|  |  |
| --- | --- |
| **Student Full Name** | **Sherif Mostafa Samy Ahmed** |
| **ID** | **221903216** |
| **Faculty** | ***Faculty of Engineering*** |
| **Department** | ***Communication and Computer*** |
| **Subject** | **Structured Programming** |
| **Project** | **Sudoku Solver** |

1. **Detailed Problem Statement: -**

The history of Sudoku dates backs to an 18th Century Swiss mathematician’s game called “Latin Squares” (according to this article from the Economist) and some of the first number puzzles to appear in newspapers were published in France in 1895. But the modern game of Sudoku as we recognize it today was invented by Howard Garns, a freelance puzzle inventor from Connersville, Indiana, USA in 1979 when it was published in Dell Pencil Puzzles and Word Games magazine. The puzzle was known as “Number Place,” since it involved placing individual numbers into empty spots on a 9 x 9 grid.

The game first appeared in Japan in 1984 where it was given the name “Sudoku,” which is short for a longer expression in Japanese – “Sūji wa dokushin ni kagiru” – which means, “the digits are limited to one occurrence.” Sudoku continues to be highly popular in Japan, where people buy over 600,000 Sudoku magazines per month.

Now, and after AI appears and becomes a trendy topic, we try to use one of its algorithms to solve the game board. This algorithm is called “backtracking algorithm” and it depends on method’s recursion.

**Backtracking:** can be defined as an artificial intelligence algorithm that considers searching every possible combination to solve a computational problem.

**What is 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).

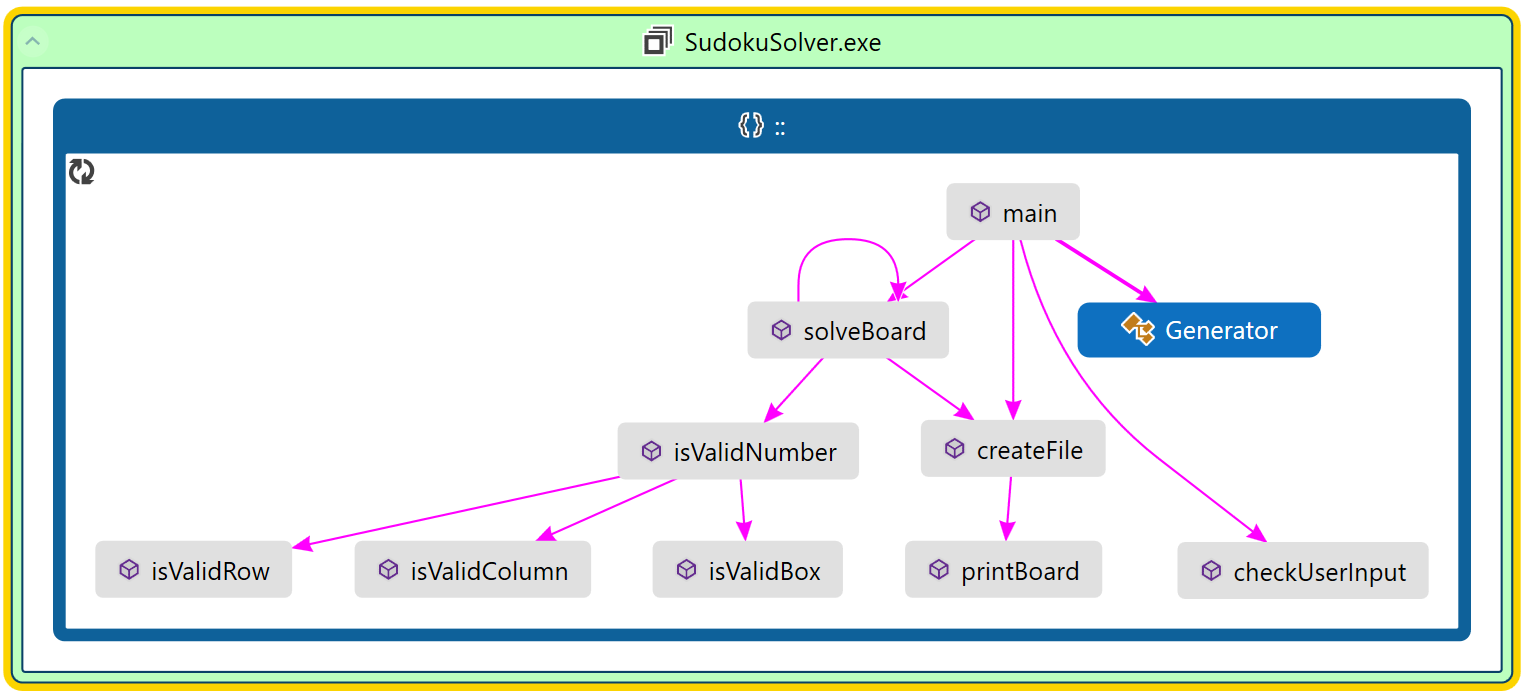
**When can be Backtracking Algorithm used?**

For example, considering the Sudoku solving Problem, we try filling digits one by one. Whenever we find that the current digit cannot lead to a solution, we remove it (backtrack) and try the next digit. This is better than a 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.

1. **Pseudo Code: -**

* User can enter his/her own board or user can choose to generate board automatically.
* User’s selection is protected against typos via (try / catch method and clearing buffer)
* If user selects to enter his/her own board, Application will ask him/her to enter 9 X 9 array that carry numbers from 0 to 9 where 0 means empty box.
* Application will ignore every invalid input (Application will empty the invalid placement) and carry only the valid numbers to the board.
* If the user selects to generate board automatically, Application will generate one using engine called “Generator”, but some necessary edits are made according to engine's outputs and "Solver" inputs.
  + The engine reference  
    <https://github.com/vaithak/Sudoku-Generator/blob/master/sudokuGen.cpp>
* Application will solve the board in each case using method called “solveBoard” via algorithm called “Backtracking”.
* Every solution is stored step by step in a text file called “SolutionFile.txt” and the application will read it in runtime in command prompt.
* The application calculates the time duration that is needed to solve game board.
* Some boards cannot be solved, this is one of two cases:
  + The board is unsolvable.
  + The time needed to solve is so much, here we need some advanced algorithms in artificial intelligence.

1. **Flowchart: -**

****

1. **Complete Code (in a ready to work state):-**