Implementasi Teori Pohon dalam Algoritma Minimax pada Permainan Tic-Tac-Toe

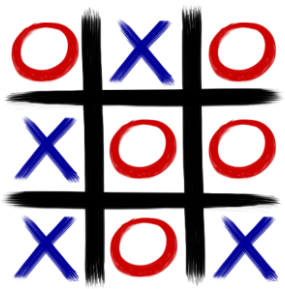
Yudy Valentino - 13517128
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517128@std.stei.itb.ac.id

***Abstrak*—Tic-tac-toe adalah permainan papan dengan menggunakan bidak x dan bidak o sebagai alatnya. Permainan ini memerlukan strategi untuk mencapai kondisi menang. Untuk permainan yang berbasis Artificial Intelligence terdapat algoritma minimax untuk menentukan langkah yang paling optimum bagi komputer, didalam penggunaan algoritma ini menggunakan implementasi dari teori pohon.**

***Kata Kunci*—Algoritma, Minimax, Pohon, Tic-Tac-Toe**

I. PENDAHULUAN

Kemajuan globalisasi pada zaman sekarang, menuntut sebuah kompetisi yang semakin kuat didalam kehidupan manusia, diantaranya kebutuhan sebuah sarana *entertainment* yang semakin diperlukan dalam membantu mengurangi stress. Salah satu area yang sangat berkembang pesat dan banyak melibatkan sumber daya manusia adalaha *gaming industry*. Dengan perkembangan ilmu pengetahuan dan teknologi yang pesat, permainan yang biasanya kita jumpai dalam kehidupan nyata, dapat kita jumpai juga dalam permainan berbasis komputer, salah satu jenis permainan komputer yang cukup terkenal adalah permainan papan.



Gambar 1.1 Permainan Tic-Tac-Toe

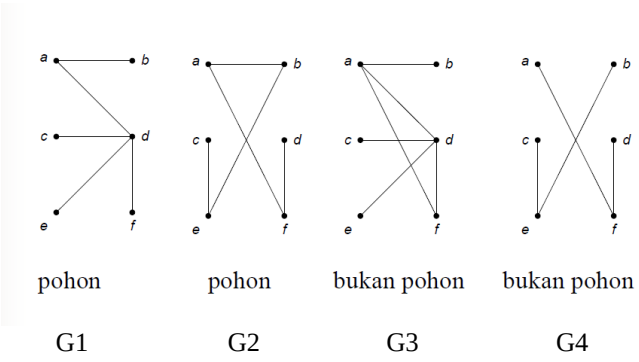
Tic-Tac-Toe merupakan salah satu permainan papan berbasis komputer yang melibatkan strategi untuk mencapai sebuah kemenangan. Tic-tac-toe melibatkan dua orang pemain dengan menggunakan symbol x dan o sebagai bidak di permainan ini. Papan permainan tic-tac-toe berupa unit sel dari matriks berukuran n x n, dengan kondisi awal yaitu papan permainan yang kosong, kemudian kedua player saling bergantian mengisi bidaknya pada kotak yang kosong, hingga tercapainya sebuah kondisi akhir permainan yang menandakan apakah permainan tersebuat menang, seri atau kalah.

Kondisi menang akan tercapai apabila ada tiga buah bidak yang sama, berada pada posisi sejajar, baik itu horizontal, vertical, maupun diagonal. Sedangkan kondisi seri akan tercapai saat kondisi papan terisi penuh tanpa ada yang membentuk tiga buah bidak dalam posisi sejajar, dan kondisi kalah akan tercapai jika lawan lebih dulu meletakkan tiga buah bidak yang sama secara sejajar.

II. LANDASAN TEORI

2.1 Pohon

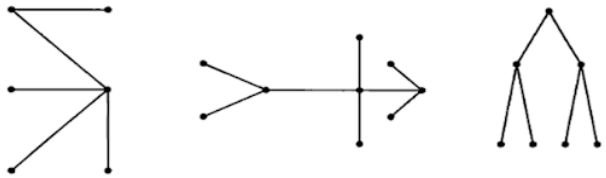
Pohon merupakan salah satu cabang dari matematika diskrit yang berkaitan dengan teori graf. Graf terhubung dan tidak memiliki lintasan sirkuit didalamnya dapat disebut sebagai **pohon.** Konsep dari pohon sangat penting dalam dunia informatika, karena akan sangat bermanfaat dalam pengolahan data.



