BACS2063 Data Structures and Algorithms

CHAPTER 6
APPLICATION OF RECURSION

(EXTRA READING)

8-QUEENS PUZZLE

Place 8 queens on a chessboard (8 x 8 square board) so that no 2 queens can attack each other. For any 2 queens to be non-attacking, they cannot be in the same row, same column, or same diagonals.

A SOLUTION TO THE 8-QUEENS PUZZLE

			Q				
					Q		
							Q
	Q						
						Q	
Q							
		Q					
				Q			

BACKTRACKING

A problem-solving and algorithm design technique.

The backtracking algorithm attempts to find solutions to a problem by constructing a partial solutions and making sure that the partial solution does not violate problem requirements.

BACKTRACKING (CONT'D)

The algorithm tries to extend a partial solution towards completion.

However, if it is determined that the partial solution would not lead to a solution, that is, the partial solution would end in a dead end, then the algorithm backs up by removing the most recently added part and trying other possibilities.

N-QUEENS PUZZLE: USING BACKTRACKING (1)

As each queen must be placed in a different row, the solution of the *n*-queens puzzle can be represented as an *n*-tuple (x_1, x_2, …, x_n), where x_i is an integer such that $1 \leq x_i \leq n$. In this tuple, x_i specifies the column number where to place the *i*th queen in the *i*th row.

Therefore, for the 8-queen puzzle the solution is an 8-tuple (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8), where x_i is the column where to place the *i*th queen in the *i*th row.

N-QUEENS PUZZLE: USING BACKTRACKING (2)

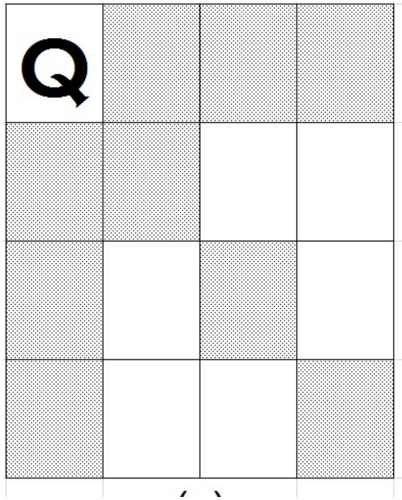
Therefore, the solution for the 8-queens puzzle shown in Slide 50 can be represented as the 8-tuple (4, 6, 8, 2, 7, 1, 3, 5). That is, the first queen is placed in the 1st row and 4th column, the second queen is placed in the 2nd row and 6th column, and so on. Clearly, each x_i is an integer such that $1 \leq x_i \leq 8$.

Note that the solution for this puzzle can in fact be applied to any number of queens.

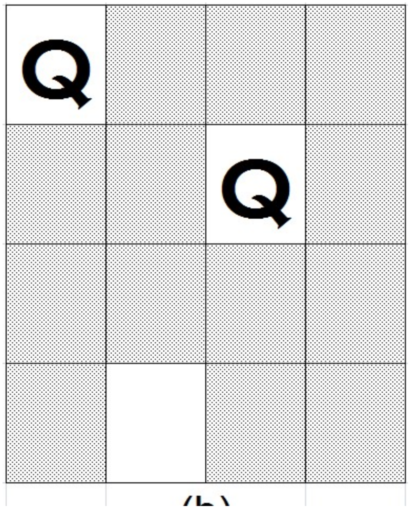
FINDING A SOLUTION TO THE 4-QUEENS PUZZLE (1)

a) We start by placing the first queen in the 1st row and 1st column.

(Note: The shaded cells means that no other queen can be placed in that cell).

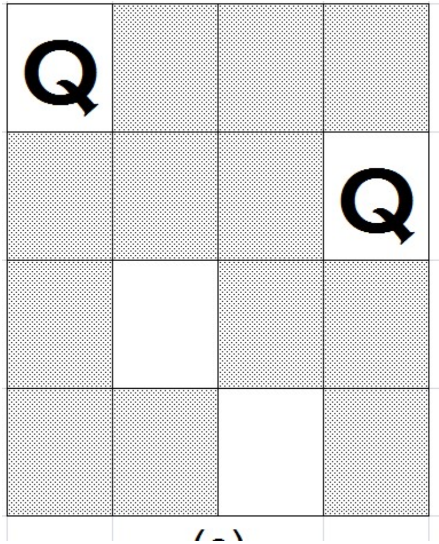


b) Next, we try to place the second queen in the 2nd row: the first cell in the 2nd row where the second queen can be placed is the 3rd column.

