

A heuristic based linear time $O(N)$ novel solution to N-Queen problem

Gaurav Kholiya [†]

Vikrant Sharma ^{*}

Satvik Vats [§]

Computer Science & Engineering

Graphic Era Hill University

Dehradun

Uttarakhand

India

Vijay Garg [‡]

School of Computer Science & Engineering

Lovely Professional University

Punjab

India

Abstract

N-Queen problem is a classic puzzle in which ‘N’ number of queens and a chessboard of size $N \times N$ is given. The puzzle is to lay all the given queens on the chessboard in a way that all queens are safe (i.e., no queen on the chessboard is intersecting the other queen). In this article we have proposed an algorithm to solve the N-Queen puzzle in $O(N)$ time-complexity, without affecting the space-complexity.

Subject Classification: 68P05.

Keywords: N-Queen problem, Polynomial time, Exponential time.

1. Introduction

The N-Queens puzzle is a famous problem in mathematics and computer engineering that implies locating ‘N’ number of queens on the

[†] E-mail: gauravkholiya.21011206@gehu.ac.in

^{*} E-mail: vsharma@gehu.ac.in (Corresponding Author)

[§] E-mail: svats@gehu.ac.in

[‡] E-mail: vijay.garg@lpu.co.in

chessboard of dimension $N \times N$ in a way that no queens threatens the other queen [1]. In other words, only a single queen can occupy the same row, diagonal, or column on the board. The challenge is to find a solution for any given value of N (see Figure 1) [2][3]. The 1850s saw one of the earliest mentions of the N-Queens problem, which was originally investigated by chess player Max Bezzel [4][5] [6]. The N-Queens problem can be solved using a backtracking algorithm [7], which is a type of recursive algorithm that incrementally builds solutions while trying out various possibilities and backtracking when a solution is found to be invalid. The complexity (time) of such solution of the N-Queens puzzle is generally considered to be $O(N!)$, which grows very quickly as the chessboard is scaled up. This implies that the magnitude of time needed to find a result, increases exponentially with the count of queens to be placed. The reason for this high time complexity is that the backtracking algorithm used to solve the N-Queens problem generates a search tree with $N!$ possible configurations, each of which must be checked for conflicts. However, the actual execution time of the algorithm can be affected by various factors, such as the execution of the algorithm and the specific instance of the problem [17-20].

In this article, we have proposed a method to solve the N-Queen puzzle/problem with time complexity equal to $O(N)$, without affecting the space complexity. The proposed method out-performs all the existing models for solving the N-Queen Problem.

The remaining part of the article is structured as follows. Section 2 consists of the survey of the existing literature. The proposed model is defined in Section 3. Results are discussed in Section 4 followed by the conclusion and future work in Section 5.

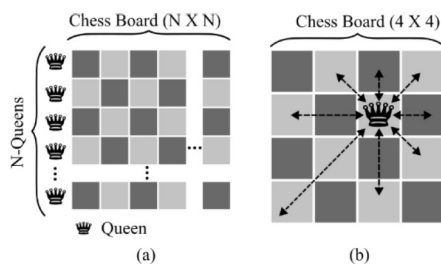


Figure 1

(a) N-Queen Problem (b) Possible directions in which a queen can move on a chess board (Horizontal, Vertical, and Diagonal)

2. Related work

N-Queen puzzle/problem was extensively studied in the past, and traditional techniques such as Backtracking and Brute-Force have been utilized [21-23]. However, it has been found that Brute-Force and backtracking are not a practical answers for the N-Queen puzzle/problem due to its inefficiency, with a worst-case time complexity of $O(N^N)$ and $O(N!)$ respectively [8]. Rok et al. [9] introduced a pair of novel local search algorithms that utilize a gradient-based heuristic and operate in a probabilistic manner. These algorithms have the remarkable capability of solving exceedingly large N-Queens problems.

Authors in [10] came up with the divide and conquer based solution for the N-Queen problem, it visualizes the chess board as multilevel structure formed by the logical grouping of the individual block. At the lowest level the size of the group is limited to the number of blocks which are easily solvable. Then the same solution is propagated to the upper levels. Authors in [11] utilized Ant Colony Optimization (ACO) for solving the N-Queen puzzle. The proposed solution was effective for solving the 8-Queen puzzle and was deemed able of being scaled to larger values of 'N'. It was concluded that ACO offers improved solutions within a reasonable timeframe for combinatorial optimization problems. However, the authors in [12] proposed a solution to the standard N-Queen problem by combining ACO and Genetic Algorithm.

