Assignment B – 1

Aim:

To write a program to solve the 8 Queens Matrix.

Problem Statement:

8-Queens Matrix is Stored using JSON/XML having first Queen placed, use back-tracking to place remaining Queens to generate final 8-queen's Matrix using Python..

Learning Objective:

1. Implementation of the problem statement using Object oriented programming.
2. Generate final 8-queen's Matrix using Python.

.

Theory:

Introduction

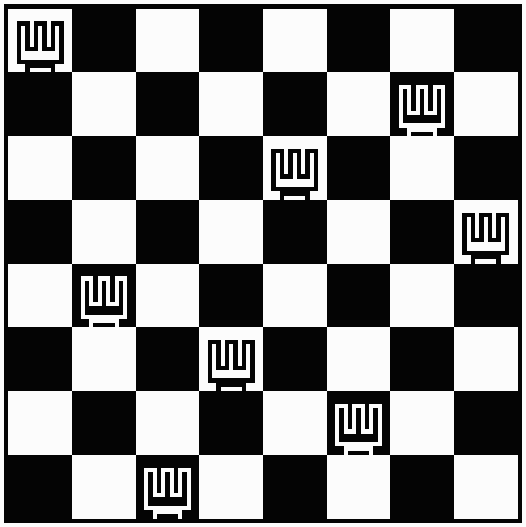
The eight queens puzzle is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. The eight queens puzzle is an example of the more general *n*-queens problem of placing *n* queens on an *n*×*n* chessboard, where solutions exist for all natural numbers *n* with the exception of *n*=2 and *n*=3.

The problem can be quite computationally expensive as there are 4,426,165,368 (*i.e.*, ${}_{64}C_8$) possible arrangements of eight queens on an 8×8 board, but only 92 solutions

The eight queens puzzle has 92 distinct solutions. If solutions that differ only by symmetry operations (rotations and reflections) of the board are counted as one, the puzzle has 12 fundamental solution.

- Find an arrangement of **8** queens on a single chess board such that no two queens are attacking one another.
- In chess, queens can move all the way down any row, column or diagonal (so long as no pieces are in the way).

Due to the first two restrictions, it's clear that each row and column of the board will have exactly one queen.



- The backtracking strategy is as follows:
    1) Place a queen on the first available square in row 1.

    2) Move onto the next row, placing a queen on the first available square there (that doesn't conflict with the previously placed queens).

    3) Continue in this fashion until either:

        a) you have solved the problem, or

        b) You get stuck.

When you get stuck, remove the queens that got you there, until you get to a row where there is another valid square to try.

- When we carry out backtracking, an easy way to visualize what is going on is a tree that shows all the different possibilities that have been tried.

Approach:

- Create a solution matrix of the same structure as chess board.
- Whenever place a queen in the chess board, mark that particular cell in solution matrix.
- At the end print the solution matrix, the marked cells will show the positions of the queens in the chess board.

Algorithm:

1. Place the queens column wise, start from the left most column
2. If all queens are placed.
    1. return true and print the solution matrix.
3. Else
    1. Try all the rows in the current column.
    2. Check if queen can be placed here safely if yes mark the current cell in solution matrix as 1 and try to solve the rest of the problem recursively.
    3. If placing the queen in above step leads to the solution return true.
    4. If placing the queen in above step does not lead to the solution , BACKTRACK, mark the current cell in solution matrix as 0 and return false.
4. If all the rows are tried and nothing worked, return false and print NO SOLUTION.

Better Solution: If you notice in solution matrix, at every row we have only one entry as 1 and rest of the entries are 0. Solution matrix takes $O(N^2)$ space. We can reduce it to O(N). We will solve it by taking one dimensional array and consider solution[1] = 2 as "Queen at 1st row is placed at 2nd column.

Conclusion

Thus, we have successfully implemented and studied the 8- Queens problem.

Mathematical Model:

Let, S be the System Such that,

S= {I, F, O, success, failure}

Where, I = Input O = Output F = Functions (Under attack() and Solve())

Let N=neighbors array contains the following three items: a, b, c.

If we were to unshift each item using neighbors[i], in the first iteration of the for loop, the value of a will be added to the first position in our open List. So far so good.

Conditions For 8 Queen Problem:

1)No two Queens should be in Same rows.

2) No two Queens should be in Same column.

3) No two Queens should be in Same diagonal.

Advantage: Backtracking is possible. Input: I=Set of 8 queens in 8*8 matrix.

Function: F1=In the second iteration, the value of b will be added to the first position in our openList.

