

**PENERAPAN ALGORITMA MINIMAX MENGGUNAKAN METODE
DEPTH-FIRST SEARCH (DFS) PADA PERMAINAN REVERSI BERBASIS
WINDOWS PHONE**

SKRIPSI

Diajukan sebagai salah satu syarat kelulusan pada
Program Studi Sistem Informasi Jenjang S1 (Strata Satu)
Fakultas Teknik dan Ilmu Komputer

**RAHMAT KURNIA
1.05.09.427**



**PROGRAM STUDI SISTEM INFORMASI
FAKULTAS TEKNIK DAN ILMU KOMPUTER
UNIVERSITAS KOMPUTER INDONESIA
2013**

LEMBAR PENGESAHAN

PENERAPAN ALGORITMA MINIMAX MENGGUNAKAN METODE *DEPTH-FIRST SEARCH* (DFS) PADA PERMAINAN REVERSI BERBASIS WINDOWS PHONE

RAHMAT KURNIA

NIM. 1.05.09.427

Telah disetujui dan disahkan di Bandung sebagai **Skripsi** pada tanggal :

Menyetujui,
Pembimbing

Citra Noviyasari, S.Si., MT.

NIP. 4127.70.26.009

Dekan Fakultas
Teknik dan Ilmu Komputer

Ketua Program Studi
Sistem Informasi

Prof. Dr. H. Denny Kurniadie, Ir., M.Sc.

NIP. 4127.70.015

Syahrul Mauluddin, S.Kom., M.Kom.

NIP. 4127.70.26.021

SURAT KETERANGAN PENYERAHAN HAK EKSKLUSIF

Bahwa yang bertanda tangan dibawah ini, penulis, bersedia :
“Bahwa hasil penelitian dapat dionlinekan sesuai dengan peraturan yang berlaku,
untuk kepentingan riset dan pendidikan”.

Bandung, 22 Juli 2013

Penulis,

Mengetahui,
Pembimbing

Rahmat Kurnia
NIM. 10509427

Citra Noviyasari, S.Si., MT.
NIP. 4127.70.26.009

Catatan

Bila keberatan dengan di-online-kan misalkan **data perusahaan di BAB III/di Bab yang mencantumkan data perusahaan**, ketikan pada catatan ini.

Contoh :

Catatan :

Kecuali Bab III Data perusahaan tidak untuk dionlinekan, dengan alasan.....

Surat keterangan ini wajib ada (scan bentuk Image/gambar/PDF), baik penelitian di perusahaan ataupun bukan di perusahaan dengan mengedit kalimat yang diperlukan, di teks yang berwenang mendandatangani oleh perusahaan atau teks catatan.

DAFTAR ISI

Lembar Pengesahan	i
Lembar Pernyataan Keaslian	ii
Abstrak	iii
<i>Abstract</i>	iv
Kata Pengantar	v
Daftar Isi	vii
Daftar Gambar	xii
Daftar Tabel	xiv
Daftar Simbol	xv
Daftar Lampiran	xvi

BAB I PENDAHULUAN

1.1. Latar Belakang Penelitian	1
1.2. Identifikasi dan Rumusan Masalah	4
1.2.1. Identifikasi Masalah	4
1.2.2. Rumusan Masalah	5
1.3. Maksud dan Tujuan Penelitian	5
1.3.1. Maksud Penelitian	5
1.3.2. Tujuan Penelitian	6
1.4. Kegunaan Penelitian	7
1.4.1. Kegunaan Praktis	7
1.4.2. Kegunaan Akademis	7
1.5. Batasan Masalah	8

1.6. Lokasi dan Waktu Penelitian	9
--	---

BAB II KAJIAN PUSTAKA

2.1. <i>Artificial Intelligence</i> (AI)	10
2.1.1. Sejarah <i>Artificial Intelligence</i> (AI)	12
2.1.2. Komponen <i>Artificial Intelligence</i> (AI)	14
2.1.3. Keuntungan <i>Artificial Intelligence</i> (AI)	15
2.2. <i>Game Playing</i>	15
2.2.1. Definisi <i>Game Playing</i>	15
2.2.2. <i>Type Game</i>	16
2.2.2.1. <i>Perfect Information Game</i>	16
2.2.2.2. <i>Imperfect Information Game</i>	17
2.3. Pohon Permainan	17
2.4. Metode Pencarian (<i>Searching Method</i>)	20
2.4.1. <i>Depth-First Search</i> (DFS)	20
2.5. Algoritma Minimax	22
2.5.1. Karakteristik Algoritma Minimax	25
2.6. Windows Phone	26
2.6.1. Sejarah Windows Phone	26
2.6.2. Perkembangan Windows Phone	27
2.6.3. Versi Windows Phone	29

BAB III OBJEK DAN METODOLOGI PENELITIAN

3.1. Objek Penelitian	31
3.1.1. Permainan Reversi	31

3.1.1.1. Aturan Permainan Reversi	32
3.1.1.2. Strategi Permainan Reversi	33
3.1.2. Algoritma Minimax pada Permainan Reversi	38
3.1.2.1. Analisa Masalah	38
3.1.2.2. Analisa Algoritma	39
3.1.2.2.1. Algoritma Minimax	39
3.1.2.2.2. Fungsi Evaluasi pada Permainan Reversi ..	40
3.1.2.2.3. Metode Pencarian <i>Depth-First Search</i>	42
3.1.2.2.4. Penelusuran <i>Level</i> Permainan	44
3.2. Metodologi Penelitian	45
3.2.1. Metode Penelitian	45
3.2.2. Jenis dan Metode Pengumpulan Data	45
3.2.2.1. Jenis Data	46
3.2.2.2. Metode Pengumpulan Data	46
3.2.3. Metode Pendekatan dan Pengembangan Perangkat Lunak ...	46
3.2.3.1. Metode Pendekatan Perangkat Lunak	46
3.2.3.2. Metode Pengembangan Perangkat Lunak	47
3.2.3.3. Alat Bantu Analisis dan Perancangan	48
3.2.3.3.1. <i>Flow Chart</i>	48
3.2.4. Pengujian Software	48

BAB IV ANALISIS DAN PERANCANGAN APLIKASI

4.1. Penerapan Algoritma Minimax pada Permainan Reversi	49
4.1.1. Basis Pengetahuan Algoritma Minimax pada Permainan	

Reversi	49
4.1.2. <i>Flow Chart</i> Algoritma pada Permainan Reversi	52
4.1.2.1. <i>Flow Chart</i> Algoritma Simpan Bidak	52
4.1.2.2. <i>Flow Chart</i> Algoritma Pemberian <i>Mobility</i> pada Setiap Langkah	53
4.1.2.3. <i>Flow Chart</i> Algoritma Pemberian Nilai Evaluasi pada Setiap Langkah	57
4.1.2.4. <i>Flow Chart</i> Algoritma Pencarian Langkah Terbaik ...	59
4.1.2.5. <i>Flow Chart</i> Algoritma Balik Bidak Lawan	63
4.1.2.6. <i>Flow Chart</i> Algoritma Penentuan Pemenang	67
4.2. Rancangan Antarmuka Pengguna	69
4.2.1. Rancangan Antarmuka Masukan	69
4.2.2. Rancangan Antarmuka Keluaran	72
4.3. Struktur Menu	73
4.4. Perancangan Basis Data	74

BAB V IMPLEMENTASI DAN PENGUJIAN APLIKASI

5.1. Spesifikasi Kebutuhan Perangkat	75
5.1.1. Spesifikasi Perangkat PC (<i>Personal Computer</i>)	75
5.1.2. Spesifikasi Perangkat <i>Smart Phone</i>	75
5.2. Implementasi	76
5.2.1. Implementasi Antarmuka	76
5.2.2. Implementasi Instalasi (<i>Release</i>) Program	78
5.2.3. Implementasi Penggunaan Program	80

5.3. Pengujian Aplikasi	84
5.3.1. Pengujian White Box	85
5.3.2. Pengujian Black Box	88
BAB VI KESIMPULAN DAN SARAN	
6.1. Kesimpulan	91
6.2. Saran	92
Daftar Pustaka	93
Lampiran	95

KATA PENGANTAR



Segala puji dan syukur bagi Allah SWT, Rabb, Malik, Ilah yang menguasai segala kekuasaan dan pemilik segala ilmu. Dengan sifat Maha Pengasih dan Penyayang-Nya memberikan kekuasaan, ilmu kepada siapa yang dikehendaki-Nya. Atas Kehendak-Nya jualah *Alhamdulillahirabbil'alamin* penulis dapat menyelesaikan laporan skripsi ini.

Laporan Skripsi dengan judul **“PENERAPAN ALGORITMA MINIMAX MENGGUNAKAN METODE *DEPTH-FIRST SEARCH* (DFS) PADA PERMAINAN REVERSI BERBASIS WINDOWS PHONE”** disusun guna sebagai Skripsi pada Program Studi Sistem Informasi, Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia (UNIKOM) Bandung.

Selama penyusunan laporan skripsi ini, tidak sedikit bimbingan dan bantuan dari semua pihak, maka dengan rasa tulus penulis ingin mengucapkan terima kasih kepada semua pihak yang telah memberikan dorongan dan semangat baik berupa material maupun spiritual.

1. Dr. Ir. Eddy Suryanto Soegoto, M.Sc. selaku Rektor Universitas Komputer Indonesia.
2. Prof. Dr. H. Denny Kurniadie, Ir., M.Sc. selaku Dekan Fakultas Teknik dan Ilmu Komputer.

3. Syahrul Mauluddin, S.Kom., M.Kom selaku Ketua Program Studi Sistem Informasi.
4. Ibu Citra Noviyasari, S.Si., MT. selaku dosen pembimbing yang telah banyak memberikan bimbingan selama pengerjaan skripsi penulis sehingga dapat terselesaikan dengan baik dan hasil yang baik pula.
5. Ibu Sintya Sukarta, ST., MT. selaku dosen wali kelas IS-10 yang telah banyak memberikan motivasi, pengarahan dan masukan-masukan berharga kepada penulis sehingga dapat diselesaikannya laporan skripsi ini dengan tepat waktu dan hasil yang optimal.
6. Seluruh dosen pengajar yang telah memberikan ilmu kepada penulis mudah-mudahan ilmu yang diajarkan bermanfaat dan menjadi amal kebaikan.
7. Semua saudara dan rekan kerja di BB, IP, GM, dan semua teman sekelas IS-10 yang telah membantu dalam penyusunan laporan ini.
8. Semua pihak yang telah membantu penulis yang tidak dapat disebutkan satu persatu terima kasih atas dorongan, do'a, serta motivasi yang sangat berharga bagi penulis.

Meski jauh dari kesempurnaan, mudah-mudahan laporan skripsi yang penulis susun ini dapat memberikan manfaat bagi diri penulis pada khususnya dan para pembaca pada umumnya. Amiin.

Bandung, Juni 2013

Penulis

DAFTAR PUSTAKA

Kusumadewi, Sri. 2003. *Artificial Intelligence (Teknik dan Aplikasinya)*. Graha Ilmu. Yogyakarta.

Barr, Avron, Edward A. Feigenbaum, Paul R. Cohen. 1982. *The Handbook of Artificial Intelligence*. Wiley Inc. New York.

Rich, Elaine, dan Kevin Knight. 1991. *Artificial Intelligence*. McGraw-Hill Inc. New York.

Coppin, Ben. 2004. *Artificial Intelligence Illuminated*. Jones and Bartlett Publishers Inc. Canada.

Mulyanto, Edy, T. Sutojo, Vincent Suhartono. 2011. *Kecerdasan Buatan*. Andi. Yogyakarta.

Pramudya, Puja. 2011. *Membuat Aplikasi untuk Windows Phone*. Andi. Yogyakarta.

Narimawati, Umi. 2008. *Metodologi Penelitian Kualitatif dan Kuantitatif, Teori dan Aplikasi*. Agung Media. Bandung.

Kristianto, Adi. 2003. *Perancangan Sistem Informasi dan Aplikasinya*. Gava Media. Yogyakarta.

<http://genethello.blogspot.com/2010/02/genethello-kembara-panjang-menuju-batas.html>. *Gene Thello : kembara panjang menuju batas intelijensia buatan*. 1 April 2013.

<http://genethello.blogspot.com/2011/06/game-theory-algoritma-minimax.html>. *Game Theory – Algoritma Minimax*. 1 April 2013.

<http://www.samssoft.org.uk/reversi/strategy.htm>. *Strategy Guide for Reversi & Reversed Reversi*. 5 April 2013.

BAB I

PENDAHULUAN

1.1. Latar Belakang Penelitian

Dalam beberapa tahun terakhir *Artificial Intelligence* (AI) atau kecerdasan buatan telah menjadi sesuatu yang berpengaruh dalam industri *game application*. Hampir semua jenis dan tipe *game application* sekarang membutuhkan AI untuk membuat komputer seolah-olah tampak cerdas. Program pertama yang dibuat AI adalah *game board playing*. Sejarah teori *game application* dimulai dari tahun 1950 ketika komputer mulai dapat diprogram. *Game application board* pertama yang menggunakan AI adalah catur. Pencetus teori *game* dalam AI adalah Konard Zuse yang merupakan penemu pertama komputer yang dapat diprogram dengan bahasa pemrograman pertama, Claude Shannon yang merupakan penemu teori informasi dan Norbert Wiener yang merupakan pencipta teori kontrol modern. Sejak saat itu mulai ada kemajuan dalam standar bermain *game* terutama *game* yang melibatkan papan main, sampai-sampai komputer dapat mengalahkan manusia dalam permainan papan seperti catur, reversi (othello), dan *game* aplikasi yang bersifat versus lainnya.

Game playing pada komputer juga saat ini banyak digunakan oleh beberapa kalangan sebagai media untuk melatih daya pikir dan strategi mereka dalam mengalahkan lawan mainnya. Sebagaimana diketahui bahwa kunci dari kecerdasan

lawan main pada suatu *game playing* adalah kedalaman memperhitungkan kemungkinan-kemungkinan langkah yang ada pada suatu permainan dan setelah itu akan dapat menentukan langkah mana yang paling menguntungkan agar kemenangan dapat diperoleh. Hal tersebut juga menjadi pembeda dari *players* yang memainkan *game board* tersebut.

Bagi komputer perbedaan perhitungan kemungkinan langkah tersebut dibedakan dengan algoritma-algoritma yang telah dikembangkan. Hingga saat ini sudah banyak metode dan algoritma yang dikembangkan untuk permasalahan ini khususnya dan dalam *Artificial Intelligence* umumnya. Salah satu contoh algoritma yang akan penulis bahas disini yaitu algoritma minimax yang digunakan untuk penerapannya pada *game playing*. Algoritma minimax merupakan salah satu algoritma yang digunakan pada permainan papan yang dimainkan oleh 2 pemain dan berbasis *zero-sum* yang artinya keuntungan untuk pemain pertama berarti kerugian untuk pemain kedua.

Ada beberapa metode pencarian yang digunakan untuk algoritma minimax diantaranya yaitu metode *Breadth-First Search* (BFS) dan metode *Depth-First Search* (DFS). Perbedaan metode tersebut yaitu penelusuran yang dilakukan pada setiap node yang terbentuk dalam bentuk *tree*. *Breadth-First Search* (BFS) menelusuri berdasarkan lebar dari suatu *tree*, sedangkan *Depth-First Search* (DFS) menelusuri berdasarkan kedalaman suatu *tree*.

Pada penelitian kali ini penulis akan mencoba membuat sebuah program aplikasi yang menerapkan algoritma minimax tersebut pada *game playing* reversi.

Reversi merupakan salah satu permainan papan yang murni berbasis strategi yang dimainkan oleh 2 pemain pada papan yang berukuran 8 baris dan 8 kolom, kemudian setiap pemain memiliki bidak (koin) yang berbeda warna (biasanya bidak atau koin warna hitam dan putih). Pemain dikatakan menang jika pada akhir permainan (seluruh papan dipenuhi bidak) mempunyai jumlah bidak lebih banyak daripada jumlah bidak lawan. Reversi juga dikenal dengan sebutan Othello karena dipasarkan oleh perusahaan Amerika yang bernama Othello. Reversi atau Othello termasuk ke dalam tipe *game* dengan informasi lengkap yang artinya yaitu suatu *game* di mana pemain mengetahui semua langkah yang terjadi dari dirinya sendiri dan dari lawan mainnya. Oleh karena itu algoritma minimax menjadi pilihan dalam pemilihan kecerdasan buatan untuk permainan ini.

Pada penerapan algoritma minimax ini penulis berencana menggunakan metode *Depth-First Search* (DFS) karena pada aplikasi permainannya akan dibuat *level* permainan dengan tingkatan mudah (*beginner*), sedang (*intermediate*), dan sulit (*expert*). Hal tersebut dibedakan dari penelusuran *level* kedalaman yang dilakukan, tingkatan mudah dengan 3 kedalaman, sedang dengan 5 kedalaman, dan sulit dengan 9 kedalaman. Kedalaman penelusuran tersebut dibuat berdasarkan keahlian atau *level* yang dipilih *human player* yang ingin bermain. Pada penerapan algoritma minimax ini juga penulis berencana mengimplementasikan program aplikasinya pada perangkat *mobile* yang berbasis Windows Phone. Hal ini dikarenakan perangkat *mobile* pada saat ini sedang mengalami perkembangan yang sangat pesat dan hampir setiap kalangan masyarakat saat ini mengenal *smart phone*. Oleh karena itu juga akan mudah untuk mengenalkan permainan reversi yang dapat

melatih daya pikir dan strategi ini pada masyarakat yang belum mengenal permainan ini.

Berdasarkan latar belakang di atas, maka dalam penulisan skripsi ini penulis mengambil judul **“PENERAPAN ALGORITMA MINIMAX MENGGUNAKAN METODE *DEPTH-FIRST SEARCH* (DFS) PADA PERMAINAN REVERSI BERBASIS WINDOWS PHONE**”. Diharapkan program aplikasi ini dapat membantu dalam pengembangan ilmu pengetahuan terutama di bidang teknologi informasi saat ini.

1.2. Identifikasi dan Rumusan Masalah

Identifikasi masalah merupakan rangkuman dari isu masalah yang terjadi yang biasanya menyangkut penyebab permasalahan. Sedangkan rumusan masalah yaitu rumusan dari masalah-masalah yang sudah teridentifikasi yang akan dijawab dalam penelitian yang dilakukan.

1.2.1. Identifikasi Masalah

Masalah-masalah yang dapat teridentifikasi berdasarkan latar belakang diatas di antaranya yaitu :

1. Kebutuhan akan kecerdasan buatan pada setiap program atau aplikasi komputer pada umumnya dan aplikasi *game playing* pada khususnya.
2. Masih terbatasnya pengetahuan masyarakat terhadap *game* reversi yang bisa melatih daya pikir dan strategi.

3. Belum atau masih terbatasnya pembahasan mengenai algoritma minimax dengan menggunakan metode penelusuran *Depth-First Search* (DFS) pada sebuah *game playing* reversi atau othello.

1.2.2. Rumusan Masalah

Berdasarkan masalah yang teridentifikasi diatas, maka di dapatkan rumusan masalah sebagai berikut :

1. Bagaimana menerapkan algoritma minimax dengan metode *Depth-First Search* pada *game* reversi berbasis Windows Phone.
2. Bagaimana merancang *game* reversi berbasis Windows Phone yang menerapkan algoritma minimax dengan metode *Depth-First Search*.
3. Bagaimana mengimplementasikan rancangan *game* reversi berbasis Windows Phone yang menerapkan algoritma minimax dengan metode *Depth-First Search*.
4. Bagaimana mengevaluasi *game* reversi berbasis Windows Phone yang menerapkan algoritma minimax dengan metode *Depth-First Search*.

1.3. Maksud dan Tujuan Penelitian

Maksud penelitian ini adalah untuk memperoleh dan mengumpulkan data atau keterangan yang relevan dengan permasalahan yang akan diteliti. Sedangkan tujuan penelitian harus menjawab dari rumusan masalah penelitian.

1.3.1. Maksud Penelitian

Maksud dilakukannya penelitian ini adalah untuk membuat sebuah aplikasi berbasis Windows Phone yang berkaitan dengan penerapan algoritma minimax

dengan metode *Depth-First Search*, guna mengembangkan ilmu pengetahuan dan wawasan terutama di bidang *Artificial Intelligence* (AI) dan juga sebagai media mengenalkan sebuah permainan papan tradisional sederhana yang bernama reversi atau othello pada masyarakat.

1.3.2. Tujuan Penelitian

Adapun tujuan dari penelitian ini di antaranya yaitu :

1. Untuk menerapkan algoritma minimax dengan metode *Depth-First Search* pada *game* reversi berbasis Windows Phone.
2. Untuk merancang *game* reversi berbasis Windows Phone yang menerapkan algoritma minimax dengan metode *Depth-First Search*.
3. Untuk mengimplementasikan rancangan *game* reversi berbasis Windows Phone yang menerapkan algoritma minimax dengan metode *Depth-First Search*.
4. Untuk mengevaluasi *game* reversi berbasis Windows Phone yang menerapkan algoritma minimax dengan metode *Depth-First Search*.

