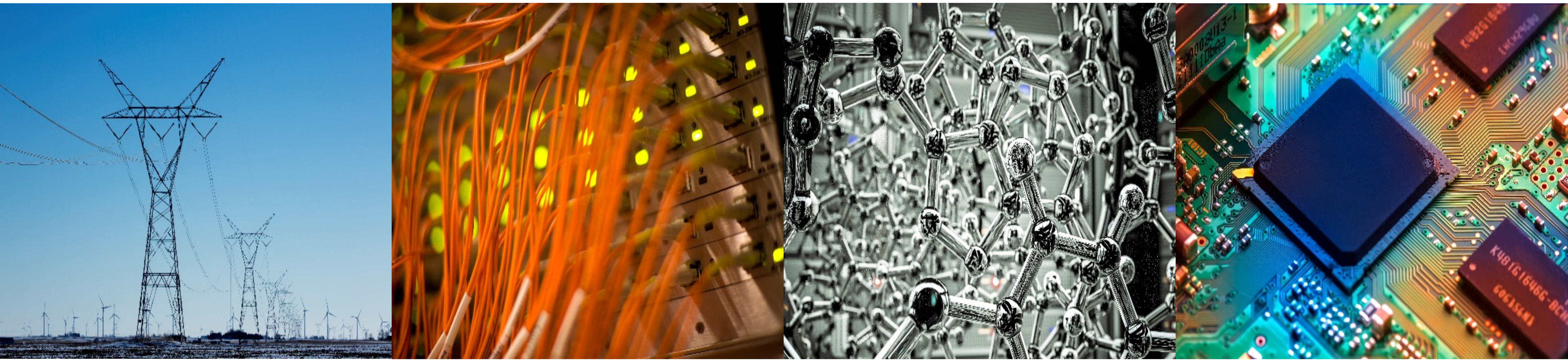
ECE 220 Computer Systems & Programming

Lecture 20 – C to LC-3 Conversion, Recursion with Backtracking

July 15, 2020



ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

- MT2 past exam & practice questions posted
- Informal Early Feedback

Stack Built-up and Tear-down

Caller function	**1. caller built-up** (push callee’s arguments onto stack)
	2. pass control to callee (invoke function)
Callee function	**3. callee built-up** (push bookkeeping info and local variables onto stack)
	4. execute function logic
	5. callee tear-down (pop local variables, caller’s frame pointer, and return address from stack)
	6. return to caller
Caller function	**7. caller tear-down** (pop callee’s return value and arguments from stack)

```
;;convert Factorial function to an LC-3 subroutine
FACTORIAL

;;callee built-up of Factorial(n)'s activation record
;push return value, return address & caller's frame pointer
;push local variable & update frame pointer

;;function logic
;base case skipped here for simplicity

;recursive case
;caller built-up of Factorial(n-1)'s activation record
;push argument n-1 on to RTS

;pass control to Factorial(n-1)

;caller tear-down of Factorial(n-1)'s activation record
;pop return value from Factorial(n-1)
;pop argument from Factorial(n-1)
;calculate n*Factorial(n-1)
;remaining function logic skipped for simplicity

;;callee tear-down of Factorial(n)'s activation record
;pop local variable
;restore caller's frame pointer and return address

;;return to caller
```

Recursion with Backtracking: n-Queen Problem

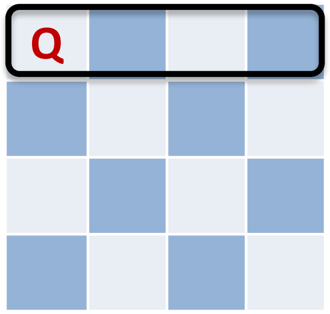
1. Find a safe column (from left to right) to place a queen, starting at row 0;
2. If we find a safe column, make recursive call to place a queen on the next row;
3. If we cannot find one, backtrack by returning from the recursive call to the previous row and find a different column.

