaran dan pengurangan" (*the book of restoration and reduction*). Perubahan kata *algorism* menjadi *algorithm* muncul karena kata *algorism* sering dikelirukan dengan dengan *arithmetic*, sehingga akhiran –sm berubah menjadi –thm. Karena perhitungan dengan angka arab sudah menjadi hal yang biasa, maka lambat laun kata algorithm berangsurangsur digunakan sebagai metode perhitungan (komputasi)secara umum, sehingga makna aslinya menjadi hilang.[9]

Dalam kecerdasan buatan (AI) terdapat empat teknik pemecahan masalah yaitu, *searching*, *reasoning*, *planning* dan *learning*. Keempat metode tersebut memiliki metode masing-masing dalam penyelesaian masalah. Penggunaan teknik dan metode tergantung kepada permasalahan yang akan diselesaikan. Suatu teknik dengan metode tertentu sangat bagus untuk masalah A tetapi belum tentu sesuai untuk masalah B atau C.

2.4 Analisis Algoritma

Sebuah algoritma tidak saja harus benar, tetapi juga harus mangkus (*efisien*), algoritma yang bagus adalah algoritma yang mangkus. Kemangkusan

algoritma diukur dari berapa jumlah waktu dan ruang (*space*) memori yang dibutuhkan untuk menjalankannya. Algoritma yang mangkus ialah algoritma yang meminimumkan kebutuhan waktu dan ruang. Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan (n), yang menyatakan jumlah data yang diproses. Kemangkusan algoritma dapat digunakan untuk menilai algoritma yang terbaik. Model kebutuhan perhitungan waktu dapat dilihat dari beberapa faktor

1. Dapat mengukur waktu yang diperlukan oleh sebuah algoritma dengan menghitung banyaknya operasi/instruksi yang dieksekusi.
2. Jika mengetahui besaran waktu (dalam satuan detik) untuk melaksanakan sebuah operasi tertentu, maka kita dapat menghitung berapa waktu sesungguhnya untuk melaksanakan algoritma tersebut.
3. Tergantung pada kecepatan komputer, kualitas program, dan *compiler.*
4. *Running time* dihitung berdasarkan operasi dasar.

Analisis algoritma dilakukan untuk mengetahui seberapa baik sebuah algoritma diimplementasikan untuk menyelesaikan suatu kasus. Penilaian terhadap algoritma melibatkan analisis algoritma sehingga terdapat 2 (dua) tipe analisis algoritma :

1. Aspek kualitatif

Aspek kualitatif yaitu analisis untuk memeriksa kebenaran algoritma. Analisis dengan aspek kualitatif dilakukan dengan penelusuran algoritma, dilakukan penelusuran logik menggunakan teknik matematika untuk membuktikan kebenaran atau implementasi algoritma atau mengujinya dengan data. Sebagai contoh adalah algoritma pengurutan data (*sorting*) tidak dapat disebut sebagai algoritma pengurutan jika algoritma tidak dapat mengurutkan sembarang masukan barisan data.

2. Aspek kuantitatif

Aspek kuantitatif yaitu analisis terhadap efisiensi algoritma. Aspek kuantitatif dilakukan dengan melakukan perhitungan kompleksitas komputasi (waktu) dan ruang. Aspek kuantitatif mencoba mengukur seberapa besar sumber daya yang diperlukan suatu algoritma 2 (dua) sumber daya yang diukur adalah kecepatan bekerja algoritma dan ruang yang diperlukan untuk bekerja. Notasi untuk menyatakan kinerja antara lain adalah big-oh (O) yaitu waktu komputasi algoritma berbanding terhadap satu fungsi tertentu. Big-O biasanya hanya dinyatakan dengan suku yang paling berarti dan menghilangkan konstanta pengalinya. Jadi O((n2-n)/2) hanya dinyatakan dengan O(n2).

