CS-218
DATA STRUCTURE

Dr. Hashim Yasin

National University of Computer and Emerging Sciences,

Faisalabad, Pakistan.

STACKS

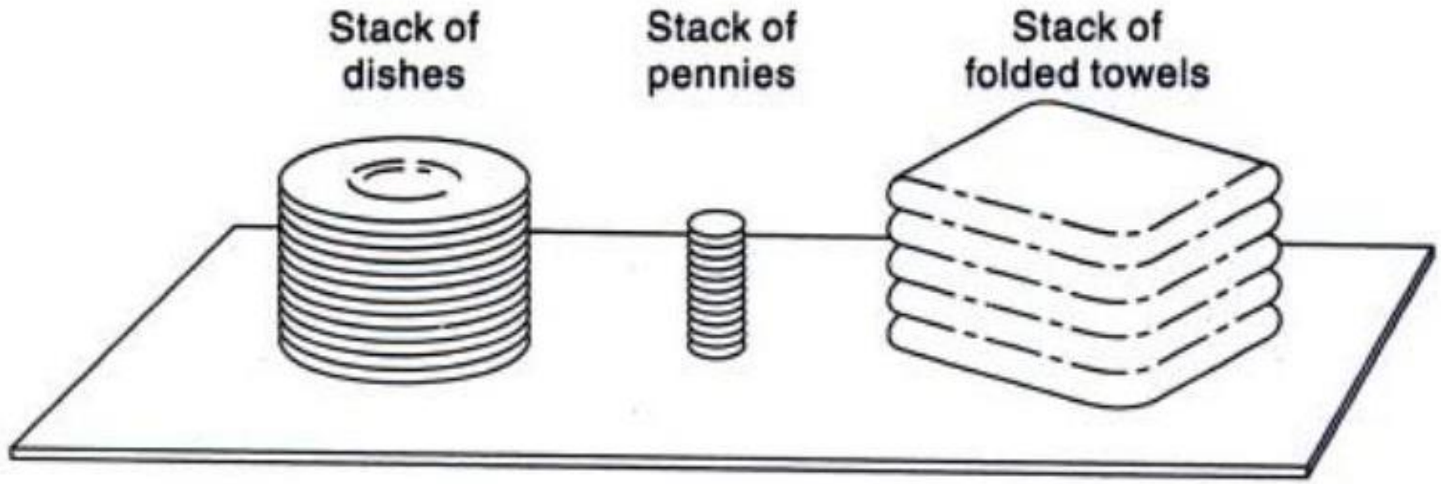
Stack

- "A ***Stack*** is a special kind of list in which all insertions and deletions take place at one end, called the ***Top***"
- Other Names,
  - Pushdown List
  - Last In First Out (LIFO)

Stack

Examples:

- Folded towels on shelf
- Dishes on a shelf
- Pennies on shelf



Common Operations

1. **MAKENULL(S):** Make Stack S be an empty stack.
2. **TOP(S):** Return the element at the top of stack S.
3. **POP(S):** Remove the top element of the stack.
4. **PUSH(S):** Insert the element x at the top of the stack.
5. **ISEMPTY(S):** Return true if S is an empty stack; return false otherwise.

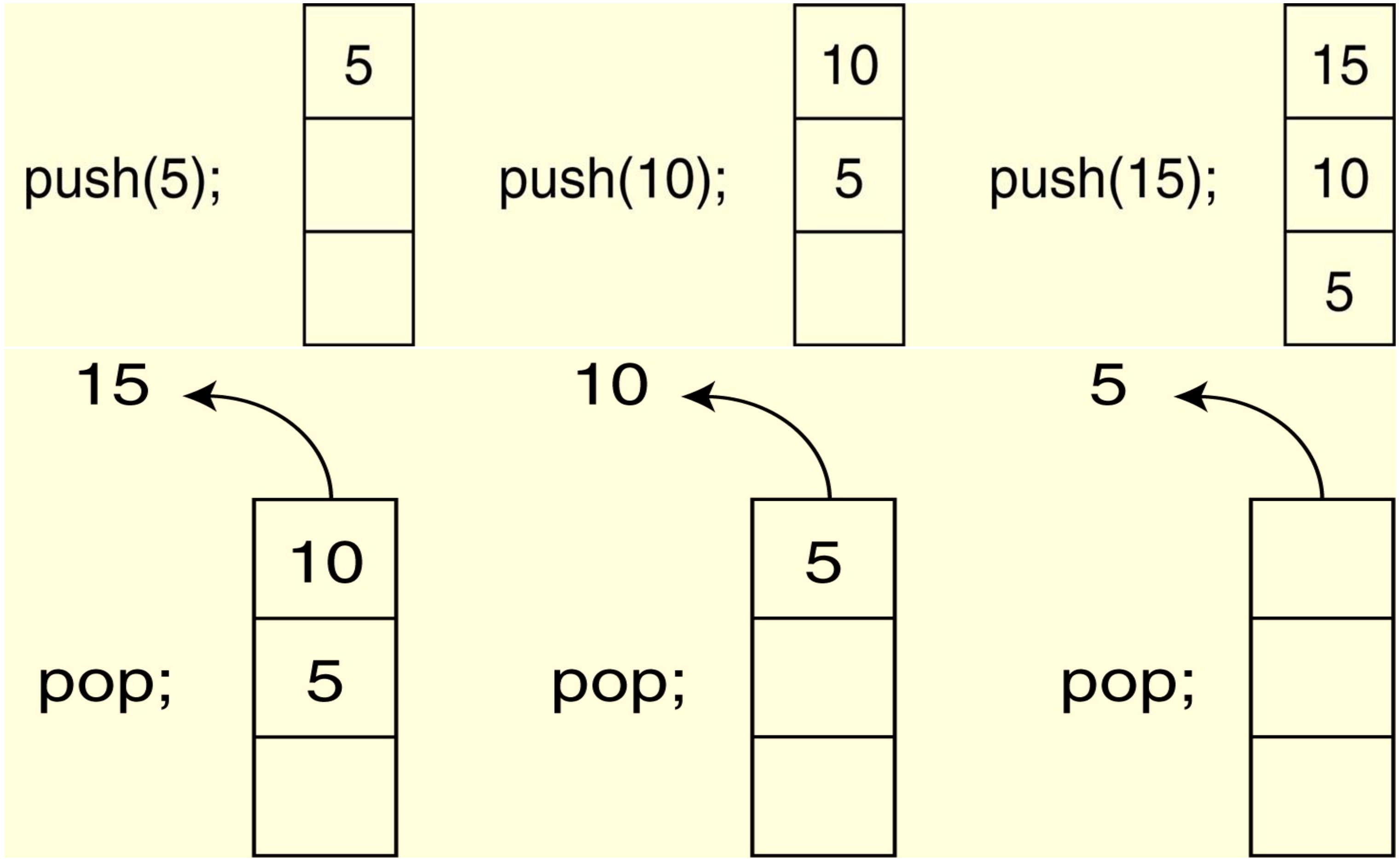
Static and Dynamic Stacks

- There are two kinds of stack data structure,

  a) **Static**, i.e., they have a **fixed size**, and are *implemented as* **arrays.**

  b) **Dynamic**, i.e., they **grow in size** as needed, and *implemented as* **linked lists**