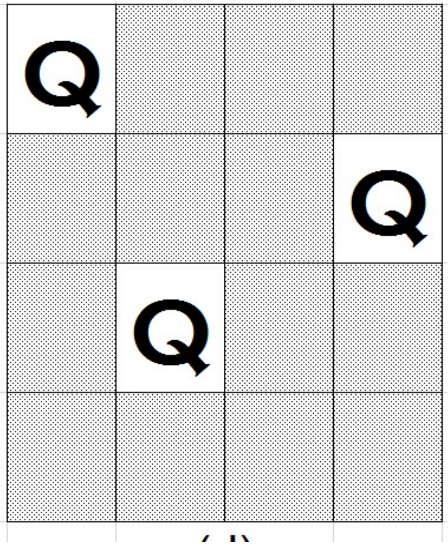


FINDING A SOLUTION TO THE 4-QUEENS PUZZLE (2)

c) Next, we try to place the third queen in the 3^{rd} row. We find that it cannot be placed in the 3^{rd} row and so we arrive at a dead end. At this point, we backtrack to the previous board configuration and place the second queen in the 4^{th} column.



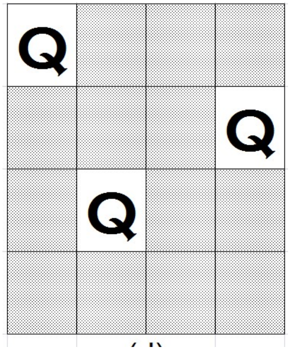
d) Next, we try to place the third queen in the 3^{rd} row. This time we successfully place the third queen in the 2^{nd} column of the 3^{rd} row.

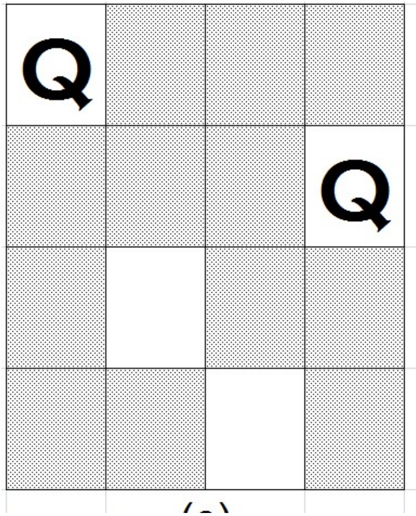


FINDING A SOLUTION TO THE 4-QUEENS PUZZLE (3)

e) When we try to place the fourth queen, we discover that the fourth queen cannot be placed in the 4th row.

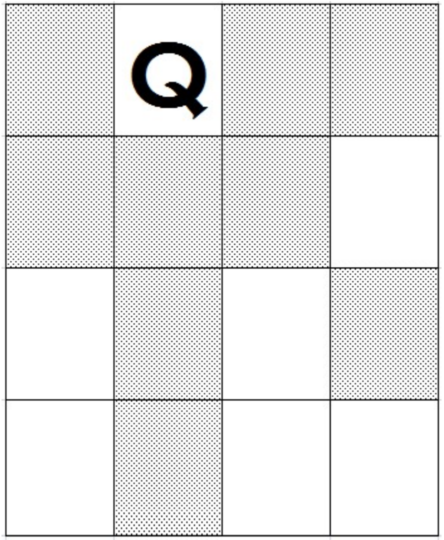
We backtrack to the 3rd row and try placing the third queen in any other column. Because no other column is available for queen 3, we backtrack to row 2 and try placing the queen in any other column, which cannot be done.



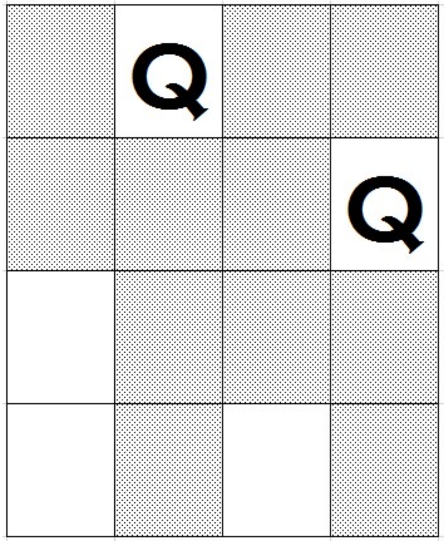


FINDING A SOLUTION TO THE 4-QUEENS PUZZLE (4)

f) Since we cannot place the second queen in another cell, we backtrack to the 1st row and place the first queen in the next column.

