Penerapan Pohon dalam Algoritma Expectiminimax untuk Permainan Stokastik

Jordhy Fernando – 13515004
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515004@std.stei.itb.ac.id

***Abstrak*—Pohon merupakan salah satu konsep yang penting dalam teori graf, karena terapannya yang luas dalam berbagai bidang ilmu, baik dalam bidang ilmu komputer maupun di luar bidang ilmu komputer. Contoh penerapan pohon adalah pohon keputusan (*decision tree*) dan pohon permainan (*game tree*). Kedua pohon tersebut dapat diterapkann untuk membuat kecerdasan buatan yang dapat digunakan untuk memainkan permainan stokastik secara optimal. Makalah ini akan membahas mengenai penerapan kedua jenis pohon tersebut dalam algoritma expectiminimax yang diimplementasikan pada kecerdasan buatan untuk memainkan permainan stokastik secara optimal.**

***Kata Kunci*— Algoritma Expectiminimax, Kecerdasan Buatan, Permainan Stokastik, Pohon.**

I. PENDAHULUAN

Permainan stokastik adalah sebuah permainan dinamis dengan trasisi probabilistik yang dimainkan oleh satu atau lebih pemain. Permainan ini diperkenalkan oleh Lloyd Shapley pada tahun 1953. Permainan stokastik dapat digunakan untuk memodelkan kehidupan nyata seperti masalah ekonomi, politik, jaringan komputer, dan sebagainya karena permainan stokastik menyediakan sebuah model untuk beraneka ragam interaksi dinamis. Contoh permainan stokastik adalah permainan *backgammon*. Dalam makalah ini permainan yang akan digunakan sebagai contoh untuk penerapan algoritma expectiminimax adalah permainan yang dimainkan oleh dua orang yaitu permainan *backgammon*.

Di dalam permainan *backgammon* terdapat faktor peluang yaitu pada angka dadu yang muncul dari hasil pengocokan dadu ketika pemain akan melakukan aksinya. Seiring perkembangan zaman, permainan yang dimainkan oleh dua orang ini pun tidak lagi harus dimainkan oleh dua orang pemain. Seseorang dapat memainkan permainan ini dengan pemain maya yang disebut kecerdasan buatan. Kecerdasan buatan ini dibuat dengan menerapkan algoritma expectiminimax sehingga seorang pemain dapat merasakan seperti bermain dengan orang sungguhan.

Algoritma expectiminimax adalah algoritma yang menerapkan prinsip pencarian *Depth-First Search* (*DFS*) dengan kedalaman terbatas yang mengikutsertakan faktor peluang pada pencariannya untuk menentukan suatu langkah terbaik pada kondisi tertentu. Keterbatasan algoritma ini terletak pada kebutuhan memorinya karena semakin banyak langkah yang mungkin terjadi memori yang dibutuhkan juga akan semakin besar. Struktur data yang biasa digunakan untuk mengimplementasikan algoritma ini adalah struktur data pohon.

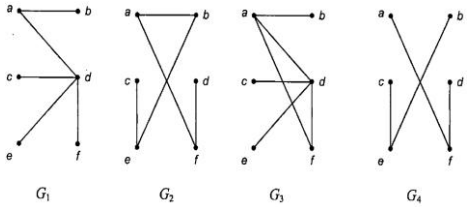
Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Struktur data pohon dipilih dalam menerapkan algoritma expectiminimax karena struktur data ini dapat merepresentasikan suatu kejadian dengan simpul sebagai keadaan saat itu dan anak-anaknya sebagai semua keadaan yang mungkin terjadi, dan sisi sebagai langkah yang dipilih. Daun pada pohon adalah akhir dari kejadian yang dalam hal ini adalah hasil akhir permainan yang dapat berupa menang, kalah, atau seri.

Dengan dibuatnya makalah ini, penulis berharap pembaca menjadi mengerti penerapan pohon pada algoritma expectiminimax yang dapat digunakan dalam pembuatan kecerdasan buatan dan untuk permainan stokastik. Selain itu, penulis juga berharap pembaca menjadi mengerti mengenai algoritma expectiminimax secara umum dan dapat menerapkannya dalam bidang lain.

II. LANDASAN TEORI

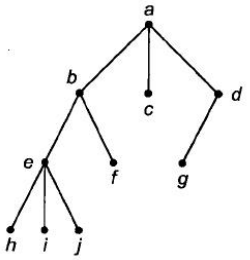
A. Pohon (*Tree*)

Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Istilah pohon pertama kali ditemukan oleh seorang matematikawan Inggris, Arthur Cayley, pada tahun 1857. Gambar 1 adalah contoh graf yang merupakan pohon dan graf yang bukan pohon.