1.4. Kegunaan Penelitian

Kegunaan penelitian menjelaskan manfaat atau kontribusi yang akan diperoleh dari hasil penelitian dan siapa pihak yang akan mendapatkan manfaat tersebut.

1.4.1. Kegunaan Praktis

Pada penelitian ini, kegunaan praktis yang dapat diperoleh di antaranya yaitu dapat mengenalkan pada masyarakat permainan papan sederhana yang dapat melatih daya pikir dan strategi yang bernama reversi atau othello.

1.4.2. Kegunaan Akademis

Adapun kegunaan akademis dari penelitian ini di antaranya yaitu :

1. Dapat menjadi sumbangsih ilmu pengetahuan bagi peneliti lain ataupun bagi para akademis yang akan mengambil skripsi atau tugas akhir dalam kajian yang sama sekaligus sebagai referensi di dalam dunia penulisan karya ilmiah.
2. Dapat berguna dalam menambah wawasan atau memperkaya wawasan pengetahuan baik teori maupun praktek, belajar menganalisa dan melatih daya pikir dalam mengambil kesimpulan atas permasalahan, khususnya di bidang kecerdasan buatan.

1.5. Batasan Masalah

Untuk memaksimalkan pembahasan agar tidak keluar dari topik maka penulis membatasi permasalahan yang akan dibahas di antaranya :

1. Pada penelitian ini hanya dibahas mengenai bagaimana menerapkan sebuah algoritma yang dalam hal ini yaitu algoritma minimax dengan menggunakan metode *Depth-First Search* pada sebuah aplikasi *game* reversi atau othello berbasis Windows Phone.
2. Metode pembangunan dan pengembangan algoritma yang digunakan adalah metode pengembangan terstruktur karena dalam hal analisis algoritma minimax membutuhkan *rules* atau aturan-aturan terlebih dahulu sebelum membuat program.
3. Implementasi rancangan program menggunakan perangkat *mobile* berbasis *platform* Windows Phone.
4. *Game* reversi ini hanya dimainkan untuk 2 pemain, *human player* dan *computer player*, dimana *human player* hanya bisa bermain melawan *computer player* saja dengan pilihan *level* yaitu mudah, sedang, dan sulit.

BAB II

KAJIAN PUSTAKA

2.1. *Artificial Intelligence* (AI)

Kecerdasan buatan atau lebih di kenal sebagai *Artificial Intelligence*, memiliki beberapa definisi, antara lain :

- a. Menurut Kusumadewi (2003), “Kecerdasan buatan atau *artificial intelligence* merupakan salah satu bagian ilmu komputer yang membuat agar mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia”.
- b. Menurut Avron Barr dan Edward E. Feigenbaum (1982), *Artificial Intellegence* adalah sebagian dari komputer sains yang mempelajari (dalam arti merancang) sistem komputer yang berintelegensi, yaitu sistem yang memiliki karakteristik berpikir seperti manusia.
- c. Menurut Rich dan Knight (1991) kecerdasan buatan merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia.

Berdasarkan definisi di atas, maka kecerdasan buatan menawarkan media maupun uji teori tentang kecerdasan. Teori-teori ini nantinya dapat dinyatakan dalam bahasa pemrograman dan eksekusinya dapat dibuktikan pada komputer nyata. Layaknya manusia yang memiliki otak, komputer juga dapat memiliki perangkat lunak yang bekerja sebagai otak. Manusia dapat menyelesaikan berbagai masalah bukan hanya karena manusia memiliki otak yang mampu menalar dan menganalisa, tapi manusia juga memiliki basis data, pengetahuan, kumpulan informasi, yang semuanya itu diperoleh dari pengalaman, dan belajar.

Semuanya itu kemudian diproses secara otomatis dan sering tidak disadari oleh manusia itu sendiri. Beberapa lingkup utama dalam kecerdasan buatan adalah :

a. Sistem Pakar (*Expert System*)

Sistem Pakar adalah suatu perangkat lunak yang menyimpan pengetahuan seorang pakar, dengan demikian komputer tersebut akan memiliki keahlian layaknya seorang pakar.

b. Pengolahan Bahasa Alami (*Natural Language Processing*)

Dengan Pengolahan Bahasa Alami ini, diharapkan dikemudian hari manusia dapat berkomunikasi dengan komputer menggunakan bahasa sehari-hari.

c. Pengenalan Ucapan (*Speech Recognition*)

Dengan teknologi ini, diharapkan manusia nantinya dapat berbincang-bincang dengan komputer.

d. *Computer Vision*

Teknologi ini merupakan upaya pengenalan citra atau objek visual pada komputer. Dengan menggunakan alat sensor atau *scanner* sebagai indra, maka komputer dapat mengenali objek apa yang ditangkap oleh indranya.

e. *Game Playing*

Mengembangkan berbagai macam bentuk permainan interaktif yang cerdas. Diasumsikan dengan teknologi tersebut dapat menarik minat para penggemar *game*.

2.1.1. Sejarah *Artificial Intelligence* (AI)

Pada awal abad ke 17, Rene Descartes mengemukakan bahwa tubuh hewan bukanlah apa-apa melainkan hanya mesin-mesin yang rumit. Blaise Pascal menciptakan mesin penghitung digital mekanis pertama pada tahun 1642. Pada abad ke 19, Charles Babbage dan Ada Lovelace bekerja pada mesin penghitung mekanis yang dapat diprogram.

Bertrand Russell dan Alfred North Whitehead menerbitkan *Principia Mathematica*, yang merombak logika formal. Warren McCulloch dan Walter Pitts menerbitkan “Kalkulus Logis” Program AI pertama yang bekerja ditulis pada tahun 1951 untuk menjalankan mesin Ferranti Mark di University of

Manchester (UK), sebuah program permainan naskah yang ditulis oleh Christopher Strachey dan program permainan catur yang ditulis oleh Dietrich Prinz. John McCarthy membuat istilah "kecerdasan buatan" pada konferensi pertama yang disediakan untuk pokok persoalan ini, pada tahun 1956. Dia juga menemukan bahasa pemrograman Lisp. Alan Turing memperkenalkan "*Turing Test*" sebagai sebuah cara untuk mengoperasionalkan test perilaku cerdas.

Selama tahun 1960-an dan 1970-an, Joel Moses mendemonstrasikan kekuatan pertimbangan simbolis untuk mengintegrasikan masalah di dalam program Macsyma, program berbasis pengetahuan yang sukses pertama kali dalam bidang matematika. Marvin Minsky dan Seymour Papert menerbitkan *Perceptrons*, yang mendemostrasikan batas jaringan syaraf sederhana dan Alain Colmerauer mengembangkan bahasa komputer Prolog. Ted Shortliffe mendemonstrasikan kekuatan sistem berbasis aturan untuk representasi pengetahuan dan inferensi dalam diagnosa dan terapi medis yang kadangkala disebut sebagai sistem pakar pertama. Hans Moravec mengembangkan kendaraan terkendali komputer pertama untuk mengatasi jalan berintang yang kusut secara mandiri.

Pada tahun 1980-an, jaringan syaraf digunakan secara meluas dengan algoritma perambatan balik, pertama kali diterangkan oleh Paul John Werbos pada 1974. Tahun 1990-an ditandai perolehan besar dalam berbagai bidang AI dan demonstrasi berbagai macam aplikasi. Lebih khusus Deep Blue, sebuah komputer permainan catur, mengalahkan Garry Kasparov dalam sebuah pertandingan 6 game yang terkenal pada tahun 1997. DARPA menyatakan

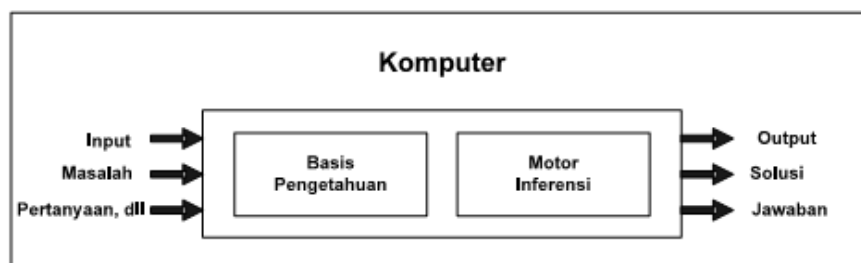
bahwa biaya yang disimpan melalui penerapan metode AI untuk unit penjadwalan dalam Perang Teluk pertama telah mengganti seluruh investasi dalam penelitian AI sejak tahun 1950 pada pemerintah AS.

Tantangan Hebat DARPA, yang dimulai pada 2004 dan berlanjut hingga hari ini, adalah sebuah pacuan untuk hadiah \$2 juta dimana kendaraan dikemudikan sendiri tanpa komunikasi dengan manusia, menggunakan GPS, komputer dan susunan sensor yang canggih, melintasi beberapa ratus mil daerah gurun yang menantang.

2.1.2. Komponen *Artificial Intelligence* (AI)

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

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



Gambar 2.1. Penerapan Konsep Kecerdasan Buatan di Komputer

(Sumber : Kusumadewi, 2003)

2.1.3. Keuntungan *Artificial Intelligence* (AI)

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

- a. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer dan program tidak mengubahnya.
- b. Kecerdasan buatan lebih mudah diduplikasi dan disebar. Mentransfer pengetahuan manusia dari satu orang ke orang lain membutuhkan proses yang sangat lama, dan juga suatu keahlian itu tidak akan pernah dapat diduplikasi dengan lengkap. Oleh karena itu, jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebut dapat disalin dari komputer tersebut dan dapat dipindahkan dengan mudah ke komputer lain.

2.2. *Game Playing*

2.2.1. Definisi *Game Playing*

Game diwakili oleh pohon pencarian di mana *node-node* menunjukkan semua kemungkinan keadaan *game* dan sisi-sisi (*edges*) mewakili langkah antara kedua pemain. *Game playing* merupakan problem pencarian yang di definisikan oleh beberapa komponen berikut :

1. Keadaan Awal (*initial state*), yaitu keadaan yang mendefinisikan konfigurasi awal permainan dan mengidentifikasi pemain pertama yang bergerak.
2. Fungsi Penerus (*successor function*), yang bertugas mengidentifikasi kemungkinan-kemungkinan yang dapat dicapai dari keadaan saat ini. Fungsi ini membuat sebuah daftar pasangan gerak dan keadaan, yang masing-masing pasangan menunjukkan langkah legal dan keadaan yang dihasilkan.
3. *Goal Test*, yang bertugas untuk memeriksa apakah suatu keadaan tertentu adalah keadaan tujuan atau bukan. Keadaan-keadaan di mana permainan berakhir disebut sebagai keadaan terminal.
4. *Path Cost / Utility / Payoff Function*, yang memberikan nilai numerik untuk keadaan-keadaan terminal. Dalam catur misalnya, hasilnya adalah menang, kalah, atau seri, dengan nilai 1, -1, atau 0. Beberapa *game* memiliki rentang yang lebih luas dari hasil yang mungkin.

2.2.2. Type Game

Berdasarkan *type*-nya, *game* dibagi menjadi dua, yaitu *game* dengan informasi lengkap dan *game* dengan informasi tak lengkap.

2.2.2.1. Perfect Information Game

Game dengan informasi lengkap adalah suatu *game* di mana pemain mengetahui semua langkah yang mungkin terjadi dari dirinya sendiri dan dari lawan

main, dan hasil akhir dari permainan mereka. Contoh *game* yang termasuk dalam type ini adalah catur, reversi (othello), dan *tic-tac-toe*.

2.2.2.2. *Imperfect Information Game*

Game dengan *type* ini adalah *game* di mana pemain tidak tahu semua kemungkinan langkah lawan. Contoh yang termasuk *game* ini adalah permainan kartu Poker dan Bridge, karena tidak semua kartu diketahui oleh para pemain.

2.3. Pohon Permainan

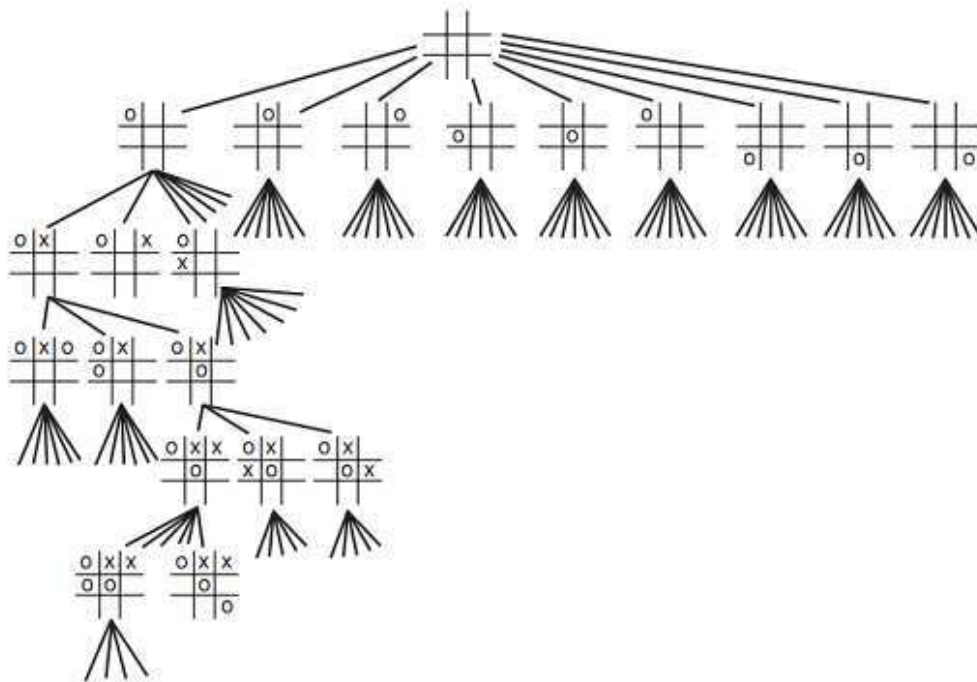
Sebuah pohon permainan dapat merepresentasikan kondisi-kondisi yang mungkin dihadapi pada permainan dimulai dari kondisi yang sedang dihadapi sekarang hingga beberapa kondisi ke depan. Sebuah pohon permainan merupakan representasi grafis dari contoh permainan. Pohon permainan menyediakan informasi akan pemain, hasil, strategi, dan pilihan langkah.

Pohon permainan dapat direpresentasikan dengan sangat baik untuk permainan yang dimainkan oleh dua pemain. Pohon permainan memiliki *root* yang merupakan representasi dari kondisi langkah terakhir atau kondisi di mana langkah belum diambil, *nodes* pada pohon yang merepresentasikan keadaan-keadaan yang mungkin diambil pada permainan, dan *arcs* yang merepresentasikan langkah.

Penggunaan pohon permainan pada permainan yang dimainkan oleh dua pemain direpresentasikan dengan cara berselingan. Untuk *edges* dari tingkat pertama ke tingkat kedua merepresentasikan langkah-langkah yang dapat diambil oleh pemain pertama, sedangkan untuk *edges* dari tingkat kedua ke tingkat ketiga

merepresentasikan langkah-langkah yang dapat diambil oleh pemain kedua, dan begitu seterusnya.

Leaf nodes pada pohon permainan merepresentasikan keadaan akhir pada permainan, di mana permainan tersebut dimenangkan, dikalahkan ataupun seri. Pada permainan yang sederhana, untuk mencapai *leaf nodes* mungkin dapat direpresentasikan, tetapi untuk permainan yang rumit seperti Catur atau Reversi, pencapaian *leaf nodes* sangat tidak dimungkinkan karena percabangan pada pohon permainan yang sangat besar sehingga hanya akan dibatasi dengan beberapa kedalaman saja.



Gambar 2.2. Contoh Pohon Permainan pada *Tic-Tac-Toe*

(Sumber : Ben Coppin, 2004)

Berikut adalah penjelasan pohon permainan *tic-tac-toe* pada Gambar 2.2 :

1. Terdapat *root* yang merupakan keadaan awal di mana permainan belum dimulai dan langkah belum diambil.
2. *Edges* yang menghubungkan tingkat pertama (*root*) dengan tingkat kedua merupakan langkah pemain pertama dan begitu seterusnya. Sehingga pohon permainan tersebut merepresentasikan langkah kedua pemain secara berselingan.
3. Untuk *nodes* pada pohon tersebut merepresentasikan keadaan-keadaan yang dapat diambil oleh pemain yang akan melangkah.
4. Percabangan pertama yang dihasilkan adalah 9, kemudian untuk percabangan berikutnya adalah 8, dan begitu seterusnya hingga mencapai keadaan akhir (*leaf nodes*).

Pada aplikasi permainan Reversi yang akan dirancang, permainan tersebut tidak memiliki unsur kemungkinan atau kesempatan pada permainan dan semua pengetahuan akan keadaan permainan tersebut diketahui secara keseluruhan, artinya pemain tidak dapat menyimpan suatu informasi agar pemain lawan tidak mengetahuinya (terkecuali untuk strategi).

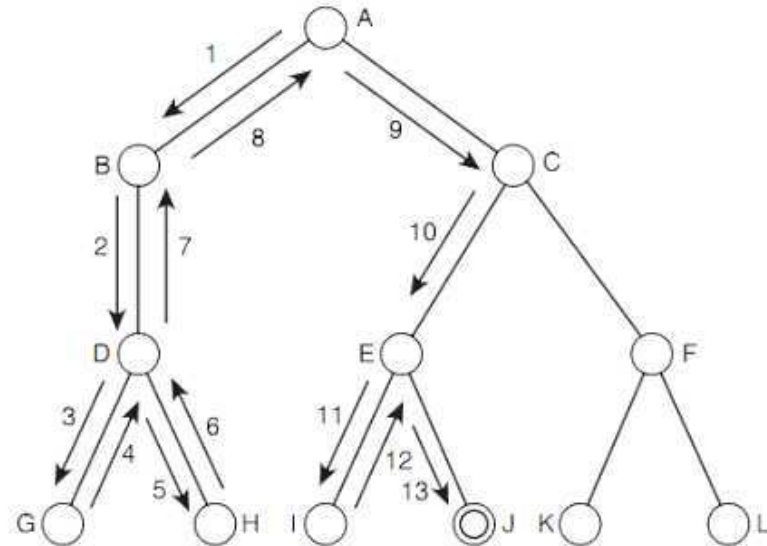
Aplikasi permainan Reversi yang akan dirancang berbasis *zero-sum*, artinya pendapatan poin untuk pemain yang satu, merupakan kehilangan poin untuk pemain yang satunya lagi. Apabila pemain yang satu menang, maka pemain yang satunya lagi kalah. Kemungkinan lainnya hanya seri.

2.4. Metode Pencarian (*Searching Method*)

Penulis telah mengetahui apa itu pohon permainan, namun penulis belum mengetahui bagaimana cara yang baik untuk menelusuri pohon permainan tersebut sehingga mendapatkan nilai-nilai dari pohon permainan tersebut. Di sini penulis akan menganalisa sebuah algoritma pencarian pada pohon permainan yaitu *Depth-First Search* (DFS).

2.4.1. *Depth-First Search* (DFS)

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 2.3 mengilustrasikan tentang bagaimana cara kerja algoritma DFS.



Gambar 2.3. Penelusuran Pohon Permainan dengan Metode DFS

(Sumber : Ben Coppin, 2004)

Pada Gambar 2.3 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* J.

Langkah-langkah cara kerja algoritma DFS adalah sebagai berikut :

1. Masukkan *root* ke dalam struktur data tumpukan (*stack*).
2. Ambil simpul dari tumpukan teratas, dan diperiksa apakah simpul merupakan solusi.

3. Jika simpul merupakan solusi, maka pencarian selesai dan hasil dikembalikan.
4. Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut ke dalam tumpukan.
5. Jika tumpukan kosong dan setiap simpul sudah ditelusuri, pencarian selesai dan solusi tidak ditemukan.
6. Ulangi pencarian dari poin kedua.

Untuk pencarian pada pohon yang memiliki kedalaman yang sangat dalam ataupun hampir tak terhingga, algoritma DFS tidak bekerja dengan baik karena solusi bisa jadi tidak ditemukan, dan apabila ditemukan pasti membutuhkan waktu operasi yang sangat lama. Namun algoritma DFS, bisa lebih cepat menemukan solusi secara kebetulan apabila jalur yang ditelusuri pertama kali merupakan jalur menuju solusi pada pohon tersebut.

Kompleksitas dalam kondisi terburuk pada algoritma DFS adalah $O(bm)$ dengan keterangan b merupakan unsur percabangan pada pohon dan m merupakan tingkat kedalaman yang ada pada pohon permainan. Sedangkan kompleksitas untuk kondisi terbaik pada algoritma DFS adalah $O(1)$, dengan kondisi pencarian simpul pertama langsung menemukan solusi.