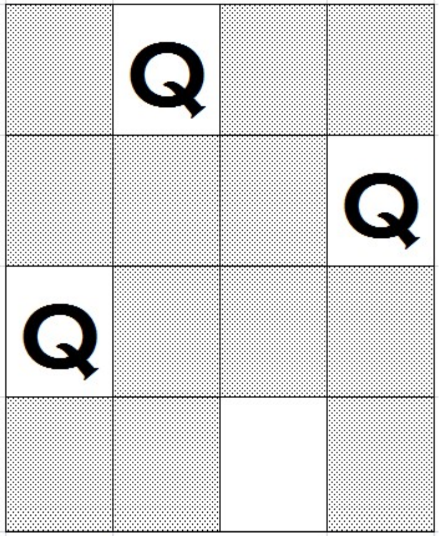


g) After placing the first queen in the 2nd column, we place second queen in the 2nd row's only available column, i.e. the 4th column.

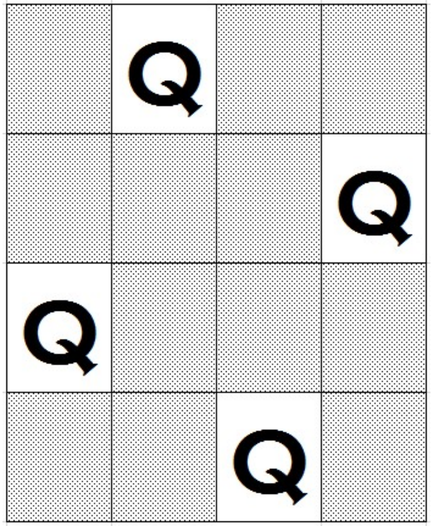


FINDING A SOLUTION TO THE 4-QUEENS PUZZLE (5)

h) Next, we place the third queen in the 3rd row's column 1.



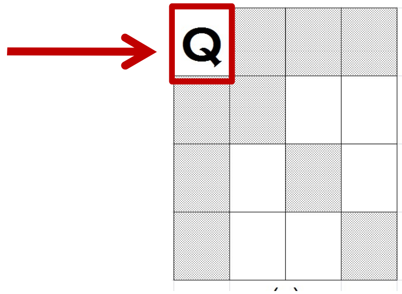
i) Finally, we place the fourth queen in the 4th row in column 3. Therefore, we have completed the solution to the puzzle.



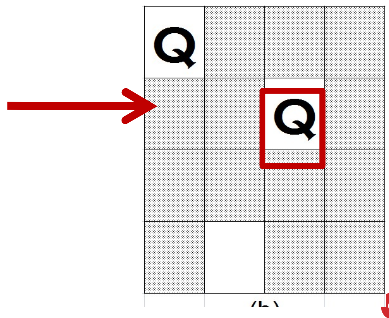
BACKTRACKING AND 4-QUEENS PUZZLE (1)

Suppose that the rows of the square board of the 4-queens puzzle are numbered 0 through 3 and the columns are numbered 0 through 3.

(Recall that in Java, array indexes starts at 0)



a) **We start by placing the first queen in the 1st column, thus generating tuple (0).**



b) **Then we place the second queen in the 3rd column and so generate the tuple (0, 2).**

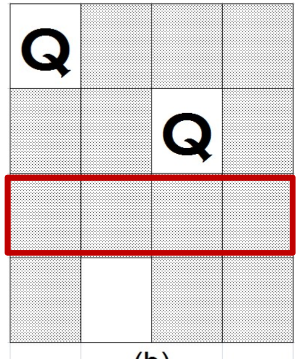
BACKTRACKING AND 4-QUEENS PUZZLE (2)

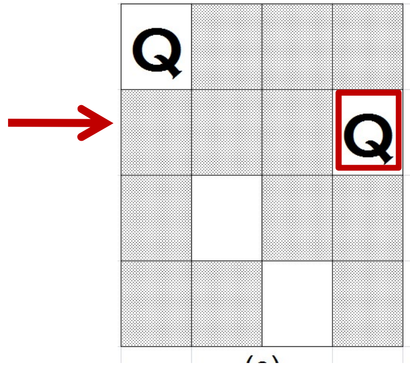
c) **When we try to place the third queen in the next row, we found that it cannot be done. Therefore, we**

   i. back up to the partial solution (0, 2),

   ii. remove 2 from the tuple, and then

   iii. generate the tuple (0, 3), i.e. the second queen is now placed in the 4th column of the second row.