Adapun tiga cara yang dapat dilakukan dalam melakukan analisis algoritma, yaitu :

1. Analisis untuk memeriksa kebenaran algoritma.
2. Analisis efisiensi algoritma (kompleksitas komputasi dan ruang).
3. Analisis optimalitas algoritma.

Berikut akan dijelaskan satu persatu cara untuk menganalisis performansi algoritma.

2.4.1 Analisis Kebenaran Algoritma

Untuk menguji kebenaran suatu algoritma maka hal yang dapat dilakukan antara lain :

1. Penelusuran algoritma
2. Penelusuran logik (*assertion*)
3. Implementasi algoritma
4. Pengujian dengan data
5. Atau menggunakan teknik matematika untuk pembuktian kebenaran

Analisis kebenaran algoritma juga sering dimasukkan sebagai proses validasi algoritma.

2.4.2 Analisis Efisiensi Algoritma

Dalam analisis efisiensi algoritma terdapat 2 (dua) fase, yaitu :

1. A priori analysis

Fase ini bertujuan untuk menemukan fungsi (beserta parameter-parameter yang relevan) yang membatasi waktu komputasi algoritma.

Adapun notasi matematika yang digunakan untuk menunjukkan hasil :

a. O-notation (Big-O)
b. Ω-notation
c. Θ-notation

2. *A posteriori testing*

Pada fase ini dikumpulkan statistik nyata konsumsi waktu dan ruang suatu algoritma pada mesin dan bahasa pemrograman tertentu.

Tujuan dari dilakukannya fase ini dalah untuk :

Menentukan jumlah waktu dan ruang penyimpanan yang diperlukan program.

Kegunaan dari fase ini adalah untuk memvalidasi *a priori analysis*.

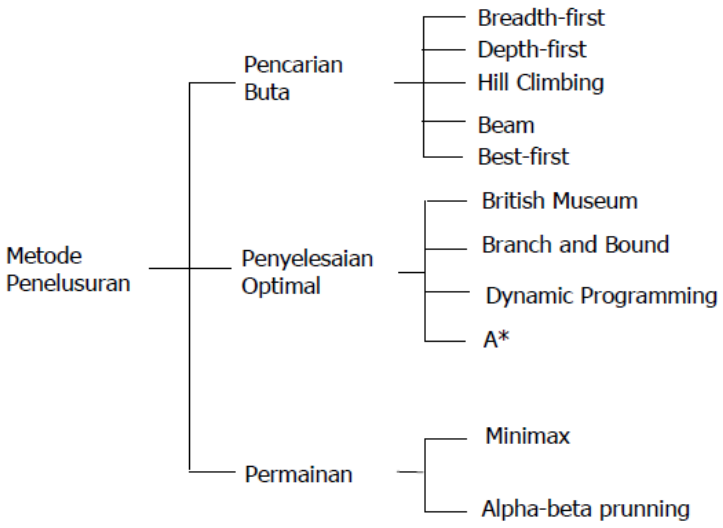
2.5 Algoritma Pencarian

Dalam ilmu komputer, sebuah algoritma pencarian dijelaskan secara luas adalah sebuah algoritma yang menerima masukan berupa sebuah masalah dan menghasilkan sebuah solusi untuk masalah tersebut, yang biasanya didapat dari evaluasi beberapa kemungkinan solusi. Sebagian besar algoritma yang dipelajari

oleh ilmuwan komputer adalah algoritma pencarian. Himpunan semua kemungkinan solusi dari sebuah masalah disebut ruang pencarian. Algortima pencarian brute-force atau pencarian naif/*uninformed* menggunakan metode yang sederhana dan sangat intuitif pada ruang pencarian, sedangkan algoritma pencarian *informed* menggunakan heuristik untuk menerapkan pengetahuan tentang struktur dari ruang pencarian untuk berusaha mengurangi banyaknya waktu yang dipakai dalam pencarian.

