BAB 2
LANDASAN TEORI

2.1 Permainan

Pada bagian ini akan dijelaskan mengenai definisi permainan dan jenis-jenis permaianan.

2.1.1 Definisi Permainan

Permainan (*Game*) terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk mengoptimalkan kemenangan sendiri atau meminimalkan kemenangan lawan. Peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai aturan dalam bermain, sejumlah kemenangan atau kekalahan dalam berbagai situasi [7]. Permainan umumnya menyediakan sistem penghargaan, misalnya nilai (*score*) yang dihitung berdasarkan tingkat keberhasilan yang dicapai dalam menyelesaikan tugas-tugas yang ada di dalam permainan tersebut. Sistem elektronik yang digunakan untuk menjalankan permainan disebut *platform,* contohnya adalah komputer dan konsol permainan (g*ame console).*

Permainan komputer adalah permainan yang dimainkan pada komputer dan bukan pada konsol permainan. Permainan komputer telah berevolusi dari sistem grafis sederhan menjadi sangat kompleks dan mutakhir. Sedangkan permainan konsol adalah sebuah bentuk media interaktif yang berfungsi sebagai hiburan. Permainan ini pada umumnya berisi gambar-gambar dan suara-suara yang dihasilkan oleh sebuah alat permainan konsol, dan ditampilkan pada sebuah televisi atau peralatan audio visual yang sejenis. Permainan ini sendiri dapat dikendalikan dengan menggunakan alat kendali yang dipegang dan terhubung pada konsol yang

dinamakan pengendali (*controller).* Pengendali ini pada umumnya terdiri dari sejumlah tombol-tombol dan kendali arah yang masing-masing bagian memiliki fungsi tersendiri. Fungsi dari alat pengendali tersebut untuk mengendalikan maupun berinteraksi dengan gambar yang ada di layar.

2.1.2 Jenis Permainan

Permainan dari tahun pertama dibuat sampai sekarang semakin banyak jenis dan teknologi yang digunakan semakin canggih. Dari semua permainan yang telah dibua tdi dunia, permainan tersebut dikelompokkan menjadi beberapa jenis. Beberapa contoh jenis-jenis permainan yang sering dimainkan dan dikelompokkan sebagai berikut:

1) *Action Game* (Permainan Aksi)

   Jenis permainan ini memiliki fitur utama berupa banyaknya aksi di mana pemain harus memiliki keterampilan dengan reaksi yang cepat untuk menghindari musuh atau menghindari rintangan. Pengembang permainan jenis ini perlu memastikan permainan yang dibuat dimaksimalkan sehingga pemain memiliki pengalaman dalam bermain yang baik, dan tidak terganggu oleh waktu proses yang lama. Contohnya adalah Metal Gear Solid

2) *Adventure Game* (Permainan Bertualang)

   Jenis permainan ini pada umumnya membuat pemain harus berjalan mengelilingi suatu tempat dengan kondisi tertentu, seperti sebuah istana atau gua yang berkelok dan planet yang jauh. Pemain melakukan navigasi terhadap suatu tempat untuk mencari pesan-pesan rahasia dan memperoleh objek yang memiliki kemampuan yang bervariasi. Untuk membuat permainan ini membutuhkan perencanaan yang akurat sehingga memiliki alur cerita yang menarik bagi pemain. Contohnya adalah Tomb Raider.

3) *Fighting Game* (Permainan Bertarung)

   Jenis permainan ini pada umumnya pemain bertarung dengan lawan tandingnya dengan cara memukul, menendang, atau menggunakan jurus sampai lawan tandingnya kalah. Contohnya adalah Tekken, Mortal Kombat, dan lain-lain.

4) *Sport Game* (Permainan Olahraga)

   Permainan jenis ini berupa kompetisi antara dua pemain atau lebih, di mana pemain dapat berupa individual atau kelompok. Contoh permainan jenis ini adalah sepakbola, bola basket, tenis, bilyard, dan lain-lain. Permainan ini tergantung seberapa lama waktu yang dibutuhkan dalam satu pertandingan. Contohnya adalah Pro Evolution Soccer.

5) *Role Playing Game* (Permainan Beralur)

   Permainan ini seringkali berupa multi-player (Lebih dari satu pemain) di mana setiap pemain memiliki karakter dengan kemampuan, kekuatan, dan kelemahan yang berbeda. Pemain saling berkompetisi, berinteraksi, dan bertempur satu sama lain. Tampilan grafik yang berbeda untuk setiap karakter pemain dan memiliki storyline (alur cerita) yang mendebarkan sehingga memberikan pengalaman yang menarik dan berbeda di dalam bermain. Contohnya adalah Final Fantasy.

6) *Strategy Game* (Permainan Strategi)

   Jenis permainan ini pada umumnya pemain menggunakan taktik dan strategi untuk melawan dan menghancurkan musuhnya dengan membangun tentara dan alat perang menggunakan sumber daya yang ada secara bijaksana dan teliti. Contohnya adalah Command & Conquer.

7) *Platform Game*

   Permainan ini mengharuskan pemain mengarahkan suatu obyek dengan melalui berbagai tahap atau tingkatan area untuk menyerang musuh dan menghindar terhadap serangan. Jenis

permainan ini hampir serupa dengan permainan aksi, tetapi aksi yang dilakukan tidak secepat permainan aksi. Contohnya adalah Sonic The Hedgehog & Mario Bros.

