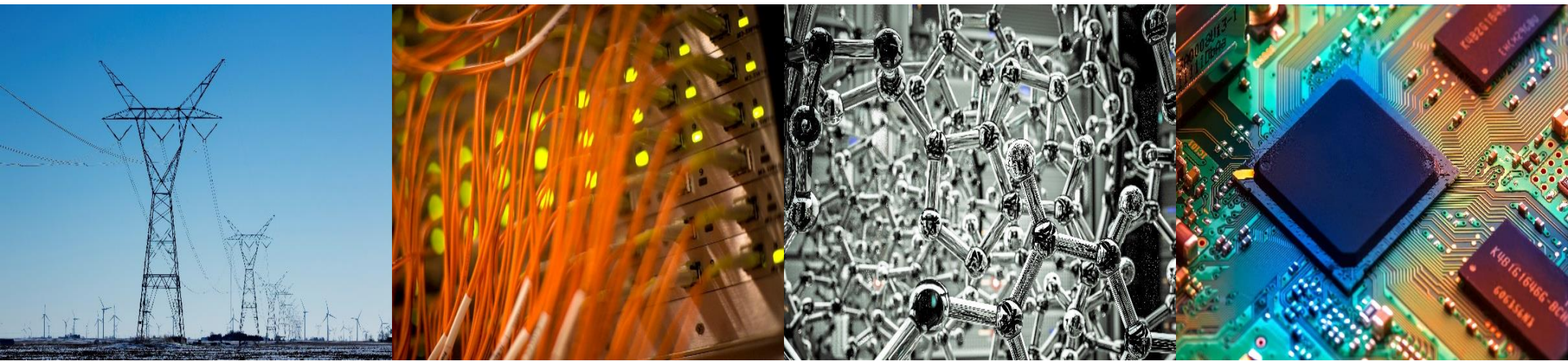


ECE 220 Computer Systems & Programming

Lecture 13 – Recursion with backtracking, C to LC-3 Conversion

October 9, 2025



I ILLINOIS

Electrical & Computer Engineering

GRAINGER COLLEGE OF ENGINEERING

Recursion Recap

- Solving a problem by **calling itself** on smaller pieces of data
- Must have at least 1 **base case** and at least 1 **recursive case**
- Similar to recurrence (using loops) but can result in **simpler implementation**
- Can incur **heavy overhead** on the Run-Time Stack (Good vs. Bad Recursion)

Recursion with Backtracking: n-Queen Problem

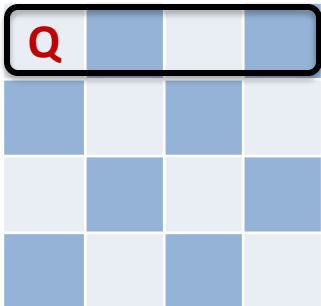
1. Find a safe column (from left to right) to place a queen, **starting at row 0**;
2. If we find a safe column, make recursive call to place a queen on the next row;
3. If we cannot find one, backtrack by returning from the recursive call to the previous row and find a different column.

