Fundamentals of Data Structures and Algorithms for AI

# **Sudoku Implementation**

Prepared By

Sintayew Zekarias

Fikir Awoke

Course Instructor - Dr. Beakal Gizachew

Addis Ababa Institute of Technology

School of Information Technology and Engineering

February 6, 2022

**1. Introduction**

Searching algorithms are used to solve problems from simple to more complex problems. Time and space is an important element to decide which algorithm is designated to solve a specific problem. There are various searching algorithms for solving problems. In this assignment we tried to show five searching algorithms to solve the Sudoku problem. The Breadth-First Search, Depth-First Search, Backtracking search, Heuristic search and A\* search.

**2. The Sudoku Puzzle**

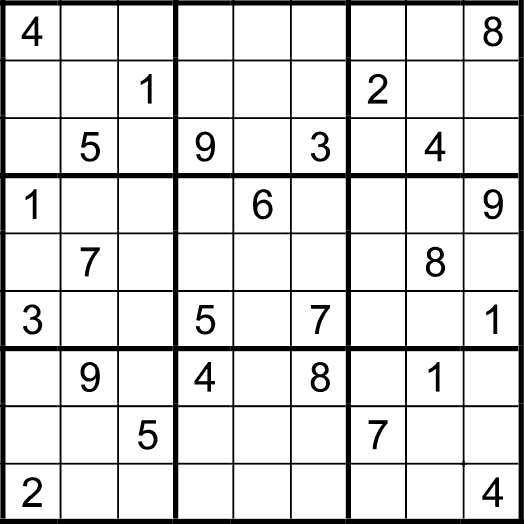
Sudoku is a game that is played by a single individual trying to fill the blank space with the appropriate numbers based on the rule of the game. The puzzle has a board that contains nine by nine squares where some of the squares are filled with random numbers and some others are empty. The objective of the game is to fill all the remaining empty spaces. To fill up the spaces there are certain rules that must be followed by the player in order to win the game. The rules are as follows:

Ø If the puzzle is played with numbers, only integer values from one to nine are acceptable.

Ø Every square in the same row and column must be filled in with a different number.

Ø Each one of the nine boxes must have different numbers.

The main idea to solve the game is to try each combination of numbers in each empty spot making sure that the rules are not violated.



*Figure1: The Sudoku puzzle*

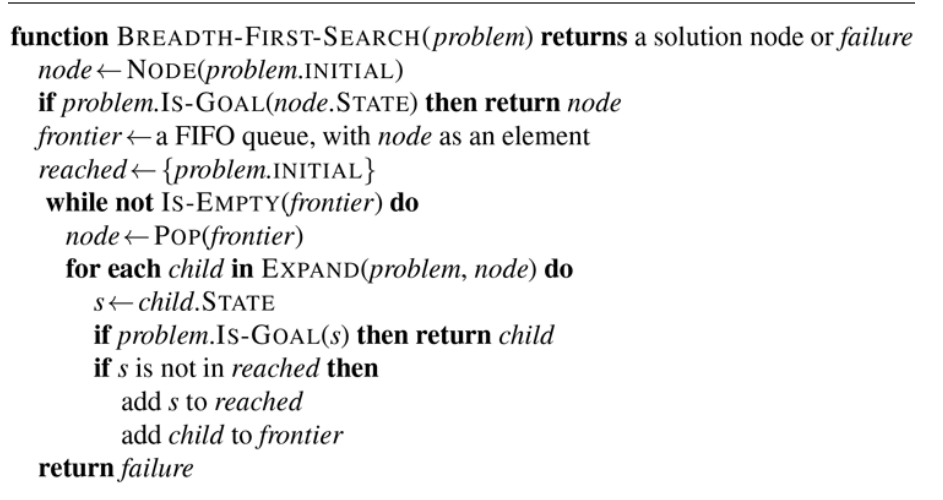
**3. The Searching Algorithms**

Breadth-First Search and Depth-First Search are uninformed search strategies that use the concept of a queue to expand the states from a graph or a tree structure.

**3.1. Breadth-First Search**

In breadth-first search the root node is expanded first, then all the successors of the root node are expanded next, then their successors, and so on until it gets to the bottom level. It uses a FIFO (First-In-First-Out) queue. This is a systematic strategy that is therefore complete even in an infinite state space.

