EIGHT QUEENS PROBLEM : FINALLY SOLVED

[1]Hula Mahmoud Alkhassawneh, [2] Kusum Yadav, [3]Shshad Talla
College of Computer Science & Engineering, University of Hail, Hail, Kingdom of Saudi Arabia
[1] h.alkhasoneh@uoh.edu.sa, [2]Kusumasyada0@gmail.com

***ABSTRACT**— In Eight queens problems, we are using a regular chess board (8X8). The challenge is to place the eight queens on the board in such a manner that no queen is attacking another (for those who are not familiar with chess board and its pieces, the queen is able to attack any square on the same row or column or even either of the diagonals). In this paper, we introduce a solution for Eight queens problem developed and implemented in java language so that no two queens can attack another one beside the designing also the interface has been implemented.*

Keywords— 8Queen, Puzzle, chesspuzzle, eightqueen solution

I. INTRODUCTION

The 8 Queen puzzle appears quite simple i.e. place eight queens on the board in a way that no two queens could attack another. The twist, however, is that the researchers want the algorithm to work on a 1,000 by 1,000 square chess board. As the number of squares goes up, so does the number of queens you need to place.

By current estimations placing 1,000 queens on a 1,000 by 1,000 square board could take thousands of years – due to the sheer number of possibilities to consider.

Any algorithm which is capable of solve the puzzle quickly would also be able to breach and crack the toughest of the online security measures. It would be incredibly powerful.[1]. In practice, nobody has ever come close to writing a program that can solve the problem quickly," said Dr Nightingale. So what our research has shown a new program solving Eight queens problems in a practical manner.

II. LITERATURE REVIEW

It is an example of a Millennium Prize Problem, of which there are seven in total. These puzzles were announced at a meeting in 2000 and are said to represent the most difficult problems facing mathematics. The CMI said this was to show people there were still many unanswered questions in maths.

The Board of Directors of CMI designated a $7 million prize fund for the solutions to the problems, with $1 million allocated to the solution of each problem. The first of these was solved in 2003 by Grigori Perelman, a Russian mathematician, but he declined the prize. The University of St Andrews has now laid down a challenge for anyone to solve this puzzle using a computer. University of St Andrews professor Ian Gent came up with the idea after he was challenged by a Facebook friend to solve the chess puzzle himself. He and his colleagues Dr Peter Nightingale and Dr Christopher Jefferson attempted to write a computer program themselves.

Their solutions use 'backtracking' – an algorithm used in programming where every possible option is considered and then 'backed away' from until the correct solution is found - and their ideas are outlined in a paper, published in the *Journal of Artificial Intelligence Research[3]*.

However, as soon as the chessboard became large enough – 1,000 by 1,000 squares – the algorithm couldn't cope. These problems are so difficult for computer programs because there are too many options to consider, and it can take many years.

"If you could write a computer program that could solve the problem really fast, you could adapt it to solve many of the most important problems that affect us all daily," said Professsor Gent. "This includes trivial challenges like working out the largest group of your Facebook friends who don't know each other, or very important ones like cracking the codes that keep all our online transactions safe."[3]

"In practice, nobody has ever come close to writing a program that can solve the problem quickly," said Dr Nightingale. "So what our research has shown is that – for all practical purposes – it can't be done."[3]

III. WHAT IS EIGHT QUEENS PROBLEM?

The Eight Queens Problem can be defined as follows: Place 8 queens on an (8 by 8) chess board such that none of the queens attacks any other. A configuration of 8 queens on the board is shown in figure 1, but this does not represent a solution as the queen in the first column is on the same diagonal as the queen in the last column.

Searching for a Solution

This problem can be solved by searching for a solution. The initial state is given by the empty chess board. Placing a queen on the board represents an action in the search problem. A goal state is a configuration where none of the queens attacks any of the others. Note that every goal state is reached after exactly 8 actions.

This formulation as a search problem can be improved when we realize that, in any solution, there must be exactly one queen in each of the columns. Thus, the possible actions can be restricted to placing a queen in the next column that does not yet contain a queen. This reduces the branching factor from (initially) 64 to 8.

Furthermore, we need only consider those rows in the next column that are not already attacked by a queen that was previously on the board. This is because the placing of further queens on the board can never remove the mutual attack and turn the configuration into a solution.

PROBLEM INVENTOR

The puzzle was originally proposed in 1848 by the chess player Max Bezzel, and over the years, many mathematicians, including Gauss, have worked on this puzzle and its generalized n-queens problem [4].

