N Queens Problem & Solution

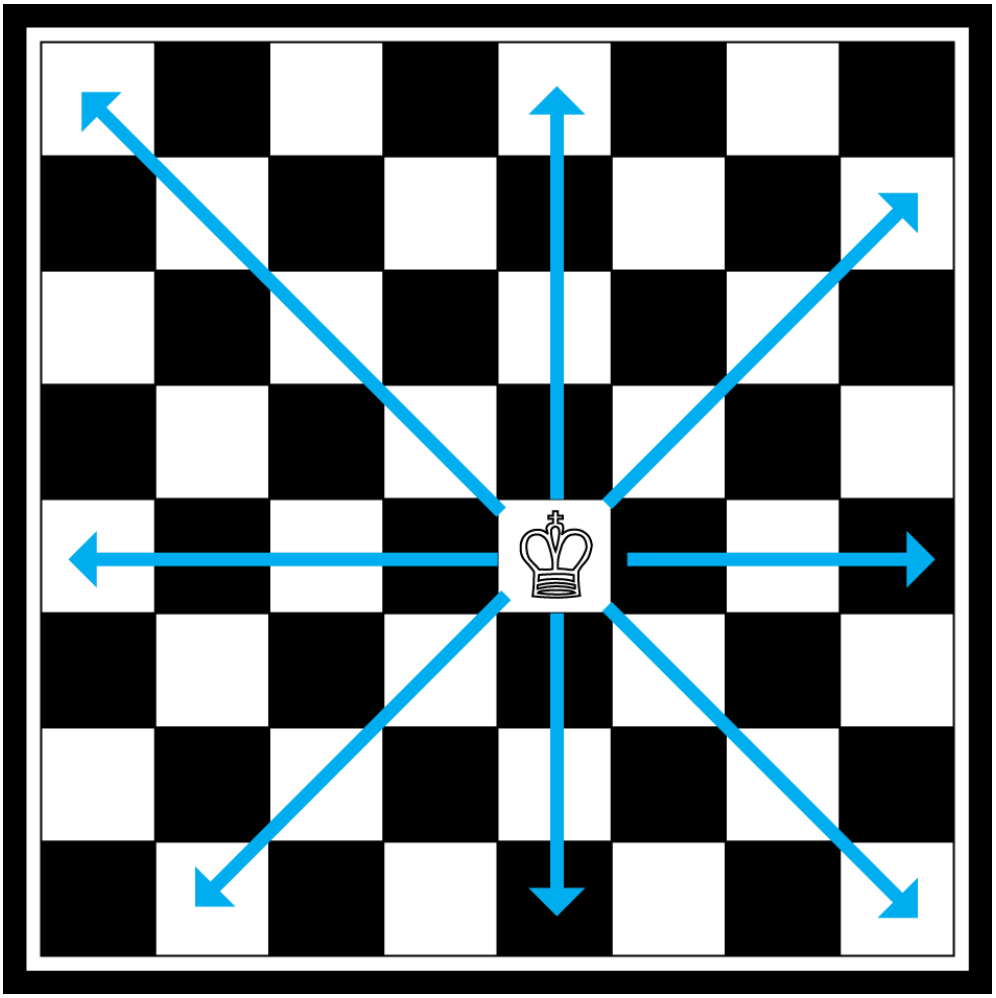

Figure : Queen’s moves

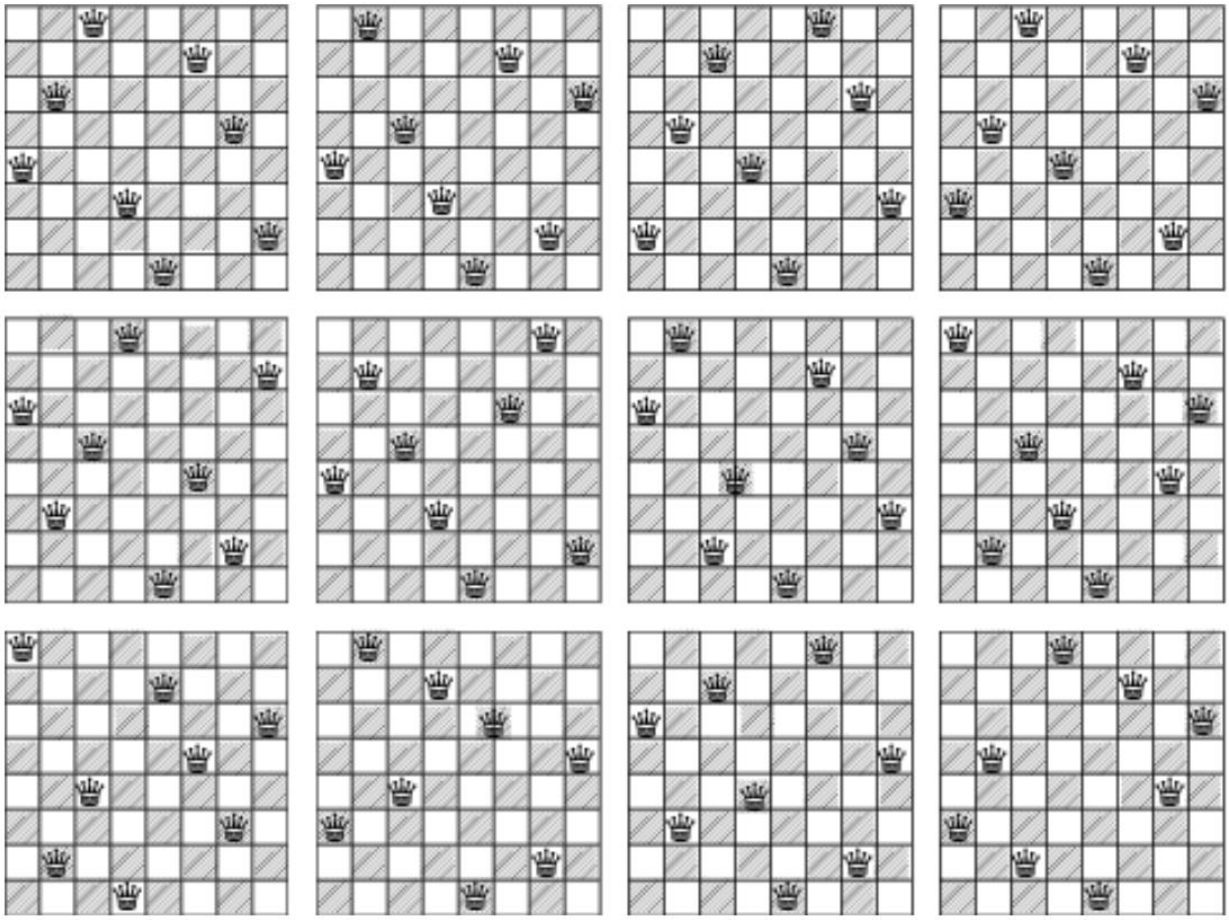

Figure : 12 Unique solutions for 8*8 chessboard

Solution

Place queen column by column from left to right, try each row one by one, making sure no two queens attack each other

Q			

Q			
	Q		

Q		X	
		X	
	Q	X	
		X	

(Time for Backtrack)

Backtracking – When we can't place a Queen in column j, change the place of the Queen placed in previous column j-1

Q			
	Q		

Q			
		Q	
	Q		

Q			X
		Q	X
			X
	Q		X

(Time for Backtrack - which will eventually result in change in position of Queen in the first column)

Q			

Q			
	Q		

		Q	
Q			
	Q		

		Q	
Q			
			Q
	Q		

```
public class NQueensProblem {
	private int[][] chessBoard;
	private int numOfQueens;

	public NQueensProblem(int numOfQueens) {
		this.chessBoard = new int[numOfQueens][numOfQueens];
		this.numOfQueens = numOfQueens;
	}

}
```

```
private boolean isPlaceValid(int rowIndex, int colIndex) {
   // Check the columns of the same row, only on left side
   for (int i = 0; i < colIndex; i++)
		if (chessBoard[rowIndex][i] == 1)
			return false;
   // Check upper diagonal, only on left side
   for (int i = rowIndex, j = colIndex; i >= 0 && j >= 0; i--, j--) {
		if (chessBoard[i][j] == 1)
			return false;
   }
   // Check lower diagonal, only on left side
   for (int i = rowIndex, j = colIndex; i < chessBoard.length && j >= 0; i++, j--) {
		if (chessBoard[i][j] == 1)
			return false;
   }
   return true;
}
```

		Q	
Q			
			Q
	Q		

```
private boolean placeQueen(int colIndex) {
		if (colIndex == numOfQueens) {
			return true;
		}
		for (int rowIndex = 0; rowIndex < numOfQueens; rowIndex++) {
