Application of Backtracking Algorithm in Sudoku Solver

Muhammad Rizki Fonna
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516001@std.stei.itb.ac.id

***Abstract*— This cover with discuss mainly about how to solve Sudoku games using backtracking algorithm. Sudoku games with size n x n can be solved by backtracking algorithm. Through the discussion in this paper, the backtracking algorithm is used when the program collects to complete the game by entering certain numbers that meet the box settings in the boxes that are still empty (not filled by the user). If in the filling process it turns out that there is an imperfection, the backtracking process will be carried out.**

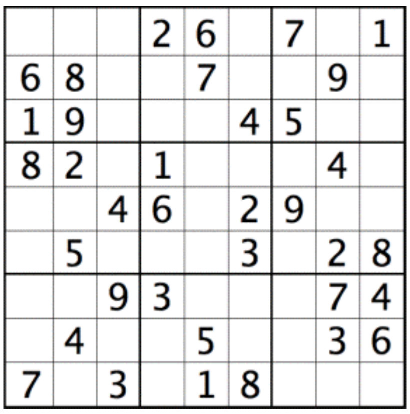
Keywords*—*Sudoku; algorithm; backtracking; solver

I. Introduction

Sudoku, also known as Number Place or Nanpure, is a kind of logic puzzle. The goal is to fill in the numbers from 1 to 9 into 9 × 9 nets consisting of 9 3 × 3 boxes without any repeated numbers in a row, column or box. First published in a French newspaper in 1895 and possibly influenced by Swiss mathematician Leonhard Euler, who made famous Latin square.

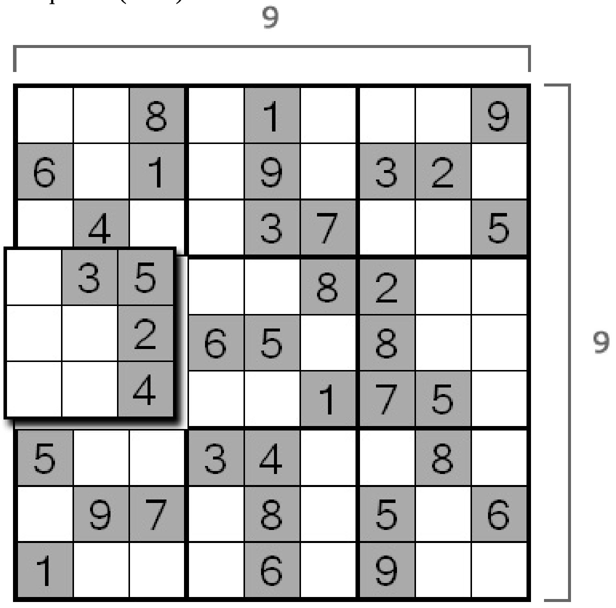
The modern version of the game began in Indianapolis in 1979. Then it became famous again in Japan in 1986, when the Nikoli publisher discovered this riddle created by Howard Garns.

The name "Sudoku" is a Japanese abbreviation of "Suuji wa dokushin ni kagiru" (数字 は 独身 に 限 る), meaning "the numbers must remain single".


Picture 1 Example of Sudoku Game Board

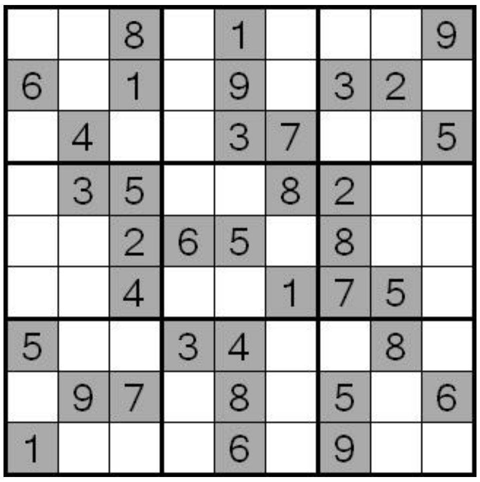
There are some rules that are applied in sudoku.