Authors in [13] proposed a solution to the N-Queens problem by employing a Parallel DNA Algorithm (PDA). Drawing inspiration from biological computing models, the DNA molecular operations are utilized to manipulate flexible length DNA strands that represent the allocation matrix elements. Through these techniques the authors claimed the solution of N-Queen problem in $O(N^2)$ time.

Authors in [14] proposed an A novel quantum solver algorithm (NQSA) to produce all possible solutions to the N-Queen problem on quantum computer. The algorithm solves the problem in polynomial time with the time complexity of $O(N^3)$. An extensive work has been done in the past, where the researchers came up with different approaches to efficiently solve the N-Queen puzzle/problem, but none has come up with a linear time solution to the problem [24-26].

3. Proposed model

It is observed that in the arrangement of N-Queens on a chess board (CB[N][N]) of Size N X N there exists several patterns which are scalable. Following are the findings:

The queens can be placed on the chess board in the following patterns.
Each row of the chess board (ranging from 0 to N-1) can accommodate only one queen.

Following pattern exists for the Even value of N.

For 0th row the first Queen Q_0 is placed at index $CB[0][x]$, where

Then for each of the consequent row (i.e., 1, 2, 3, ..., N-1), the position of the placement of the respective Queen is given by the index $CB[row][x]$ where

$$x \leftarrow \begin{cases} x + \frac{N}{2} + 1, & \text{if row is Even} \\ \left(x + \frac{N}{2}\right) \% N, & \text{otherwise} \end{cases}$$

The following pattern is found for the odd value of n.

If N is a multiple of 3 Then,

For each row ranging from row 0 to N/2, the position of the placement of respective queen is given by the index $CB[row][x]$, where $x \leftarrow 0$ initially, and then increments by 2 for each consequent row (this places approximately half of the Queens on the chess board).

For the remaining Queens:

Initializing $x \leftarrow 3$,

If $\left\lfloor \frac{N}{2} \right\rfloor \% 2 \neq 0$, then the position of the placement of Queen for each consequent row (ranging from $\left\lfloor \frac{N}{2} \right\rfloor + 1$ to $N - 1$) is given by the index $CB[row][x]$, where

$$x \leftarrow \begin{cases} x + 6, & \text{If row is Even} \\ x - 2, & \text{Otherwise} \end{cases}$$

Else, the position of placement of a Queen for each consequent row (ranging from $\left\lfloor \frac{N}{2} \right\rfloor + 1$ to $N - 1$) is given by the index $CB[row][x]$, where

$$x \leftarrow \begin{cases} x - 2, & \text{If row is Even} \\ \begin{cases} N - 2, & \text{If } (n \geq N) \\ x + 6, & \text{Otherwise} \end{cases}, & \text{Otherwise} \end{cases}$$

Else, For each row ranging from row 0 to N-1, the position of the placement of respective queen is given by the index $CB[row][x]$, where $x \leftarrow 0$ initially, and then incremented by 2 with the condition that $x \leftarrow 1$, if $x > (N - 1)$.

Algorithm 1 shows the complete operation of proposed solution and the flowchart of the same is shown in Figure 2. In this algorithm we have generalized the patterns for the arrangement of the Queens on the given chessboard. The algorithm is designed in a way that it can directly assign the coordinate to the individual Queen for placing on the chess board. Equation (1) shows the total number of steps required by the Algorithm-1, from where it is concluded that the complexity of the proposed algorithm is $O(N)$ (see Equation (2)).

$$f(N) = \begin{cases} \frac{4}{2} + \frac{9}{2}N, & \text{if } (N \% 2 = 0) \\ \begin{cases} 1 + N + \frac{2N}{2} + \frac{2N}{2} + \frac{3N}{4}, & \text{if } (z \% 2 = 0) \\ 1 + N + \frac{2N}{2} + \frac{2N}{2} + \frac{5N}{4}, & \text{otherwise} \end{cases}, & \text{if } (N \% 3 = 0) \\ 1 + 5N, & \text{Otherwise} \end{cases} \quad (1)$$