			if (isPlaceValid(rowIndex, colIndex)) {
				chessBoard[rowIndex][colIndex] = 1;
				if (placeQueen(colIndex + 1)) {
					return true;
				}

				// BACKTRACKING !!!
				chessBoard[rowIndex][colIndex] = 0;
			}
		}

		return false;
}
```

placeQueen(0)

Q			

placeQueen(0) ---------→ placeQueen(1)

Q			
	Q		

placeQueen(0) ----------→ placeQueen(1) ---------→ placeQueen(2)

We can not place the Queen in 3rd column (colIndex = 2), so placeQueen(2) call will return false to placeQueen(1), which means we will backtrack and change the place of the Queen in 2nd column

Q			
	Q		

placeQueen(0) ----------→ placeQueen(1) ---------→ placeQueen(2)

Q			
		Q	
	Q		

placeQueen(0) ----------→ placeQueen(1) ---------→ placeQueen(2) -------→ placeQueen(3)

We can not place the Queen in 4th column (colIndex = 3), so placeQueen(3) call will return false to placeQueen(2), which means we will have to backtrack.

Now placeQueen(2) will return false to placeQueen(1) and then placeQueen(1) will return false to placeQueen(0). So eventually Queen in the 1st column will be placed in 2nd row.

Q			

placeQueen(0) ---------→ placeQueen(1)

Q			
	Q		

placeQueen(0) ---------→ placeQueen(1) -------→ placeQueen(2)

		Q	
Q			
	Q		

placeQueen(0) ---------→ placeQueen(1) -------→ placeQueen(2) ---------→ placeQueen(3)

		Q	
Q			
			Q
	Q		

placeQueen(0) -----→ placeQueen(1) ---→ placeQueen(2) -----→ placeQueen(3) → placeQueen(4)

Since colIndex (4) is equal to number of queens, placeQueen(4) will return true, then placeQueen(3) will return true, then placeQueen(2), then placeQueen(1) and finally placeQueen(0), and then we print the chessboard.

Note : For 2*2 and 3*3 chessboards there are no solutions.