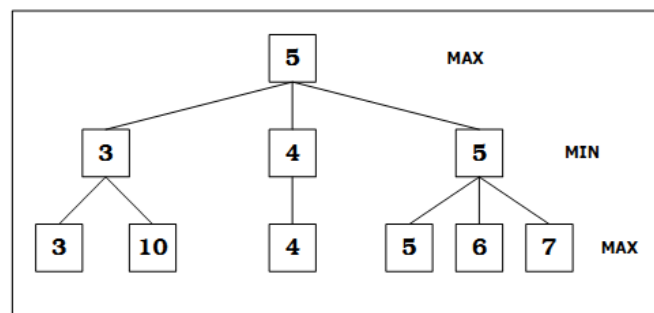
2.5. Algoritma Minimax

Algoritma Minimax dikembangkan oleh John von Neuman pada tahun 1928 dan sekaligus merupakan salah satu prinsip algoritma yang paling dikenal digunakan dalam *game*. Prinsip ini hanya berlaku pada *game* yang memiliki dua

pemain secara bergantian, yaitu manusia melawan komputer seperti dalam permainan catur, reversi (othello), dan *tic-tac-toe*. Algoritma Minimax juga merupakan salah satu algoritma yang berbasis *zero-sum* yang artinya keuntungan untuk pemain pertama berarti kerugian untuk pemain kedua.

Minimax bekerja sangat baik bila seluruh keadaan langkah yang mungkin terjadi dari pemain dapat diketahui seperti permainan catur, reversi (othello), atau *tic-tac-toe*. Algoritma Minimax tidak bisa digunakan pada permainan seperti poker karena komputer tidak bisa melihat semua langkah pemain lawan yang mungkin terjadi.

Untuk mengimplementasikan algoritma minimax, diperlukan sebuah metode untuk mengukur seberapa baik sebuah posisi saat itu. Metode ini disebut sebagai fungsi evaluasi. Fungsi evaluasi merupakan sebuah *heuristik*, yang kenyataannya sangat sulit untuk menentukan berapa nilai presisi pada sebuah posisi saat itu. Fungsi evaluasi yang baik akan memperhitungkan dari banyak aspek yang berbeda, seperti posisi papan, jumlah *pieces relative* terhadap lawan, kontrol papan, kontrol kunci tertentu pada papan, dan mobilitas.



Gambar 2.4. Sebuah Pohon Pencarian untuk Sebuah Permainan Logika

(Sumber : Ben Coppin, 2004)

Dalam Minimax, MAX adalah sebutan bagi pemain (komputer) yang bertujuan untuk mendapatkan nilai maksimum dan MIN adalah sebutan bagi lawan (*human*) yang bertujuan untuk mendapatkan nilai minimal. Berikut prosedur minimax :

1. Tandai masing-masing *level* dari ruang pencarian sesuai dengan langkahnya pada *level* tersebut.
2. Mulai pada *node* daun (*node* paling bawah), dengan menggunakan fungsi evaluasi, berikan nilai pada masing-masing *node*.
3. Arah menjalar ke atas jika *node* orangtua adalah MAX, pilihlah nilai terbesar yang terdapat pada *node* anak dan berikan nilai ini pada MAX (*node* orangtua).
4. Arah menjalar ke atas jika *node* orangtua adalah MIN, pilihlah nilai terkecil yang terdapat pada *node* anak dan berikan nilai ini pada MIN (*node* orangtua).

Berikut ini adalah *pseudocode* algoritma minimax :

```

MinMax (GamePosition game) {
    return MaxMove (game);
}
MaxMove (GamePosition game) {
    if (GameEnded(game)) {
        return EvalGameState(game);
    }
    else {
        best_move <- {};
        moves <- GenerateMoves(game);
        ForEach moves {
            move <- MinMove(ApplyMove(game));
            if (Value(move) > Value(best_move)) {
                best_move <- move;
            }
        }
        return best_move;
    }
}

```

```

    }
  }
  MinMove (GamePosition game) {
    best_move <- {};
    moves <- GenerateMoves(game);
    ForEach moves {
      move <- MaxMove(ApplyMove(game));
      if (Value(move) > Value(best_move)) {
        best_move <- move;
      }
    }
    return best_move;
  }
}

```

2.5.1. Karakteristik Algoritma Minimax

Terdapat karakteristik dari algoritma minimax menurut Edy Mulyanto (2010 : 442) yaitu sebagai berikut :

a. *Completeness*

Minimax dikatakan lengkap jika pohon pencariannya berhingga. Sebagai contoh, permainan catur memiliki pohon pencarian yang sangat besar, tetapi berhingga. Jadi, minimax lengkap dalam kasus permainan catur.

b. *Optimalitas*

Algoritma minimax akan optimal jika melawan pemain yang langkahnya juga optimal. Jika lawan tidak optimal maka minimax akan lebih optimal lagi.

c. *Kompleksitas Waktu*

Algoritma minimax melakukan eksplorasi pencarian lengkap kedalaman pertama, kompleksitas waktu adalah $O(b^m)$, di mana b adalah faktor percabangan dan m adalah kedalaman pohon. Untuk permainan catur, b kira-kira sama dengan 35 dan m kira-kira sama dengan 100. Oleh karena itu, kompleksitas waktunya kira-

kira sekitar $O(35^{100})$, apalagi untuk permainan-permainan yang lebih besar dalam permainan catur. Oleh karena itu, menerapkan algoritma minimax pada permainan catur dan permainan yang lebih besar dari permainan catur tentu saja tidak efisien. Solusi yang tepat harus dicari agar kompleksitas waktu tidak bersifat eksponensial. Salah satu pendekatan standar yang bisa diterapkan adalah pencarian batas kedalaman, di mana pencarian dibatasi pada kedalaman beberapa node dari node root (node saat ini). Node pada kedalaman tertentu itu diasumsikan sebagai node daun dan nilai fungsi evaluasinya diperkirakan.

d. Kompleksitas Ruang

Kompleksitas ruang dari algoritma minimax adalah $O(b^m)$.

2.6. Windows Phone

Berdasarkan penelusuran pada wikipedia (http://id.wikipedia.org/wiki/Windows_Phone) yang diakses pada tanggal 3 Januari 2013, ensiklopedia bahasa Indonesia dan berdasarkan situs resmi dari Windows Phone, Windows Phone merupakan salah satu sistem operasi dari keluarga Microsoft yang berjalan pada perangkat *mobile* yang sekaligus merupakan pengganti dari sistem operasi Windows Mobile. Windows Phone ditujukan pada pasar konsumen perusahaan seperti Nokia, dll.

2.6.1. Sejarah Windows Phone

Pada tanggal 11 Februari 2011 di London, CEO Microsoft Steve Ballmer dan CEO Nokia Stephen Elop mengumumkan kerja sama di antara kedua perusahaan, di mana Windows Phone akan ditetapkan menjadi sistem operasi utama

pada telepon genggam buatan Nokia di masa mendatang yang mana sebelumnya Nokia menggunakan sistem operasi Symbian.

Menurut Puja Pramudya, (2011:6) Windows Phone (sebelumnya dikenal sebagai Windows Phone 7) adalah sistem operasi genggam yang dikembangkan oleh Microsoft, dan merupakan pengganti dari Windows Mobile. Microsoft memperkenalkan Windows Phone pada tgl 15 Februari 2010 di pameran MWC, Barcelona. Sistem operasi ini dirilis di Amerika Serikat pada tgl 8 november 2010. Windows Phone mendukung sampai 25 bahasa. Sampai sekarang ini, Marketplace yang merupakan tempat untuk membeli aplikasi, telah dapat diakses di 35 negara.

2.6.2. Perkembangan Windows Phone

Dengan Windows Phone, Microsoft menciptakan *user interface* yang baru menggunakan *design language* yang di beri nama Modern Style UI. Selain itu, perangkat lunak ini terintegrasi dengan aplikasi pihak ketiga dan berbagai layanan Microsoft serta menerapkan persyaratan minimal untuk perangkat keras yang memakai sistem operasi ini.

Microsoft mengumumkan beberapa persyaratan minimal telepon genggam agar bisa menggunakan Windows Phone yaitu dapat dilihat pada tabel berikut :

Tabel 2.1. Persyaratan Minimal Windows Phone

Persyaratan Minimal Windows Phone
<i>Capacitive</i> , Layar 4-point <i>multi-touch</i> WVGA dengan resolusi (480x800)
ARM v7 "Cortex/Scorpion" – Snapdragon QSD8X50, MSM7X30, dan MSM8X55
DirectX9 <i>rendering-capable</i> GPU
256 MB of RAM dengan sedikitnya 8 GB Flash memory
Accelerometer dengan kompas, <i>ambient light</i> sensor, <i>proximity</i> sensor, Assisted GPS, dan Gyroscope
5-megapixel kamera dengan flash
FM radio tuner
Six (6) <i>dedicated hardware buttons</i> – <i>back</i> , <i>start</i> , <i>search</i> , <i>2-stage camera</i> , <i>power/sleep</i> dan <i>Volume Up - Down</i> .

Sumber : http://id.wikipedia.org/wiki/Windows_Phone

Pada tanggal 29 Oktober 2012 kemarin Microsoft merilis versi terbaru yaitu Windows Phone 8.0. Perbedaan yang mencolok dari versi sebelumnya yaitu pada Windows Phone 8.0 Microsoft menggunakan kernel Windows NT dari versi sebelumnya yang menggunakan kernel Windows CE. Kernel Windows NT ini yaitu kernel yang diterapkan juga pada sistem operasi Windows 8 pada PC dan Tablet. Jadi untuk pengembangan aplikasi pada Windows Phone 8.0, maka akan kompatibel juga pada Windows 8 karena memiliki kernel yang sama.

2.6.3. Versi Windows Phone

Sistem operasi Windows Phone terdapat beberapa versi, diantaranya :

1. Windows Phone 7.5 (Mango)

Diumumkan pada Februari 2011 di Mobile World Congress. Steve Ballmer mengumumkan *update* besar untuk Windows Phone 7 karena menjelang akhir tahun, dan meluncurkan fitur termasuk versi *mobile* Internet Explorer 9 yang mendukung standar web yang sama dan kemampuan grafis dengan versi desktop, Twitter integrasi untuk People Hub, multi-tasking dari aplikasi pihak ketiga dengan menunda tugas aktif saat beralih ke tugas aktif dalam pandangan, dan Windows Live SkyDrive.

2. Windows Phone 7.6 (Tango)

Tango akan minor *update*, mirip dengan *update* "NoDo". Ini membantu Telepon Windows untuk berjalan di perangkat-murah dengan 256 MB RAM. Ini juga mendukung Skype dan integrasi Google+ melalui hub teman, pesan, dan *chatting*. Selain itu, Tango mengubah ukuran kontrol media untuk tampilan lebih nyaman, mendapatkan dukungan untuk 120 bahasa (menambahkan dukungan untuk 85 bahasa tambahan), membuat perangkat lebih mudah diakses secara internasional, tambahkan C++ dukungan aplikasi, dan memungkinkan pengelompokan ubin layar awal ke dalam folder.

3. Windows Phone 8 (Apollo)

Apollo adalah nama kode untuk generasi berikutnya dari Windows Phone, seperti ditegaskan dalam sebuah seminar MSDN pada Agustus 2011. Engadget percaya bahwa Apollo sebenarnya mengacu pada Windows Phone 8. Hal diyakini *update* akan menambahkan teknologi NFC, dual-core dukungan chipset, lebih tinggi resolusi layar ke panggung, dan akhirnya mengarah pada konvergensi dari sistem Microsoft operasi untuk PC, ponsel, tablet, konsol video dan permainan melihat sebagai Xbox 360 juga menggunakan antarmuka Metro. Hal ini berspekulasi bahwa Microsoft memindahkan kernel Windows Phone dari Windows CE ke dalam Windows inti yang terisolasi secara informal disebut bahwa MinWin adalah dasar dari Windows 7 dan Windows 8, pengaturan ulang API untuk Windows dan Windows Phone.

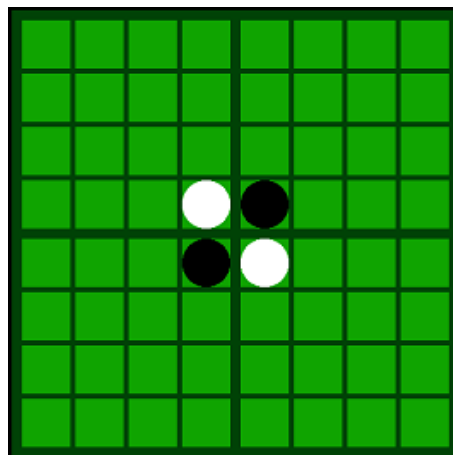
BAB III

OBJEK DAN METODOLOGI PENELITIAN

3.1. Objek Penelitian

3.1.1. Permainan Reversi

Permainan Reversi adalah permainan yang dimainkan oleh dua orang pemain. Permainan ini dimainkan di atas papan persegi yang terdiri dari 8 baris dan 8 kolom kotak-kotak kecil. Peralatan lain yang dibutuhkan adalah koin berwarna gelap dan koin berwarna terang (umumnya warna hitam dan warna putih) masing-masing sebanyak 64 buah. Pada awal permainan akan diletakkan dua koin hitam dan dua koin putih pada tengah-tengah papan. Untuk lebih jelasnya, dapat dilihat pada Gambar 3.1.



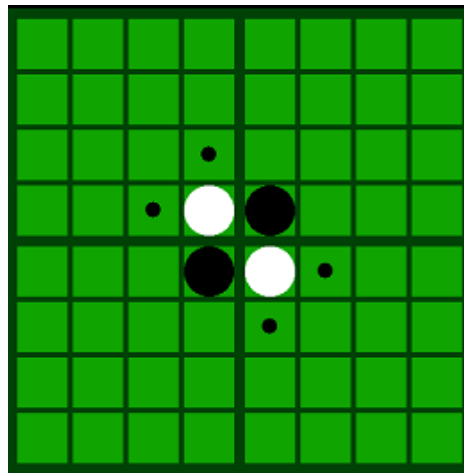
Gambar 3.1. Keadaan awal permainan Reversi

3.1.1.1. Aturan Permainan Reversi

Aturan permainan Reversi secara umum adalah sebagai berikut :

1. Pemain yang menggunakan koin hitam akan bermain terlebih dahulu.
2. Bila pemain koin hitam yang akan bermain, maka koin hitam harus diletakkan di kotak yang dapat dilompati oleh koin hitam lainnya. Dan begitu juga kondisinya untuk pemain yang menggunakan koin putih.

Gambar 3.2 memperlihatkan kondisi awal permainan Reversi.



Gambar 3.2. Kotak yang mungkin (giliran langkah hitam)

3. Apabila salah satu pemain tidak dapat bermain karena tidak ada kotak yang sesuai dengan aturan nomor 2, maka pemain yang satunya lagi yang bermain.
4. Apabila kedua pemain sama-sama tidak dapat mengambil langkah lagi, maka permainan berakhir.

Permainan Reversi berakhir dengan beberapa kondisi sebagai berikut :

1. Semua kotak pada papan sudah penuh diisi koin-koin.
2. Belum semua kotak pada papan diisi tetapi koin-koin yang ada pada papan hanya tersisa koin-koin dalam 1 warna saja.
3. Kedua pemain setuju untuk mengakhiri permainan (bisa seri ataupun menyerah).

Pemenang pada permainan Reversi ditentukan dengan jumlah koin-koin yang ada pada papan permainan. Pemain dengan jumlah koin yang lebih banyak adalah pemenangnya. Jadi, pada permainan Reversi ini para pemain diharapkan dapat memikirkan strategi-strategi agar mendapatkan jumlah koin terbanyak pada akhir permainan.

3.1.1.2. Strategi Permainan Reversi

Reversi atau Othello merupakan permainan yang mudah untuk dipelajari, tetapi sangat memerlukan waktu lama untuk menjadi handal. Seperti judul buku yang ditulis mantan juara dunia othello Brian Rose “*Othello : A Minute to Learn – A Lifetime to Master*”. Oleh karena itu diperlukan strategi dalam permainan papan ini, berikut hal-hal yang perlu diperhatikan dalam bermain reversi atau othello :

1. Jumlah Keping Koin

Tujuan permainan reversi atau othello adalah untuk memiliki jumlah keping sebanyak-banyaknya di akhir permainan. Karena inilah pemain pemula othello selalu berusaha melangkah di kotak-kotak di mana dia bisa membalik keping lawan sebanyak-banyaknya. Strategi ini disebut *greedy strategy* (strategi rakus).

Greedy Strategy memang diperlukan di akhir permainan untuk memperbanyak jumlah keping, tetapi di awal dan di tengah permainan strategi ini akan memberikan kondisi buruk dengan alasan *mobility* (dijelaskan di point 4). Untuk itu disarankan supaya menjaga jumlah keping sedikit di awal dan tengah permainan, tetapi tentu saja harus memperbanyak keping di akhir permainan.

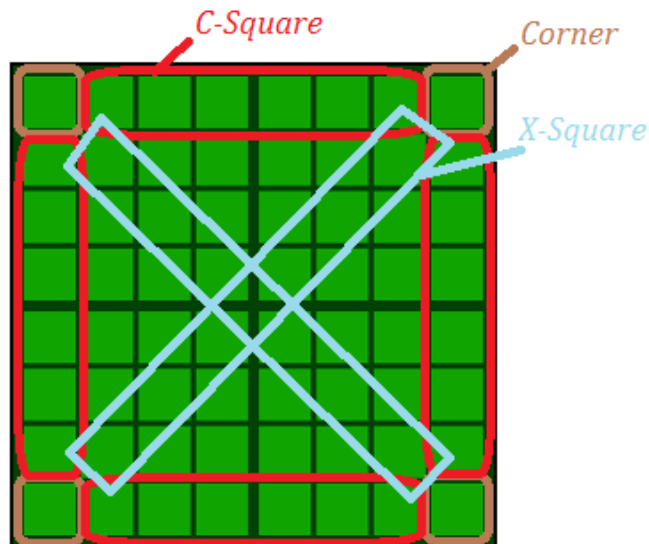
2. Kotak Sudut (*corners*), *X-Squares*, dan *C-Squares*

Jika pemain berhasil meletakkan keping di kotak sudut (*corner*), maka keping itu tidak akan pernah bisa dibalik oleh lawan, karena tidak ada keping yang dapat mengapitnya. Bahkan bermula dari keping sudut ini, seorang pemain dapat menggunakannya untuk membalik keping-keping lawan di sekitarnya. Itulah sebabnya kotak sudut adalah kotak paling penting pada permainan othello. Untuk itu pemain disarankan supaya berusaha mendapatkan kotak sudut di awal dan di tengah permainan.

Kotak *X-Square* adalah kotak di sebelah diagonal kotak sudut. Kotak ini dipandang sebagai kotak yang paling berbahaya, karena ketika pemain meletakkan keping di kotak ini, lawan dapat segera memanfaatkannya untuk mendapatkan kotak sudut di sebelahnya. Karena itulah disarankan untuk tidak melangkah di kotak *X-Square* pada awal dan tengah permainan.

Kotak *C-Square* adalah kotak di sebelah kotak sudut baik secara horizontal atau vertikal. Kotak ini juga dipandang berbahaya setelah *X-Squares*, karena pemain lawan dapat memanfaatkannya untuk mendapatkan sudut dengan trik-trik

tertentu. Untuk itu disarankan supaya tidak melangkah ke kotak *C-Square* di awal dan tengah permainan.



Gambar 3.3. Posisi *Corner*, *C-Square*, dan *X-Square* pada Papan Reversi

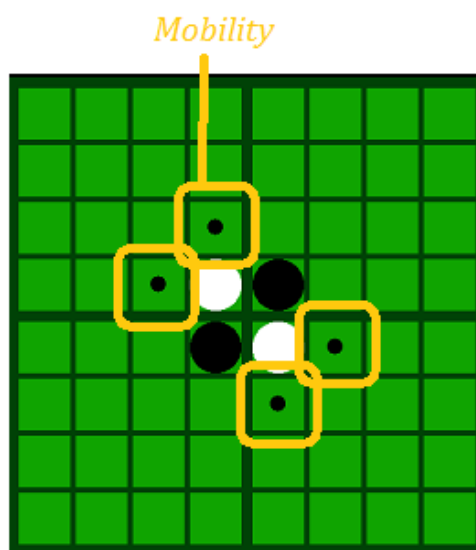
3. Keping Stabil

Keping yang tidak dapat diapit oleh keping lawan, sehingga tidak dapat dibalik disebut keping stabil. Selain itu keping di sisi papan yang bersambung dengan keping sudut juga merupakan keping stabil, atau keping yang semua baris horizontal, vertikal, dan diagonalnya bersambung dengan keping stabil lainnya juga merupakan keping stabil. Keping stabil ini menjadi penentu kemenangan secara mutlak karena tidak bisa dibalik lagi menjadi keping lawan. Untuk itu disarankan supaya pemain memperbanyak jumlah keping stabil di semua tahapan permainan.

