Notes on N-Queens Solutions

Robert Dougherty-Bliss

February 29, 2016

The n-queens problem is this: Given an $n \times n$ chessboard, can n queens be placed on it such that no queen can attack another? The answer, proved by E. Pauls in 1874, is yes, for $n = 1$ and $n > 3$ [1]. We list some notes on the running times of various solutions.

Running Times

Here we will list various solutions to the n-queens problem and compare their running times. Included with each are pseudocode algorithms.

Table 1: Running Time of n-Queens Solutions

Combinatoric Brute Force	$O(n(n^2!))$
Row-based Brute Force	$O(n^{n+1})$
Backtracking	$O(n!)$
Explicit Solution	$O(n)$

Guarded Check

Some of the solutions check if a given answer is a solution. Here is an $O(n^2)$ algorithm to check if this is true.

First, some definitions. The $n \times n$ chessboard is represented as $[0, n-1] \times [0, n-1]$, indexed by row-column points (r, c). The row $r = 0$ is the top of the board, and the column $c = 0$ is the leftmost column.

The kth sum diagonal on a chess board is the set of points (r, c) on the board such that $r + c = k$. The kth difference diagonal is the set of points (r, c) on the board such that $r - c = k$. Sum diagonals run from bottom-left to top-right, and difference diagonals run from top-left to bottom-right.

Queens on a chessboard guard their rows, columns, and diagonals. That motivates the definition of a *guarded point*. Given a point $P(r, c)$ and a set of queens Q, the point P is guarded by Q iff there exists some $R(a, b) \in Q$ such that R and P share a column, row, or diagonal. That is, iff $a + b = r + c$, $a - b = r - c$, $a = r$, or $b = c$. In the worst-case for a set of n queens, the running time of checking if a point is guarded is $O(n)$. To check if an answer is a solution, simply perform this check against all n points, which is $O(n)O(n) = O(n^2)$.

```
Input: $P(r,c)$, a point, and $Q$, a set of queens
Output: True if $P$ is guarded by $Q$, False if not
foreach $(a,b)$ in $Q$ do
    if $a+b=r+c$ or $a-b=r-c$ then
        return True;
    end
    if $a=r$ or $b=c$ then
        return True;
    end
end
return False;
```

Algorithm 1: Guarded Check

Combinatoric Brute Force

A first attempt to solve the n-queens problem might be to try every possible combination of n distinct points from an $n \times n$ chessboard. This gives

$$\binom{n^2}{n} = \frac{n^2!}{n!(n^2-n)!}$$

possible combinations. For all $n \geq 1$, we have that $n^2 - n \geq 0$, so $n!(n^2-n)! \geq 1$. So for all $n \geq 1$,

$$\frac{n^2!}{n!(n^2-n)!} \leq n^2!$$

That is, to enumerate every possible combination is $O(n^2!)$. But we also need to check if each combination is a solution, which is $O(n)$. So the total worst-case running time of this solution is $O(n^2!)O(n^2) = O(n^2(n^2!))$.

Row-Based Brute Force

An improvement on the previous technique is to note that no solution has two queens placed in the same row. Then, restrict the queens from being placed in the same row. There are n choices for the first row, n for the second, and so on until the nth row. That gives

$$\underbrace{n \times n \times n \times \cdots \times n}_{n \text{ times}} = n^n$$

combinations to check. To enumerate all of these is $O(n^n)$, but we also need to check each, which is $O(n)$. Thus the worst case running time is $O(n^n)O(n^2) = O(n^{n+2})$. The solution algorithm is identical except for the method of generating combinations.

Backtracking

Another improvment is to use backtracking. This method starts by placing a queen at $(0,0)$, then moving to the next row. Once a non-guarded point is found on that row, place a queen there, and move to the next row. Do this until there are n queens placed.

```
Input: $n$, a non-negative integer
Output: A set of $(r, c)$ points that solve the $n$-queens problem, or an empty set if no
        solution exists
combinations $\leftarrow [0, n-1] \times [0, n-1]$;
foreach answer in combinations do
    solved $\leftarrow$ True;
    foreach point in answer do
        if point is guarded by rest of answer then
            solved $\leftarrow$ False;
            break;
        end
    end
    if solved then
        return answer;
    end
end
```

Algorithm 2: Combinatoric Brute Force

If every point on a row is guarded, then move back one row and delete the queen there. Then attempt to find another non-guarded point in that row. If there are none, move back another row, and attempt to delete and move the queen there. If the queen in the first row is moved past the end of its row, then there is no solution possible.

In the worst case, this will search roughly n points in the first row, then $n-1$ in the next, and so on, so the worst-case running time is $O(n!)$.

```
Input: $n$, a non-negative integer
Output: A set of $(r, c)$ points that solve the $n$-queens problem, or an empty set if no
        solution exists
place queen at $(0, 0)$;
(row, col) $\leftarrow$ (0, 0);
while number of queens $\neq n$ do
    if (row, col) is not guarded then
        place a queen at (row, col);
        (row, col) $\leftarrow$ (row + 1, 0);
    end
    else
        while $col = n - 1$ do
            if $row = 0$ then
                /* End of first row.                                              */
                return no solution
            end
            /* End of current row; backtrack until we can move the previous
               queen.                                                             */
            (row, col) $\leftarrow$ previous queen;
            delete previous queen;
        end
        col $\leftarrow$ col + 1;
    end
end
```

Algorithm 3: Backtracking

Explicit Solution

This solution is due to Pauls, discussed in [1]. Pauls breaks down $n > 3$ by congruence classes modulo 6, then offers a solution for each of them. This requires one comparison, and then the creation of a number of sets that are joined together. The number of elements in the sets are linear in n, and so this solution is $O(n)$. Note that for Pauls the board is $[1, n] \times [1, n]$ instead of $[0, n-1] \times [0, n-1]$. An outline of Pauls proof is given by [1]. Another soluion and proof is given by [2].

```
Input: $n$, a non-negative integer
Output: A set of $(r, c)$ points that solve the $n$-queens problem, or an empty set if no
        solution exists
if $n = 1$ then
    return $\{(1, 1)\}$
end
if $n = 2, 3$ then
    return $\emptyset$;
end
switch $n \bmod 6$ do
    case 0, 4 do
        $A1 = \{(2k, k) : 1 \leq k \leq n/2\}$;
        $A2 = \{(2k-1, n/2+k) : 1 \leq k \leq n/2\}$;
        return $A1 \cup A2$;
    end
    case 1, 5 do
        $B1 = \{(n, 1)\}$;
        $B2 = \{(2k, k+1) : 1 \leq k \leq (n-1)/2\}$;
        $B3 = \{(2k-1, (n+1)/2+k) : 1 \leq k \leq (n-1)/2\}$;
        return $B1 \cup B2 \cup B3$;
    end
    case 2 do
        $C1 = \{(4, 1)\}$;
        $C2 = \{(n, n/2-1)\}$;
        $C3 = \{(2, n/2)\}$;
        $C4 = \{(n-1, n/2+1)\}$;
        $C5 = \{(1, n/2+2)\}$;
        $C6 = \{(n-3, n)\}$;
        $C7 = \{(n-2k, k+1) : 1 \leq k \leq n/2-3\}$;
        $C8 = \{(n-2k-3, n/2+k+2) : 1 \leq k \leq n/2-3\}$;
        return $C1 \cup C2 \cup C3 \cup C4 \cup C5 \cup C6 \cup C7 \cup C8$;
    end
    case 3 do
        return $(n, n) \cup$ (solution for $n-1$);
    end
end
```

Algorithm 4: Pauls' Explicit Solution

References

[1] Bell, Jordan, and Stevens, Brett, 2007: A survey of known results and research areas for n-queens, *Discrete Math*, **309**, 1–31.

[2] E.J. Hoffman, and J.C. Loessi, and R.C. Moore, 1969: Constructions for the Solution of the m Queens Problem, *Math. Mag.*, **42**, 66–72