- Sudoku is played in 9x9 boxes divided into 3x3 small squares (cells) called "areas":



Picture 2 Sudoku Board (http://lifeslittleinspirations.com/sudoku-rules-for-the-game-of-life)

- Sudoku starts with several cells already filled with numbers:

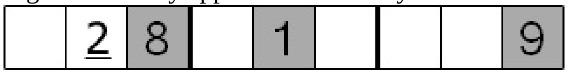


		8		1				9
6		1		9		3	2	
	4			3	7			5
	3	5			8	2		
		2	6	5		8		
		4			1	7	5	
5			3	4			8	
	9	7		8		5		6
1				6		9		

Picture 3 Sudoku Board Column (http://lifeslittleinspirations.com/sudoku-rules-for-the-game-of-life)

- The goal of the Sudoku game is to fill in empty cells with numbers between 1 and 9 (one-mark only 1 number) according to the following instructions:

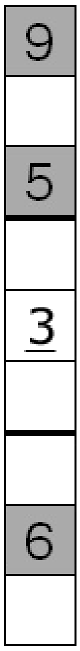
a. Figures can only appear once in every row:



	2	8		1				9

Picture 4 Sudoku Board Row (http://lifeslittleinspirations.com/sudoku-rules-for-the-game-of-life)

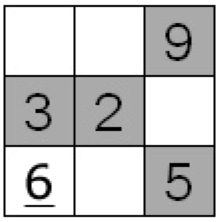
b. Numbers can only appear once in each column:



9
5
3
6

Picture 5 Sudoku Board Column (http://lifeslittleinspirations.com/sudoku-rules-for-the-game-of-life)

c. Figures can only appear once in each area:



		9
3	2	
6		5

Picture 6 Sudoku Board Column (http://lifeslittleinspirations.com/sudoku-rules-for-the-game-of-life)

- Summary of the rule is a number should appear once in each row, column, and area.

II. BACKTRACKING ALGORITHM

Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to the time elapsed till reaching any level of the search tree).

For example, consider the SudoKo solving Problem, we try filling digits one by one. Whenever we find that current digit cannot lead to a solution, we remove it (backtrack) and try next digit. This is better than naive approach (generating all possible combinations of digits and then trying every combination one by one) as it drops a set of permutations whenever it backtracks.

Generally, every constraint satisfaction problem which has clear and well-defined constraints on any objective solution, that incrementally builds candidate to the solution and abandons a candidate ("backtracks") as soon as it determines that the candidate cannot possibly be completed to a valid solution, can be solved by Backtracking. However, most of the problems that are discussed, can be solved using other known algorithms like Dynamic Programming or Greedy Algorithms in logarithmic, linear, linear-logarithmic time complexity in order of input size, and therefore, outshine the backtracking algorithm in every respect (since backtracking algorithms are generally exponential in both time and space). However, a few problems still remain, that only have backtracking algorithms to solve them until now.

Consider a situation that you have three boxes in front of you and only one of them has a gold coin in it but you do not know which one. So, in order to get the coin, you will have to open all of the boxes one by one. You will first check the first box, if it does not contain the coin, you will have to close it and check the second box and so on until you find the coin. This is what backtracking is, that is solving all sub-problems one by one in order to reach the best possible solution.

Consider the below example to understand the Backtracking approach more formally,

Given an instance of any computational problem P and data D corresponding to the instance, all the constraints that need to be satisfied in order to solve the problem are represented by C. A backtracking algorithm will then work as follows:

The Algorithm begins to build up a solution, starting with an empty solution set S. S = {}

1. Add to S the first move that is still left (All possible moves are added to S one by one). This now creates a new sub-tree s in the search tree of the algorithm.

2. Check if S+s satisfies each of the constraints in C.

3. If Yes, then the sub-tree s is "eligible" to add more "children".

4. Else, the entire sub-tree s is useless, so recurs back to step 1 using argument S.

5. In the event of "eligibility" of the newly formed sub-tree s, recurs back to step 1, using argument S+s.

6. If the check for S+s returns that it is a solution for the entire data D. Output and terminate the program. If not, then return that no solution is possible with the current s and hence discard it.

The backtracking algorithm enumerates a set of partial candidates that, in principle, could be completed in various ways to give all the possible solutions to the given problem. The completion is done incrementally, by a sequence of candidate extension steps.

