EXNO: 01
DATE:

N-Queens Problem

AIM:-

write a program to solve n-queens problem with python code.

ALMORITHM:-

Step 1: Initialize the board:

→ create an NXN chessboard initialized with all zero ('0') to indicate no queen placed on that cell.

Step 2: Place queens column by column:-

→ Start with the first column

→ Try placing queen in each row of the current column.

Step 3: check if placing queen is safe:-

→ No queens is in the same row

→ No others is in the same upper diagonal

→ No other queen is in the same lower diagonal.

Step 4: Recursion:-

→ If a safe position is found, place the queen ('1') and move to next column using recursion.

Step 5: Base case:-

→ If all queens are successfully placed the algorithm returns 'True' indicating that the solution is found.

Step 6: Backtrack:-

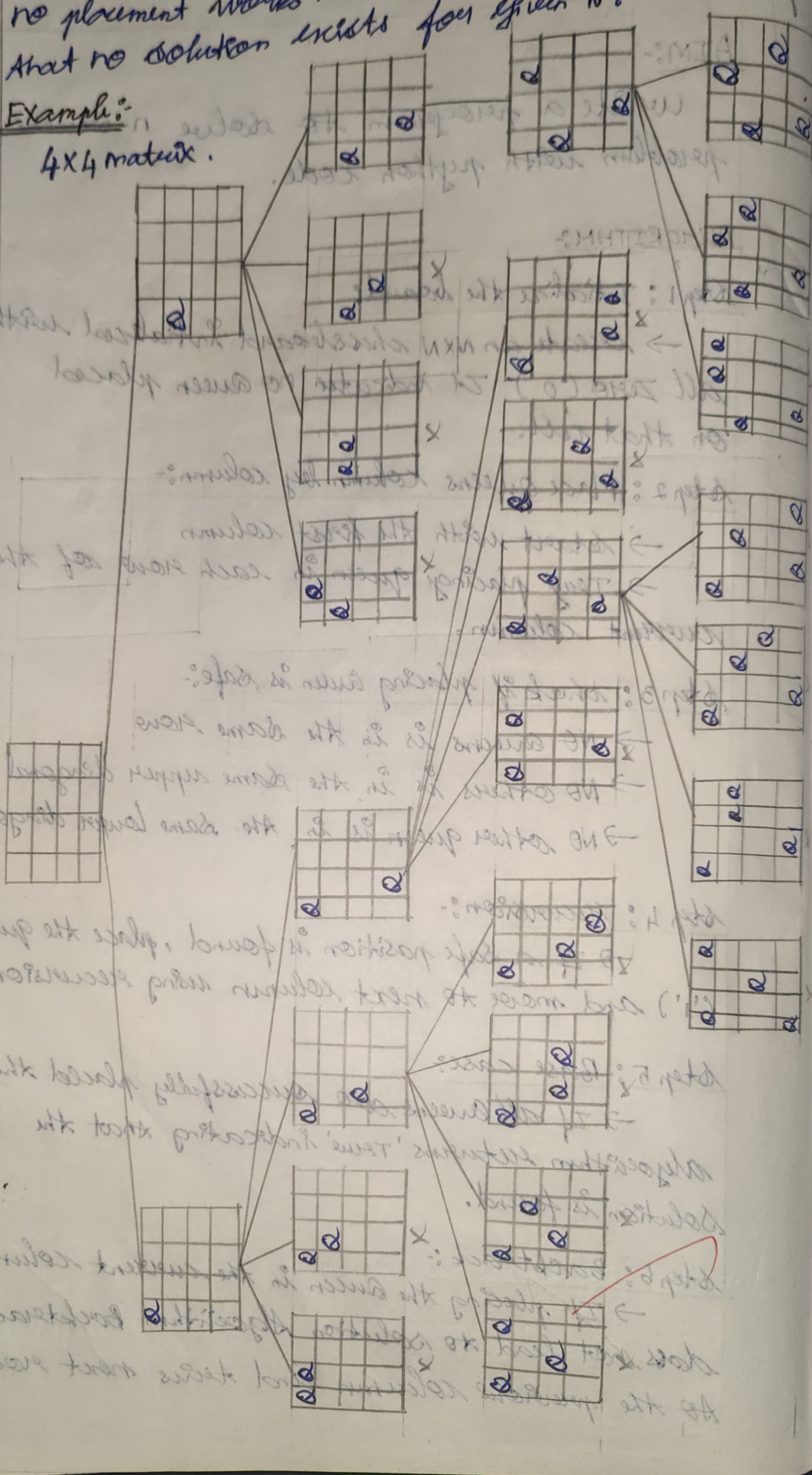
→ If placing the queen in the current column does not lead to solution Algorithm backtracks to the previous column and tries next row.

Step 7: No solution:-

→ If algorithm exhausts all possibilities and no placement works return 'False' indicating that no solution exists for given 'N'.

Example:-

4×4 matrix.



Code:-

```
def is_safe(board, row, col, N):
    for i in range(col):
        if board[row][i] == 1:
            return False

    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    for i, j in zip(range(row, N, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solve_n_queens(board, col, N):
    if col >= N:
        return True

    for i in range(N):
        if is_safe(board, i, col, N):
            board[i][col] = 1

            if solve_n_queens(board, col+1, N):
                return True

            board[i][col] = 0
    return False

def print_board(board, N):
    for row in board:
        print(" ".join(str(cell) for cell in row))

N = int(input("Enter the size of board(N): "))
board = [[0] * N for _ in range(N)]
if solve_N_queens(board, 0, N):
    print("solution found: ")
    print_board(board, N)
else:
    print("NO solution exists")
```

Output:-

* Enter the size of the board (N): 4

Solution found:

0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0

* Enter the size of the board (N): 8

Solution found:

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

~~Output~~:-

Result:-

Thus, the N-Queen problem is executed and output is verified successfully.