AI mini project Report
4 Queens

Introduction :

4 - Queens problem is to place 4 - queens in such a manner on an 4 x 4 chessboard so that no queens attack each other by being in the same row, column or diagonal. To solve this we have used the backtracking algorithm.

Constraint satisfaction problem :

A Constraint Satisfaction Problem is characterized by:

- a *set of variables* {x1, x2, .., xn},
- for each variable xi a *domain* Di with the possible values for that variable, and
- a set of *constraints*, i.e. relations, that are assumed to hold between the values of the variables.

For 4 queens problem,following will be criteria:

1. Variables - Here we are considering 4 x 4 matrix,so it will act as a Variable.
2. Domain - Queen Q1,Q2,Q3,Q4 will act as the domain.
3. Constraints - For checking the safe state, we used following constraints:

   For all the positions(columns) in the current row.

- Check all the previous rows. Is there a queen or not?
- Check for all previous diagonal columns. Is there a queen or not?
- If any of these conditions are true,then backtrack to the previous row and move the previous queen 1 step forward.
- Otherwise,put the current queen in the position and move to the next row.

Approach:

Here we have to place 4 queens, say **Q1, Q2, Q3, Q4** on the 4 x 4 chessboard such that no 2 queens attack each other.

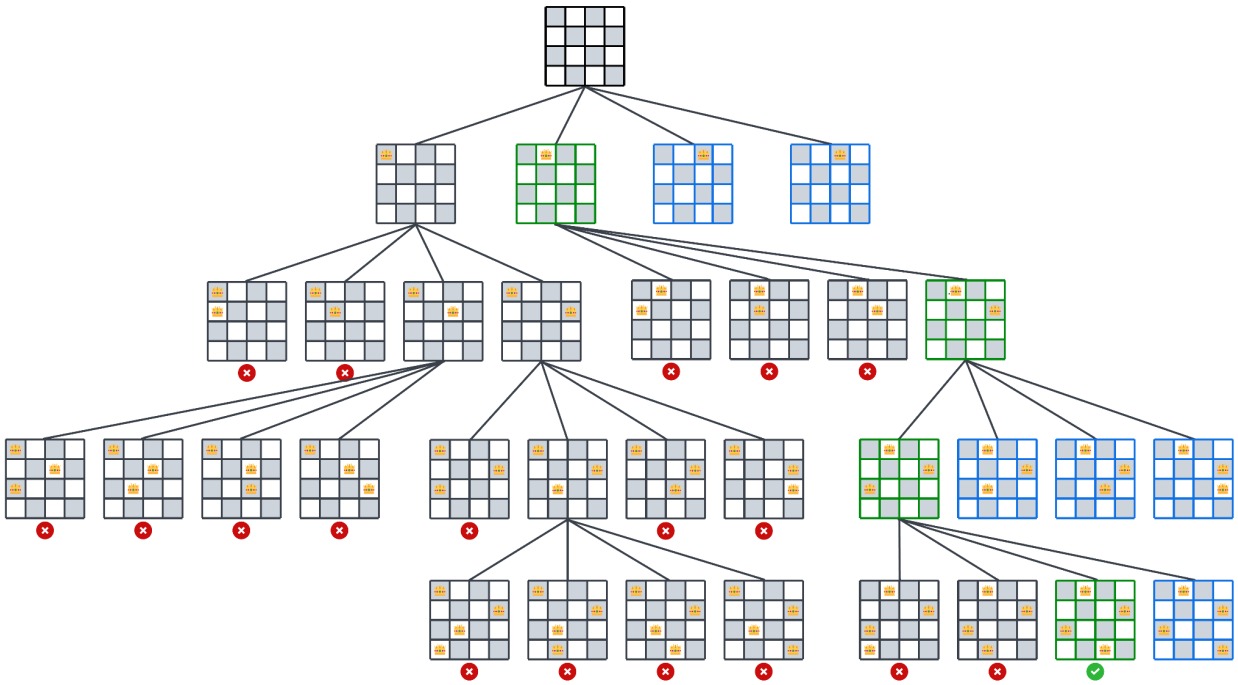
- Let's suppose we're putting our first queen Q1 at position (1, 1) now for Q2 we can't put it in 1 row(because they will conflict).
- So for Q2 we will have to consider row 2. In row 2 we can place it in column 3 I.e at (2, 3) but then there will be no option for placing Q3 in row 3.
- So we **backtrack** one step and place Q2 at (2, 4) then we find the position for placing Q3 is (3, 2) but by this, no option will be left for placing Q4.
- Then we have to backtrack till 'Q1' and put it to (1, 2) instead of (1, 1) and then all other queens can be placed safely by moving Q2 to the position (2, 4), Q3 to (3, 1), and Q4 to (4, 3).
- Hence we got our solution as **(2, 4, 1, 3)**, this is the one possible solution for the 4-Queen Problem.

