JUISIK Vol 2 No. 1 Maret (2022) – p-ISSN: 2827-8135, e-ISSN : 2827-7953 halaman 12-17

JURNAL ILMIAH SISTEM INFORMASI DAN ILMU KOMPUTER
Halaman Jurnal: HYPERLINK "http://journal.sinov.id/index.php/juisik/index"
http://journal.sinov.id/index.php/juisik/index
Halaman UTAMA Jurnal : HYPERLINK "https://journal.sinov.id/index.php"
https://journal.sinov.id/index.php

MINIMAX ALGORITHM BASED ON "MAIN INI YUK" GAME

Sirajuddin Hawari [a], Muhammad Zulfikri Maulana[b], Teuku Binzar Nawaf Musyaffa[c],
Desy Komalasari [d], Mutiara Persada Pulungan[e]
[a] Program Studi Ilmu Komputer, sirajuddin.h@students.esqbs.ac.id , STIMIK ESQ
b Program Studi Ilmu Komputer, m.zulfikri.m@students.esqbs.ac.id , STIMIK ESQ
[c] Program Studi Ilmu Komputer, t.binzar.n@students.esqbs.ac.id, STIMIK ESQ
[d] Program Studi Ilmu Komputer, desy.komalasari@esqbs.ac.id, STIMIK ESQ
[e] Program Studi Ilmu Komputer, m.persada.p@students.esqbs.ac.id, STIMIK ESQ

ABSTRACT

In the game there are rules for each game. The rules are made as a challenge to achieve victory. To achieve victory requires analysis of the right algorithm in making decisions. Minimax is one of the best decision-making algorithms to be applied in a game. The time complexity of the minimax algorithm is O(b^m) which b is the branching and m is the depth.

Keywords: Game, Algorithm, Minimax, Analysis, Complexity.

1. INTRODUCTION

In this game there is some rule that apply for both player and computer have to follow, player must find the best case to win the game but choosing the best path is not easy as it looks like, as for the computer minimax algorithm is applied to find the best route for itself also preparing the worst route for the player.

The game itself is based on indonesian classic kids game where each player have to prepare 3 finger (thumb, point, pinky), after that each player have to choose which finger to put down one by one and using turn base method for the gameplay, the winning condition is where certain finger met their weaker opponent:

- Pinky finger > Thumb finger
- Pointy finger > Pinky finger
- Thumb finger > Point finger.

For an easier picture it's like a rock paper scissors game but with more steps applied on it so the player and computer have their own strategy to win it.

Received Januari 30, 2022; Revised Februari 2, 2022; Accepted Maret 22, 2022

2. THEORITICAL BASIS

2.1. *Analysis Of Algorithm*

In computer science, algorithm analysis is the process of determining the computational complexity of an algorithm, that is, the time, memory, or other resources required to run the algorithm. Usually, a function is determined that associates the length of the algorithm's input with the number of steps to perform (time complexity) or the number of storage locations used (spatial complexity). The algorithm is said to be efficient if the value of this function is small or grows slowly relative to the size of the input. The best, worst, and average case descriptions can all be of practical concern, as different inputs of the same length can cause the algorithm to behave differently. Unless otherwise stated, the function that describes the performance of an algorithm is usually the upper bound determined from the worst-case input to the algorithm.

In the theoretical analysis of algorithms, it is common to estimate the complexity of an algorithm in an asymptotic sense. That is, estimating a complexity function for arbitrarily large input data. It uses Big O, Big Omega, and Big Theta notations. For example, a binary search is considered to be performed in steps proportional to the logarithm of the length of the sorted list over which the search is performed, or colloquially "logarithmic time", proportional to O(log(n)). Different implementations of the same algorithm may have different efficiencies, so asymptotic estimation is commonly used. However, the effectiveness of both "reasonable" implementations of this algorithm has to do with a constant multiplication factor called the hidden constant.