Gambar 1 – Pohon (G1 dan G2) dan bukan pohon (G3 dan G4) [1]

Pada kebanyakan aplikasi pohon, simpul tertentu pada pohon diperlakukan sebagai akar (*root*) dan sisi-sisi pohon yang mengikuti akar tersebut diberi arah menjauhi akar tersebut. Pohon seperti ini dinamakan pohon berakar (*rooted tree*). Sebuah pohon tak-berakar dapat diubah menjadi pohon berakar dengan cara memilih sebuah simpul sembarang pada pohon tersebut sebagai akar sehingga dari sebuah pohon tak-berakar dapat dibuat berbagai pohon berakar yang berbeda tergantung dapat simpul yang dipilih sebagai akar. Gambar 2 adalah contoh pohon berakar.



Gambar 2 – Pohon berakar [1]

Pohon berakar memiliki beberapa terminologi sebagai berikut [1]:

1. Anak (*child* atau *children*) dan Orangtua (*parent*)

Simpul *y* pada suatu pohon berakar dikatakan anak dari simpul *x* pada pohon yang sama jika terdapat sisi dari simpul *x* ke *y* dan *x* disebut sebagai orangtua dari *y*. Pada Gambar 2, *b*, *c*, dan *d* adalah anak-anak dari simpul *a*, dan *a* adalah orangtua dari dari *b*, *c*, dan *d*.

2. Lintasan (*path*)

Lintasan adalah runtunan simpul-simpul yang dilalui dari suatu simpul ke simpul lainnya sedemikian sehingga setiap simpul yang dilalui merupakan orangtua dari simpul yang berikutnya. Pada Gambar 2, lintasan dari *a* ke *j* adalah *a*, *b*, *e*, *j*.

3. Keturunan (*descendant*) dan Leluhur (*ancestor*)

Jika terdapat lintasan dari simpul *x* ke simpul *y* di dalam pohon, maka *x* adalah leluhur dari simpul *y*, dan *y* adalah keturunan dari simpul *x*. Pada Gambar 2, *b* adalah leluhur dari simpul *h*, dan simpul *h* adalah keturunan dari simpul *b*.

4. Saudara kandung (*sibling*)

Dua buah simpul dikatakan sebagai saudara kandung satu sama lain jika kedua simpul tersebut memiliki orangtua yang sama. Pada Gambar 2, simpul *h* adalah saudara kandung simpul *i*.

5. Upapohon (*subtree*)

Upapohon adalah upagraf dari suatu pohon berakar yang mengandung suatu simpul pada pohon berakar sebagai akar dan semua keturunannya beserta semua sisi dalam semua lintasan yang bersasal dari simpul tersebut.

6. Derajat (*degree*)

Derajat sebuah simpul pada pohon berakar adalah jumlah anak atau upapohon pada simpul tersebut. Pada Gambar 2 derajat simpul *e* adalah tiga, dan derajat simpul *g* adalah nol.

7. Daun (*leaf*)

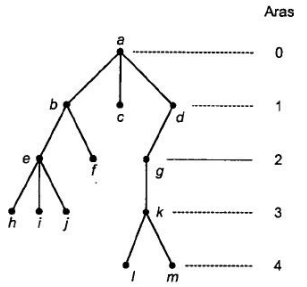
Daun adalah simpul yang berderajat nol (atau tidak mempunyai anak). Pada Gambar 2, simpul *h, i, j, f, c, g* adalah daun.

8. Simpul Dalam (*internal nodes*)

Simpul dalam adalah simpul yang mempunyai anak. Pada Gambar 2, simpul *e*, *b*, *d* adalah simpul dalam.

9. Aras (*level*) atau Tingkat

Akar beraras nol, sedangkan aras simpul lainnya adalah satu ditambah panjang lintasan dari akar ke simpul tersebut. Gambar 3 adalah contoh aras pada pohon berakar.