8) *Puzzle Game* (Permainan Teka-teki)

Permainan ini pada umumnya membuat pemain menggunakan kemampuan berpikirnya sebagai pengganti keterampilan reaksi yang cepat karena terdapat rahasia yang perlu dipecahkan. Permainan ini lebih bersifat statis dibanding permainan aksi. Pembuatan permainan jenis ini seringkali ditunjang dengan algoritma kecerdasan buatan. Contohnya adalah 7 Wonders of The Ancient World, Zuma, dan lain-lain.

9) *Simulation Game* (Permainan Simulasi)

Jenis Permainan ini memberikan pengalaman atau interaksi secara mirip dengan kendaraan aslinya, meskipun terkadang kendaraan tersebut masih berupa contoh atau bahkan fiktif, tapi ada penekanan khusus pada detail dan pengalaman nyata menggunakan kendaraan tersebut. Contohnya adalah Gran Turismo, Sim City dan lain-lain.

10) *Board Game* (Permainan Papan)

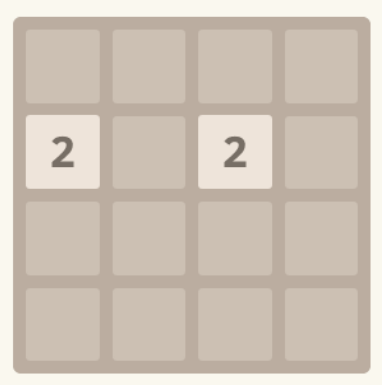
Jenis permainan ini membutuhkan papan permainan untuk memainkannya. Permainan ini biasanya melibatkan potongan yang bisa dipindahkan atau ditempatkan pada permukaan atau papan permainan dengan aturan yang sudah dibuat. Permainan ini didasarkan pada strategi murni, kesempatan (misalnya menggunakan dadu atau giliran), atau campuran dari keduanya, dan biasanya memiliki tujuan untuk mencapai sesuatu. Permainan papan berasal dari representasi pertempuran antara dua tentara dan yang pada zaman sekarang masih berdasarkan mengalahkan pemain lawan dalam hal menyerang, posisi, atau mendaptkan poin.

2.2 Permainan 2048

Pada bagian ini akan menjelaskan sejarah permainan 2048 dan aturan permainan 2048.

2.2.1 Sejarah Permainan 2048

Permainan 2048 merupakan permainan untuk pemain tunggal yang diciptakan pada bulan Maret 2014 oleh seorang programmer asal Italia yang bernama Gabriele Cirulli. Ketika pertama kali dirilisnya permainan ini, permainan ini langsung melesat ke peringkat 1 dunia dalam hanya waktu seminggu dan mencapai lebih dari 4 juta download dan hal tersebut merupakan rekor yang cukup fantastis. Pada kenyataannya, permainan ini hanyalah sebuah proyek "iseng" yang dikerjakan oleh programmer Italia tersebut yang hanya ingin mencoba apakah ia mampu membuat sebuah permainan dari awal dan ia berhasil menyelesaikan pembuatannya hanya dalam waktu seminggu. Permainan ini juga bukanlah yang pertama kali diciptakan, sebelumnya juga pernah ada permainan berjudul Three! yang memiliki tampilan serta cara bermain yang mirip, dan juga permainan 1024 yang cara bermainnya sama persis, hanya saja dengan tujuan permainan yang sedikit berbeda [1].



Gambar 2.1 Permainan 2048

2.2.2 Aturan Permainan 2048

Untuk aturan cara bermain permainan 2048 adalah sebagai berikut [1]:

1) Permainan 2048 dimainkan di papan abu-abu 4×4.
2) Di atasnya terdapat beberapa kotak dengan warna berbeda yang bergeser ketika pemain menekan empat tombol arah di keyboard.
3) Kotaknya dapat bergerak sejauh mungkin sampai bertemu kotak lain atau pinggir papan.
4) Apabila dua kotak bernomor sama bergabung saat bergeser, keduanya akan bergabung dan menghasilkan kotak baru yang nomornya merupakan hasil penjumlahan nomor di dua kotak yang bergabung tadi.
5) Misal pada Gambar 2 kotak yang bernilai 2 bisa digabung saat pemain menggeserkan ke arah kiri atau kanan dan akan menghasilkan kotak yang bernilai 4 seperti pada Gambar 2.1.
6) Kotak yang baru ini tidak bisa bergabung dengan kotak lain dengan arah yang sama.
7) Di setiap pergeseran akan ada satu kotak baru bernomor 2 atau 4 yang muncul secara acak pada ruang kosong di papan.
8) Pemain baru menang ketika sebuah kotak bernomor 2048 muncul di papan.
9) Saat pemain tidak bisa bergerak lagi maka permainan berakhir.
10) Skor didapatkan dari hasil penggabungan kotak.

2.3 Kecerdasan Buatan

Kecerdasan buatan sering disebut juga dengan *Artificial Intelligence* (AI). Kecerdasan buatan adalah ilmu dan rekayasa, yang membuat mesin mempunyai kecerdasan tertentu khususnya program komputer yang cerdas.

