Journal of Applied Intelligent System (e-ISSN : 2502-9401 | p-ISSN : 2503-0493)
Vol. 9 No. 1 2024, pp. 88 – 100

Completing Sudoku Games Using the Depth First Search Algorithm

Fauzan Maulana Alfany[1], **Christy Atika Sari***[2], **Cahaya Jatmoko**[3]
[1,2,3]Study Program in Informatics Engineering, Faculty of Computer Science, University of Dian Nuswantoro
Imam Bonjol 207, Semarang, 50131, Indonesia
E-mail : fauzanmaulanaalfany16@gmail.com[1], christy.atika.sari@dsn.dinus.ac.id[2]
cahayajatmoko@dsn.dinus.ac.id[3]
**Corresponding author*

Deddy Award Widya Laksana[3], **Candra Irawan**[4], **Solichul Huda**[5]
[3]Study Program in Visual Communication Design, Faculty of Computer Science, University of Dian Nuswantoro
[4]Study Program in Information System, Faculty of Computer Science, University of Dian Nuswantoro
[5]Study Program in Informatics Engineering, Faculty of Computer Science, University of Dian Nuswantoro
Imam Bonjol 207, Semarang, 50131, Indonesia
E-mail : deddyaward.widyalaksana@dsn.dinus.ac.id[3], candra.irawan@dsn.dinus.ac.id[4]
solichul.huda@dsn.dinus.ac.id[5]

Received 8 December 2015; Revised 10 February 2016; Accepted 2 March 2016

Abstract - Sudoku is a digital game that is included in the type of logic-based puzzle game where the goal is to fill in the puzzle with random numbers. Therefore, in this research it is proposed to use Artificial Intelligence which contains the Depth First Search Algorithm to track the number of possible solutions that lead to only one so that it becomes efficient. This game has different levels of difficulty such as easy, medium and difficult. The time and complexity of execution will vary depending on the difficulty so it is proposed to use Android Studio software. The experimental results prove that there is an increase in playing the Sudoku game quickly and accurately by applying the Depth First Search Algorithm method. This is proven by the ability to complete this game using the Depth First Search Algorithm using the Android Studio programming language. The average time at the easy level is 11:04 minutes, at the normal level is 10:52 minutes, at the hard level is 25:46 minutes, and at the extreme level is 38 minutes

Keywords - Sudoku, Depth First Search Algorithm, Backtracking Algorithm, Android Studio

1. INTRODUCTION

Artificial intelligence or often referred to Artificial Intelligence as very important in today's modern era in solving problems and simplifying them to be precise and concise. Artificial intelligence is widely used in applications such as functions that have the aim of solving various problems depending on the desired application goals. An example of a problem that artificial intelligence has successfully solved is playing the sudoku game briefly and accurately. Sudoku is a digital game that is included in the puzzle game type. Sudoku is generally known as a number puzzle game based on the knowledge of thinking rules where the goal is to enter something into the puzzle with a number without a pattern, usually 3x3, 4x4, and up to 9x9 [1], [2]. The sudoku game has a unique basis, creating many number combinations that bring many possibilities to the sudoku game. Entering numbers in this sudoku game requires patience and accuracy. Because sudoku involves NP-complete problems, it is difficult or impossible to solve them simultaneously [3]. To solve this problem, you need Artificial Intelligence containing an algorithm that is suitable for solving sudoku games. The Depth First Search algorithm is one of them. The algorithm keeps track of the number of possible solutions or a solution that leads to only one, so it can find the solution or solutions in a small number of efficient search sequences. With the application of Artificial Intelligence, the sudoku game is completed shorter and more precisely. Dynamic Problem Solving is defined as solving problems whose circumstances are constantly changing and where there are many possible solutions. These changes follow definite patterns that can be grouped into functions or time patterns [4]. The basic principle of Dynamic Problem Solving is to analyze all possible ways to provide a solution and determine the best way to implement it. In some conditions, the program does not only examine or analyze existing steps as a reference when determining a problem solution, because there may be many solutions to a problem, such as board games such as chess or checkers. If you try all these solutions one by one, the program will not run efficiently. The algorithms used by the solver are two: one that checks whether the question/answer given by the player follows the basic rules of sudoku or not (possible) and another that solves the resulting question or response (problem solving). Consisting of two volumes, the algorithm used to generate the problem consists of two volumes: the algorithm that describes the game board, contents, and the algorithm that solves the problem.

The Backtracking algorithm is based on the Depth First Search (DFS) algorithm, but the search results only solve problems based on references [5]. The Depth First Search (DFS) algorithm is a problem-solving search algorithm used in artificial intelligence. This algorithm is an algorithm that lacks information. That is, algorithms that search in a specific position, but do not contain any information on which to base their search but instead they follow a certain way of working. Depth First Search performs a search in a tree structure. It is a collection of all possible states implemented in a tree structure. At the top is the root which contains the initial state (the initial state of the search), and below it contains the successor state of the target state.

Sudoku is a digital game that is included in the puzzle game type. Generally have sizes 3x3, 4x4, up to 9x9. Sudoku puzzles allocate one number from 1 to 9 for each position in a 9x9 matrix. Numbers may not be repeated in any row or column, and each puzzle has several cells filled with numbers, in nine 3x3 blocks [6]. Unlike other number games, sudoku does not train arithmetic intelligence (the ability to add or subtract numbers), but sudoku trains players' logical intelligence to solve sudoku by filling in all the boxes without breaking the existing rules with difficulty levels (easy, medium, hard, and very hard.