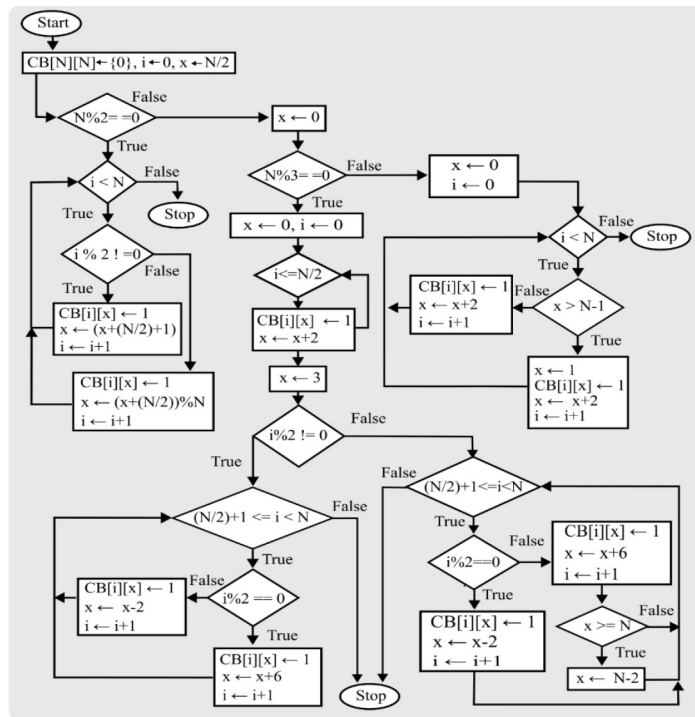


Figure 2

Flow Chart (Proposed Solution to N-Queen Problem)

Algorithm 1 shows the complete operation of proposed solution and the flowchart of the same is shown in Figure 2. In this algorithm we have generalized the patterns for the arrangement of the Queens on the given chessboard. The algorithm is designed in a way that it can directly assign the coordinate to the individual Queen for placing on the chess board. Equation (1) shows the total number of steps required by the Algorithm-1, from where it is concluded that the complexity of the proposed algorithm is $O(N)$ (see Equation (2)).

$$Complexity(f(N)) = \begin{cases} O(N), & \text{if}(N \% 2 == 0) \\ \begin{cases} O(N), & \text{if} \left(\left\lfloor \frac{N}{2} \right\rfloor \% 2 == 0 \right), \\ O(N), & \text{Otherwise} \end{cases} & \text{if}(N \% 3 == 0) \\ O(N), & \text{Otherwise} \end{cases} \quad (2)$$

$$f(N) = \begin{cases} \frac{4}{2} + \frac{9}{2}N, & \text{if}(N \% 2 == 0) \\ \begin{cases} 1 + N + \frac{2N}{2} + \frac{2N}{2} + \frac{3N}{4}, & \text{if} \left(\left\lfloor \frac{N}{2} \right\rfloor \% 2 == 0 \right), \\ 1 + N + \frac{2N}{2} + \frac{2N}{2} + \frac{5N}{4}, & \text{otherwise} \end{cases} & \text{if}(N \% 3 == 0) \\ 1 + 5N, & \text{Otherwise} \end{cases} \quad (3)$$

$$Complexity(f(N)) = \begin{cases} O(N), & \text{if}(N \% 2 == 0) \\ \begin{cases} O(N), & \text{if} \left(\left\lfloor \frac{N}{2} \right\rfloor \% 2 == 0 \right), \\ O(N), & \text{Otherwise} \end{cases} & \text{if}(N \% 3 == 0) \\ O(N), & \text{Otherwise} \end{cases} \quad (4)$$

4. Results and Discussion

The comparative analysis of the proposed solution with the classic algorithm designed to solve N-Queen puzzle is shown in Table 1. It is observed that in comparison to the existing solutions to the N-Queen Problem, the proposed algorithm solves the problem more efficiently with the worst-case time complexity of $O(N)$. Moreover, the rate of growth of functions depicting the time required by various solutions is represented in Figure 3. It clearly shows that the most optimal solution is the one that can be completed within a linear time frame. However, the exceptional case for the proposed algorithm is when the value of N is an odd multiple of 3. The solution of this will be addressed in the future work.

