N-Queens Problem

- Place *n* queens on an *n* by *n* chess board so that no two of them are on the same row, column, or diagonal

Queen's Move

- A queen can move any number of squares horizontally, vertically, and diagonally.

- Here, the possible target squares of the queen Q are marked with an x.

x			x			x	
	x		x		x		
		x	x	x			
x	x	x	Q	x	x	x	x
		x	x	x			
	x		x		x		
x			x			x	
			x				x

4-Queens (1)

- Lets take a look at the simple problem of placing 4 queens on a 4x4 board
- The brute-force solution is to place the first queen, then the second, third, and forth
  - After all are placed we determine if they are placed legally
- There are 16 spots for the first queen, 15 for the second, etc.
  - 16*15*14*13 = 43,680 different combinations

4-Queens (2)

- Lets use the fact that no two queens can be in the same column
- So we can place the first queen into the first column, the second into the second, etc.
- Now there are 4 spots for the first queen, 4 spots for the second, etc.
  - 4*4*4*4 = 256 different combinations
- We can still do better because a new queen is not in the same row or diagonal as a previously placed queen

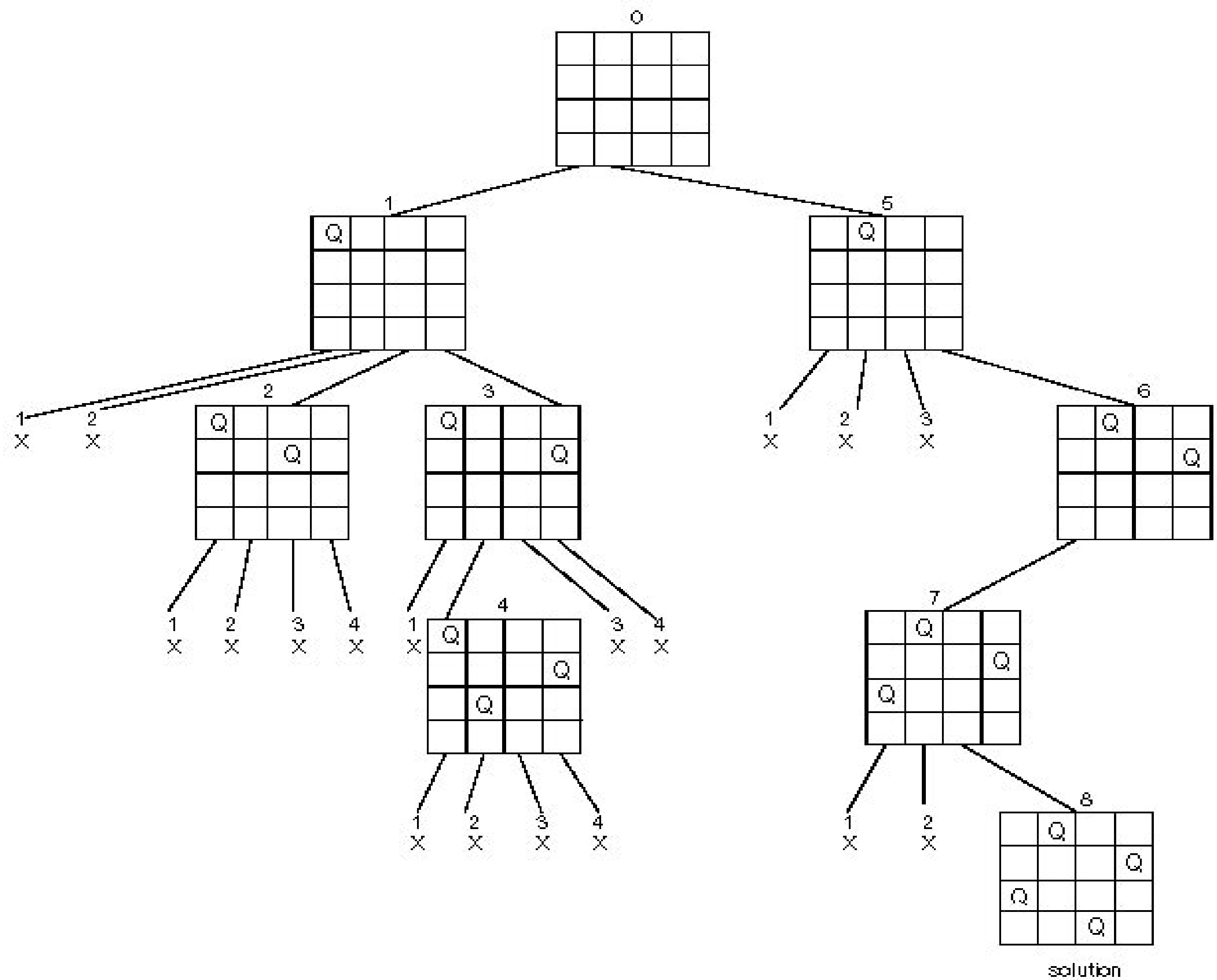
N-Queens: Decision

In any solution of the N-Queens problem, there must be **exactly one queen in each row/column**.

Therefore, we can describe the solution of this problem as a **sequence of n decisions**:

Decision 1: Place a queen in the first row/column.
Decision 2: Place a queen in the second row/column.
⋮
Decision n: Place a queen in the n-th row/column.

