N-Queen Problem

MIT2018087

15 November 2018

1 Problem Statement

The n-queens problem is to determine Q(n), the number of different ways in which n queens may be placed on an n x n chessboard so that no two queens are on the same row, column or diagonal.

2 Introduction

N-Queen problem is a typical combination optimization problem and it belongs among NP-hard problems. The problem has a long history. The 8 x 8 case was posed in an anonymous problem in 1848.

A simple algorithm is discussed for computing Q(n) or T(n) in time $O(f(n)8^n)$, where f(n) is a low-order polynomial. The algorithm is based on dynamic programming. The eight queens puzzle has 92 distinct solutions. If solutions that differ only by the symmetry operations of rotation and reflection of the board are counted as one, the puzzle has 12 solutions.

3 N-Queens Problem (NQP)

The problem is to place N-Queens on a chessboard so that no two attack each other. Since each queen must be on a different row and column, we can assume that queen 'i' is placed in i^{th} column. All solutions to the NQP can therefore be represented as n-tuples $(q_1, q_2, ..., q_n)$ that are permutations of an n-tuple (1, 2, 3,., n). Position of a number in the tuple represents queen's column position, while its value represents queen's row position (counting from the bottom) using this representation, the solution space where two of the constraints (row and column conflicts) are already satisfied should be searched in order to eliminate the diagonal conflicts.

widely used as a benchmark because of its simple and regular structure Problem involves placing. Benchmark code versions include finding the first solution and finding all solutions. Input for this System: A positive integer 'n'. Task for this system: place ‘n’ Queens on an n by n chessboard so that no two Queens attack each other (on same row, column, diagonal), or report that this is impossible. Solving particular problem for the different values of n = 1, 2, 3, 4...n

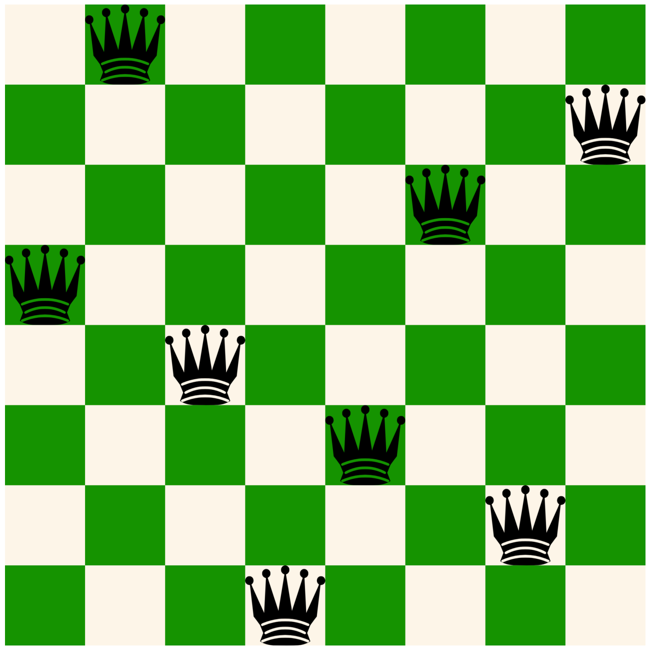


Figure 1: Chess board of size 8 X 8 with 8 queens

4 Algorithmm and analysis

The idea is to place queens one by one in different columns, starting from the leftmost column. Our algorithm consists of performing dynamic programming using
Theorem: ”Suppose C, and C, are feasible candidates having the same set of closed lines. Then any completion of C, is also a completion of G. ”

where, A candidate C consists of a placement of at most n queens on an n-by-n chessboard. A candidate is feasible if no two queens attack each other, i.e., there is no line containing two queens. A completion of a candidate is a placement of the remaining queens that results in a solution.

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

4.1 Algorithm Queens

1. Initialization: Set QUEUE to (ϕ, 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) ϵ QUEUE such that S n T = ϕ do the following.
   (a) Compaction: If (S U T, j) ϵ QUEUE for some j, replace j by i + j.
   (b) Creation: Otherwise, add (S U T, i) to QUEUE.
4. Termination: If an unexamined square remains, go to step 2. Otherwise, terminate.