Conceptually, the partial candidates are represented as the nodes of a tree structure, the potential search tree. Each partial candidate is the parent of the candidates that differ from it by a single extension step; the leaves of the tree are the partial candidates that cannot be extended any further.

The backtracking algorithm traverses this search tree recursively, from the root down, in depth-first order. At each node c, the algorithm checks whether c can be completed to a valid solution. If it cannot, the whole sub-tree rooted at c is skipped (pruned). Otherwise, the algorithm (1) checks whether c itself is a valid solution, and if so reports it to the user; and (2) recursively enumerates all sub-trees of c. The two tests and the children of each node are defined by user-given procedures.

Therefore, the actual search tree that is traversed by the algorithm is only a part of the potential tree. The total cost of the algorithm is the number of nodes of the actual tree times the cost of obtaining and processing each node. This fact should be considered when choosing the potential search tree and implementing the pruning test.Run-back is an improvement from the exhaustive-search algorithm that will systematically search for solutions to problems among all possible solutions. Only searches that lead to solutions will be considered. As a result, search time can be saved. A more natural run-off is expressed in a recursive algorithm. To facilitate this search, the solution space is organized into a tree structure. Each tree node states the status (state) of the problem, while the side (branch) is labeled with x values. The path from root to leaf expresses a possible solution. The entire path from root to leaf forms a solution space.

III. SOLVING WITH BACKTRACKING ALGORITHM

In this game the backtracking algorithm becomes a component in the application of algorithms to the program, so the program can complete the game by getting a solution. The program will fill each box with a certain value that feels right, then will continue to fill the next box or will return to the previous box if it does not lead to a possible solution.

Scanning

It is a process of moving rows or columns to identify lines in a block that have certain numbers. This process is then repeated in each column (or row) systematically. Then determine the value of a cell by removing impossible values.

Marking

In the form of logic analysis, by marking candidate numbers that can be entered in a cell.

Analysis

It is the elimination of candidates, where progress is achieved by eliminating candidates in a row until a cell has only one candidate.

The steps for finding a solution in the status space tree are built

dynamically:

1) Solutions are sought by forming a path from root to leaf. Rules the formation used is to follow the deep search method (DFS). The nodes that have been born are called live nodes (live node). The live node that is being expanded is called the E-node (Expand-node). Knots are numbered from top to bottom according to the order of birth.
2) Each time the E-node is expanded, the track built by it increases long. If the track being formed does not lead to a solution then E-node is "killed" so that it becomes a dead node. Function what is used to kill node-E is to apply a function bounding function. Dead knots will never expanded again.
3) If the formation of the path ends with a dead node, then the search process continued by generating another child knot. If nothing else child node that can be raised, then the search for solutions is continued with do backtracking to the closest live node (parent node). Then this node becomes the new E-node. A new track was built return until the path forms a solution.
4) The search is stopped if we have found a solution or there is no node live for backtracking or vertices that can be expanded.

Each node in the status space tree is associated with a call recursive. If the number of vertices in the status space tree is 2n or n!, Then for worst case, backtracking algorithms require time in O (p (n) 2n) or O (q (n) n!), With p (n) and q (n) is the polynomial degree n which states time computing each node.

The general constraint satisfaction problem consists in finding a list of integers x = (x[1], x[2], …, x[n]), each in some range {1, 2, …, m}, that satisfies some arbitrary constraint (boolean function) F.

For this class of problems, the instance data P would be the integers m and n, and the predicate F. In a typical backtracking solution to this problem, one could define a partial candidate as a list of integers c = (c[1], c[2], …, c[k]), for any k between 0 and n, that are to be assigned to the first k variables x[1], x[2], …, x[k].

The use of this backtracking algorithm will be seen in the process of filling cells with a number where there are several possible numbers that are suitable for that cell. In subsequent replenishment, the numbers entered will be matched with the numbers on the cells in the corresponding rows, columns and subgrids. The method of comparing and searching numbers leading to solutions is recursively done. The process of finding a solution is illustrated in the block diagram above.

For example, G is a simple graph, and PG (k) is a lot of ways to color vertex G with k colors so that no two adjacent vertices get the same color. The function of PG (k) is called polynomial chromatic G or many chromatic tribes G.