SOLUTION INVENTOR $\alpha + \beta = \chi$. (1)

The first solution for 8 queens were provided by Franz Nauck in 1850. Nauck also extended the puzzle to n-queens problem (on an n n board-a chessboard of arbitrary size)[ ]. In 1874, S. Günther proposed a method of finding solutions by using determinants, and J.W.L. Glaisher refined this approach.Edsger Dijkstra used this problem in 1972 to illustrate the power of what he called structured programming. He published a highly detailed description of the development of a depth-first backtracking algorithm [5].

IV. FORMULATION

States: any arrangement of 0 to 8 queens on the board.
Initial state: 0 queens on the board.
Successor function: add a queen in any square.
Goal test: 8 queens on the board, none attacked.

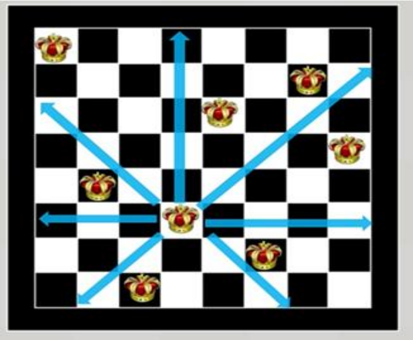


Figure 1: A configuration of 8 queens on chess board.

Different size boards

It's very easy to expand (and contract) this puzzle to other sized chess boards.

below is a table of the number of solutions for different sized n x n boards.For each size board I've shown the number of total solutions, and also the number of distinct types of solutions (unique before rotations and reflections).

Trivially, there is only one solution for a 1 x 1 board, and it's not hard to see that there are no possible solutions for a 2 x 2 or 3 x 3 sized board. It's interesting that, whilst there are 10 solutions to a 5 x 5 board, the number of solutions drops to just 4 solutions on a 6 x 6 board.There is (currently) no known formula for determining the number possible solutions for an n x n board, and an internet search reveals that the highest calculated board size to-date is 26 x 26 [3].

Board	Total Solutions	Unique Solutions
1 x 1	1	1
2 x 2	0	0
3 x 3	0	0
4 x 4	2	1
5 x 5	10	2
6 x 6	4	1
7 x 7	40	6
8 x 8	92	12
9 x 9	352	46
10 x 10	724	92
11 x 11	2,680	341
12 x 12	14,200	1,787
13 x 13	73,712	9,233
14 x 14	365,596	45,752
15 x 15	2,279,184	285,053
16 x 16	14,772,512	1,846,955
17 x 17	95,815,104	11,977,939
18 x 18	666,090,624	83,263,591
19 x 19	4,968,057,848	621,012,754
20 x 20	39,029,188,884	4,878,666,808
21 x 21	314,666,222,712	39,333,324,973
22 x 22	2,691,008,701,644	336,376,244,042
23 x 23	24,233,937,684,440	3,029,242,658,210
24 x 24	227,514,171,973,736	28,439,272,956,934
25 x 25	2,207,893,435,808,350	275,986,683,743,434
26 x 26	22,317,699,616,364,000	2,789,712,466,510,280

Figure 2: **number of solutions for different sized n x n boards**

EIGHT QUEEN PROBLEM ALGORITHM

EIGHT QUEEN PROBLEM: ALGORITHM

```
putQueen(row)
{
  for every position col on the same row
       if position col is available
            place the next queen in position col
       if (row<8)
            putQueen(row+1);
       else success;
   remove the queen from position col
}
```

V. BACKTRACKING CONCEPT

Each recursive call attempts to place a queen in a specific column. For a given call, the state of the board from previous placements is known (i.e. where are the other queens?) Current step backtracking: If a placement within the column does not lead to a solution, the queen is removed and moved "down" the column Previous step backtracking: When all rows in a column have been tried, the call terminates and backtracks to the previous call (in the previous column) [7].

THE PUTQUEEN RECURSIVE METHOD

```
void putQueen(int row)
 {
    for (int col=0;col<squares;col++)

         if (column[col]==available &&
         leftDiagonal[row+col]==available &&
         rightDiagonal[row-col]== available)
           {
               positionInRow[row]=col;
               column[col]=!available;
               leftDiagonal[row+col]=!available;

               rightDiagonal[row-col]=!available;
                if (row< squares-1)
                       putQueen(row+1);
               else
                       print(" solution found");
               column[col]=available;
                leftDiagonal[row+col]=available;
                rightDiagonal[row-col]= available;
           }
 }
```

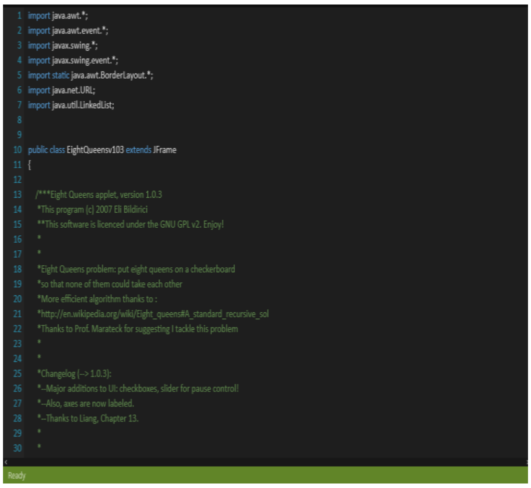
VI. BACKTRACKING ALGORITHM ADVANTAGE

Is the ability to find and count all the possible solutions rather than just one while offering decent speed.