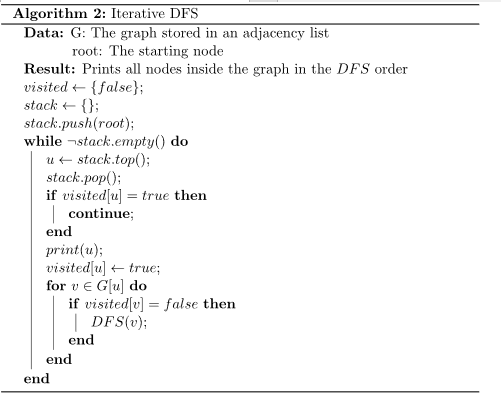
The BFS algorithm is shown below:



**3.2. Depth-First Search**

The depth-first search always expands to the deepest node in the frontier first. It will go deeper until it finds a goal or a dead end. The search then “backs up” to the next deepest node that still has unexpanded successors. The algorithm uses the LIFO (Last-In-First-Out) queue. Depth-first search is not cost-optimal; it returns the first solution it finds, even if it is not the cheapest solution.

The DFS algorithm is shown below:



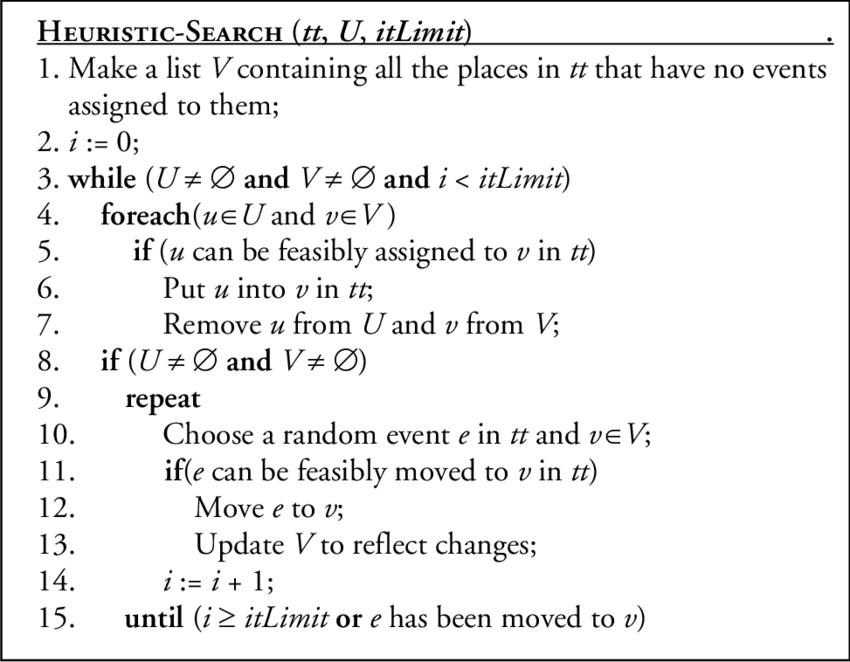
**3.3. Backtracking search**

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.

**3.4. Heuristic search**

Heuristic search is defined as a procedure of search that works to upgrade an issue by iteratively improving the arrangement dependent on a given heuristic capacity or a cost measure. This technique makes decisions by ranking every available choice at each branch of a search and then chooses the best option of those presented. Rather than focusing on finding an optimal solution like other search methods, heuristic searching is designed to be quick, and therefore finds the most acceptable option within a reasonable time limit or within the allocated memory space. It is considered an Informed Search, and it finds the most promising path.

