International Journal of Artificial Intelligence & Robotics (IJAIR) E - ISSN : 2686-6269
Vol.6, No.1, 2024, pp.40-47

Analysis of the Concept of Solving the Sudoku Game Using Backtracking and Brute Force Algorithms

Bonifacius Vicky Indriyono[1]*, Ratna Wardani[2],Titin Susanti[3], Laili Wulandari[4], Moh. Fathurrohim[5], Rifka Sari Pratiwi[6], Endah Wulan Safitri[7]
[1,2,3,4,5,6,7]Institut Ilmu Kesehatan (IIK) STRADA Indonesia, Kediri, Indonesia
[1]bonifaciusv @strada.ac.id*; [2]ratnawardani61278@gmail.com; [3]titin@strada.ac.id; [4]lailiwulandari86@gmail.com; [5]fathur@strada.ac.id; [6]rifkas@strada.ac.id ; [7]endahw@strada.ac.id
*corresponding author

ABSTRACT

Sudoku is a puzzle game that arranges numbers from one to nine on a board consisting of 9x9 squares. This game has a rule: the numbers arranged on a board cannot be the same in each row or column. The main goal of this Sudoku game is to complete the board and win by entering numbers into each empty box. Therefore, in this Sudoku game, there is only one valid solution. At the start of the game, several numbers are provided, which serve as clues for players to fill in the next box. The number of clues at the start of the game determines whether the Sudoku game level is difficult. The fewer the clues given, the more difficult it is to solve Sudoku. Solving Sudoku problems cannot be done manually, so several algorithms are needed, which require a lot of repetition and search functions. In this research, we will compare concepts in solving Sudoku using the Backtracking algorithm and the backtracking algorithm using the brute force technique. This comparison is seen from the effectiveness of solving Sudoku. The results of this research conclude that to solve Sudoku logic quickly and efficiently, the backtracking algorithm can be used compared to the Brute Force algorithm. This is because the backtracking algorithm has better complexity, and the solution steps are fast and "smart". In contrast, the brute algorithm takes longer to complete because the brute force algorithm has to search for all the possibilities. The average time required to complete 10 9 x 9 puzzle tests with backtracking is 10910.96 ms, while the brute force algorithm takes longer, around 15871.90 ms.

Keywords: Puzzles; Sudoku; Brute Force; Backtracking; Algorithm.

Article History

Received : June, 05th 2024	Accepted : June, 22nd 2024	Published : July, 09th 2024

I. INTRODUCTION

Puzzle games are games that demand reasoning and logic to solve. There are various types of puzzles, including the Sudoku game. Sudoku is a world-famous number puzzle game. Sudoku is a grid of 81 squares organized in 9 rows and 9 columns. The Sudoku grid itself contains a 3x3 subgrid. In Sudoku, players must fill each box with one number, ranging from 1 to 9, so each box has a unique number for each row, column, and subgrid on the board. The number of clues on the grid at the beginning of the Sudoku game determines its difficulty level. There are fewer clues on the grid.

Sudoku offers advantages, one of which is that it can help strengthen logic-based reasoning and thinking skills. Sudoku is a 9 x 9 grid divided into 3 x 3 squares known as subgrids, with numbers ranging from 1 to 9. Sudoku Puzzles asks players to employ their brain power and concentration in problem-solving tactics. [1]. The Sudoku game is in great demand because it trains the brain in terms of logic. The mission of this game is to fill in the numbers from 1 to 9 into a 9x9 net consisting of 9 3x3 boxes without any numbers repeating themselves in a row, column, or box. Sudoku encourages players to think using logic carefully. When filling in numbers, high accuracy is required because if you enter numbers carelessly, the game cannot be finished. The Sudoku game has a unique principle and creates many number combinations, which makes the Sudoku game produce many possibilities. Filling in the numbers in this Sudoku game takes patience and sharp accuracy.

Based on the challenge described above, solving the Sudoku game requires using an artificial intelligence idea and the appropriate algorithm. Backtracking and Brute Force are two algorithms that can be used. This backtracking algorithm searches for numbers that can only lead to a solution or resolution. The answer can be obtained with a brief and efficient search sequence because it does not check all possible solution numbers. [2]. At the same time, the Brute Force algorithm is an approach based on a problem statement and a definition of the concepts involved. Brute-force algorithms solve problems very simply, directly, and clearly [3].