4. *Mobility*

Di permulaan permainan hitam dapat melangkah di empat buah kotak. Selanjutnya putih dapat melangkah di tiga buah kotak, demikian seterusnya jumlah kotak di mana pemain dapat melangkah, yaitu dapat mengapit keping lawan, berubah-ubah tergantung langkah lawan sebelumnya. Jumlah kotak di mana pemain dapat melangkah ini disebut *mobility*.

Semakin banyak *mobility*, maka semakin banyak kemungkinan pilihan langkah yang bagus, sebaliknya semakin sedikit *mobility* maka semakin sedikit pilihan langkahnya sehingga semakin besar kemungkinan hanya tersisa langkah-langkah buruk. Karena itu disarankan untuk memperbanyak *mobility* di semua tahapan permainan. *Mobility* berkaitan erat dengan jumlah keping. Apabila jumlah keping pemain banyak, maka semakin susah pemain mengapit keping lawan, sehingga *mobility* menjadi sedikit. Sebaliknya ketika jumlah keping pemain sedikit, akan semakin mudah pemain mengapit keping lawan sehingga *mobility* meningkat.



Gambar 3.4. *Mobility* pada Permainan Reversi

5. Keping Tepi (*Frontier Disc*)

Ini adalah keping yang terletak di tepi kotak kosong. Ketika jumlah keping tepi yang dimiliki banyak, maka semakin besar kemungkinan lawan dapat melangkah di tepi kotak kosong sampingnya, sehingga mobility lawan meningkat. Untuk itu disarankan supaya pemain mempersedikit keping tepi di semua tahapan permainan. Strategi mempersedikit keping tepi ini juga berarti pemain harus berusaha supaya kepingnya selalu tersambung satu sama lainnya dan berada di dalam kepungan keping lawan.

6. *Parity* (Ganjil – Genap)

Di akhir permainan, kotak kosong di papan sering terbagi menjadi beberapa kelompok, dan *parity* adalah kesempatan untuk melangkah terakhir di sebuah kelompok kotak kosong. Pemain yang dapat melangkah terakhir di sebuah kelompok kotak kosong akan diuntungkan karena mendapat kesempatan terakhir untuk membalik keping-keping lawan menjadi kepingnya. Karena itulah disarankan supaya pemain menjadi pemain terakhir yang melangkah di semua kotak kosong di akhir permainan.

Caranya adalah dengan menghitung jumlah kotak kosong di sebuah kelompok. Ketika jumlahnya genap maka pemain jangan melangkah kesana, biarkan lawan yang melangkah kesana. Sebaiknya ketika jumlahnya ganjil, maka usahakan untuk melangkah kesana sebelum lawan melangkah kesana. Dengan demikian pemain akan menjadi yang terakhir untuk melangkah di semua kelompok kotak kosong.

3.1.2. Algoritma Minimax pada Permainan Reversi

3.1.2.1. Analisa Masalah

Pada permainan reversi ini terdapat beberapa masalah yang teranalisis oleh penulis terkait bagaimana permainan reversi pada aplikasi diantaranya yaitu :

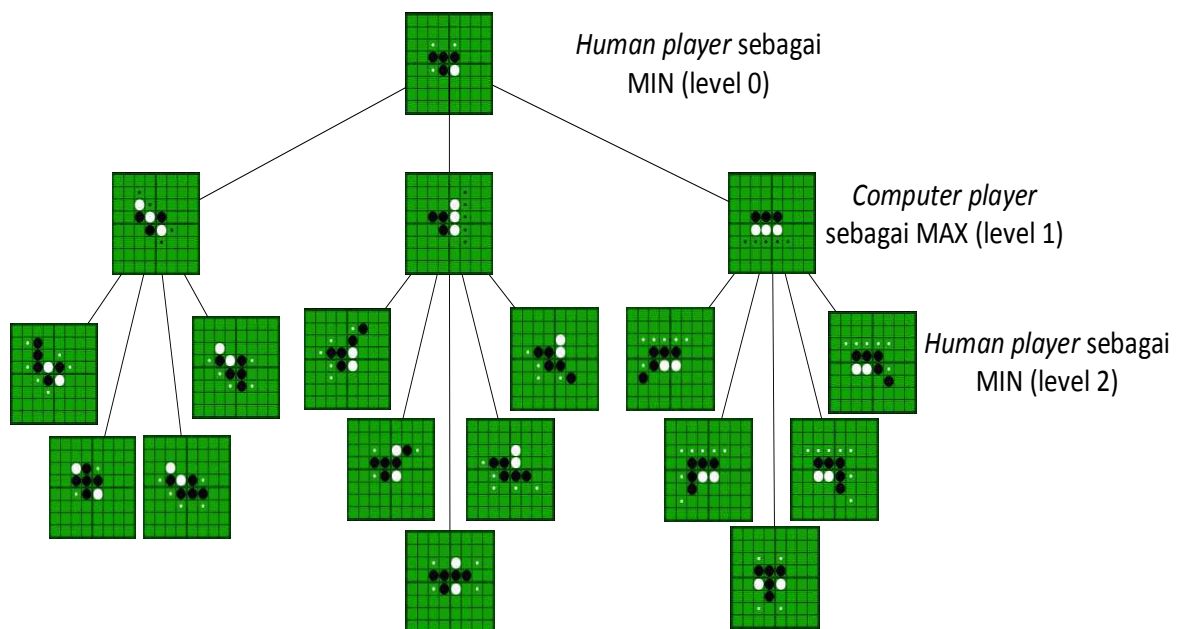
1. Penentuan *mobility* dari berbagai posisi bidak yang terjadi dengan mencari bidak lawan yang dapat diapit atau dibalik oleh bidak pemain secara horizontal, vertikal, dan diagonal.
2. Penentuan langkah yang terbaik dari beberapa *mobility* yang ada dengan memberikan nilai evaluasi pada setiap *mobility*.
3. Pemberian nilai evaluasi pada setiap *mobility* ditentukan dengan memperhitungkan beberapa faktor diantaranya adalah jumlah bidak, posisi *corner*, posisi *x-corner*, posisi *c-corner*, *mobility*, dan bidak stabil (artinya bidak yang tidak dapat dibalik oleh bidak lawan).
4. Pemakaian algoritma minimax tidak dapat terlepas dari metode pencarian yang dalam hal ini metode *depth-first search* sehingga akan digunakan untuk melakukan pencarian nilai evaluasi terbaik berdasarkan posisi MIN (*rival player*) atau MAX (*current player*) dari algoritma minimax.
5. Tidak mungkin dilakukan penelusuran secara menyeluruh pada setiap kemungkinan langkah sehingga harus dilakukan optimalisasi langkah menggunakan prosedur *alpha-beta pruning* sehingga akan memotong langkah yang tidak perlu.

3.1.2.2. Analisa Algoritma

3.1.2.2.1. Algoritma Minimax

Sebagaimana diketahui sebelumnya algoritma minimax merupakan algoritma yang sangat cocok untuk *type game playing* dengan dua player bergantian. Dalam minimax, MAX adalah sebutan bagi *computer player* yang bertujuan untuk mendapatkan nilai maksimal dan MIN adalah sebutan bagi *human player* yang bertujuan untuk mendapatkan nilai minimal.

Pada permainan reversi kali ini MAX akan ditetapkan sebagai *computer player* yang memiliki bidak berwarna putih dan MIN ditetapkan sebagai *human player* yang memiliki bidak berwarna hitam. Sebagai contoh langkah permainan reversi yang menggunakan konsep minimax dapat dilihat pada Gambar 3.3.



Gambar 3.5. Pohon Pencarian pada Permainan Reversi (2 kedalaman)

Pada keadaan sekarang (*root*) dapat dilihat *human player* (bidak hitam) sudah mendapatkan giliran bermain dan *computer player* (bidak putih) akan melakukan langkah berikutnya. Berdasarkan keadaan ini, *computer player* membuat pohon pencarian yang berisi kemungkinan langkah yang bisa diambil oleh *computer player* dan *human player* dengan kedalaman tertentu.

Kemudian *computer player* menganalisis keadaan tersebut berdasarkan prosedur minimax. *Node root* (akar) menunjukkan keadaan papan sekarang, sedangkan *node-node* anak menunjukkan kemungkinan langkah-langkah yang dapat dijalankan.

3.1.2.2.2. Fungsi Evaluasi pada permainan Reversi

Berdasarkan informasi yang dikutip dari sebuah situs atau blog milik Bowo Prasetyo (<http://genethello.blogspot.com/2010/02/genethello-kembara-panjang-menuju-batas.html>) yang diakses pada tanggal 23 Maret 2013. Secara umum, fungsi evaluasi yang sering digunakan pada aplikasi othello atau reversi mempertimbangkan beberapa variabel utama, antara lain :

1. *Discs* yaitu selisih jumlah keping yang dimiliki pemain dan yang dimiliki lawan main setelah dilakukannya sebuah langkah. Pada umumnya semakin tinggi nilai *discs* maka semakin baik langkah tersebut.
2. *Mobility* yaitu jumlah langkah yang dimiliki lawan main setelah dilakukannya sebuah langkah. Ketika jumlah langkah yang dimiliki lawan main banyak, maka memungkinkannya memilih langkah-

langkah yang bagus sehingga perlu memperkecil nilai *mobility* ini supaya lawan tidak mempunyai pilihan langkah yang bagus.

3. *Corners* yaitu jumlah sudut yang ditempati pemain dikurangi jumlah sudut yang ditempati lawan main setelah dilakukannya sebuah langkah. Posisi sudut pada permainan othello sangat penting, karena keping pada posisi ini bersifat permanen, yaitu tidak bisa dibalik lagi oleh lawan sampai permainan berakhir. Untuk itu perlu memperbesar nilai *corners* ini sehingga jumlah sudut yang kita tempati lebih banyak daripada yang ditempati lawan.
4. *Stables* yaitu jumlah keping stabil, artinya keping yang tidak bisa dibalik oleh lawan sampai akhir permainan.
5. *Frontier* yaitu jumlah keping tepi yang ada, keping yang terletak di tepi kotak kosong. Ketika jumlah keping tepi yang kita miliki banyak, maka semakin besar kemungkinan lawan dapat melangkah di tepi kotak kosong sampingnya, sehingga *mobility* lawan meningkat.

Dengan demikian perhitungan score sebuah langkah pada permainan othello pada umumnya akan mengikuti rumus :

$$\text{score} = \text{wd} * \text{discs} + \text{wc} * \text{corner} + \text{ws} * \text{stables} + \text{wm} * \text{mobility} + \text{wf} * \text{frontier}$$

Keterangan :

wd = -3 (bobot untuk discs)

wc = 5 (bobot untuk corners)

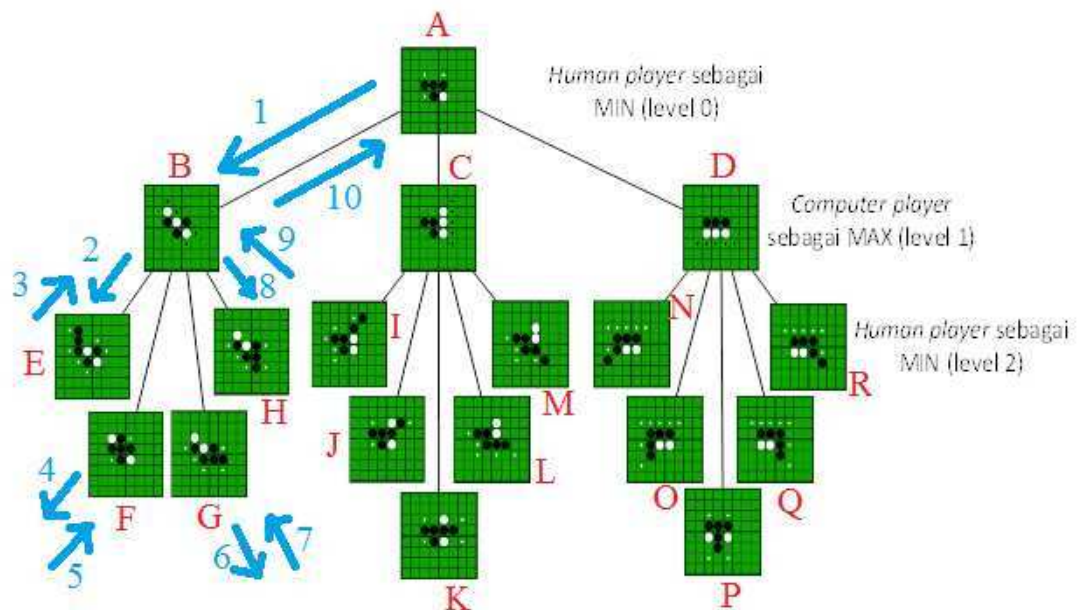
ws = 9 (bobot untuk stables)

wm = 7 (bobot untuk mobility)

wf = -5 (bobot untuk frontier)

3.1.2.2.3. Metode Pencarian *Depth-First Search* (DFS)

Adapun untuk proses penelusuran pada pohon pencarian yang diperlihatkan Gambar 3.5 diatas dilakukan menggunakan metode pencarian kedalaman pertama (*depth-first search*). Metode ini dikatakan pada beberapa literatur merupakan metode yang paling cocok digunakan pada algoritma minimax karena *depth-first search* pada awalnya menelusuri pada kedalaman tertentu (*leaf node*) terlebih dahulu.



Gambar 3.6. Metode DFS pada Pohon Permainan Reversi

Pada gambar 3.6 dapat dilihat langkah awal penelusuran dimulai dari *node root* (*node A*) ke *node B* kemudian ke *node E* yang merupakan *leaf node* (karena dalam hal ini hanya 2 kedalaman), dari *node E* ini mulai ada perhitungan nilai evaluasi pada kondisi tersebut dan akan menjadi bahan pertimbangan bagi *node B* sebelum melangkah ke *node F*, *G*, dan *H*. Jika ternyata *node F*, *G*, atau *H* memiliki nilai evaluasi lebih baik dari *node E* (dalam hal ini dicari nilai evaluasi terbesar atau MAX) maka *node* itulah yang menjadi tetapan untuk *node B* yang tentu akan menjadi bahan pertimbangan juga oleh *node A* dari *node C* dan *D*. Tanda panah pada gambar menunjukkan alur proses penelusuran dan nomor menunjukkan *step by step* yang dilakukan. Panah nomor 3, 5, 7, 9, 10 merupakan proses *backtrack* yang berarti kembali pada *node* sebelumnya. Demikian seterusnya hal yang sama juga dilakukan pada *node C* beserta *child node*-nya dan *node D* beserta *child node*-nya.

3.1.2.2.4. Penelusuran *Level* Permainan

Dalam hal ini metode pencarian atau metode penelusuran DFS (*Depth-First Search*) yang telah dijelaskan diatas akan dimanfaatkan untuk menciptakan perbedaan *level* permainan. Berikut beberapa *level* permainan yang akan dibuat berdasarkan perbedaan penggunaan metode DFS (*Depth-First Search*) :

a. *Beginner*

Beginner (pemula) adalah tingkatan pada permainan dengan memanfaatkan metode DFS (*Depth-First Search*) terhadap 3 kedalaman penelusuran suatu pohon pencarian yang terbentuk.

b. *Intermediate*

Intermediate (menengah) adalah tingkatan pada permainan dengan memanfaatkan metode penelusuran DFS (*Depth-First Search*) terhadap 6 kedalaman penelusuran suatu pohon pencarian yang terbentuk.

c. *Expert*

Expert (mahir) adalah tingkatan pada permainan dengan memanfaatkan metode penelusuran DFS (*Depth-First Search*) terhadap 9 kedalaman penelusuran suatu pohon pencarian yang terbentuk.

Jadi suatu tingkatan permainan dapat mempengaruhi langkah dari *Artificial Intelligence* (AI) karena suatu kecerdasan dalam bermain suatu *game playing* ditentukan oleh seberapa dalam memperhitungkan kemungkinan-kemungkinan

langkah yang terjadi dan dapat mengambil langkah terbaik dari kemungkinan tersebut.

3.2. Metodologi Penelitian

Metode penelitian merupakan suatu mekanisme dan teknik untuk mencari, memperoleh, mengumpulkan, atau mencatat data yang dapat digunakan untuk keperluan menyusun penelitian.

3.2.1. Metode Penelitian

Pada penelitian kali ini penulis akan menggunakan metode penelitian tindakan (*Action Research*).

Metode penelitian tindakan merupakan metode penelitian yang bertujuan untuk mengembangkan keterampilan-keterampilan baru, cara pendekatan baru, atau produk pengetahuan yang baru dan untuk memecahkan masalah dengan penerapan langsung di lapangan (Narimawati, 2008).

Oleh karena definisi di atas, penulis memperoleh gambaran mengenai perancangan aplikasi berbasis *mobile* yang dapat menerapkan algoritma minimax pada permainan reversi.

3.2.2. Jenis dan Metode Pengumpulan Data

Jenis data adalah mengenai dari mana data diperoleh. Apakah data diperoleh dari sumber langsung (data primer) atau data diperoleh dari sumber tidak langsung (data sekunder). Sedangkan Metode Pengumpulan Data merupakan teknik atau cara yang dilakukan untuk mengumpulkan data. Metode menunjuk suatu cara sehingga dapat diperlihatkan penggunaannya melalui angket, wawancara, pengamatan, tes, dokumentasi dan sebagainya.

3.2.2.1. Jenis Data

Jenis data pada penelitian kali ini hanya berupa data sekunder yaitu berupa data-data hasil penelitian yang telah ada sebelumnya, berupa fakta-fakta, dan data-data hasil studi pustaka dari berbagai buku mengenai objek penelitian.

3.2.2.2. Metode Pengumpulan Data

Untuk metode pengumpulan data yang dilakukan pada penelitian ini penulis hanya menggunakan metode pengumpulan data berupa studi pustaka dari beberapa literatur, referensi, dan buku-buku terkait dengan objek penelitian diatas.

3.2.3. Metode Pendekatan dan Pengembangan Perangkat Lunak

Pendekatan yang dilakukan untuk pengembangan dilakukan untuk menggantikan atau membuat sistem / perangkat lunak / aplikasi. Dalam hal ini terdapat metode yang digunakan untuk melakukan pendekatan dan pengembangan terhadap penelitian ini.

3.2.3.1. Metode Pendekatan Perangkat Lunak

Metode pendekatan perangkat lunak yang digunakan penulis dalam membangun perangkat lunak di sini yaitu menggunakan metode pendekatan evolusioner. Alasannya karena dengan metode pendekatan ini penulis dapat fokus menentukan *rules* atau aturan-aturan pada *artificial intelligence* yang akan dibuat dan kemudian baru dilakukan pembuatan aplikasi *artificial intelligence* berdasarkan *rules* tersebut.

3.2.3.2. Metode Pengembangan Perangkat Lunak

Ada salah satu tipe atau jenis yang sering dipakai pada metode pengembangan evolusioner yaitu metode pengembangan pemrograman evolusioner. Dalam tipe ini tujuan proses adalah untuk mengetahui kebutuhan-kebutuhan dan mengembangkan definisi kebutuhan yang lebih baik untuk sistem. Pemrograman evolusioner sulit untuk membuat spesifikasi sistem secara rinci, oleh karena itu pemrograman evolusioner banyak digunakan dalam pengembangan sistem kecerdasan buatan yang menyamai kemampuan manusia.



Gambar 3.7. Model Pengembangan Evolusioner

(Sumber : Adi Kristianto, 2003)

Berikut penjelasan dari gambar 3.7 diatas :

Pada tahapan awal yang dilakukan yaitu dengan membuat deskripsi secara garis besar atau penjelasan secara garis besar tentang sistem kecerdasan buatan

yang akan dibuat, pada tahapan ini mencakup di dalamnya penentuan *rules* pada sistem kecerdasan buatan.

Kemudian tahapan yang paling penting yaitu spesifikasi, pengembangan, dan validasi yang merupakan aktivitas berulang sampai tercapainya perangkat lunak yang menjawab kebutuhan *user*.

3.2.3.3. Alat Bantu Analisis dan Perancangan

3.2.3.3.1. *Flow Chart*

Flow Chart merupakan penggambaran alur proses suatu program yang diwakili dengan bagan-bagan atau *shapes* tertentu. Dalam kasus ini *Flow Chart* digunakan untuk menggambarkan alur algoritma yang diterapkan pada aplikasi reversi dan alur kerja dari aplikasi secara keseluruhan.

3.2.4. Pengujian Software

Adapun teknik pengujian yang dilakukan terhadap perangkat lunak yaitu menggunakan teknik pengujian *Black Box* yaitu dengan menguji fungsi-fungsi pada perangkat lunak, sedangkan untuk pengujian algoritma dalam hal penerapannya pada perangkat lunak menggunakan teknik pengujian *White Box*.

