Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)
Edisi. .. Volume. .., Bulan 20.. ISSN : 2089-9033

PENERAPAN ALGORITMA *NEGAMAX* UNTUK MENGHASILKAN LANGKAH YANG OPTIMAL PADA PERMAINAN DAM DAMAN

Panji Febriyanto[1]

[1] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung

E-mail : pfebri01@gmail.com[1]

ABSTRAK

Permainan dam daman merupakan permainan tradisional, permainan ini dimainkan oleh dua pemain. Permainan ini dimainkan di papan atau karton yang telah digambar. Dalam permainan ini terdapat 16 bidak untuk setiap pemain. Tujuan permainan ini adalah untuk menghabiskan bidak lawan dengan aturan yang telah disepakati bersama sebagai akhir permainannya.

Pada permainan dam daman ini terdapat banyak langkah. Satu bidak bisa mempunyai lebih dari dua kemungkinan langkah yang bisa diambil. Dari beberapa langkah tersebut ada yang menguntungkan, ada juga yang merugikan pemain. Jika pemain mengambil langkah bidak secara acak, maka kemungkinan kalah akan lebih besar. Untuk mencari langkah pergerakan bidak yang menguntungkan digunakan algoritma Depth-First Search. Salah satu algoritma DFS adalah algoritma *negamax*. Algoritma ini menggunakan dua buah fungsi, satu fungsi untuk memaksimalkan dan satu untuk meminimalkan kedua fungsi tersebut digabung menjadi satu fungsi yang dapat menegasikan dan terbalik setiap kali pemanggilannya. Dan untuk mempersempit pencarian digunakan algoritma alpha beta prunning.

Tujuan dari penelitian ini adalah untuk mengetahui banyaknya kemungkinan langkah yang bisa diambil dan tingkat kemenangan dengan menggunakan algoritma *negamax*. Banyaknya kemungkinan langkah yang bisa diambil berdasarkan hasil pencarian algoritma *negamax* adalah sebanyak 10 langkah kemungkinan. Sedangkan tingkat kemenangan komputer masih rendah, yaitu hanya 40%. Algoritma *negamax* yang diterapkan pada permainan dam daman belum optimal.

Kata kunci : Dam daman, *negamax, alpha beta prunning*.

1. PENDAHULUAN

Permainan tradisional merupakan kekayaan khasanah budaya lokal, ragam permainan tradisional sangat banyak ditemukan di Indonesia sehingga permainan tradisional ini seharusnya dipertahankan dan dilestarikan sebagai warisan budaya bangsa. Salah satu permainan tradisional adalah Damdaman, permainan ini berasal dari daerah jawa sekalipun di daerah lain ada. Damdaman biasanya dimainkan oleh anak laki-laki walaupun tidak ada larangan anak perempuan untuk memainkankannya karena termasuk permainan kompetitif, mengasah otak, ada yang kalah dan menang. Permainan ini dimainkan oleh dua orang pemain. Tujuan permainan ini adalah untuk menghabiskan bidak lawan dengan aturan yang telah disepakati bersama sebagai akhir permainannya [1].

Pada permainan dam daman ini terdapat banyak langkah. Dari beberapa langkah tersebut ada yang menguntungkan, ada juga yang merugikan pemain. Jika pemain mengambil langkah bidak secara acak, maka kemungkinan kalah akan lebih besar. Untuk mencari langkah pergerakan bidak digunakan algoritma Depth-First Search. Salah satu algoritma DFS adalah algoritma *negamax*. Algoritma *negamax* adalah sebuah penyederhanaan dari algoritma minimax. Algoritma ini menggunakan dua buah fungsi, satu fungsi untuk memaksimalkan dan satu untuk meminimalkan kedua fungsi tersebut digabung menjadi satu fungsi yang dapat menegasikan dan terbalik setiap kali pemanggilannya [2].

Pada penelitian mengenai permainan dam daman yang diterapkan dengan menggunakan algoritma minimax. Algoritma minimax cocok dalam menyelesaikan masalah dalam penelitian tersebut [3]. Penelitian lain pada aplikasi permainan gomoku dengan algoritma *negamax* dan alpha-beta search, didapatkan bahwa penggunaan metode *negamax* dan dapat memberikan solusi pencarian langkah terbaik [4].

Algoritma *negamax* digunakan untuk mencari semua kemungkinan langkah yang ada. Belum ada penelitian yang menggunakan algoritma *negamax* untuk

diterapkan kedalam permainan dam daman. Berdasarkan penjelasan tersebut, maka dalam penelitian ini akan menerapkan algoritma *negamax* untuk menghasilkan langkah yang optimal pada permainan dam daman. Algoritma *negamax* diharapkan dapat menyelesaikan permainan dam daman.

Berdasarkan latar belakang diatas, masalah yang akan dibahas dalam penelitian ini adalah bagaimana cara menerapkan algoritma *negamax* kedalam permainan dam daman untuk mencari kemungkinan langkah yang bisa diambil.

