APPLICATION OF NEGAMAX ALGORITHM FOR OPTIMIZING MOVEMENT ON TURN BASED BOARD GAME OF DAM DAMAN

Panji Febriyanto[1]

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

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

ABSTRACT

Game of dam daman is a game of traditional, this game is played by two players. The game is played on a board or cardboard that has been drawn. In this game there are 16 pawns for each other. The purpose of the game is to spend pawns opponent with the rules that have been agreed upon as the end of the game.

In game of dam daman there are mant steps, one pawn can have more than two possible steps that could be taken. From some of the steps there is a profitable, there is also a disadvantage to the player. If the player takes a pawn move randomly, then the possibility of losing will be bigger. To search for a profitable move movement pawn used algorithms Depth-First Search. One of the DFS algorithm is an algorithm negamax. This algorithm uses two functions, one function to maximize and one to minimize both functions are combined into a single function that can be negated and reversed each time calling. And to narrow the search algorithm, alpha beta prunning.

The purpose of this research was to determine the number of possible steps that can be taken and the level of victory by using algorithm negamax. The number of possible steps that can be taken based on the results of the search algorithm negamax is as many as 10 steps possibility. As for winning rate computer is still low at only 40%. Negamax algorithm is applied to the game of dam daman not optimal.

Keywords : Dam daman, *negamax, alpha beta prunning*

1. INTRODUCTION

The traditional game is a wealth of local cultural treasures, the variety of traditional games are very commonly found in Indonesia so that this traditional game ought to be maintained and preserved as a national heritage. One of the traditional game is Damdaman, this game comes from the Javanese though in other areas there. Damdaman usually played by boys though there is no ban on girls to memainkankannya because it includes competitive games, sharpen the brain, there is winning and losing. The game is played by two players. The purpose of the game is to spend a pawn opponent with the rules that have been agreed upon as the end of the game [1].

Daman dam on the game, there are many steps. From some of the steps there is a profitable, there is also a disadvantage to the player. If the player takes a pawn move randomly, then the possibility of losing will be greater. To search for step movement of pawns used algorithms Depth-First Search. One algorithm is an algorithm negamax DFS. Negamax algorithm is a simplification of the minimax algorithm. This algorithm uses two functions, one function to maximize and one to minimize both functions are combined into a single function that can be negated and reversed each time calling [2].

In research on drafts daman applied by using minimax algorithm. Minimax algorithm is suitable to solve the problems in the study [3]. Another study on the application gomoku game with negamax algorithm and alpha-beta search, it was found that the use of methods negamax and can provide the best search solution step [4].

Negamax algorithm used to search for all possible available measures. There is no research that uses algorithms to negamax incorporated into drafts daman. Based on these explanations, so in this study will apply negamax algorithm to generate the optimum step on drafts daman. Negamax algorithm is expected to be completed drafts daman.

Based on the above, the issues to be addressed in this study is how to implement the algorithm negamax into drafts daman for possible steps that could be taken.

2. CONTENTS OF RESEARCH

2.1 Game Of Dam Daman

Game of dam daman including traditional games, the game is played by two players, the game is played on a board or cardboard that has been drawn. In this game there are 16 pieces for each player.

To start a game of checkers daman, the steps are as follows:

1. Players must 2 players
2. The player must find the stone / gravel to pawn but every player should not be the same

3. Number of pawns for each player there are 16 pieces
4. Picture daman checkers board game, like in figure 1
5. Arrange each stone / gravel had been on the game board as in Figure 1
6. Pawns can only move one step forward or sideways or diagonal
7. To take pawns opponents pawns way to pass an opponent, can be seen in Figure 2
8. Do a suit to determine a player's first turn.
9. The game ends when one player is already running out of pawn

