The eight queens puzzle as an exercise in algorithm design

(Wikipedia.com, N-Queens Problem)

Finding all solutions to the eight queens puzzle is a good example of a simple but nontrivial problem. For this reason, it is often used as an example problem for various programming techniques, including nontraditional approaches such as constraint programming, logic programming or genetic algorithms. Most often, it is used as an example of a problem which can be solved with a recursive algorithm, by phrasing the n queens problem inductively in terms of adding a single queen to any solution to the problem of placing \(n-1\) queens on an \(n\)-by-\(n\) chessboard. The induction bottoms out with the solution to the 'problem' of placing 0 queens on an \(n\)-by-\(n\) chessboard, which is the empty chessboard.

This technique is much more efficient than the naïve brute-force search algorithm, which considers all \(64^8 = 2^{48} = 281,474,976,710,656\) possible blind placements of eight queens, and then filters these to remove all placements that place two queens either on the same square (leaving only \(64!/56! = 178,462,987,637,760\) possible placements) or in mutually attacking positions. This very poor algorithm will, among other things, produce the same results over and over again in all the different permutations of the assignments of the eight queens, as well as repeating the same computations over and over again for the different sub-sets of each solution. A better brute-force algorithm places a single queen on each row, leading to only \(8^8 = 2^{24} = 16,777,216\) blind placements.

It is possible to do much better than this. One algorithm solves the eight rooks puzzle by generating the permutations of the numbers 1 through 8 (of which there are \(8! = 40,320\)), and uses the elements of each permutation as indices to place a queen on each row. Then it rejects those boards with diagonal attacking positions. The backtracking depth-first search program, a slight improvement on the permutation method, constructs the search tree by considering one row of the board at a time, eliminating most nonsolution board positions at a very early stage in their construction. Because it rejects rook and diagonal attacks even on incomplete boards, it examines only 15,720 possible queen placements. A further improvement which examines only 5,508 possible queen placements is to combine the permutation based method with the early pruning method: the permutations are generated depth-first, and the search space is pruned if the partial permutation produces a diagonal attack. Constraint programming can also be very effective on this problem.