BAB IV

ANALISIS DAN PERANCANGAN APLIKASI

4.1. Penerapan Algoritma Minimax pada Permainan Reversi

Penerapan algoritma minimax pada permainan reversi ini akan di rancang menggunakan *array* dua dimensi 8x8 yang merepresentasikan setiap kotak pada papan permainan reversi. Representasi papan main tersebut akan ditetapkan menggunakan sebuah variabel global bernama *board*[baris, kolom]. Pada keseluruhan *flow chart* pada sub-bagian 4.1.2 akan dapat dilihat penggunaan *board*[i, j], i dan j tersebut menandakan bahwa keseluruhan proses manipulasi *board* berada di dalam perulangan bersarang (*nested loop*) dengan *count* i dan j.

4.1.1. Basis Pengetahuan Algoritma Minimax pada Permainan Reversi

Basis Pengetahuan (*knowledge based*) pada *artificial intelligence* adalah sekumpulan informasi yang berupa fakta-fakta, teori-teori, dan perhitungan yang akan membantu untuk mengambil kesimpulan menemukan langkah terbaik dari setiap langkah yang diberikan.

Adapun *knowledge based* pada permainan reversi ini adalah :

1. Diketahui 4 posisi *corner* (sudut) pada papan direpresentasikan dengan *board* [0, 0], [0, 7], [7, 0], [7, 7]. Jumlah posisi *corner* ini akan dikali dengan poin 5 sebagai nilai evaluasi terhadap *current board*.

0, 0							0, 7
7, 0							7, 7

Gambar 4.1. Representasi posisi *corners* pada array (*board*)

2. Diketahui 24 posisi *c-squares* pada papan direpresentasikan dengan *board* [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 0], [2, 0], [3, 0], [4, 0], [5, 0], [6, 0], [7, 1], [7, 2], [7, 3], [7, 4], [7, 5], [7, 6], [1, 7], [2, 7], [3, 7], [4, 7], [5, 7], [6, 7].

	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	
1, 0							1, 7
2, 0							2, 7
3, 0							3, 7
4, 0							4, 7
5, 0							5, 7
6, 0							6, 7
	7, 1	7, 2	7, 3	7, 4	7, 5	7, 6	

Gambar 4.2. Representasi posisi *c-squares* pada *array (board)*

3. Diketahui 12 posisi *x-squares* pada papan direpresentasikan dengan *board* [1, 1], [2, 2], [3, 3], [4, 4], [5, 5], [6, 6], [1, 6], [2, 5], [3, 4], [4, 3], [5, 2], [6, 1].

	1, 1					1, 6	
		2, 2			2, 5		
			3, 3	3, 4			
			4, 3	4, 4			
		5, 2			5, 5		
	6, 1					6, 6	

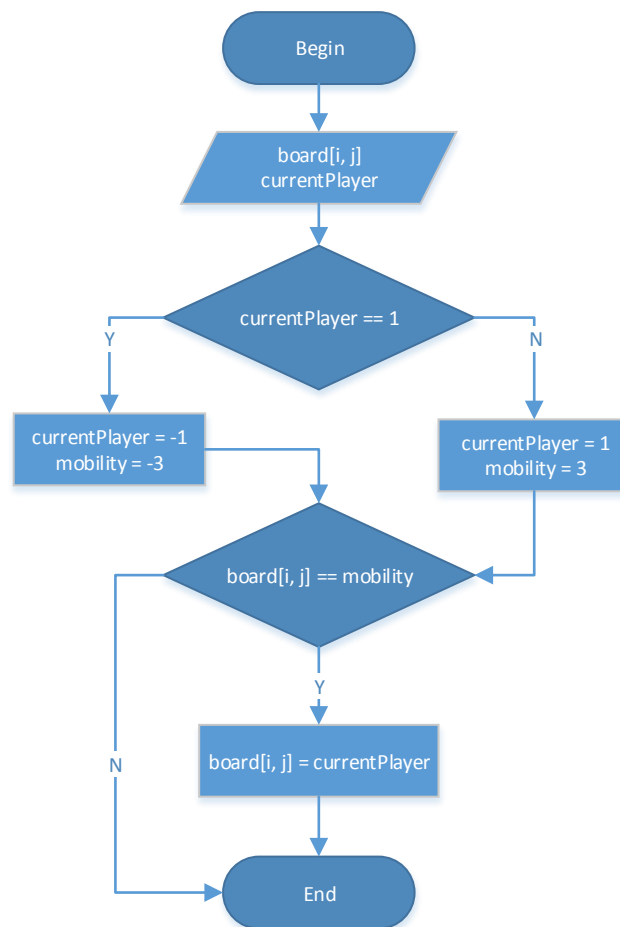
Gambar 4.3. Representasi posisi *x-squares* pada *array (board)*

4. Jika posisi-posisi *c-Square* dan *x-Square* ditempati bidak (koin) dan tersambung dengan posisi *corner* maka *c-Square* dan *x-Square* ini dapat dikatakan sebagai bidak stabil.
5. Ditetapkan bahwa representasi bidak hitam pada *board* dengan nilai 1, bidak putih dengan nilai -1, dan kosong dengan 0. Sedangkan untuk *mobility* bidak hitam dengan nilai 3, *mobility* bidak putih dengan -3.
6. Inisialisasi pada awal permainan yaitu *board* [3, 3] dan *board* [4, 4] bernilai -1 (bidak putih) sedangkan *board* [3, 4] dan *board* [4, 3] bernilai 1 (bidak hitam).

4.1.2. Flow Chart Algoritma pada Permainan Reversi

4.1.2.1. Flow Chart Algoritma Simpan Bidak

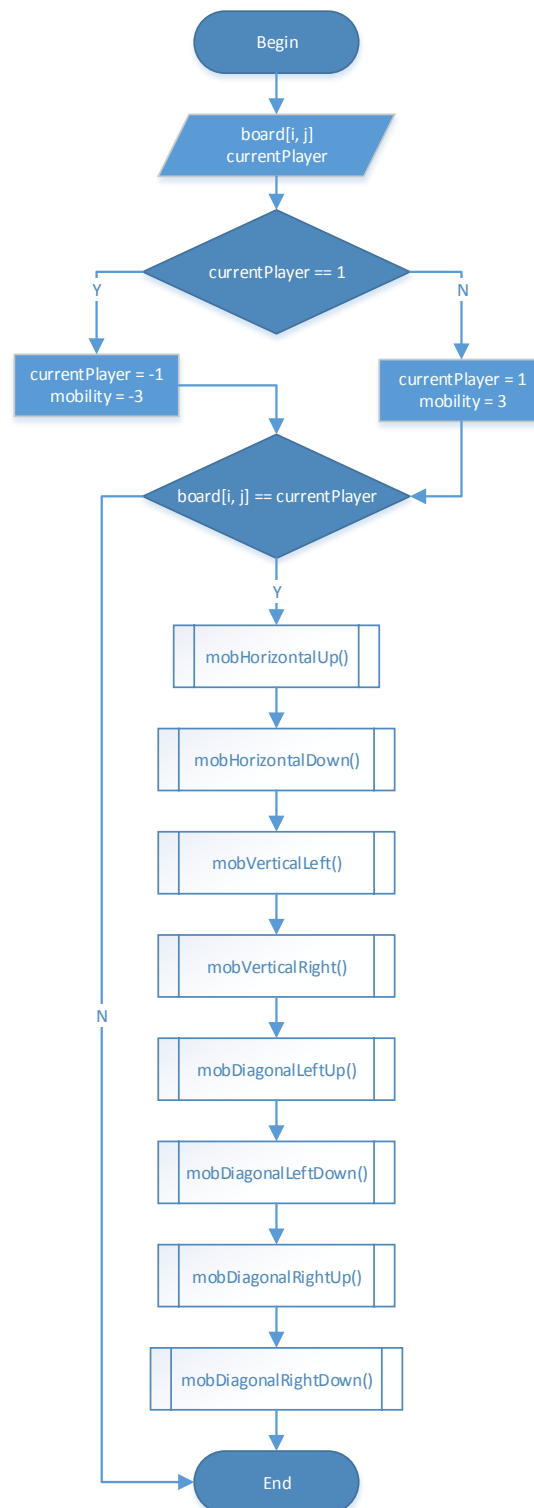
Algoritma Simpan Bidak merupakan algoritma untuk menentukan apakah pada papan akan disimpan Bidak (hitam atau putih) atau tidak. Simpan bidak ini ditentukan oleh papan (*board*[i, j]) yang menempati suatu *mobility*.



Gambar 4.4. Flow Chart Algoritma Simpan Bidak

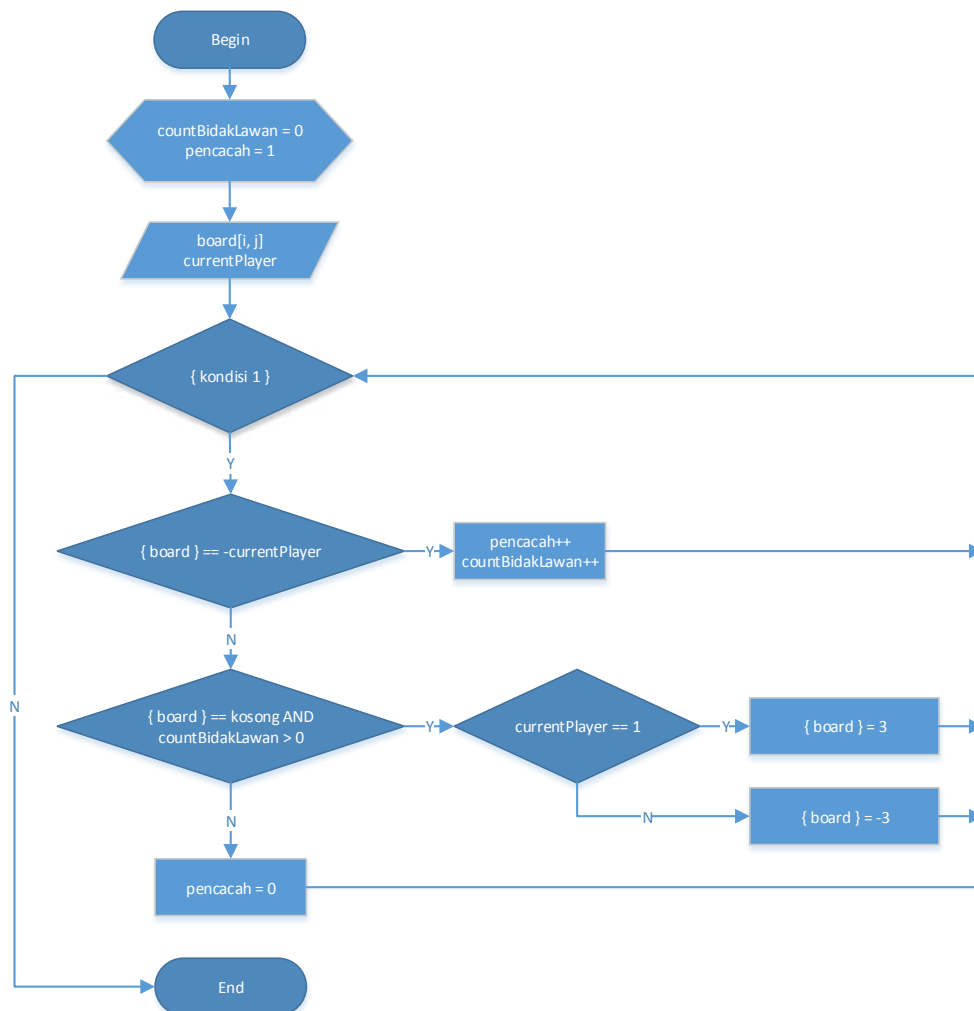
4.1.2.2. Flow Chart Algoritma Pemberian *Mobility* pada Setiap Langkah

Sebagaimana diketahui di awal bahwa *mobility* adalah tempat-tempat atau kotak-kotak yang dapat dijadikan langkah selanjutnya (penyimpanan bidak), dalam hal rancangan algoritmanya, pertama-tama penulis melakukan pencarian terhadap semua kotak atau tempat yang ditempati oleh bidak *current play*, kemudian dilakukan pencarian *mobility* secara horizontal ke atas, horizontal ke bawah, vertikal ke kiri, vertikal ke kanan, diagonal kiri atas, diagonal kiri bawah, diagonal kanan atas, dan diagonal kanan bawah.



Gambar 4.5. Flow Chart Algoritma Pemberian *Mobility* pada Setiap Langkah

Pada *flow chart* gambar 4.5 dapat dilihat bahwa ada beberapa prosedur yang mana prosedur tersebut memiliki tahapan proses yang hampir sama. Hanya dibedakan dari kondisi dan posisi pencarian ke berbagai arah dari posisi *current*, ada yang pencarian secara horizontal ke atas, horizontal ke bawah, vertikal ke kiri, vertikal ke kanan, diagonal kiri atas, diagonal kiri bawah, diagonal kanan atas, dan diagonal kanan bawah. Jika digambarkan dengan *flow chart* prosedur tersebut seperti berikut :



Gambar 4.6. Flow Chart Prosedur mobility

Pada *flow chart* gambar 4.6 tersebut yang membedakan kedelapan prosedur *mobility* pada gambar 4.5 adalah bagian { kondisi 1 } dan { board } saja. Penjelasannya sebagai berikut :

{ kondisi 1 } untuk :

- a. Horizontal ke atas : $(pencacah \neq 0) \text{ AND } (i \geq 0) \text{ AND } (i - pencacah \geq 0)$
- b. Horizontal ke bawah : $(pencacah \neq 0) \text{ AND } (i \leq 7) \text{ AND } (i + pencacah \leq 7)$
- c. Vertikal ke kiri : $(pencacah \neq 0) \text{ AND } (j \geq 0) \text{ AND } (j - pencacah \geq 0)$
- d. Vertikal ke kanan : $(pencacah \neq 0) \text{ AND } (j \leq 7) \text{ AND } (j - pencacah \leq 7)$
- e. Diagonal kiri atas : $(pencacah \neq 0) \text{ AND } (i \geq 0) \text{ AND } (j \geq 0) \text{ AND } (i - pencacah \geq 0) \text{ AND } (j - pencacah \geq 0)$
- f. Diagonal kiri bawah : $(pencacah \neq 0) \text{ AND } (i \leq 7) \text{ AND } (j \geq 0) \text{ AND } (i + pencacah \leq 7) \text{ AND } (j - pencacah \geq 0)$
- g. Diagonal kanan atas : $(pencacah \neq 0) \text{ AND } (i \geq 0) \text{ AND } (j \leq 7) \text{ AND } (i - pencacah \geq 0) \text{ AND } (j + pencacah \leq 7)$
- h. Diagonal kanan bawah : $(pencacah \neq 0) \text{ AND } (i \leq 7) \text{ AND } (j \leq 7) \text{ AND } (i + pencacah \leq 7) \text{ AND } (j + pencacah \leq 7)$

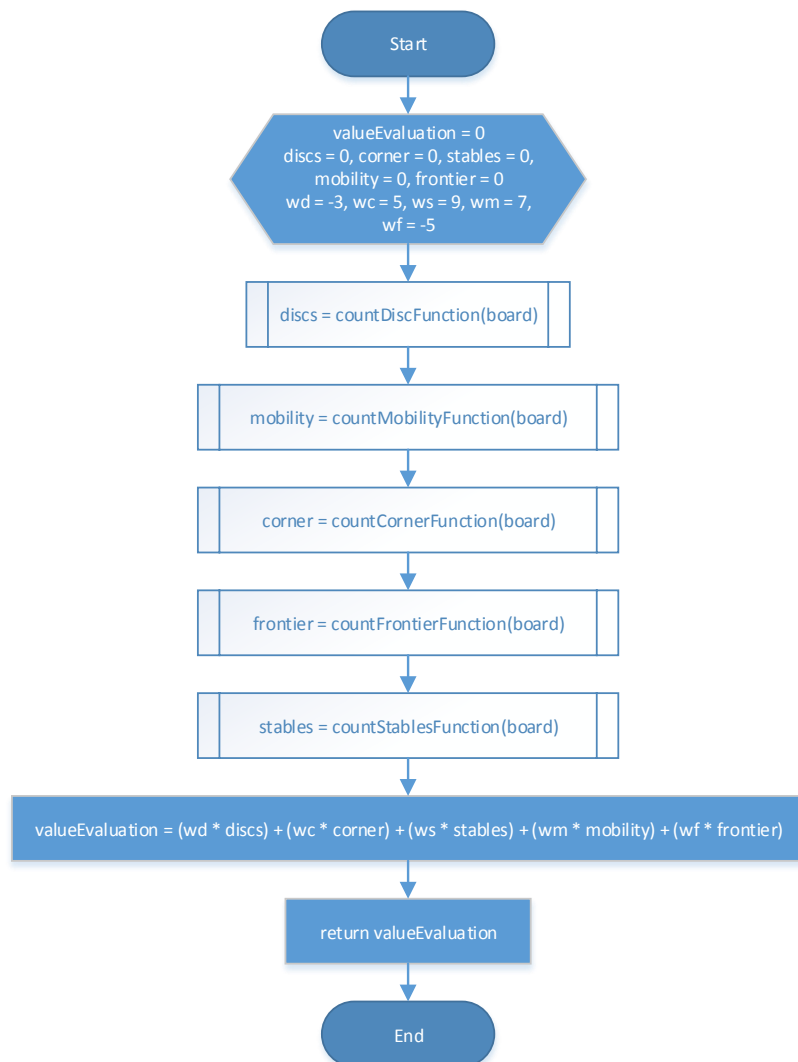
{ board } untuk :

- a. Horizontal ke atas : $board[i - pencacah, j]$
- b. Horizontal ke bawah : $board[i + pencacah, j]$

- c. Vertikal ke kiri : $board[i, j - pencacah]$
- d. Vertikal ke kanan : $board[i, j + pencacah]$
- e. Diagonal kiri atas : $board[i - pencacah, j - pencacah]$
- f. Diagonal kiri bawah : $board[i + pencacah, j - pencacah]$
- g. Diagonal kanan atas : $board[i - pencacah, j + pencacah]$
- h. Diagonal kanan bawah : $board[i + pencacah, j + pencacah]$

4.1.2.3. **Flow Chart Algoritma Pemberian Nilai Evaluasi pada Setiap Langkah**

Berikut ini merupakan *flow chart* algoritma untuk pemberian nilai evaluasi pada setiap langkah atau pada setiap kondisi papan yang terbentuk. Hal ini diperlukan karena komputer (AI) akan menemukan kondisi dari langkah terbaik dari nilai evaluasi yang dihasilkan. Algoritma ini merupakan sebuah *function* dengan parameter kondisi papan terakhir dan *function* ini akan mengembalikan sebuah nilai evaluasi bertipe *integer*.

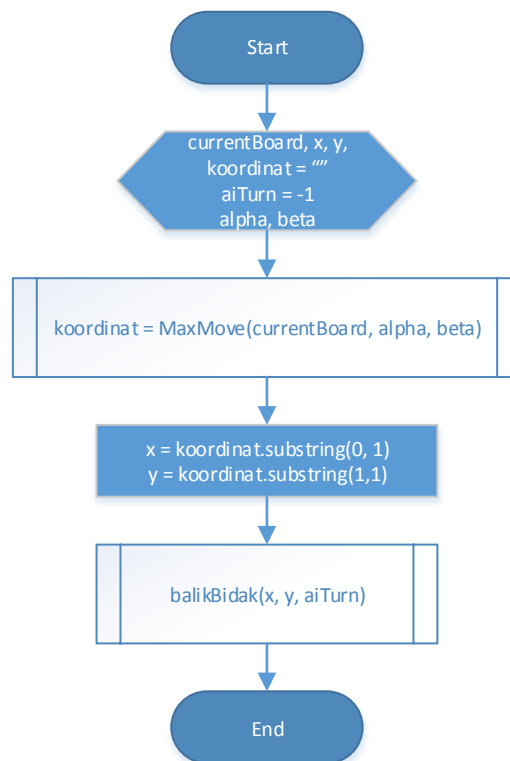


Gambar 4.7. Flow Chart Algoritma Pemberian Nilai Evaluasi

Pada Gambar 4.7 terdapat beberapa *function* untuk melakukan perhitungan nilai pada papan dengan menghitung jumlah bidak keseluruhan, jumlah *mobility* yang terbentuk, jumlah bidak pojok, dan jumlah bidak stabil yang kemudian akan dilakukan perhitungan dengan sebuah rumus yang telah ditetapkan sebelumnya.