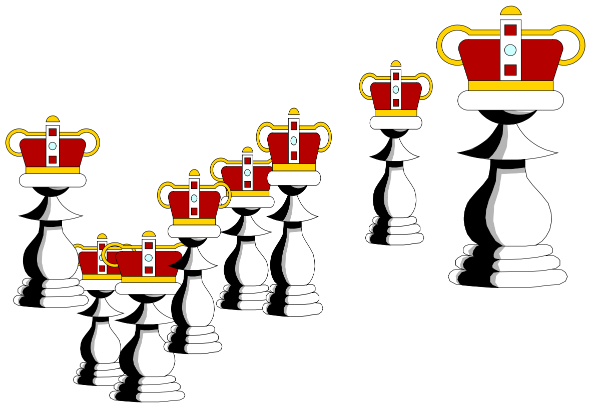
Common Operations

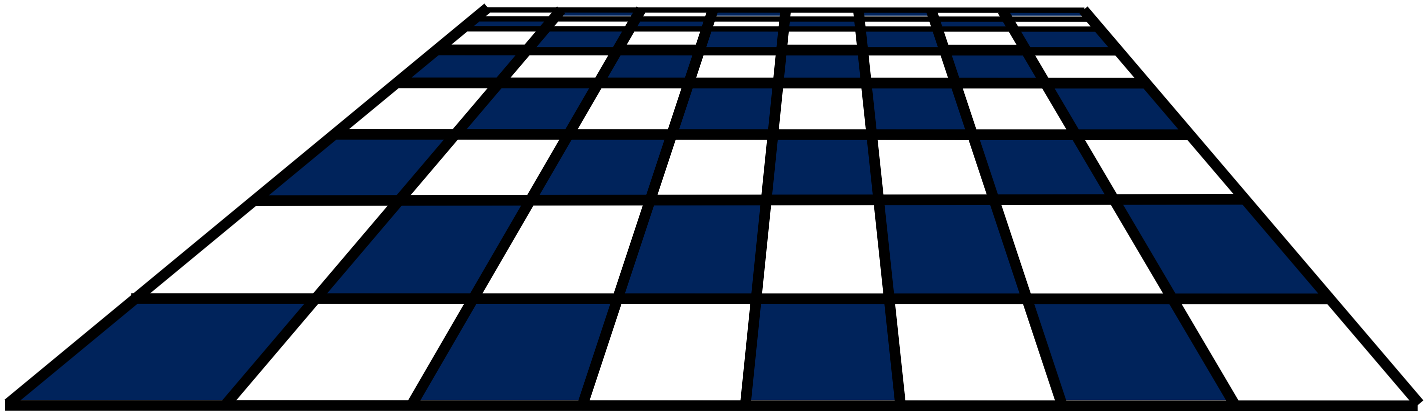


N-QUEEN PROBLEM

The N-Queens Problem

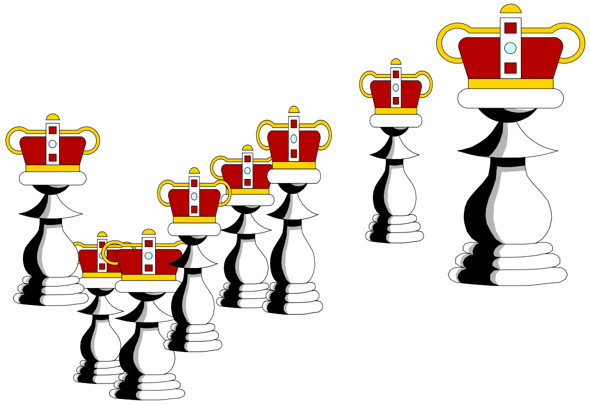
- Suppose you have 8 chess queens...
- ...and a chess board

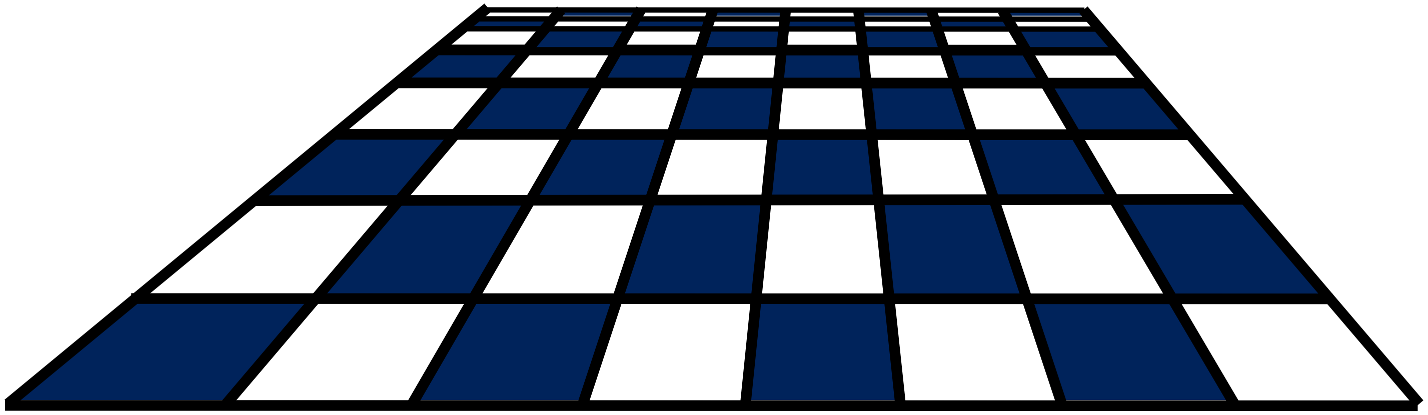




The N-Queens Problem

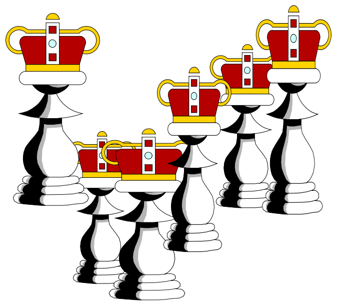
Can the queens be placed on the board so that NO two queens are attacking each other.