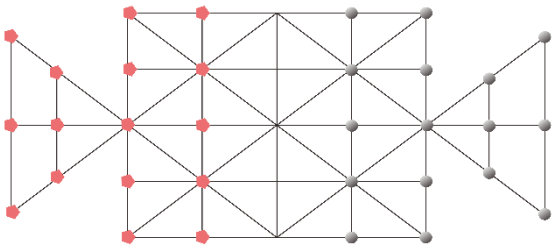


Figure 1. Board Game Of Dam Daman

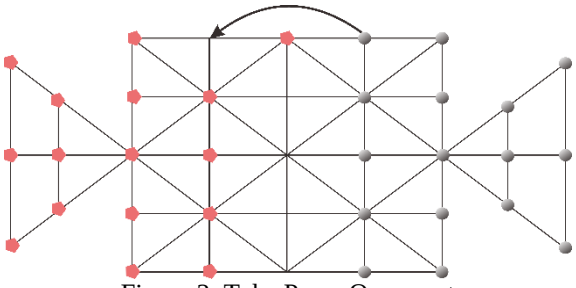


Figure 2. Take Pawn Opponent

2.2 Artificial Intelligence

Some people translate Artificial Intelligence as artificial intelligence, artificial intelligence, artificial intelligence or artificial intelligence. The term Artificial Intelligence deliberately not translated into Indnesia because that term is already very familiar to the people of Indonesia. as well as the abbreviation AI, already inherent in the various scientific and non-scientific media.

AI definitions are grouped into four categories, namely Thinking Humanly, Acting Humanly, Rationally Thinking and Acting Rationally [5].

2.3 Algorithm Negamax

Algorithm negamax is a simple form of minimax algorithm that searches using techniques DFS algorithm that searches for any node to obtain maximum results. Negamax algorithm is identical to the minimax algorithm for performance characteristics. Although simple to implement and quick to execute, the scale in the same way with big trees. Most of optimization that can be applied to negamaxing can be made to work closely approaching minimaxing [6].