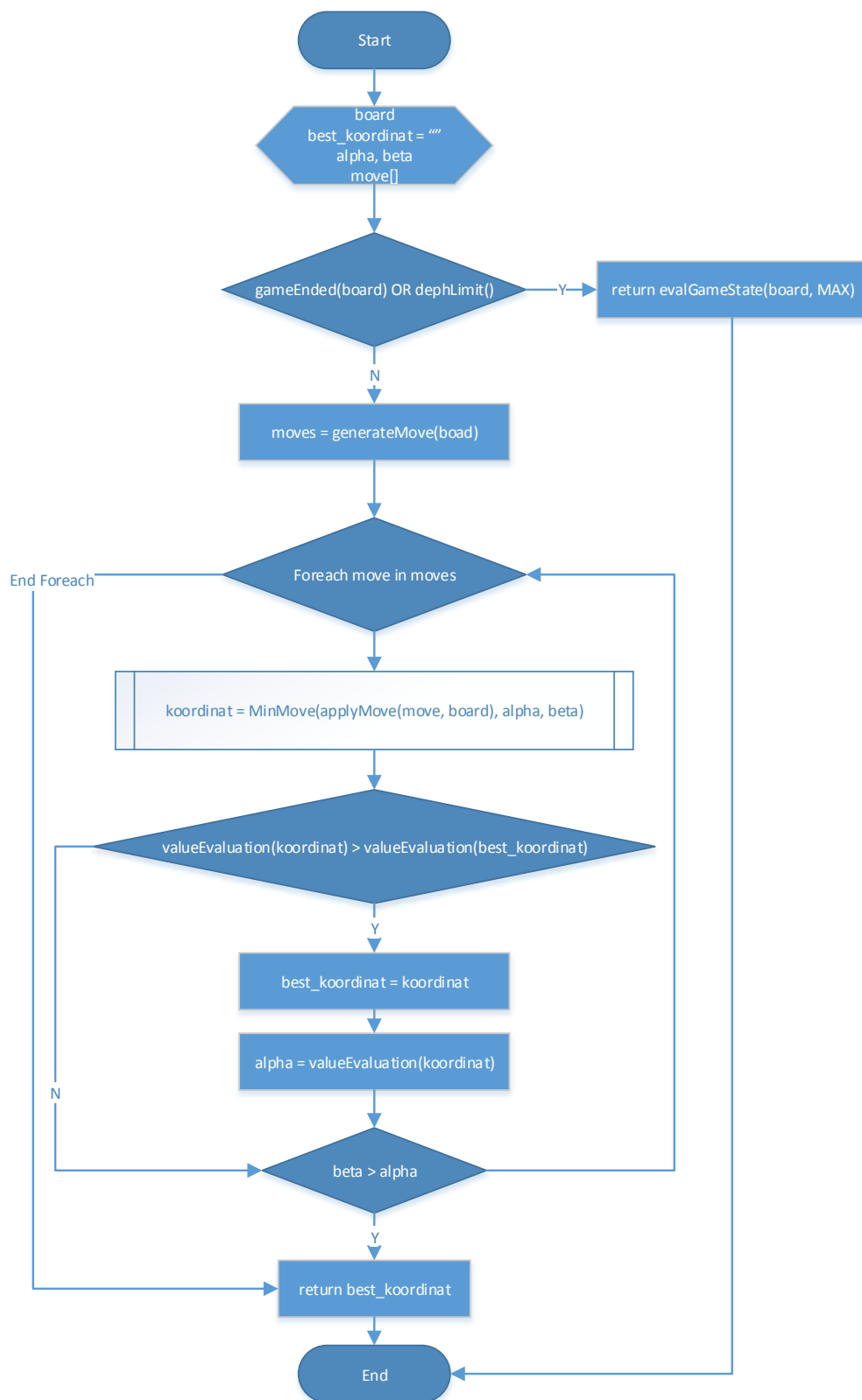
4.1.2.4. Flow Chart Algoritma Pencarian Langkah Terbaik

Pada *flow chart* algoritma pencarian langkah terbaik ini penulis akan menggunakan penerapan algoritma minimax sehingga terdapat tiga fungsi yang harus dibuat diantaranya fungsi **MinMax** itu sendiri, kemudian fungsi **MaxMove** dan **MinMove**. Ketiga fungsi ini mengembalikan nilai berupa suatu koordinat (*mobility*) yang terbaik dari beberapa *mobility* yang terbentuk berdasarkan posisi *current board*.



Gambar 4.8. Flow Chart Algoritma Pencarian Langkah Terbaik (MinMax)

Pada *flow chart* algoritma diatas terdapat pemanggilan terhadap suatu fungsi bernama **MaxMove** yang mana fungsi ini akan mengembalikan nilai berupa string. Berikut *flow chart* fungsi **MaxMove** dapat dilihat pada Gambar 4.9 :

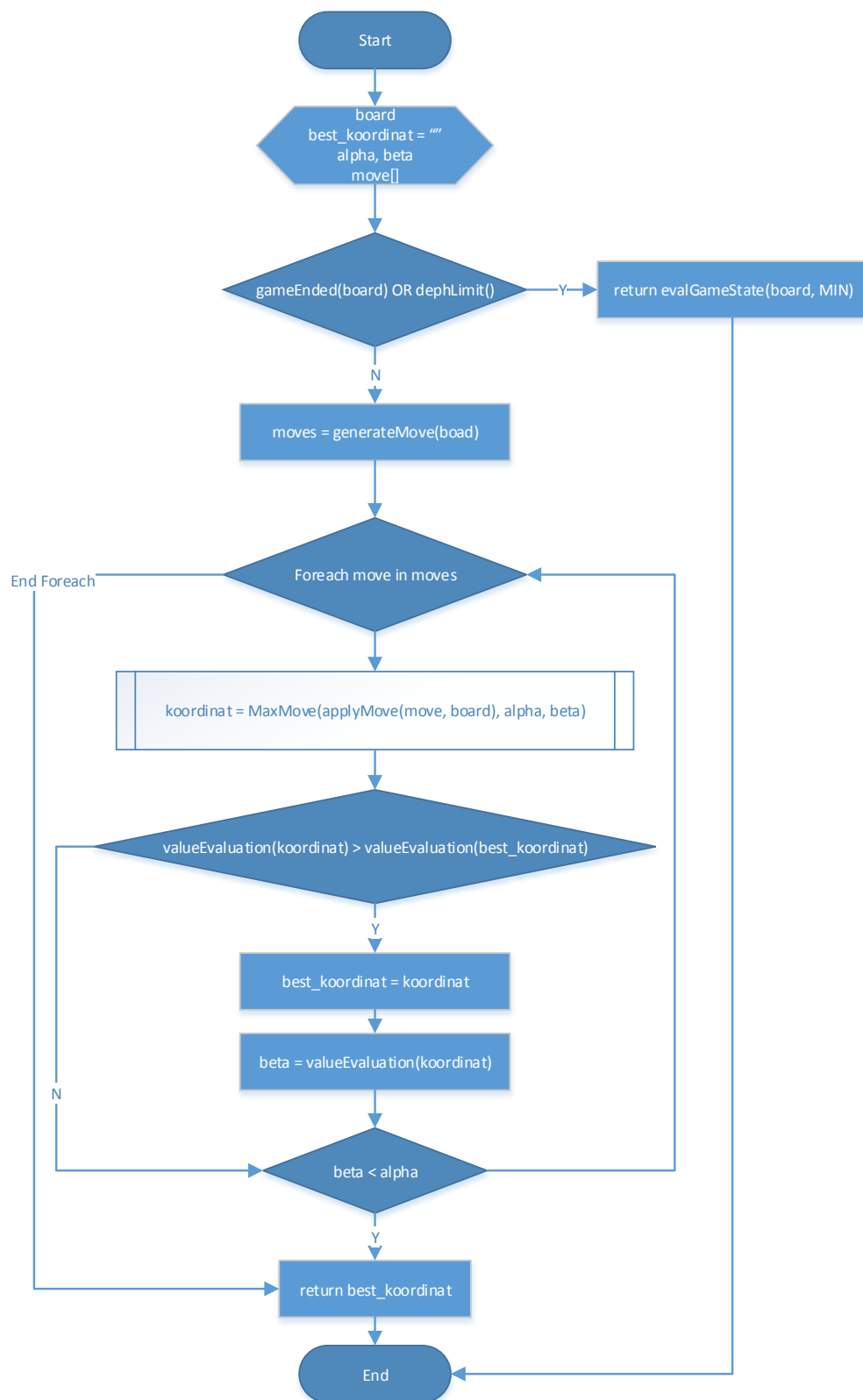


Gambar 4.9. Flow Chart Algoritma Pencarian Langkah Terbaik (MaxMove)

Secara garis besar *flow chart* algoritma fungsi **MaxMove** diatas yaitu pertama melakukan pemeriksaan terhadap langkah, apakah merupakan *leaf node* atau merupakan batas kedalaman yang telah ditentukan karena adanya *level* permainan (baik *beginner*, *intermediate*, atau *expert*). Kemudian jika bukan merupakan kedua hal tersebut maka akan dilakukan iterasi langkah berikutnya yang kemudian pada setiap langkah yang terbentuk akan dilakukan pemanggilan fungsi **MinMove** dikarenakan langkah **MinMove** tersebut merupakan representasi langkah *human player* dalam bermain dan harus dicari nilai semimum mungkin. Sedangkan untuk **MaxMove** akan dicari nilai semaksimal mungkin karena merupakan representasi langkah *computer player*.

Pada fungsi **MinMove** akan dilakukan proses yang hampir sama dengan fungsi **MaxMove** yaitu pemeriksaan kondisi papan dan kedalaman kemudian akan dilakukan iterasi langkah berikutnya yang mana untuk setiap langkah yang terbentuk akan dilakukan pemanggilan kembali fungsi **MaxMove**, dan begitu seterusnya sampai menemukan kondisi akhir.

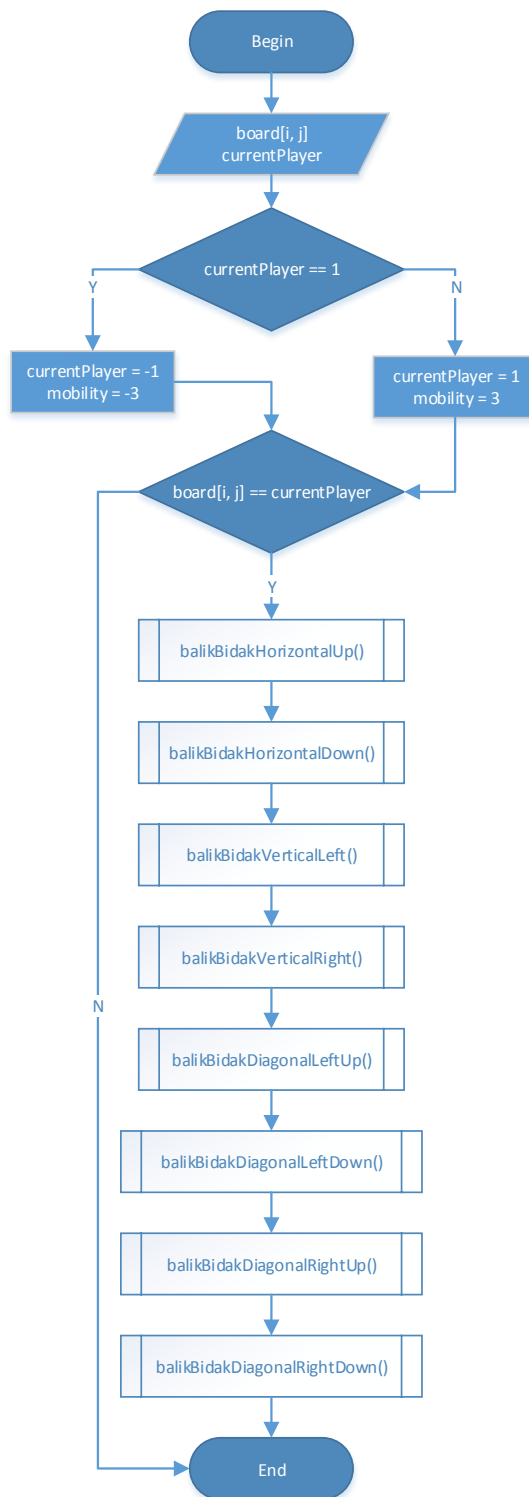
Berikut *flow chart* algoritma fungsi **MinMove** dapat dilihat pada Gambar 4.10 :



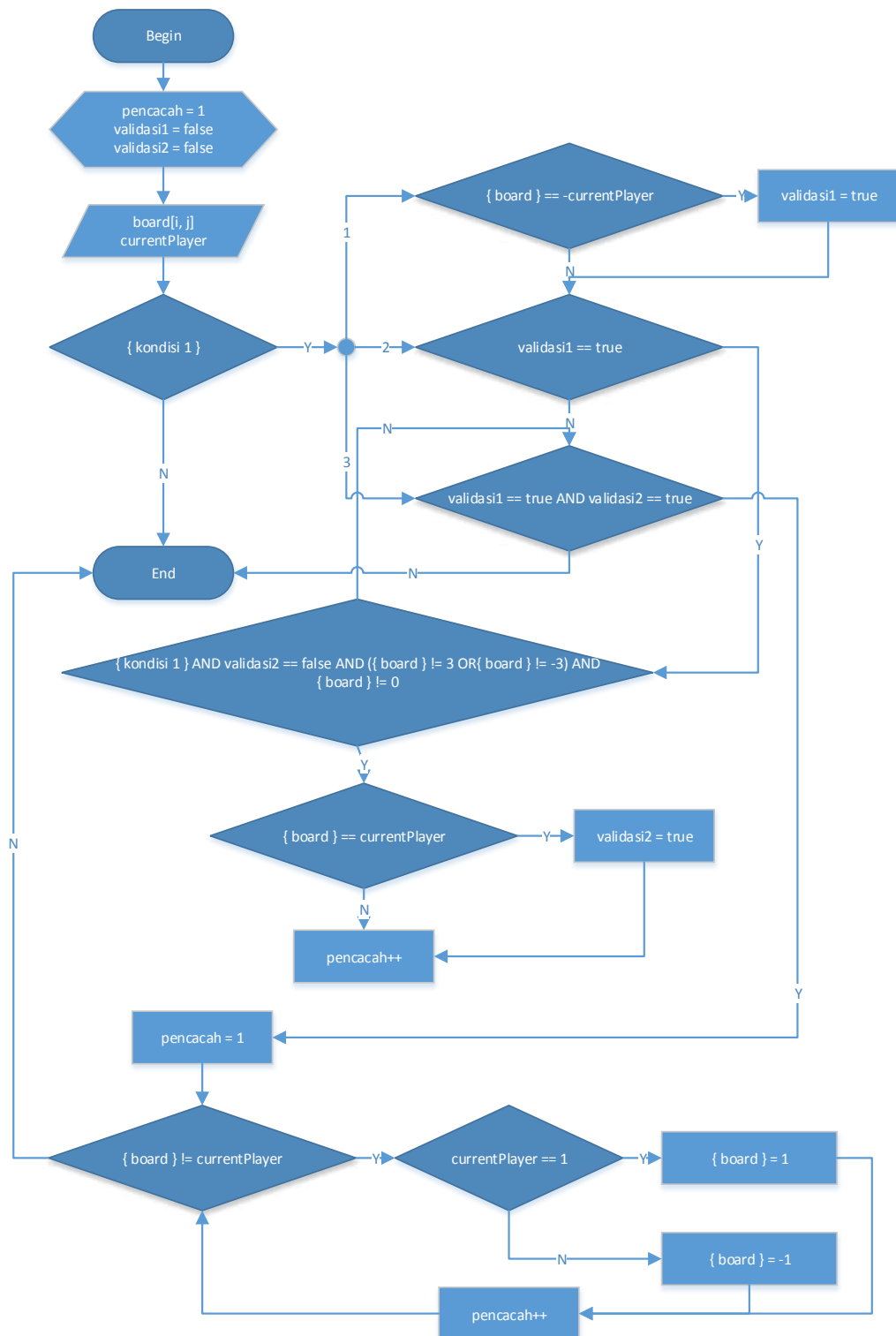
Gambar 4.10. Flow Chart Algoritma Pencarian Langkah Terbaik (MinMove)

4.1.2.5. *Flow Chart* Algoritma Balik Bidak Lawan

Algoritma balik bidak lawan merupakan algoritma untuk membalik (mengganti) semua bidak lawan yang terlewati dengan bidak *current play* baik ke arah horizontal atas, horizontal bawah, vertikal kiri, vertikal kanan, diagonal kiri atas, diagonal kiri bawah, diagonal kanan atas, dan diagonal kanan bawah. Inti dari algoritma ini yaitu adanya pemanggilan prosedur untuk membalik bidak secara horizontal, vertikal, dan diagonal ke berbagai arah menyeluruh dengan total terdapat delapan prosedur yaitu **horizontalUp()**, **horizontalDown()**, **verticalLeft()**, **verticalRight()**, **diagonalLeftUp()**, **diagonalLeftDown()**, **diagonalRightUp()**, dan **diagonalRightDown()**. Kedelapan prosedur tersebut pada intinya sama, yang membedakan hanya arah penelusurannya saja, untuk lebih jelas dapat dilihat pada Gambar 4.12 *Flow Chart* balik bidak dengan pemanggilan prosedurnya pada Gambar 4.11.



Gambar 4.11. Flow Chart Algoritma Balik Bidak Lawan



Gambar 4.12. Flow Chart Prosedur balikBidak

Pada flow chart gambar tersebut yang membedakan kedelapan prosedur *balikBidak* pada gambar adalah bagian { kondisi 1 } dan { board }. Penjelasannya sebagai berikut :

{ kondisi 1 } untuk :

- a. Horizontal ke atas : $(i - \text{pencacah}) \geq 0$
- b. Horizontal ke bawah : $(i + \text{pencacah}) \leq 7$
- c. Vertikal ke kiri : $(j - \text{pencacah}) \geq 0$
- d. Vertikal ke kanan : $(j + \text{pencacah}) \leq 7$
- e. Diagonal kiri atas : $(i - \text{pencacah}) \geq 0 \text{ AND } (j - \text{pencacah}) \geq 0$
- f. Diagonal kiri bawah : $(i + \text{pencacah}) \leq 7 \text{ AND } (j - \text{pencacah}) \geq 0$
- g. Diagonal kanan atas : $(i - \text{pencacah}) \geq 0 \text{ AND } (j + \text{pencacah}) \leq 7$
- h. Diagonal kanan bawah : $(i + \text{pencacah}) \leq 7 \text{ AND } (j + \text{pencacah}) \leq 7$

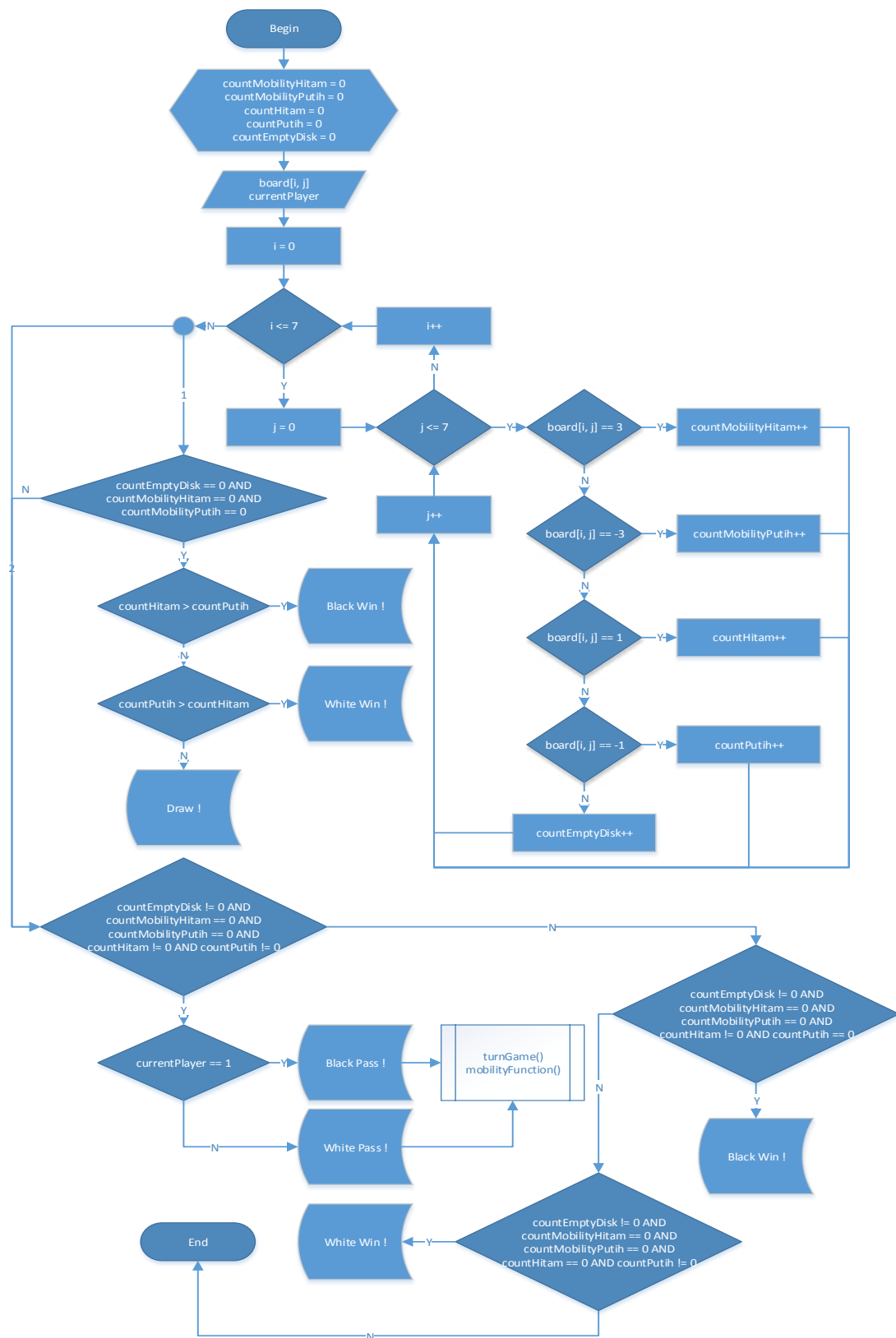
{ board } untuk :

- a. Horizontal ke atas : $\text{board}[i - \text{pencacah}, j]$
- b. Horizontal ke bawah : $\text{board}[i + \text{pencacah}, j]$
- c. Vertikal ke kiri : $\text{board}[i, j - \text{pencacah}]$
- d. Vertikal ke kanan : $\text{board}[i, j + \text{pencacah}]$
- e. Diagonal kiri atas : $\text{board}[i - \text{pencacah}, j - \text{pencacah}]$
- f. Diagonal kiri bawah : $\text{board}[i + \text{pencacah}, j - \text{pencacah}]$
- g. Diagonal kanan atas : $\text{board}[i - \text{pencacah}, j + \text{pencacah}]$
- h. Diagonal kanan bawah : $\text{board}[i + \text{pencacah}, j + \text{pencacah}]$

4.1.2.6. *Flow Chart Algoritma Penentuan Pemenang*

Adapun flow chart untuk algoritma penentuan pemenang yaitu algoritma untuk melakukan pemeriksaan terhadap bidak mana yang memenangkan permainan. Algoritma ini memeriksa setiap langkah yang dilakukan baik human player maupun computer player, algoritma ini juga yang mengatur jalannya permainan, misalnya apakah masih terdapat langkah yang mungkin untuk human player atau sebaliknya. Jika kedua pemain sudah tidak bisa melangkah dalam arti seluruh papan sudah dipenuhi koin atau sudah tidak ada lagi langkah yang mungkin untuk kedua pemain, maka permainan berakhir dan dilakukan perhitungan terhadap bidak atau koin yang dihasilkan.