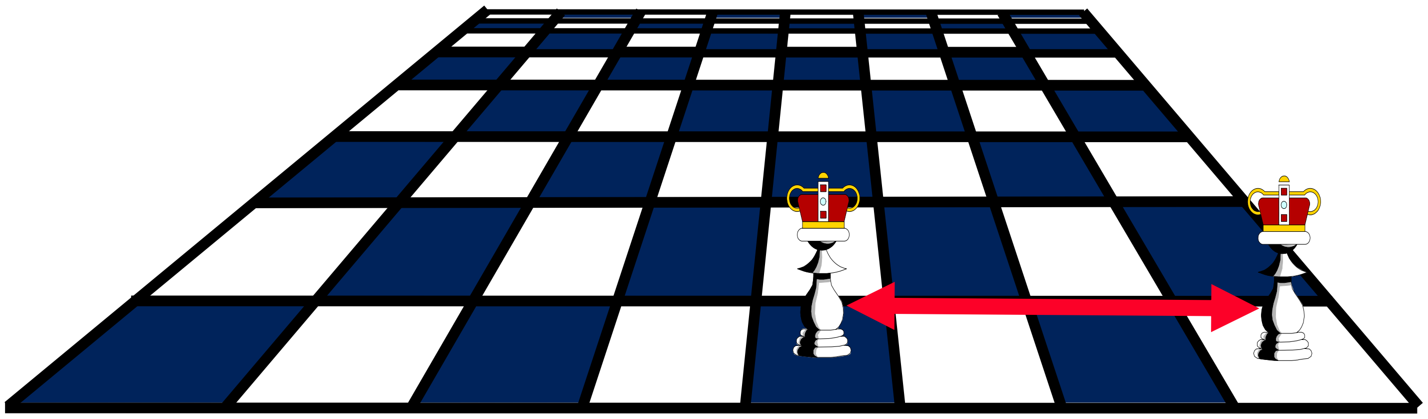




The N-Queens Problem

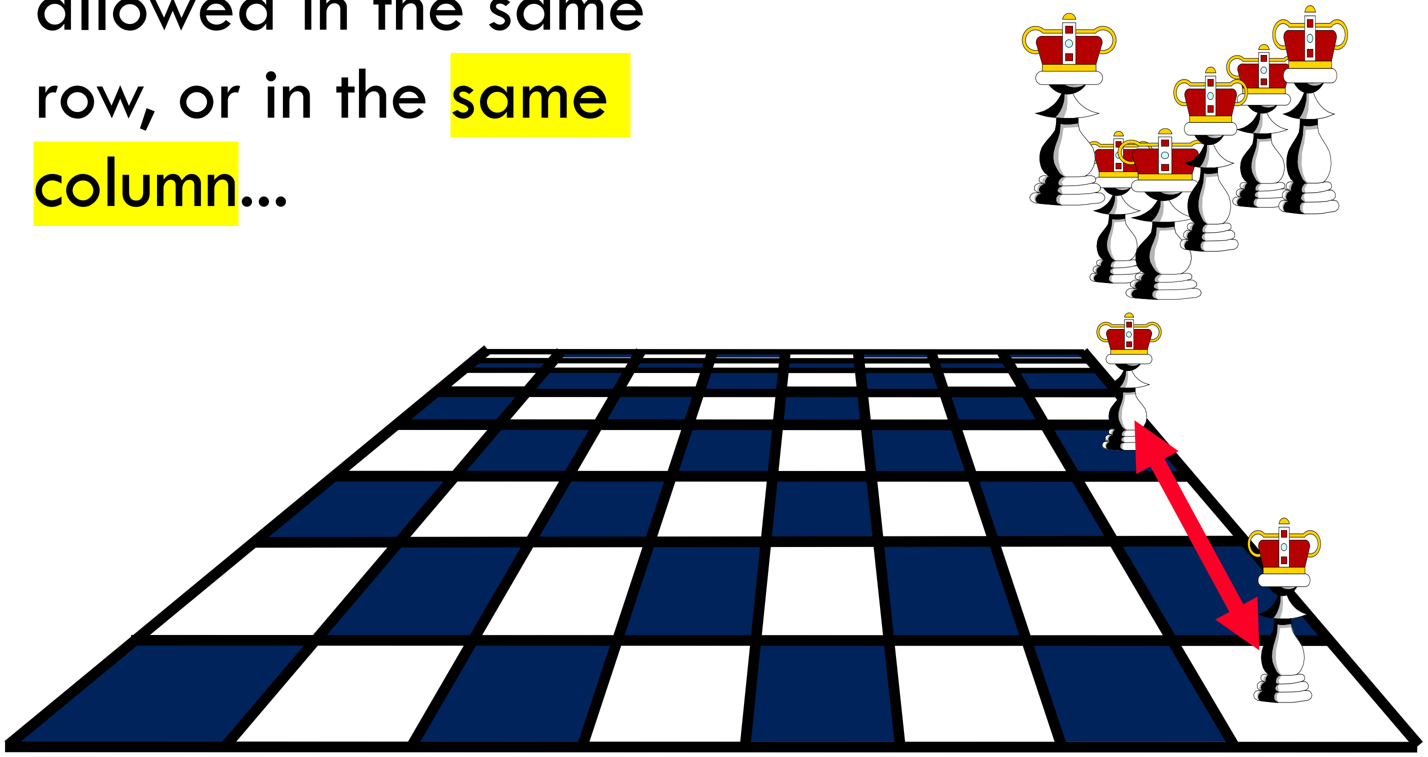
Two queens are not allowed in the same row...