4-Queens: State Space Tree



4-Queens: Backtracking

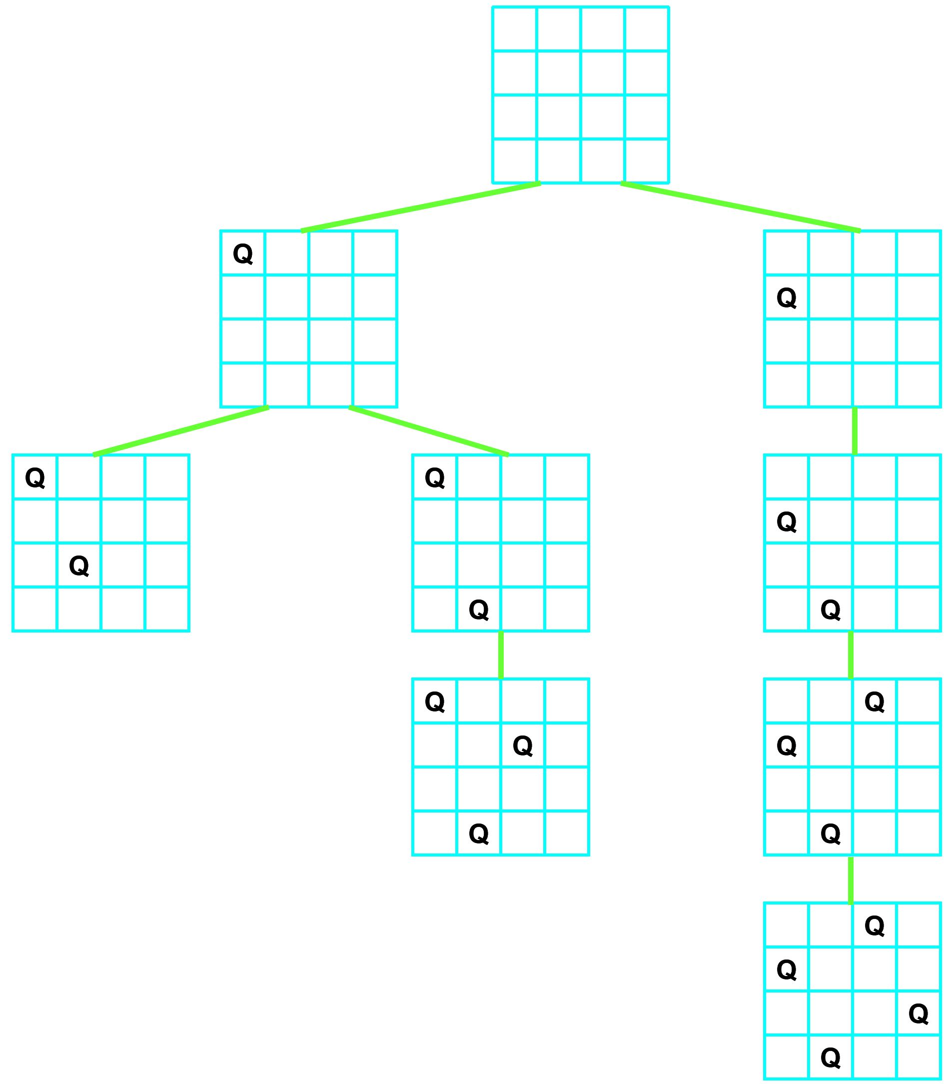
Empty board

Place 1st queen