Here is algorithm 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
```

Figure 3. Algorithm Negamax

2.4 Algorithm Alpha Beta Prunning

2.4.1 Alpha Beta Prunning

To summarize this way, we need to track down the best score that can be achieved. In fact, these values form the lower limit on the scores achieved. The chances of finding a better sequence of step in the search, but we will never accept the sequence of steps which give us a low value. This lower bound is called an alpha value (sometimes, but rarely, written as the Greek letter α), and trimming so-called alpha pruning. By tracking the value of alpha, we can avoid considering any measures which the opponent has a chance to make it worse. We do not need to worry about how many steps are worse than opponents could make it. we already know that we will not give him a chance [6].

2.4.2 Beta Prunning

Beta pruning work in the same way. Value beta (β again, rarely written) track down an upper limit in the hope to get the value. We are updating the beta value when we find a sequence of movements that the opponent can to force us into.

At that time we knew there was no way to print more than the value of beta, but there might be some undiscovered sequences that can be used to limit us even further. If we find a sequence of movements that score is greater than the value of beta, so we can ignore it, because we knew we would never be given a chance. Alpha and beta values together gives the window a score. We would never choose to make a motion that score less than alpha, and our opponents will never let us create a movement to get more value out of beta. If the branch of a tree is found that is outside these values, the branches can be trimmed. Because of the change between the minimize and

maximize for each player, only one value needs to be checked at each position on the board. On the board in the position where it is opposed to the play, we minimize the score, so that only a minimum score can change and we just need to check against alpha. If it is our turn to play, we maximize value, and so the beta is only required [6].

2.5 Analysis Of The Problem

This game is a strategy game that can sharpen the skills and mindset of the players, the game uses a board or board to draw a game.

This game is a simple game that can be played anywhere as long as there is nowhere to draw the game board. Drafts daman need 16 pieces pawns for each player. Before doing do suit the game beforehand to determine the first player. The game is said to be complete when one player has lost all of the pieces. In drafts daman if there are no rules where players get the chance to take opponents pawns but do not eat it then it will be subject to punishment, the punishment is called the dam. The dam penalty that took three pawns players were punished.