SOLUTION

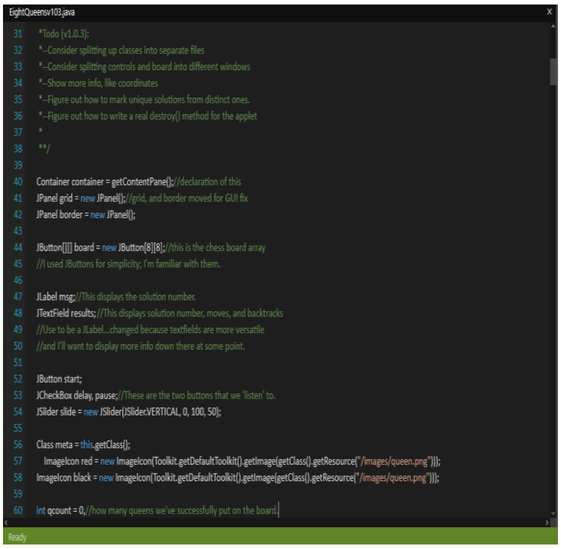
The eight queen puzzle has 92 distinct solutions.
If solutions that differ only by symmetry operations (rotations and reflections) of the board are counted as one the puzzle has12 unique solutions.

CODE SAMPLE



```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import static java.awt.BorderLayout.*;
import java.net.URL;
import java.util.LinkedList;


public class EightQueensv103 extends JFrame
{

  /***Eight Queens applet, version 1.0.3
   *This program (c) 2007 Eli Bildirici
   **This software is licenced under the GNU GPL v2. Enjoy!
   *
   *
   *Eight Queens problem: put eight queens on a checkerboard
   *so that none of them could take each other
   *More efficient algorithm thanks to :
   *http://en.wikipedia.org/wiki/Eight_queens#A_standard_recursive_sol
   *Thanks to Prof. Marateck for suggesting I tackle this problem
   *
   *
   *Changelog (-->1.0.3):
   *--Major additions to UI: checkboxes, slider for pause control!
   *--Also, axes are now labeled.
   *--Thanks to Liang, Chapter 13.
   *
   *
```

Ready

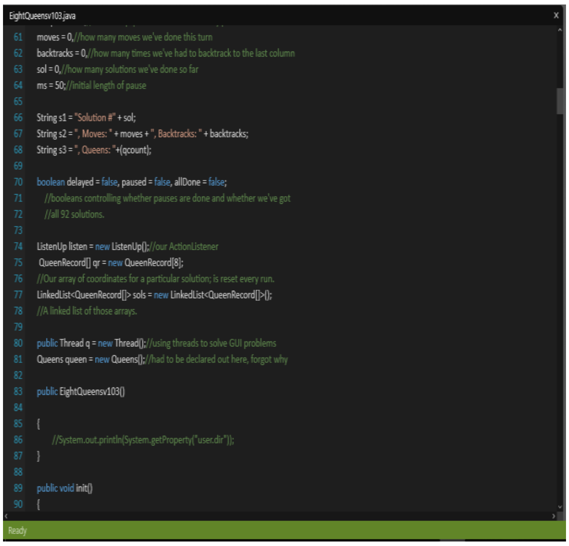


EightQueensv103.java

```java
   *Todo (v1.0.3):
   *--Consider splitting up classes into separate files
   *--Consider splitting controls and board into different windows
   *--Show more info, like coordinates
   *--Figure out how to mark unique solutions from distinct ones.
   *--Figure out how to write a real destroy() method for the applet
   *
   **/

  Container container = getContentPane();//declaration of this
  JPanel grid = new JPanel();//grid, and border moved for GUI fix
  JPanel border = new JPanel();

  JButton[][] board = new JButton[8][8];//this is the chess board array
  //I used JButtons for simplicity; I'm familiar with them.

  JLabel msg;//This displays the solution number.
  JTextField results;//This displays solution number, moves, and backtracks
  //Use to be a JLabel...changed because textfields are more versatile
  //and I'll want to display more info down there at some point.

  JButton start;
  JCheckBox delay, pause;//These are the two buttons that we 'listen' to.
  JSlider slide = new JSlider(JSlider.VERTICAL, 0, 100, 50);

  Class meta = this.getClass();
    ImageIcon red = new ImageIcon(Toolkit.getDefaultToolkit().getImage(getClass().getResource("/images/queen.png")));
  ImageIcon black = new ImageIcon(Toolkit.getDefaultToolkit().getImage(getClass().getResource("/images/queen.png")));

  int qcount = 0,//how many queens we've successfully put on the board.
```

Ready

.