2.2. *Minimax Algorithm*

Minimax algorithm is an algorithm that is applied to Artificial Intelligence-based games, such as chess or others. The Minimax algorithm will always check all the possibilities until the end of the game, then the check will produce a game tree that contains all of these possibilities [1]. In Minimax-based games such as chess, of course, very large resources are needed to process the search for the game tree, because the combinations of possibilities for playing chess in each move are numerous. The Minimax algorithm works recursively and is able to analyze all possibilities in the game that is used to produce the best decisions, so that it can make the opponent lose (Minimum). All opponent strategies will be calculated with the same algorithm over and over again. In the first step the computer will analyze the entire game tree, then for each step, the computer will select the move that will make the opponent lose (Minimum) and make the computer itself gain (Maximum). In order to determine the best decision, it takes a value that represents the loss or gain that will be obtained if this step is chosen. Therefore, the Minimax algorithm uses a Heuristic function to evaluate the value as a value that represents the result of the game that will occur if this step is selected, for example, in the game Tic Tac Toe, the value 1.0,-1 is used to represent the final result of the game in the form of win, draw and lose. From these Heuristic values, the computer will determine which node of the game tree is the most suitable for the computer to use to determine the steps in the game so as to achieve a victory.

3. RESEARCH METHOD

Parts of Research Methods may include analysis, architecture, methods used to solve problems, or implementation. To start doing research, we first analyze the game system that will be made. The game will be played by two players, each of whom will take turns to choose which finger to remove until only one finger remains on their hand. To determine who wins there are conditions rules that must be followed. The rules can be seen from the rules table that will be discussed in the next chapter.

From this analysis, we determine how the program will run according to the flow of the game. First, the player should input the selection according to the finger on the enemy (we use an array as the finger element on player and enemy). Then the program will delete the array on the enemy according to the input that has been entered. Next, the enemy will choose the finger of the player to be removed. In this section, the minimax algorithm is used to determine the best choice that the enemy must choose to reach victory.

The minimax algorithm itself will take the highest winning weight. We use the weights from the previously analyzed rules by giving a value 10 for the condition when the enemy wins, -10 for the condition when the enemy loses, and 0 for the condition of the draw. Minimax will check the condition that will occur (recursively) every time the player or enemy makes a decision until each condition has one finger remaining and then reads the rules in the game. After that it returns the value of the previous rule by concatenating them. When it’s the player’s turn, the smallest value will be taken, while when it’s the enemy’s turn the largest value will be taken.

After everything has been analyzed and got a clear idea, we build a program using the Python programming language to create a game based on command line interface and also apply the minimax algorithm to make enemy decisions when it is their turn.

4. RESULT AND ANALYSIS

In this section, we will discuss the results that have been carried out.

4.1 Result

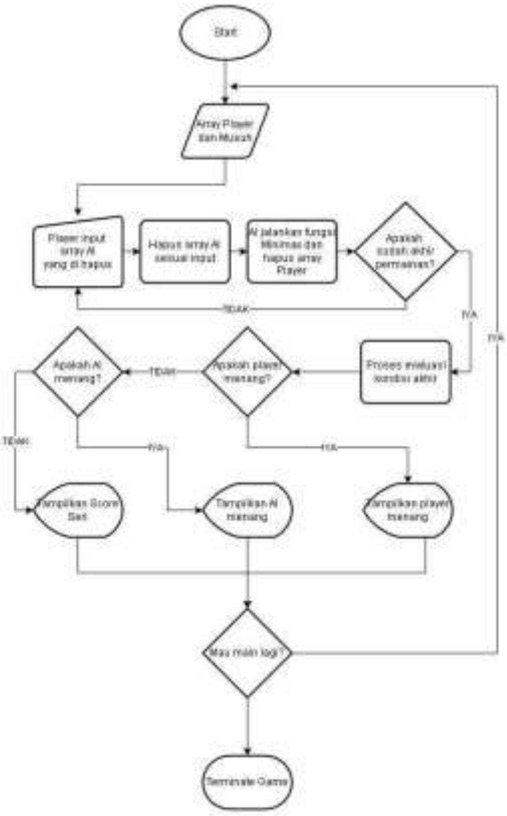
4.1.1 Table And Picture

Show placement tables in text and have exact labels from source, column headers, and other notations. Once the text has been received for publication, complex tables and figures (diagrams, tables, graphs, etc.) should be prepared in order to be ready for production.