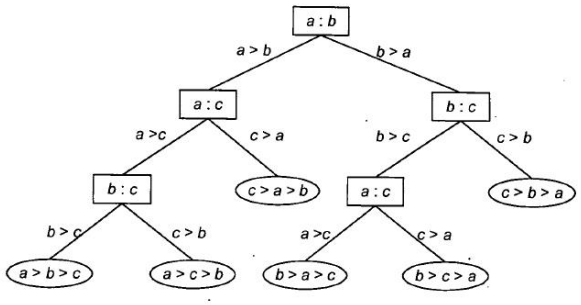


Gambar 3 – Aras pohon berakar [1]

10. Tinggi (*height*) atau Kedalaman *(depth*)

Tinggi atau kedalaman adalah aras maksimum dari suatu pohon berakar. Pohon pada Gambar 3 mempunyai tinggi empat.

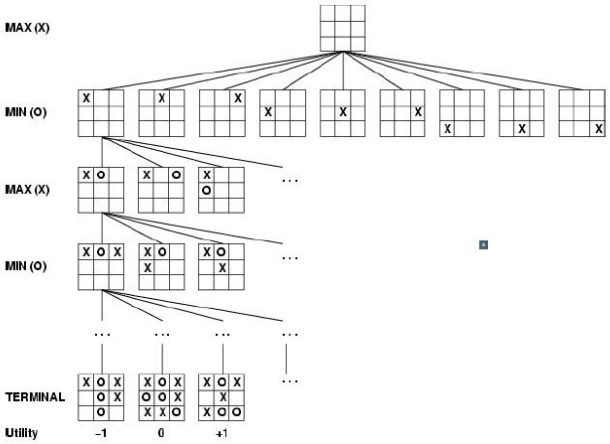
Salah satu aplikasi dari pohon berakar adalah pohon keputusan (*decision tree*). Pohon keputusan digunakan untuk memodelkan suatu persoalan yang terdiri dari serangkaian keputusan yang mengarah ke solusi dengan simpul menyatakan keputusan dan daun menyatakan solusi [1]. Contoh permasalahan yang dapat dimodelkan dengan pohon keputusan adalah mengurutkan tiga buah bilangan *a*, *b*, dan *c*. Pohon keputusan untuk permasalahan ini ditunjukkan pada Gambar 4.



Gambar 4 – Pohon keputusan untuk mengurutkan 3 buah bilangan [1]

Aplikasi lain dari pohon berakar adalah pohon permainan (*game tree*). Pohon permainan digunakan untuk memodelkan suatu permainan yang terdiri dari serangkaian langkah pemain yang mengarah ke akhir permainan (menang, kalah, atau seri) dengan simpul menyatakan posisi pada permainan, sisi menyatakan

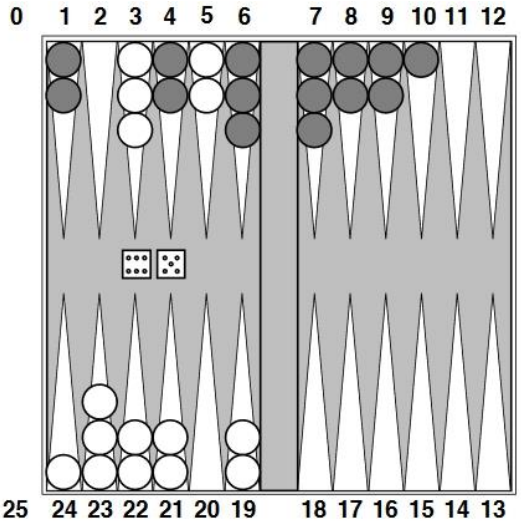
langkah dari seorang pemain, dan daun menyatakan akhir permainan. Gambar 5 adalah contoh pohon permainan pada permainan *tic-tac-toe* yang dimainkan oleh dua orang pemain, *max* dan *min*.



Gambar 5 – Pohon permainan (parsial) untuk permainan *tic-tac-toe* [2]

B. Permainan Stokastik (*Stochastic Game*)

Dalam teori permainan, permainan stokastik adalah sebuah permainan dinamis dengan trasisi probabilistik yang dimainkan oleh satu atau lebih pemain [3]. Permainan stokastik diperkenalkan oleh Lloyd Shapley pada tahun 1953. Permainan stokastik dimainkan dalam serangkaian babak. Pada awal setiap babak, permainan berada pada suatu kondisi (*state*) tertentu. Keadaan permainan pada setiap waktu bergantung secara probabilistik pada keadaan permainan sebelumnya dan aksi pemain. Salah satu contoh permainan stokastik adalah permainan *backgammon* yang dapat dilihat pada Gambar 6. Pada permainan *backgammon* setiap langkah pemain ditentukan oleh angka dadu yang muncul.