2.5.1 Macam-Macam Algoritma Pencarian

Ada 3 bagian dalam metode pencarian yaitu yang pertama adalah metode yang sederhana yang berusaha mencari kemungkinan penyelesaian. Yaitu metode deep-first search, hill climbing, breadth-first search, beam search dan best-first search. Kemudian metode yang kedua adalah metode yang lebih kompleks dalam pencarian jarak terpendek melalui metode British Museum Procedure, Branch and Bound, Dynamic Programming, dan A*. dan yang terakhir adalah metode yang diterapkan ketika berhadapan dengan musuh. Dimana metode ini sering digunakan dalam program permainan tic-tac toe, catur dll yaitu Minimax Search dan Alpha Beta Prunning.



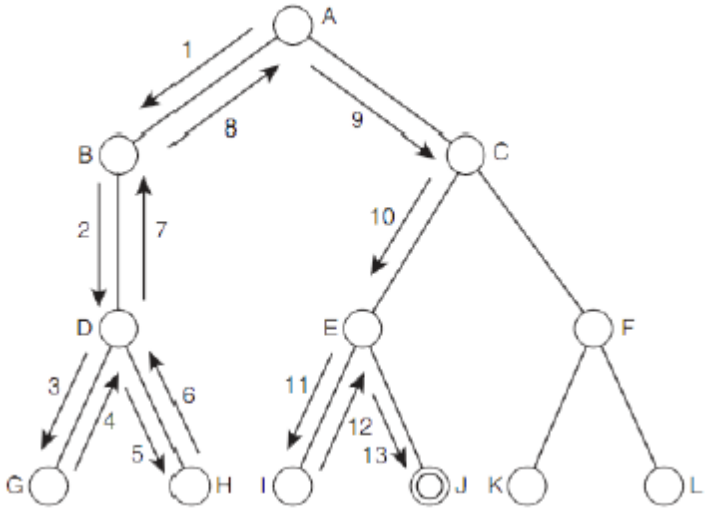
Gambar 2.8 Metode Searching Algoritma

Metode pencarian sangat penting untuk perencanaan karena dalam sebuah permainan akan menentukan apa yang harus dilakukan dimana setiap state (keadaan) yang menggambarkan suatu posisi dalam penyelesaian masalah. Mendefinisikan masalah sebagai suatu ruang keadaan yang dideskripsikan melalui tahapan-tahapan berikut, yaitu :

1. Mendefinisikan suatu ruang keadaan
2. Menerapkan satu atau lebih keadaan yang awal
3. Menetapkan satu atau lebih tujuan
4. Menetapkan aturan kumpulan

2.5.1.1 DFS(Depth First Search)

Depth-First Search (DFS) merupakan algoritma pencarian yang paling umum digunakan. DFS akan melakukan pencarian pada pohon dengan cara menelusuri suatu jalur tertentu hingga mencapai kedalaman yang paling akhir (leaf node) terlebih dahulu sebelum melanjutkan ke jalur selanjutnya. Jika DFS telah mencapai leaf node tetapi belum menemukan goal state maka DFS akan melakukan proses backtrack ke node teratas yang belum pernah ditelusuri. Penelusuran pada pohon permainan akan dilakukan hingga DFS menemukan goal state atau telah selesai menelusuri keseluruhan pohon permainan. Pada Gambar 2.9 mengilustrasikan tentang bagaimana cara kerja algoritma DFS.



Gambar 2.9 penelusuran pohon permainan dengan DFS(Ben Coppin, 2004)

Pada Gambar 2.9 dapat diketahui bahwa proses penelusuran DFS dimulai dari A-B-D-G-D-H-D-B-A-C-E-I-J. A merupakan root yang menandakan keadaan belum diambil, lalu ditelusuri hingga kedalaman yang paling dalam sebelah kiri yaitu G, lalu melakukan proses backtrack ke D lalu lanjut lagi ke H. Proses tersebut berhenti karena telah mencapai *goal state* yaitu *leaf node* G.

