N-Queens Problem

Team Members

1- Name: محمد أحمد إبراهيم الدسوقي

ID: 201900616

Level: 3 (IS)

2- Name:محمد إبراهيم عبد الفتاح محمد

ID: 201900613

Level: 3 (IS)

3- Name: محمد إبراهيم عجمي محمد

ID: 201900614

Level: 3 (IS)

4- Name: محمد ياسر محمد مرسي

ID: 201900752

Level: 3 (IS)

5- Name: مي رفعت راشد سيد

ID:201900867

Level: 3 (IS)

6- Name: صباح شريف صلاح أحمد

ID:201900379

Level: 3 (IS)

Project idea

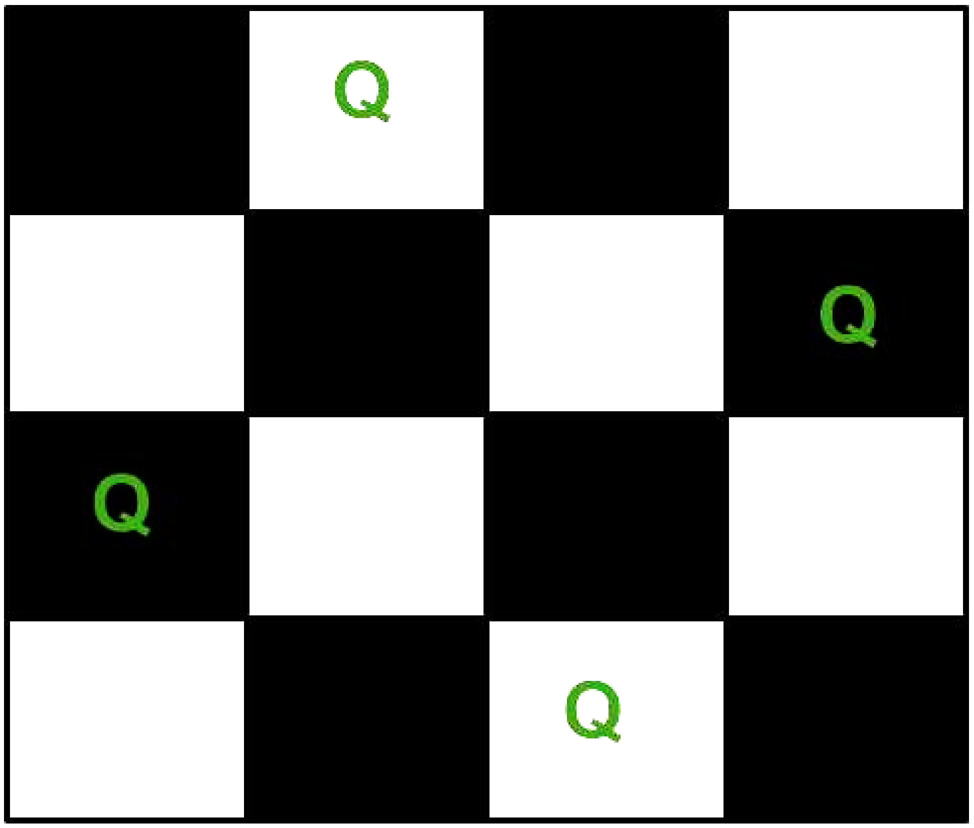
1. we have NxN chessboard and want to placing max number of queens in chessboard so that no two queen attacks each other (i.e., no two queens occupy the same row, column, or diagonal)

2. input to the function is matrix with zeros represent empty place

3. solving by Backtracking and Deferential Evaluation Algorithms

4. output is matrix with zeros and ones, zeros represent empty places and one represent we put queen here

Example 4x4 chessboard:



<u>Main functionalities</u>

<u>Backtraking</u>

- Function called isSafe() which checks if the place of the queen is safe or not bytaking the board and the position where I want to place the queen.
- Function called solveNQueens() which finds the solution of the N*N chessboard byusing Backtracking Algorithm.