Intelegensi merupakan bagian kemampuan komputasi untuk mencapai tujuan di dalam dunia. Ada beberapa macam jenis dan derajat intelegensi untuk manusia, binatang dan beberapa mesin. Kecerdasan buatan berhubungan dengan penggunaan komputer untuk melaksanakan tugas atau menyelesaikan masalah yang memerlukan kecerdasan atau kepintaran. Pada umumnya komputer memerlukan aplikasi atau perangkat lunak untuk melaksanakan tugas atau menyelesaikan masalah, maka aplikasi atau perangkat lunak yang berdasarkan teknologi Kecerdasan buatan di perlukan.

Agar komputer bisa bertindak seperti apa yang dapat dilakukan manusia, maka komputer juga harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Untuk itu kecerdasan buatan akan mencoba untuk memberikan beberapa metode untuk membekali komputer agar bisa menjadi mesin yang pintar. Dari beberapa perspektif, kecerdasan buatan dapat dipandang sebagai berikut [8]:

1) Perspektif kecerdasan, AI adalah bagaimana membuat mesin yang cerdas dan dapat melakukan hal-hal yang sebelumnya hanya dapat dilakukan manusia.
2) Perspektif bisnis, AI adalah sekelompok alat bantu (tools) yang berdayaguna dan metodologi yang menggunakan alat-alat bantu tersebut untuk menyelesaikan masalah-masalah bisnis.
3) Perspektif pemrograman, AI meliputi studi tentang pemrograman simbolik, pemecahan masalah, dan proses pencarian (search).
4) Perspektif penelitian, riset tentang AI dimulai pada awal tahun 1960an, percobaan pertama adalah membuat program permainan catur, membuktikan teori, dan general problem solving.

2.4 Algoritma *Minimax*

Algoritma *minimax* (juga sering disebut Minmax) adalah sebuah algoritma yang mendasari pola pikir langkah penyelesaian masalah dalam beberapa jenis permainan komputer, seperti tic-tac-toe, othello, checkers,

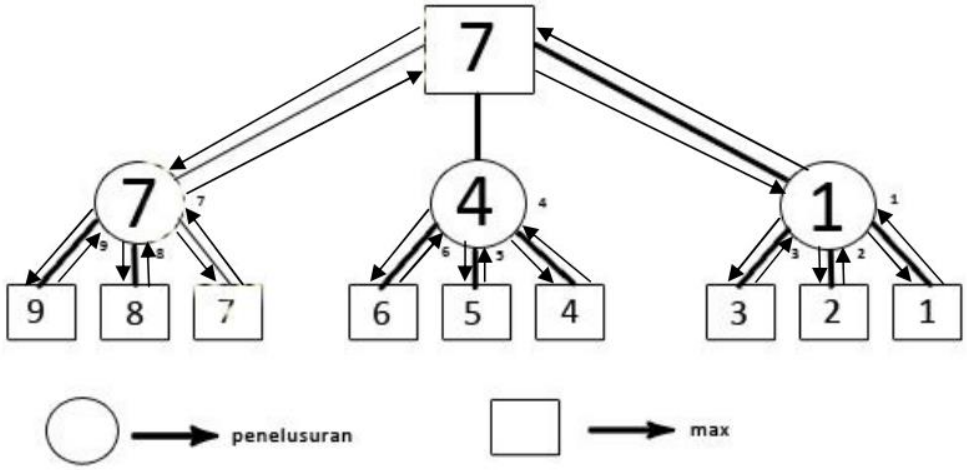
catur. Pada dasarnya, algoritma *minimax* sangat andal untuk menyelesaikan segala masalah dalam pencarian langkah untuk permainan komputer dengan jumlah kemungkinan penyelesaian yang kecil, seperti pada permainan tic-tac-toe. Secara teori, algoritma *minimax* didefinisikan sebagai berikut: "Untuk setiap permainan satu-lawan satu, ada sebuah nilai yang bernilai V dan strategi yang dipilih oleh tiap 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, sehingga:

1) Jika diberikan strategi dari pemain ke-2, maka langkah penyelesaian terbaik dari pemain pertama adalah V.
2) Jika diberikan strategi dari pemain pertama, maka langkah penyelesaian terbaik dari pemain kedua adalah –V".

Singkatnya, pemain pertama memberikan langkah penyelesaian yang bernilai V terhadap permainan pemain kedua, dan sebaliknya, pemain kedua memberikan langkah peyelesaian bernilai –V. Pemikiran inilah yang mendasari asal usul penamaan algoritma *Minimax*, dimana pemain yang satu berjuang untuk mendapat nilai maksimum, sedangkan lawannya berjuang untuk mendapat nilai minimum [9].

Cara kerja algoritma *Minimax*:

1) Misalkan ada 2 pemain yang terlibat, kita namakan MAX dan MIN
2) Lakukan pencarian atau penulusuran dari simpul yang paling dalam
3) setelah simpul yang paling dalam ditelusuri akan dilakukan pencarian tingkat kedua, ketiga dan seterusnya .
4) Pencarian dilakukan secara rekursif
5) lakukan pembandingan nilai MAX dan MIN
6) Nilai MIN dan MAX yang telah dibandingkan akan disimpan pada temp sementara
7) Tentukan Nilai MAX untuk menentukan nilai Root.