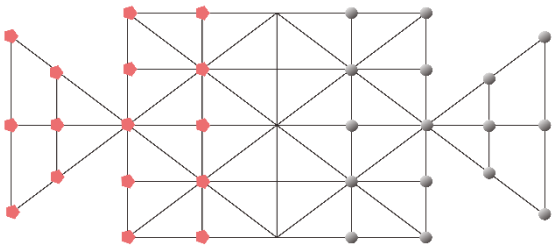
2. ISI PENELITIAN

2.1 Permainan Dam Daman

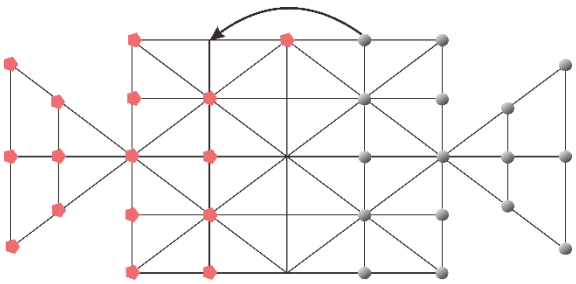
Permainan dam daman termasuk permainan tradisional, permainan ini dimainkan oleh dua pemain, permainan ini dimainkan di papan atau karton yang telah digambar. Dalam permainan ini terdapat 16 bidak untuk setiap pemain.

Untuk memulai permainan dam daman, langkah-langkahnya sebagai berikut:

1. Pemain harus berjumlah 2 orang
2. Pemain harus mencari batu/kerikil untuk bidaknya tetapi setiap pemain tidak boleh sama
3. Jumlah bidak untuk setiap pemain ada 16 buah
4. Gambar papan permainan dam daman, seperti pada gambar 1
5. Susun setiap batu/kerikil tadi pada papan permainan seperti pada gambar 1
6. Bidak hanya bisa melangkah satu langkah ke samping atau kedepan atau diagonal
7. Untuk makan bidak lawan caranya dengan melewati bidak lawan, bisa dilihat pada gambar 2
8. Lakukan suit untuk menentukan pemain giliran pertama.
9. Permainan berakhir apabila salah satu pemain sudah kehabisan bidak



Gambar 1. Papan Permainan Dam Daman



Gambar 2. Makan Bidak Lawan

2.2 *Artificial Intelligence*

Sebagian kalangan menerjemahkan *Artificial Intelligence* sebagai kecerdasan buatan, kecerdasan artifisial, intelijensia artifisial, atau intelijensia buatan. Istilah *Artificial Intelligence* sengaja tidak diterjemahkan ke dalam bahasa Indnesia karena istilah tersebut sudah sangat akrab bagi orang Indonesia. begitu juga dengan singkatannya yaitu AI, sudah sangat melekat di berbagai media ilmiah maupun non ilmiah.

Definisi AI dikelompokkan menjadi 4 kategori, yaitu *Thinking Humanly*, *Acting Humanly*, *Thinking Rationally* dan *Acting Rationally* [5].

2.3 Algoritma *Negamax*

Algoritma *negamax* merupakan bentuk sederhana dari algoritma minimax yang melakukan pencarian dengan menggunakan teknik algoritma DFS yang akan menelusuri setiap node untuk memperoleh hasil yang maksimum. Algoritma *negamax* identik dengan algoritma minimax untuk karakteristik kinerja. Meskipun sederhana untuk menerapkan dan cepat untuk mengeksekusi, itu skala dengan cara yang sama dengan pohon-pohon besar. Sebagian besar optimasi yang dapat diterapkan untuk *negamax*ing dapat dibuat untuk bekerja dengan ketat mendekati minimaxing [6].

Berikut ini adalah algoritma *negamax*.

```
def abNegamax(board, maxDepth, currentDepth, alpha, beta):

# Check if we're done recursing
if board.isGameOver() or currentDepth == maxDepth:
return board.evaluate(player), None

# Otherwise bubble up values from below

bestMove = None
bestScore = -INFINITY

# Go through each move
for move in board.getMoves():

newBoard = board.makeMove(move)

# Recurse
recursedScore, currentMove = abNegamax(newBoard,
maxDepth,
currentDepth+1
-beta,
-max(alpha, bestScore))
currentScore = -recursedScore

# Update the best score
if currentScore > bestScore:
bestScore = currentScore
bestMove = move

# If we're outside the bounds, then prune: exit immediately
if bestScore >= beta:
return bestScore, bestMove

return bestScore, bestMove
```

Gambar 3. Algoritma *Negamax*

2.4 Algoritma *Alpha Beta Prunning*

2.4.1 *Alpha Prunning*

Untuk memangkas dengan cara ini, kita perlu melacak skor terbaik yang dapat dicapai. Faktanya, nilai ini membentuk batas bawah pada skor yang dicapai. Kemungkinan menemukan urutan yang lebih baik dari dari langkah dalam pencarian, tetapi kita tidak akan pernah menerima urutan langkah yang memberi kita nilai rendah. Ini terikat lebih rendah disebut nilai alpha (kadang-kadang, tapi jarang, ditulis sebagai huruf Yunani α), dan pemangkasan yang disebut *alpha pruning*. Dengan melacak nilai *alpha*, kita dapat menghindari mempertimbangkan

langkah apapun dimana lawan memiliki kesempatan untuk membuatnya lebih buruk. Kita tidak perlu khawatir tentang berapa banyak langkah yang lebih buruk dari lawan yang bisa dibuatnya. kita sudah tahu bahwa kita tidak akan memberinya kesempatan [6].