<u>Differential Evolution</u>

- Function called maxFitness() which find best Fitness (GOAL)
- Function called population() which Initialize Population 'two dimensional array of range 100
- Function called generateChromosome() which randomly generates a sequence of Chromosomes
- Function called mutation() which change the value of a random index in chromosome
- Function called crossover() which reproduce new chromosome from two Chromosomes
- Function called probability() which calculate chromosome fitness probability
- Function called randomChromosome() which pick random chrosome, based on probability
- Function called DEQueen() which Differential Evolution Algorithm
- Function called restart_program() which restart program

<u>The difference between backtracking and differential evaluation</u>

Differential Evolution is considered the best and most efficient solution than backtracking and faster than it because the backtracking is considered to be looking for a specific solution and walking on all of it, but the differential evaluation is looking for all solutions and leads to the best solution and shows it as the best goal

Similar Application

NQueen

https://play.google.com/store/apps/details?id=com.ShinyWorld.NQueen

An initial literature review of Academic publications to N-Queens Problem

Abstract— This paper presents a study on the N-Queens Problem. Different approaches to its solution discussed in the scientific literature are analyzed. The implementation of an algorithm based on the backtracking method is also presented. The algorithm is optimized to find solutions in a specific subset of configurations among all possible ones. With this approach, the computational complexity of the algorithm is reduced from exponential to quadratic. In this way, the algorithm finds all possible solutions in a shorter time: fundamental and their symmetrical equivalents. The methodology for conducting the experiments is presented. The purpose of the study, the tasks to be performed, and the conditions for conducting the experiments are presented as well. In connection with the research, an application that implements the presented algorithm has been developed. This application generated all the results obtained in this study. The experimental results show that with a linear increase in the number of queens (equivalent to a quadratic increase in the number of fields on the board, the number of recursive calls made by the algorithm increases exponentially. Similarly, the number of possible solutions, as well as the execution time of the algorithm (in the different modes of the application - internal, interactive, and combined), also increases exponentially. However, the algorithm's execution time in the internal mode is significantly shorter than in the other two modes - interactive and combined. The future guidelines for the study are presented.

Introduction

The N-Queens problem (NQP) is a classical, combinatorial optimization problem that has been actively studied in recent years.

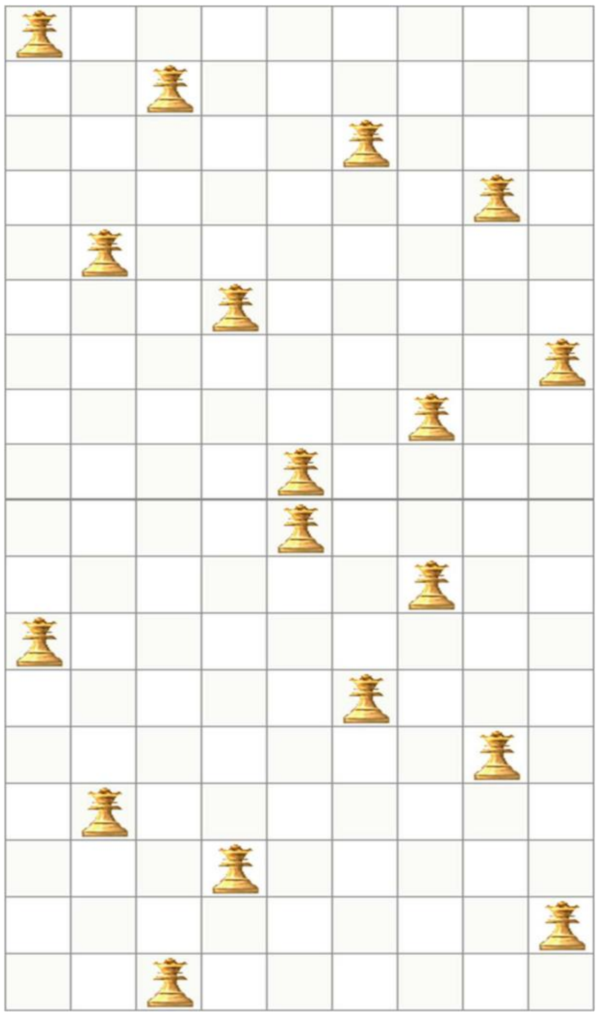
In other words, the goal is to place the pieces on the board so that they do not endanger each other (which in turn means that there can be at most one queen on each horizontal, vertical, and diagonal).

Many of these methods are discussed in the scientific literature, comparing different approaches, and publishing improved versions of existing algorithms.

Other studies show that the N-Queens problem can be successfully solved by other techniques, such as simulated annealing, methods based on local search, heuristic and meta-heuristic approaches

1 shows two (asymmetric) solutions of the N-Queens the problem is at a board size of 9 x 9.

When using this approach, it is very important to apply a method that reduces the number of analyzed individual cases of the problem.