Gambar 2.2 Pohon Pencarian Minimax

Algoritma *minimax* merupakan salah satu algoritma yang digunakan pada permainan papan (*board game)* yang pada umumnya dimainkan oleh 2 orang pemain atau lebih dan berbasis zero-sum yang artinya keuntungan untuk pemain pertama berarti kerugian untuk pemain kedua. Algoritma *minimax* akan melakukan pengecekan terhadap seluruh kemungkinan yang ada sampai permainan selesai. Keuntungan yang didapat dengan menggunakan algoritma *minimax* yaitu algoritma *minimax* mampu menganalisis segala kemungkinan posisi permainan untuk menghasilkan keputusan yang terbaik karena algoritma *Minimax* ini bekerja secara rekursif dengan mencari langkah yang akan membuat lawan mengalami kerugian minimum. Semua strategi lawan akan dihitung dengan algoritma yang sama dan seterusnya. Ini berarti, pada langkah pertama komputer akan menganalisis seluruh kondisi permainan, dan untuk setiap langkahnya, komputer akan memilih langkah yang paling membuat lawan mendapatkan keuntungan minimum, dan yang paling membuat komputer itu sendiri mendapat keuntungan maksimum. Dalam penentuan keputusan tersebut dibutuhkan suatu nilai yang merepresentasikan kerugian atau keuntungan

yang akan diperoleh jika langkah tersebut dipilih. Pada Gambar 2.3 adalah *pseudocode* dari algoritma *minimax.*

```
function minimax(node, depth, turn)
       if depth = 0 or node is a terminal node
           return the heuristic value of node
       if turn == player1
           bestValue := -∞
           foreach child of node
             val := minimax(child, depth - 1, player2)
             bestValue := max(bestValue, val);
           return bestValue
       else
           bestValue := +∞
           for each child of node
             val := minimax(child, depth - 1, player1)
             bestValue := min(bestValue, val);
           return bestValue
```

Gambar 2.3 *Pseudocode* Algoritma *Minimax*

Contoh implementasi algoritma *minimax*: misalkan ada sebuah permainan sederhana yang dengan hanya ada satu langkah, untuk tiap pemain dengan kemungkinan situasi seperti pada Tabel 2.1.

Tabel 2.1 Contoh Nilai Akhir Pergerakan MAX dan MIN

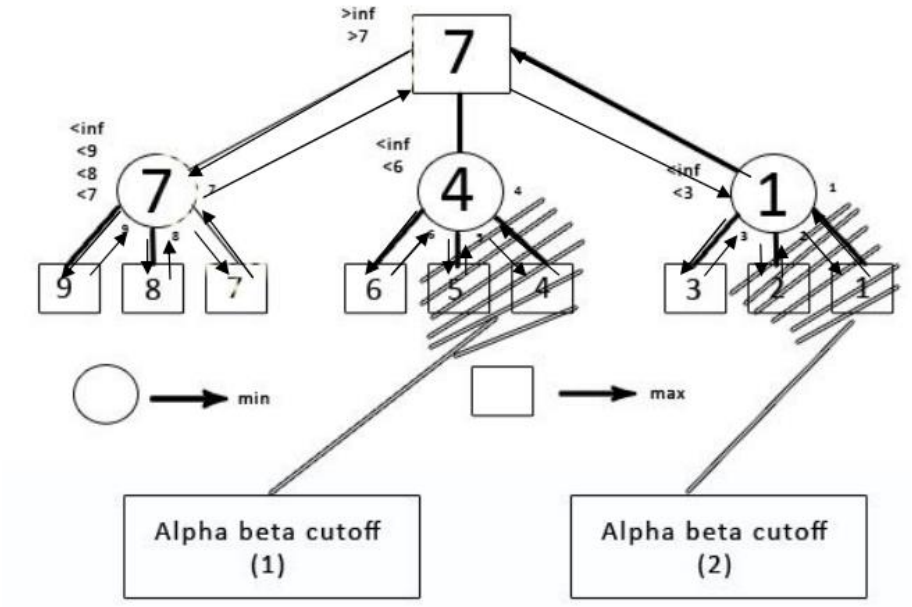
MAX	MIN	Nilai Evaluasi
A	C	12
A	D	-2
B	C	5
B	D	6

Fungsi evaluasi berlaku untuk posisi akhir papan, yang berupa kombinasi langkah dari MIN dan MAX. MAX berasumsi jika MIN akan bermain dengan baik. Maka dari itu, MAX tahu jika dia melakukan pergerakan A, maka MIN akan membalasnya dengan melakukan pergerakan

D, yang mengakibatkan nilai evaluasi -2 (kemenangan bagi MIN). Bagaimanapun juga, jika MAX melakukan gerakan B, dia pasti akan menang karena pergerakan MIN yang terbaik hanya akan menghasilkannilai evaluasi terbaik sebesar 5 saja. Jadi, dengan menggunakan algoritma minimax, MAX akan selalu memilih untuk melakukan langkah B, walaupun dia sebenarnya akan mendapatkan kemenangan yang lebih baik jika melakukan A dan MIN melakukan kesalahan dengan memilih langkah C.