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

σ(S, i) = (i, if M $\subseteq$ S. 0, otherwise)

After our algorithm has terminated, the number of solutions is: Σ σ (S, i)

5 Analysis

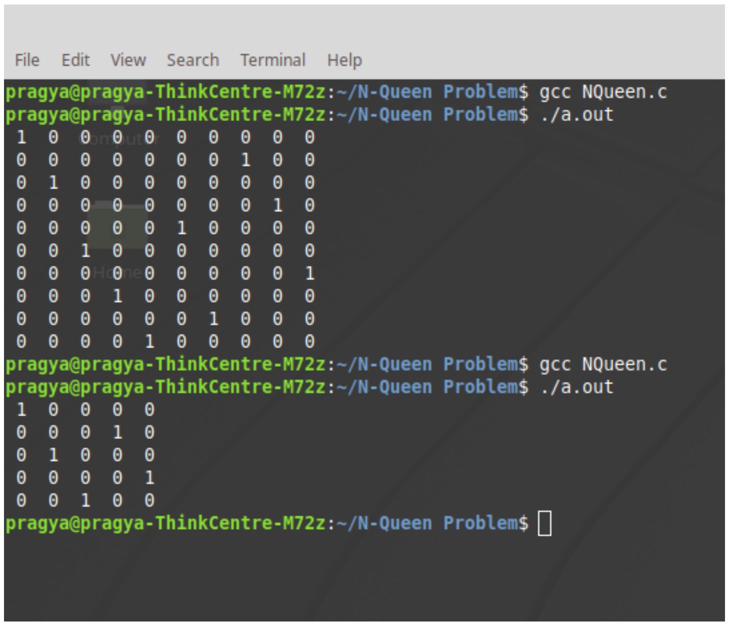
The space requirements of algorithm depend on the maximum size of QUEUE at any point in the algorithm, which is the possible number of sets of closed lines. If there are α possible closed lines, we will need to store 2^α equivalence classes of feasible candidates. QUEUE can be represented as an array of 2^α elements. The number of possible candidates is $0(2^\alpha)$, so the size of an element in the array is $O(n^2)$, and the space necessary for our algorithm is $O(n^2 2^\alpha)$.

The time requirements of our algorithm also depend on α. Step 3 is executed n^2 times, once for each square on the board. Each execution of this step takes time $O(g(n)2^\alpha)$, where g(n) comes from the complexity of integer arithmetic. So the total running time of our algorithm is $O(f(n)2^\alpha)$, where $f(n) = n^2 g(n)$ is a low-order polynomial.

6 Results of implementation

The following image is the result of the program that is written in C programming language and the algorithm used is the one that is mentioned above.

In the image 1 represents the queen placed in that particular row and 0 represents the other positions in the chess board. The first output is for 10 X 10 chess board and the second one is for 5 X 5 chess board.



```
pragya@pragya-ThinkCentre-M72z:~/N-Queen Problem$ gcc NQueen.c
pragya@pragya-ThinkCentre-M72z:~/N-Queen Problem$ ./a.out
 1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  1  0  0
 0  1  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  1  0
 0  0  0  0  0  1  0  0  0  0
 0  0  1  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  1
 0  0  0  1  0  0  0  0  0  0
 0  0  0  0  0  0  1  0  0  0
 0  0  0  0  1  0  0  0  0  0
pragya@pragya-ThinkCentre-M72z:~/N-Queen Problem$ gcc NQueen.c
pragya@pragya-ThinkCentre-M72z:~/N-Queen Problem$ ./a.out
 1  0  0  0  0
 0  0  0  1  0
 0  1  0  0  0
 0  0  0  0  1
 0  0  1  0  0
pragya@pragya-ThinkCentre-M72z:~/N-Queen Problem$ 
```

Figure 2: Output of implementation

7 Conclusions

For the n-queens problem, O(f(n)2^{α}) algorithm . There is some evidence that the number of solutions to the problem is super-exponential. If this is true, then the algorithm is superior to any approach (such as backtracking) that explicitly constructs all solution to the problem.

8 References

-Igor Rivin and Ramin Zahib "A dynamic programming solution to the n-queens problem", Vol 41 p. 5

- "https://en.wikipedia.org/wiki/Eightqueenspuzzle"