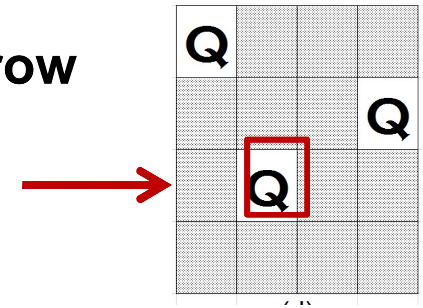




BACKTRACKING AND 4-QUEENS PUZZLE (3)

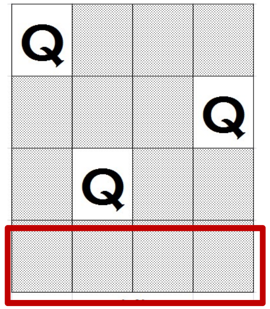
d) With the partial solution (0, 3),

next we try to place the third queen in the third row and generate the tuple (0, 3, 1).



e) Then with the partial solution

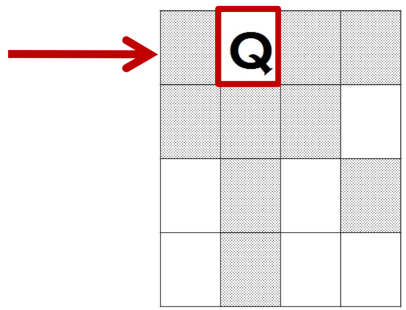
(0, 3, 1), when we try to place the fourth queen in the fourth row, we find that it cannot be done and so the partial solution (0, 3, 1) ends up in a dead end.



BACKTRACKING AND 4-QUEENS PUZZLE (4)

f) From the partial solution (0, 3, 1), the backtracking algorithm backs up to placing the first queen and so removes all the elements of the tuple.

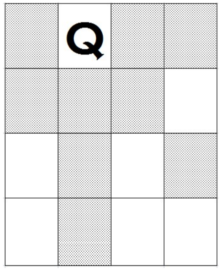
The algorithm then places the first queen in the second column of the first row, and thus generates the partial solution (1).



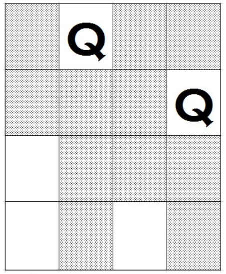
BACKTRACKING AND 4-QUEENS PUZZLE (5)

g) In this case, the sequence of partial solutions generated is:

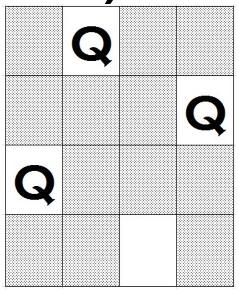
(1),



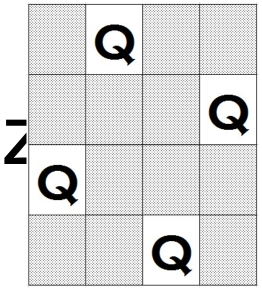
(1, 3),



(1, 3, 0)