Gambar 2.1 Contoh dari pohon dan bukan pohon

Pada gambar 2.1.1. , gambar G1 dan G2 disebut pohon karena memenuhi syarata sebuah pohon dari definisi diatas, sedangkan gambar G3 dan G4 tidak dapat disebut pohon, karena di gambar G3 terdapat lintasan sirkuit didalamnya dan di gambar G4, graf yang ada tidak saling terhubung.

Kumpulan dari pohon-pohon yang saling tidak saling terhubung disebut dengan **Hutan.**



Hutan yang terdiri dari tiga buah pohon

Gambar 2.2 Hutan

2.1.1 Sifat-sifat pohon

Sifat dari sebuah pohon sebagai berikut :

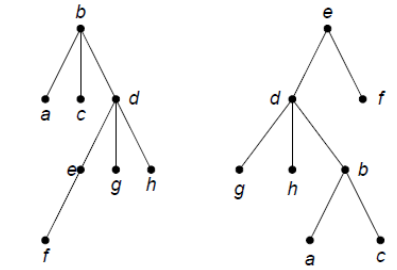
1. Graf G, dengan $G = (V, E)$ adalah sebuah pohon, dengan V sebagai simpul dengan n sebagai *variabel* jumlah dari simpul dan E sebagai sisi dari graf.
2. Setiap pasang dari simpul (V) dihubungkan oleh sebuah lintasan tunggal.
3. Pohon G memiliki $m = n - 1$ buah sisi.
4. Pohon G tidak mengandung sirkuit.
5. Jika ditambahkan sebuah sisi pada pohon G akan menimbulkan sebuah sirkuit.
6. Semua sisi pada pohon G adalah jembatan.

2.1.2 Pohon Merentang

Pohon merentang adalah pohon yang didapatkan dengan cara memutuskan sirkuit yang ada didalam graf, dimana pohon merentang dari graf terhubung adalah upagraf merentang tetapi dalam bentuk pohon.

2.1.3 Pohon Berakar

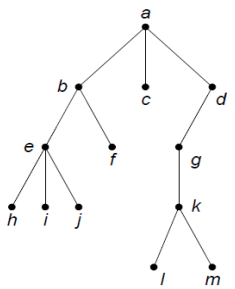
Pohon berakar adalah pohon yang salah satu dari simpulnya dapat dijadikan sebagai titik acu pertama kali. Sisi dari pohon diberi arah yang menjadikannya sebuah graf berarah. Tanda panah pada sisi pohon dapat dihilangkan sebagai perjanjian.



b sebagai akar *e* sebagai akar

Gambar 2.3 Pohon berakar

2.1.4 Terminologi Pohon



Gambar 2.4 Pohon berakar a

2.1.4.1 Anak dan Orangtua

Jika terdapat dua simpul yang saling bersisian dan salah satu dari simpul tersebut berada pada satu tingkat yang lebih rendah dari simpul lainnya, maka simpul yang berada pada satu tingkat lebih rendah dinamakan orangtua *(parent)*, dan yang lebih tinggi satu tingkat diatasnya dinamakan anak *(child)*.

Pada gambar 2.4, *b*,*c*, dan *d* adalah anak-anak dari simpul *a* sedangkan *a* adalah orangtua dari anak-anak tersebut.

2.1.4.2 Lintasan

Lintasan dari simpul v_i ke sebuah simpul v_n adalah runtutan simpul-simpul $v_1, v_2, \ldots, v_n$ sehingga v_i adalah orangtua dari v_{i+1} untuk $1 \leq i \leq n$. panjang sebuah lintasan adalah jumlah sisi yang dilalui dalam sebuah lintasan dengan $n - 1$.

Pada gambar 2.4 lintasan dari *a* ke *j* adalah a,b,e,j. panjang lintasan *a* ke *j* adalah 3.