Table 1. Possibility Case

Player Finger	Comp Finger	Condition	Score
		Player-WIN	-10
		Player-WIN	-10
		Player-WIN	-10
		Draw	0
		Draw	0
		Draw	0
		Comp:Win	10
		Comp:Win	10
		Comp:Win	10

For the flow of the game process that will be carried out, we also have created a flowchart that is in accordance with the original game. The following is an overview of the flow of the game in the program created:



Gambar 1. Game Flowchart

4.1.2 Algorithm Analysis

When the minimax algorithm performs a search, each step removes the finger array one by one. until there is one remaining enemy array and one player array remaining. after reading the rules to return the score value, the algorithm returns the deleted array. When viewed from how the algorithm runs, minimax looks for branching to the most basic condition and then explores other basic conditions. The algorithm is almost the same as the DPS algorithm which searches for the lowest root first.

In theory, what I've read about DFS language has a time complexity of b to the power of m in Big-O notation, which is O(b^m), where b is the branch and m is the depth of the tree data structure. and there is also space complexity with the same variables, namely b and m with Big-O notation, namely O(bm).

4.2 *Analysis*

Implementation of the minimax algorithm in the case of traditional games is quite powerful in computer decision making in choosing the best way to win. But for implementation to a case that has a lot of possible conditions, this algorithm is not good. In the program that we made there is a limit where only the player gets the first turn while the enemy just doesn’t get the first turn. This is considered because we can still implement the minimax algorithm when the enemy’s condition gets the first turn.

5. CLOSING STATEMENT

5.1. *Conclusion*

1) The application of the minimax algorithm in games made for computer decision making gives good results because the computer never loses.
2) The Minimax algorithm is very good to use when it's the turn to take decisions alternately from each player.
3) The time complexity of the minimax algorithm is the same as the time complexity of the DFS algorithm, it's O(b^m) which b is the branch, and m is the depth..

5.2. *Suggestions*

Suggestions for further research, if you want to use the minimax algorithm, you can add optimization algorithms such as alpha-beta pruning or dynamic programming in order to provide optimal results and reduce the time complexity required by the minimax algorithm itself.

DAFTAR PUSTAKA

[1] GeeksforGeeks, "Minimax Algorithm in Game Theory | Set 1 (Introduction)." GeeksforGeeks.org, 24 11 2021. [Online]. Available: https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction. [Accessed 27 12 2021].

[2] GeeksforGeeks, "Minimax Algorithm in Game Theory | Set 3 (Tic-Tac-Toe AI - Finding Optimal move)," GeeksforGeeks.org, 19 10 2021. [Online]. Available: Minimax Algorithm in Game Theory | Set 3 (Tic-Tac-Toe AI - Finding optimal move) - GeeksforGeeks. [Accessed 27 12 2021].

[3] JavaTPoint, "Mini-max Algorithm in Artificial Intelligence," Javatpoint.com. [Online]. Available: Artificial Intelligence | Mini-Max Algorithm - Javatpoint. [Accessed 29 12 2021]

[4] Sumali, Brian, Siregar, Ivan Michael, Rosalina. "Implementation of Minimax with Alpha-Beta Pruning as Computer Player in Congklak". *Jurnal Teknik Informatika dan Sistem Informasi*, Volume 2 Nomor 2, Agustus 2016.

[5] Videgaín, Santiago & Sánchez, Pablo García. "Performance Study of Minimax and Reinforcement Learning Agents Playing the Turn-based Game Iwoki". *Applied Artificial Intelligence*, 35:10, 717-744, DOI: 10.1080/08839514.2021.1934265