Depth First Search is the basis of an algorithm for finding connected elements in undirected graphs, strongly connected elements in directed graphs, and the topological order of undirected graphs [7], [8]. Specifically, the overall order in which the DFS visits nodes G is called depth-first order, if the DFS starts from a node that is connected to all nodes. In this backtracking technique we use recursive calls to find the solution by building the solution step by step. In each step it finds all possible solutions of the problem and removes solutions that cannot lead to a solution of the problem based on the problem constraints.

In previous research, the main objective of this game was to fill the entire space with numbers from 1 to 9. In this research, using the backtracking algorithm method is a refinement of the brute force and exhaustive search algorithms, namely establishing a state-space tree (state space) to looking for a solution, but the candidate solutions found are the same, the algorithm becomes inefficient [9]. As candidate solutions are found by the exhaustive search algorithm, the algorithm creates partial solutions from the candidate solutions and evaluates the solutions one by one. If the partial solution built does not meet the requirements, no further candidate solutions are built, but are traced back to the sudoku puzzle for candidate solutions that meet the requirements. The results of this test are carried out by comparing the normal backtracking algorithm with the hidden single backtracking algorithm when solving the puzzle. sudoku puzzle.

The problem studied in this article is how to use the depth first search algorithm to solve a sudoku game using the Android Studio programming language. The limitations of the problems in this research include the method used in this research using a backtracking algorithm and the Depth First Search (DFS) algorithm. The programming language used is Java and the software used to create the Sudoku game is Android Studio, the Sudoku puzzle used is standard box size 9 x 9.

2. RESEARCH METHOD

2.1. State of the Art

In a previous investigation by [10], it was found that the backtracking algorithm in this case was limited to the number of backtracking processes and calculation of the results obtained. The test program was run with random numbers and 16 different sudoku puzzles of increasing difficulty. This test process was carried out on a PC with AMD Rayzen 5, 8 GB DDR4 RAM and 1000 GB HDD. It can be run in software called Android Studio and uses the Java programming language. The test results and description below show the results of the sudoku design process on 16 different sudoku puzzles with different levels of difficulty. The number of blank squares for each test was set between 43 and 46 blank squares. This test tests two parameters: puzzle completion time in microseconds (ms) and the number of backtracking change sequences. Looking at the test results from the 16 tests run, we can see that the empty squares in a sudoku puzzle is not directly proportional to the increase in time and amount of backtracking. Based on the test results, create a graph according to the parameters tested. In previous research [9], the main objective of this game is to fill the entire space with numbers from 1 to 9. In this research, using the backtracking algorithm method is a refinement of the brute force and exhaustive search algorithm, namely establishing a state-space tree (status space) to find solutions, but candidate solutions are found to be the same, the algorithm becomes inefficient. As candidate solutions are found by the exhaustive search algorithm, the algorithm creates partial solutions from the candidate solutions and evaluates the solutions one by one. If the partial solution built does not meet the requirements, no further candidate solutions are built, but are traced back to the sudoku puzzle for candidate solutions that meet the requirements. The results of this test are carried out by comparing the normal backtracking algorithm with the hidden single backtracking algorithm when solving sudoku.

In this research, apart from using the Backtracking Algorithm, we also used the Depth First Search (DFS) Algorithm, problem solving search algorithm used in Artificial Intelligence. Dynamic Problem Solving is defined as a solution to a problem where the problem situation continues to change and there are many possible solutions. According to H. Lloyd and M. Amos in 2019, the Backtracking Algorithm is a recursive algorithm, and the search process is based on the Depth First Search (DFS) Algorithm. The Depth First Search algorithm aims to get a systematic problem solver for all possible problem solvers, and the search for the answer is done by imitating the arrangement of the root tree.

2.2. Sudoku

One of the popular puzzle games originating from Japan is sudoku. Sudoku consists of 81 squares consisting of 9 rows and 9 rows. The large plot is divided into 9 sub-plots, each consisting of 3 x 3 squares as shown in Figure 2.1. Players can fill in the boxes with numbers 1-9. The condition is that no number can be repeated in a row, nor in a series, and in any partial 3 x 3 grid. As a starting point, several boxes have been filled with opening numbers [4], [5], [8]. Players are welcome to continue. Unlike other number games, sudoku does not train arithmetic intelligence (the ability to add or subtract numbers),

However, sudoku trains players' logical intelligence to solve sudoku by filling in all the boxes without breaking the existing rules. According to Azizah, et al in 2018, sudoku is a digital game that is classified as a puzzle type. Sudoku is generally known as a kind of logic-based number puzzle game where the goal is to fill in the puzzle with numbers without a pattern. Generally, sudoku have sizes 3x3, 4x4, up to 9x9. Sudoku puzzles allocate one number from 1 to 9 for each position in a 9x9 matrix. Numbers may not be repeated in any row or column, and each puzzle has several cells filled with numbers, in nine 3x3 blocks. Sudoku is a popular cognitively stimulating leisure time activity requiring the subject's attention to analyze a grid and fill in numbers, it essentially does not require mathematics but is based on solving logical puzzles. It has long been thought to keep the brain healthy and has been shown to delay the onset of dementia.

Sudoku is a number puzzle game that is popular among people with various levels of difficulty (easy, medium, hard, and very hard). The Sudoku game has a unique principle, and has many combinations of numbers which bring many possibilities to the sudoku game. Entering numbers in this game requires patience and accuracy to fill in the numbers in the sudoku game. Because sudoku involves NP-complete problems, it is difficult or impossible to solve them simultaneously [1], [3], [6]. Sudoku puzzles are puzzles with different difficulty levels such as Easy, Medium, Difficult and Extreme. Execution time and complexity will vary depending on the difficulty. In general, the difficulty of this puzzle depends on the number of clues and the location of the clues.