2.1.4.3 Saudara Kandung

Saudara kandung adalah simpul yang memiliki orangtua yang sama. Pada gambar 2.4, *l* adalah saudara kandung dari *m* karena orangutan mereka sama.

2.1.4.4 Upa-pohon

Upa-pohon atau sub-pohon adalah upagraf pohon dari graf pohon tersebut. Misalkan *x* sebagai akarnya, maka upa-pohonnya adalah subgraf $T' = (V', E')$ sedemikian rupa sehingga V' mengandung x dan semua turunannya dan E' mengandung sisi-sisi dalam yang bersumber dari *x*. kemungkinan dari upapohon yang terbentuk tidaklah unik, melainkan banyak kemungkinan membentuk upapohon dari sebuah pohon .

2.1.4.5 Derajat

Derajat sebuah simpul pohon berakar adalah jumlah dari upapohon sebuah simpul atau jhumlah anak dari sebuah simpul. Terdapat sedikit perbedaan dalam menentukan derajat dari sebuah simpul di graf dengan sebuah simpul di pohon, karena dalam graf, orangtua dari sebuah simpul tidak bisa dihitung sebagai derajat sebuah simpul. Pada gambar 2.4, derajat dari simpul *e* adalah 3.

2.1.4.6 Daun

Seluruh simpul yang tiddak meiliki anak atau simpul yang mempunyai derajat nol dinamakan daun. Pada gambar 2.4, daun dari pohon tersebut adalah *h*, *i*, *j*, *f*, *c*, *l*, *m*.

2.1.4.7 Simpul Dalam

Berkebalikan dari daun, simpul yang tidak berderajat nol, dinamakan simpul dalam. Pada gambar 2.4, simpul dalam dari pohon tersebut adalah *a*, *b*, *d*, *e*, *g*, *k*

2.1.4.8 Tingkat

Tingkat dari sebuah simpul pohon dapat dilihat dari jumlah lintasan yang sudah ditempuh, dari simpul awal akar. Akar mempunyai tingkat terendah yaitu nol. Pada gambar 2.4, jika ingin mencari nilai dari tingkat simpul *e*, maka tinggal dilihat jumlah lintasan dari akar hingga simpul *e* yaitu tiga.

2.1.4.9 Tinggi atau Kedalaman

Tinggi atau kedalaman dari sebuah pohon adlah tingkat maksimum dari sebuah pohon tersebut. Pada gambar 2.4 tinggi pohon tersebut adalah empat.

2.1.5 Pohon Berakar Terurut

Pohon berakar terurut adalah pohon yang memperhatikan urutan dari peletakan anak-anaknya. Hal ini membuat penggambaran pohon menjadi terurut yaitu dari kiri ke kanan. Pohon yang berakar simpul anak kiri, dinamakan upapohon kiri, sedangkan yang berakar simpul anak kanan dinamakan upapohon kanan. Penamaan ini berlaku untuk berapapun jumlah simpul dari pohon tersebut.

2.1.6 Pohon *m-ary*

Pohon *m-ary* adalah pohon yang mempunyai setiap simpul dalam maksimal *m* anak dan merupakan pohon berakar. Pohon tersebut dikatakan penuh apabila setiap simpul didalamnya memiliki tepat *m* anak. Untuk kasus $m = 2$, maka pohon tersebut disebut sebagai pohon biner.

2.1.7 Pohon Biner

Pohon biner adalah pohon *m-ary* yang mempunyai m bernilai tepat 2. Maka setiap simpulnya pun mempunyai maksimal 2 anak, jika penuh maka tiap simpul tepat mempunya 2 anak.

2.2 Algoritma Minimax

Algoritma minimax adalah algoritma yang digunakan dalam permainan dengan dua orang, dan berfungsi sebagai algoritma dalam memprediksi langkah terbaik untuk pergerakan selanjutnya. Secara garis besar algoritma ini berusaha untuk membuat nilai menjadi minimal untuk setiap usaha maksimal dari pemain.