EightQueensv103.java

```java
  moves = 0,//how many moves we've done this turn
  backtracks = 0,//how many times we've had to backtrack to the last column
  sol = 0,//how many solutions we've done so far
  ms = 50;//initial length of pause

  String s1 = "Solution #" + sol;
  String s2 = ", Moves: " + moves + ", Backtracks: " + backtracks;
  String s3 = ", Queens: "+(qcount);

  boolean delayed = false, paused = false, allDone = false;
    //booleans controlling whether pauses are done and whether we've got
    //all 92 solutions.

  ListenUp listen = new ListenUp();//our ActionListener
   QueenRecord[] qr = new QueenRecord[8];
  //Our array of coordinates for a particular solution; is reset every run.
  LinkedList<QueenRecord[]> sols = new LinkedList<QueenRecord[]>();
  //A linked list of those arrays.

  public Thread q = new Thread();//using threads to solve GUI problems
  Queens queen = new Queens();//had to be declared out here, forgot why

  public EightQueensv103()

  {
        //System.out.println(System.getProperty("user.dir"));
  }

  public void init()
  {
```

Ready

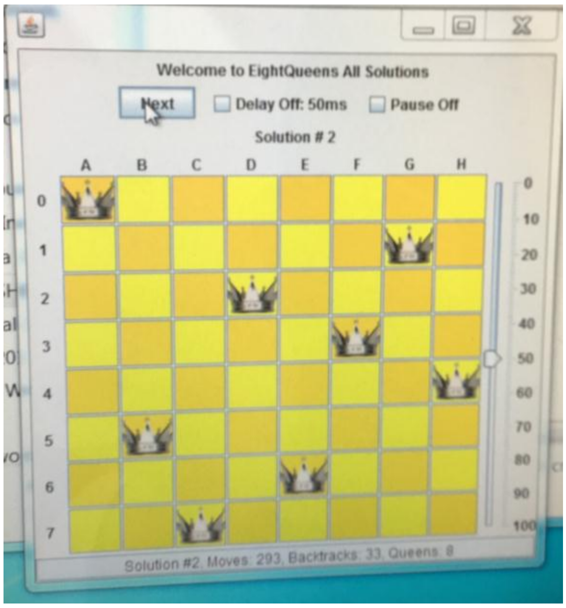


Figure 3: Screenshots from the program

CONCLUSION

With the above, mentioned algorithm and interface for the program the Eight Queen Problem has been solved with a regular size chess board implemented by java program so that no two queens could attack each other. The 8 Queen puzzle appears quite simple as that the researchers want the algorithm to work on a 1,000 by 1,000 square chess board. The possible actions can be restricted to placing a queen in the next column that does not yet contain a queen. This reduces the branching factor from (initially) 64 to 8. Using the same we solved the Wight queen problem with 8X8 chess board.

REFERENCES

[1]http://www.alphr.com/artificial-intelligence/1006866/win-million-eight-queens-puzzle#

[2] "The 8-Queens Puzzle". www.claymath.org. Clay Mathematics Institute. September 2, 2017. Retrieved September 7, 2017

[3] Gent, Ian P.; Jefferson, Christopher; Nightingale, Peter (August 2017). "Complexity of n-Queens Completion". Journal of Artificial Intelligence Research. 59: 815–848. doi:10.1613/jair.5512. ISSN 1076-9757. Retrieved September 7, 2017.

[4] Bezzel, M. (1848). Schachfreund. Berliner Schachzeitung, 3, 363. Cited in (Campbell, 1977).

[5] Nauck, F. (1850). Schach: Eine in das Gebiet der Mathematik fallende Aufgabe von Herrn Dr. Nauck in Schleusingen. Illustrirte Zeitung, 14 (361), 352. Cited in (Campbell,1977).

[6]http://www.datagenetics.com/blog/august42012/

[7] Backtracking Algorithms in MCPL using Bit Patterns and Recursion by Martin Richards mr@uk.ac.cam.cl http://www.cl.cam.ac.uk/users/mr/Computer Laboratory University of Cambridge February 23, 2009.