COMP 333
Eight Queens Java Solution

Eight Queens is a famous mathematical puzzle inspired by the game of chess and the rules for valid moves and attacks of the queen. See Wikipedia article for some background.
https://en.wikipedia.org/wiki/Eight_queens_puzzle

The goal of 8 queens is to start with an empty 8-by-8 chess board and 8 queens, then find positions on the board for all 8 queens in such a way that no queen attacks another queen. Based on the rules of the queen's move in chess, there can be only 1 queen per row, only 1 queen per column, and only 1 queen along any diagonal.

The puzzle is well studied and it has been shown that there are 92 unique solutions for the 8-by-8 case. Within this set of solutions, groups of solutions can be defined such that all solutions in a group are related to one another via rotations or reflections. Counting this way there are 12 fundamental solution groups. In the program you write for this project, you will focus on generating the 92 unique solutions without worrying about which solution group they belong to.

A generalized version of the problem is called N Queens where we use an N-by-N board instead of the regular 8-by-8 one. If we consider N to be the dimension of the problem, then we can talk about its time complexity with respect to board dimension N. Since there is no efficient (polynomial) algorithm for finding the positions of the queens, solutions are found by searching, which means it is $O(N^N)$ or exponential, and it can furthermore be shown to be NP-complete. A typical search works by positioning the queens one at a time. For each newly added queen, we pick a position that is not in conflict with the queens already on the board. The problem is that the chances of simply sticking 8 pieces on the board and having it be a solution are very small. Even restricting arrangements to one queen per column gives 8^8 = 16,777,216 possible arrangements, and only 92 of them are solutions. During the positioning, we commonly get into a situation where there is no position left for the new queen that is not in conflict with some previously positioned queen. At that point we have to **backtrack** by returning to a previously positioned queen and finding another position for it, then trying to move forward again with positioning the remaining queens. This approach will find all 92 solutions, but it requires bookkeeping to keep track of which positions have been previously tried so that we don't get stuck in an infinite loop trying positions that were already checked.

Since Eight Queens is fundamentally a **search** problem, it is a good problem for Prolog programming. Prolog has many built-in features to manage the searching and bookkeeping automatically, so the programmer doesn't need to program it. For this reason, a Prolog solution will be quite compact. It only describes the logical relationships and constraints. There is no need for the programmer to define how to do bookkeeping and search details as part of the program, since these are handled automatically by the Prolog interpreter. The Prolog interpreter keeps searching (matching plus backtracking) to find one or more solutions that meets all our constraints.

Although the Prolog program will be short, that doesn't mean it will be simple to understand. Even short Prolog programs can be challenging to read and understand. Before trying to solve the problem in

Prolog, you are required to write a Java solution and submit it with your Prolog solution. This is to ensure that you first understand the problem before trying to solve it in Prolog, which will be hard enough even if you understand the problem. A Java solution to the problem is outlined below.

Variations
It is easy to generalize the Eight Queens problem to the N Queens problem by imagining a hypothetical chess board of dimension n-by-n. Also, there is a small difference between a program that finds the first solution and stops, and a program that finds all 92 solutions. We will write a solution for N Queens that finds all solutions (92 if N=8, 352 if N=9, 724 if N=10, etc.).

Java Solution
We will start with an empty n-by-n board and position the queens by iterating over the columns 0 through n-1. We will keep track of the current column in a variable **curcol**. We will use a large while loop that tries to position a queen in the current column. If we successfully find a position for a queen for curcol, we record its position, increment curcol, and let the loop move us to the next column. If we cannot find any valid position for a queen in curcol, we must backtrack by clearing the record of tried positions for curcol, then decrementing it, and letting the loop move us back to the previous column, where we must find another position for the queen in that column. If all goes well, we will start moving through the columns again in the positive direction.

If the value of curcol reaches n as it moves in a positive direction, this means that we have found a solution. We backtrack to the previous column, and continue from that point to find the next solution. If the value of curcol reaches -1 as it moves in a negative position, this means that we have exhausted all possible positions and there are no more solutions to be found.

There are two state values that must be maintained for each column: the current position of the queen for that column, and the number of the next row in that column that has not yet been tried.
We will maintain a couple of static constants for state indicators:

- **Unassigned**: used to indicate a column has no queen positioned assigned currently
- **NoPosAvail**: used to indicate that there is no position available to place a queen in the column.