Table 1
Comparative Analysis of available solutions to the N-Queen problem

S no.	Algorithm	Computer Type	Time Complexity
1	Brute force [16]	Classical	$O(N^N)$
2	Backtracking [7]	Classical	$O(N!)$
3	PDA [13]	Classical	$O(N^2)$
4	NQSA [14]	Quantum	$O(N^3)$
6	Proposed	Classical	$O(N)$

Algorithm1: N-Queen Problem	
NQueen(N)	
CB[N][N] ← {0}, i ← 0, x ← N/2	
If N%2 == 0 Then	
For i ← 0 to N, step1	1
If (i%2) != 0 Then	1
CB[i][x] ← 1 ; x ← (x+(N/2) + 1)	N
Else	N
CB[i][x] ← 1 ; x ← (x+(N/2)) % N	N/2
EndIf	N/2
End	N/2
Else If N%3 == 0 Then	
x ← 0, i ← 0	1
For i ← 0 to N/2, step 1	N/2
CB[i][x] ← 1, x ← x+2	N/2
End	
j ← 3	
If (i%2) != 0 Then	1
For i ← ((N/2)+1) to N, Step1	N/2
If i%2 == 0 Then	N/2
CB[i][j] ← 1, j ← j+6	N/2
Else	N/4
CB[i][j] ← 1, j ← j-2	N/4
EndIf	N/4
End	
Else	
For i ← ((N/2)+1) to N, Step1	N/2
If i%2 == 0 Then	N/2
CB[i][x] ← 1, x ← x-2	N/4
Else	N/4
CB[i][x] ← 1, x ← x+6	N/4
If j >= N Then	N/4
j ← N-2	N/4
EndIf	
EndIf	
End	
EndIf	
Else	
i ← 0, j ← 0	1
For i ← 0 to N, Step 1	N
If j > N-1 Then	N
j ← 1	N
CB[i][j] ← 1, j ← j+2	N
Else	N
CB[i][j] ← 1, j ← j+2	N
EndIf	
End	
EndIf	

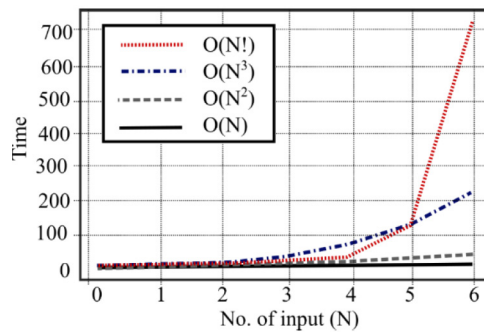


Figure 3

Comparison of the performance of various techniques used to solve N-Queen problem in terms of time complexity.

5. Conclusion and future work

N-Queen puzzle/problem is a classic puzzle in which ‘n’ number of queens and a chessboard of size $N \times N$ is given. The problem is to lay all the given queens on the chessboard in such a way that all queens are safe. In this article we have proposed a linear time solution to the N-Queen problem (i.e., $O(N)$). Although, multiple variants of the queen arrangement on the chess board can be obtained by applying various transformations (such as rotating, mirroring, etc.) on the output of the proposed solution in constant time, but still all possible solutions to N-Queen Problem in linear time is a matter for the future research.

References

- [1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press (2022).
- [2] Jain, M. K. Bohra, M. K. Sharma, and V. G. Shankar, “A modi-fied cuckoo search for the N-Queens problem,” in *Data Engineering for Smart Systems: Proceedings of SSIC 2021*, pp. 615–620 (2022).
- [3] R. M. Haralick and G. L. Elliott, “Increasing tree search efficiency for constraint satisfaction problems,” *Artif. Intell.*, vol. 14, no. 3, pp. 263–313 (Oct. 1980), doi: 10.1016/0004-3702(80)90051-X.
- [4] J. Bell and B. Stevens, “A survey of known results and research areas for N-Queens,” *Discrete Math.*, vol. 309, no. 1, pp. 1–31 (Jan. 2009), doi: 10.1016/J.DISC.2007.12.043.