Previous studies that explored problems relating to the application of Backtracking and Brute Force algorithms included research from [4], which discussed the application of backtracking methods in the Sudoku game. This research concludes that the Backtracking algorithm can help solve the Sudoku game well. The difference between current and current research is the method used to solve it. Furthermore, research by [5] reviews the creation of applications based on the Backtracking algorithm in the Sudoku game. This research concludes that the Backtracking algorithm-based application can quickly solve problems, with an

DOI: http://dx.doi.org/10.25139/ijair.v6i1.8377

average problem-solving time of 0.0880295 seconds from 20 sudoku questions. The difference between the current research and the final goal is that if the previous research aimed to test the average time, then this research aims to analyze the efficiency of the steps of the backtracking and brute force algorithms. According to [6], comparing the performance testing results of the backtracking algorithm to the genetic algorithm revealed that the backtracking algorithm solved the Sudoku game optimally in the 9 x 9 format. The current research differs from the approach being compared in it [7] and examines brute force and backtracking strategies for solving Sudoku. Then, [8] demonstrates how to solve Android-based Sudoku puzzles with a backtracking technique. [9] describes how to complete the Su Doku 9x9 Pattern via backtracking. The type of Sudoku and the solving technique employed distinguish the two studies mentioned above from the current research. Research from [10] discusses how evolutionary algorithms can help overcome Sudoku puzzle logic challenges.

This research aims to examine and analyze which algorithm is more appropriate and more practical in terms of the steps in solving the Sudoku game. The test results show that the Backtracking algorithm is more effective than the Brute Force algorithm in solving the Sudoku game. The Backtracking algorithm has better complexity because the solution search process is carried out more "smartly" by not using many repetition steps compared to the brute force algorithm, which searches for all existing possibilities and has many repetition steps. It takes a long time to find a solution.

II. METHOD

A. Method of collecting data

Data collection methods are one aspect of searching for the data needed for research. Data collection in this research took the form of a literature study. A literature study collects data through search surveys on existing documents, journals, and proceedings regarding the research topic. Book sources can also serve as a roadmap for the system development process that will be implemented due to the completed research.

B. Data Analysis Method

All data collected before and during the research will be processed and organized as neatly as possible. Several stages in this analysis method include :

1) Data editing: This process is used to organize the document as a whole after obtaining the results of previous research.

2) Coding. This process is carried out to develop logic according to the researcher's wants. This process is also used to double-check the algorithms used so that the final results follow the initial objectives of the application being built.

3) Tabulation. This process compiles data from the initial study of the system, which is then presented in tabular form to make it easier to understand.

C. Backtracking Algorithm

The Depth First Search (DFS) technique is the foundation for the dynamic solution search process known as the backtracking algorithm. This algorithm operates by following potential numbers that only result in a solution. A brief and effective search sequence is used to find a solution, as it does not examine every alternative. [2]. Fig.1 the search process of the backtracking algorithm [11].

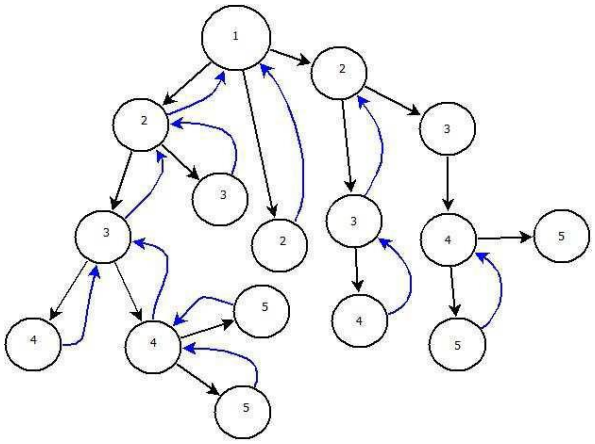

Fig 1. Searching with Backtracking

The above retracing picture illustrates how the search phase starts by going from one of the points to the last point, 1 to 4. If you haven't arrived at your chosen location, carry on looking at one of the earlier points, two to three. If it is felt that there are still other points below it, then the process will go down. This process is repeated until the desired outcome (points 1 through 5) is achieved. This backtracking goes directly to a single answer—it does not lead to every potential option. This is the benefit of

backtracking because it is efficient, effective, and takes less time to search. Pseudocode (1) is the algorithm of the backtracking [6]. The steps for tracing using backtracking are [12]:

1) Search using this backtracking method takes the form of a path from root to leaf: This search produces a live node node. This live node can be expanded further and is known as an expanded node.