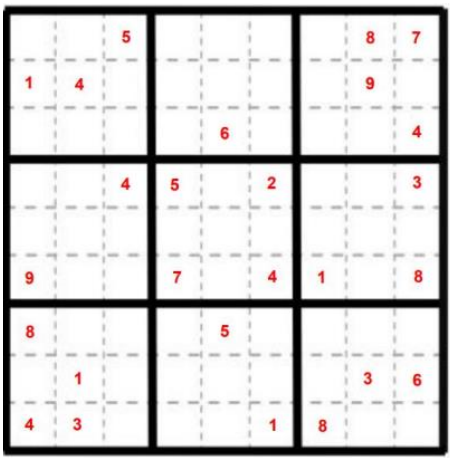


Figure 1. Sudoku Problems

2.3. Dynamic Problem Solving (DPS)

Dynamic Problem Solving is meant as a process of solving problems whose circumstances are constantly changing and have many possible solutions. These changes follow a certain model which can be interpreted as a function of the model. The principle of Dynamic Problem Solving is to analyze all possible ways to provide a solution and find the best way to implement it. In some circumstances, the plan does not simply display or analyze the steps available to consider when determining a solution. Examples of some board games such as chess and checkers, because it is very possible that the problem has many solutions [4], [9]. If all these solutions are checked one by one, the program will not work effectively. The algorithm used in the solver consists of two parts, namely an algorithm that checks whether the question/response the player will enter according to the basic sudoku method or not (Feasible) and an algorithm that solves the question or response obtained (Problem solving).

Solve a sudoku puzzle that each cell is filled with a number value so that three conditions: first, the number value in each cell on the same row must be different; second, the number values in each cell of the same column must be different; finally, the number values in the same minigrid must be different [11]. The resulting application can be used to solve sudoku puzzles. With this app users can solve puzzles with pencil marks, mark cells, or have detailed clues generated about the current state of the puzzle. Additionally, an undo function has been introduced and incorrectly filled cells will be highlighted. Simple cooperative Sudoku Multiplayer was implemented as a proof of concept using Firebase Firestore. A questionnaire designed for use in evaluating applications

2.4. Deep First Search (DFS)

The Depth First Search (DFS) algorithm is a solution search algorithm used in Artificial Intelligence. This algorithm is an uninformed algorithm that lacks information. That is, algorithms that search for specific parts of a sequence, but have no information on which to base their search unless they follow a specific pattern [1], [2]. Depth First Search performs a search in a tree structure. It is a collection of all possible conditions of development in a tree structure. At the top is the root (root) which will contain the initial state of the process (initial state), and below it is the next state up to the target state. Depth First Search is the basis of an algorithm for finding connected elements in undirected graphs, strongly connected elements in directed graphs, and topological sequences in undirected graphs.

Depth First Search (DFS) is a fundamental way to study node-by-node properties of graphs and is widely used in the graph field. To visit a node u in graph G, DFS first marks u as visited, then recursively visits all unmarked neighboring nodes of U. Specifically, the overall order in which DFS visits nodes G is called depth-first order, if DFS starts from a nodes that are connected to all other nodes. Depth-first order or DFS-Tree computing is a key operation for many graph problems, such as finding strongly connected components, topological ordering, reachability queries, etc. This process makes DFS a fundamental operation in the field of graphics. For example, current algorithms for finding strongly connected components (SCCs) must find a DFS-Tree T of G or compute a total depth-first order. For example, the Kosaraju-Sharir algorithm executes DFS to obtain the total depth-first order of G, and executes DFS again on transposed G to find all SCCs of G.

2.5. Backtracking Algorithm

The Backtracking Algorithm is a solution search algorithm based on Depth First Search which can solve a problem. The backtracking algorithm is one way the problem process will be integrated into the procedure for searching the problem solving space, it does not require exploration of all existing possibilities, only those that refer to the existing problem solver. The backtracking algorithm is actually a recursive algorithm and the search process is based on the Depth First Search (DFS) algorithm [2], [5], [7]. The Depth First Search algorithm is designed to systematically search for a problem solver from all possible solutions, and the search for solutions is carried out by analyzing. The structure is in the form of a tree with roots. Handayani, et al.'s research revealed that the use of the DFS algorithm could create an Othello game application for Android-based cellphones. This investigation examines the use of the DFS algorithm in the sudoku game. The process begins by looking at empty plots and then calculating the number of possible candidates, assuming that not every row, box and column has the same number. It can then be applied to the tree for each row according to the DFS algorithm. Tree construction starts from the root node, namely at level 0. The next level is defined as the candidate number that corresponds to the empty box.

Backtracking is a technique for solving problems by building solutions over time by eliminating solutions that do not meet the problem constraints at a certain point in time (based on time, here, called the elapsed time until it reaches any level of the search tree) [12], [13]. In this backtracking technique we use recursive calls to find the solution by building the solution step by step. In each step, find all possible solutions to the problem and delete solutions that cannot lead to a solution to the problem based on the problem constraints. Backtracking algorithm is a solution search method for problem solving based on state space search. This algorithm works recursively to find a solution to a problem with several possible solutions. This algorithm is based on the Depth First Search (DFS) algorithm to find a more efficient problem solver. The Backtracking algorithm itself is a typical form of a recursive algorithm and is based on the Depth First Search (DFS) algorithm. This recursive algorithm tries several alternatives until it finds one that can be executed and makes a decision. With this method, there is no need to continue (cut) the possibility of finding a dead end or getting stuck in a decision, so the time to find a solution can be shortened [14], [15]. Backtracking algorithms are often used when solving problems in games and artificial intelligence. An example of a tree backtracking algorithm is as follows in Figure 2.