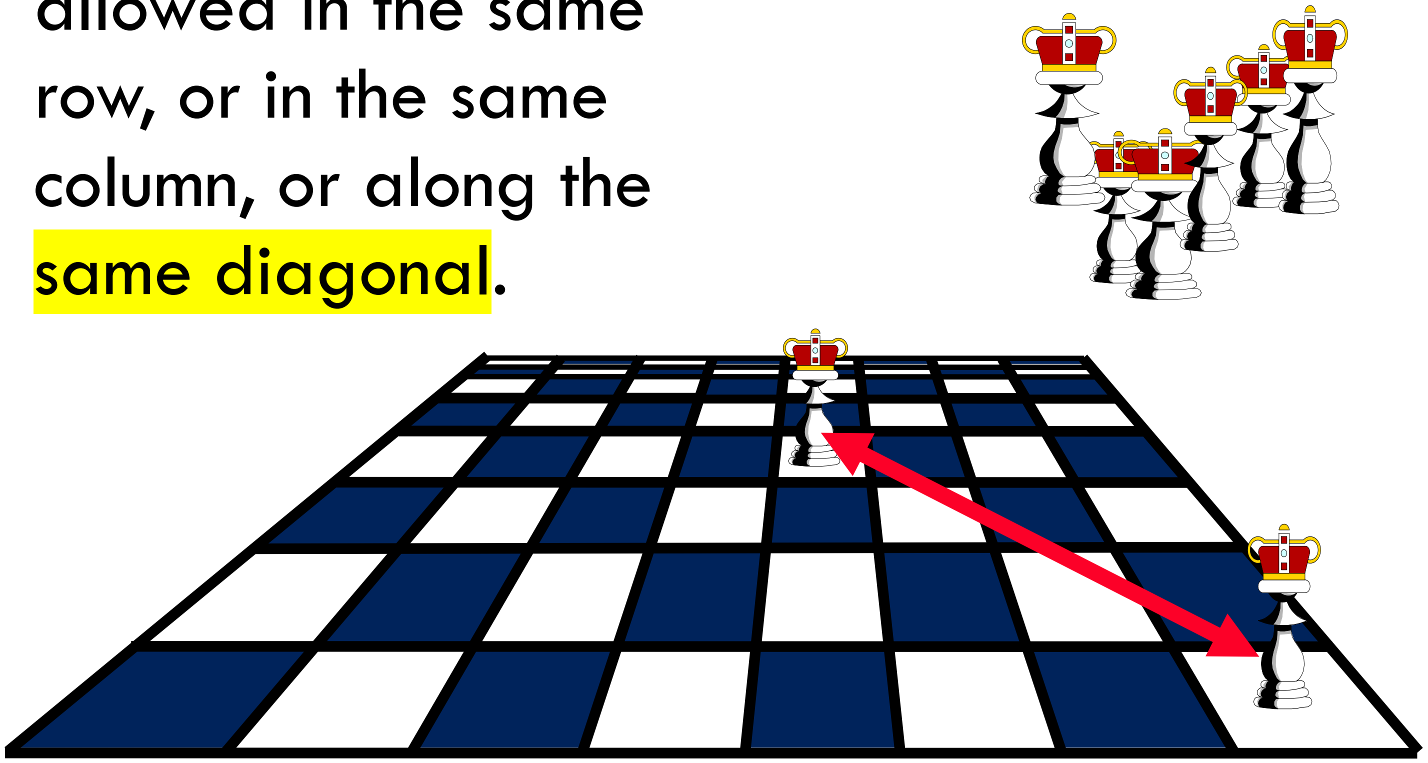
The N-Queens Problem

Two queens are not allowed in the same row, or in the same column...



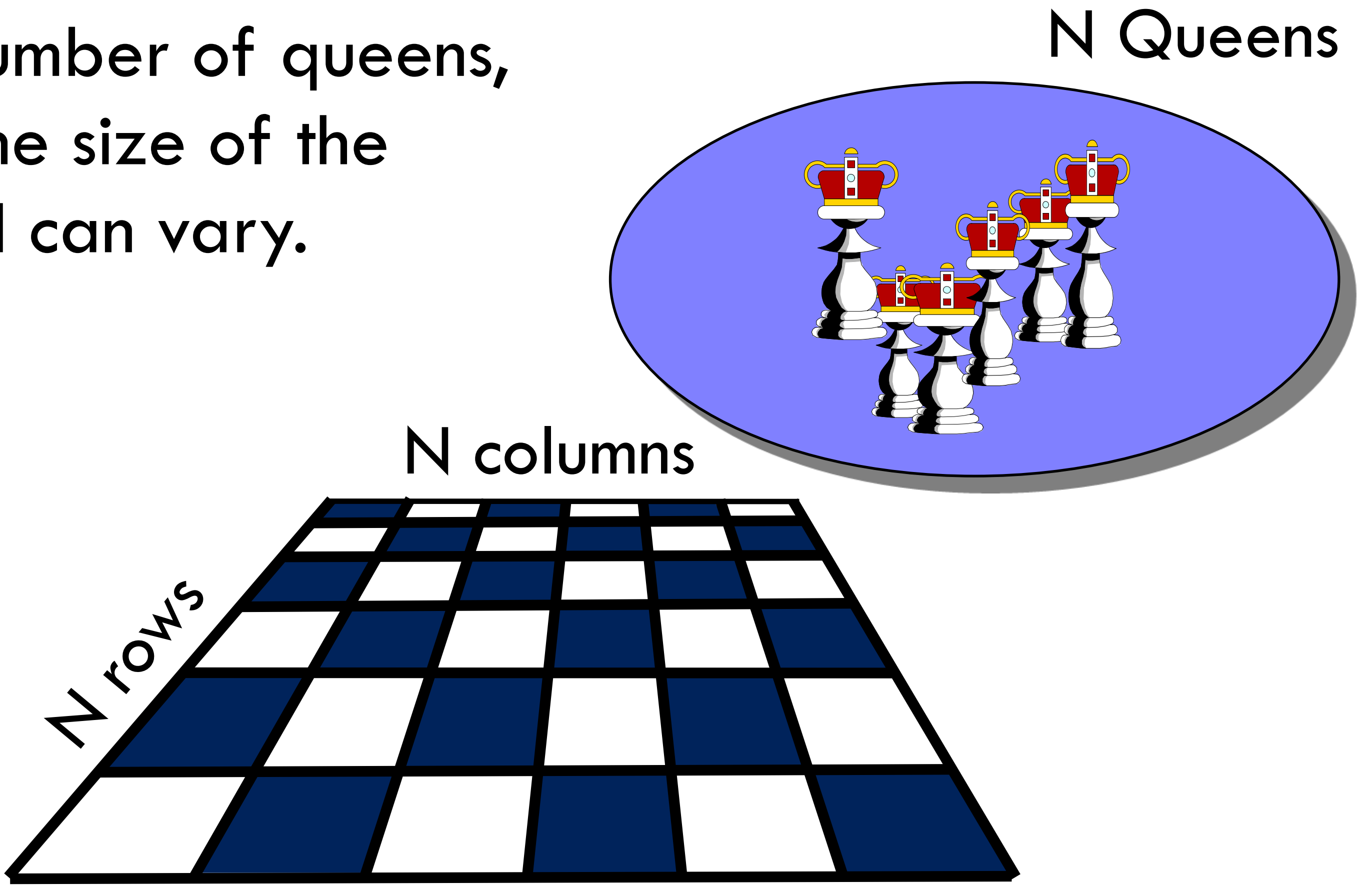
The N-Queens Problem

Two queens are not allowed in the same row, or in the same column, or along the same diagonal.



