CONFIGURATION PROJECT

Intelligent Systems Programming

21-4-2016

N-Queens problem with BDD

by Braydon Sorenson and Abhay Singh

Overview

The goal of this assignment is to find solutions for the n-Queen problem using Binary Decision Diagram, in a way that none of the queens is able to Threaten any other queen using the standard chess moves for a queen (row-wise, column-wise, and diagonally).

This java program uses JavaBDD to solve the N-Queens problem. An Interactive Configurator aids the user by forbidding some cells based on the user's present selection of cell for placing the queen.

Objectives

We would say actual idea was to learn the use of BDD using this specific problem. A nice GUI was provided to us, all we were supposed to build a BDD, add rules to it

and implement it using the GUI provided. Most of the work was eased because of java library(JavaBDD) provided to us.

Implementation

For our implementation of the given problem we used the BDD API that was already included in the .jar file for the project.

We added several new methods to the QueensLogic.java file, which are described as follows:

- buildTheRules()

  Implement basic rule as in “Each row is supposed to have only one queen”.This was done to rule out those “Strange Crosses”.

  Also calls on buildRule() method to ensure the rules for each position on the game board.

- buildRule()

  Creates the rules for the cell to make sure that there are no other queens in the same column, row, or diagonal.

- updateGameBoard()

  Checks the BDD to find the spots that can no longer have a queen. When a user selects a cell to place a queen, it updates the game board to reflect the invalid positions on the GUI.

Additionally, we created a few new class level variables in the QueensLogic.java file:

- BDDFactory bddFact

  Used to create the BDD

- BDD bdd

  Represents the BDD for the position of the queens

- Int boardSize

  The number of cells in a row or column.

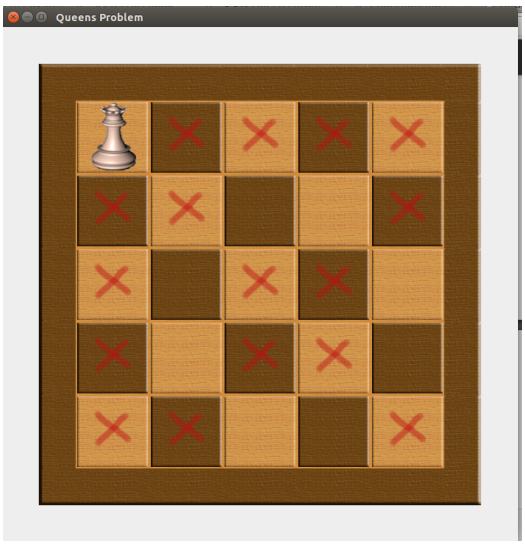
Some global variables(auto & counter) are also defined for auto placement of queens.

How it Works?

When the game is initialized, the BDD will be built by calling on the buildBDD method. This method will utilizes the BDD API. It also calls on the buildTheRules function, which enforces that no other queens can be placed in the same row, column, or diagonal as another queen for every position on the game board.

When the user clicks on a cell on the game board, the InsertQueen function will be called. This function calls on the updateGameBoard function to update the GUI. The cells that can no longer have a queen placed in them will display a red cross.

For example, let's say user click on 1st cell on a 5*5 board:

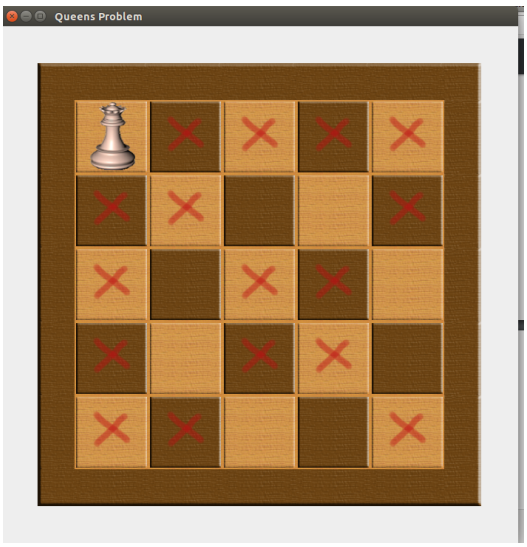


We can see that apart from rows,columns and diagonals there also exist some “strange crosses”, this is mainly because if you would place queen in any of that cell no solution would come out.

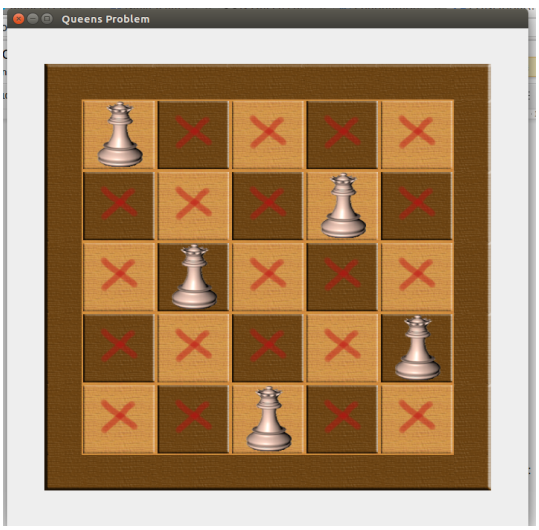
Automatic placement of Queens, if No. of valid cells equals to No. of Queens remaining.

For ex, on a 5*5 cell,

After placing 1st queen,



Adding 2nd queen at 3rd row 2nd column,



We can see that rest of queens are auto filled by our algorithm.

How to run?

We have tried running on Linux terminal, wherein the command is as provided:

```
javac -cp "javabdd-1.0b2.jar:." *.java && java -cp "javabdd-1.0b2.jar:." ShowBoard
```

Provided you have the javabdd-1.0b2.jar in same folder as in your other files

Conclusions

It was a nice experience working on BDD library, although there was not much to do for us since almost everything was provided. But since we have only a little experience with java, so it was overall moderate level for us.