## INTRODUCTION

The n-queens problem is to determine Q(n), the number of different ways in which II queens may be placed on an n-by-n chessboard so that no two queens are on the same row, column or diagonal. There appears to be no algorithm whose complexity is known to be better than the brute force approach.

A related problem is the formal n-queens problem. This problem is identical to the (regular) n-queens problem, except that all diagonals are of length n and wrap as if the chessboard were on a torus. If we denote the number of solutions to the toroidal problem as T(n), it is obvious that T(n) < Q(n). In this paper we describe a simple algorithm for computing Q(n) or T(n) in time O(f(n>8”>, where f(n) is a low-order polynomial. The algorithm is based on dynamic programming. which views certain search problems as integer linear programming problems and solves them via polynomial multiplication.

## Algorithm

Our algorithm consists of performing dynamic programming. We will perform a breadth-first search of the feasible candidates, where candidates with the same set of closed lines will be placed in an equivalence class. The data structure we need is a set of tuples (S, i), where S is a set of (closed) lines and i is an integer. This will be used to represent an equivalence class of i feasible candidates whose set of closed lines is S.

**ALGORITHM Queens**

1. [Initialization] Set QUEUE to {(PI, 1)).
2. [Square selection] Choose an unexamined square. Let T be the set of four lines containing this square.
3. [Iteration] For every tuple (S, i) E QUEUE such that S n T = 0 do the following.
   [Compaction] If (S U T, j) E QUEUE for some j, replace j by i + j.
   [Creation] Otherwise, add (S U T, i> to QUEUE.
4. [Termination] If an unexamined square remains, go to step 2. Otherwise, terminate.

Determine the number of solutions from QUEUE. First, define A4 to be the set of 2n lines that are At the end of this algorithm we can easily either rows or columns. A solution places a queen on each line in M.

## Conclusion

We have described an 0(f(n>8n) algorithm for the n-queens problem. There is some evidence that the number of solutions to the problem is super-exponential. If this is true, then our algorithm is superior to any approach (such as backtracking) that explicitly constructs all solutions to the problem.

## References

https://www.researchgate.net/publication/220191466_A_survey_of_known_results_and_research_areas_for_-queens

https://www.cs.cornell.edu/~rdz/Papers/RZ-IPL92.pdf