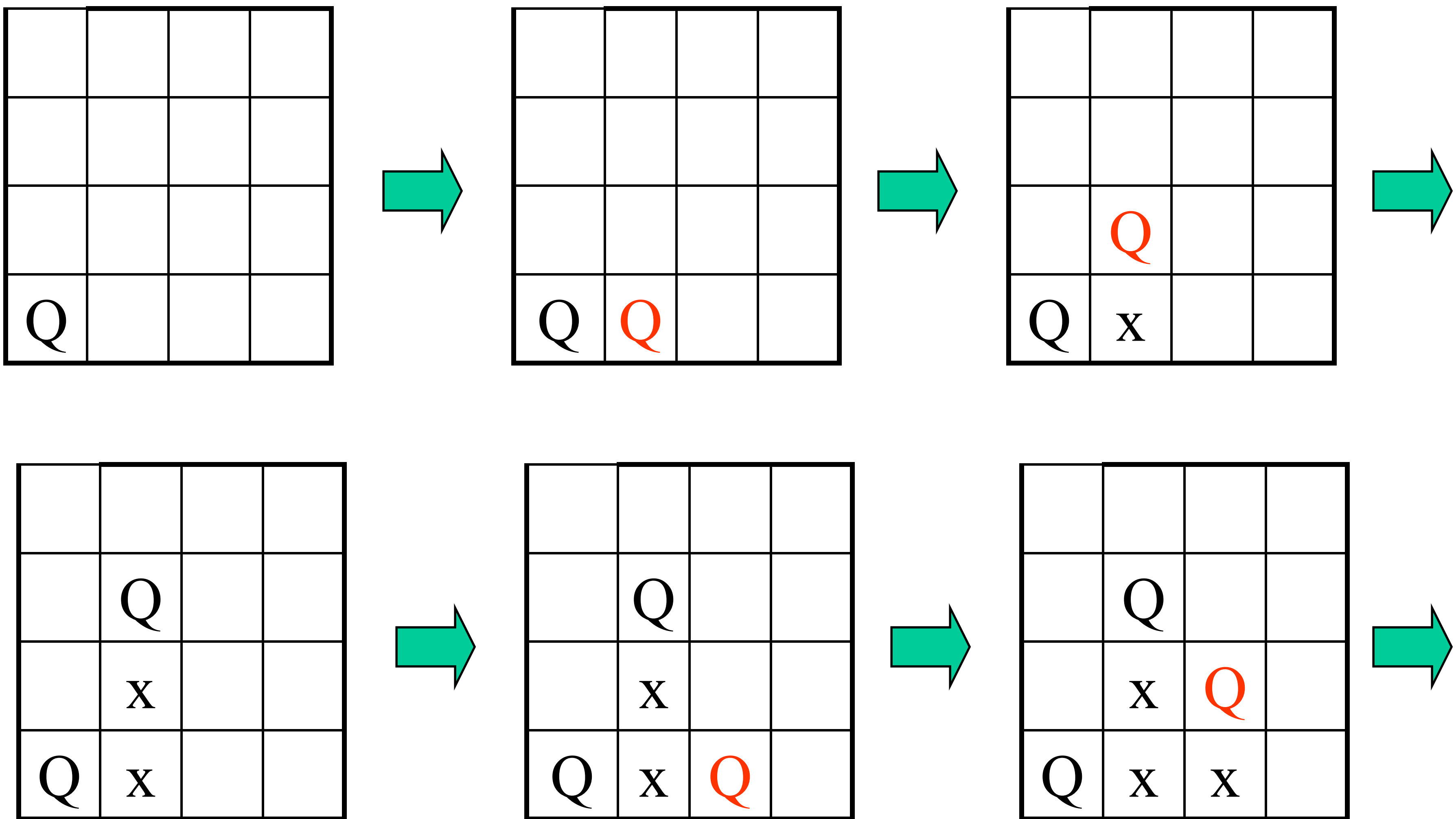
Place 2nd queen

Place 3rd queen

Place 4th queen



4-Queens: Simulation (1)