<u>Material and Method</u>

In this section, an implementation of a recursive algorithm for solving the NQP will be presented.

If at the last (successful) placement of a queen on the board it turns out that the placed queens are exactly N (i.e., QeueenCount = N), then this means that an acceptable solution has been found. Global data structures are accessible from all functions (methods) of the application.The values used are as follows: 1 - the field is "attacked", 0 - the field is not "attacked".

The Solution array (also of type Byte) stores a partial or complete solution of the problem, as the values used are as follows 1 - there is a queen placed in this field and 0 otherwise. The DisableFields method receives the variables ACol and ARow as input parameters.

In the declarative part of the DisableFields method, the local variables Col, Row, DCol and DRow (of Integer type) are declared.

The next code that executes the DisableFields method is to disable (i.e. mark as "attacked") all non-disabled fields from column ACol, all non-disabled fields from row ARow, all non-disabled fields from one of the left diagonals, which contains the field ACol, ARow and all non-disabled fields from one of the right diagonals, which also contains the field ACol, ARow.

An inefficient approach to solving NQP is to test all possible combinations of queen’s placements on the board. The implementation of the algorithm follows. This method checks if it is possible to position the next queen on any of the free fields on the board.

The number of recursive calls to FindSolutions is stored in the global variable RecursionCount.

However, if no free position can be found for the current queen, the algorithm takes a step back (backtracking). This process is repeated until all possible solutions are generated.

The time is obtained from the GetTickCount function. The difference between the values of the variables Finish and Start is actually the execution time of the FindSolution method (in milliseconds).

Finally, the InitializeAndStart method concatenates in the string variable Msg the number of generated solutions (the SolutionCount variable), the number of recursive calls (the RecursionCount variable), and the execution time of the whole process.

Result and Discussion

A. Development of an Application for Conducting Experiments There are many programming languages and application development environments.

In addition, these applications can be run on different target platforms (operating systems and servers). During the application design stage, the developer (designer) creates the layout of the application.

For the purposes of the study, an application was developed to perform the planned experiments. This ratio shows that the number of fundamental (asymmetric) solutions represents 12.5% of the number of all possible solutions.

The values in the "Internal", "Interactive" and "Combined" columns are arithmetic mean of four different application runs (for each of the modes).

Conclusion

In this paper, a study of the N-Queens Problem was presented.

In this way, the algorithm finds in a shorter time all possible solutions, both fundamental and their symmetrical equivalents.

The methodology for conducting the experiments was presented.

As part of the methodology, 14 board sizes were presented, from 8 x 8 (standard chessboard) to 21 x 21, respectively.

However, the execution time of the algorithm in the internal mode was significantly shorter than in the other two modes - interactive and combined.

The ratio between the number of recursive calls and the number of all solutions was also calculated.