2.4.2 *Beta Prunning*

Beta *pruning* bekerja dengan cara yang sama. Nilai beta (β lagi, jarang ditulis) melacak sebuah batas atas dengan harapan untuk mendapatkan nilai. Kami memperbarui nilai beta ketika kita menemukan urutan pergerakan yang lawan bisa untuk memaksa kita ke dalam.

Pada saat itu kami tahu tidak ada cara untuk mencetak lebih dari nilai beta, tapi mungkin ada beberap urutan belum ditemukan yang dapat digunakan untuk membatasi kita bahkan lebih jauh. Jika kita menemukan urutan dari gerakan bahwa skor lebih besar dari nilai beta, maka kita bisa mengabaikan itu, karena kita tahu kita akan tidak pernah diberi kesempatan. Alpha dan beta nilai bersama-sama memberikan jendela skor. Kita tidak akan pernah memilih untuk membuat gerakan yang skor kurang dari alpha, dan lawan kami tidak akan pernah membiarkan kita membuat gerakan untuk mendapatkan nilai lebih dari beta. Jika cabang pohon ditemukan yang berada di luar nilai-nilai ini, maka cabang dapat dipangkas. Karena pergantian antara meminimalkan dan memaksimalkan untuk setiap pemain, hanya satu nilai yang perlu diperiksa pada setiap posisi papan. Pada posisi papan di mana itu adalah lawan untuk bermain, kita meminimalkan skor, sehingga hanya skor minimal dapat berubah dan kita hanya perlu memeriksa terhadap alpha. Jika itu adalah giliran kami untuk bermain, kita memaksimalkan nilai, dan sehingga hanya beta yang diperlukan [6].

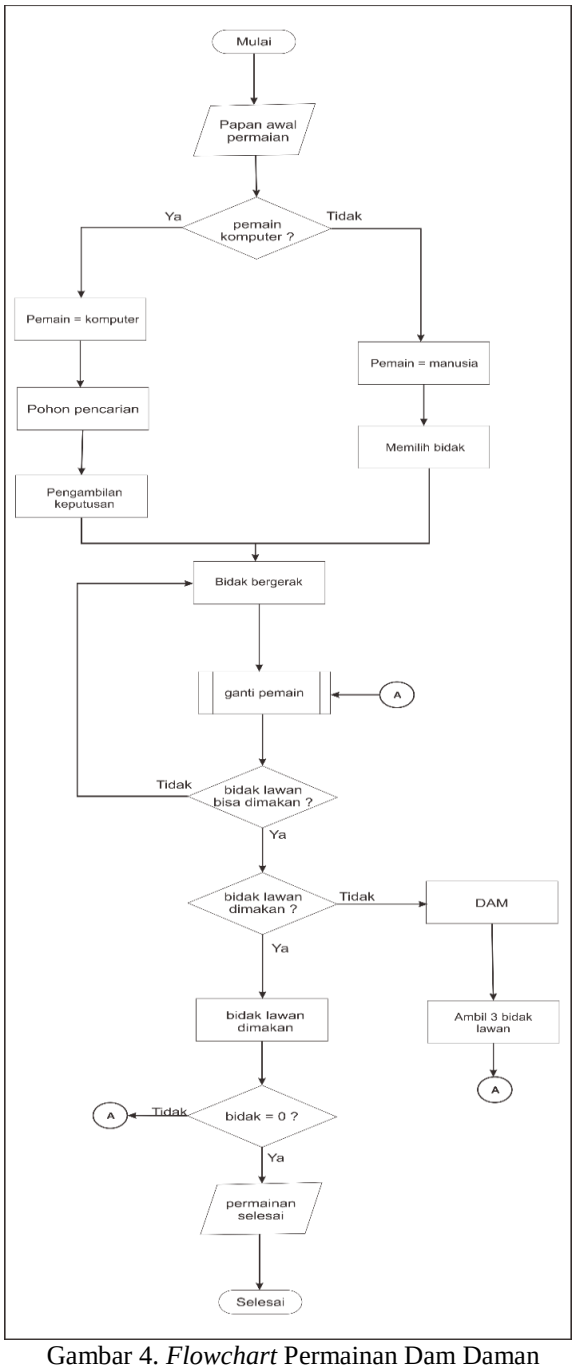
2.5 Analisis Masalah

Permainan ini merupakan permainan strategi yang dapat mengasah keterampilan dan pola pikir pemain, permainan ini menggunakan papan atau alas untuk menggambar tempat permainannya.