Backtracking:

Backtracking uses recursive calling to find the solution by building it step by step. It helps to reach a solution by backtracking the current move if it doesn't satisfy the constraint given in the problem.

- Backtracking is an algorithm used to solve the problem of 4 queens.
- Here we place the queens one by one row wise.
- When we place the Queen in a row we check the clashes with other queens.
- First we check the clashes in a column then we check the right and left diagonals.
- If no clash is found then we place the Queen, Otherwise we backtrack the current move.

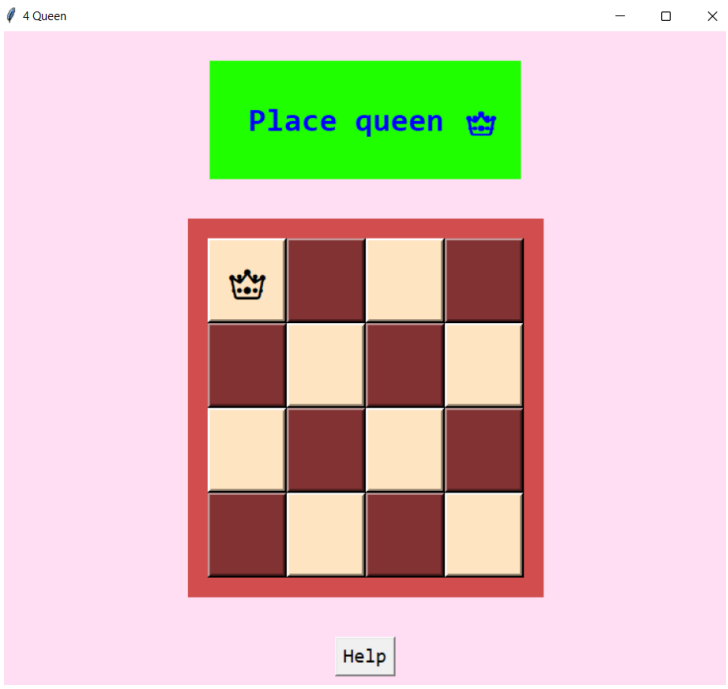
Its Time Complexity is O(N!).



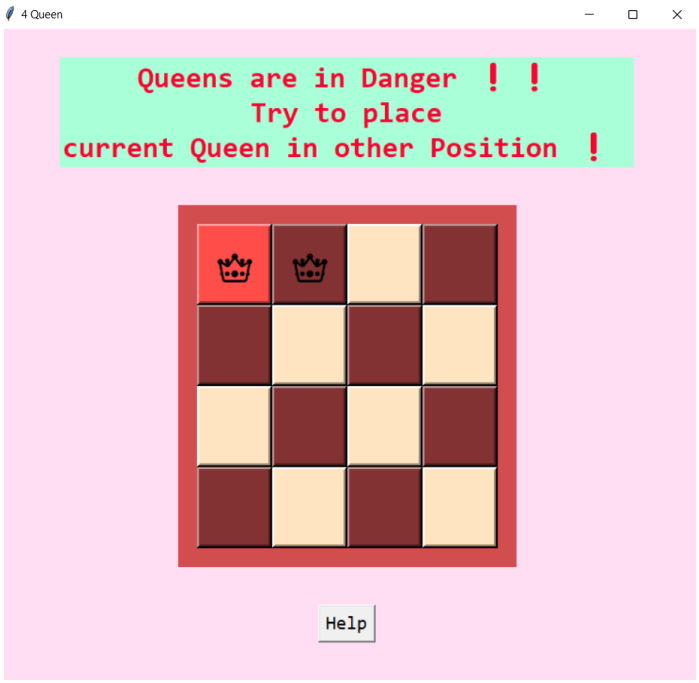
Environment :

- Observability -
  Environment is fully observable. Chess board window is an environment.
- Determinism -
  Deterministic - next state of environmental completely described by the current state and agent action.
- Episodicity -
  Sequential - The Agent may need to plan ahead or decide its action based on what happened in the previous episode. Next queen position depends on previous queens only.
- Dynamism -
  Static - Environment does not change by itself when agent is thinking.
- Continuity -
  Discrete - The number of actions are limited.