Gambar 6 – Permainan *backgammon* [4]

Secara formal, permainan stokastik dinyatakan sebagai *tuple* (*Q*, *N*, *A*, *P*, *R*), dengan:

1. *Q* adalah himpunan berhingga keadaan (*states*) permainan,
2. *N* adalah himpunan berhingga *n* pemain,
3. $A = A_1 \times A_2 \times \ldots \times A_n$ dengan A_i adalah himpunan berhingga aksi pemain ke-*i*,
4. $P : Q \times A \times Q \rightarrow [0,1]$ adalah fungsi transisi probabilistik, $P(q, a, q')$ adalah peluang transisi dari keadaan *q* ke keadaan *q'* setelah diberi aksi *a*, dan
5. $R = r_1, r_2, \ldots, r_n$ dengan $r_i : Q \times A \rightarrow \mathbb{R}$ adalah fungsi hasil (*payoff function*) untuk pemain ke-*i* [4].

Permainan stokastik dapat digunakan untuk memodelkan kehidupan nyata seperti masalah ekonomi, politik, jaringan komputer, dan sebagainya karena permainan stokastik menyediakan sebuah model untuk beraneka ragam interaksi dinamis [3]. Agar hasil analisis permainan menghasilkan prediksi dan rekomendasi yang tepat, data yang digunakan harus memiliki fitur-fitur tertentu. Aplikasi lain dari permainan stokastik adalah pada permainan pasar (*market games*) yang menggunakan uang, misalnya jual-beli saham.

C. Algoritma Expectiminimax

Algoritma Expectiminimax adalah variasi dari algoritma Minimax yang selain mempunyai simpul *MAX* dan *MIN*, juga mempunyai simpul peluang (*chance node*) yang mengambil nilai yang diharapkan (*expected value*) yang merupakan nilai rata-rata suatu kejadian acak [5]. Algoritma ini pertama kali dikemukakan oleh Donald Michie pada tahun 1966.

Nilai-nilai yang akan dipilih dari tiap jenis simpul (anak dari simpul tersebut) pada algoritma Expectiminimax adalah sebagai berikut:

1. Simpul *MAX*

   Nilai yang akan dipilih dari simpul *MAX* adalah nilai utilitas (*utility value*) terbesar di antara anak-anak dari simpul tersebut.
2. Simpul *MIN*

   Nilai yang akan dipilih dari simpul *MIN* adalah nilai utilitas (*utility value*) terkecil di antara anak-anak dari simpul tersebut.
3. Simpul Peluang

   Nilai yang akan dipilih dari simpul peluang adalah nilai rata-rata dari semua nilai utilitas (*utility value*) anak-anak simpul tersebut yang bergantung juga pada peluang untuk mencapai anak tersebut.

Berikut adalah *pseudocode* dari algoritma Expectiminimax [5]:

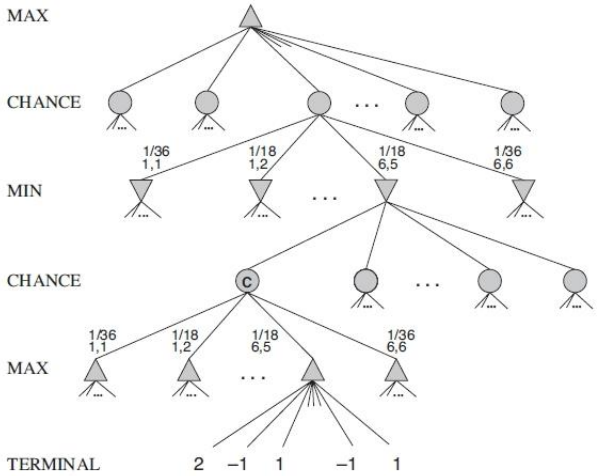
```
function expectiminimax(node, depth)
    if node is a terminal node or depth = 0
        return the heuristic value of node
    if the adversary is to play at node
        // Return value of minimum-valued child
node
        let α := +∞
        foreach child of node
            α := min(α, expectiminimax(child,
depth-1))
```

```
    else if we are to play at node
        // Return value of maximum-valued child
node
        let α := -∞
        foreach child of node
            α := max(α, expectiminimax(child,
depth-1))
    else if random event at node
        // Return weighted average of all child
nodes' values
        let α := 0
        foreach child of node
            α := α + (Probability[child] *
expectiminimax(child, depth-1))
    return α
```