2.5.1.2 Algoritma Minimax

Algoritma minimax merupakan salah satu pencarian yang dapat diterapkan untuk pencarian solusi dalam suatu game. Minimax pertama kali dikembangkan oleh John Von Neumann pada tahun 1944 dengan menguraikan sebuah algoritma search pada game yang dikenal dengan *Minimax* atau *Minimize the maximum possible loss*. Minimax merupakan algoritma yang melihat beberapa langkah ke depan, dengan cara melakukan pencarian pada pohon permainan yang menyimpan nilai pada setiap simpul.

Algoritma minimax dibuat berdasarkan teorema :

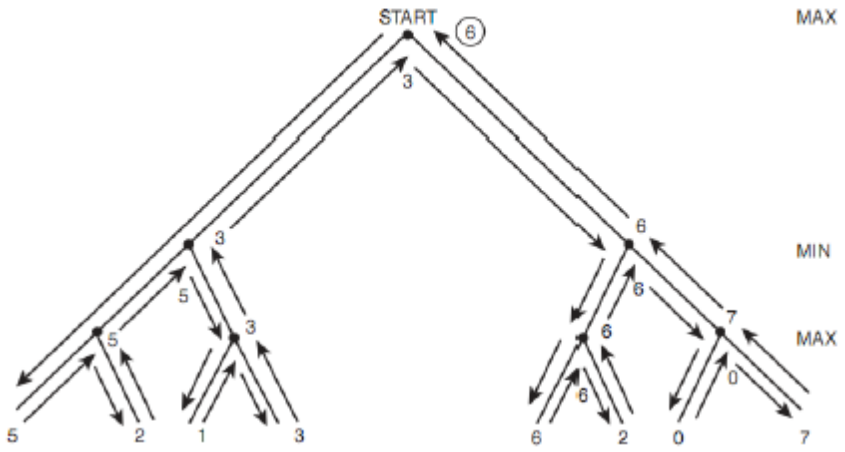
> "Untuk setiap permainan zero-sum dengan dua pemain dan strategi yang terbatas, ada sebuah nilai V dan strategi tertentu untuk setiap pemain sedemikian sehingga dengan strategi dari pemain-2, pemain-1 dapat memperoleh nilai terbaik V dan sebaliknya dengan strategi dari pemain-1, pemain-2 dapat memperoleh nilai terbaik –V." [5]

Artinya, pemain-1 harus memperoleh nilai V untuk memenangkan permainan dan pemain-2 harus memperoleh nilai –V untuk memenangkan permainan. Nilai 0 menunjukkan bahwa permainan seri, dan nilai-nilai lain di antara –V dan V menunjukkan pemain yang lebih dominan atau menguasai permainan (peluang untuk menang lebih besar).

Pada metode pencarian sebelumnya pada *blind search* tidak terdapat nilai di setiap simpul pohon pencarian, komputer akan memeriksa setiap simpul yang ada, sedangkan dalam metode pencarian Heuristik, pohon pencarian telah memiliki nilai disetiap simpulnya, akan tetapi fungsi Heuristik saja tidak cukup untuk memenangkan permainan, oleh karena itu diperlukan sebuah metode tambahan agar computer lebih cerdas menentukan keputusan.

Algoritma Minimax merupakan algoritma dasar pencarian DFS untuk melakukan traversal dalam pohon. DFS akan mengekspansi simpul paling dalam terlebih dahulu. Setelah simpul akar dibangkitkan, algoritma ini akan membangkitkan simpul pada tingkat kedua, yang akan dilanjutkan pada tingkat ketiga, dst. Dalam melakukan traversal, misalkan dimulai dari suatu simpul i, maka simpul selanjutnya yang akan dikunjungi adalah simpul tetangga j, yang bertetangga dengan simpul k, selanjutnya pencarian dimulai lagi secara rekursif dari simpul j. Ketika telah mencapai simpul m, di mana semua simpul yang bertetangga dengannya telah dikunjungi, pencarian akan dirunutbalik ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul j yang belum dikunjungi. Selanjutnya pencarian dimulai kembali dari j. Ketika tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi maka pencarian selesai. Untuk proses dan cara kerja algoritma yang

lebih jelasnya lagi, dapat dilihat pada Gambar 2.11 yang merepresentasikan cara kerja algoritma Minimax.