Berikut flow chart algoritma penentuan pemenang dapat dilihat pada Gambar 4.13 :



Gambar 4.13. Flow Chart Algoritma Penentuan Pemenang

4.2. Rancangan Antarmuka Pengguna

Berikut ini merupakan rancangan antarmuka (UI) untuk aplikasi permainan reversi ini, diantaranya yaitu rancangan antarmuka masukan dan rancangan antarmuka keluaran.

4.2.1. Rancangan Antarmuka Masukan

Rancangan antarmuka atau tampilan aplikasi pada saat awal *running* terdapat beberapa *button* yang merupakan menu utama aplikasi.



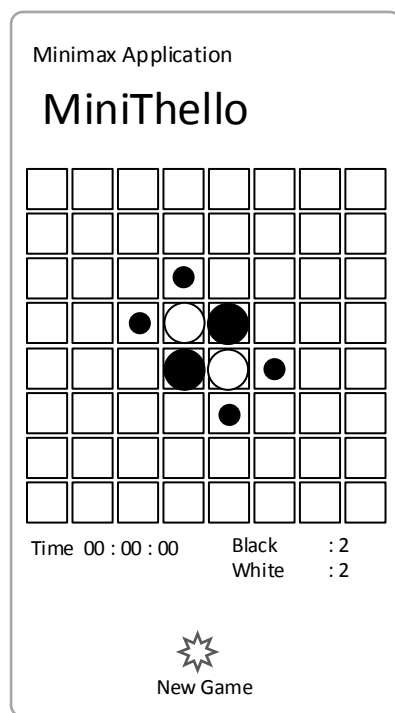
Gambar 4.14. Tampilan Antarmuka Awal Aplikasi

Terdapat 4 *button* yang ada pada tampilan menu awal aplikasi ini, yaitu :

- a) Play merupakan *button* untuk memulai permainan, jika mengalami perubahan status pada *button* tersebut, maka akan menampilkan tampilan antarmuka

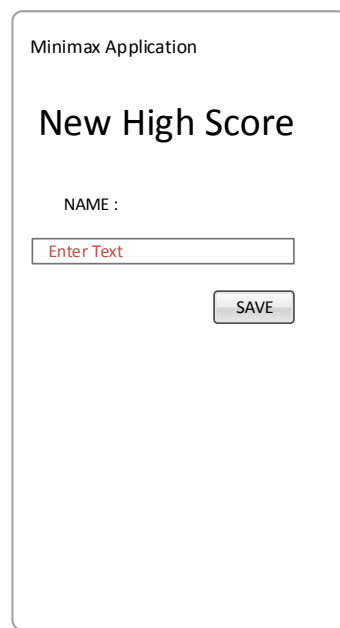
papan permainan seperti pada gambar 4.15. Permainan dan *timing* akan dimulai ketika *human player* meletakkan bidak pada papan yang terdapat *mobility* dan berakhir jika papan permainan telah ditentukan pemenangnya atau pada tengah-tengah permainan menekan *icon* New Game pada *application bar*.

- b) About merupakan *button* untuk mengetahui tentang aplikasi permainan Reversi / Othello ini, untuk mengetahui versi dan *developer* aplikasi.
- c) High Scores merupakan *button* untuk mengetahui top 15 besar peringkat waktu (*timing*) yang diperoleh.
- d) Help merupakan *button* bantuan tentang penggunaan aplikasi permainan ini.



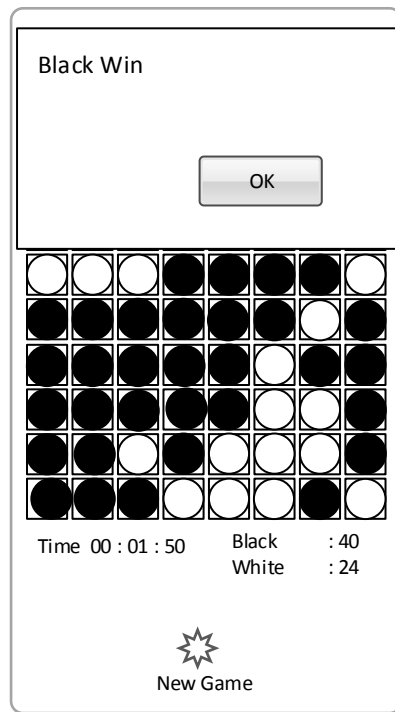
Gambar 4.15. Tampilan Antarmuka Papan Permainan

Pada gambar 4.15 diatas merupakan tampilan antarmuka papan permainan reversi, dapat dilihat ada komponen *label* yang merupakan *timing* dan *count* untuk bidak hitam dan putih. Permainan dimulai dan *timing* akan mulai berjalan jika *human player* meletakkan bidak warna hitam pada *mobility* warna hitam (*mobility* direpresenasikan dengan bidak yg lebih kecil) kemudian berakhir jika pemenang sudah ditentukan atau dapat dengan menekan *icon* New Player pada *application bar*. Jika permainan berakhir dan dimenangkan oleh *human player* (*black win*), maka jika *timing* yang dihasilkan merupakan *timing* yang paling cepat dibandingkan dengan *timing players* sebelumnya, maka dapat menginputkan nama untuk masuk pada daftar top 15 *high scores*.

The image shows a mobile application interface titled "Minimax Application". Below the title is a heading "New High Score". Underneath the heading is a label "NAME :". Below the label is a text input field with the placeholder text "Enter Text" in red. To the right of the input field is a button labeled "SAVE".

Gambar 4.16. Tampilan Antarmuka *Input New High Score*

4.2.2. Rancangan Antarmuka Keluaran



Gambar 4.17. Tampilan Antarmuka Pemberitahuan Pemenang

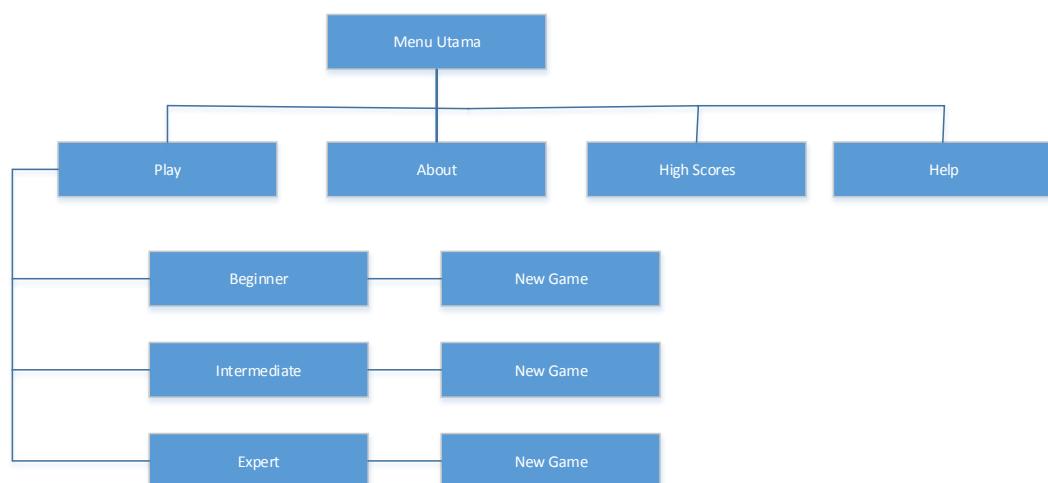
Pada gambar 4.17 merupakan tampilan antarmuka keluaran berupa pemberitahuan pemenang (menang, kalah, atau seri).

Minimax Application		
High Scores		
NO	NAME	TIME
1	NAME1	00 : 00 : 05
2	NAME2	00 : 00 : 10
3	NAME3	00 : 00 : 15
4	NAME4	00 : 00 : 20
5	NAME5	00 : 00 : 25
6	NAME6	00 : 00 : 30
7	NAME7	00 : 00 : 35
8	NAME8	00 : 00 : 40
9	NAME9	00 : 00 : 45
10	NAME10	00 : 00 : 50
11	NAME11	00 : 00 : 55
12	NAME12	00 : 01 : 00
13	NAME13	00 : 02 : 00
14	NAME14	00 : 03 : 00
15	NAME15	00 : 04 : 00

Gambar 4.18. Tampilan Antarmuka Informasi *High Scores*

4.3. Struktur Menu

Berikut ini adalah struktur menu pada aplikasi permainan Reversi :



Gambar 4.19. Struktur Menu Program Aplikasi

4.4. Perancangan Basis Data

Oleh karena pada aplikasi ini hanya akan ada satu tabel pada penyimpanan data, jadi penulis tidak akan menjelaskan konsep perancangan basis data secara normal yang mencakup normalisasi, ERD, dsb. Tetapi hanya dijelaskan mengenai satu tabel yang mempunyai spesifikasi sebagai berikut :

Nama File : Score

Media : Windows Phone Isolated Storage Settings

Isi : Data Score User

Primary Key : Id

Struktur File :

No	Nama Field	Jenis	Lebar	Desimal	Keterangan
1	Id	Numeric	4	0	Primary Key
2	Name	Character	20	0	
3	Time	Character	12	0	Timing
4	Second	Numeric	6	0	Time in Seconds (not show in application)

BAB V

IMPLEMENTASI DAN PENGUJIAN APLIKASI

5.1. Spesifikasi Kebutuhan Perangkat

Pada bab ini akan dilakukan implementasi dan pengujian terhadap *game* aplikasi sebagai terapan dari algoritma minimax yang telah dijelaskan sebelumnya. Berikut ini spesifikasi perangkat keras dan perangkat lunak yang dibutuhkan baik dari perangkat PC yang digunakan untuk pengembangan aplikasi, maupun dari perangkat *Smart Phone* yang dipakai *user*.

5.1.1. Spesifikasi Perangkat PC (*Personal Computer*)

1. 1.6 GHz Processor
2. 4 GB RAM
3. 10 GB (NTFS) *of avaiable of* Hard Disk *space*
4. 5400 RPM Hard Drive
5. DirectX 10 *video card running at* 1366 x 768
6. *Operating System* Windows 8 Pro 64-bit (x64) *client versions*

5.1.2. Spesifikasi Perangkat *Smart Phone*

1. Layar 4-point *multi-touch* WVGA dengan resolusi (480x800)
2. Processor Qualcomm Snapdragon™ S4
3. DirectX9 *rendering-capable* GPU
4. 512 MB *of* RAM

5. Accelerometer dengan kompas, ambient light sensor, proximity sensor, Assisted GPS, dan Gyroscope
6. 5-megapixel kamera dengan flash
7. *Six (6) dedicated hardware buttons – back, start, search, 2-stage camera, power/sleep dan Volume Up - Down*

5.2. Implementasi

5.2.1. Implementasi Antarmuka

Tabel 5.1 berikut ini merupakan implementasi dari setiap halaman aplikasi permainan Reversi :

Tabel 5.1 Implementasi Antarmuka

Sub Menu	Deskripsi	Nama File
Menu Utama	File program untuk menampilkan halaman utama	MainPage.xaml
Play	File program untuk menampilkan halaman pemilihan level sebelum bermain	LevelSelection.xaml
Beginner	File program untuk memulai permainan dengan level mudah dengan memilih warna bidak terlebih dahulu	ChooseColor.xaml?depth=3

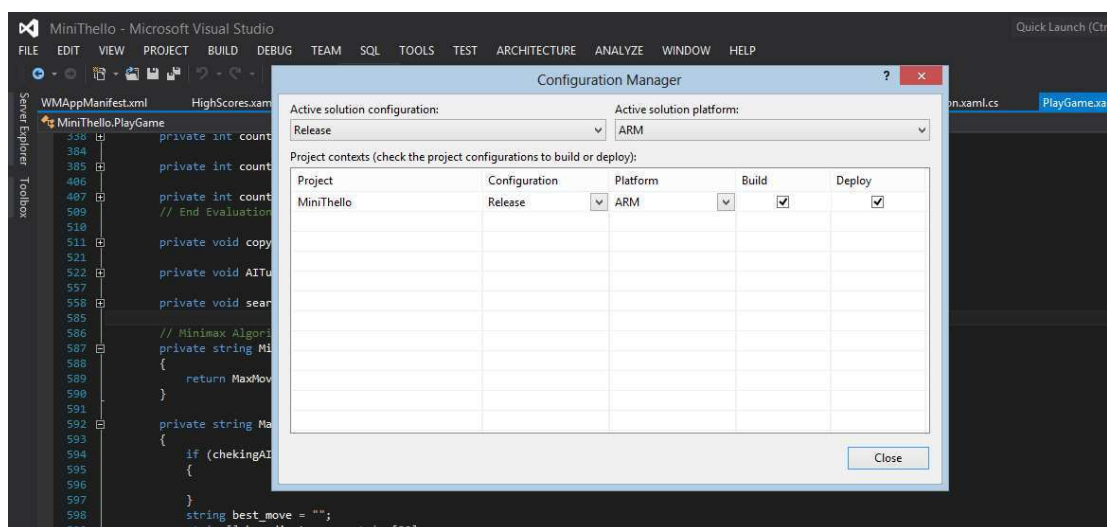
Intermediate	File program untuk memulai permainan dengan level sedang dengan memilih warna bidak terlebih dahulu	ChooseColor.xaml?depth=6
Expert	File program untuk memulai permainan dengan level sulit dengan memilih warna bidak terlebih dahulu	ChooseColor.xaml?depth=9
Black	File program untuk memulai permainan dengan pilihan bidak hitam	PlayGame.xaml
White	File program untuk memulai permainan dengan pilihan bidak putih	PlayGameWhite.xaml
About	File program untuk menampilkan info program dan developer	About2.xaml
High Scores	File program untuk menampilkan informasi score yang tersimpan	HighScores.xaml
Help	File program untuk menampilkan info penggunaan program	Help.xaml

New High Score	File program untuk save nama pemain	NewHighScore.xaml
----------------	-------------------------------------	-------------------

5.2.2. Implementasi Instalasi (*Release*) Program

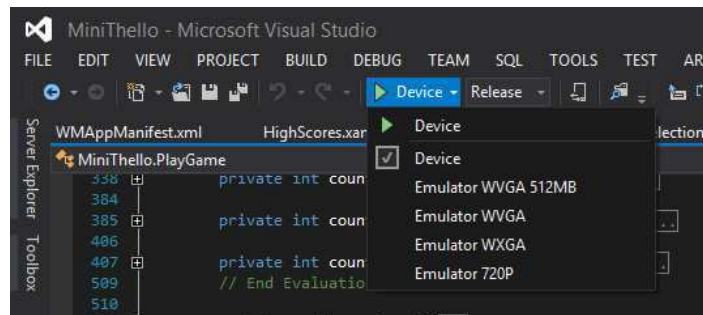
Untuk implementasi instalasi program, penulis menggunakan *Smart Phone* Nokia Lumia 620 dengan spesifikasi seperti yang telah dijelaskan diatas, kemudian sebelum proses instalasi atau *debuging*, *smart phone* tersebut sudah ter-unlock terlebih dahulu dan terdaftar di Microsoft sebagai *developer* Windows Phone. Setelah semua kode program berhasil ter-*debug* pada *emulator*, maka berikut implementasi instalasi program ke *Smart Phone* :

1. Ubah terlebih dahulu Platform pada Configuration Manager pada Microsoft Visual Studio (**BUILD -> Configuration Manager**) menjadi ARM (Advanced RISC Machines).



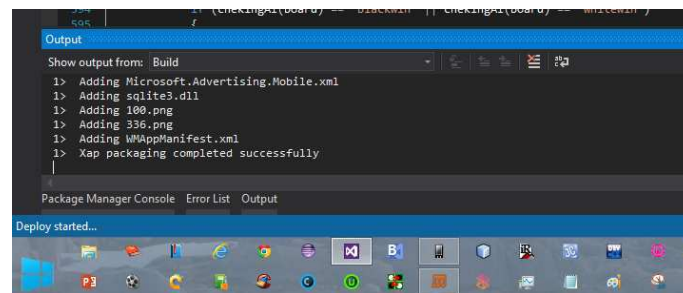
Gambar 5.1. Configurasi Platform

2. Kemudian ubah *debugging with Emulator* menjadi *Device*.



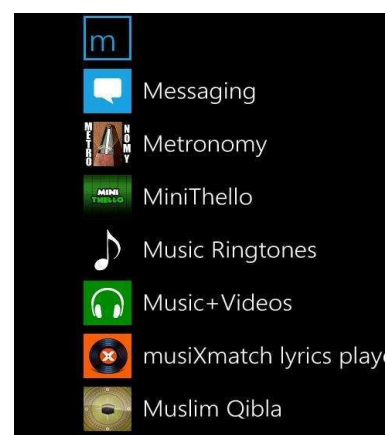
Gambar 5.2 Debugging Perangkat

3. Tunggu hingga proses instalasi pada Device selesai.



Gambar 5.3 Proses Instalasi

4. Aplikasi telah terinstal.



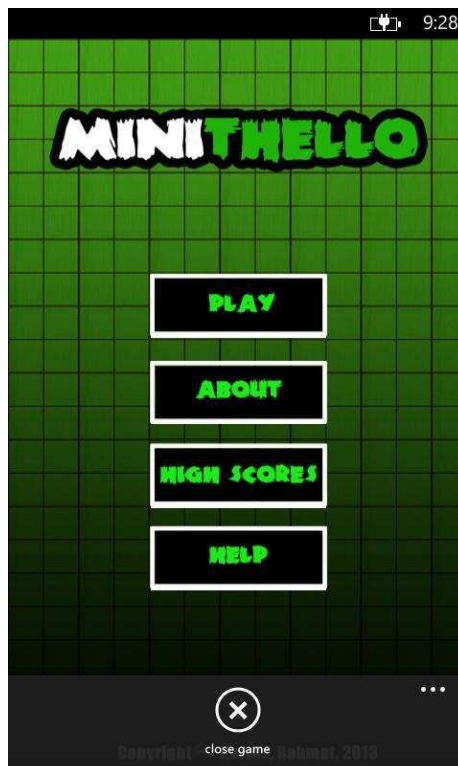
Gambar 5.4 Aplikasi terinstal

File instalasi yang tercipta setelah melakukan langkah diatas yaitu memiliki ekstensi *.xap dengan size “mentahan” berukuran 2.20 MB (2.312.472 bytes).