2) After the search is conducted, it appears that the expansion node creates an unproductive path: The expansion node eventually becomes a "dead node," meaning it can no longer be expanded.

3) If the last location is in a dead node, the tracing process generates multiple more child nodes; however, if the child node is absent, a backtracking procedure is initiated to reach the parent node.

4) The living nodes found: This search process will be a sign of the end of the search process.

Pseudocode (1). Backtracking Algorithm

```
Backtracking procedure (input k:integer)
{
   // This is merely a comment: searching for every answer using the backtracking
approach entry: k, which is the component vector n[k]'s index
   print: resolution

   X=([1].[2],...,[m])
Backtrack_Algorithm: if (n[1],n[2],...n[k]) is the path from root to leaf then
PrintSolution(n) end; else, for each untested n[k] such that (n[k]=T(k)) and
B(n[1].n[2],...n[k])=true doIf BacktrackR(k+1) Enddforeach, then
```

D. Brute Force Algorithm

Brute force is a direct approach to solving a problem. The working principle of brute force is usually based on a problem statement and the definition of its concepts. This algorithm works in a simple way to solve problems and in a clear way [14]. The Brute Force algorithm is a technique for matching words or strings in text with patterns in each character from left to right. The Brute Force algorithm uses techniques when you want something you are looking for to optimize the time to get it. Apart from that, this algorithm works by sorting each solution in a structured manner one by one and saving them to find the best solution [15].

A simple example of the Brute Force algorithm is matching a string to a pattern. An example of the "D A Y A" string with 4 characters is given. Then, the pattern given is "*A Y A*" with a length of 3 characters. The first step is to match the string with the pattern. The search results for the first step are shown in Table 1, where there is no match between the string and the pattern, namely characters D and A, so the search continues with a shift to the right. This process finds matches between strings and patterns, namely characters A and A in Tables I and II.

TABLE I
THE FIRST STEP OF CHARACTER MATCHING

D	A	Y	A
A	Y	A	
0	1	2	3

TABLE II
STEP TWO OF CHARACTER MATCHING

	A	Y	A
	A	Y	
0	1	2	3

E. Puzzle Definition

Puzzles are often interpreted as a game in the form of a collection of pieces of images that can be arranged in such a way as to form one complete image. Puzzles are games categorized as an image-learning medium when seen from the perspective of shape. This puzzle can be played by adults and children alike. Puzzle games help players improve their memory and teach them to be cautious. You can play this puzzle game anywhere—at home, school, or other settings where kids play. [16].

A puzzle is a game made up of several image components with varying degrees of complexity that may be changed to fit the growing child's needs. Because players must concentrate on arranging the puzzle pieces until they form a full picture, this game demands precision and develops mental attention. [17].

F. Sudoku Game

Sudoku is a mathematical logic game because you must think like a mathematician to solve it by looking for patterns and careful logic [18]. Players cannot play carelessly because if they play carelessly, the game cannot be finished.

Sudoku is categorized as a logic puzzle game in the form of an n x n matrix, where the parts consist of rows, columns, and blocks. Sudoku has game rules: filling in the numbers in all the boxes based on the clue numbers provided. Each row, column, and block must be filled with the numbers 1 to 9, which add up to one each or do not have the same number [19].

III. RESULT AND DISCUSSION

This study examines brute force methods and backtracking in the Sudoku puzzle game, employing a 9x9 square puzzle. The research stages describe the sequence of this research until the final testing and implementation stage. Fig. 2 shows the stages of this research. From searching for relevant literature, designing the backtracking and brute force algorithms in solving Sudoku, system testing, evaluation, implementation in the Sudoku game, drawing conclusions, and documenting.