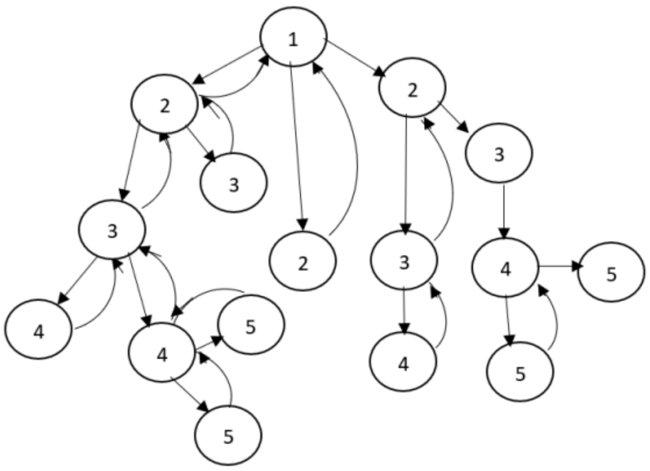


Figure 2. Tree of Backtrackign Algorithm [5]

Based on Figure 2, one of the first branches is visited up to the end node (1 to 4). If the desired target is not found, the next process continues the search in one of the previous branches (2-3) and goes down to another branch. Continue until the final target (1-5) is achieved. In this backtracking method, the search process does not lead to all possible problem solvers, but only one. When solving a problem, backtracking algorithms have many options and can find several different solutions. During the problem solved process, the backtracking algorithm progresses from one opportunity to the next. At the end of the sudoku game, the backtracking algorithm finds the appropriate number of solutions to fill the empty squares of the sudoku. The backtracking algorithm searches until it finds a solution. If no candidate node is found, the algorithm returns and looks for other possible candidate nodes.

2.6. Research instrumentation

In this analysis, the sudoku program process and data analysis performed on 16 different sudoku puzzles with varying random numbers and levels of difficulty. This test was carried out on a PC device with details LAPTOP-SMRA81UG, Windows 10 Home Single Language 64-bit, Acer, Nitro AN515-57, Processor: 11th Gen Intel(R) Core (TM) i5-11400 @2.70Ghz, Memory: 8192MB RAM, Page file: 10357 used, 6301MB available, DirectX Version: DirectX 12. This test was run in software called Android Studio and uses the Java programming language. To support research, various tools and materials are needed.

Tools and materials include: Android Studio Software, USB Cable, Laptop (LAPTOP-SMRA81UG, Windows 10 Home Single Language 64-bit, Acer, Nitro AN515-57, Processor: 11th Gen Intel® Core™ i5-1140 @2.70Ghz, Memory: 8192MB RAM, Page file: 10357 used, 6301MB available, DirectX Version: DirectX 12), and cellphone to test the results.

This research is a case study, with the data used being the average time of playing Sudoku with each level of difficulty. The method used is the Backtracking Algorithm method. The following is the Backtracking algorithm flow as shown in Figure 1.