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

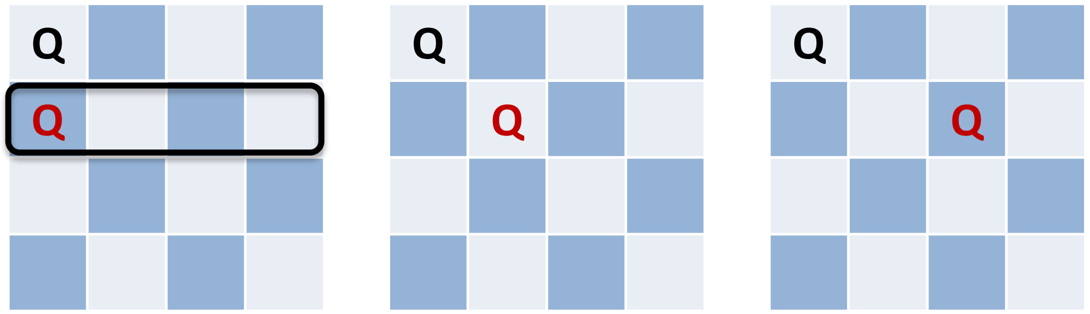
0	1	0	0
0	0	0	1
1	0	0	0
0	0	1	0

Example: 4x4 n-Queen

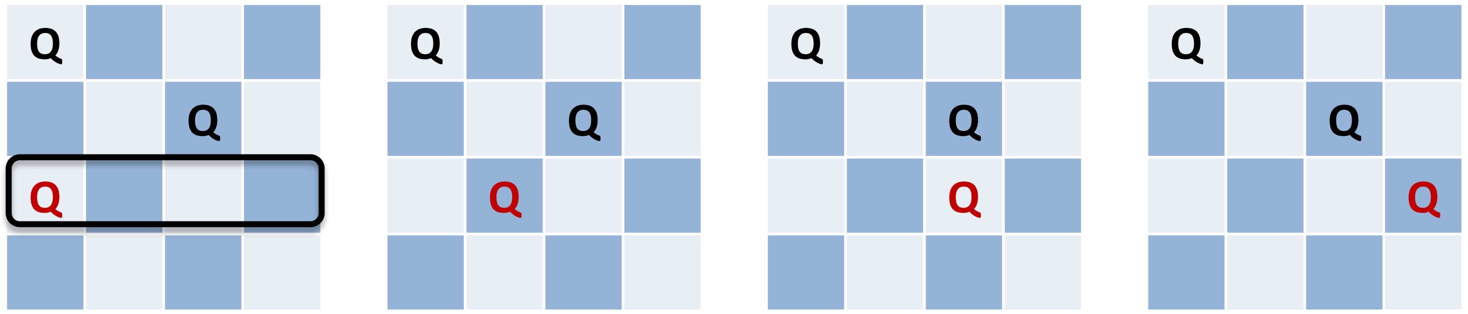
row 0:



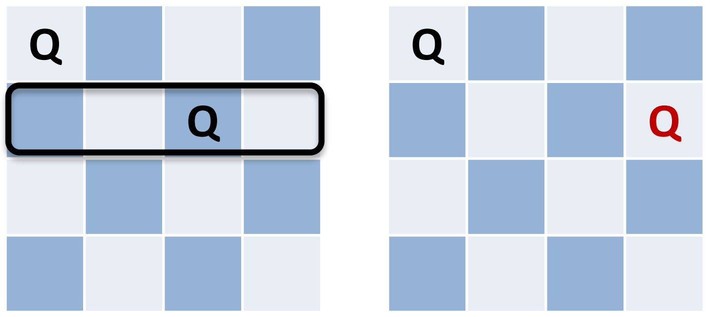
row 1:



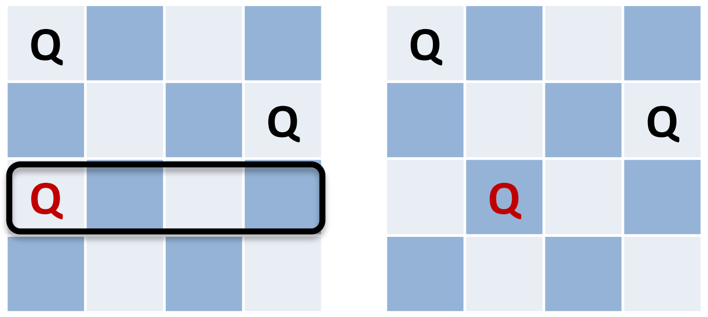
row 2:



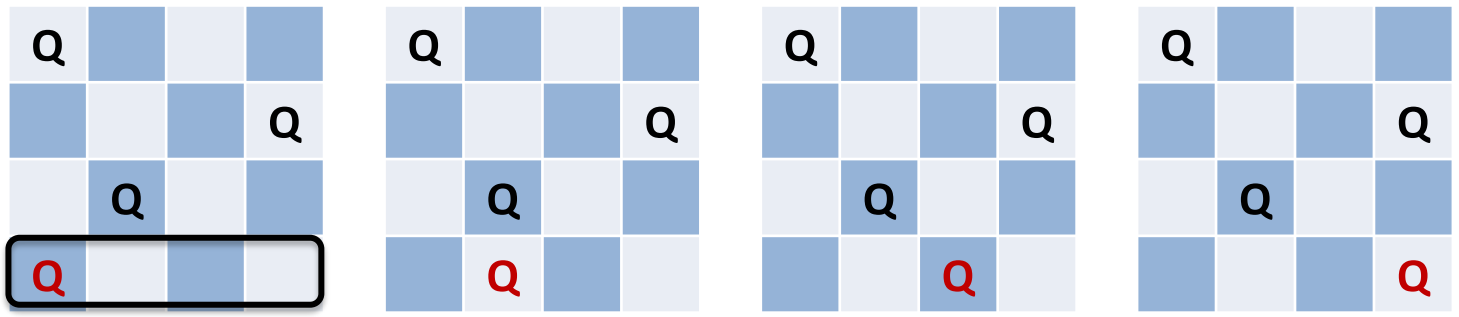
Backtrack to row 1 and make a new choice:



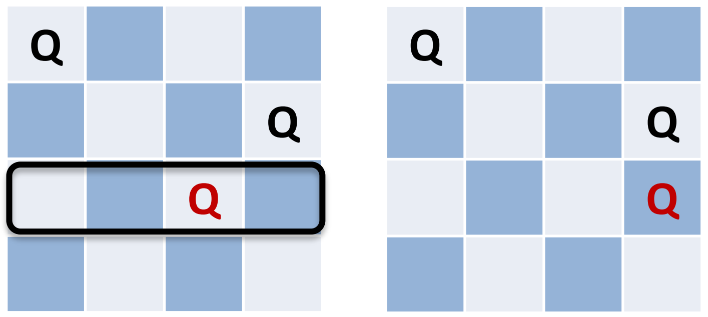
row 2:



row 3:

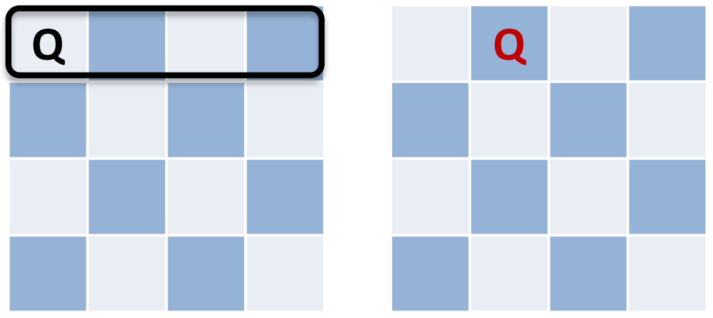


Backtrack to row 2 and make a new choice:

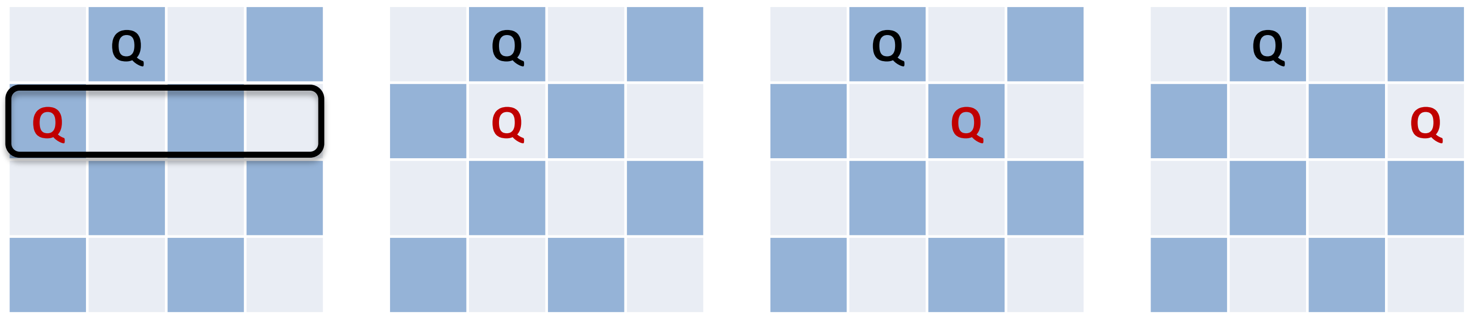


(Backtrack to row 1, but no columns left)

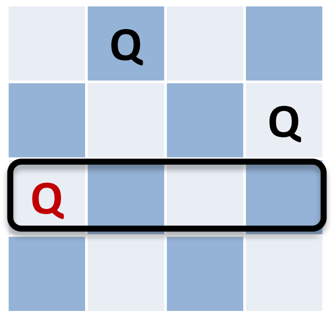
Backtrack to row 0 and make a new choice:



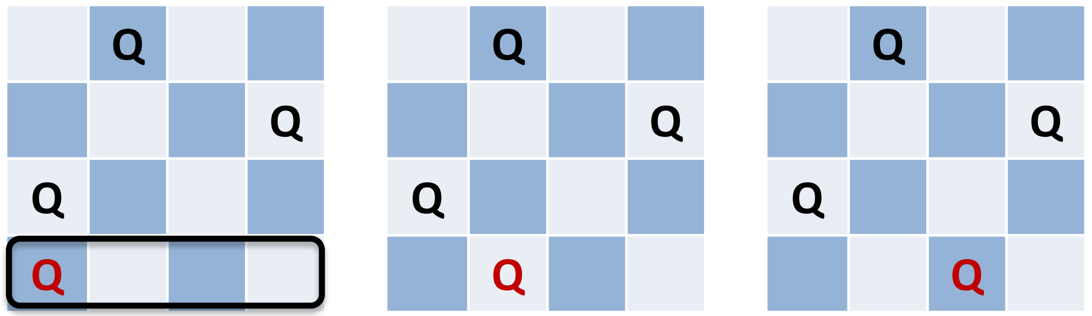
row 1:



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 Q*/



      /*recursive case with backtracking*/
















}
```