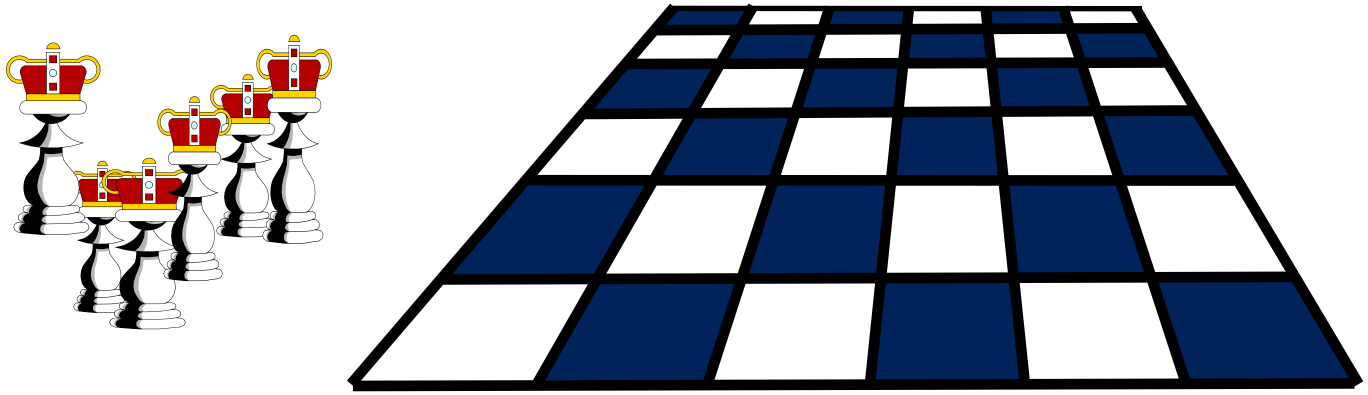
The N-Queens Problem

The number of queens, and the size of the board can vary.



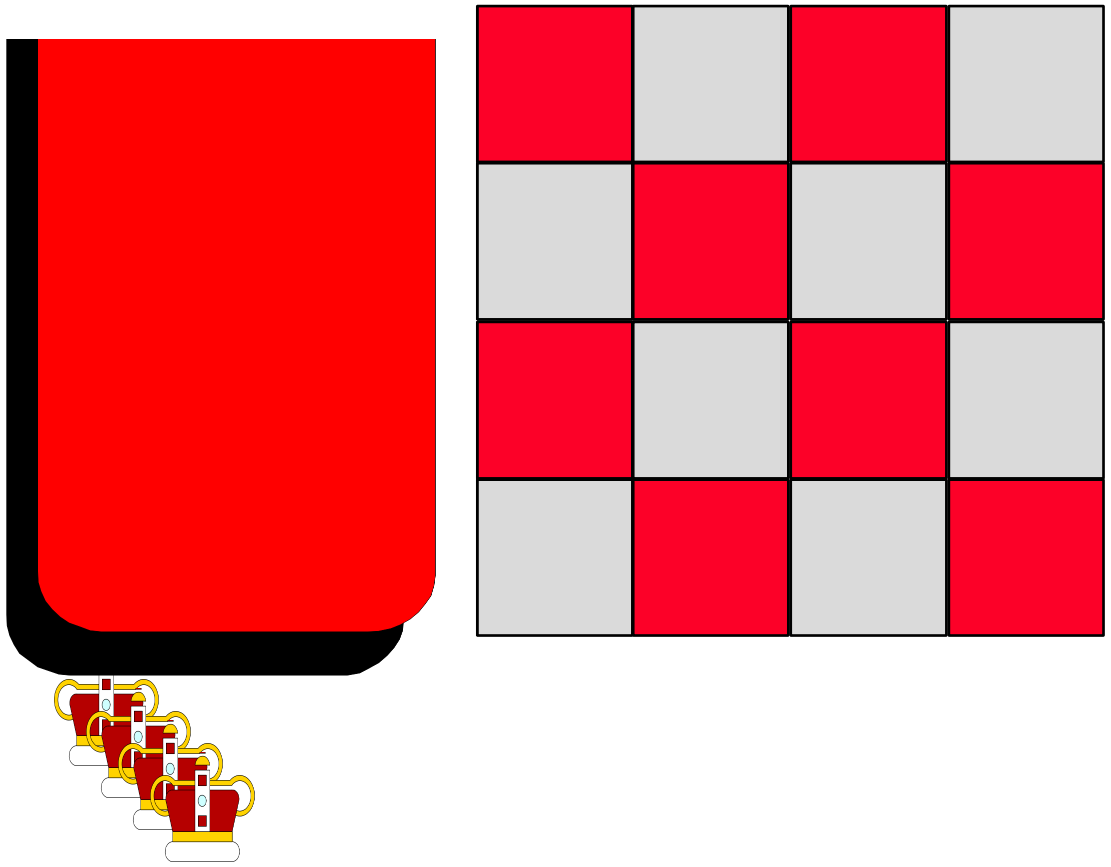
The N-Queens Problem

We will write a program which tries to find a way to place N queens on an N X N chess board.



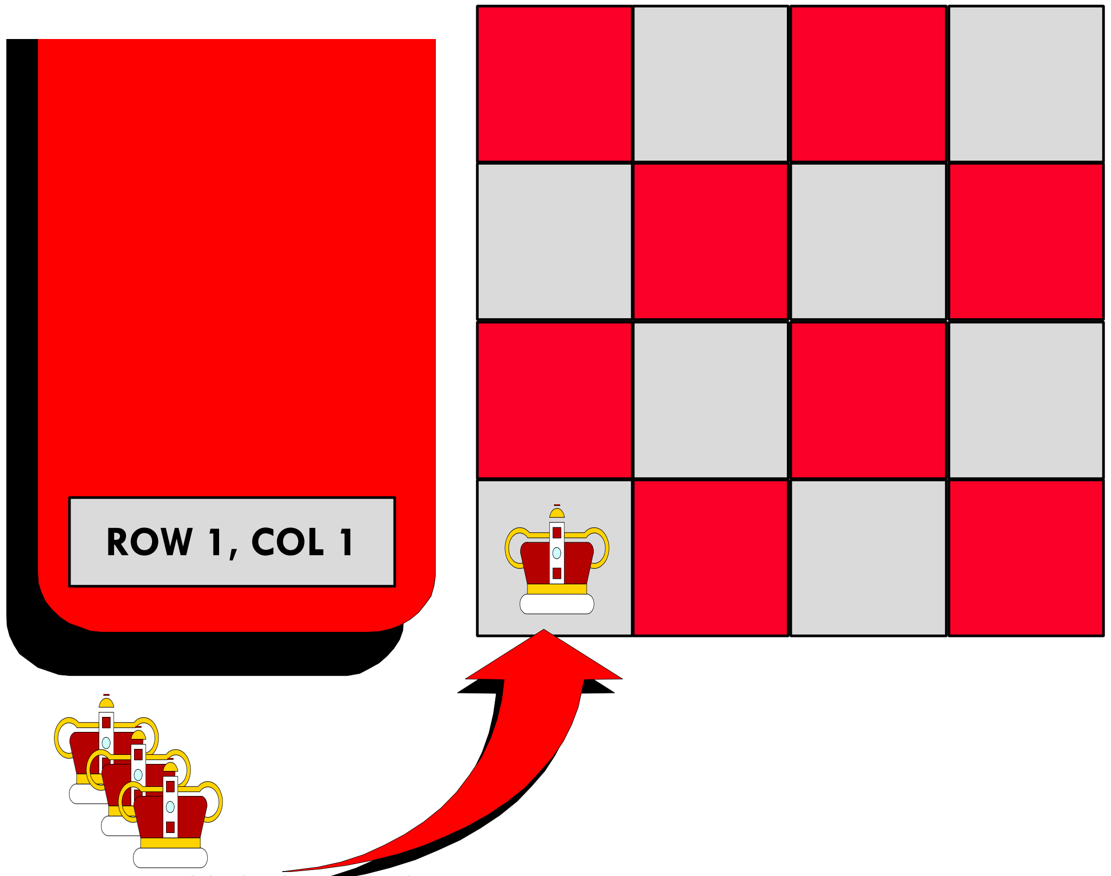
How the program works

The program uses a **stack** to keep track of where each queen is placed.