2.6 Analysis Game Of Dam Daman

At the beginning of the game to determine who the first player. If it has been determined then the first player to start the game by moving one of his pawns forward, sideways or diagonally on track. Rota players declared over after the player finishes moving the pieces, in one turn a player can only move one piece only. Next turn your opponent to move the pieces. And so on each in turn move the pawns each until one opponent pawn exhausted.

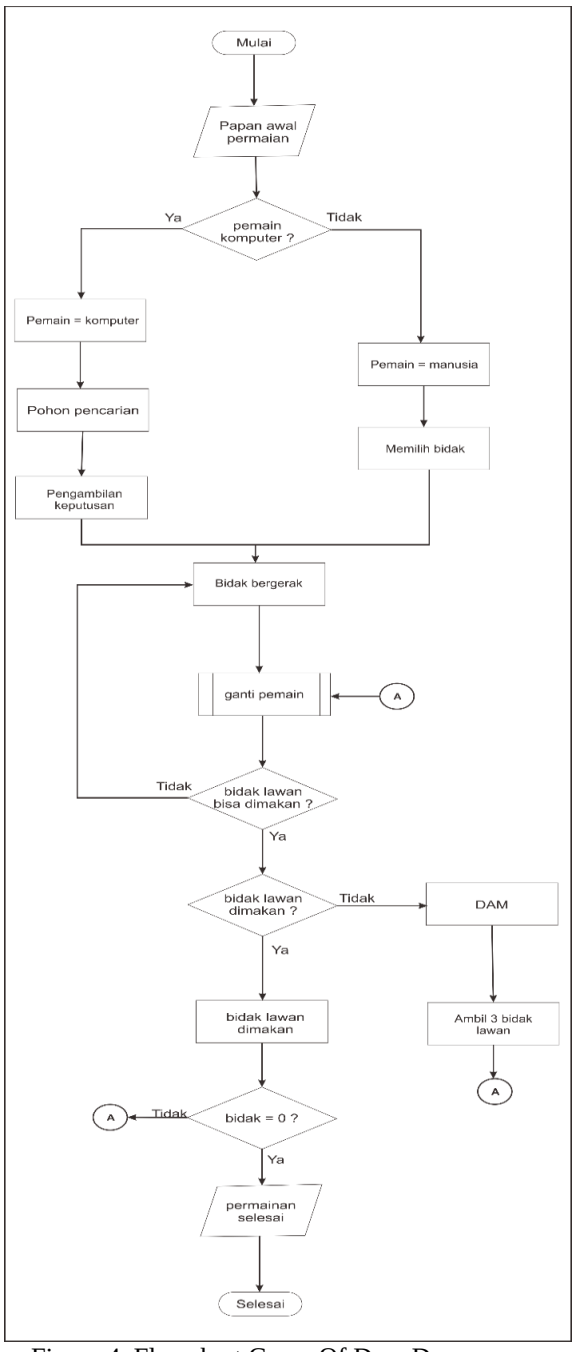


Figure 4. Flowchart Game Of Dam Daman

2.7 Analysis Of Algorithm

In this section will explain the game of checkers daman solving using algorithms negamax. Negamax implement tree search that opponents bad step is the step that you are good. Here is a picture of a flowchart algorithm negamax.

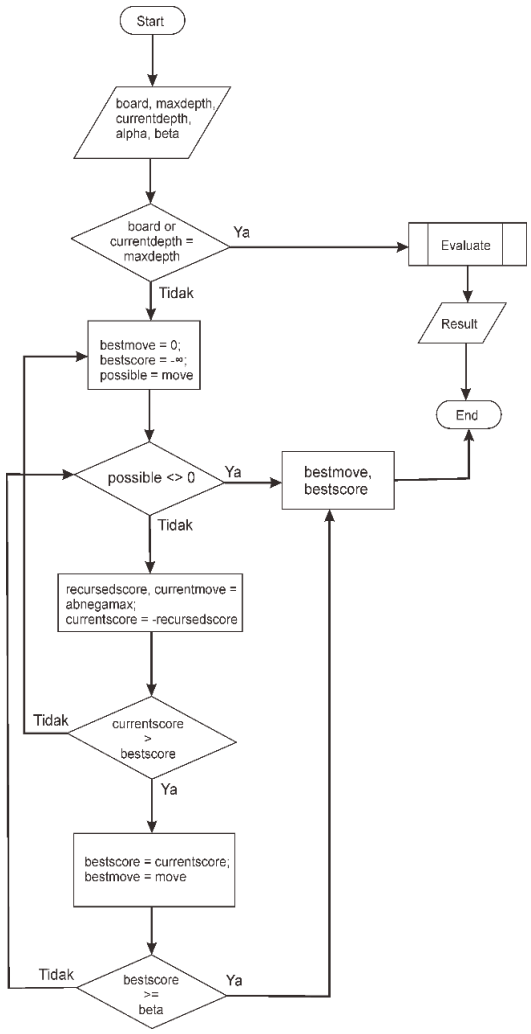


Figure 5. Flowchart Algorithm Negamax

The first step of this flow is the provision of board initialization value, maxdepth, currentdepth, alpha and beta. After that negamax algorithm will check the value of board and currentdepth whether one of them is equal to the value of maxdepth, if it is equal then the search process is complete. If the value is not the same as the next step to search in the search tree. At the root position alpha value filled with an integer min to beta max filled integer, for CurrentValue filled with 0 and for bestvalue also filled with integer min.

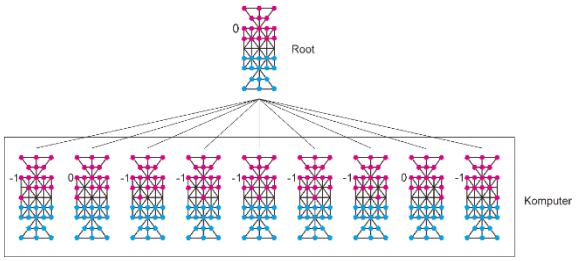


Figure 6. Tree Search Algorithm Negamax

At the time of searching for the maximum value of the evaluation value of its child nodes must be negative evaluation value, because at the time of taking the value of the evaluation value will be negated in advance. Negatives of the negative value of the evaluation is the evaluation value, so when looking for maximum value will get the maximum value of the evaluation value. At the time of searching for the minimum value of the evaluation value of child nodes should be the positive value of the evaluation, because at the time of taking the value of the evaluation value will be negated in advance. Negative than positive evaluation score evaluation score is negative, then when it sought the maximum value will be obtained from the minimum value of the evaluation value.

2.8 Analysis of Non-Functional Requirements

2.8.1 Analysis Of Hardware Requirements

The hardware needed to run an application in order to support the work processes of a system. Analysis of the hardware used to know about the hardware specifications to build and deploy applications, there are two specifications that will be explained, the hardware specifications developers and hardware specifications Users.