5.2.3. Implementasi Penggunaan Program

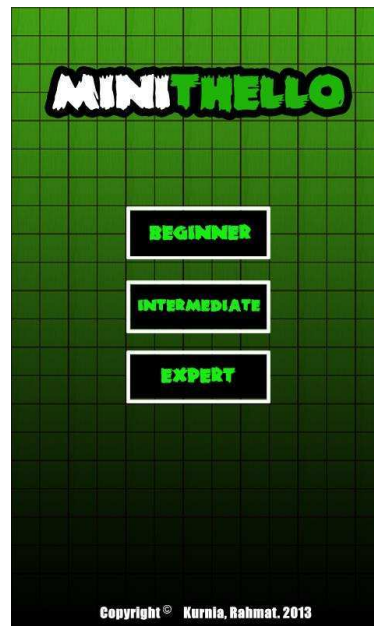
Setelah aplikasi terpasang maka user sudah dapat menggunakan atau memainkan aplikasi tersebut. Berikut implementasi penggunaan programnya :

1. Jalankan aplikasi Reversi (MiniThello) yang sudah ter-*install* kemudian akan muncul halaman utama sebagai berikut :



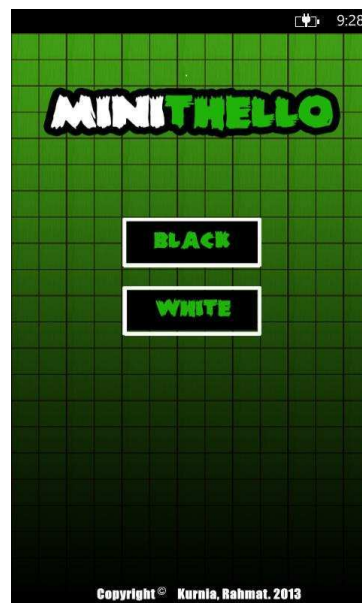
Gambar 5.5 Halaman Utama Program Aplikasi

2. Selanjutnya jika ingin langsung memainkan permainannya klik / tap *button* Play maka akan muncul halaman pemilihan *level* yang diinginkan.



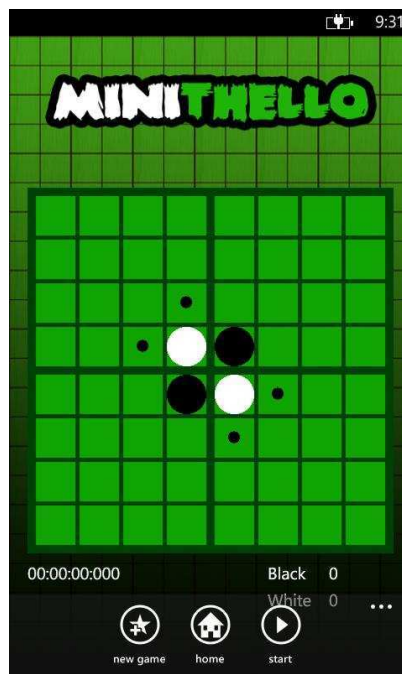
Gambar 5.6 Halaman Pemilihan *Level* Permainan

3. Jika telah memilih *level* yang diinginkan maka akan muncul pilihan untuk memilih warna bidak yang ingin digunakan yaitu hitam atau putih (dengan catatan warna bidak hitam akan bermain terlebih dahulu).



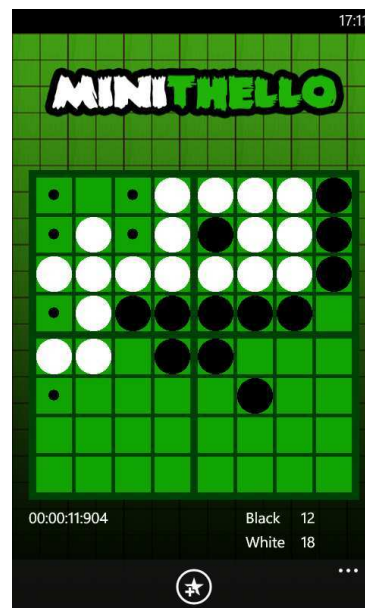
Gambar 5.7 Halaman Pemilihan warna bidak

4. Jika telah memilih warna bidak / koin yang akan digunakan maka akan muncul papan permainan berukuran 8x8. *Player* hanya dapat melangkah pada kotak yang memiliki *mobility* saja yang artinya hanya dapat melangkah pada kotak yang terdapat bulatan hitam atau putih yang kecil saja. Jika memilih warna bidak putih maka untuk memulai permainan harus diawali dengan *button* “start” pada Application Bar.



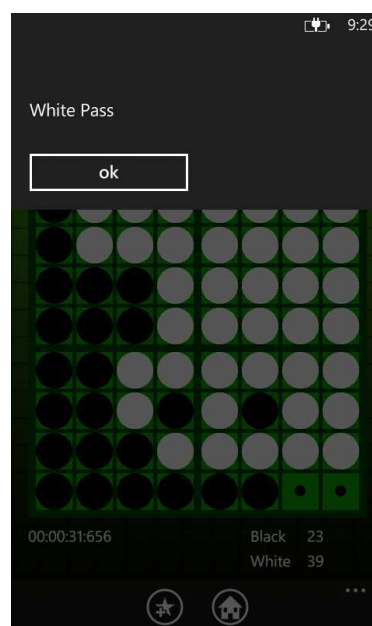
Gambar 5.8 Kondisi awal (*default*) papan main dengan giliran hitam

5. Setelah itu giliran komputer (AI) yang bermain dengan bidak (koin) warna putih.



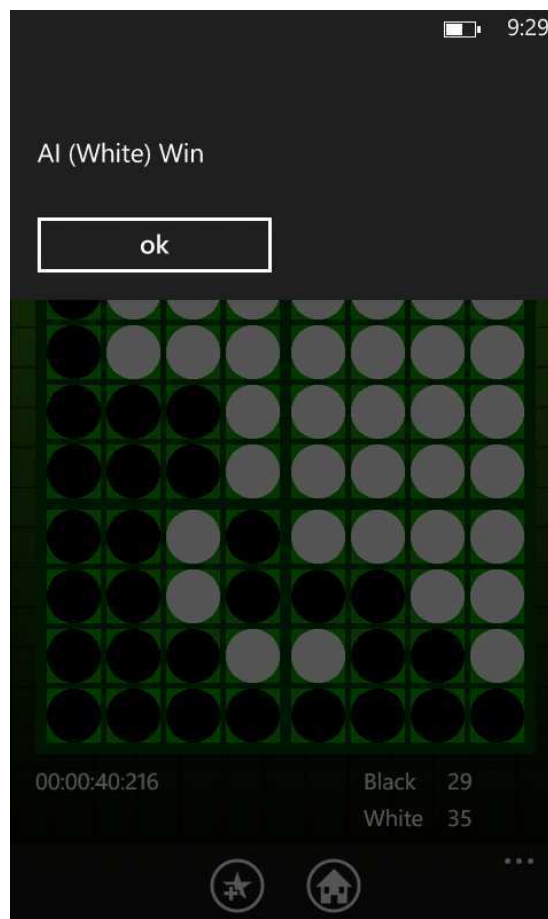
Gambar 5.9 Contoh Langkah Komputer (AI)

6. Jika didapatkan sudah tidak ada lagi *mobility* bagi pemain baik *user* atau komputer maka akan muncul peringatan “pass” yang berarti permainan dilanjutkan dengan masih di pemain sebelumnya (bisa hitam atau putih).



Gambar 5.10 Peringatan “pass” bagi pemain (*user* atau komputer)

7. Sampai pada kondisi dimana seluruh papan main sudah dipenuhi bidak (koin), maka permainan berakhir dan dilakukan perhitungan koin yang didapat dan menentukan pemenang dari permainan ini.



Gambar 5.11 Penentuan Pemenang

5.3. Pengujian Aplikasi

Pengujian merupakan tahapan penting dalam pembangunan aplikasi atau perangkat lunak. Pengujian dilakukan untuk mengetahui kelayakan suatu perangkat lunak sehingga perangkat lunak tersebut dapat berjalan sesuai dengan yang diharapkan dan mempunyai kualitas yang handal.

5.3.1. Pengujian *White Box*

Pengujian *White Box* merupakan pengujian yang berkaitan dengan struktur kontrol prosedural pada perangkat lunak atau aplikasi, dengan pengujian *White Box logical path* (jalur logika) perangkat lunak akan diuji dengan menyediakan *test case* yang akan mengerjakan sekumpulan kondisi atau perulangan secara spesifik. Adapun metode yang digunakan dalam pengujian *White Box* ini adalah metode *Basis Path*. Metode *Basis Path* mengizinkan pendesain kasus uji untuk membuat perkiraan logik yang kompleks dari desain prosedural dan menggunakan perkiraan ini untuk mendefinisikan aliran eksekusi. Dalam hal ini pengujian secara *White Box* dibatasi, penulis hanya menguji alur algoritma prosedur minimax saja (**MaxMove function** dan **MinMove function**).

Berikut *pseudocode* fungsi **MaxMove** :

```

1 : MaxMove (Integer[] board , Integer alpha, Integer beta) {
2 :   if (GameEnded(board) || DepthLimit()) {
3 :     return EvalGameState(board, MAX);
4 :   }
5 :   else {
6 :     best_koordinat ← {};
7 :     moves ← GenerateMoves(board);
8 :     ForEach move in moves {
9 :       koordinat ← MinMove(ApplyMove(move,board), alpha, beta);
10 :      if (valueEval(koordinat) > valueEval(best_koordinat)) {
11 :        best_koordinat ← koordinat;
12 :        alpha ← valueEval(koordinat);
13 :      }
14 :      if (beta > alpha)
15 :        return best_koordinat;
16 :    }
17 :    return best_koordinat;
18 :  }
19 : }
```

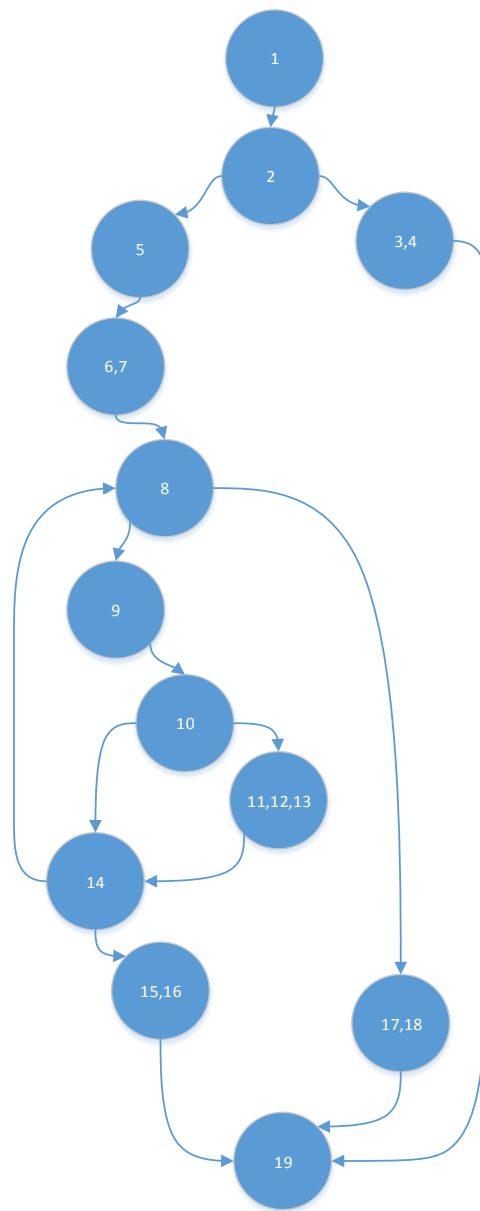
Sedangkan untuk *pseudocode* fungsi **MinMove** yaitu :

```

1 : MinMove (Integer[] board , Integer alpha, Integer beta) {
2 :   if (GameEnded(board) || DepthLimit()) {
3 :     return EvalGameState(board, MIN);
4 :   }
5 :   else {
6 :     best_koordinat ← {};
7 :     moves ← GenerateMoves(board);
8 :     ForEach move in moves {
9 :       koordinat ← MaxMove(ApplyMove(move,board), alpha, beta);
10 :      if (valueEval(koordinat) > valueEval(best_koordinat)) {
11 :        best_koordinat ← koordinat;
12 :        beta ← valueEval(koordinat);
13 :      }
14 :      if (beta < alpha)
15 :        return best_koordinat;
16 :    }
17 :    return best_koordinat;
18 :  }
19 : }

```

Pada kedua *pseudocode* tersebut tidak terlihat perbedaan yang signifikan, jadi jika kedua *pseudocode* tersebut diubah menjadi bentuk *flow graph*, akan seperti gambar berikut :



Gambar 5.12 Flow Graph fungsi MaxMove dan MinMove

Dari *flow graph* diatas dapat dihitung *cyclomatic complexity* yaitu mempunyai 5 *independent path*. Dihitung menggunakan rumus $V(G) = E - N + 2$ $= 16 - 13 + 2 = 5$, dimana E merupakan jumlah *edge* pada *flow graph* dan N merupakan jumlah *node* pada *flow graph*. Berikut kelima *independent path* pada *flow graph* diatas :

Path 1 : 1 – 2 – 3,4 – 19

Path 2 : 1 – 2 – 5 – 6,7 – 8 – 9 – 10 – 11,12,13 – 14 – 15,16 – 19

Path 3 : 1 – 2 – 5 – 6,7 – 8 – 9 – 10 – 14 – 15,16 – 19

Path 4 : 1 – 2 – 5 – 6,7 – 8 – 9 – 10 – 11,12,13 – 14 – 8 – 17,18 – 19

Path 5 : 1 – 2 – 5 – 6,7 – 8 – 9 – 10 – 14 – 8 – 17,18 – 19

5.3.2. Pengujian *Black Box*

Pengujian *Black Box* merupakan pengujian yang berfokus pada fungsionalitas perangkat lunak atau aplikasi, fungsionalitas perangkat lunak ini berkaitan dengan masukan-masukan *user* dan kesesuaiannya dengan keluaran yang diinginkan. Dalam hal ini pengujian *Black Box* yang dilakukan terhadap aplikasi permainan Othello atau Reversi ini berkaitan dengan kesesuaian setiap menu dan mobilitas jalannya permainan. Berikut tabel pengujian alpha (*Black Box*) :

Tabel 5.2 Pengujian Aplikasi

Kasus dan Hasil Uji (Data Normal)			
Item Pengujian	Realisasi yang diharapkan	Hasil Pengujian	Kesimpulan
Klik Button Play	Masuk ke Page Memilih Level Permainan	Memilih Level Permainan	[X] Diterima [] Ditolak

Klik Button About	Untuk menampilkan informasi Aplikasi dan Pembuat	Tombol berfungsi sesuai yang diharapkan	[X] Diterima [] Ditolak
Klik Button High Scores	Untuk Menampilkan <i>score</i> pemain 15 teratas	Tombol berfungsi sesuai yang diharapkan	[X] Diterima [] Ditolak
Klik Button Help	Untuk Menampilkan cara bermain atau petunjuk permainan	Tombol berfungsi sesuai yang diharapkan	[X] Diterima [] Ditolak
Klik Button Beginner	Untuk memulai permainan dengan <i>depth 3</i>	Permainan dimulai dengan <i>depth 3</i>	[X] Diterima [] Ditolak
Klik Button Intermediate	Untuk memulai permainan dengan <i>depth 6</i>	Permainan dimulai dengan <i>depth 6</i>	[X] Diterima [] Ditolak
Klik Button Expert	Untuk memulai permainan dengan <i>depth 9</i>	Permainan dimulai dengan <i>depth 9</i>	[X] Diterima [] Ditolak
Klik Application Bar New Game	Untuk Mengawali Permainan Kembali	Permainan Dimulai Kembali	[X] Diterima [] Ditolak

Klik Application Bar Play Again	Untuk memulai permainan kembali dengan memilih level terlebih dahulu	Application Bar berfungsi sesuai dengan harapan	[X] Diterima [] Ditolak
Tap mobility pada board	Untuk menyimpan koin hitam pada papan main	Berfungsi sesuai dengan harapan	[X] Diterima [] Ditolak
Klik Oke pada saat Player Pass	Giliran kembali player sebelumnya	Berfungsi sesuai dengan harapan	[X] Diterima [] Ditolak
Klik Oke pada saat permainan berakhir	Menampilkan kondisi terakhir permainan (menang,kalah,seri)	Berfungsi sesuai dengan harapan	[X] Diterima [] Ditolak
Klik Button Save pada saat nama pemain terisi	Menampilkan halaman Highscore	Berfungsi sesuai dengan harapan	[X] Diterima [] Ditolak
Kasus dan Hasil Uji (Data tidak Normal)			
Aktivitas Pengujian	Realisasi yang diharapkan	Hasil Pengujian	Kesimpulan
Klik Button Save pada saat nama pemain kosong	Menampilkan alert nama tidak boleh kosong	Hasil sesuai dengan harapan	[X] Diterima [] Ditolak

BAB VI

KESIMPULAN DAN SARAN

6.1. Kesimpulan


Berdasarkan dari hasil penelitian, maka penulis dapat menarik kesimpulan bahwa :

1. Algoritma Minimax dapat dengan baik diterapkan pada langkah komputer dalam permainan reversi atau othello dikarenakan *game* reversi merupakan salah satu *game* dengan informasi lengkap dengan semua langkah dapat diketahui.
2. Dalam penelusuran setiap langkahnya, metode penelusuran DFS (*Depth-First Search*) dapat menjadi optimalisasi langkah AI (*Artificial Intelligence*) dan dapat menjadi penentu kedalaman penelusuran pada permainan reversi ini sehingga melahirkan tingkatan *beginner*, *intermediate*, dan *expert* pada permainan ini.
3. Implementasi rancangan permainan reversi pada Windows Phone dapat berjalan dengan baik dan dalam proses publikasi pada Windows Phone Store dapat diproses secara resmi oleh pihak Microsoft Development.

6.2. Saran

Adapun hal-hal yang perlu dikembangkan pada aplikasi ini diantaranya yaitu dapat ditambahkan fitur 2 *player* menggunakan teknologi WiFi atau *bluetooth*, dapat juga dibuatkan web *service* untuk *game* ini agar dapat bermain secara *online*. Dalam hal pengembangan algoritma minimax juga masih perlu algoritma-algoritma lain dalam hal optimalisasi langkah *Artificial Intelligence* (AI).

CURRICULUM VITAE

Personal Data	
	<i>Name :</i> RAHMAT KURNIA
	<i>Place / Date Of Birth :</i> SUKABUMI / February 15, 1991
	<i>Gender :</i> MALE
	<i>Nationality :</i> INDONESIAN
	<i>Religion :</i> ISLAM
	<i>Marital Status :</i> SINGLE
<i>Indonesian Citizenship ID Number :</i> 3272021502910021	
<i>Indonesian Passport ID Number :</i> -	

Address and Contact		
<i>Current Address :</i>	26 A, JALAN CIHAUR (KAWASAN DAGO, BANGBAYANG) RT.03 / RW.08, KEL SEKELOA, KEC COBLONG. BANDUNG WEST JAVA, INDONESIA. 40135	
<i>Permanent Address :</i>	NO.5, JALAN VETERAN 1 GANG KENANGA RT.01 / RW.03, KEL GUNUNGPANG, KEC CIKOLE. SUKABUMI WEST JAVA, INDONESIA. 43111	
<i>Telephone :</i>	<i>Home :</i> (+62) (22) 91981412	<i>Mobile :</i> (+62) 82121300162
<i>E-Mail Address :</i>	rahmat.kurnia@windowslive.com	

Formal Education		
1996 to 1997	TK Tunas Muda	(Kindergarten)
1997 to 2003	SD Negeri Ir. H. Juanda	(Elementary School)
2003 to 2006	SMP Negeri 2 Kota Sukabumi	(Junior High School)
2006 to 2009	SMK Negeri 1 (Kelompok Teknologi dan Industri) Kota Sukabumi	(Senior High School)
2009 to 2013	S1 Sistem Informasi UNIKOM Indonesia	(Bachelor Of Computer)

IT Capabilities and Skills	
<i>Analysis and Design of Information Systems :</i>	Structured Approaches and Object Oriented
<i>Programming Language :</i>	<ol style="list-style-type: none"> 1) PHP 2) MySQL 3) Java Script 4) CSS / J-Query 5) HTML 6) Pascal 7) C / C++ Programming 8) C# Programming (Windows Phone) 9) Java Programming (J2SE) 10) ORACLE
<i>Microsoft Office Application :</i>	<ol style="list-style-type: none"> 1) Microsoft Word 2) Microsoft Excel 3) Microsoft Access 4) Microsoft Power Point