**N Queens**

bool isSafe(int row, int col, vector<vector<int>>& board, int n)

{

for (int r = 0; r < row; r++)

{

if (board[r][col] == 1)

return false;

}

int r = row;

int c = col;

while (r >= 0 && c >= 0)

{

if (board[r][c] == 1)

return false;

r--;

c--;

}

r = row;

c = col;

while (r >= 0 && c < n)

{

if (board[r][c] == 1)

return false;

r--;

c++;

}

return true;

}

void solveNQueensUtil(int row, int n, vector<vector<int>>& board, vector<vector<int>>& solutions)

{

if (row == n)

{

vector<int> temp;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

temp.push\_back(board[i][j]);

}

}

solutions.push\_back(temp);

return;

}

for (int col = 0; col < n; col++)

{

if (isSafe(row, col, board, n))

{

board[row][col] = 1;

solveNQueensUtil(row + 1, n, board, solutions);

board[row][col] = 0;

}

}

}

vector<vector<int>> solveNQueens(int n)

{

vector<vector<int>> board(n, vector<int>(n, 0));

vector<vector<int>> solutions;

solveNQueensUtil(0, n, board, solutions);

return solutions;

}