Example :

If G is the P3 path, then to label the graph point, the point in the middle can be named any of A. colors. The color cannot be used again to color the other point so that it meets the dye requirements. So that the points that are set aside by these points can be colored with one color from (A. - I) the remaining color, then P (G, A) = A. (A. - 1) 2

Example :

If G is a complete graph K3, then the top tick can be any arbitrary from A. color, the other points can be named with A - I color because the point is set aside with the top point. On the other k points (leftovers) can be given a color range from A. - 2 colors that have not been given kao to the other two points, then P (G, /,) = /, (I, - I) (A .- 2)

Example :

Take graph G as a blank graph with p points each of which can be colored by A. way, then the graph is valid P (G, A.) =) P.

Theorem 4:

G is a simple graph, eg A and B two non-fractional pins in the G. G '

is a graph obtained from graph G by connecting a line between RA and AB. While G "is a graph of the gain obtained by tapping A and B, which becomes a single point, then P (G, A.) = P (G ', A.) + P (G' ', A.)

Proof:

The number of ways of staining from G can be grouped n in two groups. Group I is coloration G where litik A and B have the same color and group 2 is G coloration where t and t A and B have different wats. A lot of coloration -A. d ima na titi k A and B have different colors. If a line connects points A and B, then a lot of coloration - /,. from G '. Number of colorants n- A. of G where the tick A and B are the same color is not pronounced, ji ti k-titi k A and B is united so it will be equal to the number of dye n - A. from G ". So that the value of a G is: P (G, A.) = P (G '. A.) + P (G ", /,). To illustrate the writing of P (G, /,) = P (G ', A.) + P (G ", / ..), I am pictured where graph G is taken simple graph By using the above theorem by re-signing i A and B as titi k which should be observed at every step.

From the results obtained, it appears that the polynomial of the chromite from a graph is simplified to be a biological factorial.

Theorem 4:

G is a simple graph, missal G 'and G "there is a graph obtained by G G by removing and tuning between points A and B into a thole mat l, then P (G, A.) = P (G', A.) - P (G ".A.)

Proof:

In the previous theorem of the process, the process of adding the line and ending with the chromatic polynomial expressed in the form of factoria on the complete graphs. But in this theoretical process that is done is the process of removal of the line. Apabi la A and B are the two points that are split in G ma, then G = G '- (A, B) and G "are obtained from G by synthesizing (A, B) in a single pole. will be r in the form of polynomial blank graphs From the description is obtained

P (G, A.) = A. 4 -4 / .. 3 + 6 / .. 2 -3 A.

The line or rib coloring on a graph is the determination of the color of the ribs of a graph so that each adjacent rib gets a different color. The size of the coloring of a graph is defined as the size of point coloring, which refers to the number of colors that are possible so that each rib with a tan gets a different color. The color minima l can be used to color the ribs in a graph G called chromatic chromatic G.

Definition 4:

The coloring A of graph G. From a graph G is a given A. color on each line of G is so that two or more lines that meet at a point are given a different color. If G has a coloring A. , then G is said to be A colorized.

Definition 5:

The chromatic index of G is notated as K (G). is the minimum number A needed when G has the A colorized.

Definition 6:

K(Km,n) =d = max (m , n)

Each square (small box) of Sudoku is represented by a dot. So that the graph of sudoku 9 x 9 has 81 points that correspond to the number of boxes. Each point of the sudoku adjasen graph with a point in line, a column, and a box (3x3).

Steps that can be used for Sudoku work using graphs are as follows:

1) Changing Sudoku elements into Sudoku graphs where each element is a vertex of the graph and denoted by vi, j with Vertex vi, j and vi, j are said to be neighbors if i i = i 'or j = j' which is connected by an edge. Edge-edge is seen as the relation of each number element in Sudoku

2) Giving color to each Sudoku graph vertex. The trick is to give a certain color to a vertex, then find another vertex that has not been given a color and do the same process. This search is continued on other vertices that have not been colored until all vertices have been properly colored. In graph theory, if the vertexes connected by an edge do not have the same color, they are called 'the right color'.

3) Re-change Sudoku graphs that have been colored in the early form of Sudoku games, namely Sudoku tables. The colors on Sudoku graphs are converted into numbers in Sudoku games.