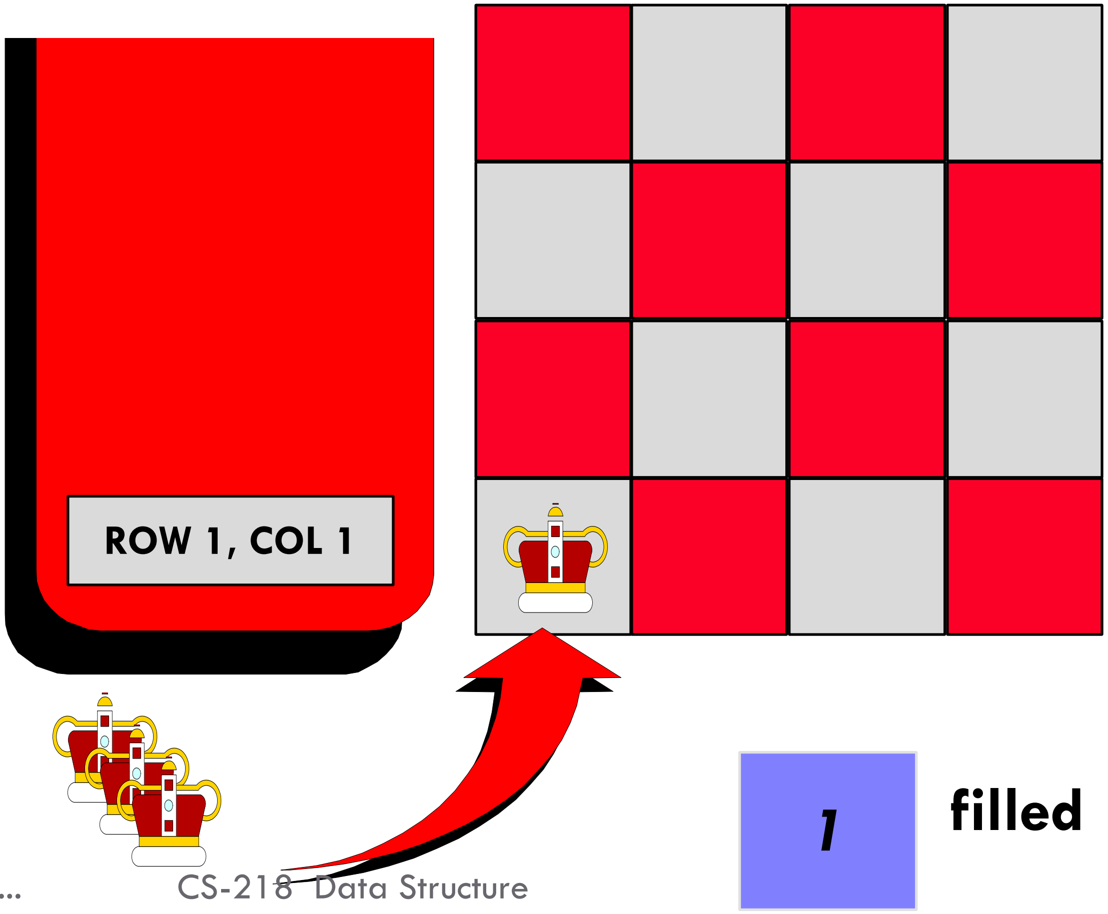
How the program works

- Each time the program decides to place a queen on the board,
- The position of the new queen is stored in a record which is placed in the stack.



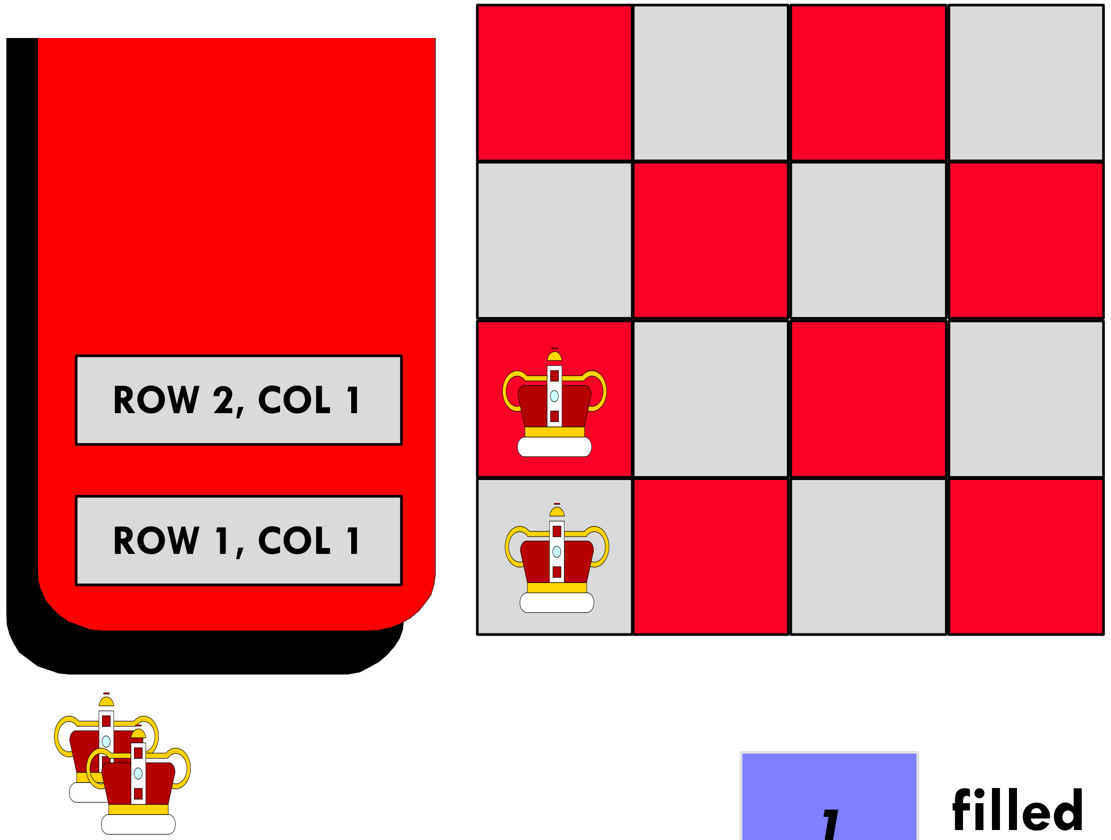
How the program works

We also have an **integer variable** to keep track of how many rows have been filled so far.