Algoritma ini bekerja secara rekursif dengan basisnya adalah ketika mencapai daun atau kedalamannya nol. Ketika mencapai basis akan dikembalikan nilai heuristik dari simpul yang sedang diproses. Nilai heuristik biasanya berupa transformasi lanjar dari peluang menang dari suatu posisi (atau secara lebih umum, dari utilitas yang diharapkan (*expected utility*) suatu posisi). Tingkat ketelitian ini sangat bergantung pada batas kedalaman yang ditentukan. Semakin tinggi batasnya maka hasilnya akan semakin akurat. Pada bagian rekurens, algoritma ini akan memeriksa simpul sekarang dan mengembalikan nilai berdasarkan simpul yang sedang diperiksa tersebut. Pada dasarnya algoritma ini merupakan algoritma pencarian *Depth-First Search* (*DFS*) dengan kedalaman terbatas.

Biasanya algoritma ini digunakan pada sistem kecerdasan buatan untuk memainkan permainan stokastik dengan jumlah pemain sebanyak dua orang. Simpul *MAX* merepresentasikan giliran kecerdasan buatan, simpul *MIN* merepresentasikan giliran lawan main yang kemungkinan optimal, dan simpul peluang merepresentasikan kejadian acak atau aksi lawan yang tidak optimal [6].

III. Penerapan Pohon dalam Algoritma Expectiminimax untuk Permainan Stokastik



Gambar 7 – Pohon expectiminimax pada permainan *backgammon* [6]

Gambar 7 menunjukkan pohon yang digunakan untuk menerapkan algoritma expectiminimax. Pohon ini merupakan gabungan dari pohon permaianan (*game tree*) dan pohon keputusan (*decision tree*) dengan simpul merepresentasikan keadaan (*state*) permainan, daun merepresentasikan akhir dari permainan, dan sisi merepresentasikan langkah yang dipilih. Penerapan pohon keputusannya terletak pada penentuan simpul yang akan dikunjungi selanjutnya (anak dari simpul yang sedang diproses) oleh algoritma expectiminimax.

Simpul *MAX* merepresentasikan giliran pemain yang menjalankan permainan (*maximizing player*), simpul *MIN* merepresentasikan giliran lawan main (*minimizing player*), dan simpul peluang (*chance*) merepresentasikan faktor peluang pada pelemparan dadu. Pohon yang terbentuk dari algoritma Expectiminimax terdiri dari simpul *MAX*, *MIN*, dan peluang (*chance*) yang tersesun secara selang-seling dengan simpul peluang (*chance*) berada di antara simpul *MAX* dan *MIN*.

Sesuai dengan cara kerja algoritma expectiminimax, simpul *MAX* akan memilih anak yang memiliki nilai utilitas terbesar, simpul *MIN* akan memilih anak dengan nilai utilitas terkecil, dan simpul peluang *(chance*) akan memilih anak yang memiliki nilai paling dekat dengan rata-rata dari semua nilai utilitas anak-anak simpul tersebut. Algoritma ini akan terus berjalan sampai dicapai daun atau simpul terminal atau ketika kedalaman yang ditentukan telah dicapai. Hal ini dapat dilihat pada Gambar 7. Sebelum mencapai daun, simpul terakhirnya adalah simpul *MAX*. Oleh karena itu, daun yang akan dipilih oleh simpul tersebut adalah daun yang bernilai dua karena simpul *MAX* pasti akan memilih simpul yang memiliki nilai utilitas terbesar.

Karena algoritma expectiminimax mempertimbangkan semua kemungkinan (dalam hal ini adalah kemungkinan nilai dadu), kompleksitas dari algoritma ini adalah $O(b^m n^m)$ dengan b adalah faktor percabangan, m adalah tinggi dari pohon, dan n adalah banyaknya nilai dadu yang mungkin. Melihat komplesitas dari algoritma ini, dengan ketinggian atau kedalaman pohon yang cukup kecil saja sudah akan membutuhkan waktu yang cukup lama karena kompleksitas waktunya bergantung secara eksponensial terhadap kedalaman pohon. Ditambah lagi, dalam permainan *backgammon* nilai n adalah 21 dan nilai b biasanya sekitar 20, tetapi dalam situasi tertentu dapat mencapai sampai 4000 untuk pasangan angka dadu yang sama.