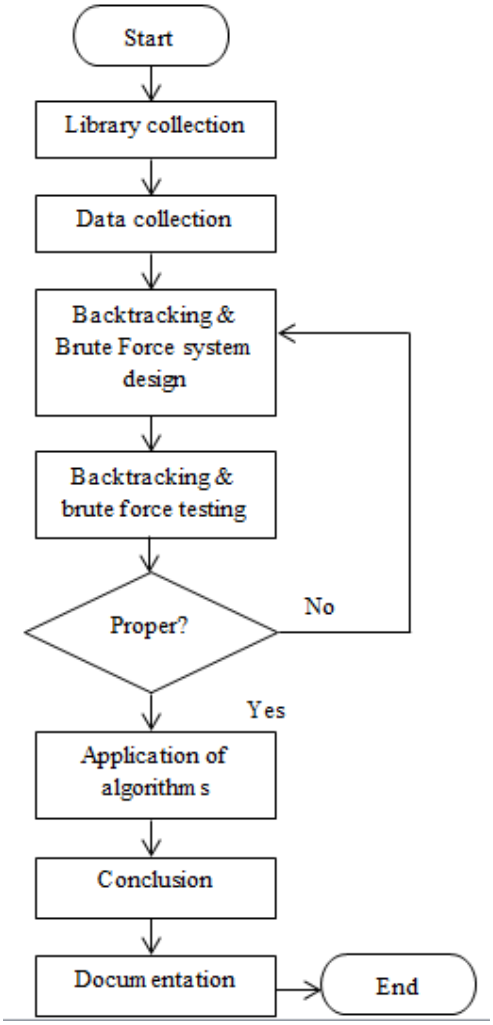

Fig 2. Research Stages

A. *Procedure for Solving Sudoku Using Backtracking Algorithm*

In general, there are several stages that players must complete to complete this Sudoku game, including:

1) Perform identification: This process is a step where players must identify which rows contain certain numbers. Players must repeat this process in each column or row repeatedly and systematically.

2) Logic analysis: Participants mark potential numbers that fit inside a box in this step of the game.

3) Removal of several candidates: To move the process along, this procedure consists of progressively removing candidate numbers until a box contains just one contender.

The use of this backtracking strategy in the Sudoku game involves multiple steps, such as:

a) The search process begins in the first row and column's matrix elements.
b) All possible numbers that the row can have are checked.
c) If a valid number is found, proceed with checking the next element of the matrix.
d) If a valid number is not found, it is traced back to the previous matrix element.
e) The process stops if a solution has been found or if there is no possibility of a solution.

DOI: http://dx.doi.org/10.25139/ijair.v6i1.8377

The pseudocode (2) of the backtracking algorithm [6].

Pseudocode (2). The Pseudocode of Solving Sudoku Algorithm

```
Procedure SolvingSudoku(entryarray:matriks)
Solution : boolean, i,j:integer
Algorithm1.solution=false;
I=1; J=1;
While (!solution&&i>0)
{ Array[i][j].SetValues(array[i][j].GetValue()+1);}
While (array[i][j].GetValue()<=9&&!place(i,j))
{ array [i][j].SetValue(array[i][j].GetValue()+1); }
If (array[i][j].GetValue()<=max)
{ If(i==9&&j==9) { Solution=true }
Else { j=j+1; if (j>9) { i=i+1; j=1;}}}
else array[i][j].
SetValue(0);
 j- -;
 if (j==0) {j=column i=i-1;}}}
```

Fig.3 is an example of a 9 x 9 Sudoku game and the solution to the problem. For example, in Fig. 3, in this Sudoku game, players must fill empty boxes with certain numbers, provided the same numbers cannot be in either row or column. The solution to the case in Fig. 4.

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

Fig 3. Example of Sudoku 9 x 9

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

Fig 4. Solving Sudoku 9 x 9

In Fig. 4, you won't find the same input numbers in each row and column or 3 x 3 block. Based on pictures 2 and 3, an example of a Sudoku case with a size of 9 x 9, a backtracking algorithm was tested using test data from 10 Sudoku puzzles with different difficulty levels. The data in Table III shows Sudoku testing results using 10 9 x 9 puzzles. This test is based on completion time and the number of backtracking processes. Based on the test results, it can be concluded that if there are many empty boxes in the puzzle, not all are directly proportional to the increase in time and the amount of backtracking. Fig. 5 is a graph from the backtracking test according to the data in Table III.

TABLE III
TESTING WITH BACKTRACKING

Process	EMPTY BOXES	Times Required	Number of Processes Backtracking
1	43	62320	552
2	43	70865	815
3	43	64201	712
4	43	123402	1265
5	44	71086	755
6	44	120961	1204

DOI: http://dx.doi.org/10.25139/ijair.v6i1.8377

Process	EMPTY BOXES	Times Required	Number of Processes Backtracking
7	44	130576	1286
8	44	110987	1253
9	45	210119	2785
10	45	126579	1278

Figure 5 presents a graph illustrating the time in microseconds (ms) needed to solve a Sudoku puzzle. Based on the data displayed, it is possible to infer that the seventh and ninth tests have a longer completion time and are more challenging than the other tests.