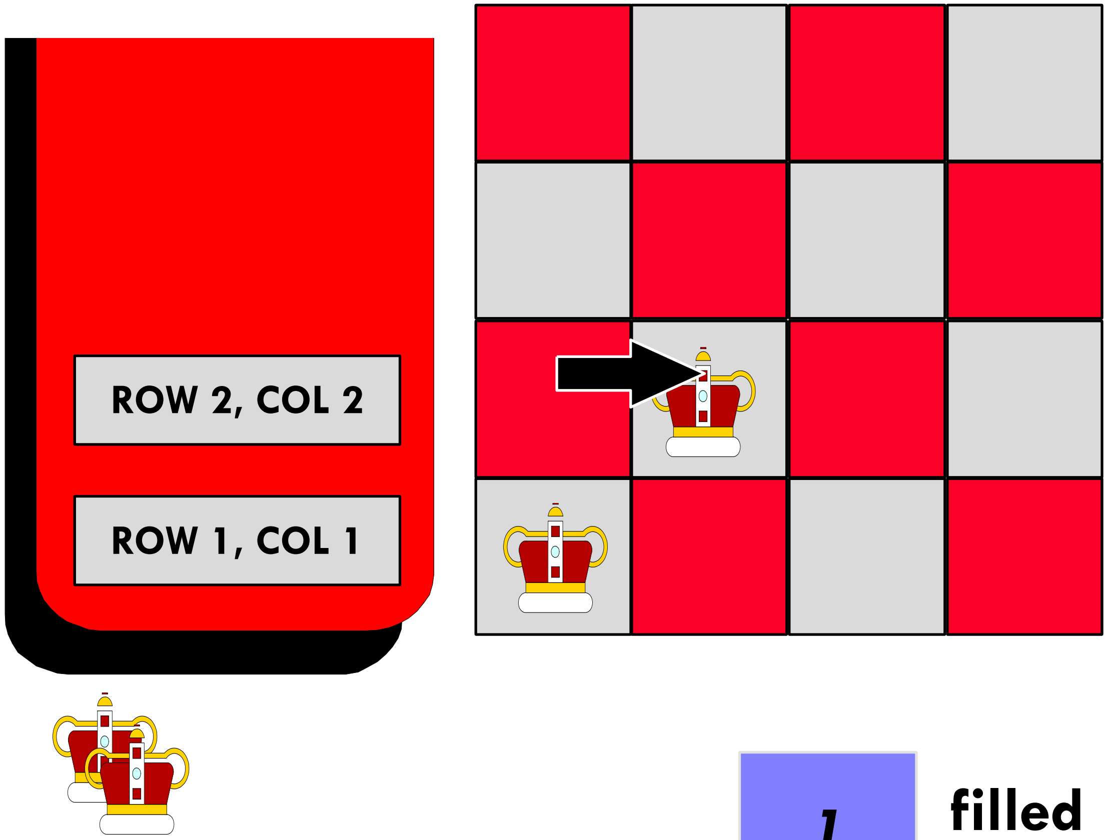
How the program works

Each time we try to place a new queen in the next row, we start by placing the queen in the first column...



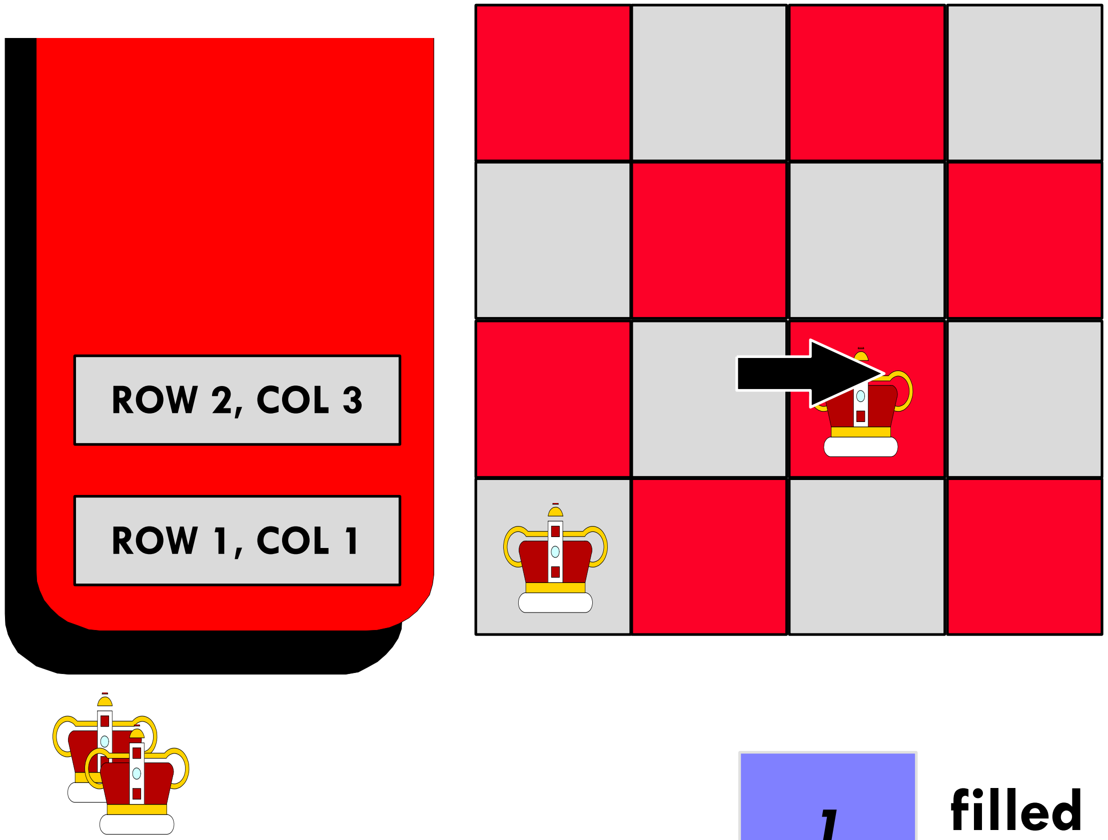
How the program works

…if there is a conflict with another queen, then we shift the new queen to the next column.