Data Structures

```
int n;              // holds size of board
int curcol;         // holds value of the current column, initialized to 0
int[] queenPos;     // array of queen positions for each row, initialized to Unassigned for all values
int[] nextAvail;    // array of next available positions for each row, initialized to 0 for all values
```

For any column we must keep track of which values or rows for that column have already been tried and which ones have not. We can simplify the bookkeeping if we try values in order starting at 0. In this case, we don't have to mark each individual value as tried or not tried, we only have to keep track of the next value to try, and assume all smaller values have been tried. For example, nextAvail[3] is 5 means that for column 3, all row values 0 through 4 have been tried, and the next one to try is row 5.

Main Loop

This loop runs the search for all solutions:

```
Get value for n
Create and initialize queenPos and nextAvail arrays
Set curcol to 0

While (true) {                          can also use “while (curcol >= 0) { … }”
    If (curcol==n) {
        One solution found, and we have advanced off the right end of the board
        to search for next solution
            backtrack to previous column (curcol = curcol – 1)
            undo queen position for that column
            but don’t erase tried positions for the column
            return to top of the loop with continue
    }
    Else if (curcol== -1) {
        No more solutions (all solutions found)
        We have backtracked all the way off the left end of the board
        End loop with break and end the search
    }
    Else {
        Find suitable position for queen in curcol
        int pos = getNextAvail(curcol);
        this function will check the nextAvail row to get the next available row
        it will also check it to confirm it has no conflicts with queen positions
            in previous rows (rows 0 through curcol-1).
        It returns the special value NoPosAvail if there are no untried conflict free
            positions for the queen in the current column.

        If (pos == NoPosAvail) {
            Backtracking Required
            Set queen position to unassigned for curcol
            Reset next available position to 0 for curcol
            set curcol to curcol – 1, return to top of loop with continue
        }
        Else {
            Normal Forward Progress, Position Found for curcol, advance to next col
            Pos is acceptable as position of queen for this column
            Set queenPos[curcol] to pos
            Set curcol = curcol + 1
            Return to top of loop with continue
        }
    }
}

Here on end of loop, to end the program
```

Detecting Conflicts/No Conflicts Between Columns
Theoretically, any new queen position must be checked for conflicts against all existing queen positions. There are three kinds of conflict to check for:

- Column conflict: two queens in the same column
- Row conflict: two queens in the same row
- Diagonal conflict: two queens in the same diagonal

Columns: we will organize the search for a solution by sweeping back and forth across the columns, allowing only one queen position per column. So the way we are searching for a solution will automatically eliminate the need to check for column conflict.

Rows: if row positions for two different columns are equal, then this is a row conflict.

Diagonal: Two queens are on the same diagonal if the absolute value of the row difference equals the absolute value of the column difference. Imagine we have two columns col_a and col_b, and two positions pos_a and pos_b. Furthermore assume that col_a < col_b. We need to calculate the difference in column values and the difference in position values.

- col_diff = col_b – col_a
- pos_diff = pos_b – pos_a

The col_diff value is > 0, but pos_diff could be < 0, = 0, or > 0, so final definition of diag conflicts

- Diag conflict: abs(pos_diff) == abs(col_diff)

If there is no row conflict and no diagonal conflict for the two column positions, then the two positions are considered to have no conflict. But before a new queen position is marked conflict free, it must be confirmed conflict free with all previous columns.

For example, look at the following 8-by-8 board with some sample queen positions marked by letters a-f. Each position has both a column and row coordinate that defines that position. “a” is at row 0 col 0, “b” is at row 4 col 1, etc.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	a							
1					e			
2			c					
3								f
4		b						
5								
6								
7				d				

Here are some example pairs of positions and how to check for diagonal conflict:

- a-b: col diff = 1-0 = 1, row diff = 4-0 = 4: **no diagonal conflict**
- a-c: col diff = 2-0 = 2, row diff = 2-0 = 2: **diagonal conflict** because 2 == 2
- b-d: col diff = 3-1 = 2, row diff = 7-4 = 3: **no diagonal conflict**
- b-e: col diff = 4-1 = 3, row diff = 4-1 =3: **diagonal conflict** because 3 == 3
- d-f: col diff = 7-3 = 4, row diff = 7-3 = 4: **diagonal conflict** because 4 == 4
- c-e: col diff 4-2 = 2, row diff = 2-1 = 1: **no diagonal conflict**