The Heuristic search algorithm is shown below:



**3.5. A\* search**

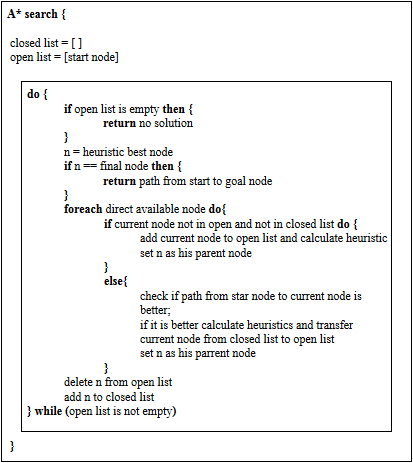
A\* search is a [graph traversal](https://en.wikipedia.org/wiki/Graph_traversal) and [path search](https://en.wikipedia.org/wiki/Pathfinding) [algorithm](https://en.wikipedia.org/wiki/Algorithm), which is often used in many fields of computer science due to its completeness, optimality, and optimal efficiency. One major practical drawback is its O ( b d ) {\displaystyle O(b^{d})} space complexity, as it stores all generated nodes in memory.

Unlike other algorithms, A\* decides to take up a step only if it is convincingly sensible and reasonable as per its functions. Which means, it never considers any non-optimal steps. This is why A\* is a popular choice for AI systems that replicate the real world – like video games and machine learning.

A\* algorithm works based on heuristic methods and this helps achieve optimality. A\* is a different form of the best-first algorithm. Optimality empowers an algorithm to find the best possible solution to a problem. Such algorithms also offer completeness. If there is any solution possible to an existing problem, the algorithm will definitely find it.

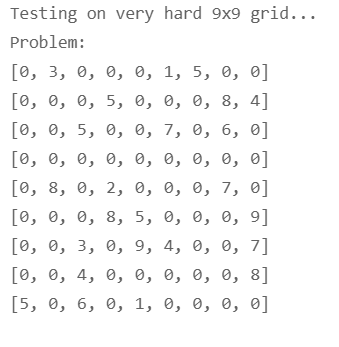
When A\* enters into a problem, firstly it calculates the cost to travel to the neighboring nodes and chooses the node with the lowest cost. If The f(n) denotes the cost, A\* chooses the node with the lowest f(n) value. Here ‘n’ denotes the neighboring nodes.The heuristic value has an important role in the efficiency of the A\* algorithm. To find the best solution, you might have to use different heuristic functions according to the type of the problem.

The A\* algorithm is shown below:



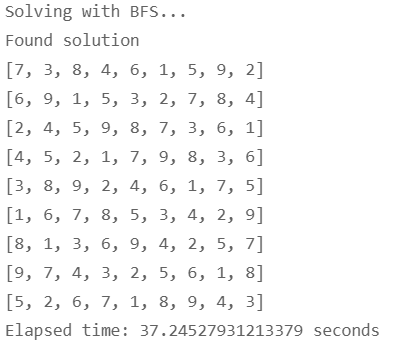
**4. Experimental Result**

We took as an exampl testing on very hard 9x9 girid problem, the result and time complexity of each algorithm pressented as follow



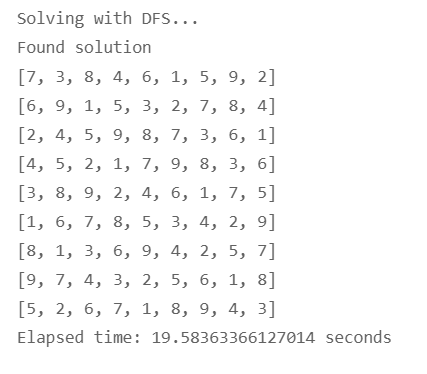
**4.1. BFS Result**

The following result shows the solved puzzle using the BFS algorithm, with the respective elapsed time in seconds.



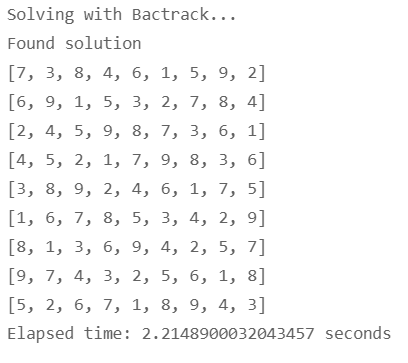
**4.2. DFS Result**

The following result shows the solved puzzle using the DFS algorithm, with the respective elapsed time in seconds.