Gambar 2.10 Ilustrasi Cara kerja Algoritma Minimax

Dari Gambar 2.10, proses pencarian dimulai dari jalur paling kiri terlebih dahulu, sehingga DFS akan menelusuri simpul paling kiri bawah yaitu 5. Nilai 5 disimpan sebagai nilai maksimum sementara karena berada di tingkat max, kemudian DFS melakukan backtrack dan menelusuri simpul yang bertetangga dengan simpul 5 yaitu simpul 2. Karena nilai 5 lebih besar dari nilai 2, maka nilai 2 tidak disimpan. Lalu DFS akan melakukan backtrack ke tingkat min sehingga nilai 5 yang diperoleh akan disimpan sebagai nilai minimum sementara. Untuk simpul 1 dan 3, nilai 3 yang akan disimpan karena merupakan nilai maksimum di tingkat max. Saat mencapai tingkat min, sudah ada nilai minimum sementara yaitu 5, namun karena nilai 3 lebih kecil daripada nilai 5, maka nilai 5 akan digantikan dengan nilai 3. Nilai 3 akan disimpan sebagai nilai maksimum sementara di tingkat paling atas karena merupakan tingkat max. Lalu penelusuran jalur kanan akan dilakukan dengan cara yang sama seperti penelusuran jalur kiri sehingga diperoleh nilai 6. Karena nilai maksimum sementara pada tingkat paling atas adalah nilai 3, maka nilai 3 akan digantikan dengan nilai 6 karena nilai 6 lebih besar daripada nilai 3. Dengan demikian, jalur yang akan dipilih menggunakan

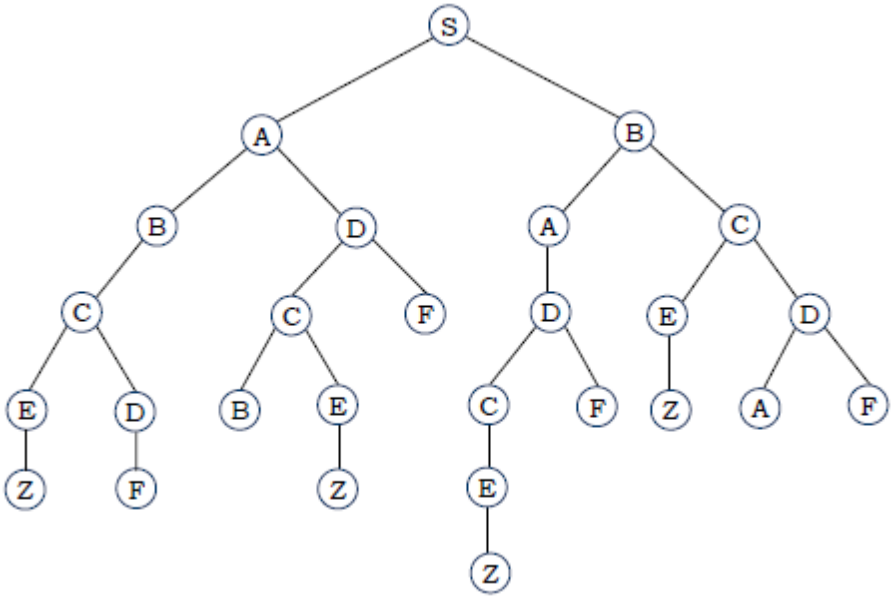
algoritma Minimax adalah jalur sebelah kanan karena untuk kondisi terburuknya, penulis akan mendapatkan nilai 6 sedangkan jika penulis memilih jalur kiri, penulis hanya akan mendapatkan nilai 3.