2.5 Algoritma *Alphabeta Prunning*

Dalam algoritma *minimax*, pencarian dilakukan pada seluruh bagian pohon, sementara sebagian pohon tidak seharusnya diperiksa. *Alpha-beta pruning* merupakan modifikasi dari algoritma *minimax*, yang akan mengurangi jumlah node yang dievaluasi oleh pohon pencarian. Pencarian untuk node berikutnya akan dipikirkan terlebih dahulu. Algoritma ini akan berhenti mengevaluasi langkah ketika terdapat paling tidak satu kemungkinan yang ditemukan dan membuktikan bahwa langkah tersebut lebih buruk jika dibandingkan dengan langkah yang diperiksa sebelumnya. Sehingga, langkah berikutnya tidak perlu dievaluasi lebih jauh. Dengan algoritma ini hasil optimasi dari suatu algoritma tidak akan berubah [5]. Berikut merupakan pohon dengan algoritma *alpha-beta pruning.*



Gambar 2.4 Pohon Pencarian *AlphaBeta Prunning*

Diperlihatkan, pada pohon tersebut, bahwa terdapat pemotongan pencarian dengan menggunakan algoritma ini. Pada algoritma ini, terdapat dua nilai yang diatur, yaitu alpha dan beta, yang merepresentasikan nilai minum dari *max* yang diyakini dan nilai maksimum dari *min* yang diyakini. Nilai awal alpha adalah tak hingga negatif dan nilai awal beta adalah tak hingga positif. Sebagai hasil dari proses rekursif, area pencarian akan semakin kecil. Ketika beta menjadi lebih kecil dari alpha, akan berarti posisi saat itu tidak dapat menjadi hasil terbaik permainan untuk kedua pemain dan pencarian tidak perlu dilakukan lebih jauh. Pada Gambar 2.5 dapat dilihat *pseudocode* dari algoritma *alphabeta*.

```
function alphabeta(node, depth, α, β, maximizingPlayer)
      if depth = 0 or node is a terminal node
          return the heuristic value of node
      if maximizingPlayer
          v := -∞
          for each child of node
              v := max(v, alphabeta(child, depth - 1, α, β,
FALSE))
              α := max(α, v)
              if β ≤ α
                  break (* β cut-off *)
```

```
            return v
        else
            v := ∞
            for each child of node
                v := min(v, alphabeta(child, depth - 1, α, β,
TRUE))
                β := min(β, v)
                if β ≤ α
                    break (* α cut-off *)
            return v
```

Gambar 2.5 *Pseudocode* Algoritma *Alphabeta*

2.6 Algortima MTD(f)

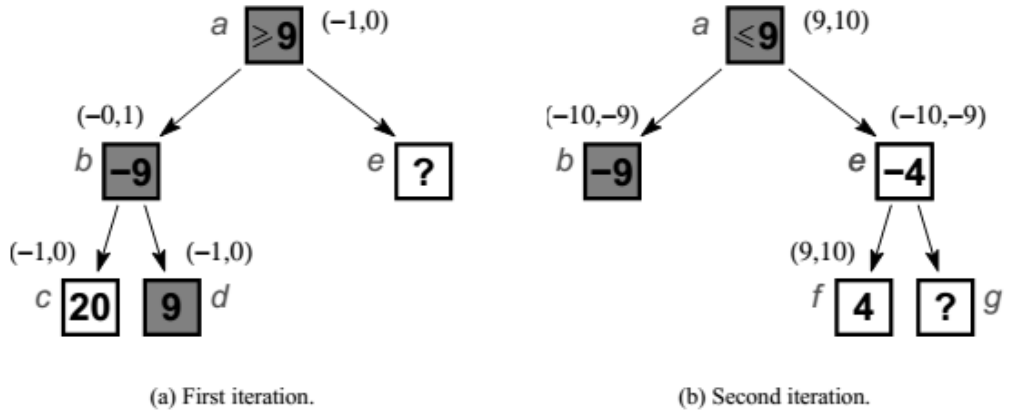
Algoritma MTD(f) adalah sebuah algoritma optimasi Minimax baru yang lebih sederhana dan lebih sangkil daripada beberapa pendahulunya [4]. Nama dari algoritma ini adalah kependekan dari MTD(n,f), yang disingkat dari *Memory-enhanced Test Driver with node n and value f*. MTD adalah nama dari sekumpulan driver program yang mencari pohon Minimax menggunakan pemanggilan *zero-window AlphaBeta*. Dalam beberapa percobaan permainan komputer seperti catur, othello, dan checkers, algoritma ini mempunyai performa rata-rata lebih baik daripada Negascout (variasi dari *AlphaBeta* yang diimplementasikan dalam hampir semua permainan catur, checkers, dan othello). Salah satu program catur terkuat, Cilkchess milik MIT yang menggunakan metode komputasi pararel, juga menggunakan MTD(f) sebagai algoritma pencariannya menggantikan Negascout yang digunakan oleh program catur pendahulunya, StarSocrates. Algoritma MTD(f) terdiri dari 10 baris kode saja, yaitu seperti pada Gambar 2.6.

```
function MTDF(root, f, d)
    g := f
    upperBound := +∞
    lowerBound := -∞
       while lowerBound < upperBound
          if g = lowerBound then
```

```
            β:= g+1
        else
            β := g
        g := AlphaBeta (root, β-1, β, d)
        if g < β then
            upperBound := g
        else
            lowerBound := g
    return g
```