Untuk permainan tic-tac-toe, algoritma ini menganalisa tiga hasil yang mungkin didapatkan di akhir permainan yaitu menang, seri atau kalah. Algoritma minimax ini merupakan perkembangan dari algoritma backtracking yang memanfaatkan pohon keputusan dan breadth-first-search atau depth-first-searc, algoritma ini dapat dikenai batasan masalah agar pohon keputusan yang terjadi tidak begitu luas dan memakan banyak memori.

III. POHON PADA ALGORITMA MINIMAX

A. Data permainan Tic-Tac-Toe

Gambar 3.1 Kondisi menang dalam tic-tac-toe

Meskipun permainan ini sederhana, tetapi secara teori, kemungkinan yang dihasilkan dari permainan ini tidak sesederhana cara memainkannya, untuk lebih rincinya ada 19.683 tata letak papan mungkin (3^9 didapat dari sembilang ruang yang dapat diisi dengan x, o, atau kosong) dan terdapat kemungkinan 362.880 (9!) kemungkinan peletakan x dan o di papan permainan.perlu diketahui hal ini tanpa memperhitungkan kombinasi pemenang. Jika kombinasi pemanang dilakukan maka terdapat 255.168 kemungkinan permainan yang terjadi, dengan asumsi bahwa x akan selalu berjalan pertama kali untuk setiap game, dengan rincian sebagai berikut :

1. 131.184 permainan diselesaikan oleh x
2. 1440 kemungkinan dimenangkan x setelah 5 langkah
3. 47.952 kemungkinan dimenangkan x setelah 7 langkah
4. 81.792 kemungkinan dimenangkan x setelah 9 langkah
5. 77.904 permainan diselesaikan oleh o
6. 5328 kemungkinan dimenangkan o setelah 6 langkah
7. 72.576 kemungkinan dimenangkan oleh o setelah 8 langkah
8. 46.080 permainan diselesaikan dengan seri

Kedelapan langkah diatas dapat disederhanakan menjadi lebih singkat dengan menghilangkan urutan langkah x dan o serta menghilangkan hasil-hasil yang simetris baik itu rotasi, refleksi atau yang lainnya langkah yang didapatkan cukup signifikan dengan menyisahkan 138 hasil unik, tentunya asumsi yang digunakan masih sama seperti kedelapan langkah diatas yaitu dengan x melakukan langkah pertama kali. Berikut secara rinci kemungkinan langkah yang terjadi :

1. 91 hasil unik dimenangkan oleh x
2. 21 kemungkinan dimenangkan oleh x setelah 5 langkah
3. 58 kemungkinan dimenangkan oleh x setelah 7 langkah
4. 12 kemungkinan dimenangkan oleh x setelah 8 langkah
5. 44 hasil unik dimenangkan oleh o
6. 21 kemungkinan dimenangkan oleh o setelah 6 langkah
7. 23 kemungkinan dimenangkan oleh o setelah 8 langkah
8. 3 hasil unik untuk seri

B. Aturan dan strategi permainan

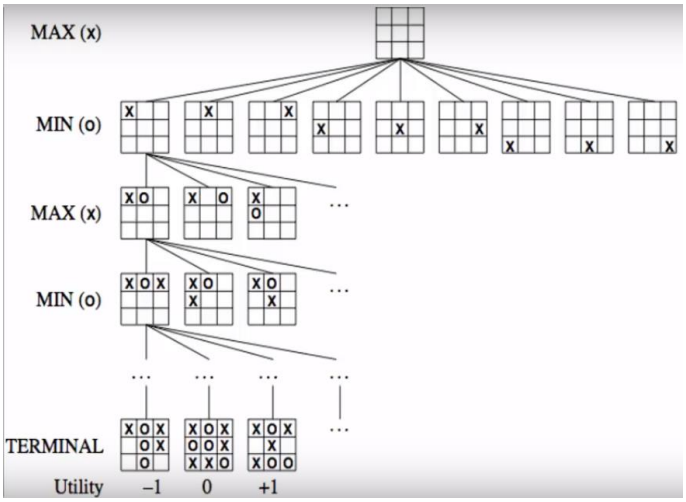
Dalam permainan tic-tac-toe terdapat beberapa aturan yang disepakati oleh kedua belah pihak, diantaranya :