Algoritma minimax mencari solusi terbaik dengan "melihat ke depan" hingga ke akhir permainan dan memutuskan atau memilih langkah yang harus diambil saat itu untuk mencapai solusi tersebut. Nilai terbaik bisa bisa terdapat pada posisi Max ataupun pada posisi Min, dalam artian bahwa mengikuti posisi Min bukan berarti akan terjebak dalam kondisi minimum, melainkan penandaan Max dan Min bertujuan untuk member identitas langkah pemain. Misal ketika lawan yaitu human mengikuti dengan cara mencari nilai min, maka bisa dipastikan adalah bahwa human adalah pintar, karena langkah Max setelah Min atau Min setelah Max telah diramalkan oleh computer dengan mencari node simpul nilai heuristic terbesar. Minimax pada permain congklak digunakan untuk menemukan nilai akurat sebuah posisi papan dari permainan congklak. Dengan mengasumsikan bahwa kedua pemain selalu mengambil langkah terbaik pada setiap posisi.

2.5.1.2.1 Pohon pelacakan(*Search Tree*)

Struktur pohon digunakan untuk menggambarkan keadaan secara hirarkis, dari suatu proses pelacakan node secara berulan. Node yang terletak pada level-0 disebut juga akar, yang menunjukan keadaan awal biasanya merupakan topic atau objek yang terletak pada level-0. Node akar memiliki beberapa percabangan yang terdiri dari node successor yang disebut juga anak yang merupakan perantara. Node yang tidak memiliki anak disebut node daun yang merupakan akhir dari suatu pencarian dapat berupa sebuah tujuan yang di capai (goal), atau juga

jalanbuntu (*dead end*). Untuk lebih jelasnya digambarkan dibawah ini :



Gambar 2.11 contoh struktur pohon (*tree*)

Pada gambar di atas simpul S merupakan simpul awal dimulainya penelusuran dan simpul Z merupakan simpul yang akan menjadi tujuan.

Pohon pencarian dalam algoritma Minimax bertugas untuk melakukan pncarian seluruh kemungkinan langkah permainan (panjang cabang pohon ditenntukan oleh level permainan). Dalam langkah congklak, setiap simpul pohon pencarian dapat berupa posisi papan congklak yang mungkin untuk komputer.

2.5.1.2.2 Fungsi Evaluasi Heuristik

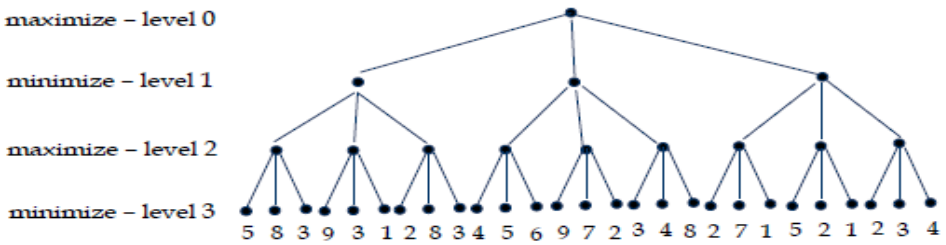
Heuristik adalah criteria, metode, atau prinsip-prinsip untuk menentukan pilihan dai sejumlah altenatif mencapai sasaran dengan efektif. Heuristik dipergunakan untuk mempersempit ruang pelacakan. Algoritma Minimax mampu menganalisa segala kemungkinan posisi permainan untuk menghasilkan keputusan terbaik. Algoritma Minimax bekerja secara rekursif dengan mencari langkah yang akan di hitung dengan algoritma yang sama dan seterusnya. Dalam penentuan keputusan langkah minimum dan maksimum dibutuhkan suatu nilai

yang merepresentasikan kerugian dan keuntungan yang akan diperoleh jika langkah tersebut dipilih.