Permainan ini merupakan permainan yang sederhana karena bisa dimainkan dimana saja asalkan ada tempat untuk menggambar papan permainannya. Permainan dam daman memerlukan 16 buah bidak untuk setiap pemain. Sebelum melakukan permainan dilakukan suit terlebih dahulu untuk menentukan pemain pertama. Permainan dikatakan selesai apabila salah satu pemain sudah kehilangan semua bidaknya. Pada permainan dam daman ada peraturan dimana apabila ada pemain yang mendapatkan kesempatan untuk memakan bidak lawan tapi tidak memakannya maka akan kena hukuman, hukuman tersebut dinamakan dam. Hukuman dam tersebut yaitu mengambil tiga buah bidak pemain yang mendapat hukuman.

2.6 Analisis Permainan Dam Daman

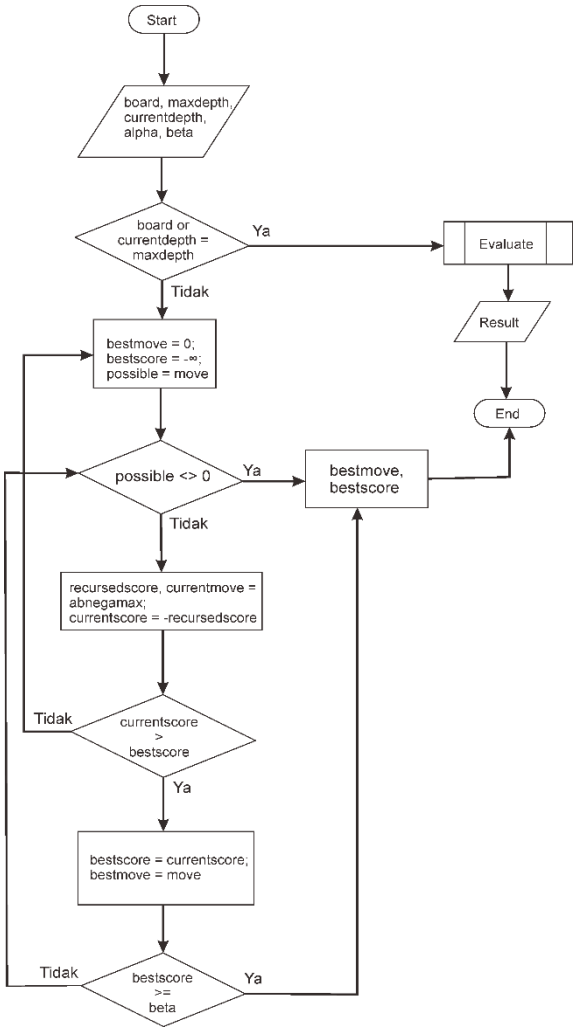
Pada awal permainan ditentukan siapa pemain pertama. Jika sudah ditentukan kemudian pemain pertama memulai permainan dengan menggerakan salah satu bidak miliknya ke depan, ke samping atau diagonal sesuai dengan jalurnya. Giliran pemain dinyatakan selesai setelah pemain selesai menggerakan bidaknya, dalam satu giliran pemain hanya bisa menggerakan satu bidak saja. Selanjutnya giliran lawan untuk menggerakan bidaknya. Begitu seterusnya saling bergantian menggerakan bidak masing-masing sampai salah satu bidak lawannya habis.



Gambar 4. *Flowchart* Permainan Dam Daman

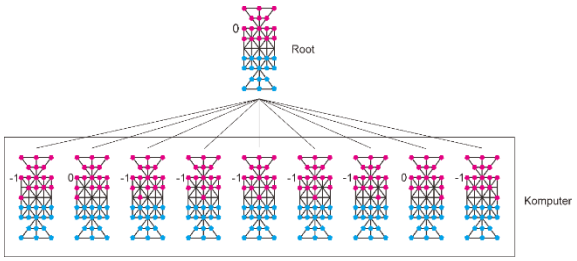
2.7 Analisis Algoritma

Pada bagian ini akan dijelaskan mengenai pemecahan permainan dam daman menggunakan algoritma *negamax*. Pencarian pohon *negamax* mengimplementasikan bahwa langkah lawan yang buruk adalah langkah yang anda yang baik. Berikut adalah gambar dari *flowchart* algoritma *negamax*.



Gambar 5. *Flowchart* Algoritma *Negamax*

Langkah pertama dari alur ini adalah pemberian inisialisasi nilai *board*, *maxdepth*, *currentdepth*, *alpha* dan *beta*. Setelah itu algoritma *negamax* akan mengecek nilai dari *board* dan *currentdepth* apakah salah satunya sama dengan nilai dari *maxdepth*, jika nilainya sama maka proses pencarian selesai. Jika nilainya tidak sama maka langkah selanjutnya melakukan pencarian pada pohon pencarian. Pada posisi *root* nilai *alpha* diisi dengan min integer untuk beta diisi dengan max integer, untuk *currentvalue* diisi dengan 0 dan untuk *bestvalue* juga diisi dengan min integer.