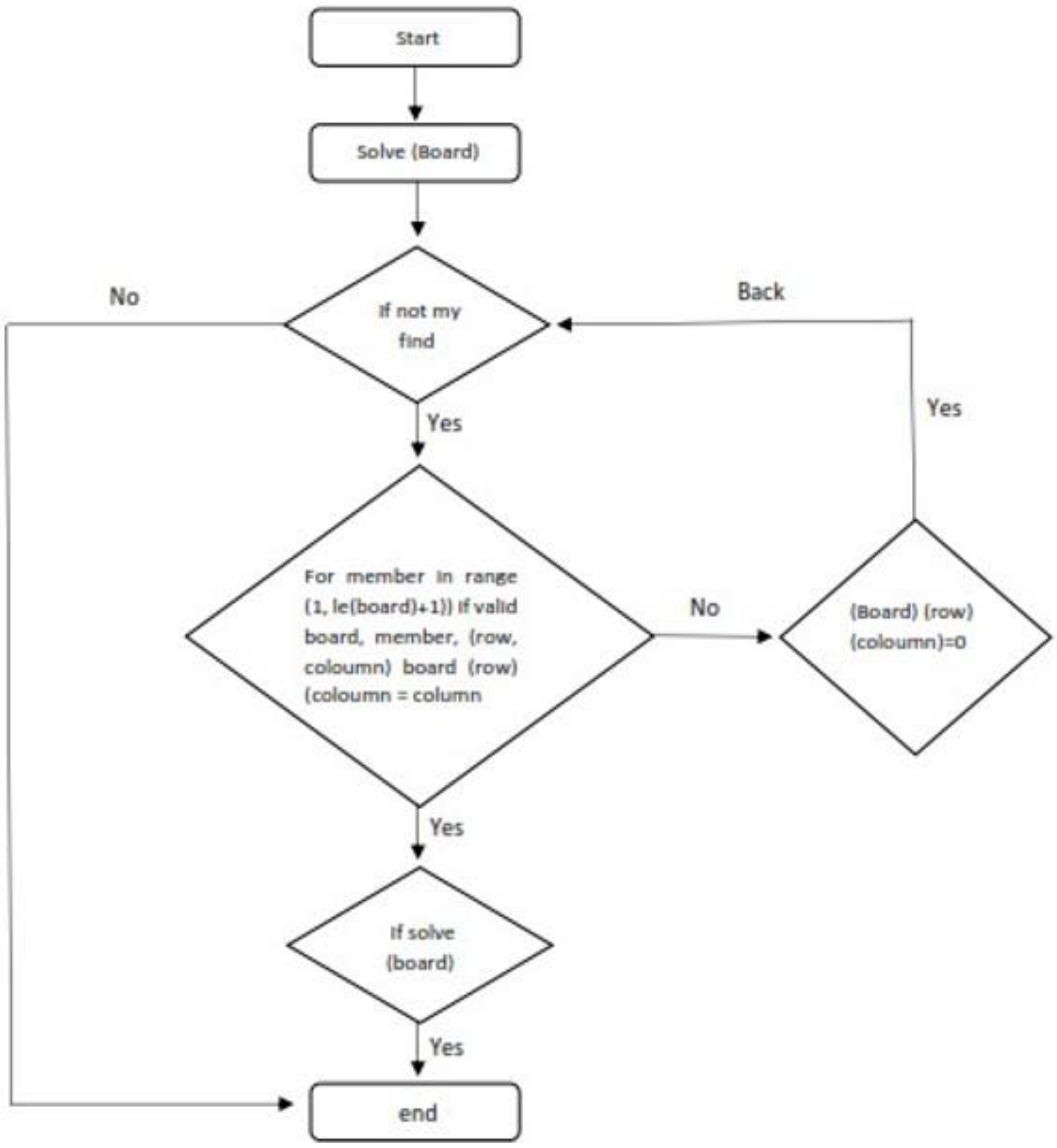

Start
Solve (Board)
No
If not my find
Back
Yes
Yes
For member in range (1, le(board)+1)) if valid board, member, (row, coloumn) board (row) (coloumn = column
No
(Board) (row) (coloumn)=0
Yes
If solve (board)
Yes
end

Figure 3. Backtracking scheme

Stages of the Sudoku game Backtracking Algorithm Flowchart as in Figure 3 as follow:

1) First enter the solve function in the board parameters.

2) In the next step, declare My_find using the find_empty (board) function, a branch is obtained if the condition is not my_find. until the program is complete or it can be said that the sudoku is complete. The find_empty function is used to check whether the column and row are empty or not.
3) The next step, else branching above is row and column =my_find, then the search is carried out using the algorithm for number in range (1, len(board) + 1): if valid (board, number, (row, column)): board[row][coloumn] = number in this algorithm is a valid function that is useful for checking the coincidence of the number of possible solutions that will be given by the solve function.
4) In the next step there is an intersection and solving the number obtained from the algorithm above solve returns true, this means that the number is the solution. Otherwise, return false and go back to looking for the troubleshooter number.

The Depth First Search (DFS) algorithm is implemented as a function in a Layout Resource File in the activity_game.xml file called "submit". The Backtracking algorithm is implemented into a description symbol that will show whether the sudoku game has been completed or not. The sudoku generation process is implemented in the Java file folder and layout file folder functions. Initially, this function will fill in all the empty sudoku boxes and remove some sudoku boxes to be used as questions according to the level of difficulty. Since this function, it can be seen that the sudoku generation process begins by creating questions that fill all the empty gaps based on the results of the Depth First Search Algorithm. To differentiate the number of empty boxes from questions, the quota variable is populated based on the level selected by the user. The DFS function is called again to generate random numbers according to probability. Completion is done by filling in manually according to the user's ability, you can also use the help button to make a choice of solution numbers for the problem, or use the answer key button.

3. RESULTS AND DISCUSSION

The Depth First Search algorithm is implemented into a function in a Layout Resource File in the activity_game.xml file called "submit". The Submit parameter function is to check whether the sudoku game that has been played has been completed or not, if there are still incorrect numbers then you must replace the answer with a more correct one. The Backtracking algorithm is implemented into a description symbol that will show whether the sudoku game has been completed or not. If it is not completed or is incorrect then the description "Wrong answer" will appear and if the response is correct or completed then the statement "Congratulations, your answer is correct" will appear. The data required in this research is the result of data collection from the average trial time of respondents with various levels of difficulty, which can be seen from Table 1 below.

The sudoku generation process is implemented in the Java file folder and layout file folder functions. Initially, this function will fill in all the empty sudoku boxes and remove some sudoku boxes to be used as questions according to the level of difficulty. Since this function, it can be seen that the sudoku generation process begins by creating questions that fill all the empty gaps based on the results of the Depth First Search Algorithm. To differentiate the number of empty boxes from questions, the quota variable is populated based on the level selected by the user. The DFS function is called again to generate random numbers according to probability. In this research, the solution is carried out by filling in manually according to the user's abilities, you can also use the help button to make a choice of solution numbers for the problem, or use the answer key button. Figure 2 is implementation of the design into the Sudo-KU application.