Gambar 2.6 *Psuedo Code* Algoritma MTD(f)

Algoritma MTD(f) tersebut memanggil fungsi *Alpha Beta* berkali kali dengan metode *zero-window* pencarian *Alpha-Beta*, tidak seperti Negascout yang menggunakan pencarian *wide-window*. Pemanggilan *Alpha Beta* mengembalikan batas dari nilai evaluasi Minimax. Batas dari nilai itu kemudian disimpan dalam upperbound (batas atas) dan lowerbound (batas bawah), membentuk sebuah interval yang melingkupi nilai Minimax yang sebenarnya pada pencarian dengan kedalaman tertentu. Positif dan negatif tak hingga adalah kependekan dari nilai diluar interval pada daun pohon. Ketika batas atas dan batas bawahnya bernilai sama atau batas bawah telah melampaui nilai batas atas, maka nilai Minimax telah ditemukan.



Gambar 2.7 Pohon Pencarian MTD(F)

Efisiensi dari MTD(f) berasal dari pencarian *Alpha-Beta* dengan *zero-window*, dan menggunakan sebuah nilai batas yang baik (variabel *Beta*) untuk melakukan pencarian *zero-window* tersebut. Dalam versi yang sebelumnya, *Alpha-Beta* dipanggil dengan pencarian wide-window seperti ini *AlphaBeta*(root, -INFINITY, +INFINITY, depth), yang memastikan nilai kembaliannya berada dalam interval *Alpha* dan *Beta*. Sedangkan dalam algoritma MTD(f), digunakan pencarian *zero-window*, jadi untuk setiap pemanggilan *Alpha-Beta* akan mengembalikan batas bawah dan batas atas dari nilai Minimax berturut-turut. Pencarian dengan *zero-window* memberikan lebih banyak jalan pintas, tetapi lebih sedikit informasi, yaitu hanya batas dari nilai minimax saja.

2.7 *Object Oriented Programming* (OOP)

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunkan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam suatu entitas. Model berorientasi objek bermanfaat untuk memahami masalah, komunikasi dengan ahli aplikasi, pemodelan suatu organisasi, menyiapkan dokumentasi serta perancangan program dan basis data. Pertama-tama suatu model analisis dibuat untuk menggambarkan aspek dasar dari domain aplikasi, dimana model tersebut berisi objek yang terdapat dalam domain aplikasi termasuk deskripsi dari keterangan objek dan perilakunya. Secara spesifik, pengertian berorientai objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, polimorfisme dan pewarisan.

Karakteristik dari objek identitas berarti bahwa data diukur mempunyai nilai tertentu yang membedakan entitas dan disebut objek. Suatu paragraf dari dokumen, suatu windows dari workstation, dan raja putih dari buah catur adalah contoh dari objek. Objek dapat kongkrit, seperti halnya arsip dalam sistem, atau konseptual seperti kebijakan penjadualan dalam multiprocessing pada sistem operasi. Setiap objek mempunyai sifat yang melekat pada identitasnya. Dua objek dapat berbeda walaupun bila semua atributnya identik Klasifikasi berarti bahwa suatu kegiatan mengumpulkan data (atribut) dan perilaku (operasi) yang mempunyai struktur data sama ke dalam satu grup yang disebut kelas. Paragraf, window, buah catur adalah contoh dari kelas. Kelas merupakan abstraksi yang menjelaskan sifat penting pada suatu aplikasi dan mengabaikan yang lain. Setiap kelas menunjukan suatu kumpulan infinite yang mungkin dari objek. Suatu objek dapat dikatakan sebagai instans dari kelas, setiap instansi dari kelas mempunyai nilai individu untuk setiap nama atribut dan operasi, tetapi memiliki bersama atribut dan operasi dengan instansi lain dalam kelas.

Karakteristik Metodologi Berorientasi Objek Metodologi pengembangan sistem berorientasi objek mempunyai tiga karateristik utama yaitu [10]:

1) *Encapsulation*

   Encapsulation (pengkapsulan) merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas dalam bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain kecuali prosedur yang berada dalam objek itu sendiri.

2) *Inheritance*

   Inheritance (pewarisan) adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi atribut dan metoda dari

induknya langsung. Atribut dan metoda dari objek induk diturunkan kepada anak objek, demikian seterusnya. Pendefinisian objek dipergunakan untuk membangun suatu hirarki dari objek turunannya, sehingga tidak perlu membuat atribut dan metoda lagi pada anaknya, karena telah mewarisi sifat induknya.

3) *Polymorphism*

Polymorphism (polimorfisme) yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda. Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.

