Lab Report - Queens Problem

Hinweise zu dieser Aufgabe

Wir haben zu Beginn dieser Aufgabe mehrere Klassen geschrieben welche ohne weiteres zu einem tatsächlichen Schach erweitert werden kann. Auf Gründen der Performance haben wir diese jedoch nicht für das “8 Queens Problem” verwendet. Die Klassen liegen in de Ordner “Chesst for fun”. Das eigentliche Lab liegt wie gewohnt im Ordner “src”.

Our goal is to write a program to determine if eight queens can be placed on an 8 x 8 chess board without them threatening each other! Start by deciding how to represent a chess board with a data structure. Don't worry about the colors of the board yet. Write a Chessboard class! What methods will you need?

Wir haben uns entschieden lediglich die Queens in einem Array zu halten. Der Index des Array entspricht einer Reihe - der Wert einer Spalte im Schachbrett. Über die Funktion printMatrix(int[] queens) wird ein Schachbrett erzeugt.

Write a method for determining if the current board state contains a queen that is threatening another one. What is the complexity of your method in terms of N, the number of rows on the board? If your algorithm is worse than linear, you might want to look for something better.

Mithilfe der Mathematik kann N statisch gehalten werden. Es musste nicht abgeprüft werden ob sich eine Queen in der selben Reihe befindet da dies in unserem Algorithmus nicht passieren kann. Es bewegt sich immer eine Dame in einer Reihe.

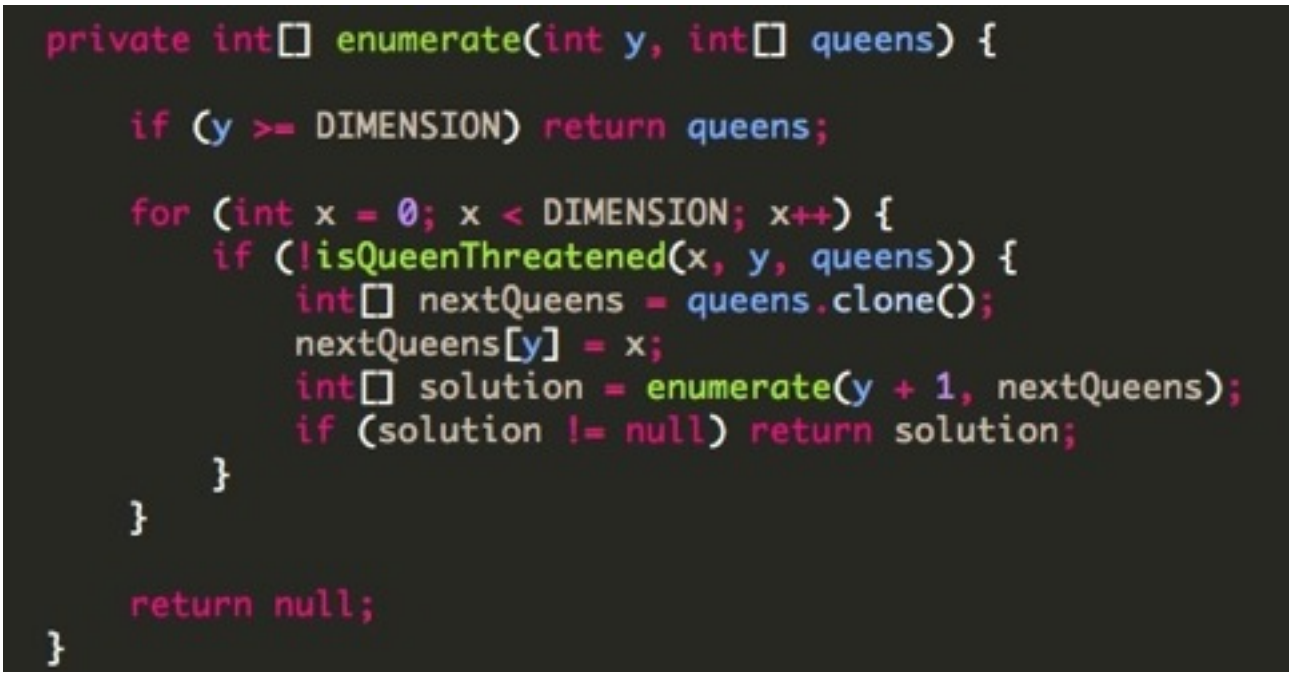
- x1 == x2: Befinden sich zwei Queens in der selben Spalte?
- y1-x1 == y2-x2 Befinden sich zwei Queens diagonal zueinander
- x1 + y1 == x2 + y2 Befinden sich Damen negativ diagonal zueinnder?

```
private boolean conflictingPositions(int x1, int y1, int x2, int y2) {
    return (x1 == x2)
        || ((y1 - x1) == (y2 - x2))
        || ((x1 + y1) == (x2 + y2));
}
```

Jorin Vogel, Elias Klemm

We speak of "backtracking" when we go back to a previous state and try a different branch. Use some coins on a paper chess board to figure out what to do when you reach a state in which one queen is threatened by another. There are iterative, recursive, and random solutions to this problem. Try and implement a recursive solution!

Wir haben uns lediglich die rekursive Form angesehen. In unserer Lösung werden die Queens je Zeile behandelt. Es befindet sich immer eine Queen in einer Spalte. Bei jedem Aufruf der Funktion wird geprüft auf welchem Feld in der Reihe die Möglichkeit besteht diese zu setzten. Falls keine Möglichkeit besteht eine Dame zu setzten wird wieder ein Schritt zurück gegangen d.h. die Funktion erneut aufgerufen mit dem alten Reihenwert. Falls eine Dame gesetzt werden kann kann in die nächste Reihe vorgerückt werden. Für alles weitere siehe kommentierten code.



```
private int[] enumerate(int y, int[] queens) {

    if (y >= DIMENSION) return queens;

    for (int x = 0; x < DIMENSION; x++) {
        if (!isQueenThreatened(x, y, queens)) {
            int[] nextQueens = queens.clone();
            nextQueens[y] = x;
            int[] solution = enumerate(y + 1, nextQueens);
            if (solution != null) return solution;
        }
    }

    return null;
}
```

Jorin Vogel, Elias Klemm