	0	1	2	3
0		Q		
1				Q
2	Q			
3			Q	

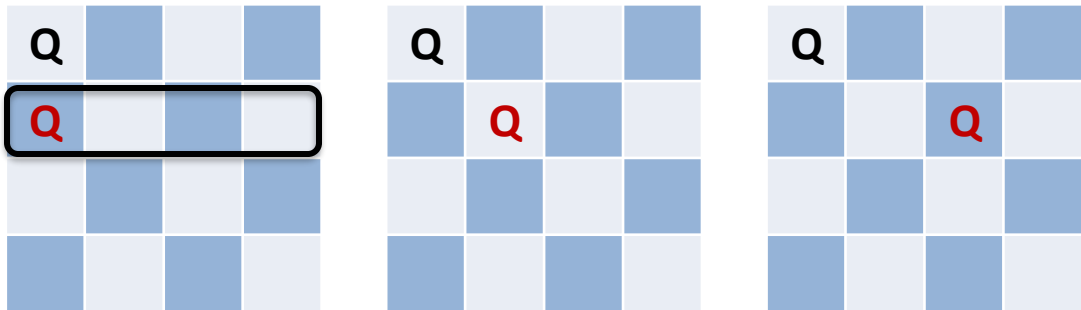
0	1	0	0
0	0	0	1
1	0	0	0
0	0	1	0

Example: 4x4 n-Queen

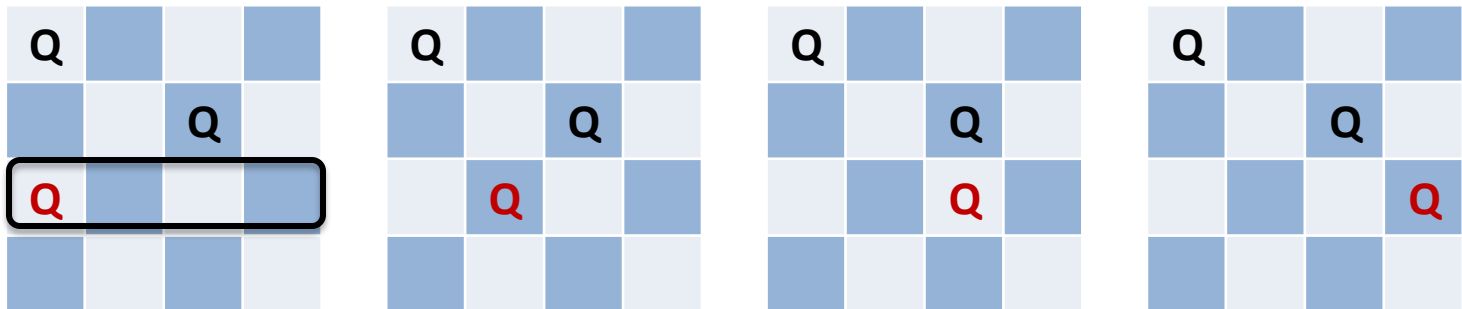
row 0:



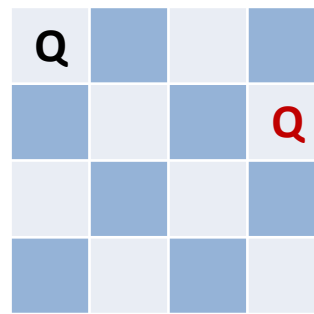
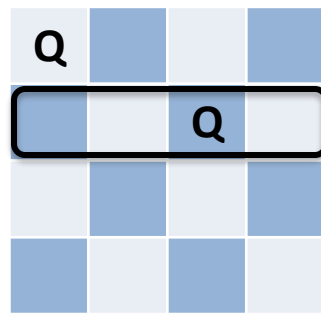
row 1:



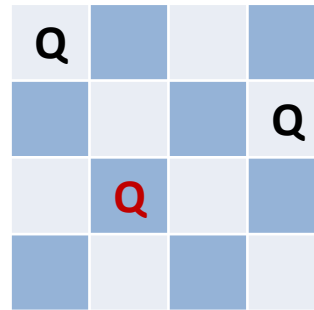
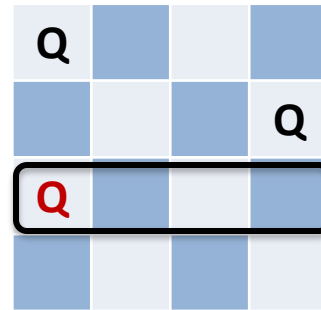
row 2:



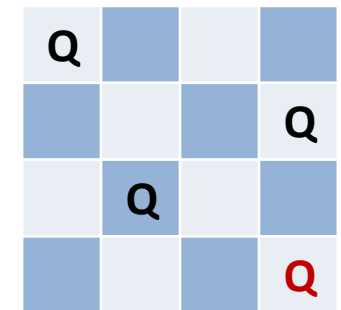
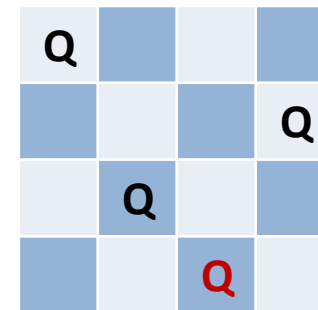
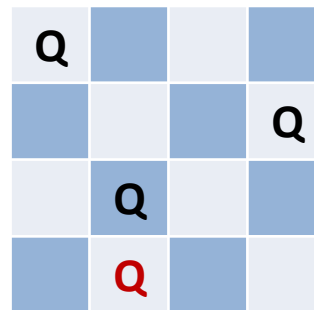
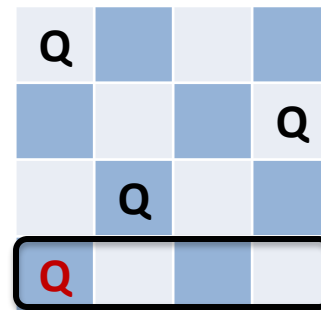
*Backtrack
to row 1 and
make a new
choice:*



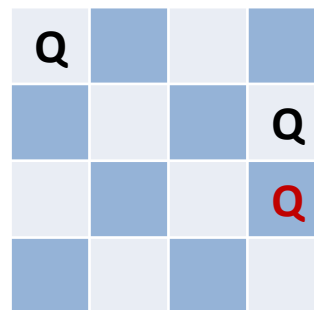
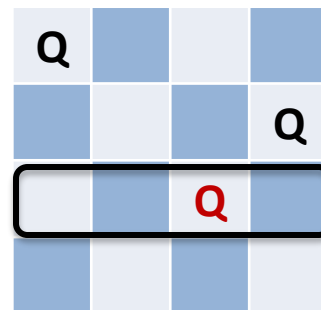
row 2:



row 3:

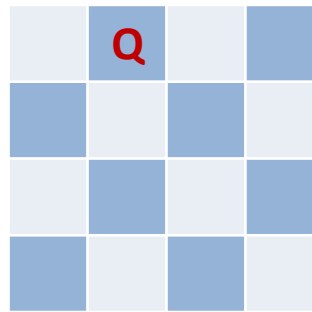
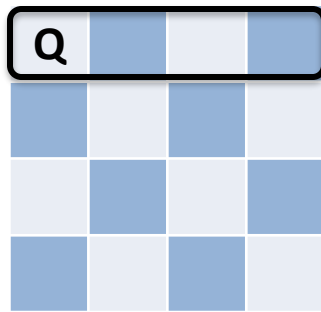


*Backtrack
to row 2 and
make a new
choice:*

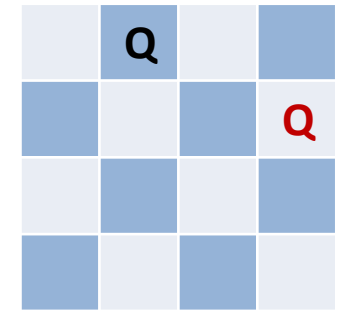
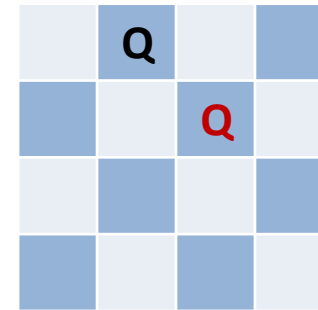
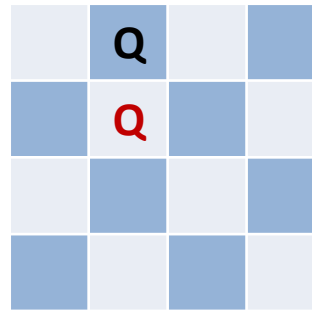
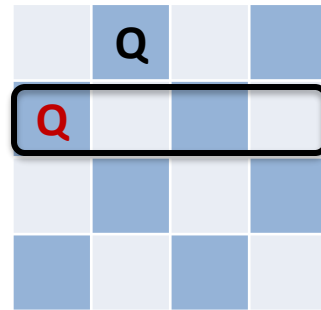


(Backtrack to row 1,
but no columns left)

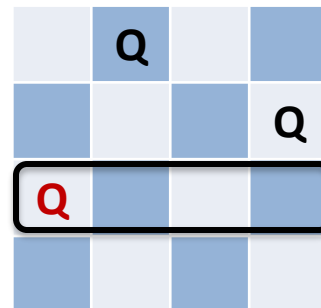
*Backtrack to row 0
and make a new
choice:*



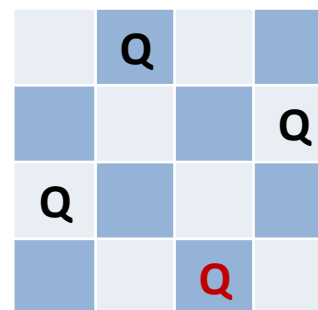
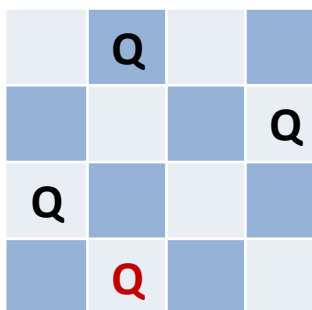
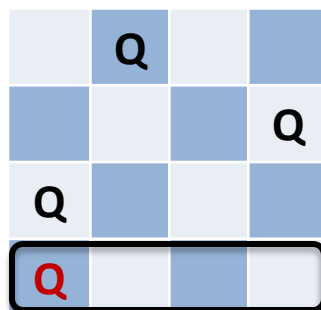
row 1:



row 2:



row 3:



/* isSafe() is a helper function to check whether it's safe to place a queen at board[row][col].

If it's safe, return 1; otherwise, return 0. */

```
int isSafe(int board[N][N], int row, int col){
```

```
}
```

Recursion with Backtracking Template

```
bool solve (configuration conf){
    if (no more choices) /*base case*/
        return (config is goal state);

    for(all available choices){
        try one choice c;
        /*recursively solve after making choice*/
        ok = solve(config with choice c made);
        if (ok)
            return true;
        else
            unmake choice c;
    }

    return false; /*tried all choices and no solution found*/
}
```

```

int nqueen(int board[N][N], int row){

    /*base case - reach solution, no more rows to place queen*/
    if(
        )
        return 1;

    /*recursive case with backtracking*/
    int c;
    for(c=0; _____; c++){ /*find a safe col to place queen*/
        /*if col 'c' is safe, place queen here and then solve
           subsequent steps recursively*/
        if(isSafe(
            ) == 1){

            board[
                ][
                ] = 1;

            /*continue to solve the next step*/
            if(nqueen(
                ) == 1)
                return 1;

            /*could not solve subsequent steps, backtrack*/
            board[
                ][
                ] = 0;
        }
    }
    return _____;
}

```