2.8 *Unified Modeling Language* (UML)

UML adalah sebuah bahasa yang telah menjadi standar dalam industri untuk menvisualisasi, menspesifikasi, merancang dan mendokumentasi sistem piranti lunak. *Unified Modelling Language* (UML) menawarkan sebuah standar untuk merancang model sebuah sistem. Tujuan UML adalah :

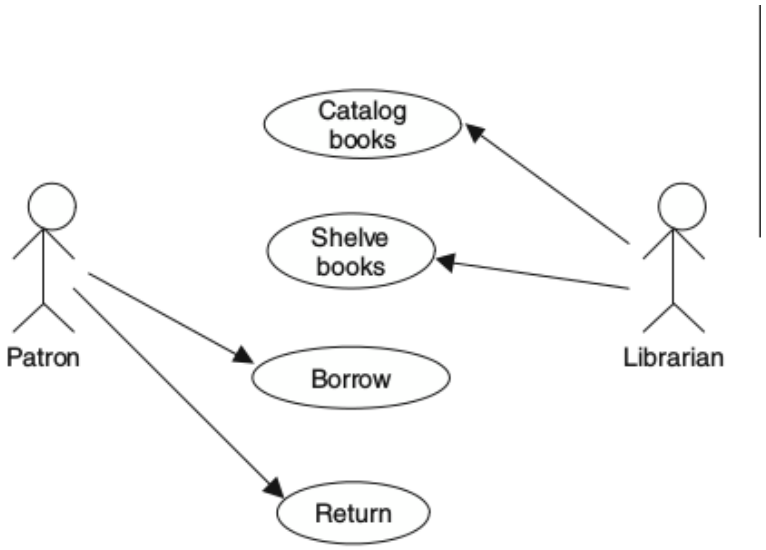
1) Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2) Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3) Menyatukan praktik-praktik terbaik yang terdapat dalam pemodelan. UML menyediakan beberapa notasi dan artifact standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. Artifact didalam UML didefinisikan sebagai informasi dalam bentuk

yang digunakan atau dihasilkan dalam proses pengembangan perangkat. Contohnya adalah source code yang dihasilkan oleh proses pemrograman.

Diagram-diagram pada UML akan dijelaskan sebagai berikut [11]:

1) *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". *Use Case* dilakukan oleh satu aktor yang digambarkan dengan simbol orang yang dihubungkan dengan garis yang menunjukan hubungan komunikasi. Setiap *Use Case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksi dengan Aktor. Nama *Use Case* boleh terdiri dari beberapa kata dan tidak boleh ada *Use Case* yang memiliki nama yang sama.

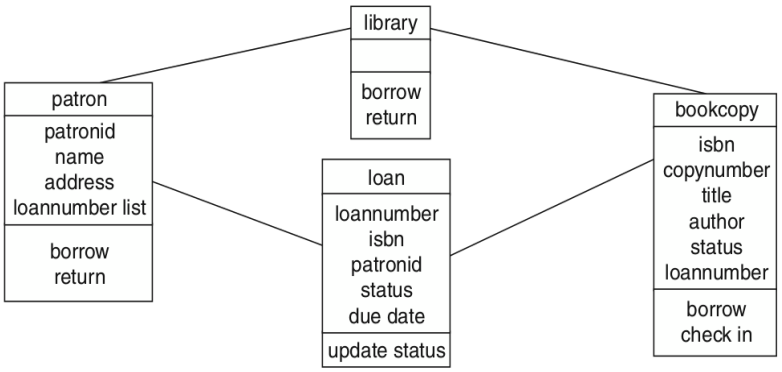


Gambar 2.8 Contoh *Usecase Diagram*

2) *Class Diagram*

Class Diagram menggambarkan tipe-tipe objek dalam sistem dan berbagai jenis hubungan atau relasi statis yang ada diantara mereka. Diagram ini memberikan gambaran umum dari sistem. Seperti tipe-tipe dari objek dengan menunjukan kelasnya dan hubungan diantara mereka, serat penjelasan detail tiap-tiap kelas ke dalam model suatu

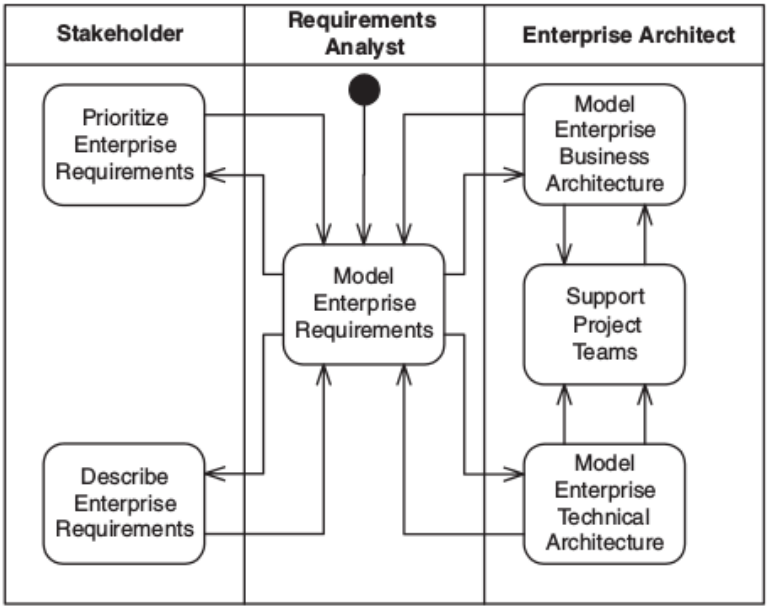
sistem. *Class Diagram* bersifat statik (tidak berubah) yang akan menunjukan apa itu interaksi tapi tidak menjelaskan apa yang terjadi ketika mereka melakukan interaksi.



