**ASSIGNMENT NO.**

**Title**:8 queen matrix using backtracking.

**Aim:**To generate 8 queen matrix using backtracking method.

**Objective:**To identify the position of 8 queens without any conflict by backtracking method.

**Theory:**

1. 8 Queens Problem:

The 8 queen puzzle is the problem of placing 8 chess queens on 8\*8 chess board. So that no two queens threaten each other. Thus the solution requires no two queens share the same row column and diagonal. The eight queens problem is an example of solving more general n queens problem of placing ‘n’ queens on n\*n chessboard, where solution exists for all natural numbers. ‘n’ with the exception of n=2 and n=3.

The 8 queen’s puzzle has eight variants obtained by rotating 90, 180 and 270 degrees and then reflecting each of the four rotational variants in a mirror in a fixed position.

1. 8 queens using backtracking strategy.

Backtracking is technique is used to solve problems with a large search space by systematically trying and eliminating possibilities. The backtracking strategy for 8 queen’s problem is as follows,

1. Place a queen on the 1st available square in rows /
2. Move onto next row, placing a queen on the first available square there (that desn’t conflict with previously placed queen)
3. Continue in this fashion until either
4. You have solved the problems;
5. You get stack.

When you get stuck, the queen get you there, until you get row where there is another valid square to try.

1. Time complexity using recursive and non recursive algorithm.

Time complexity of 8 queens is O(n-1) using recursive and non recursive algorithms.

**Mathematical Model:**

Let S be the system such that:

S={s,e,X,Y,F,Sc,Fc}

Where,

s= initial state

e= end state

X= set of inputs

Y= set of outputs

F= set of function

Sc= Success cases

Fc= Failure cases

Let S’ be system in observation

Where S’ C S

S’ = {s,e,X,Y,F,Sc,Fc}

* S= start state

{init\_arr }

* e= end state

exit(0) ….success

* X= {(i) | i<8}
* Y= {Y1, Y2} Є Y

Where ,

{Y1,}Є success

{ Y2} Є failure

* F= {F1, F2}

F1 = main()

F2 = queen(i)

* Sc= {Y1,Y2}

where Y1 = {(qj,k) | 1<qj<9, 1<k<9}

* Fc = {Y3}

where Y1 = {(qj,k) | 1>qj>9, 1>k<9}

**Class diagram:**

* Class diagrams are one of the most fundamental diagram types in UML.
* They are used tocapture the static relationships of your software; in other words, how things are put together.
* A class represents a group of things that have common state and behaviour.
* Each class has class name, class attributes and class operations.
* Details of a class (the color of a car, the number of sides in a shape, etc.) are represented asattributes.
* Operations are features of classes that specify how to invoke a particular behaviour.
* Classes in isolation would not provide much insight into how a system is designed. UMLprovides several ways of representing relationships between classes.
* Different relationships are:

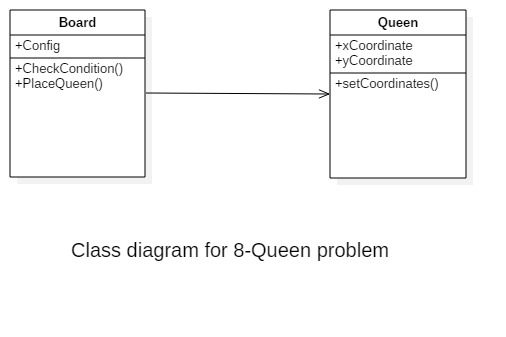
Dependency

Association

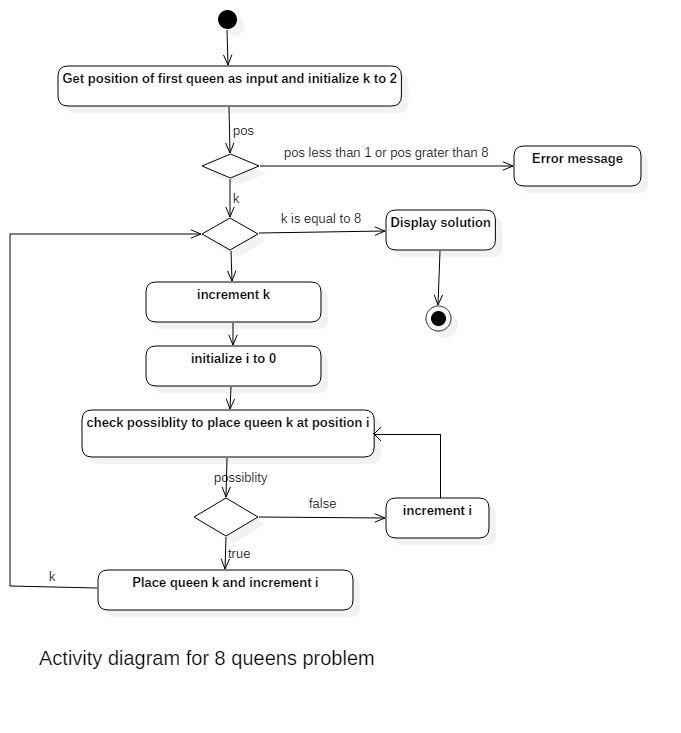
Aggregation

Composition

Generalization



**Activity Diagram:**



**Manual Testing:**

**Test Case Name:** 8 Queen’s Position

**Test Case ID:** 8QP

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No** | **Test Case Name** | **Input** | **Expected o/p** | **Actual o/p** | **Status** |
| **1** | 8 Queen’s Problem | X=1 | [1, 0, 0, 0, 0, 0, 0, 0],  [0, 0, 0, 0, 0, 0, 1, 0],  [0, 0, 0, 0, 1, 0, 0, 0],  [0, 0, 0, 0, 0, 0, 0, 1],  [0, 1, 0, 0, 0, 0, 0, 0],  [0, 0, 0, 1, 0, 0, 0, 0],  [0, 0, 0, 0, 0, 1, 0, 0],  [0, 0, 1, 0, 0, 0, 0, 0] | [1, 0, 0, 0, 0, 0, 0, 0],  [0, 0, 0, 0, 0, 0, 1, 0],  [0, 0, 0, 0, 1, 0, 0, 0],  [0, 0, 0, 0, 0, 0, 0, 1],  [0, 1, 0, 0, 0, 0, 0, 0],  [0, 0, 0, 1, 0, 0, 0, 0],  [0, 0, 0, 0, 0, 1, 0, 0],  [0, 0, 1, 0, 0, 0, 0, 0] | Pass |
| **2** | 8 Queen’s Problem | X=7 | [0, 0, 0, 0, 0, 0, 1, 0],  [0, 0, 1, 0, 0, 0, 0, 0],  [0, 0, 0, 0, 0, 0, 0, 1],  [0, 1, 0, 0, 0, 0, 0, 0],  [0, 0, 0, 0, 0, 0, 0, 1],  [1, 0, 0, 0, 0, 0, 0, 0],  [0, 0, 0, 0, 0, 1, 0, 0],  [0, 0, 0, 1, 0, 0, 0, 0] | Error message: No solution available | Fail |
| **3** | 8 Queen’s Problem | X=10 | Display Error: invalid input | Display Error: invalid input | Pass |
| **4** | 8 Queen’s Problem | X=a | Display Error: : invalid input | System crash | Fail |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Input:**

Position of first queen.

**Output:**

Placement of all queens.

**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.

**Platform:**

Ubuntu 12.04

**Language:**

Python

**Conclusion:**

        Hence, We Implement and design 8-Queens Matrix.