Table 1. Sudoku Level Easy Trial Results

No	Umur	Percobaan	Posisi Kotak Soal	Waktu (menit)
1	21 Tahun	Mellina Sekararum	2,5,6,10,12,19,24,26,28,30,36,38,39,40,41,44,46,47,50,51,53,55,56,60,65,66,67,69,72,79	20:35 menit
2	21 Tahun	Dicandra Filliya Tareniya	1,5,6,8,13,16,17,20,23,27,31,35,37,38,39,50,55,60,63,65,66,70,72,76,78,79,80,81	09:32 menit
3	22 Tahun	Reza Qoidah Putri	5,11,12,15,16,19,20,21,28,29,37,39,41,43,44,45,48,49,52,56,59,60,71,72,73,76,77,78,79,80	20:03 menit
4	21 Tahun	Dhea Suci Amalia	1,4,5,6,8,10,11,14,15,19,20,26,27,29,34,36,39,40,48,49,51,54,57,58,63,65,67,69,72,78	22:46 menit
5	21 Tahun	Ailsa Frederica	1,5,6,7,8,12,14,15,16,20,29,31,33,34,35,36,38,43,45,49,50,52,58,61,67,73,74,76,78,80	13:52 menit
6	21 Tahun	Karina Febriani	1,2,4,5,6,7,8,10,11,12,25,29,33,37,44,49,55,57,58,60,64,67,69,71,77,80	05:20 menit
7	21 Tahun	Anisatul Millah	6,7,10,12,13,17,19,20,29,30,33,34,35,36,37,40,41,42,43,47,49,50,56,61,64,67,68,72,77,78	11:03 menit
8	21 Tahun	Tri Setio Utami	2,9,10,11,12,13,14,21,22,23,25,32,33,36,38,41,42,45,46,47,53,54,55,56,60,65,67,70,71,78	01:28 menit
9	18 Tahun	Muhammad Naufal Ammar	1,2,5,6,7,8,13,14,17,22,27,29,31,32,34,35,43,48,49,50,56,60,62,63,64,66,67,70,71,78	03:02 menit
10	19 Tahun	Rhifky Arhifkha	1,2,3,4,5,6,8,9,17,24,27,28,32,37,44,46,47,50,52,53,54,60,65,67,68,70,77,80	03:04 menit

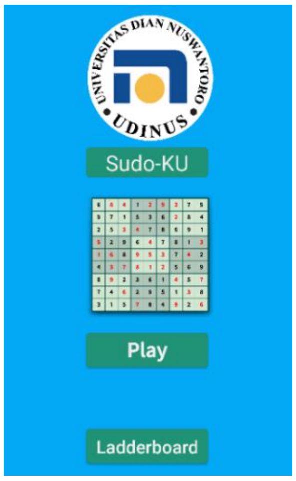


Figure 2. Main Menu Sudoku

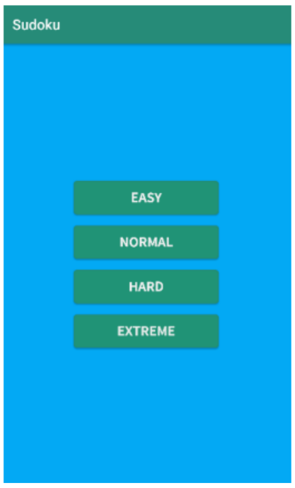
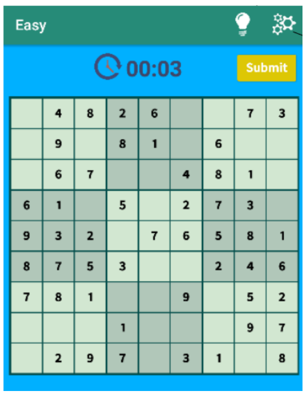
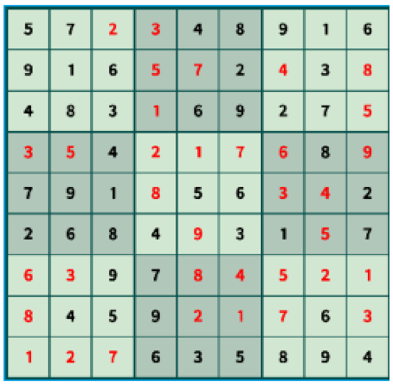


Figure 3. Sudoku Leveling



	4	8	2	6			7	3
	9		8	1		6		
	6	7			4	8	1	
6	1		5		2	7	3	
9	3	2		7	6	5	8	1
8	7	5	3			2	4	6
7	8	1			9		5	2
			1				9	7
	2	9	7		3	1		8

Figure 4. Sample Sudoku in Easy Level

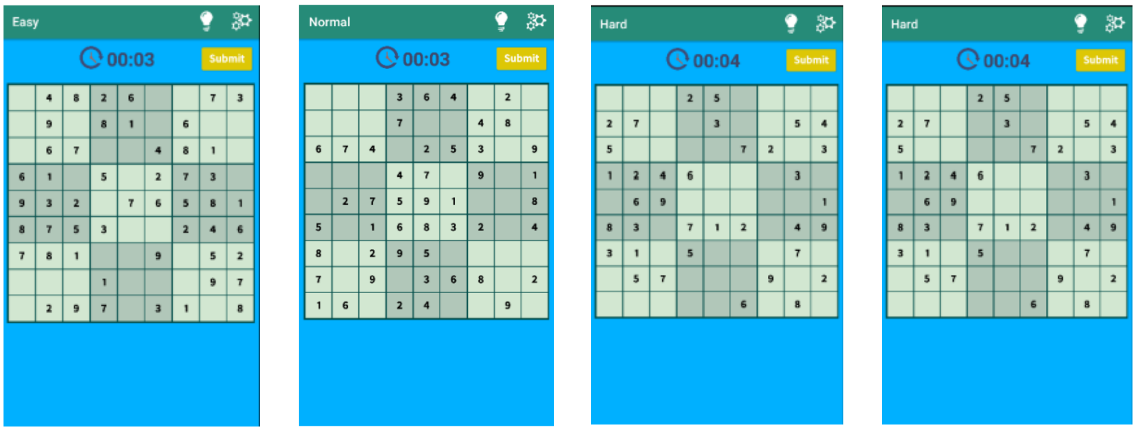


5	7	2	3	4	8	9	1	6
9	1	6	5	7	2	4	3	8
4	8	3	1	6	9	2	7	5
3	5	4	2	1	7	6	8	9
7	9	1	8	5	6	3	4	2
2	6	8	4	9	3	1	5	7
6	3	9	7	8	4	5	2	1
8	4	5	9	2	1	7	6	3
1	2	7	6	3	5	8	9	4