Here’s what the check for conflict might look like in your code. Consider writing a function called “nc(a,b)” to confirm “no conflict” for columns a and b. A return value of true means “no conflicts.”

```
public boolean nc(int a, int b) {
        int qpa = queenPos[a];   // look up position of queen for column a
        int qpb = queenPos[b];   // look up position of queen for column b

        // now check for no row conflict and no diagonal conflict
        // if no conflict return true
        // else return false
}
```

Determining Next Available Position for a Column
Keep track of the next available row to try for each column

```
int[] nextAvail;
```

The dimension of the array will be set to 8 for the 8-queens problem, or to N if you want to generalize to the N-queens version. Each index 0 through 7 or N-1 represents a column, and the value of the array at that index indicates the next row that is available to try for that column. All values are initialized to 0. If all values 0 through 7 or 0 through N-1 are tried and none work, use a special value NoPosAvail to indicate all row values for that column have been tried, and no more are available. When this value is found during the search, it triggers backtracking. The entry for the column is reset to 0 and current column is decremented to return to the previous column. A new position must be tried for the previous column and the search resumes at that point. Note that backtracking sometimes requires backtracking by more than one column.

Systematically Checking All Column Pairs For Conflict
Solutions are found whenever you find a set of queen positions for all columns and all possible column pairs for the whole board are free of conflict.

There are several ways to proceed. We could just randomly position all 8 queens then check all column pairs for conflicts. But this would be inefficient and bookkeeping might be difficult since we would need to make a record of the board state to ensure we didn't keep repeatedly trying it again later in the same search.

A more efficient search would be to do what is described in the main loop above. We start with an empty board (no queens positioned). We move to column 0 (curcol = 0) and position the queen at row 0. Actually any row would work for the 1st queen since there are not yet any other queens to conflict with, but bookkeeping is easier if we always try the next available position for a column. Then we move to column 1 (curcol = 1) and find a position for the column 1 queen that does not conflict with column 0. We continue to move to the right across the columns of the board. When we position the queen in column n, we check for no conflicts for all columns to the left of n:

(0,n), (1,n), (2,n) … (n-1,n).

If any one of these pairs has a conflict then the proposed position for curcol is rejected and another position must be selected.

During the process of positioning all the queens, we will frequently run into a situation where there will be **no** remaining position to try that is free of conflict. At that point, we are forced to **backtrack** by erasing the state for curcol, then decrementing curcol to move backward to the previous column. For the previous column, even though the queen position was selected earlier, we must pick a new position with no conflicts, then increment curcol and try to move forward again.

During backtracking, it can sometimes occur that multiple column must be backtracked at the same time. In other words, suppose we are working on curcol 6 and discover we must backtrack to 5. But on checking 5, we must immediately backtrack to 4, and so on. This is a normal part of the search. The algorithm described above for the main loop will handle cases like this without any additional logic.

Displaying Progress/Solutions
The following is a detailed trace of the search for the 1st solution in the 8-by-8 case. It is very long and it is not necessary to read it in detail. But by skim reading it, you can see how much forward and backtrack work is required just to find one solution. Note that at multiple points, forward progress reaches the last column before having to backtrack all the way back to the first column. So it's common to make progress but then discard a large amount of work to look for a better set of positions. For any two snapshots, an increase in the curcol value indicates forward progress, a decrease in the curcol value indicates backtrack.

It’s not required, but as part of your own program you might find it helpful to define a snapshot method that prints out an ASCII drawing of the board showing current column and current queen positions. Call the snapshot from some place inside the main loop when you want to see the update. A caution that as you can see from the trace below that the search will go through many states before locating a solution, and the snapshots of the run can generate a lot of output.