Images :

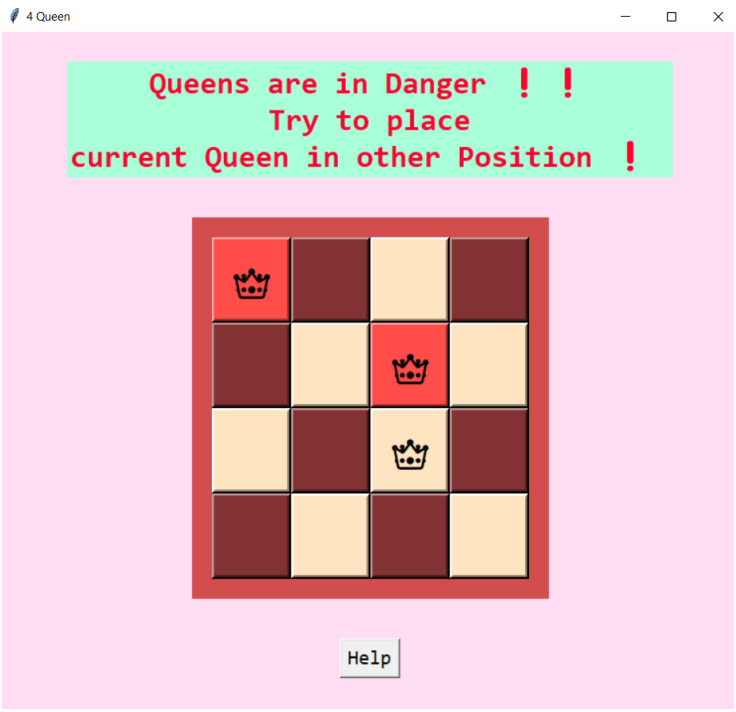


Here we have placed the first queen in the first row.



Now, the second queen is placed in the same row therefore it is asking to place it in another place.

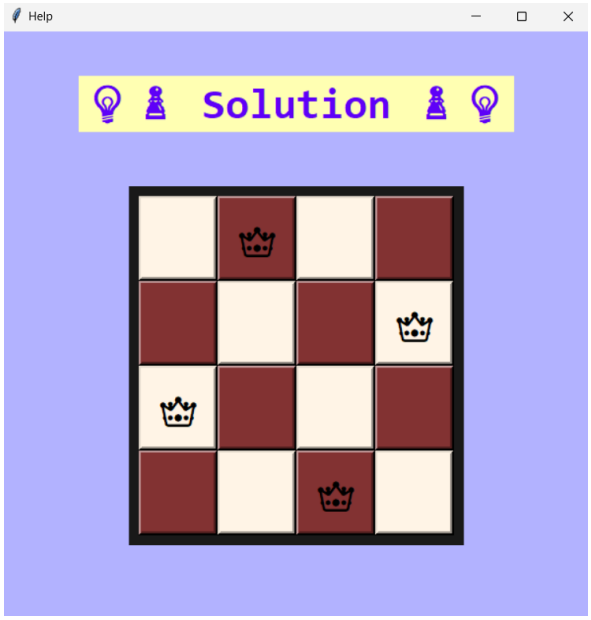
Here if have diagonally placed therefore its asking to place in another position.



Here we placed the third Queen in the same column of the second queen therefore it's asking to back track.

So in this way if constraints are not satisfied we are asked to backtrack the current move and try another one.

Here we are able to put all the queens in their appropriate positions hence it gives us a congratulatory message.



If we are not able to place a queen in the correct position in every trial,then by clicking on the “Help” button, the algorithm will provide a final solution.

Libraries and functions used :

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI for N queen using Tkinter -

- In this game, we have created two windows. Main window for game playing and second window for solution which appears on clicking button help.
- Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets and are used for creating chess boards and displaying messages.
- These widgets have standard attributes such as size, color, fonts and they are used in game, to make user interface interactive and styling.
- Tkinter has the following geometry manager classes: pack, grid, and place.
- The pack() Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The grid() Method − This geometry manager organizes widgets in a table-like structure in the parent widget. This is used to make the grid structure of a chess board.

Algorithm :

Backtracking Algorithm -

The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the

solution. If we do not find such a row due to clashes, then we backtrack and return false.

1) Start in the topmost row

2) If all queens are placed

return true

3) Try all columns in the current r.

Do the following for every tried row.

a) If the queen can be placed safely in this row then mark this [row,column] as part of the solution and recursively check if placing the queen here leads to a solution.

b) If placing the queen in [row, column] leads to a solution then return true.

c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.

4) If all rows have been tried and nothing worked,

return false to trigger backtracking.

Project done by -

3611 - Sakshi Bharambe

3612 - Sakshi Bhosale

3617 - Bhakti Chavan

3631 - Sakshi Jadhav