Algorithms used

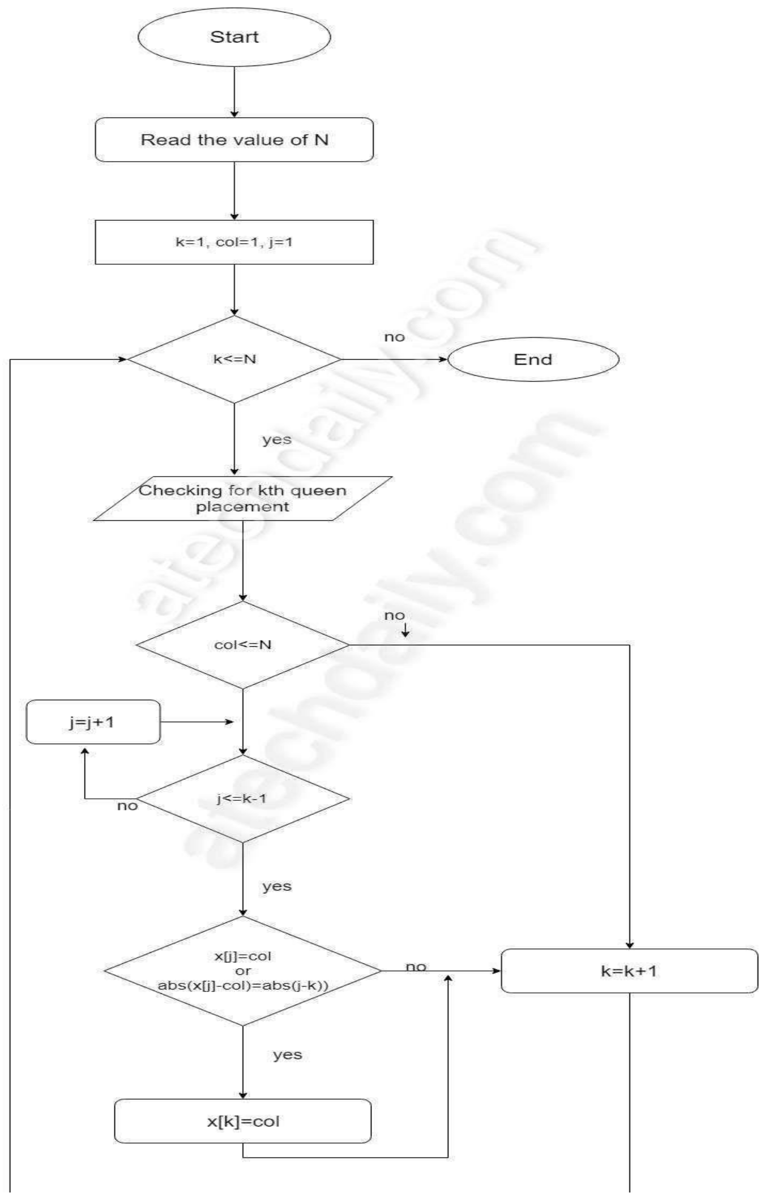
- Backtracking Algorithm

Similar applications in the market

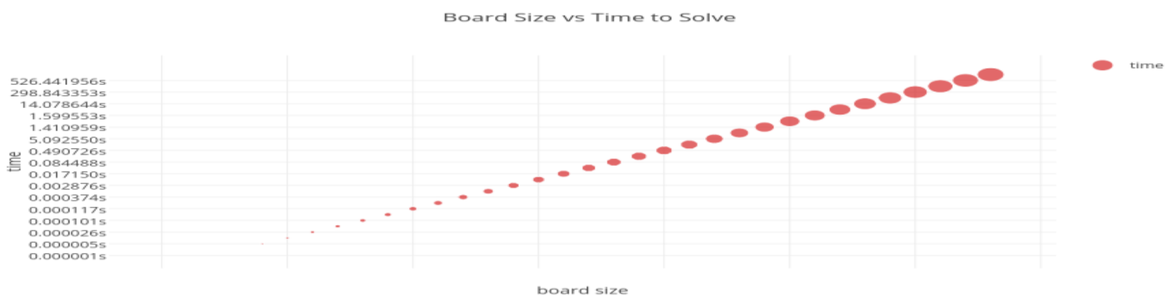
- Backtracking Algorithm
  - To find all Hamiltonian Paths present in a graph.
  - Maze solving problem.
  - The Knight's tour problem.
  - Puzzles such as eight queens’ puzzle
- Differential Evolution Algorithm
  - Determine a good correlation between model predictions and experimental data.
  - Solving timetabling problems.

Diagrams

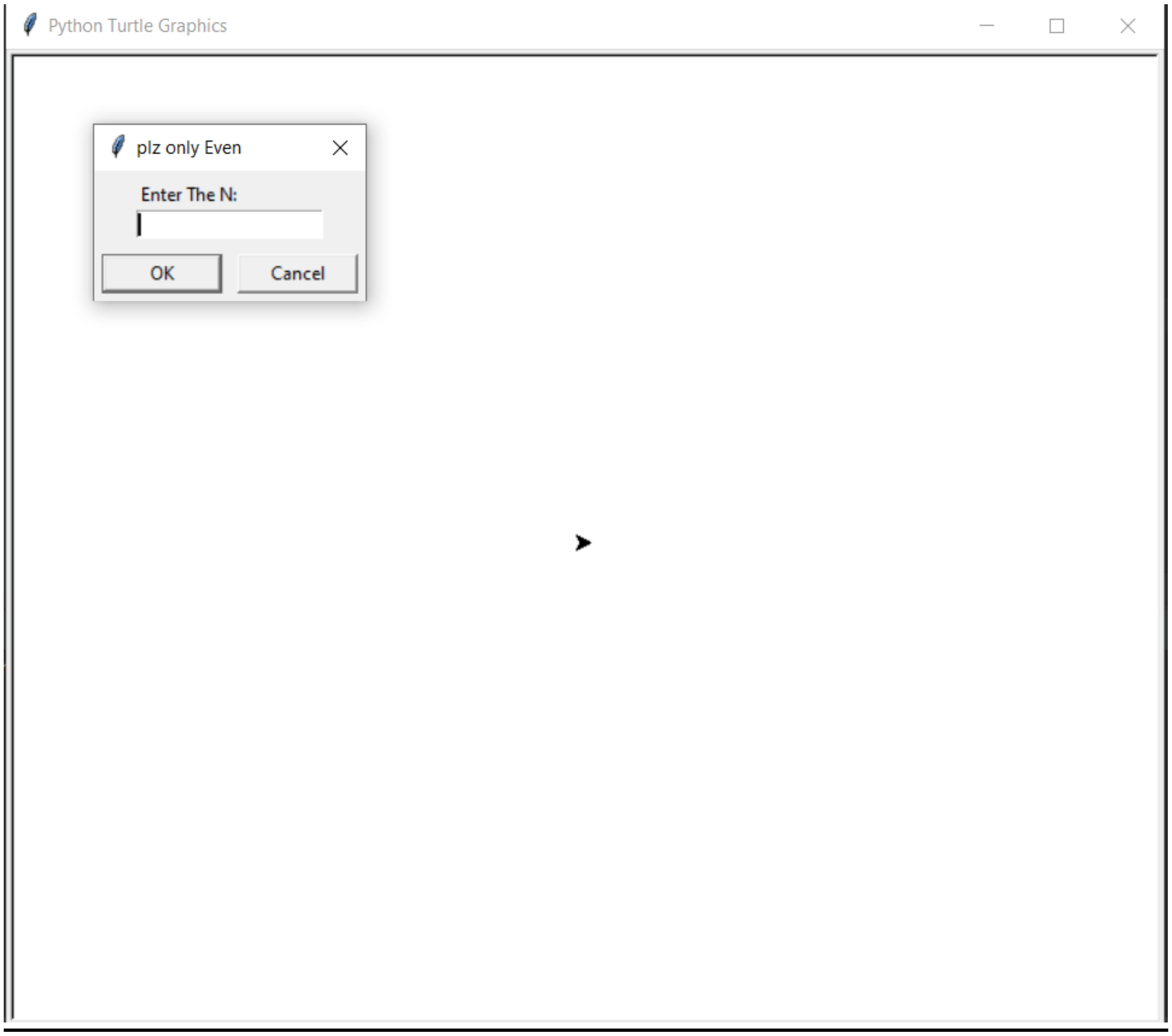
Flowchart


Start
Read the value of N
k=1, col=1, j=1
k<=N
no
End
yes
Checking for kth queen placement
col<=N
no
j=j+1
j<=k-1
no
yes
x[j]=col or abs(x[j]-col)=abs(j-k))
no
k=k+1
yes
x[k]=col

Plots

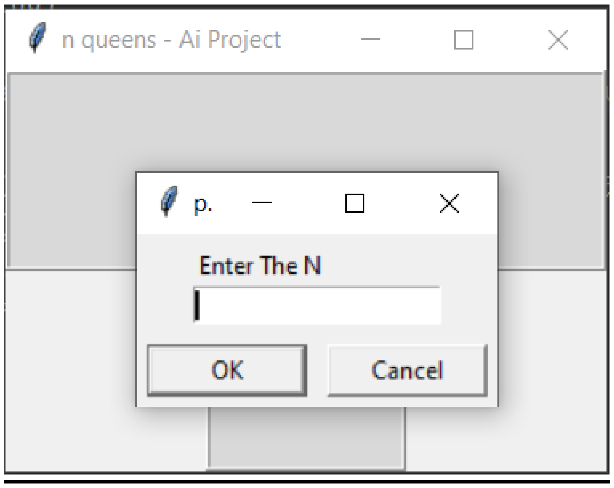

Board Size vs Time to Solve
time
526.441956s
298.843353s
14.078644s
1.599553s
1.410959s
5.092550s
0.490726s
0.084488s
0.017150s
0.002876s
0.000374s
0.000117s
0.000101s
0.000026s
0.000005s
0.000001s
time
board size