Gambar 2.9 Contoh *Class Diagram*

3) *Activity Diagram*

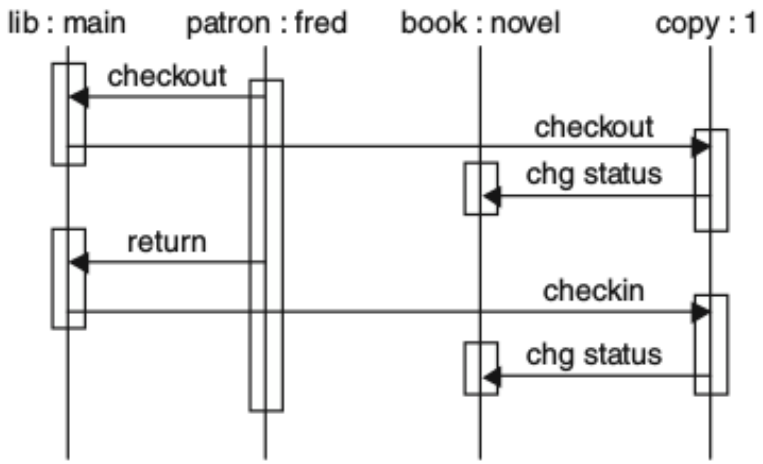
Activity Diagram memodelkan alur kerja. Sebuah proses bisnis dan urutan aktifitas dalam suatu proses. Diagram ini sangat mirip dengan sebuah *flowchart* karena dapat memodelkan sebuah alur kerja dari aktifitas keaktifitas lainnya atau dari suatu aktifitas kedalam keadaan sesaat.



Gambar 2.10 Contoh *Activity Diagram*

4) *Sequence Diagram*

Sequence Diagram merupakan gambaran interaksi antar objek di dalam dan di sekitar sistem berupa pesan yang digambarkan terhadap waktu. Diagram ini secara khusus berasosiasi dengan u*se case*. *Sequence Diagram* juga digunakan untuk menggambarkan skenario atau rangkain langkah-langkah apa yang seharusnya terjadi sebagai respons dari sebuah event untuk menghasilkan sesuatu didalam *use case* sebagai *output*. Untuk pesan digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, pesan dipetakan menjadi operasi/metode dari class. *Activation bar* menunjukan lamanya eksekusi sebuah proses dan biasanya diawali dengan sebuah pesan.



Gambar 2.11 Contoh *Sequence Diagram*

2.9 C#

C# (dibaca: C sharp) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain) dengan beberapa penyederhanaan. Menurut standar ECMA-334 C# Language Specification, nama C# terdiri atas sebuah huruf Latin C (U+0043) yang diikuti oleh tanda pagar yang menandakan angka # (U+0023). Tanda pagar # yang digunakan memang bukan tanda kres dalam seni musik (U+266F), dan tanda pagar # (U+0023) tersebut digunakan karena karakter kres dalam seni musik tidak terdapat di dalam keyboard standar. [12]

Standar European Computer Manufacturer Association (ECMA) mendaftarkan beberapa tujuan desain dari bahasa pemrograman C#, sebagai berikut:

1) Bahasa pemrograman C# dibuat sebagai bahasa pemrograman yang bersifat bahasa pemrograman general-purpose (untuk tujuan jamak), berorientasi objek, modern, dan sederhana.

2) Bahasa pemrograman C# ditujukan untuk digunakan dalam mengembangkan komponen perangkat lunak yang mampu mengambil keuntungan dari lingkungan terdistribusi.
3) Portabilitas programmer sangatlah penting, khususnya bagi programmer yang telah lama menggunakan bahasa pemrograman C dan C++.
4) Dukungan untuk internasionalisasi (multi-language) juga sangat penting.
5) C# ditujukan agar cocok digunakan untuk menulis program aplikasi baik dalam sistem klien-server (hosted system) maupun sistem embedded (embedded system), mulai dari perangkat lunak yang sangat besar yang menggunakan sistem operasi yang canggih hingga kepada perangkat lunak yang sangat kecil yang memiliki fungsi-fungsi terdedikasi.
6) Meskipun aplikasi C# ditujukan agar bersifat 'ekonomis' dalam hal kebutuhan pemrosesan dan memori komputer, bahasa C# tidak ditujukan untuk bersaing secara langsung dengan kinerja dan ukuran perangkat lunak yang dibuat dengan menggunakan bahasa pemrograman C dan bahasa rakitan.
7) Bahasa C# harus mencakup pengecekan jenis (type checking) yang kuat, pengecekan larik (array), pendeteksian terhadap percobaan terhadap penggunaan Variabel-variabel yang belum diinisialisasikan, portabilitas kode sumber, dan pengumpulan sampah (garbage collection) secara otomatis.

2.10 Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (suite) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual

Studio mencakup kompiler, SDK, Integrated Development Environment (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe [13].

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam native code (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun managed code (dalam bentuk Microsoft Intermediate Language di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework).