Domino Tiling an N-Queens Chessboard

Christopher Sakaliyski[*]

Advisor
Gene Li[†]

November 24, 2022

Abstract

Every avid chess player knows the n-queens puzzle and every recreational mathematician finds both interesting and challenging the problem of calculating all possible domino tiling arrangements of a chessboard. In this paper, the two problems are combined and calculation of the number of domino tiling arrangements of solutions to the n-queens puzzle are performed. Based on results presented for $n \leq 15$, it is hypothesized that as long as the necessary condition of having equal white and black squares left in a given n-queen solution, the remaining chessboard will be tileable. This is not always true for other mutilated chessboards, as the mutilations could split the board in small disjoint groups. In the example of n-queens, the queens, or mutilations, are spread out so far apart that the entire board is connected. The number of black and white square being equal seems to sometimes occur when $n \equiv 0, 1 \mod 4$. On the other hand, for $n \equiv 2, 3 \mod 4$, none of the placements of queens in the n-queens solutions satisfies this property of equivalence.

[*]Whitney M. Young Magnet High School
[†]Toyota Technological Institute

Contents

1 Introduction

In this paper, the n-queens puzzle solutions and the corresponding 1x2 tiling arrangements are observed and calculation of the number of domino tiling arrangements of solutions to the n-queens puzzle are performed. Based on results presented for $n \leq 15$, it is hypothesized that as long as the necessary condition of having equal white and black squares left in a given n-queen solution, the remaining chessboard will be tileable. This is not always true for other mutilated chessboards, as the mutilations could split the board in disjoint groups. In the example of n-queens, the queens, or mutilations, are spread out so far apart that the entire board is connected. The number of black and white squares being equal seems to sometimes occur when $n \equiv 0, 1 \mod 4$. On the other hand, for $n \equiv 2, 3 \mod 4$, none of the placements of queens in the n-queens solutions satisfies this property of equivalence.

In Section 1 and 2 the two problems are introduced and defined. In Section 3, the programs to solve each problem are presented. Sections 4 describes the idea behind the mutilated chessboard. Section 5 presents the pseudo code for combining the two problems. Observations and Conclusion are presented in Sections 6 and 7, respectively.

1.1 N-Queens Problem

The problem for $n = 8$ was first introduced by the chess composer Max Bazzel in 1848 [Bal60]. Two years later, in 1850, Franz Nauck published his solutions in the Leipziger Illustrierte Zeitung and expanded the problem by changing it to n-queens [EST92]. Although there are $\binom{64}{8} = 4,426,165,368$ ways to place 8 queens on an 8 by 8 chessboard without restrictions, only 92 of these are solutions to the problem. The number of fundamental solutions has only been proved to be 12 by English mathematician J. W. L. Glaisher [EST92]. Many famous mathematicians such as Gauss worked on this problem.

In 2021, Michael Simkin proved that for large numbers of n, the number of solutions is approximately $(0.143n)^n$, which grows asymptotically [Sim21]. More specifically, the formula is

$$\mathcal{Q}(n) = ((1 \pm o(1))ne^{-\alpha})^n$$

1.2 Domino Tiling

The 1x2 domino tiling problem originates from Tatami, which are Japanese floor mats in the shape of a 1x2 rectangle. These mats are used to tile rooms, but with a bit more restriction on the tiling. An example of this is shown in Figure 2.

The formula for 1x2 domino tiling arrangements for board m by n was discovered independently by Temperley and Fischer [TF61], and Kasteleyn [Kas61], both in 1961:

$$\prod_{j=1}^{\lceil \frac{m}{2} \rceil} \prod_{j=1}^{\lceil \frac{n}{2} \rceil} \left(4\cos^2 \frac{\pi j}{m+1} + 4\cos^2 \frac{\pi k}{n+1} \right)$$

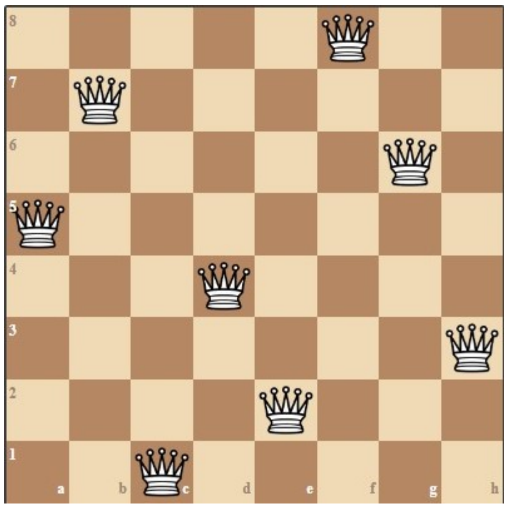
2 Definition

2.1 N-Queens

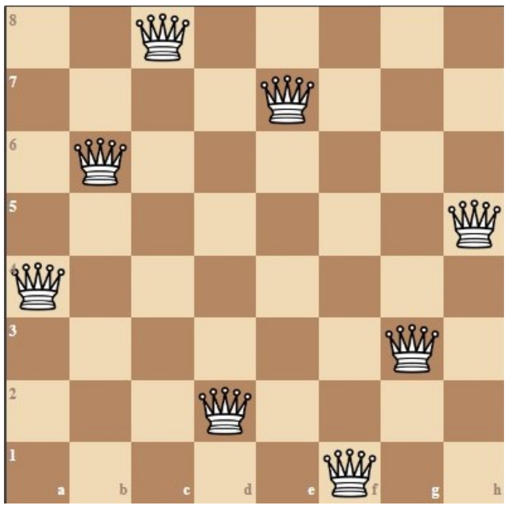
This puzzle goes as follows. Place n queens on an n by n board such that no queens are attacking each other. Queens move horizontally, vertically, and diagonally. Thus, queens will attack each other if they are in the same row, column, or diagonal. Along with regular solutions, the fundamental solutions are all the solutions which are not rotations or reflections of each other. There are about $\frac{1}{8}$ as many fundamental solutions due to there being 4 rotations and 4 reflections of a placement of queens [EST92]. These rotations and reflections are:

- Rotation 90° degrees clockwise
- Rotation 180° degrees clockwise
- Rotation 270° degrees clockwise
- Rotation 360° degrees clockwise
- Reflection about the horizontal axis
- Reflection about the vertical axis
- Reflection about the diagonal containing points (1, 1) and (8, 8)
- Reflection about the diagonal containing points (1, 8) and (8, 1)

Two possible solutions are shown in Figure 1a and Figure 1b.



(a) Solution of 8-queens Problem That Has no Three Queens Are in a Straight Line



(b) Solution of 8-queens Problem in a Staircase Pattern

Figure 1: Two solutions of 8-queens

2.2 Domino Tiling

This problem goes as follows. On an n by n board, with the number of squares being a multiple of 2, place 1x2 non-overlapping tiles to fill up the board. How many ways are there to fill up the board? A form of this is the mutilated tiling, in which some squares are required to be left out and the rest be tiled. This can be seen later when the two problems are combined.

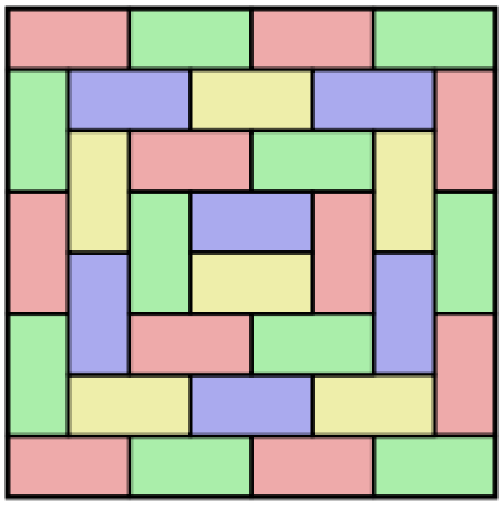


Figure 2: Tatami Tiling of an 8x8 square [Wik22]

3 Programs

3.1 N-Queens Program

Although brute-force always works, it is extremely slow. Checking all $\binom{64}{8} = 4,426,165,368$ placements of queens is wildly inefficient. For comparison, C++ can perform 10^8 operations per second. For $n = 8$, it will take 44 seconds to run. Using a brute-force approach on larger n will be too inefficient and will take lots of time.

Instead of brute force, the method used in this program is called backtracking. Backtracking is used to search for all possibilities one step at a time. It checks the constraints every time, and if the constraints are not met, it backtracks and goes one step back. If the constraints are met than it goes to the next step. This is significantly faster, as the steps for the n-queens algorithm just place one queen in every column. This gives about $n!$ placements to check, which is much more feasible than $\binom{n^2}{n}$.

Our constraints are that no two queens are on the same row, column, or diagonal. The program starts at the first column of the chessboard. At every column, the program tries to place a queen in that column. For every square in that column, the program checks if the queen can be placed with the constraints being followed. This means that the new queen is not on the same row or diagonal as previous queens. Columns do not need to be checked due to every step being on different columns. If the queen can be placed in that column, then it is placed and the program moves onto the next column. After the program is done going through the rest of the columns, it goes back and removes the queen it placed. The program does the same thing for the next square in the column, and so on.

```
placeQueens(current column)
{
    if current column is n
    {
        add one to counter of solutions
        break
    }
    for all squares in the current column
    {
        check if the square is attacked by any queens in the same row
        check if the square is attacked by any queens on both diagonals
        if the square is not attacked
        {
            place the queen
            placeQueens(next column)
            remove the queen
        }
    }
}
```

In addition to the algorithm for all solutions, there exist an algorithm for finding only the fundamental solutions. Rodney Topor, optimized and simplified the algorithm presented by Fillmore and Williamson [Top82] by using orbits and groups to generate the fundamental solutions.

3.2 Tiling Program

In the tiling program, instead of repeatedly trying to place tiles by brute-forcing, the program goes step by step and backtracks. The program goes through the rows, starting from the top left on the top most row and ending at the bottom right on the bottom row. There are two types of tiles that can be placed, tiles that go left-right, and tiles that go up-down.

The program starts at the first square. On the current square, it checks if a tile can be placed such that it covers the current square and the square to the right. If it can, it places the tile and goes on to the next square with this same process. If it cannot, checks if a tile can be placed such that it covers the current square and the square below. Additionally, a boolean array is used to know whether a square is already covered by a tile or not, and it can be toggled when tiles get placed. A tile always needs to be placed on the current square because later steps cannot place a tile above or to the left of the square they are on. The program ends when the next square it lands on is not on the chessboard as that is when every square is filled.

```
placeTiles(current x, current y)
{
    if all tiles have been placed
    {
        add one to counter of solutions
        break
    }
    if the current square has been covered
    {
        placeTiles(next x, next y)
    }
    set the tile to be covered in the boolean array
    if it is possible to place a left-right tile
    {
        place the left-right tile
        placeTiles(next x, next y)
        remove the tile
    }
    if it is possible to place an up-down tile
    {
        place the up-down tile
        placeTiles(next x, next y)
        remove the tile
    }
    set the current square to not be covered
}
```

4 Mutilated Chessboards

A form of a chessboard is a mutilated chessboard. In this chessboard, some of the squares are missing. The most common example of tiling a mutilated chessboard is for an 8 by 8 chessboard with two opposite corners cut off. A chessboard is made up of alternating black and white squares. This fact can be used to see that every tile covers 1 black square and 1 white square. Removing two opposite squares gives us 30 of one color square and 32 of the other color square, making it impossible to place the tiles where the number of squares of the colors are equivalent.

When cutting of corners that are next to each other, 1 is subtracted from the amount of each color square as those two corners are off different color. This gives us 31 of each color, which requires placing 31 tiles in the remaining squares.

A mutilated chessboard can never be tiled if the number of black and white squares are nonequivalent. The contrapositive of this statement is not always true, as empty squares could split the board into groups which have an odd number of tiles. In the case of n-queens, the empty squares, or the squares on which the queens are placed, are spread far enough apart so that the whole board is connected as a group.

5 Combining the Programs

When combining the programs, the new program generates the queens as usual, but the tiling is a bit different. Instead of tiling a regular chessboard, the program needs to tile a mutilated chessboard. Sometimes, the mutilated board cannot be tiled, in which the number of black and white tiles are nonequivalent. To make the board mutilated, the squares of the queens need to be marked as covered. Here is the pseudo code:

```
placeTiles(current x, current y)
{
    if all tiles have been placed
    {
        add one to counter of solutions
        break
    }
    if the current square has been covered
    {
        placeTiles(next x, next y)
    }
    set the tile to be covered in the boolean array
    if it is possible to place a left-right tile
    {
        place the left-right tile
        placeTiles(next x, next y)
        remove the tile
    }
    if it is possible to place an up-down tile
    {
        place the up-down tile
        placeTiles(next x, next y)
        remove the tile
    }
    set the current square to not be covered
}
placeQueens(current column)
{
    if current column is n
    {
        if number of black squares = number of white squares
        {
            placeTiles(0, 0)
        }
        break
    }
    for all squares in the current column
    {
        check if the square is attacked by any queens in the same row
        check if the square is attacked by any queens on both diagonals
        if the square is not attacked
        {
            place the queen
            mark the square as covered
            placeQueens(next column)
            remove the queen
            mark the square as not covered
        }
    }
}
```

6 Observations

The exact number of solutions for placing n queens on n by n board are computed and presented in Table 1.

From the results in Table 1, it can be hypothesized that:

- $n \equiv 0, 1 \mod 4$: Some but not all arrangements of queens in this case can be tiled.
- $n \equiv 2, 3 \mod 4$: None of the arrangements of queens in this case can be tiled.

For a given n by n chessboard, the number of white squares is $\lceil \frac{n^2}{2} \rceil$, the number of black squares is $\lfloor \frac{n^2}{2} \rfloor$, and the number of all squares is n^2. A necessary condition to have a domino tiling is the number of white squares be equal to the number of black squares. If n is even, then the number of black and white queens will need to be $\frac{n}{2}$ each. If n is odd, then the number of white queens needs to be $\frac{n+1}{2}$ and the number of black queens needs to be $\frac{n-1}{2}$. Using these facts, a Table 2 can be made for different $n \mod 4$.

n	All solutions	Tileable Solutions
1	1	0
2	0	0
3	0	0
4	2	2
5	10	8
6	4	0
7	40	0
8	92	92
9	352	304
10	724	0
11	2,680	0
12	14,200	12,996
13	73,712	58,392
14	365,596	0
15	2,279,184	0

Table 1: Exact Number of Solutions for Placing n Queens on n by n Board

n	All Squares	White Squares	Black Squares	White Queens	Black Queens
$4k$	$16k^2$	$8k^2$	$8k^2$	$2k$	$2k$
$4k+1$	$16k^2+8k+1$	$8k^2+4k+1$	$8k^2+4k$	$2k+1$	$2k$
$4k+2$	$16k^2+16k+4$	$8k^2+8k+2$	$8k^2+8k+2$	$2k+1$	$2k+1$
$4k+3$	$16k^2+24k+9$	$8k^2+12k+5$	$8k^2+12k+4$	$2k+2$	$2k+1$

Table 2: Exact Number of White/Black Queens and Squares When Placing n Queens on n by n board, to be 1x2 Tileable

The combined program is ran to obtain all solutions and tiling arrangements when possible. The program checks if the conditions in Table 2 are satisfied before trying to tile the chessboard. For each $1 \leq n \leq 15$, the program outputs the solutions and tiling arrangements frequency. Also, the program shows the number of white/black queens along with the possible numbers of remaining white and black squares. It shows the number of arrangements of n queens that satisfy these properties. The results for $n = 8$ are shown in Table 3 and 4, while results for $n = 15$ are displayed in Table 5 and 6.

Tiling Arrangements	Frequency
382	8
653	8
3,806	8
5,271	8
6,378	8
6,705	8
7,408	8
8,416	8
9,424	8
9,981	8
15,012	8
21,940	4

Table 3: Tiling Arrangements and Frequency on an 8 by 8 chessboard

White Queens	Black Queens	White Squares	Black Squares	Arrangements
4	4	28	28	92

Table 4: Possible Numbers of White and Black Queens Along with the Possible Numbers of Remaining White and Black Squares and Arrangements on an 8 by 8 chessboard.

Tiling Arrangements	Frequency
0	2,279,184

Table 5: Tiling Arrangements and Frequency on an 15 by 15 chessboard

White Queens	Black Queens	White Squares	Black Squares	Arrangements
5	10	108	102	33,912
7	8	106	104	1,739,936
9	6	104	106	504,776
11	4	102	108	560

Table 6: Possible Numbers of White and Black Queens Along with the Possible Numbers of Remaining White and Black Squares and Arrangements on an 15 by 15 chessboard.

7 Conclusion

It seems that as long as the necessary condition of having equal and black squares left in a given n-queen solution, the remaining chessboard will be able to be tiled. This is not always true for other mutilated chessboards, as the mutilations could split the board into not tileable groups. In the example of n-queens, the queens, or mutilations, are spread out so far apart that the entire board is connected. The number of black squares and number of white squares being equal seems to sometimes occur when $n \equiv 0, 1 \mod 4$. On the other hand, for $n \equiv 2, 3 \mod 4$, none of the placements of queens satisfy this property of equivalence.

8 Code and Data

The source code used to solve the two problems and obtain the results in the paper can be found at https://github.com/cks524/QED2022.

The results can be found in an excel sheet with tabs that has the data output from $n \leq 15$.

References

[Bal60] W. W. Rouse Ball. The eight queens problem. *Mathematical Recreations and Essays*, pages 165–171, 1960.

[EST92] Cengiz Erbas, Seyed Sarkeshik, and Murat M. Tanik. Different perspectives of the n-queens problem. In *Proceedings of the 1992 ACM Annual Conference on Communications*, CSC '92, page 99–108, New York, NY, USA, 1992. Association for Computing Machinery.

[Kas61] P.W. Kasteleyn. The statistics of dimers on a lattice: I. the number of dimer arrangements on a quadratic lattice. *Physica*, 27(12):1209–1225, 1961.

[Sim21] Michael Simkin. The number of n-queens configurations, 2021.

[TF61] H. N. V. Temperley and Michael E. Fisher. Dimer problem in statistical mechanics-an exact result. *The Philosophical Magazine: A Journal of Theoretical Experimental and Applied Physics*, 6(68):1061–1063, 1961.

[Top82] Rodney W. Topor. Fundamental solutions of the eight queens problem. *BIT Numerical Mathematics*, 22:42–52, 1982.

[Wik22] Wikipedia. Domino tiling — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Domino%20tiling&oldid=1119365150`, 2022. [Online; accessed 22-November-2022].