Figure 5. Example of solving sudoku at easy level

The Backtracking algorithm, which is an aid to disbursing numbers by implementing root tree numbers, gives rise to various number choices which are then chosen by the user as the most correct one. Meanwhile, the settings button functions to open all the correct answers to sudoku, this settings button is an application of the Depth First Search Algorithm, which is the fastest solution search algorithm, so in this sudoku game the answers are in an empty box. If you click on this settings button, all the answers will automatically come out. with red number symbols. According to the results obtained, system design, and testing on students, it was concluded that the Depth First Search Algorithm and Backtracking Algorithm could be realized in the sudoku game. The Backtracking Algorithm and Depth First Search Algorithm are applied to generate the start and finisher in the sudoku game, so you can get numbers where there are no numbers that are the same in rows, columns and minigrids. At easy, normal, hard and extreme levels with test results by users with average values.

Implementation of the Depth First Search Algorithm in the Sudo-KU game produces solutions at 4 levels of difficulty, namely easy level, normal level, hard level and extreme level with an average time at the easy level of 11:04 minutes, at the normal level of 10 :52 minutes, at hard level for 25:46 minutes, and at extreme level 38 minutes. Evaluation of the use of questionnaires.



(a) (b) (c) (d)

Figure 6. Proposed Level Sudoku: (a) easy, (b) normal, (c) hard, (d) extreme

Average time to complete a sudoku game according to the number of numbers will be submitted. This experiment was carried out 10 times at each given level with the help of Microsoft Excel software. Easy difficulty trials were then averaged, giving a total duration of 664 seconds. The average number of attempts on Normal difficulty was 625 seconds. The average number of attempts on Hard difficulty was 1546 seconds. The average number of attempts on Extreme difficulty was 625 seconds. The sudoku application test was carried out for student respondents in the city of Semarang in February 2023. There were 10 respondents who determined the user's assessment of the suitability and usefulness of the sudoku application as shown in Table 1, which is easy level testing. Here, we used likert to esure testing performance using Table 2.

Table 2. Leveling Likert Scale

Level	Weight
Strongly agree	5
Agree	4
Simply Agree	3
Don't agree	2
Strongly Disagree	1

Based on Table 2 and Table 3, it was found that, out of a total of 10 respondents, from the first question about knowing the use of each menu on the main page, there were 0 people who strongly disagreed, 1 person disagreed, 2 people quite agreed, 5 people agreed. people, and 2 people strongly agree. For questions regarding explanations from the help menu, there were 0 people who strongly disagreed, 0 people disagreed, 2 people quite agreed, 2 people agreed, and 6 people strongly agreed. Then there were 1 respondent who knew how to fill in the blank boxes in Sudoku, who strongly disagreed, 1 person disagreed, 2 people quite agreed, 2 people agreed, and 4 people strongly agreed. Then respondents who know how to use the menu complete if they are unable to fill in every empty box in sudoku, there are 0 people who strongly disagree, 1 person disagrees, 2 people quite agree, 5 people agree, and 2 people strongly agree. For the question of players feeling bored playing the sudoku game, there were 2 people who strongly disagreed, 2 people disagreed, 3 people quite agreed, 2 people agreed, and 1 person strongly agreed. Respondents who felt the sudoku game was fun and interesting, 0 people strongly disagreed, 1 person disagreed, 2 people quite agreed, 6 people agreed, and 1 person strongly agreed. For the question, differences in difficulty levels have an influence on completing the sudoku game, 1 person strongly disagrees, 0 people disagree, 4 people quite agree, 1 person agrees, and 4 people strongly agree. Respondents who felt that playing Sudoku can sharpen your brain is like learning mathematical calculations. There were 0 people who strongly disagreed, 2 people disagreed, 2 people quite agreed, 3 people agreed, and 3 people strongly agreed. Then to design the appearance of the sudoku game, it is interesting to play, there are 0 people who strongly disagree, 1 person disagrees, 3 people quite agree, 6 people agree, and 0 people strongly agree. The help button can help solve the sudoku game, there are 0 people who strongly disagree, 0 people disagree, 3 people quite agree, 3 people agree, and 4 people strongly agree.

Table 3. Testing Questions

Questions	Result				
	Strongly agree	Agree	Simply Agree	Don't agree	Strongly Disagree
Players know how to use each menu on the main page?	-	1	2	5	2
Players understand the explanation from the help menu?	-	-	2	2	6
Players know how to fill in the blanks in sudoku?	1	1	2	2	4
Players know how to use the solve menu if they are unable to fill in every empty box in sudoku?	-	1	2	5	2
Players feel bored playing sudoku games?	2	2	3	2	1
Sudoku games are fun and interesting?	-	1	2	6	1
How does the difference in difficulty level affect sudoku games?	1	-	4	1	4
Sudoku games can sharpen your brain like learning mathematical calculations?	-	2	2	3	3
The appearance design of the sudoku game is interesting to play?	-	1	3	6	-
Help button can help solve sudoku game?	-	-	3	3	4

4. CONCLUSION

According to the results obtained, system design, and testing on students, it was concluded that the Depth First Search algorithm and the Backtracking Algorithm could be realized in the sudoku game. The Backtracking Algorithm and Depth First Search Algorithm are applied to generate the start and finisher in the sudoku game, so you can get numbers where there are no numbers that are the same in rows, columns and minigrids. At easy, normal, hard and extreme levels with test results by users with average values. Implementation of the Depth First Search Algorithm in the Sudo-KU game produces solutions at 4 levels of difficulty, namely easy level, normal level, hard level and extreme level with an average time at the easy level for

11:04 minutes, at the normal level for 10:52 minutes, at the hard level for 25:46 minutes, and at the extreme level 38 minutes. Evaluation of the use of questionnaires.