- [5] L. E. Mou-hua, L. I. Wei-Xuan, and E. T. Wang, "A generalization of the N-Queen problem," *J. Syst. Sci. Math. Sci.*, vol. 9, no. 2, p. 158 (1989).
- [6] O. Demiroers, N. Rafraf, and M. M. Tanik, "Obtaining N-Queens solutions from magic squares and constructing magic squares from N-Queens solutions," *J. Recreat. Math.*, vol. 24, no. 4, pp. 272–280 (1992).
- [7] H. Izadkhah, "Backtracking Algorithms," in *Problems on Algorithms: A Comprehensive Exercise Book for Students in Software Engineering*, Springer, pp. 487–496 (2022).
- [8] E. Horowitz, *Fundamentals of computer algorithms*. Galgotia Publications (1978).
- [9] R. Sosic and J. Gu, "A polynomial time algorithm for the N-Queens problem," *ACM SIGART Bull.*, vol. 1, no. 3, pp. 7–11 (1990).
- [10] B. Abramson and M. Yung, "Divide and conquer under global constraints: A solution to the N-Queens problem," *J. Parallel Distrib. Comput.*, vol. 6, no. 3, pp. 649–662 (Jun. 1989), doi: 10.1016/0743-7315(89)90011-7.
- [11] S. Khan, M. Bilal, M. Sharif, M. Sajid, and R. Baig, "Solution of n-Queen problem using ACO," INMIC 2009 - 2009 IEEE 13th Int. Multitopic Conf. (2009), doi: 10.1109/INMIC.2009.5383157.
- [12] K. Agarwal, A. Sinha, and M. Hima Bindu, "A novel hybrid approach to N-Queen problem," *Adv. Intell. Soft Comput.*, vol. 166 AISC, no. VOL. 1, pp. 519–527 (2012), doi: 10.1007/978-3-642-30157-5_52/COVER.
- [13] Z. Wang, D. Huang, J. Tan, T. Liu, K. Zhao, and L. Li, "A parallel algorithm for solving the N-Queens problem based on inspired computational model," *Biosystems*, vol. 131, pp. 22–29 (May 2015), doi: 10.1016/J.BIOSYSTEMS.2015.03.004.
- [14] R. Jha, D. Das, A. Dash, S. Jayaraman, B. K. Behera, and P. K. Pani-grahi, "A novel quantum N-Queens solver algorithm and its simulation and application to satellite communication using IBM quantum experience," arXiv Prepr. arXiv1806.10221 (2018).
- [15] H. Ahrabian, A. Mirzaei, and A. Nowzari-Dalini, "A DNA Sticker Algorithm for Solving N-Queen Problem.," *Int. J. Comput. Sci. Appl.*, vol. 5, no. 2, pp. 12–22 (2008).

- [16] S. Mukherjee, S. Datta, P. B. Chanda, and P. Pathak, "Comparative study of different algorithms to solve N queens problem," *Int. J. Found. Comput. Sci. Technol*, vol. 5, no. 2, pp. 15-27 (2015).
- [17] Su, T. S., Wu, C. C., & Chang, L. W. : Integration of audio surveillance on a queen bee rearing and breeding management system. *Journal of Information and Optimization Sciences*, 42(4), 719-734 (2021).
- [18] Tang, H. C., Hsieh, K. H., & Chao, T. L. : A backward heuristic algorithm for two-term multiple recursive random number generators. *Journal of Discrete Mathematical Sciences and Cryptography*, 13(6), 593-600 (2010).
- [19] Reddy, V. D., & Rao, G. S. V. R. K. : Meta-heuristic approaches to solve shortest lattice vector problem. *Journal of Discrete Mathematical Sciences and Cryptography*, 24(1), 81-91 (2021).
- [20] Chen, W. J., Lin, S. M., & Tsou, J. C. : Sequencing heuristic for bicriteria scheduling in a single machine problem. *Journal of Information and Optimization Sciences*, 27(2), 459-468 (2006).
- [21] Vimala Rani, M., & Mathirajan, M. : Performance evaluation of ATC based greedy heuristic algorithms in scheduling diffusion furnace in wafer fabrication. *Journal of Information and Optimization Sciences*, 37(5), 717-762 (2016).
- [22] Chiang, C. P. : An enhanced formulation and simple heuristic for scheduling jobs on unrelated parallel machines. *Journal of Information and Optimization Sciences*, 26(1), 193-204 (2005).
- [23] Kumar, A., Singh, K., & Khan, T. : L-RTAM: Logarithm based reliable trust assessment model for WBSNs. *Journal of Discrete Mathematical Sciences and Cryptography*, 24(6), 1701-1716 (2021).
- [24] Khan, T., & Singh, K. : Resource management based secure trust model for WSN. *Journal of Discrete Mathematical Sciences and Cryptography*, 22(8), 1453-1462 (2019).
- [25] Shariq, M., Singh, K., Lal, C., Conti, M., & Khan, T.: ESRAS: An efficient and secure ultra-lightweight RFID authentication scheme for low-cost tags. *Computer Networks*, 217, 109360 (2022).
- [25] Kumar, A., Singh, K., Khan, T., Ahmadian, A., Saad, M. H. M., & Manjul, M. : ETAS: An efficient trust assessment scheme for BANs. *IEEE Access*, 9, 83214-83233 (2021).