BACKTRACING

```
def is_safe(board, row, col):
    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, len(board)), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def nqueens(n):
    board = [[0]*n for _ in range(n)]
    solutions = []  # empty list of solutions

    def backtrack(col):   #when the no. of rows and no. of columns are equal it appends everything
        if col == n:
            solutions.append([list(row) for row in board])
            return

        for row in range(n):
            if is_safe(board, row, col):
                board[row][col] = 1
                backtrack(col+1)
                board[row][col] = 0
```

```
    backtrack(0)
    return solutions

n = int(input("Enter the board (size) : "))

solutions = nqueens(n)

print(f"Number of solutions {len(solutions)}")

for i, solutions in enumerate(solutions):
    print(f"\nSolution {i+1}:")

    for row in solutions:
        print(" ".join(["Q" if cell== 1 else "-" for cell in row]))
```