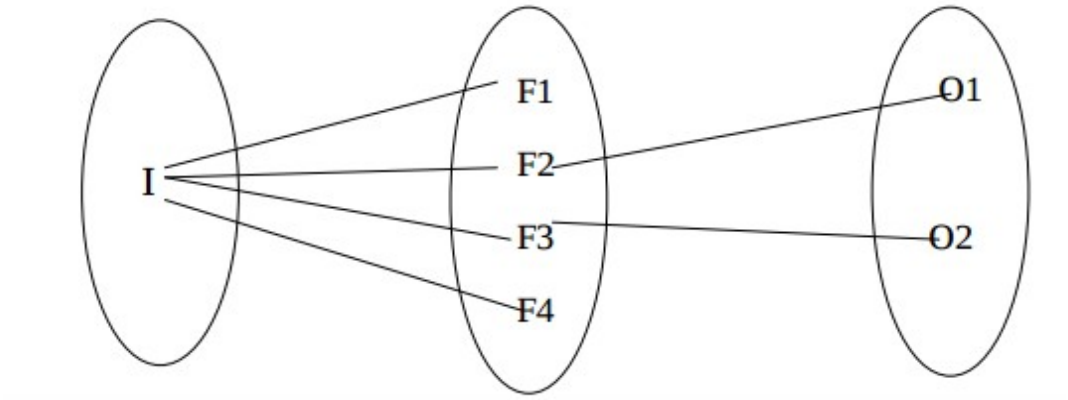
F2= openList contains b first, then a: [b, a].

F3=In the third iteration, our openList will look like [c, b, a].

F4=The end result is that the data is reversed, add the items to our OPENLIST in reverse. First c gets inserted, then b, then a.

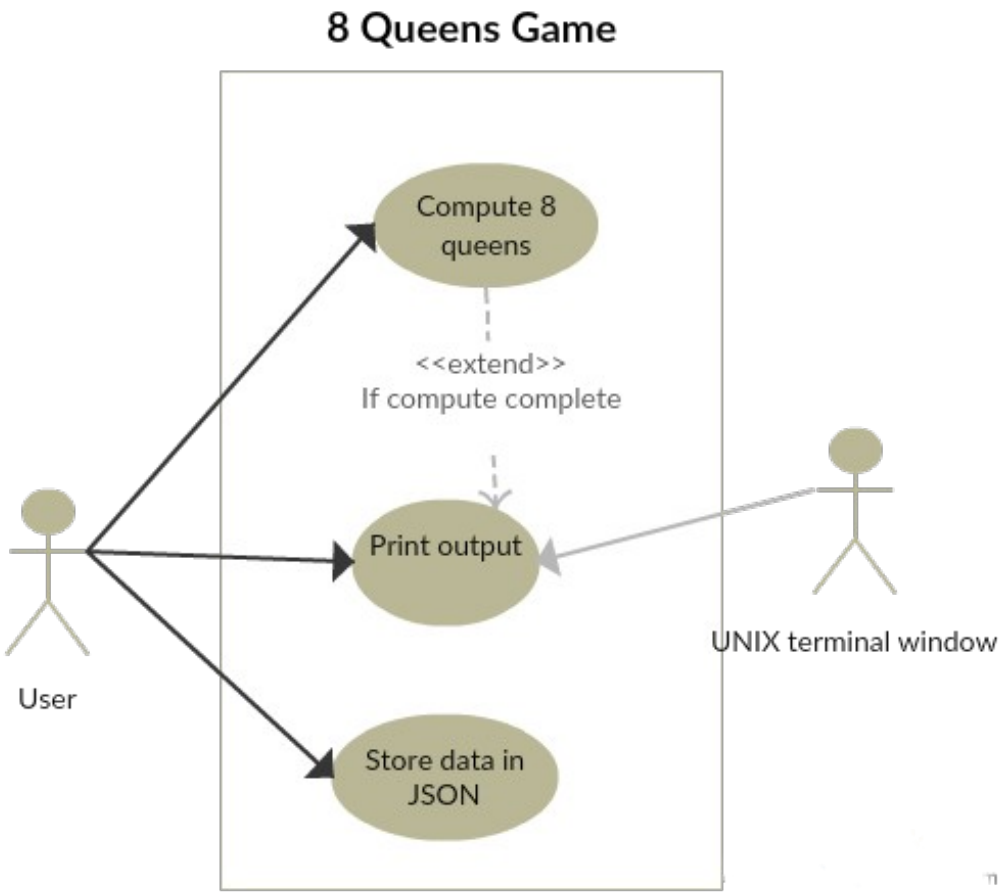
Output: O1=Success Case: It is the case when all the inputs are given by system are entered correctly and 8-Queen problem is solved.

O2=Failure Case: It is the case when the input does not match the validation Criteria.