The following are the hardware specifications used in building this application,

1. Processor 2.7 GHz
2. VGA 2 GB
3. 2 GB RAM
4. Hard Drive 1 TB
5. Display Resolution 1280 x 1024
6. Keyboard and Mouse standards

The following are the specifications of the user's hardware or the minimum specifications to run this application,

1. Processor 1 GHZ
2. 512 MB RAM
3. Hard Drive 200 MB

2.8.2 Analysis Of Software Requirements

Analysis software is used to determine the specifications of the software to build and deploy applications, there are two specifications that will be explained, namely the specification of software developers and software specifications of the user.

The following are the specifications of the software used in building this application, namely:

1. Operating System Microsoft Window 7
2. Visual Studio 2012

2.8.3 Analysis Of User Needs

In addition to the need for software and hardware, Player or the user is a very important part to implement this application. User analysis is needed to ascertain the target potential users who can use the application. In this research, application development aimed at testing the implementation of the algorithm

so that users suggested herein has the following characteristics:

1. Can operate the computer with the Windows operating system.
2. Know the rules of the game of dam daman

2.9 Analysis of Functional Requirements

Analysis of functional requirements regarding system analysis done by using UML. The stages of the analysis system is through the use of UML use case diagrams, activity diagrams, sequence diagrams, and class diagrams.

2.9.1 Use Case Diagram

Use case diagram is a construction for describe the relationship between actors with activities contained in the system. Following This is the use case diagram on the drafts.

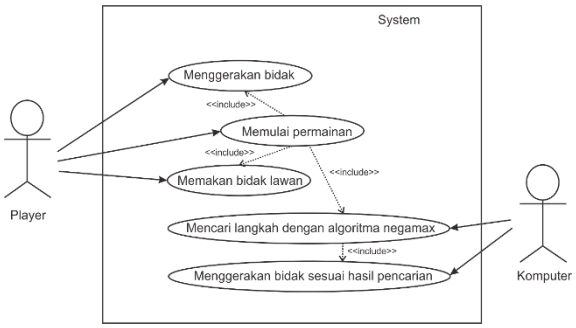


Figure 7. Use Case Diagram

2.9.2 Activity Diagram

Activity diagrams contain a fuller explanation of the sequence of a process that occurs between the user and the system. The first activity is always initiated from the user, then you will get a response from the system will display the end of the process.

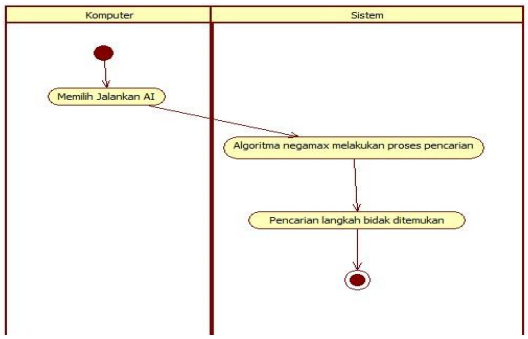


Figure 8. Activity diagrams looking for a step by the algorithm negamax

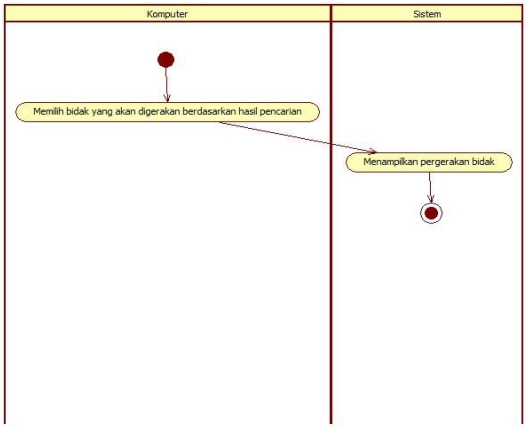


Figure 9. Activity diagrams Moving pawns appropriate search results

2.9.3 Class Diagram

Class diagram describing the system in terms of defining the structure of the classes that will be created to build the system. Class diagrams can be seen in Figure 10.

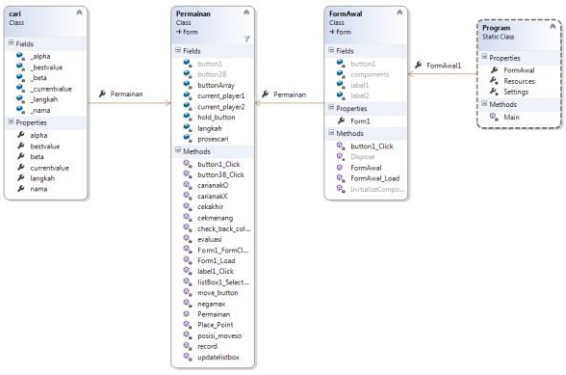


Figure 10. Class Diagram Dam Daman

2.10 Design Interface

The design of the interface is needed to represent the true state of the application to be built. The design of the interface on this application is as follows