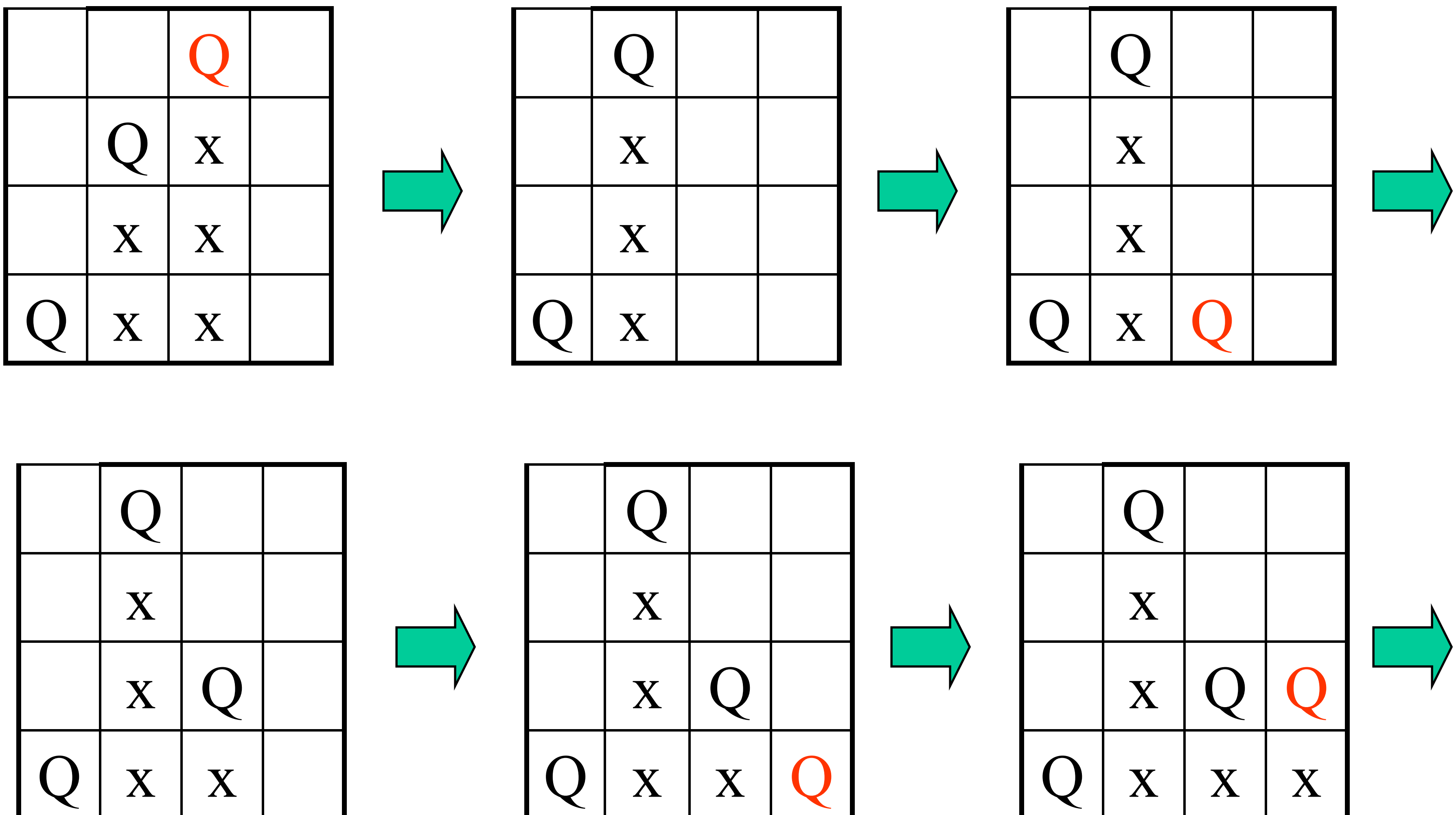
4-Queens: Simulation (2)

	Q	Q	
	x	x	
Q	x	x	

		Q	
	Q	x	
	x	x	
Q	x	x	

We are stuck!

