

## EXPERIMENT - 1

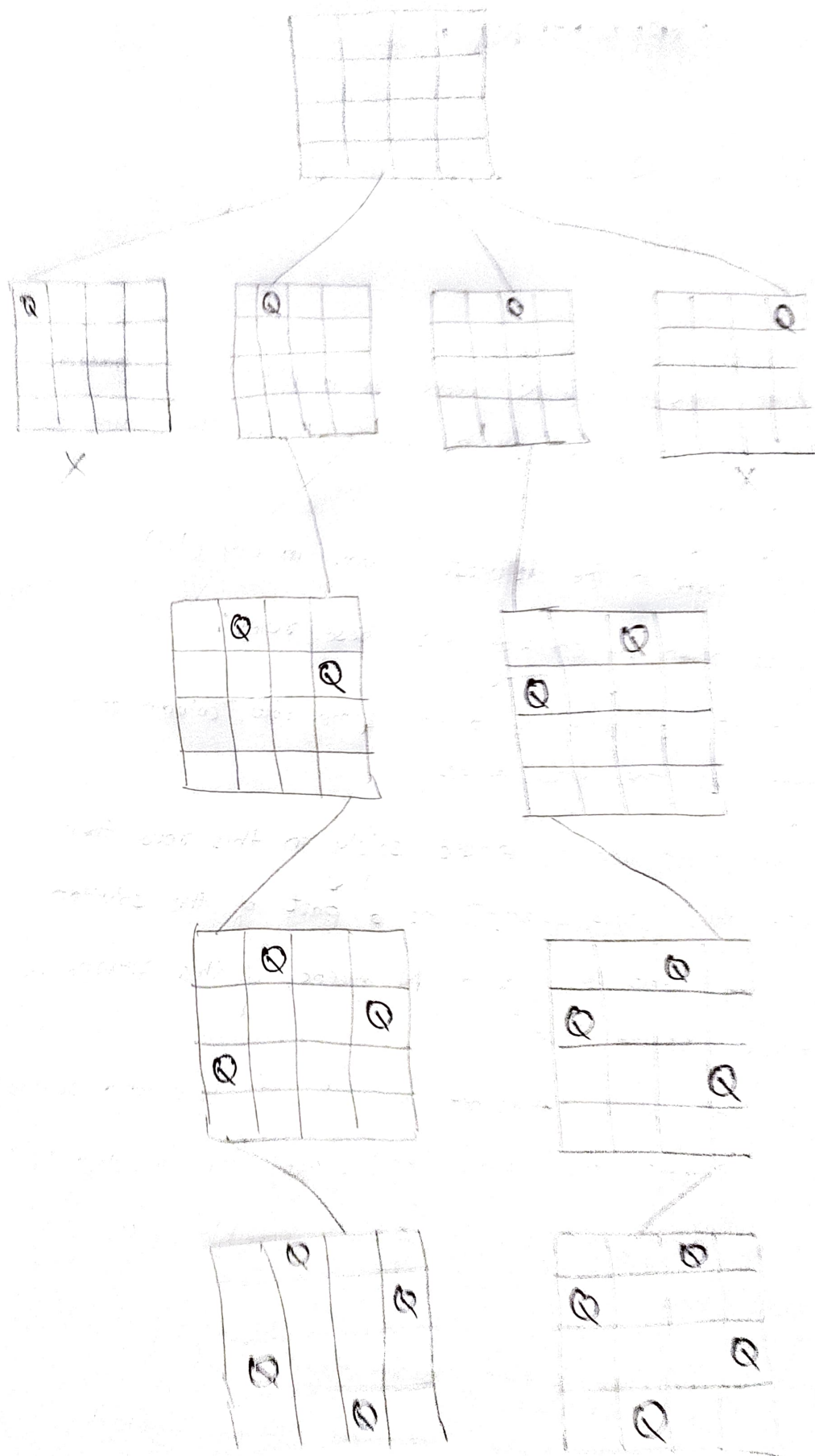
## AIM

To implement the N-Queens (Toy Problem) using python programming.

## ALGORITHM

- 1) N-queens problem aims to place N-queens on a  $N \times N$  chess-board such that no two queens can attack each other in the same row column or diagonally
- 2) Place the first queen in the leftmost column in cell (1,1)
- 3) Place the second queen by checking for these rules:
  - i) The queen shouldn't be in the same row, column or diagonal as the first queen
  - ii) If the queen can be placed safely in this row then make this [row, column] as a part of the solution
  - iii) Repeat step (ii) for all N queens, if this returns a solution return true.
  - iv) If placing this queen doesn't lead to a solution then unmark this [row, column]. Back track to step (2)
  - v) Repeat the step from (3) for all possible cells in the first row.
- 4) If all the queens are placed, return true.
- 5) If all the rows have been tried and nothing worked, return false to trigger backtracking.

# STATE SPACE TREE



## CODE AND OUTPUT SCREENSHOT

The screenshot displays a web-based IDE interface. The top bar shows the user 'Dr. M.Ferni Ukrit / Ms.J.Sujithra-B' and the AWS console URL 'us-east-2.console.aws.amazon.com/cloud'. The left sidebar contains a file explorer with a tree view of folders (063 to 077) and files (exp1.py, lab3, README.md). The main editor area shows the code for 'exp1.py', which implements a recursive solution for the N-Queens problem. The code defines an 'attack' function to check for conflicts, an 'N\_queens' function to recursively find solutions, and a main loop to print all solutions for N=8. The bottom panel shows the terminal output, where the user runs 'python exp1.py', enters '8', and the program prints 8 different board configurations as lists of 8 integers.

```
1 print ("Enter the number of queens")
2 N = int(input())
3
4 board = [[0]*N for _ in range(N)]
5
6 def attack(i, j):
7     for k in range(0,N):
8         if board[i][k]==1 or board[k][j]==1:
9             return True
10
11     for k in range(0,N):
12         for l in range(0,N):
13             if (k+l==i+j) or (k-l==i-j):
14                 if board[k][l]==1:
15                     return True
16     return False
17
18 def N_queens(n):
19     if n==0:
20         return True
21     for i in range(0,N):
22         for j in range(0,N):
23             if (not(attack(i,j))) and (board[i][j]!=1):
24                 board[i][j] = 1
25                 if N_queens(n-1)==True:
26                     return True
27                 board[i][j] = 0
28     return False
29
30 N_queens(N)
31
32 for i in board:
33     print (i)
```

```
Mferni:~/environment/75/Lab2 $ python exp1.py
Enter the number of queens
8
[1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0]
Mferni:~/environment/75/Lab2 $
```