- Display Main Page

This display is an early look at the game when the dam daman run. His picture can be seen in Figure 11

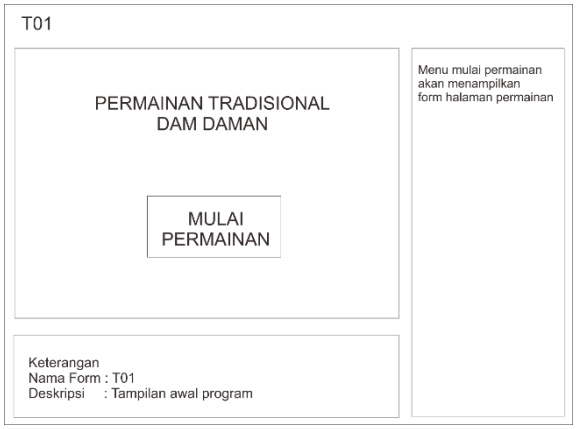


Figure 11. Display Main Page

- Display Game Page

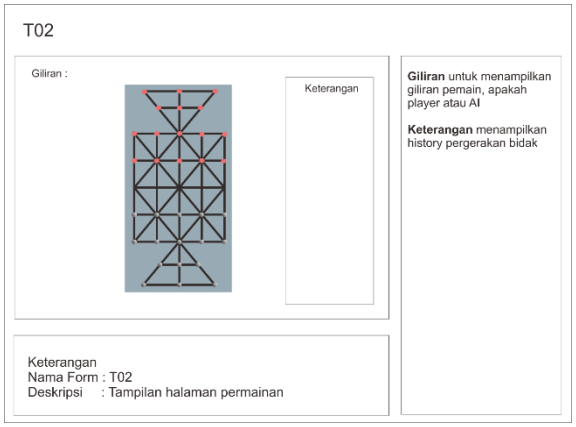


Figure 12. Display Game Page

- Design Messages

This display is an early look at the game when the dam daman run. His picture can be seen in Figure 13 and Figure 14

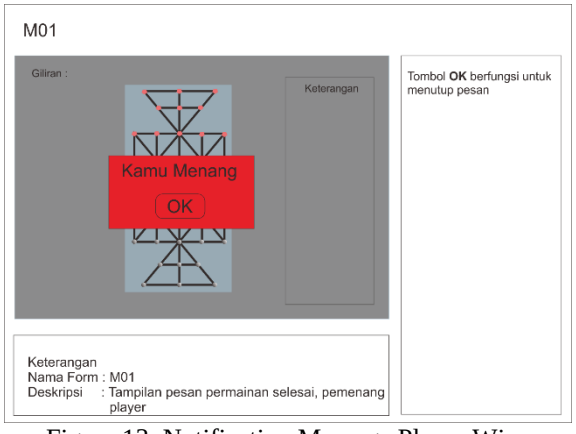


Figure 13. Notification Message Player Wins

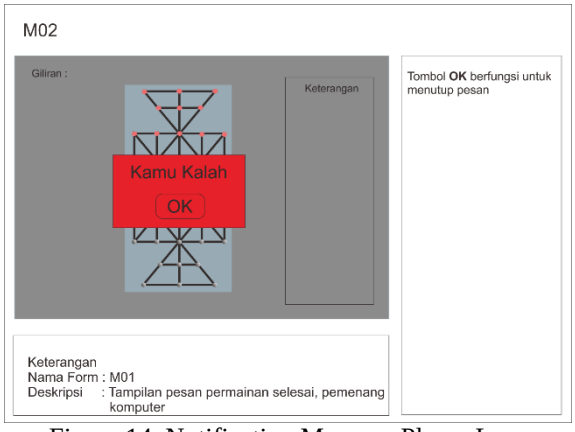


Figure 14. Notification Message Player Lose

- Semantic Network

Semantic network is a picture of the relation of each page which can be accessed by the user.

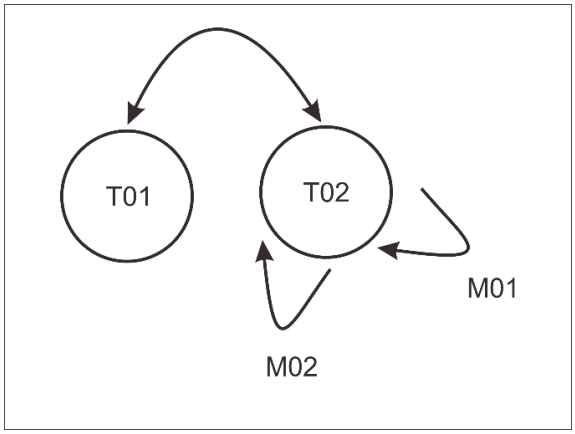


Figure 15. Semantic Network

2.11 Testing System

Tests done of blackbox and whitebox testing.

2.11.1 Blackbox Testing

The following is the result of black box testing

Table 1. Testing Button Mulai Permainan

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

Table 2. Testing Button 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 Whitebox Testing

White-box testing is a test case design method that uses the control structure of the procedural design in getting test case. The methods used in the white box testing is a method Basis Path. Basis Path method allows the designer to create a test case estimate of the complex of procedural design and use these estimates to define the flow of execution.

Here is a flow diagram of negamax algorithms that have been implemented into the application.

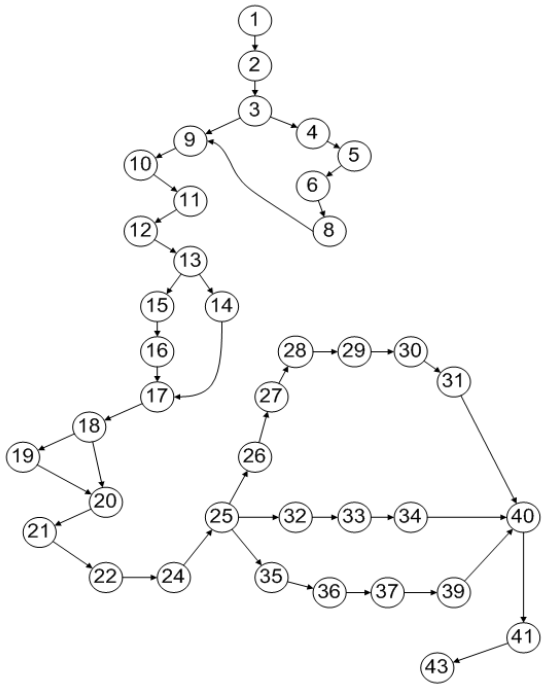


Figure 16. Flow Algorithm Negamax

From the flow diagram above, cyclomatic complexity can be calculated using the formula:
V (G) = Edge - Node + (2P)
V (G) = 43-39 + 2 = 6

From the calculation of cyclomatic complexity, there are 6 independent path, namely:
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

Here is a graph matrix in the flow above to search methods negamax algorithm as follows:

Table 3. Matrix Graph Search Algorithm Negamax

node	1	2	3	4	5	6	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	24	25	26	27	28	29	30	31	32	33	34	35	36	37	39	40	41	43	N
1		1																																						0
2			1																																					0
3				1					1																															1
4					1																																			0
5						1																																		0
6							1																																	0
8								1																																0
9									1																															0
10										1																														0
11											1																													0
12												1																												0