4-Queens: Simulation (3)



4-Queens: Simulation (4)

	Q		
	x		Q
	x	Q	x
Q	x	x	x

	Q		Q
	x		x
	x	Q	x
Q	x	x	x

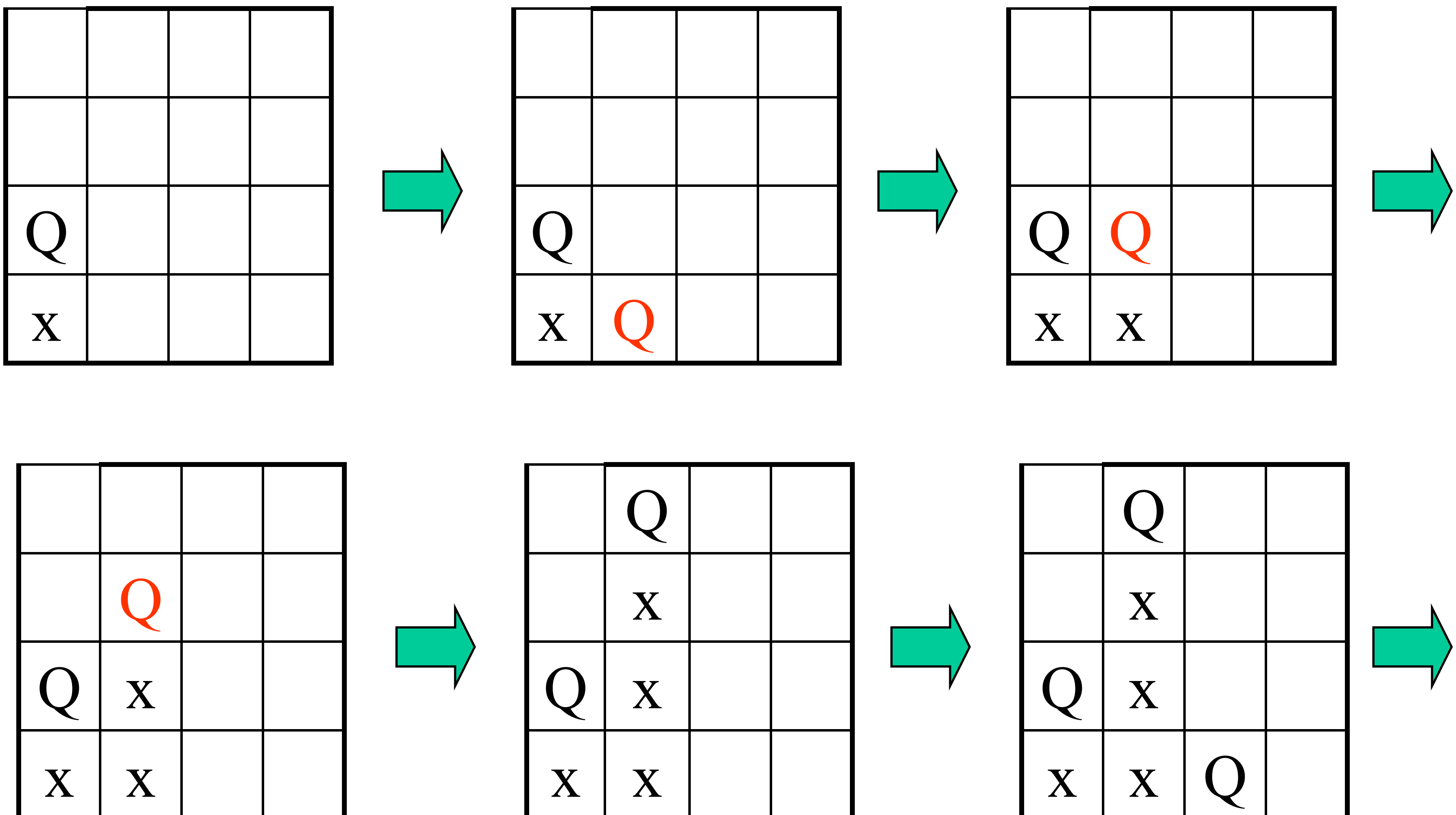
	Q		
	x	Q	
	x	x	
Q	x	x	

	Q	Q	
	x	x	
	x	x	
Q	x	x	

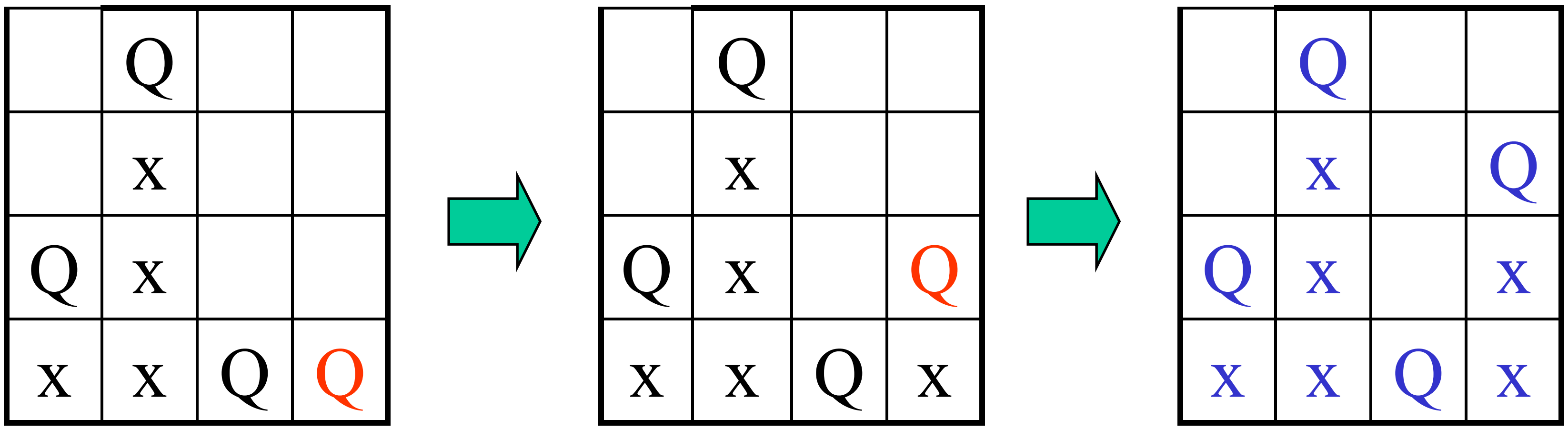
	Q		
	x		
	x		
Q	x		

Q			
x			

4-Queens: Simulation (5)



4-Queens: Simulation (6)



N-Queens: Diagonal Conditions

Two queens are placed at positions (i, j) and (k, l)

They are on the same diagonal only if

$i - j = k - l$ or $i + j = k + l$

The first equation implies

$J – l = i – k$

The second implies

$J – i = k – i$

Two queens lies on the same diagonal iff

$|j – l| = |i - k|$

N-Queens: Diagonal Conditions (Example)

For instance P1=(8, 1) and P2=(1, 8)

So $i = 8$, $j=1$ and $k=1$, $l=8$

$$i + j = k + l$$

$$8+1 = 1+8 = 9$$

$$J - l = k - i$$

$$1 - 8 = 1 - 8$$

Hence P1 and P2 are on the same diagonal

N-Queens: Algorithm (Sahni[2nd Ed.]-354)

```
Algorithm NQueens(k, n)
// Using backtracking, this procedure prints all
// possible placements of n queens on an n × n
// chessboard so that they are nonattacking.
{
    for i := 1 to n do
    {
        if Place(k, i) then
        {
            x[k] := i;
            if (k = n) then write (x[1 : n]);
            else NQueens(k + 1, n);
        }
    }
}
```

N-Queens: Algorithm (Place)

```
Algorithm Place(k, i)
// Returns true if a queen can be placed in kth row and
// ith column. Otherwise it returns false. x[ ] is a
// global array whose first (k − 1) values have been set.
// Abs(r) returns the absolute value of r.
{
    for j := 1 to k − 1 do
        if ((x[j] = i) // Two in the same column
             or (Abs(x[j] − i) = Abs(j − k)))
                // or in the same diagonal
            then return false;
    return true;
}
```

Thank You

Stay Safe