and(1, 3, 0, 2), which represents the solution



to the 4-queens

puz

4-QUEENS SOLUTION TREE

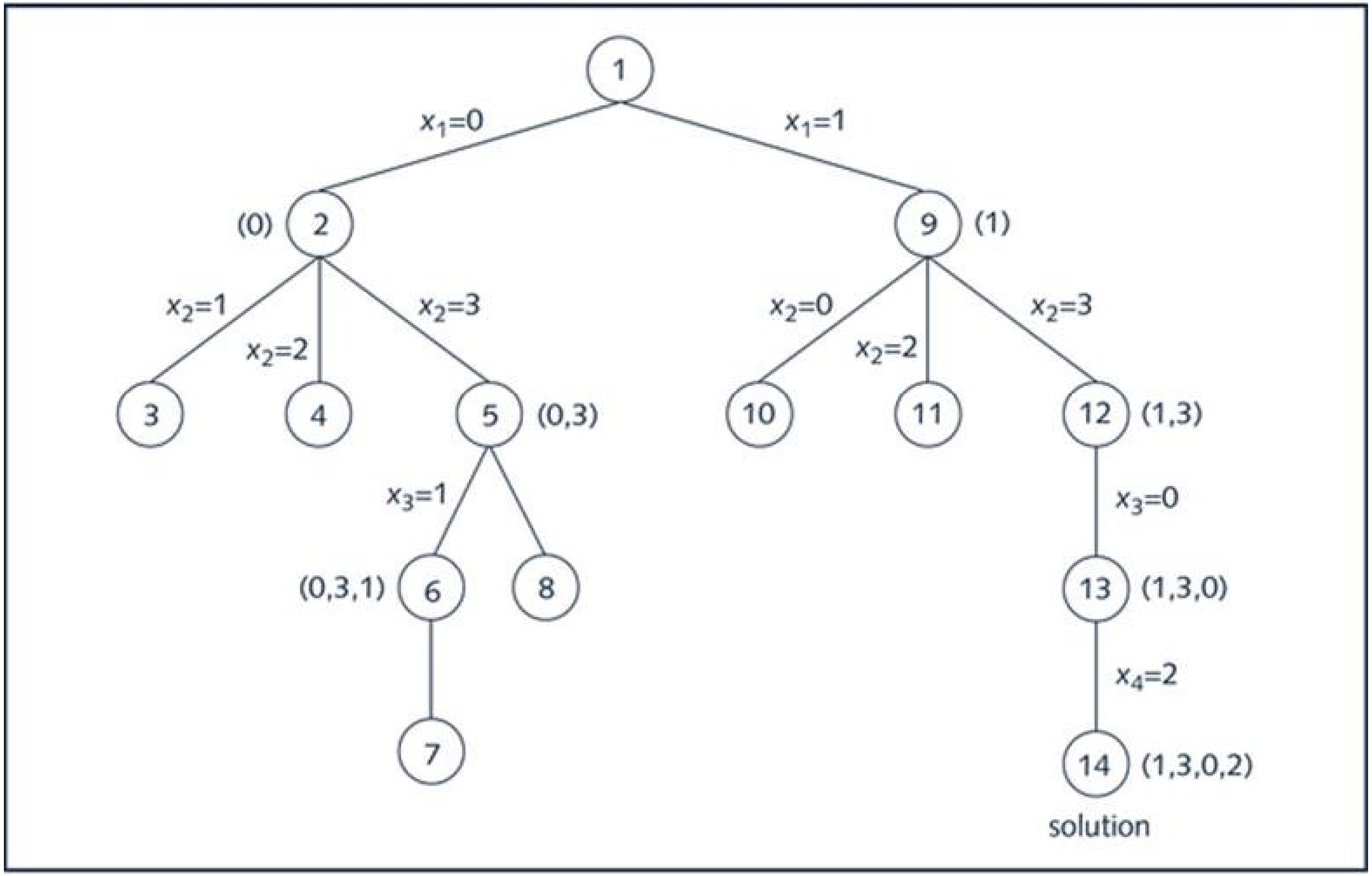


Figure 5-18 4-queens tree

8-QUEENS PUZZLE: IMPLEMENTATION (1)

Recall the requirements: no 2 queens can be in the same row, same column or same diagonal.

Determining whether 2 queens are in the same row or same column is easy because we can check their row and column positions.

How do we describe how to determine whether 2 queens are in the same diagonal?

Consider Figure 5-19 in the next slide: note that the rows and columns are both numbered 0 through 7.

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,0	6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,0	7,1	7,2	7,3	7,4	7,5	7,6	7,7

Figure 5-19 8 × 8 square board

8-QUEENS PUZZLE: IMPLEMENTATION (2)

Consider the diagonal from upper left to lower right (as indicated by the arrow). The positions of the cells on this diagonal are (0, 4), (1, 5), (2, 6), and (3, 7).

Notice that for these entries,

`rowPosition-columnPosition` is -4.

E.g., 0 – 4 = 1 – 5 = 2 – 6 = 3 – 7 = -4.

I.e., for each cell on a diagonal from upper left to lower right, `rowPosition-columnPosition` is the same.

8-QUEENS PUZZLE: IMPLEMENTATION (3)

Now, consider the diagonal from upper right to lower left (as indicated by the arrow). The positions of the cells on this diagonal are (0, 6), (1, 5), (2, 4), (3, 3), (4, 2), (5, 1), and (6, 0).

Here, `rowPosition+columnPosition` is 6.

I.e., **for each cell on a diagonal from upper right to lower left, `rowPosition+columnPosition` is the same.**

8-QUEENS PUZZLE: IMPLEMENTATION (4)

We use the preceding two results to determine if 2 queens are on the same diagonal or not:

Suppose that there is a queen at position (i, j) and another queen at position (k, l). These queens are on the same diagonal if either:

a) $i + j = k + l$ or

b) $i - j = k - l$

- a) implies that $j - l = k - i$ and b) implies that $j - l = i - k$

This means that 2 queens are on the same diagonal if $| j - l | = | i - k |$.

8-QUEENS PUZZLE: IMPLEMENTATION (5)

Because a solution to the 8-queens puzzle is represented as an 8-tuple, we use the array `queensInRow` of size 8, where `queensInRow[k]` specifies the column position of the *k*th queen in row `k`.

E.g., `queensInRow[0]` means that the first queen is placed in column 3 (i.e. the 4th column) of row 0 (which is the 1st row)