Untuk mengatasi masalah tersebut dapat digunakan algoritma *alpha-beta pruning* untuk meminimalisasi pencarian simpul dengan cara tidak memeriksa keadaan yang tidak mungkin terjadi. Algoritma ini akan menganalisis setiap simpul dan memilih simpul yang terbaik dari kumpulan-kumpulan simpul yang ada. Tetapi untuk menerapkan algoritma *alpha-beta pruning* pada pohon ini diperlukan sedikit modifikasi karena ada faktor ketidapastian di dalamnya. Selain itu semakin tinggi pohonnya kemungkinan untuk mencapai suatu simpul tertentu semakin mengecil sehingga algoritma *alpha-beta pruning* akan menjadi semakin kurang efektif karena pada

dasarnya algoritma ini akan memeriksa setiap simpul yang ada dan memilih yang terbaik namun karena peluangnya mengecil akan sulit menentukan simpul yang terbaik.

Salah satu penerapan algoritma expectiminimax adalah pada pembuatan kecerdasan buatan yang dapat memainkan permainan *backgammon* dengan optimal. Kecerdasan buatan ini disebut TDGammon. TDGammon menggunakan pencarian (algoritma expectiminimax) dengan kedalaman atau ketinggian pohon dua dan fungsi evaluasi yang sangat bagus dan hasilnya adalah sebuah pemain maya (kecerdasan buatan) yang dapat memainkan permainan *backgammon* dengan kemampuan setingkat juara dunia. Fungsi evaluasi dari TDGammon dibuat secara otomatis menggunakan teknik *machine learning* yang disebut *Temporal Difference*.

IV. KESIMPULAN

Pohon dapat digunakan sebagai struktur data untuk menerapkan algoritma expectiminimax yang dapat digunakan dalam permainan stokastik. Algoritma expectiminimax ini mempunyai peranan yang penting dalam perkembangan kecerdasan buatan karena algoritma ini dapat digunakan oleh kecerdasan buatan untuk menentukan langkah yang harusnya dilakukannya yang dalam hal ini adalah langkah optimal dalam bermain yang dipengaruhi juga oleh faktor peluang. Pohon merupakan representasi yang sangat tepat karena pohon adalah graf yang setiap simpulnya dapat ditelusuri ketika penulusuran dimulai dari akar pohon.

Algoritma expectiminimax ini masih mempunyai kelemahan terutama pada masalah waktunya yang memiliki kompleksitas yang bergantung secara eksponensial. Namun, masalah ini dapat diatasi dengan memberikan fungsi evaluasi yang bagus pada kecerdasan buatan contohnya pada TDGammon. TDGammon yang hanya menggunakan algoritma expectiminimax dengna kedalaman dua dapat menjadi juara tingkat dunia.

V. UCAPAN TERIMA KASIH

Pertama penulis ingin mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena dengan rahmat dan karunai-Nya penulis dapat menyelesaikan makalah dengan judul "Penerapan Pohon dalam Algoritma Expectiminimax untuk Permainan Stokastik" ini dengan baik. Penulis juga berterima kasih kepada dosen yang memberikan tugas ini, Dr. Ir. Rinaldi Munir, M.T., dan kepada dosen pengajar, Dra. Harlili S., M.Sc., atas bimbingan beliau selama ini dalam mengajar dan memberikan ilmu pada mata kuliah matematika diskrit sehingga penulis mampu membuat makalah ini. Penulis juga berterima kasih kepada rekan-rekan yang telah memberikan semangat dan dorongan kepada penulis.

REFERENSI

[1] R. Munir, "Pohon," dalam *Matematika Diskrit*, ed. 3. Bandung: Informatika, 2010, hlm. 443–475.

[2] Anon., *Games and Adversarial Search*. 2015. [*Online*] Tersedia dalam:http://slazebni.cs.illinois.edu/fall15/lec08_adversarial_search.pdf. [diakses 4 Desember 2016 pukul 16.33 WIB].

[3] L. S. Shapley, "Stochastic Games," *Proceeding of the National Academy of Sciences of the United States of America*, vol. 39, hlm. 1095–1100, 1953.

[4] D. Nau, *Introduction to Game Theory: Stochastic Games*. University of Maryland, 2012. [*Online*] Tersedia dalam: https://www.cs.umd.edu/users/nau/game-theory/8%20Stochastic%20games.pdf. [diakses 4 Desember 2016 pukul 18.05 WIB].

[5] D.Michie, "Game-Playing and Game-Learning Automata," dalam *Advances in Programming and Non-Numerical Computation*, 1966, hlm. 183–200.

[6] S. J. Rusell dan P. Norvig, "Adversarial Search," dalam *Artificial Intelligence: A Modern Approach*, ed. 3. USA: Prentice Hall, 2009, hlm. 177–180.

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, 4 Desember 2016

Jordhy Fernando – 13515004