Gambar 6. Pohon Pencarian Algoritma *Negamax*

Pada saat kondisi mencari nilai maksimum maka nilai evaluasi dari node anaknya haruslah negatif nilai evaluasi, karena pada saat pengambilan nilai evaluasi nilainya akan dinegatifkan terlebih dahulu. Negatif dari negatif nilai evaluasi adalah nilai evaluasi, maka ketika dicari nilai maksimumnya akan didapatkan nilai maksimum dari nilai evaluasi. Pada saat kondisi mencari nilai minimum maka nilai evaluasi dari node anaknya haruslah positif nilai evaluasi, karena pada saat pengambilan nilai evaluasi nilainya akan dinegatifkan terlebih dahulu. Negatif dari positif nilai evaluasi adalah negatif nilai evaluasi, maka ketika dicari nilai maksimumnya akan didapatkan nilai minimum dari nilai evaluasi.

2.8 Analisis Kebutuhan Non Fungsional

2.8.1 Analisis Kebutuhan Perangkat Keras

Perangkat keras diperlukan untuk menjalankan suatu aplikasi guna mendukung proses kerja dari suatu sistem. Analisis perangkat keras digunakan untuk mengetahui mengenai spesifikasi perangkat keras dalam membangun dan menerapkan aplikasi, terdapat dua spesifikasi yang akan dijelaskan, yaitu spesifikasi perangkat keras pengembang dan spesifikasi perangkat keras Pengguna.

Berikut ini merupakan spesifikasi perangkat keras yang digunakan dalam membangun aplikasi ini, yaitu:

1. *Processor* 2.7 GHz
2. VGA 2 GB
3. RAM 2GB
4. *Harddisk* 1 TB
5. Resolusi Layar 1280 x 1024
6. *Keyboard* dan *Mouse* standar

Berikut ini merupakan spesifikasi perangkat keras pengguna atau spesifikasi minimum untuk menjalankan aplikasi ini, yaitu:

1. *Processor* 1 GHZ
2. RAM 512 MB
3. *Harddisk* 200 MB

2.8.2 Analisis Kebutuhan Perangkat Lunak

Analisis perangkat lunak digunakan untuk mengetahui mengenai spesifikasi perangkat lunak untuk membangun dan menerapkan aplikasi, terdapat dua spesifikasi yang akan dijelaskan, yaitu spesifikasi perangkat lunak pengembang dan spesifikasi perangkat lunak pengguna.

Berikut ini merupakan spesifikasi perangkat lunak yang digunakan dalam membangun aplikasi ini, yaitu:
1. Sistem Operasi Microsoft Window 7
2. Visual Studio 2012

2.8.3 Analisis Kebutuhan Pengguna

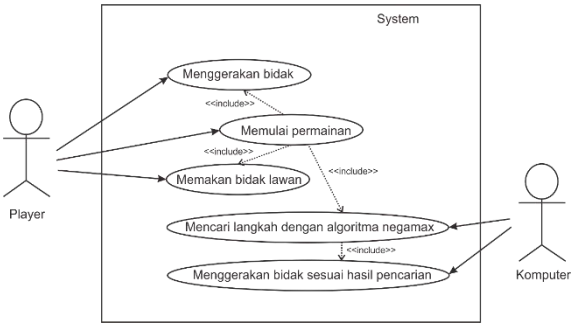
Selain dibutuhkannya perangkat lunak dan perangkat keras, Player atau penguna adalah bagian yang sangat penting untuk dapat menerapkan aplikasi ini. Analisis pengguna diperlukan untuk dapat mengetahui sasaran calon pengguna yang dapat menggunakan aplikasi. Pada penelitian ini pembangunan aplikasi ditujukan untuk pengujian implementasi algoritma sehingga pengguna disini disarankan memiliki karakteristik sebagai berikut:
1. Bisa mengoperasikan komputer dengan sistem operasi windows.
2. Mengetahui aturan permainan dam daman

2.9 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional menyangkut analisis sistem yang dilakukan yaitu dengan menggunakan UML. Adapun tahapan analisis sistem menggunakan UML adalah melalui *use case* diagram, *activity* diagram, dan *class* diagram.

2.9.1 *Use Case* Diagram

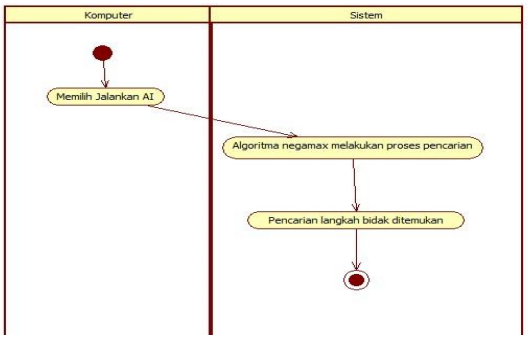
Use case diagram merupakan konstruksi untuk mendeskripsikan hubungan yang terjadi antara aktor dengan aktivitas yang terdapat pada sistem. Berikut ini merupakan *use case* diagram pada permainan dam daman.