During the creation of this Android-based sudoku game application, of course there were still many gaps in game application research that it is hoped that the next Sudo-KU can overcome. Therefore, the author recommends several things for further development of the material, including:

1. Develop a sudoku application that is more attractive in terms of appearance and playability, especially as a learning tool.
2. Development of sudoku applications for other mobile operating systems such as Windows Phone, BlackBerry, iOS and others.
3. Develop a sudoku application that goes beyond the 9x9 grid.
4. Developing a sudoku application with different modes, for example Sudoku-MU (Kross Sudoku), where the crossed numbers cannot be the same.
5. Develop a sudoku application where the contents of each grid do not always have to be numbers, but can be developed into letters or images.

REFERENCES

[1] Herimanto, P. Sitorus, and E. M. Zamzami, “An Implementation of Backtracking Algorithm for Solving A Sudoku-Puzzle Based on Android,” *J Phys Conf Ser*, vol. 1566, no. 1, p. 012038, Jun. 2020, doi: 10.1088/1742-6596/1566/1/012038.

[2] H. Lloyd and M. Amos, “Solving Sudoku with Ant Colony Optimization,” *IEEE Trans Games*, vol. 12, no. 3, pp. 302–311, Sep. 2020, doi: 10.1109/TG.2019.2942773.

[3] M. Praba, S. Radha, Priyam P. M., and B. S. Dhiya, “Sudoku Solver Using Minigrid Based Backtracking Algorithm,” *International Journal of Research in Engineering, Science and Management*, vol. 5, no. 6, pp. 138–140, 2022.

[4] N. Kitsuwan, P. Pavarangkoon, H. M. Widiyanto, and E. Oki, “Dynamic load balancing with learning model for Sudoku solving system,” *Digital Communications and Networks*, vol. 6, no. 1, pp. 108–114, Feb. 2020, doi: 10.1016/j.dcan.2019.03.002.

[5] N. A. Hasanah, L. Atikah, D. Herumurti, and A. A. Yunanto, “A comparative study: Ant colony optimization algorithm and backtracking algorithm for sudoku game,” in *Proceedings - 2020 International Seminar on Application for Technology of Information and Communication: IT Challenges for Sustainability, Scalability, and Security in the Age of Digital Disruption, iSemantic 2020*, Institute of Electrical and Electronics Engineers Inc., Sep. 2020, pp. 548–553. doi: 10.1109/iSemantic50169.2020.9234267.

[6] A. Z. Sevkli and K. A. Hamza, “General variable neighborhood search for solving Sudoku puzzles: unfiltered and filtered models,” *Soft comput*, vol. 23, no. 15, pp. 6585–6601, Aug. 2019, doi: 10.1007/s00500-018-3307-6.

[7] R. Effendi, I. Gunawan, and Y. Efendi, “Software Design Completion of Sudoku Game with Branch and Bound Algorithm,” in *International Multidisciplinary Conference on Education, Technology, and Engineering (IMCETE 2019)*, 2019, pp. 126–130.

[8] C. M. Rodríguez-Peña, J. Ramón Martínez Batlle, and W. M. Maurer, “Correlation and Regression Analyses using Sudoku Grids,” *International Journal of Science and Research*, 2018, doi: 10.21275/ART2020922.

[9] T. N. Lina and M. S. Rumetna, “Comparison Analysis of Breadth First Search and Depth Limited Search Algorithms in Sudoku Game,” *Bulletin of Computer Science and Electrical Engineering*, vol. 2, no. 2, pp. 74–83, Dec. 2021, doi: 10.25008/bcsee.v2i2.1146.

[10] B. Vicky Indriyono, N. Pamungkas, Z. Pratama, E. Mintorini, I. Dimentieva, and P. Mellati, “Comparative Analysis of the Performance Testing Results of the Backtracking and

Genetics Algorithm in Solving Sudoku Games," *International Journal of Artificial Intelligence & Robotics (IJAIR)*, vol. 5, no. 1, pp. 29–35, 2023, doi: 10.25139/ijair.v5i1.6501.
[11] H. R. R. Zaman and F. S. Gharehchopogh, "An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization problems," *Eng Comput*, vol. 38, pp. 2797–2831, Oct. 2022, doi: 10.1007/s00366-021-01431-6.
[12] Nurdin *et al.*, "The Implementation of Backtracking Algorithm on Crossword Puzzle Games Based on Android," in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Nov. 2019. doi: 10.1088/1742-6596/1363/1/012075.
[13] A. Candra, M. A. Budiman, and R. I. Pohan, "Application of A-Star Algorithm on Pathfinding Game," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jun. 2021. doi: 10.1088/1742-6596/1898/1/012047.
[14] H. R. R. Zaman and F. S. Gharehchopogh, "An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization problems," *Eng Comput*, vol. 38, pp. 2797–2831, Oct. 2022, doi: 10.1007/s00366-021-01431-6.
[15] A. Agnesina, K. Chang, and S. K. Lim, "VLSI Placement Parameter Optimization using Deep Reinforcement Learning," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, Institute of Electrical and Electronics Engineers Inc., Nov. 2020. doi: 10.1145/3400302.3415690.