CS330 – Backtracking

Purpose: Relationships may not be expressed explicitly using a graph but instead may be represented implicitly by the problem constraints. A backtracking algorithm may by used to search through the implicit graph.

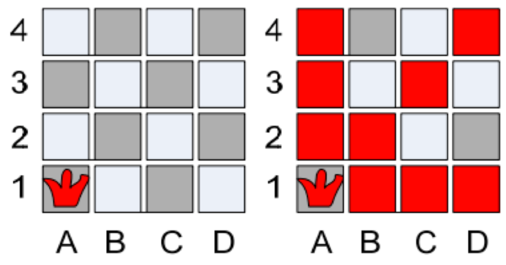
Knowledge: This activity will help you become familiar with the following content knowledge:

- backtracking in an implicit graph.

Activity: With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 15 minutes.

1. Relationship graphs for problems may not be explicit but instead may be implicit. For example, consider the n queens problem where we want to place n queens on an n by n chess board so that no queen is on the same row, column, or diagonal of any of the other queens.

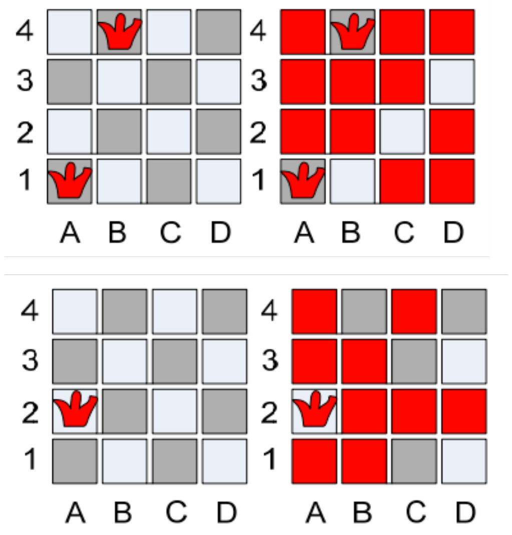
   Consider the scenario where we have placed the first queen at column A and row 1.

   What choices do we have for placing the next queen in column B?

   After placing a queen in B3, what choices do we have for placing a queen in column C?
   What should we do next?

2. We need to backtrack and make another choice for the placement of the second queen. This search technique where we make attempts and then undo the choices if there is a conflict is called **backtracking**.



Complete the **recursive** backtracking algorithm for the n queens problem.

```
bool queens(board,col, n){
     if (col < n){ // there are more queens to place
            for (int row = 0; row < n; row++){ // look at the possible placements
                    board[col] = row
                    if (noconflicts(board){
                           ...
                    }
           }
           return false // none of the rows work
     }
     return true
}
```