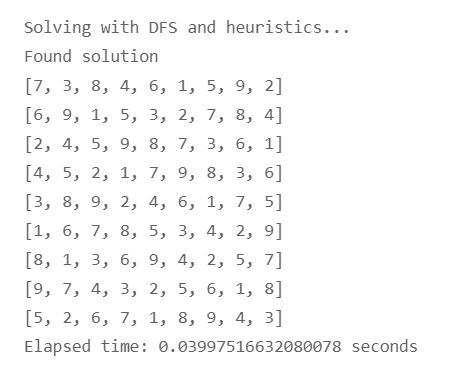
**4.3. Backtracking Result**

The following result shows the solved puzzle using the Backtracking search algorithm, with the respective elapsed time in seconds.



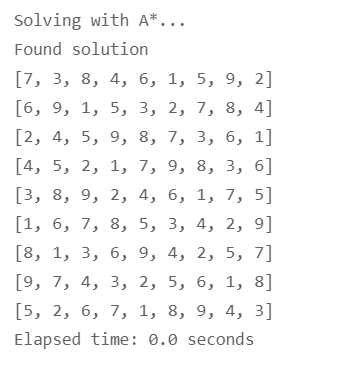
**4.4. Heuristic Result**

The following result shows the solved puzzle using the Heuristic Search algorithm, with the respective elapsed time in seconds.

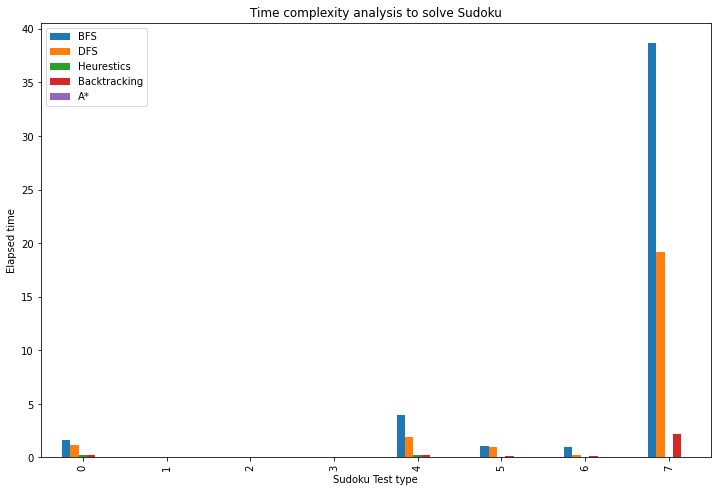


**4.5. A\* result**

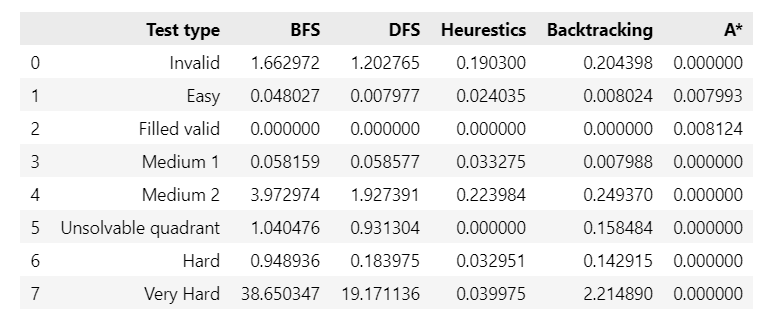
The following result shows the solved puzzle using the A\* algorithm, with the respective elapsed time in seconds.



**4.7. Plotted sudoku Result**



**4.8. Comparison of Algorithms**



**5. Conclusion**

This work included five searching algorithms to solve the sudoku puzzle, which are the Breadth-First Search, Depth-First Search, Backtracking search, Heuristic search and A\* search. Different concepts were implemented throughout the process in order to obtain the best performer in solving the puzzle. Based on our testing, the sudoku puzzle on A\* search has a better result with the lowest elapsed time compared to all the other algorithms implemented.

**Reference**

<https://en.wikipedia.org/wiki/Sudoku_solving_algorithms>

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.646.8115&rep=rep1&type=pdf>

[https://en.wikipedia.org/wiki/A\*\_search\_algorithm](https://en.wikipedia.org/wiki/A*_search_algorithm)

<https://www.mygreatlearning.com/blog/a-search-algorithm-in-artificial-intelligence/>