Algoritma Minimax menggabungkan fungsi evaluasi heuristik dengan pengetahuan yang sedemikian higga dimiliki oleh komputer. pengetahuan yang dimaksud adalah strategi memenangkan permainan. Strategi ini akan diubah ke dalam bentuk heuristik. Strategi ini akan dipengaruhi oleh 2 faktor dalam permainan Congklak yaitu kondisi tembak dan jumlah biji yang ada di dalam mangkuk pemain dan lawan.

2.5.1.3 Alpha Beta Pruning

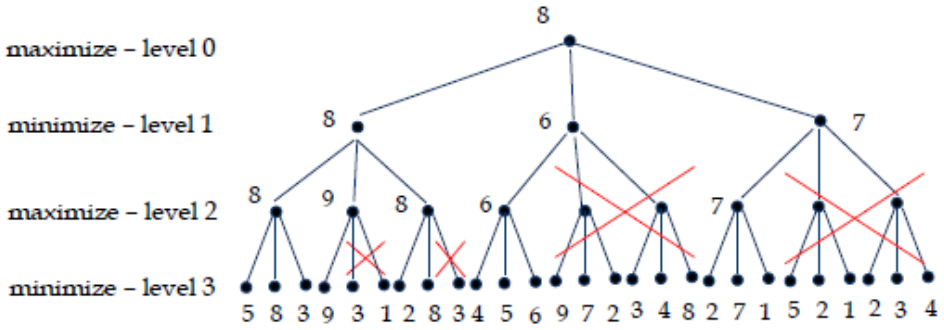
Untuk membantu memangkuskan algoritma minimax, dapat digunakan beberapa algoritma seperti *alpha-beta pruning*. *Alpha beta pruning* merupakan algoritma yang mengurangi jumlah simpul yang harus diproses pada algoritma minimax. Algoritma ini "memotong" simpul yang sedang diproses jika ditemukan bahwa *Alpha-Beta Pruning* akan memotong simpul yang tidak perlu diproses yang artinya simpul tersebut belum tentu akan lebih baik dari pada simpul sebelumnya.



Gambar 2.12 *Tree* kondisi awal pada alpha beta pruning algoritma

Node pertama pada leaf dengan nilai 5, naik ke parent level 2 masukkan nilai 5, hampiri 8, karena 8 > 5 maka ganti parent dengan 8 hampiri 3. Setelah ketiga leaf pertama terhampiri, naik lebih tinggi lagi ke level 1 masukan nilai 8. Hampiri node keempat dengan pada leaf dengan nilai 9, naik ke parent level 2 masukan nilai 9. Jika kita menghampiri leaf berikutnya, kita mencari nilai yang

lebih tinggi dari 9, sementara pada level 1 kita mencari yang lebih kecil dari 8, maka leaf 3 dan 1 kita potong. Dengan cara seperti itu maka akan didapatkan hasil akhir dari tree, yang dapat dilihat dari gambar dibawah ini :



Gambar 2.13 *Tree* untuk hasil metode Alpha betta pruning

Kompleksitas pencarian dari algoritma alpha-beta ini adalah O(bd/2) dibandingkan dengan kompleksitas algoritma minimax biasa yaitu O(bd). Jadi, jika pada algoritma minimax diproses 400 simpul, maka dengan adanya algoritma alpha-beta, simpul yang diproses hanya 20 buah saja.

Manfaat dari Alpha-Beta pruning terletak pada kenyataan bahwa cabang-cabang pohon pencarian dapat dihilangkan. Dengan cara ini, waktu pencarian dapat dibatasi pada 'lebih menjanjikan' *subtree*, dan pencarian yang lebih dalam dapat dilakukan dalam waktu yang sama. Seperti pendahulunya, dimiliki *branch* dan *bound* dari *class* algoritma. Optimasi mengurangi kedalaman efektif untuk sedikit lebih dari setengah dari minimax, jika node dievaluasi dalam urutan yang mendekati optimal (pilihan terbaik untuk sisi bergerak adalah pertama di setiap node).