Trace of all forward and backtrack steps for solution #1 (very long, just skim)

```
curcol = 0
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:
  4:
  5:
  6:
  7:

curcol = 1
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:
  4:
  5:
  6:
  7:

curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:
  4:        Q
  5:
  6:
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:
  4:        Q
  5:
  6:
  7:
```

```
curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:              Q
  4:        Q
  5:
  6:
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:
  4:        Q
  5:
  6:
  7:              Q

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:
  4:        Q
  5:
  6:           Q
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:     Q
  3:
  4:        Q
  5:
  6:           Q
  7:

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:     Q
  3:                 Q
  4:        Q
  5:
  6:           Q
  7:
```

```
curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:     Q
  3:                 Q
  4:        Q
  5:                    Q
  6:           Q
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:              Q
  4:        Q
  5:
  6:           Q
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:
  4:        Q
  5:
  6:
  7:           Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:     Q
  3:
  4:        Q
  5:
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:     Q
  3:                 Q
  4:        Q
  5:
  6:
  7:           Q
```

```
curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:     Q
  3:                 Q
  4:        Q
  5:                    Q
  6:
  7:           Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:              Q
  4:        Q
  5:
  6:
  7:           Q

curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:
  4:
  5:        Q
  6:
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:
  4:
  5:        Q
  6:
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:
  4:
  5:        Q
  6:              Q
  7:
```

```
curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:
  4:                 Q
  5:        Q
  6:              Q
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:
  4:
  5:        Q
  6:
  7:           Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:     Q
  3:
  4:
  5:        Q
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:     Q
  3:                 Q
  4:
  5:        Q
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:     Q
  3:
  4:                 Q
  5:        Q
  6:
  7:           Q
```

```
curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:
  4:
  5:
  6:        Q
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:
  4:
  5:
  6:        Q
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:              Q
  4:
  5:
  6:        Q
  7:

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:              Q
  4:
  5:
  6:        Q
  7:                 Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:
  4:
  5:
  6:        Q
  7:              Q
```

```
curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:
  4:                 Q
  5:
  6:        Q
  7:              Q

curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:
  4:
  5:
  6:
  7:        Q

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:
  4:
  5:
  6:
  7:        Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:              Q
  4:
  5:
  6:
  7:        Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:     Q
  3:
  4:
  5:
  6:              Q
  7:        Q
```

```
curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:
  4:
  5:           Q
  6:
  7:        Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:     Q
  3:
  4:
  5:           Q
  6:
  7:        Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:     Q
  3:              Q
  4:
  5:           Q
  6:
  7:        Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:                 Q
  2:     Q
  3:              Q
  4:
  5:           Q
  6:
  7:        Q

curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:                 Q
  2:     Q
  3:              Q
  4:                    Q
  5:           Q
  6:
  7:        Q
```

```
curcol = 1
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:     Q
  4:
  5:
  6:
  7:

curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:     Q
  4:
  5:
  6:
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:     Q
  4:           Q
  5:
  6:
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:              Q
  3:     Q
  4:           Q
  5:
  6:
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:     Q
  4:           Q
  5:
  6:
  7:              Q
```

```
curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:     Q
  4:
  5:
  6:           Q
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:              Q
  3:     Q
  4:
  5:
  6:           Q
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:     Q
  4:
  5:
  6:
  7:           Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:              Q
  3:     Q
  4:
  5:
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:              Q
  3:     Q
  4:
  5:
  6:                 Q
  7:           Q
```

```
curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:     Q
  4:
  5:              Q
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:                 Q
  3:     Q
  4:
  5:              Q
  6:
  7:           Q

curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:     Q
  4:
  5:        Q
  6:
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:           Q
  3:     Q
  4:
  5:        Q
  6:
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:     Q
  4:
  5:        Q
  6:
  7:           Q
```

```
curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:
  3:     Q
  4:
  5:        Q
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:
  3:     Q
  4:                 Q
  5:        Q
  6:
  7:           Q

curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:                    Q
  3:     Q
  4:                 Q
  5:        Q
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:
  3:     Q
  4:
  5:        Q
  6:                 Q
  7:           Q

curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:                    Q
  3:     Q
  4:
  5:        Q
  6:                 Q
  7:           Q
```

```
curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:              Q
  3:     Q
  4:
  5:        Q
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:              Q
  3:     Q
  4:                 Q
  5:        Q
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:              Q
  3:     Q
  4:
  5:        Q