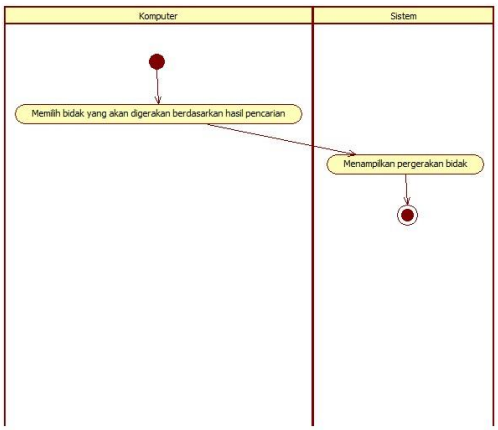
Gambar 7. *Use Case* Diagram

2.9.2 *Activity* Diagram

Activity diagram berisi penjelasan lebih lengkap mengenai urutan suatu proses yang terjadi antara pengguna dan sistem. Aktivitas pertama selalu dimulai dari pengguna, kemudian akan mendapat respon dari sistem yang nantinya akan menampilkan akhir dari proses tersebut.



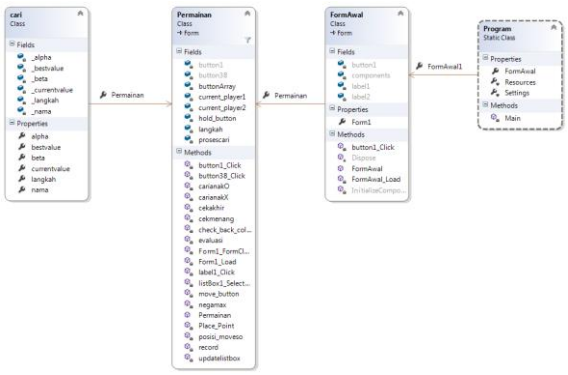
Gambar 8. *Activity* diagram mencari langkah dengan algoritma *negamax*



Gambar 9. *Activity* diagram Menggerakan bidak sesuai hasil pencarian

2.9.3 *Class* Diagram

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class* diagram dapat dilihat pada gambar 10.



Gambar 10. *Class* Diagram Dam Daman

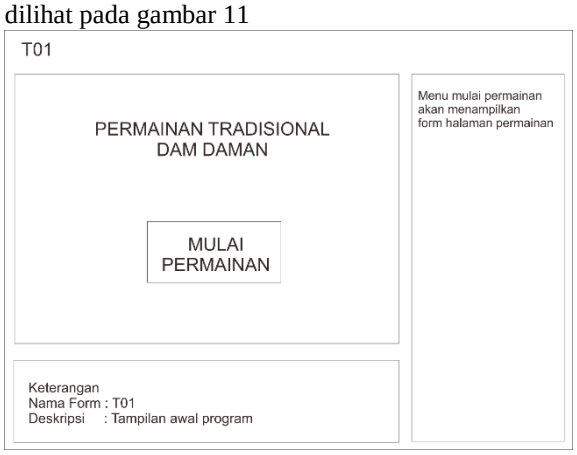
2.10 Perancangan Antarmuka

Perancangan antarmuka dibutuhkan untuk mewakili keadaan sebenarnya dari aplikasi yang akan dibangun. Perancangan antarmuka pada aplikasi ini adalah sebagai berikut

- Tampilan Halaman Utama

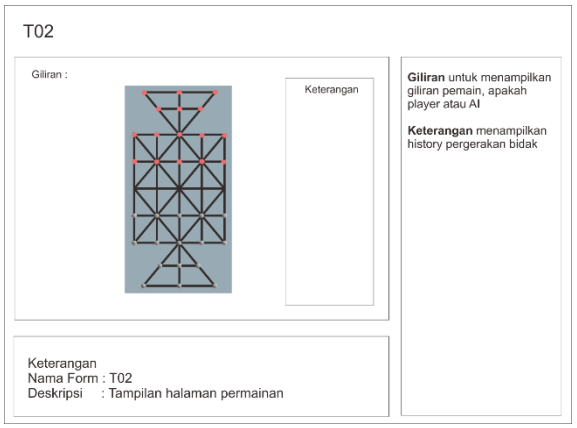
Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)
Edisi. .. Volume. .., Bulan 20.. ISSN : 2089-9033

Tampilan ini merupakan tampilan awal pada saat permainan dam daman dijalankan. Gambarnya dapat dilihat pada gambar 11