1. Besar papan permainan adalah 3 x 3 dengan kemenangan diraih oleh 3 bidak yang sama secara sejajar.
2. Permainan ini hanya untuk satu lawan satu, serta saling berusaha untuk menang dan menghalangi lawan untuk menang
3. Mengisi kotak papan permainan dengan bidak baik itu x maupun o
4. Setiap pemain hanya bisa melakukan sekali dalam setiap putaran
5. Tidak bisa menimpa kotak yang telah terisi oleh bidak lain
6. Pemenang ditentukan oleh siapa yang lebih dahulu menyusun kondisi menang pada permainan tic-tac-toe
7. Permainan akan dihentikan setelah ada kondisi akhir untuk tic-tac-toe.

Untuk mengalahkan lawan dalam permainan ini terdapat beberapa strategi yang dapat digunakan, yaitu :

1. *Win* : jika pemain memiliki dua gambar bersebelahan, letakkan bidak terakhir agar memperoleh posisi menang. Kemungkinan ini bernilai paling tinggi pada pohon algoritma minimax
2. *Block* : jika sebaliknya ternyata lawan yang mempunyai dua buah bidak yang berselahan, maka Anda dapat memangkas sebagian kemungkinan menangnya dengan menghalangi kemungkinan posisi menangnya.
3. *Fork* : mencari cara agar bisa memenangkan permainan dengan dua cara
4. *Block fork* :
    a. Pilihan 1 : buat dua buah bidak bersebelahan untuk memaksa lawan melakukan block, tanpa membuat lawan mempunyai kondisi fork atau win
    b. Pilihan 2 : jika ada kondisi dimana lawan bisa melakukan fork maka blockir fork tersebut
5. *Center* : menempatkan bidak di tengah papan permainan
6. *Opposite corner* : jika lawan melakukan sudut, maka letakkan pada sudut yang berlawanan
7. *Empty corner* : meletakkan di sudut
8. *Empty side* : meletakkan di posisi sisi dari bujur sangkar papan permainan

C. Implemntasi pohon pada algoritma minimax



Gambar 3.2 Pohon dari algoritma minimax

Minimax merupakan salah satu bentuk proses implementasi pohon *m-ary* pada sebuah *gaming industry*. Implementasi struktur pohon sangat diperlukan dalam pembuatan dan penerapan Artificial Intelligence, dimana minimax ini menjadi basis untuk semua permainan yang mengusung konsep AI.

Pada minimax, computer akan melakukan pengecekan terhadap semua kemungkinan sebagai dampak dari pergerakan bidak pemain. Hal ini memanfaatkan fungsi rekursif, hingga didapatkan hasil yang paling optimal dengan cara membandingkan setiap nilai atau bobot dari simpul tersebut, sehingga komputer mengalami keuntungan yang maksimum.

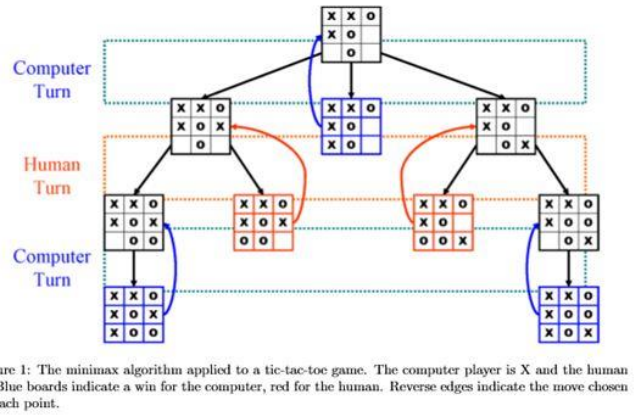


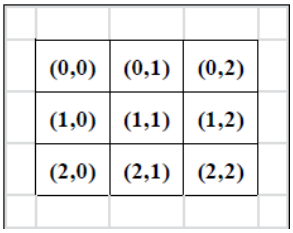
Figure 1: The minimax algorithm applied to a tic-tac-toe game. The computer player is X and the human O. Blue boards indicate a win for the computer, red for the human. Reverse edges indicate the move chosen at each point.

