|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **ID** | **Department** | **Level** |
| محمد مصطفي محمود عبدالمجيد | 20210837 | CS | 3 |
| يوسف محمد نادي حسن | 20211104 | CS | 3 |
| محمد عبدالرزاق فؤاد | 20210807 | CS | 3 |
| محمد أيمن محمد قنديل | 20210751 | CS | 3 |
| احمد رضا شعبان عبد الوهاب | 20210050 | CS | 3 |
| محمد ناصر رفعت على | 20210841 | CS | 3 |
| محمد عوض الله سري احمد | 20210822 | CS | 3 |

Code & Documentation link:

<https://github.com/muhammed-elsherif/N-Queens-multiple-alg>

Introduction and Overview:

-Project idea and overview:

**Abstract** 🡪 The N-Queens Problem Solver project aims to develop an intelligent system capable of solving the N-Queens problem for various board sizes.

The N-Queens problem is a classic chessboard puzzle where the objective is to place N queens on an N×N chessboard in such a way that no two queens threaten each other. Threatening means no two queens share the same row, column, or diagonal.

The value of N, representing the board size and the number of queens, can be chosen by the user.

User Interaction:

The user has the flexibility to choose the size of the chessboard (N), making the project adaptable to different scenarios.

The system will present solutions generated by each algorithm, allowing the user to compare their effectiveness and efficiency.

Example:

User Input:

Board size (N): 8  
OR

Number of queens: 8

Output:

Solution found: Yes

The chessboard is an 8x8 grid.

The queens are represented by 'Q' in the following positions:

. Q . . . . . .

. . . . Q . . .

. . . . . . . Q

. . . . . Q . .

Q . . . . . . .

. . . Q . . . .

. . . . . . Q .

. . Q . . . . .

- Similar applications in the market:

Parallel memory storage schemes.

[VLSI](https://www.sciencedirect.com/topics/mathematics/very-large-scale-integration) testing.

Traffic control.

Resource Allocation and Scheduling.

Deadlock prevention.

-Functionalities/Features:

-Random Board Generation: The system can generate random initial configurations of queens on the chessboard.

-User Input: Allow users to specify the board size and the number of queens they want to place.

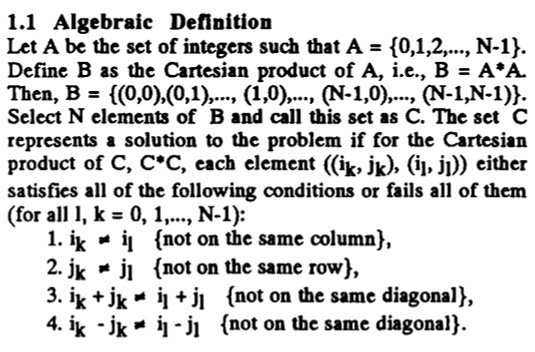
-Solving Algorithm: Implement an intelligent solving algorithm to find a valid solution for the N-Queens problem.

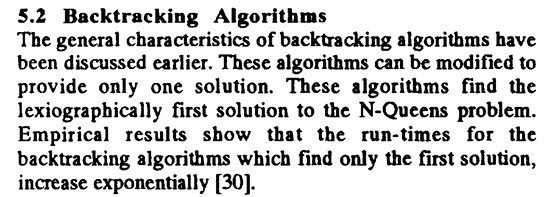
-Solution Visualization: Display the final solution on the chessboard, highlighting the positions of the queens.

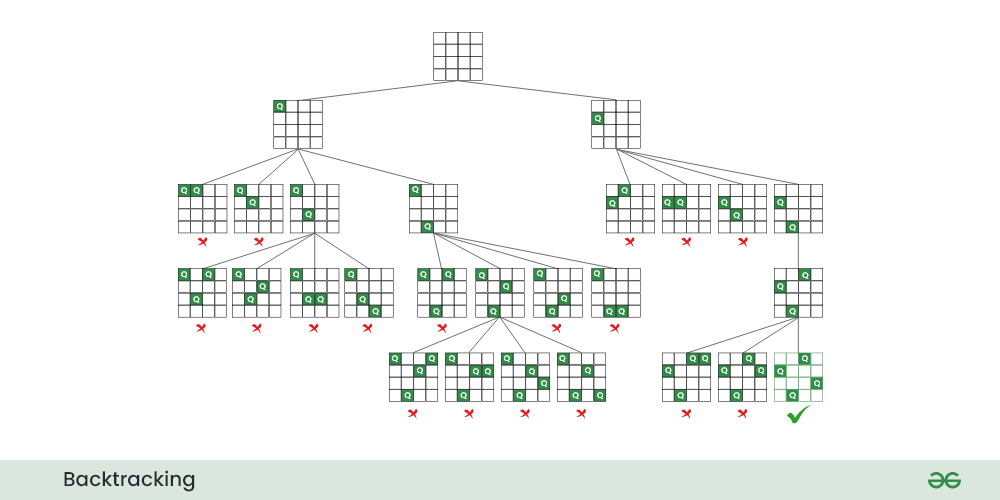
-Multiple Solutions: Provide the option to find and display multiple valid solutions if they exist.