Gambar 11. Tampilan Halaman Utama

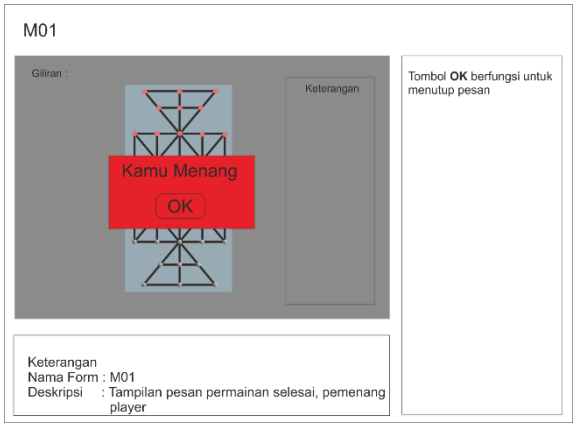
- Tampilan Halaman Permainan



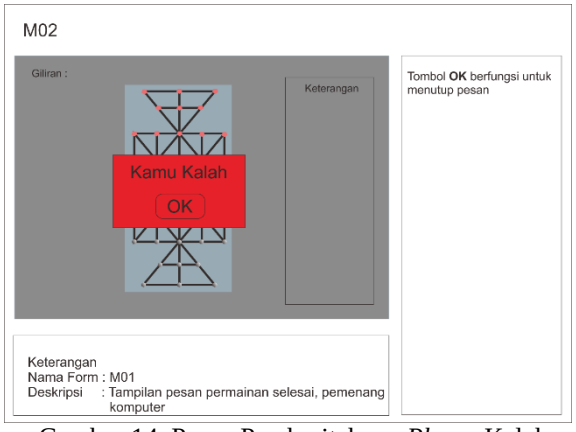
Gambar 12. Tampilan Halaman Permainan

- Perancangan Pesan

Tampilan ini merupakan tampilan awal pada saat permainan dam daman dijalankan. Gambarnya dapat dilihat pada gambar 13 dan gambar 14



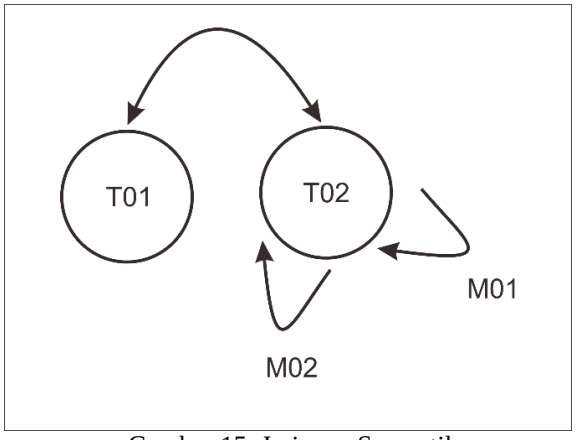
Gambar 13. Pesan Pemberitahuan *Player* Menang



Gambar 14. Pesan Pemberitahuan *Player* Kalah

- Jaringan Semantik

Jaringan semantik merupakan gambaran tentang relasi dari masing-masing halaman yang bisa diakses oleh pengguna.



Gambar 15. Jaringan Semantik

2.11 Pengujian Sistem

Pengujian yang dilakukan yaitu pengujian *blackbox* dan *whitebox.*

2.11.1 Pengujian *Blackbox*

Berikut merupakan hasil dari pengujian *blackbox*

Tabel 1. Pengujian Tombol Mulai Permainan

Kasus dan Hasil Uji			
Data Masukan	**Yang Diharapkan**	**Hasil**	**Kesimpulan**
Tombol Mulai Permainan	Menampilkan halaman permainan	Sistem menampilkan halaman permainan	[√] Diterima [ ] Ditolak

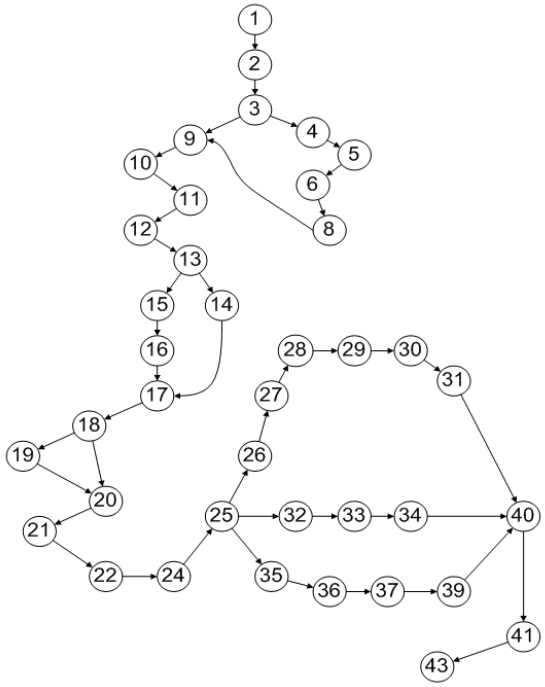
Tabel 2. Pengujian Tombol Jalankan AI

Kasus dan Hasil Uji			
Data Masukan	**Yang Diharapkan**	**Hasil**	**Kesimpulan**
Tombol Jalankan AI	Komputer menggerakan bidak sesuai dengan hasil pencarian algoritma *negamax*	Sistem menampilkan bidak komputer bergerak	[√] Diterima [ ] Ditolak

2.11.2 Pengujian *Whitebox*

Pengujian *white box* merupakan metode perancangan *test case* yang menggunakan struktur kontrol dari perancangan prosedural dalam mendapatkan *test case*. Adapun metode yang digunakan dalam pengujian *white box* ini adalah metode *Basis Path*. Metode *Basis Path* mengijinkan pendesain kasus uji untuk membuat perkiraan yang kompleks dari desain prosedural dan menggunakan perkiraan ini untuk mendefinisikan aliran eksekusi.