  6:                 Q
  7:           Q

curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:     Q
  4:
  5:
  6:        Q
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:           Q
  3:     Q
  4:
  5:
  6:        Q
  7:
```

```
curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:           Q
  3:     Q
  4:
  5:              Q
  6:        Q
  7:

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:                 Q
  2:           Q
  3:     Q
  4:
  5:              Q
  6:        Q
  7:

curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:                 Q
  2:           Q
  3:     Q
  4:                    Q
  5:              Q
  6:        Q
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:           Q
  3:     Q
  4:
  5:
  6:        Q
  7:              Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:                 Q
  2:           Q
  3:     Q
  4:
  5:
  6:        Q
  7:              Q
```

```
curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:                 Q
  2:           Q
  3:     Q
  4:                    Q
  5:
  6:        Q
  7:              Q

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:     Q
  4:           Q
  5:
  6:        Q
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:
  3:     Q
  4:           Q
  5:
  6:        Q
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:              Q
  3:     Q
  4:           Q
  5:
  6:        Q
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:     Q
  4:           Q
  5:
  6:        Q
  7:              Q
```

```
curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:                 Q
  2:
  3:     Q
  4:           Q
  5:
  6:        Q
  7:              Q

curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:     Q
  4:
  5:
  6:
  7:        Q

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:           Q
  3:     Q
  4:
  5:
  6:
  7:        Q

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:     Q
  4:           Q
  5:
  6:
  7:        Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:              Q
  2:
  3:     Q
  4:           Q
  5:
  6:
  7:        Q
```

```
curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:              Q
  3:     Q
  4:           Q
  5:
  6:
  7:        Q

curcol = 1
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:
  4:     Q
  5:
  6:
  7:

curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:
  4:     Q
  5:
  6:
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:
  4:     Q
  5:           Q
  6:
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:              Q
  3:
  4:     Q
  5:           Q
  6:
  7:
```

```
curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:              Q
  3:
  4:     Q
  5:           Q
  6:                 Q
  7:

curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:              Q
  3:                    Q
  4:     Q
  5:           Q
  6:                 Q
  7:

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:
  4:     Q
  5:
  6:
  7:           Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:              Q
  3:
  4:     Q
  5:
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:              Q
  3:
  4:     Q
  5:
  6:                 Q
  7:           Q
```

```
curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:              Q
  3:                    Q
  4:     Q
  5:
  6:                 Q
  7:           Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:
  4:     Q
  5:              Q
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:                 Q
  3:
  4:     Q
  5:              Q
  6:
  7:           Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:        Q
  2:
  3:                 Q
  4:     Q
  5:              Q
  6:
  7:           Q

curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:
  4:     Q
  5:
  6:        Q
  7:
```

```
curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:
  3:
  4:     Q
  5:
  6:        Q
  7:

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:
  3:              Q
  4:     Q
  5:
  6:        Q
  7:

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:
  3:              Q
  4:     Q
  5:
  6:        Q
  7:                 Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:
  3:
  4:     Q
  5:              Q
  6:        Q
  7:

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:                 Q
  3:
  4:     Q
  5:              Q
  6:        Q
  7:
```

```
curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:
  3:
  4:     Q
  5:              Q
  6:        Q
  7:                 Q

curcol = 2
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:
  4:     Q
  5:
  6:
  7:        Q

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:
  3:
  4:     Q
  5:
  6:
  7:        Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:
  3:              Q
  4:     Q
  5:
  6:
  7:        Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:
  3:              Q
  4:     Q
  5:
  6:                 Q
  7:        Q
```

```
curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:                    Q
  3:              Q
  4:     Q
  5:
  6:                 Q
  7:        Q

curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:
  3:
  4:     Q
  5:
  6:              Q
  7:        Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:                 Q
  3:
  4:     Q
  5:
  6:              Q
  7:        Q

curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:           Q
  2:                 Q
  3:
  4:     Q
  5:                    Q
  6:              Q
  7:        Q

curcol = 3
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:
  3:
  4:     Q
  5:           Q
  6:
  7:        Q
```

```
curcol = 4
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:              Q
  3:
  4:     Q
  5:           Q
  6:
  7:        Q

curcol = 5
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:
  2:              Q
  3:
  4:     Q
  5:           Q
  6:                 Q
  7:        Q

curcol = 6
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:                    Q
  2:              Q
  3:
  4:     Q
  5:           Q
  6:                 Q
  7:        Q

curcol = 7
Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:                    Q
  2:              Q
  3:                       Q
  4:     Q
  5:           Q
  6:                 Q
  7:        Q

Snapshot
      0  1  2  3  4  5  6  7
-----------------------------
  0:  Q
  1:                    Q
  2:              Q
  3:                       Q
  4:     Q
  5:           Q
  6:                 Q
  7:        Q
```

```
sol #1:  QP[(0:0)(1:4)(2:7)(3:5)(4:2)(5:6)(6:1)(7:3)
```