To find a solution, a graph coloring technique is based on the known chromatic number. The minimum color that must be owned by Sudoku 9 x 9 is 9 colors, so Sudoku has a solution:

Here's one way to solve the problem of Sudoku Figure 3.2.

Given 9 colors as follows

• 1 = blue

• 2 = red

• 3 = green

• 4 = yellow

• 5 = purple

• 6 = orange

• 7 = brown

• 8 = ash

• 9 = pink

Vertex v2,2, v3,5, v1,8, v5,3, v6,6, v4,9, v8,1, v9,4 and v7,7 are not neighboring so the initial color is blue, so the initial stage The following picture is obtained:

Then v1,9, v2,3, v3,6, v4,1, v5,4, v6,7, v7,8, v8,1 and v9,4 are adjacent to vertices that have been given a blue color and between the nine vertices are not neighboring, so that it is given a red color.

Examples of solutions that can be done as one of the application of backtracking algorithms:

1. Search for empty cells.

The first empty cell is in the 1st row of the 2nd column.

2. The number to be filled is 1,2,3,4,5,6,7,8,9.

3. Search for candidate numbers to be filled Candidates for possible numbers are as follows:

1st row, 2nd column (S1.2)

a. Possible candidates in line 1, B = {3,6,9}

b. Possible candidates in column 2, K = {3,4,7,8,9}

c. Possible candidates in the 1st block, G = {1,5,7,9}

4. Checking

Set A = {3,6,9}

Set B = {3,4,7,8,9} Set C = {1,5,7,9} A ∩ B ∩ C = {9}

So, the solution is 9.

5. The solution is filled in the cell.

6. Next, search for the next empty cell, namely the cell in the 1st row of the 6th column. The process is repeated until a solution for the empty cell in line 1 is obtained:

S1,6 = {6}, S1,9 = {3}

7. The search for solutions in line 2 will be faced with a different case, which is a set of solutions that are more than one. In this case a backtrack will be made to another number candidate. The results of checking a set of solutions will be stored which can then be used for the backtracking process.

Search for candidate numbers for line 2:

S2,2 = {7}, S2,3 = {1,5}, S2,5 = {6,8,9}, S2,6 = {2,6,8,9},

S2,7 = {6.9}, S2,8 = {5,6}, S2,9 = {5}

8. Backtrack is then based on the set of solutions that already exist for each cell. So we get a solution:

S2,2 = {7}, S2,3 = {1}, S2,5 = {8}, S2,6 = {2}, S2,7 = {9}

S2,8 = {6}, S2,9 = {5}

9. The process of finding a solution is done repeatedly according to the number of cells that are still empty until all cells are filled with numbers according to the existing limiting function.

Acknowledgment

First, the writer thanks God because with His mercy, the report on the assignment of Strategi Algoritma can be done by the author and the writer completes it on time. The author also thanked Mrs. Dr. Nur Ulfa Maulidevi ST,M.Sc. as the author's lecturer in the Strategi Algoritma for all the guidance and teaching she has given to the author.

References

[1] Ronald L Graham, "Concrete Mathematics (Second Edition),"California: McGraw-Hill, 1993.

[2] Ralph P. Grinaldi, Discrete and Combinatorial Mathematics. Indiana, CA: Pearson Addison Wesley, 2001, pp. 145–178.

[3] Kenneth H. Rosen, DiscreteMathematics and Its Applications (Seventh Edition). New Jersey: McGraw Hill , 2007, pp. 104–139.

[4] Kenneth H. Rosen, Handbook of Discrete and Combinatorial Mathematics. Florida: CRC Press , 2000, pp. 922–987.

[5] Munir, Rinaldi.Matematika Diskrit. Bandung: Informatika , 2012, edisi kelima.

[6] http://viemagazine.com/article/thinking-inside-the-box/ (diakses tanggal 8 Desember 22.21 GMT+7)

[7] https://plus.maths.org/content/anything-square-magic-squares sudoku (diakses tanggal 9 Desember 09.33 GMT+7)G. Eason, B. Noble, and I.N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (*references*)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 25 April 2019

[illegible]

Muhammad Rizki Fonna
13516001