-Performance Metrics: Calculate and display metrics such as the time taken to find a solution and the number of backtracks performed.

-A Literature Review of Academic publications:



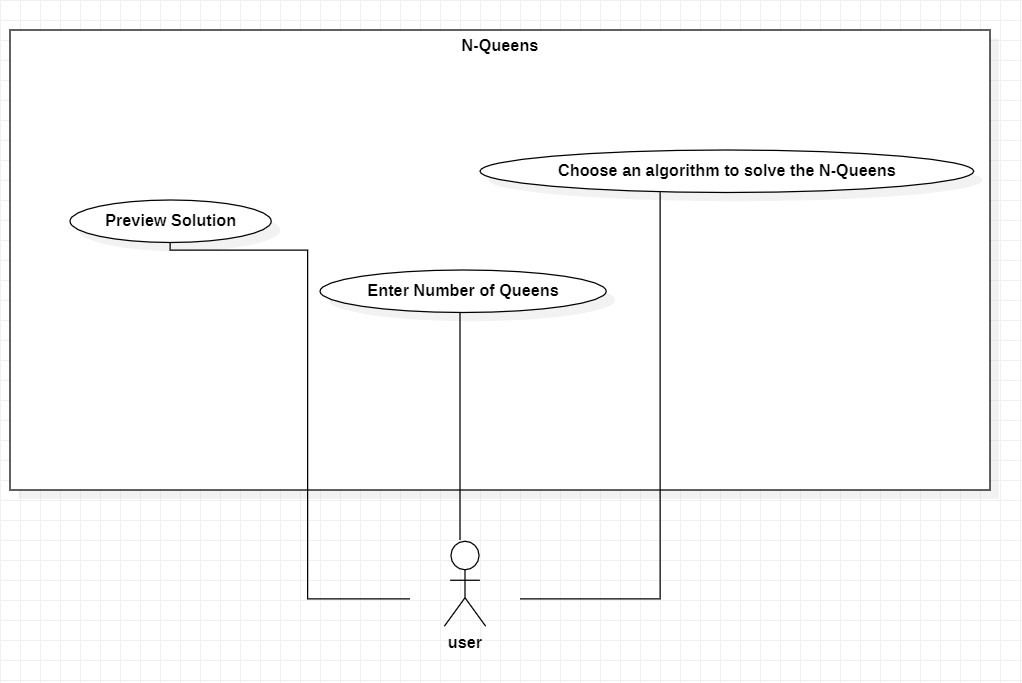




Proposed Solution:

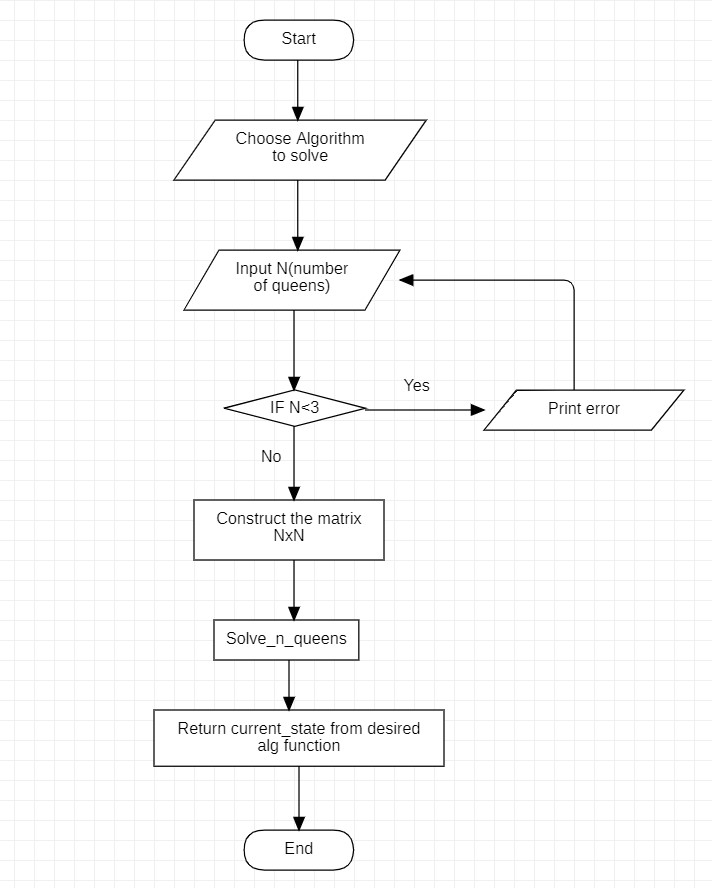
-Functionalities/Features (users’ perspective):

use-case:



Applied Algorithms:

flowchart:



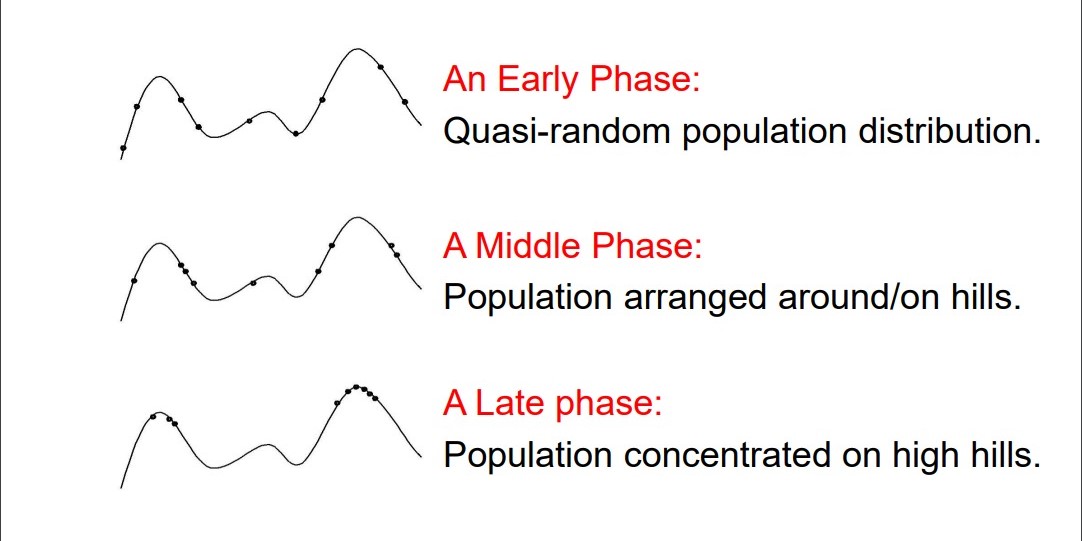
-Backtracking:

Backtracking Search Algorithm: The Backtracking Search Algorithm is a systematic method for exploring potential solutions to a problem. In the context of the N-Queens problem, it involves placing queens on the board one by one and backtracking if a conflict is detected. This algorithm guarantees finding all possible solutions.

-Genetic:

The Genetic Algorithm mimics the process of natural selection to evolve solutions. In the N-Queens context, a population of potential solutions undergoes genetic operations like crossover and mutation.

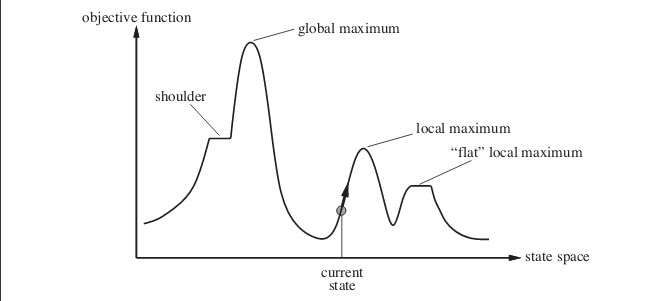
This approach explores a diverse solution space, often finding effective solutions.



-Hill-Climbing:

Hill-Climbing is a local search algorithm that continually moves towards higher elevations in the solution space. Applied to the N-Queens problem, it involves iteratively adjusting queen

placements to ascend towards a configuration with fewer conflicts. It may get stuck in local optima.



Local maximum: It is a state which is better than its neighboring state however there exists a state which is better than it(global maximum). This state is better because here value of objective function is higher than its neighbors.

Global maximum: It is the best possible state in the state space diagram. This because at this state, objective function has highest value.

Plateau/flat local maximum: It is a flat region of state space where neighboring states have the same value.

Ridge: It is region which is higher than its neighbors but itself has a slope. It is a special kind of local maximum.

Current state: The region of state space diagram where we are currently present during the search.

Shoulder: It is a plateau that has an uphill edge.

Steepest-Ascent Hill-Climbing: It is a variant of Hill Climbing algorithm. In this algorithm, we consider all possible states from the current state and then pick the best one as successor.

Allowing sideways moves: When stuck on a ridge or plateau (i.e., all successors have the same value), allow it to move anyway hoping it is a shoulder and after some time, there will be a way up.

Random-restart hill-climbing: If the first hill-climbing attempt doesn’t work, try again and again and again. That is, generate random initial states and perform hill-climbing again and again. This is random-restart. The number of attempts needs to be limited, this number depends on the problem.

-Best-First:

Best-First Search is an algorithm that intelligently selects the most promising path based on a heuristic evaluation. In the N-Queens context, this algorithm evaluates board configurations using a heuristic to guide the placement of queens, prioritizing paths that seem most likely to lead to a solution.

Experiments & Results:

Development platform tools:

VS Code, PyCharm

Programming Language: Python

Python Libraries: random, tkinter, heapq