Gambar 3.3 Salah satu upapohon dari pohon algoritma minimax

D. Minimax Equations

Sistem yang digunakan minimax dalam menelusuri kemungkinan yang paling besar untuk menang adalah rekursif, tentunya diperlukan sumber skala besar untuk melakukan pencarian tersebut.

Papan permainan tic-tac-toe berukuran matriks 3 x 3, yang mempunyai 9 buah kotak kecil. Untuk mempermudah penelusuran pohon serta memangkuskan algoritma minimax, maka setiap kotak kecil pada papan permainan tic-tac-toe diberi indeks, baik itu indeks untuk baris (*i*) maupun indeks untuk kolom (*j*).

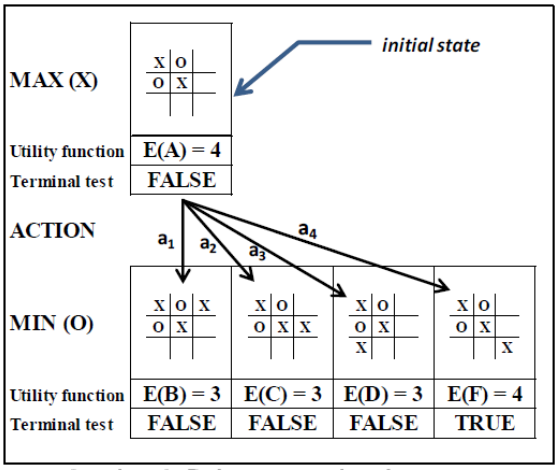


Gambar 3.4 Papan permainan yang sudah diberi nilai indeks

Pergerakan komputer dilakukasn oleh algoritma minimax untuk menentukan langkah terbaik. Parameter sebuah algoritma minimax adalah sebagai berikut :

1. *Initial state* : keadaan papan sebelum algoritma memulai pekerjaannya
2. *Player* : pihak mana yang mendapat untuk menjalankan bidaknya pada sebuah state.
3. *Action* : himpunan keseluruhan pergerakan yang dilakukan oleh player.
4. *Result* : keadaan papan setelah algoritma bekerja pada action
5. *Terminal test* : fungsi pengecekan tujuan atau final state.
6. *Utility function* : nilai evaluasi untuk melakukan fungsi rekursi pada pohon.

Tiap level pohon akan dilakukan penandaaan level, yaitu MIN dan MAX. penamaan MIN untuk mewakili pemain yang sedang bermain, sedangkan penamaan MAX dilakukan untuk komputer saat meletakkan bidak.



Gambar 3.5 Skema pencarian nilai terbaik pada pohon algoritma minimax

Untuk menjelaskan bagaimana dilakukan pencarian, dapat dilihat pada gambar 3.5. Asumsi yang digunakan saat ini adalah komputer dengan *artificial intelligence* menggunakan bidak x dan pemain menggunakan bidak o. Karena *initial state* terjadi pada label MAX, maka sebelum itu pemain meletakkan bidak o nya di matriks indeks atau kotak nomor (1,0).

Kemudian komputer akan menelusuri pohon dengan algoritma minimax untuk mencari kemungkinan yang paling tepat untuk *block* langkah pemain. Pada gambar 3.5, B,C,D, dan F adalah anak dari orang tua A, untuk nilai dari E sendiri akan dihitung dari sisi yang bersisian antar simpulnya.

Kemungkinan untuk komputer menang pada kondisi tersebut adalah 4, dengan perhitungan tiga kotak terisi dengan bidak x semua, yaitu :

1. (2,0), (2,1), (2,2)
2. (0,2), (1,2), (2,2)
3. (0,2), (1,1), (2,0)
4. (0,0), (1,1), ()2,2

Maka kemungkinan menang komputer saat ini adalah 4, yang akan disimpan ke nilai *utility* sementara dari simpul B, sedangkan untuk pemain hanya memiliki sebuah kemungkinan menang pada kondisi gambar 3.5, yaitu :

1. (2,0), (2,1), (2,2)

Sehingga dapat disimpulakan nilai *utility* total dari simpul B adalah E(B) = 4 – 1 = 3. Dengan perhitungan yang sama, dilakukan pengecekan nilai *utility* terhadap calon respon komputer terhadap bidak pemain. Setelah semua nilai dari *utility* simpul anak telah dihitung, maka nilai dari *utility* simpul orangtua dapat dihitung sesuai label yang tertera pada saat itu.