Berikut adalah diagram alir dari algoritma *negamax* yang telah diterapkan kedalam aplikasi.



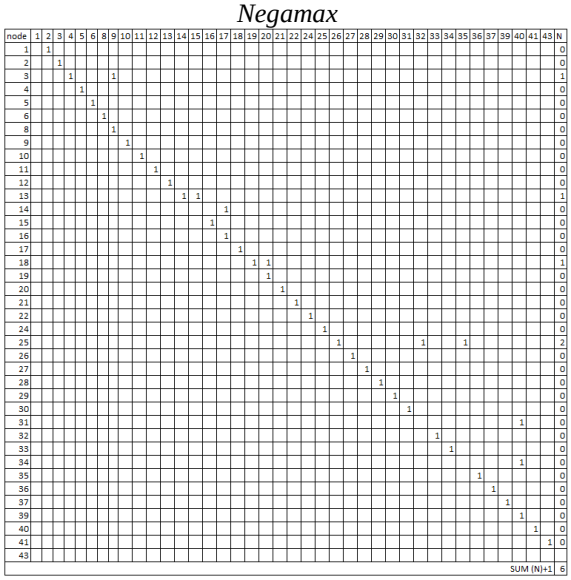
Gambar 16. Diagram Alir Algoritma *Negamax*

Dari diagram alir diatas, dapat dihitung *cyclomatic complexity* dengan rumus :
V(G) = Edge – Node + (2P)
V(G) = 43 – 39 + 2 = 6

Dari hasil perhitungan *cyclomatic complexity* terdapat 6 *independent path* yaitu:
Path 1 :1-2-3-4-5-6-8-9-10-11-12-13-14-17-18-20-21-22-24-25-26-27-28-29-30-31-40-41-43
Path 2 :1-2-3-9-10-11-12-13-14-17-18-20-21-22-24-25-26-27-28-29-30-31-40-41-43
Path 3 : 1-2-3-9-10-11-12-13-15-16-17-18-19-20-21-22-24-25-26-27-28-29-30-31-40-41-43
Path 4 : 1-2-3-9-10-11-12-13-15-16-17-18-19-20-21-22-24-25-32-33-34-40-41-43
Path 5 : 1-2-3-9-10-11-12-13-15-16-17-18-20-21-22-24-25-32-33-34-40-41-43
Path 6 : 1-2-3-9-10-11-12-13-15-16-17-18-19-20-21-22-24-25-35-36-37-39-40-41-43

Berikut adalah graph matriks pada diagram alir di atas untuk metode pencarian algoritma *negamax* sebagai berikut:

Tabel 3. Graph Matriks Pencarian Algoritma *Negamax*



V(G) = X + 1
V(G) = 5 + 1
V(G) = 6

Berdasarkan pengujian pada setiap metode, dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 6. Maka dapat disimpulkan bahwa pengujian white box pada algoritma *negamax* berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

3. PENUTUP

Kesimpulan merupakan ringkasan yang diambil dari pembahasan perangkat lunak yang dibuat, dimana perangkat lunak yang dibuat adalah penerapan algoritma *negamax* pada permainan dam daman, sedangkan saran diberikan untuk penggunaan sistem ini yang merupakan hasil perancangan yang

dapat menjadi bahan referensi untuk meningkatkan kinerja sistem ini agar lebih baik.

3.1 Kesimpulan

Berdasarkan hasil penelitian mengenai penerapan algoritma negamax kedalam permainan dam daman, dapat disimpulkan bahwa

1. Banyaknya kemungkinan langkah yang bisa diambil berdasarkan hasil pencarian algoritma negamax adalah sebanyak 10 langkah kemungkinan.
2. Tingkat kemenangan komputer masih rendah,yaitu hanya 40%. Algoritma negamax yang diterapkan pada permainan dam daman belum optimal.

3.2 Saran

Dari keterbatasan yang ada dan untuk pengembangan perangkat lunak ini ke depannya agar diperoleh hasil yang lebih baik, beberapa hal yang mungkin dilakukam sebagai berikut

1. Menambah nilai evaluasi dan menambah kedalaman pencarian agar tingkat kemenangan komputer lebih tinggi.
2. Menambahkan algoritma lain sebagai optimasi.

DAFTAR PUSTAKA

[1] S. Dharmamulya, Permainan Tradisional Jawa, Yogyakarta: Penerbit Kepel Press, 2005.

[2] C. M. Camacho, Adaptive Behavior in Artificial Intelligence, Helsinki Metropolia University of Applied Sciences, 2009.

[3] A. Prasetyo, Implementasi Kecerdasan Buatan Pada Permainan Damdaman Menggunakan Algoritma Minimax, Bandung: Skripsi : Universitas Pendidikan Indonesia, 2013.

[4] H. Kurniawan, "Aplikasi Permainan Gomoku dengan Algoritma Negamax," *Citec Journal*, vol. 1, no. 3, pp. 231-242, 2014.

[5] Suyanto, Artificial Intelligence, Bandung: Informatika, 2007.

[6] I. Millington dan J. Funge, Artificial Intelligence for Games Second Edition, CRC Press, 2012.