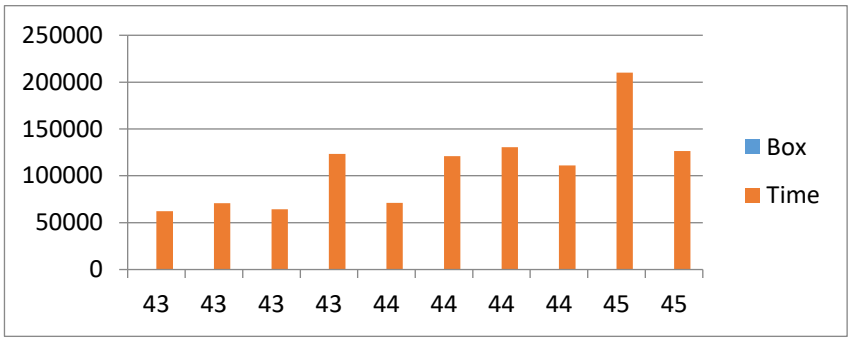

Fig 5. Test Results Graph

Equation (1) is the formula used to determine the average completion time needed for the backtracking technique. The time required (in milliseconds), the number of trials, and the average completion time are represented by the x, y, and s variables..

$$y = \frac{\sum x}{s} \quad (1)$$

It is possible to compute the average value. $y = \frac{109109,6}{10}$ = *10910,96 ms.*

B. Procedure for Solving Sudoku Using Brute Force

Implementation of the brute force algorithm in finding Sudoku solutions is carried out with the following steps:

a) Find the location of unfilled/empty rows and columns.
b) Repeat each number starting from 1-9.
c) In each repetition, if the number can be placed correctly in that position, then the repetition will continue to the next position.
d) If the loop has finished until the 9th number and no invalid numbers are found, then the number in that position is deleted and one step back to the previous position.
e) The search is carried out until a solution or a legal solution is found.

In its application, the brute force algorithm requires a helper function to check whether a number is valid at a certain position. The helper functions created are used to help validate numbers in certain positions. This validation process is carried out in 3 directions: tracing a row, column, and block/box. Implementing this helper function is quite easy because it only repeats sequentially. If the same number is found, it will return false, which means the number is invalid. Conversely, if the number entered is not found in the row or column, it will return true. The pseudocode (3) is a general overview of the brute force algorithm for solving Sudoku games.

Pseudocode (3). Function Solve Brute Force

```
Function solve (input board:matrix of integer)-> bool
Algorithm

   For row <- 0 to row = 9 - 1
      For column <- 0 to column = 9 - 1
         If board[row][column] = 0 then
            For number <- 1 to number = 9
              If validate(number) then
                 Board[row][column] <- number
                    If solve(board) then
                       Return
                    Else
                       Return false
```

The pseudocode (3), the program runs as follows :

a) The search starts from an empty position with a board mark containing the value 0.
b) Once an empty board position is found, Brute Force will be applied from numbers 1 to 9.
c) Validation will be carried out for each iteration.
d) If the number is valid, the board will be filled with that number.
e) After the filling, the settlement function is carried out on the same board.
f) If there are no valid numbers, the program will return a false value, which can be used to backtrack the recursive program or inform you that the puzzle has no solution.

Fig.6 and Fig.7 are examples of solving a Sudoku game using the Brute Force algorithm.

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

Fig 6. Evil Model 9 x 9 Sudoku boar

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

Fig 7. Solving Sudoku Model Evil 9 x 9

IV. CONCLUSION

It may be inferred from the experiments comparing the work processes of backtracking and brute force algorithms that employing these two approaches to solve Sudoku is equally optimal, particularly in the 9x9 model. However, regarding steps, the backtracking approach is considered quick because it doesn't need extra functions or iterations. Regarding time and speed, the backtracking method can be utilized as an alternate Sudoku game-solving technique because it only needs five search steps.