Kita ambil contoh pada gambar 3.5, komputer mendapat label MAX, yang berarti nilai *utility* dari simpul orangtua A, bernilai MAX[E(B), E(C), E(D), E(F)], yang tiap simpulnya mempunyai nilai, maka nilai *utility* simpul orangtua akan berubah menjadi MAX[3, 3, 3, 4] = 4. Dengan demikian nilai 4 dari nilai utility simpul F akan dipilih komputer untuk langkah selanjutnya.

Untuk cabang-cabang yang lain pada pohon algoritma minimax akan ditelurusi seperti cara diatas dengan rekursif dari kondisi *initial state.*

E. Pseudocode Algoritma Minimax

Implementasi algoritma minimax dalam source code untuk mencari nilai terbaik di sebuah cabang pohon algoritma minimax adalah sebagai berikut :

```
Algoritma FindMax
Input: state, piece
Output: MaxValue, BestMove
  set k←0, MaxValue←-10
  for x = 0 to 2 do
    for y = 0 to 2 do
      if Board(x,y) = null then
        Board(x,y)←piece
        E(k)←UtilityFunction(state)
        if E(k)>MaxValue then
          MaxValue←E(k)
          bestrow←x
          bestcol←y
        k←k+1
      Board(x,y)←null
    next y
  next x
  return MaxValue
  return BestMove(bestrow,bestcol)
end
```

Gambar 3.6 Pseudocode algoritma findmax/minimax

IV. KESIMPULAN

Implementasi pohon dalam matematika diskrit tidak hanya sebatas sebuah materi kuliah, namun dapat dikembangkan lebih jauh salah satunya dalam permainan berbasis Artificial Intelligence pada permainan papan tic-tac-toe. Penggunaan pada algoritma minimax membuat komputer mencari langkah yang paling tepat untuk memaksimalkan peluang menang serta meminimalkan peluang menang dari pemain. Namun masih terdapat kekurangan pada algoritma ini dikarenakan harus menelusuri cabang dari pohon algoritma minimax, yang tentu memakan waktu yang cukup lama dalam pengerjaannya sehingga untuk kasus yang lebih besar dari tic-tac-toe diperlukan sebuah batasan masalah agar tidak terlalu kompleks dalam perhitungan

V. UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan terimakasih kepada Tuhan Yang Maha Esa atas berkatnya, penulis dapat menyelesaikan tugas makalah ini. Penulis juga mengucapkan terimakasih kepada kedua orangtua penulis yang selalu mendukung dan mendoakan penulis. Penulis mengucapkan terimakasih kepada seluruh dosen pengampu mata kuliah IF2120 dalam membimbing penulis dalam menyelesaikan tugas makalah ini, dan yang terakhir penulis mengucapkan terimakasih kepada teman-teman penulis peserta mata kuliah IF2120 yang selalu mendukung penulis dalam menyelesaikan makalah ini.

REFERENCES

[1] Munir, Rinaldi. *Matematika Diskrit*, Bandung: Informatika, 2010, edisi ketiga.

[2] http://poetra70.blogspot.com/2015/09/pohon-matematika-diskrit.html. Diakses pada 8 Desember 2018

[3] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf Diakses pada 8 Desember 2018

[4] https://www.academia.edu/17027684/tic_tac_toe . Diakses pada 8 Desember 2018

[5] http://abyantama-el.blogspot.com/2017/01/algoritma-minimax.html Diakses pada 8 Desember 2018

[6] https://www.academia.edu/12206303/Desain_Non-Player_Character_Permainan_Tic-Tac-Toe_dengan_Algoritma_Minimax Diakses pada 8 Desember 2018

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 9 Desember 2017

Yudy Valentino - 13517128