Inputs


Python Turtle Graphics

plz only Even

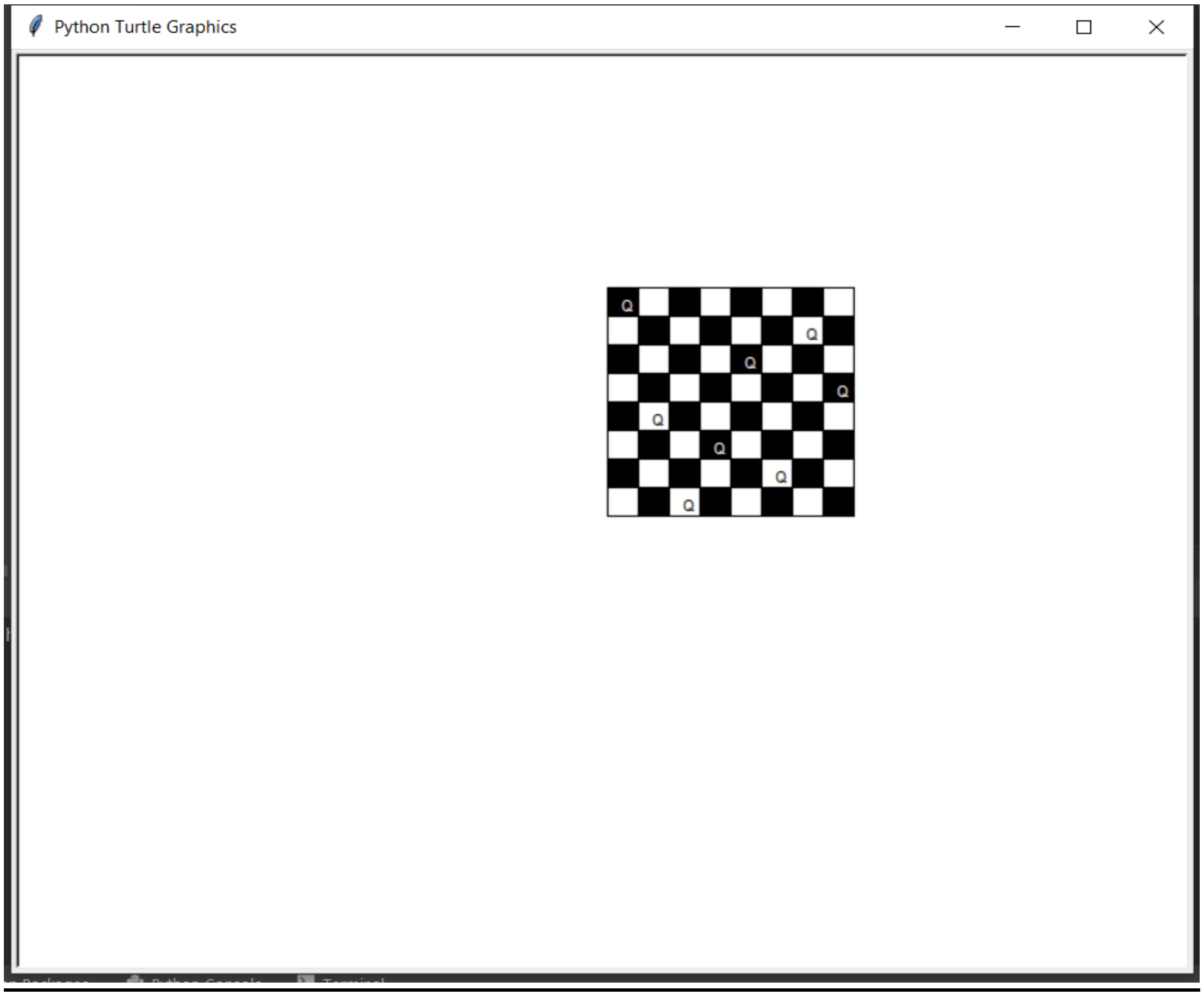
Enter The N:

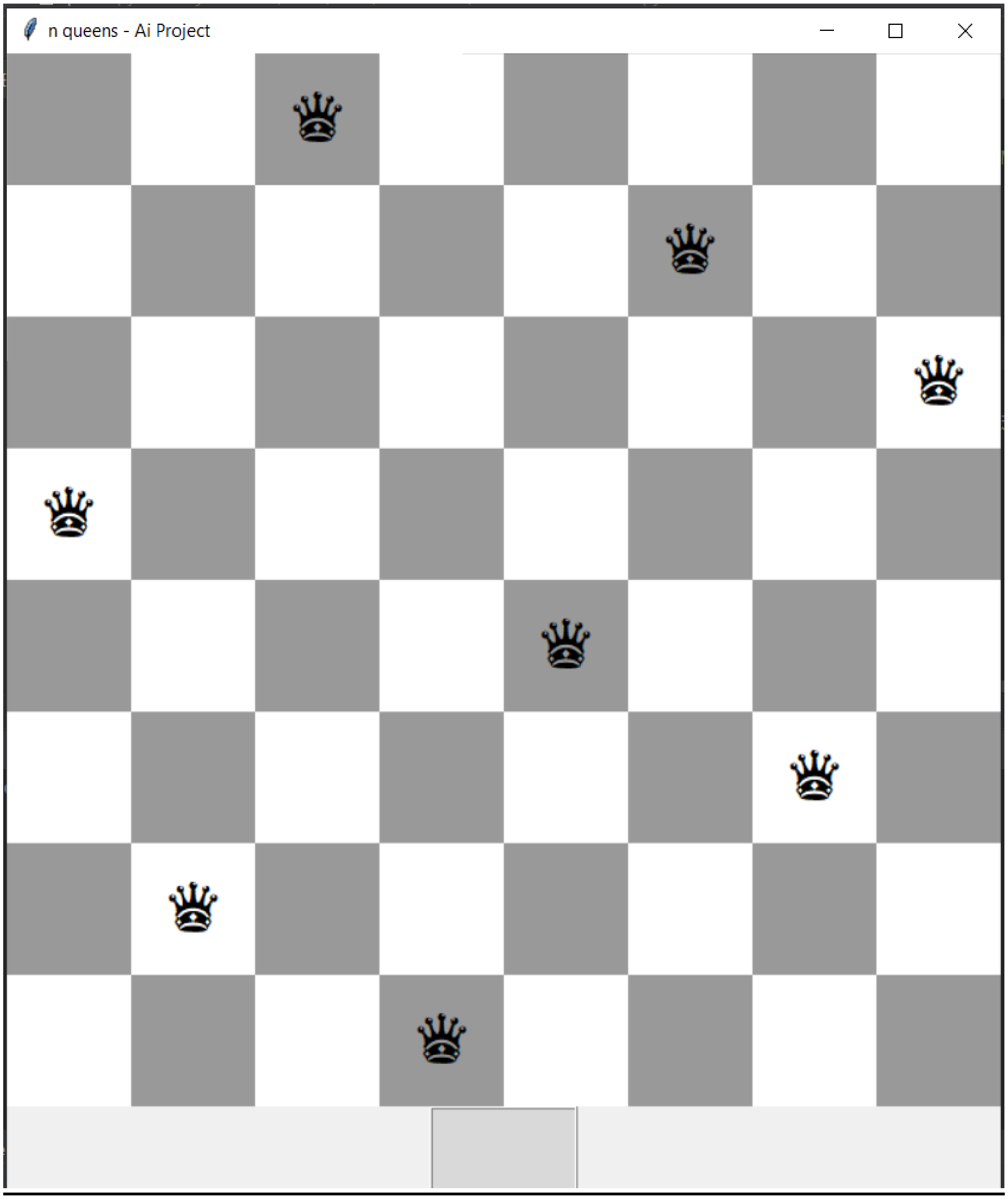

n queens - Ai Project

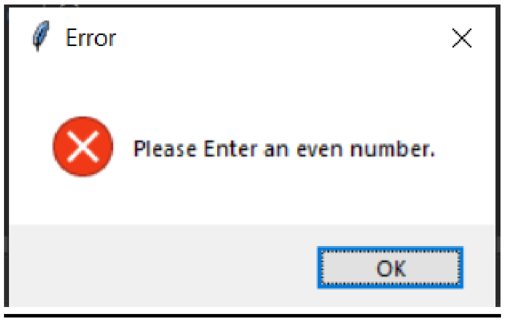
p.

Enter The N

Outputs


Python Turtle Graphics
Q
Q
Q
Q
Q
Q
Q
Q


n queens - Ai Project


Error

Please Enter an even number.

Libraries

Backtraking

- turtle

- tkinter

Differentiational evaluation

- tkinter

- random

Source Code URL

https://github.com/mahmedaldosoky/n-queens

References

- Artificial Intelligence - A Modern Approach (3rd Edition 2010)
- Backtracking Algorithms – geeksforgeeks
- Backtracking Algorithms – Programiz
- Differential evolution - Wikipedia
- https://www.academia.edu/Documents/in/N-Queens_Problem