Biasanya selama alpha-beta, subpohon untuk sementara didominasi oleh baik keunggulan pemain pertama (ketika banyak bergerak pemain pertama yang baik, dan pada setiap kedalaman pencarian langkah pertama diperiksa oleh pemain pertama memadai, tapi semua tanggapan pemain kedua yang diperlukan untuk

mencoba untuk menemukan penyangkalan), atau sebaliknya. Keuntungan ini dapat beralih sisi berkali-kali selama pencarian jika langkah memesan tidak benar, setiap kali menyebabkan inefisiensi. Sebagai jumlah posisi dicari menurun secara eksponensial setiap langkah lebih dekat posisi saat ini, ada baiknya menghabiskan banyak upaya pada menyortir bergerak lebih awal. Sebuah semacam ditingkatkan pada setiap kedalaman eksponensial akan mengurangi jumlah posisi dicari, namun menyortir semua posisi pada kedalaman dekat akar relatif murah karena ada begitu sedikit dari mereka. Dalam prakteknya, langkah memesan sering ditentukan oleh hasil sebelumnya, pencarian yang lebih kecil, seperti melalui iteratif pendalaman.

Variabel alpha (α) digunakan sebagai batas bawah *node* max, sedangkan variabel beta (β) digunakan sebagai batas atas *node* min. Pada *node* min, evaluasi akan dihentikan apabila telah didapat *node* anak yang memiliki nilai lebih kecil dibanding dengan nilai batas bawah (α), sebaliknya pada *node* max evaluasi akan dihentikan apabila telah didapat *node* anak yang memiliki nilai lebih besar dibanding dengan nilai batas atas (β). Pada *root* pohon pencarian, nilai α ditetapkan sama dengan -∞ sedangkan nilai β diset sama dengan +∞. *Node* yang melakukan maksimasi akan memperbaiki nilai α dari nilai anak-anaknya, sedangkan *node* yang melakukan minimasi akan memperbaiki nilai β dari nilai anak-anaknya. Jika $\alpha > \beta$, maka evaluasi dihentikan [12]. Selain itu, algoritma ini dapat dimodifikasi sepele untuk mengembalikan seluruh modal variasi selain skor. Beberapa algoritma yang lebih agresif seperti MTD (f) tidak mudah mengizinkan seperti modifikasi.

2.6 Flowchart

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam

pengoperasian. Ada beberapa petunjuk yang harus diperhatikan bagi seseorang analis atau programmer dalam membuat *flowchart* yaitu :

1. Flowchart digambarkan dari halaman atas ke bawah dan dari kiri ke kanan.
2. Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini harus dapat dimengerti oleh pembacanya.
3. Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas.
4. Setiap langkah dari aktivitas harus diuraikan dengan menggunakan deskripsi kata kerja, misalkan "Menghitung Pajak Penjualan".
5. Setiap langkah dari aktivitas harus berada pada urutan yang benar.
6. Lingkup dan *range* dari aktifitas yang sedang digambarkan harus ditelusuri dengan hati-hati. Percabangan-percabangan yang memotong aktivitas yang sedang digambarkan tidak perlu digambarkan pada flowchart yang sama. Simbol konektor harus digunakan dan percabangannya diletakan pada halaman yang terpisah atau hilangkan seluruhnya bila percabangannya tidak berkaitan dengan sistem.
7. Gunakan simbol-simbol *flowchart* yang standar.

Flowchart terbagi dalam lima jenis yaitu :

1. *Flowchart* Sistem (*System Flowchart*)
2. *Flowchart* Paperwork / Flowchart Dokumen (*Document Flowchart*)
3. *Flowchart* Skematik (*Schematic Flowchart*)
4. *Flowchart* Program (*Program Flowchart*)
5. *Flowchart* Proses (*Process Flowchart*)

Simbol-simbol flowchart yang biasanya dipakai adalah simbol-simbol flowchart standar yang dikeluarkan oleh ANSI dan ISO.

Simbol-simbol ini dapat dilihat pada Gambar simbol standar *flowchart***.**