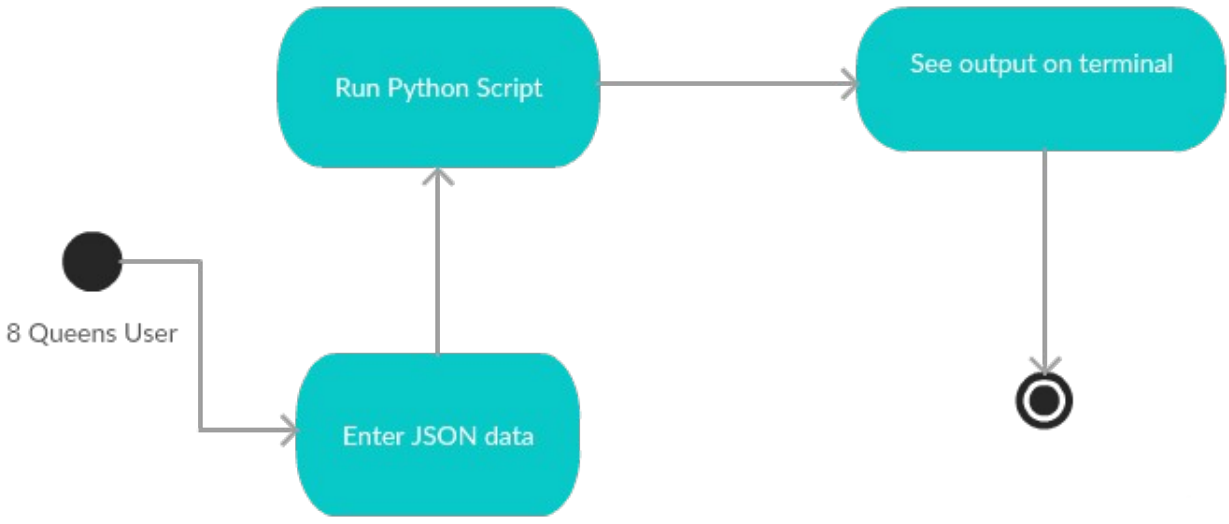


UML Diagrams

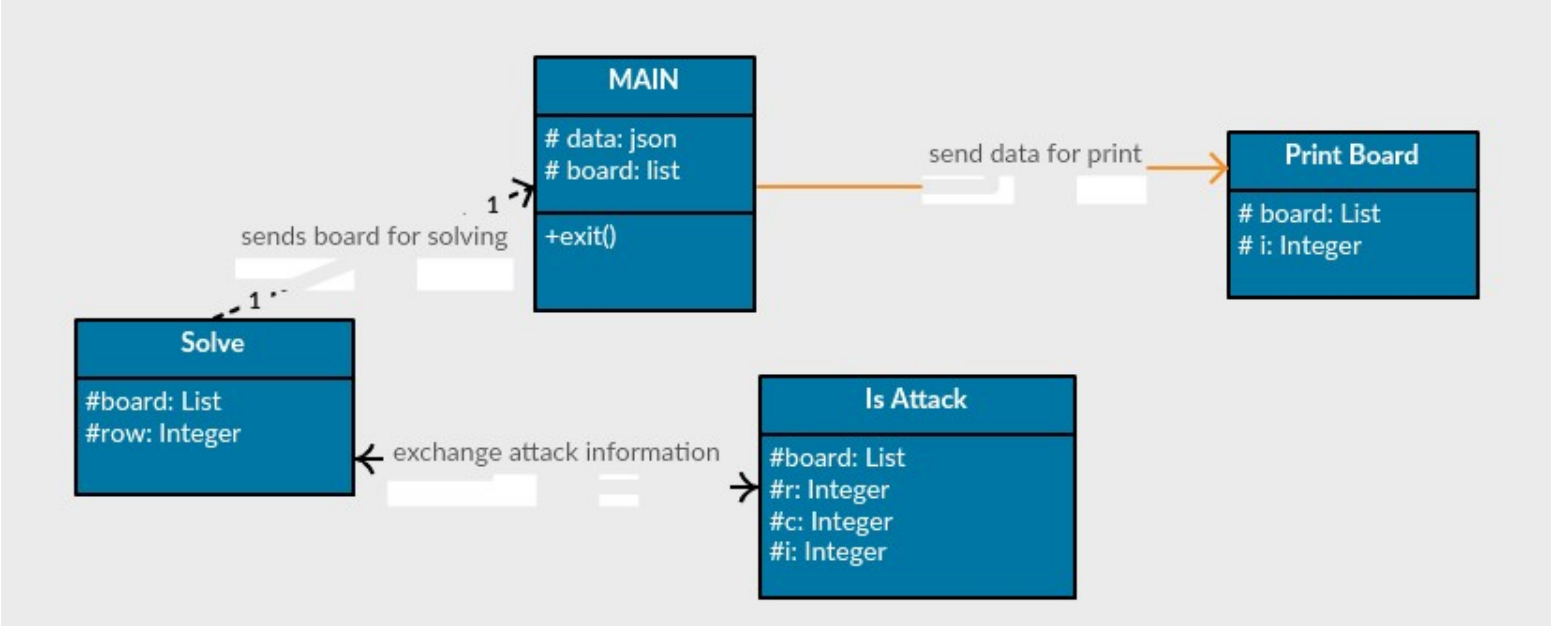
Use Case Diagram



Activity Diagram



Class Diagram



Deployment Diagram