13													1	1																										1
14															1																										0
15																1																									0
16																	1																								0
17																		1																							0
18																			1	1																					1
19																				1																					0
20																					1																				0
21																						1																			0
22																							1																		0
24																								1																	0
25																									1					1		1									2
26																										1															0
27																											1														0
28																												1												0
29																													1											0
30																														1										0
31																																					1			0
32																															1									0
33																																1								0
34																																					1			0
35																																		1						0
36																																			1					0
37																																				1				0
39																																					1			0
40																																						1		0
41																																							1	0
43																																								0
																																						SUM (N)+1	6

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

Based on testing on each method, resulting Cyclomatic Complexity same value is 6. So we can conclude that white-box testing on negamax algorithms work well, because each test on the same value.

3. CLOSING

The conclusion is a summary drawn from the discussion made software, where software is created is the application of algorithms negamax on drafts daman, while the advice given to the use of these systems is the result of design that can be a reference to improve the performance of these systems in order to better.

3.1 Conclusion

Based on the results of research on the application of algorithms negamax daman dam into the game, it can be concluded that

1. The number of possibilities steps can be taken based on the results of the search algorithm is as many as 10 steps negamax possibilities.
2. The level of computer victory was still low at only 40%. Negamax algorithm is applied to the game of checkers daman has not been optimal.

3.2 Suggestion

Of the limitations of existing funds for the development of this software in the future in order to obtain better results, some things may dilakukam as follows

1. Add value and add depth evaluation of the search so that higher-level computer wins.
2. Adding another as an optimization algorithm.

REFERENCES

[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 and J. Funge, Artificial Intelligence for Games Second Edition, CRC Press, 2012.