REFERENCES

[1] S. Sauri, et al, "Pembuatan Media Pembelajaran Sudoku Puzzles Pada Materi Sistem Periodik Unsur", Gunung Djati Conference Series, Volume 2, Seminar Nasional Tadris Kimiya 2020, ISSN: 2774-6585, 2021.
[2] A.A. Hanafi, et al, “Penyelesaian Permainan Sudoku Menggunakan Algoritma Backtracking Berbasis Artificial Intelligence”, Jurnal ICTEE, Vol. 2, No. 2, E-ISSN : 2746-7481, pp. 50-57, 2021.
[3] A. Sinaga; Nuraisana, “Implementasi Algoritma Brute Force Dalam Pencarian Menu Pada Aplikasi Pemesanan Coffee (Studi Kasus : Tanamera Coffee)”, JIKOMSI [Jurnal Ilmu Komputer dan Sistem Informasi], Vol.4 No.1, pp 6-15, 2021.
[4] F.A. Rahman; D. Anubhakti, “Implementasi Algoritma Backtracking Pada Permainan Sudoku”, MEANS (Media Informasi Analisa dan Sistem), Vol. 5, No.1, pp. 67-71, 2020.
[5] C. Danuputri; N. Santosa, “Aplikasi Pemecahan Soal Sudoku dengan Metode Backtracking” , Jurnal Informatika Universitas Pamulang, Vol. 6, No. 3, pp. 506-511, 2021.
[6] B.V. Indriyono, et al, “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.
[7] A. Yusuf; Hendra, “Penyelesaian Puzzle Sudoku Menggunakan Algoritma Brute Force Dan Backtracking”, Jurnal Techno Nusa Mandiri, Vol.X, No.1, pp. 207-215, 2013.
[8] Herimanto et al, “An Implementation of Backtracking Algorithm for Solving A Sudoku-Puzzle Based on Android”, Journal of Physics: Conference Series, pp. 1-6, 2019. https://doi.org/10.1088/1742-6596/1566/1/012038.
[9] F. Utama, et al , “Implementasi Backtracking Algorithm Untuk Penyelesaian Permainan Su Doku Pola 9x9”, Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer, Vol. 11, No. 1,pp. 29–36. 2017. https://doi.org/10.30872/jim.v11i1.200,

[10] Afriyudi et al, “Penyelesaian Puzzle Sudoku menggunakan Algoritma Genetik”, Prosiding SNASTI, ISBN 978-979-89683-31-0.
[11] D.S. Rahayu, et al, "Evaluasi Algoritma Runut Balik dan Simulated Annealing pada Permainan Sudoku", Jurnal Teknik Informatika dan Sistem Informasi, Vol. 3, No. 1, pp. 170-178, 2017.
[12] Yulyanto, et al, "Pengembangan Game Puzzle Find GrassMenggunakan Algoritma Backtracking", Bulletin of Information Technology (BIT), Vol 4, No 2, pp. 275-280, 2023..
[13] H.S. Sulun; R. Muni, “Pembangkit Teka-Teki Silang Dengan Algoritma Backtracking Serta Aplikasi Permainannya Yang Berbasis Web”, Jurnal Informatika, Vol.4 No.2, pp. 457-466, 2010.
[14] B.W. Santoso, et al, “Implementasi Algoritma Brute Force Sebagai Mesin Pencari (Search Berbasis Web Pada Database”, Jurnal Sisfotek Global, Vol. 6 No. 1, pp. 1-8, 2016.
[15] A.T. Ramadhoni, et al, “Penerapan Algoritma Brute Force Pada Aplikasi Sidayko Berbasis Android”, Jurnal MNEMONIC, Vol 5, No. 1, pp. 1-8, 2020.
[16] M.E. Riadi, “Penggunaan Media Puzzle Untuk Meningkatkan Hasil Belajar Siswa Pada Tema Lingkungan Kelas Ii Sdn Jajartunggal Iii Surabaya”, JPGSD, Vol. 02, No. 02, pp. 1-11, 2014.
[17] N. Nari et al, “Penerapan Permainan Puzzle Untuk Meningkatkan Kemampuan Membilang”, Jurnal Pembangunan dan Pendidikan: Fondasi dan Aplikasi, Vol.7, No. 1, pp. 44-52, 2019.
[18] A.T. Benjamin, “The Mathematics of Games and Puzzles: From Cards to Sudoku”. Amerika: Harvey Mudd College; 2013.
[19] R.D.I Sari, “Analisis Penyelesaian Puzzle Sudoku dengan Menerapkan Algoritma Backtracking”. Jurnal Ilmiah Teknologi dan Informasi Asia.Vol. 2, No. 2, pp. 1-18, 2011.

DOI: http://dx.doi.org/10.25139/ijair.v6i1.8377