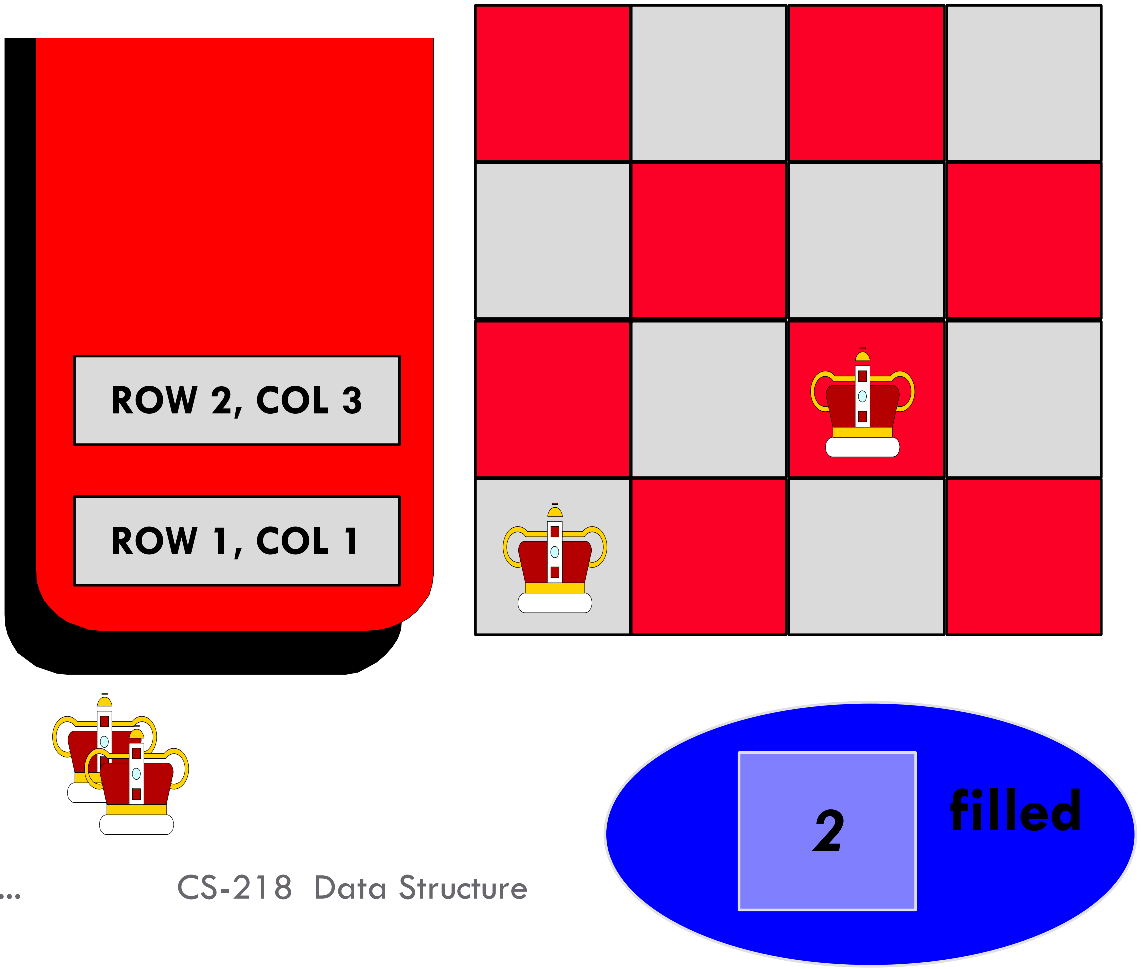
How the program works

If another conflict occurs, the queen is shifted rightward again.



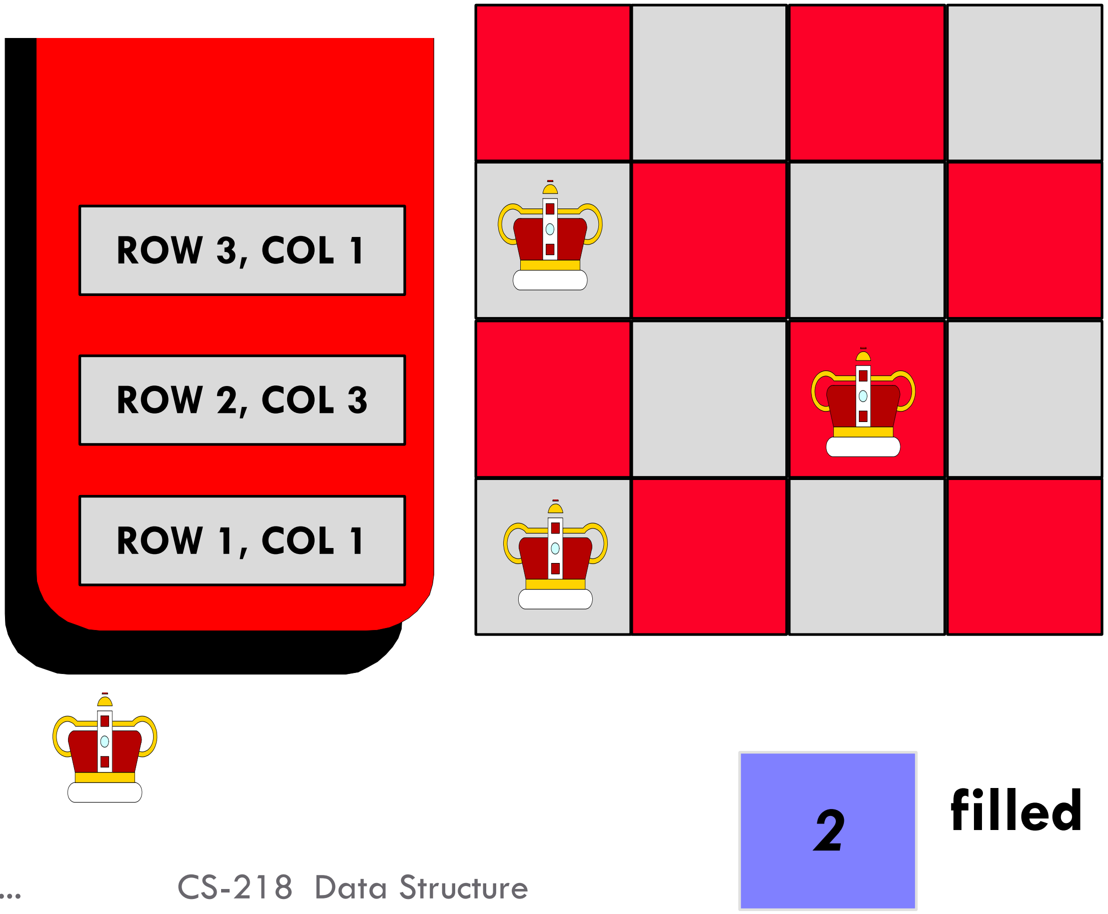
How the program works

When there are no conflicts, we stop and add one to the value of filled.



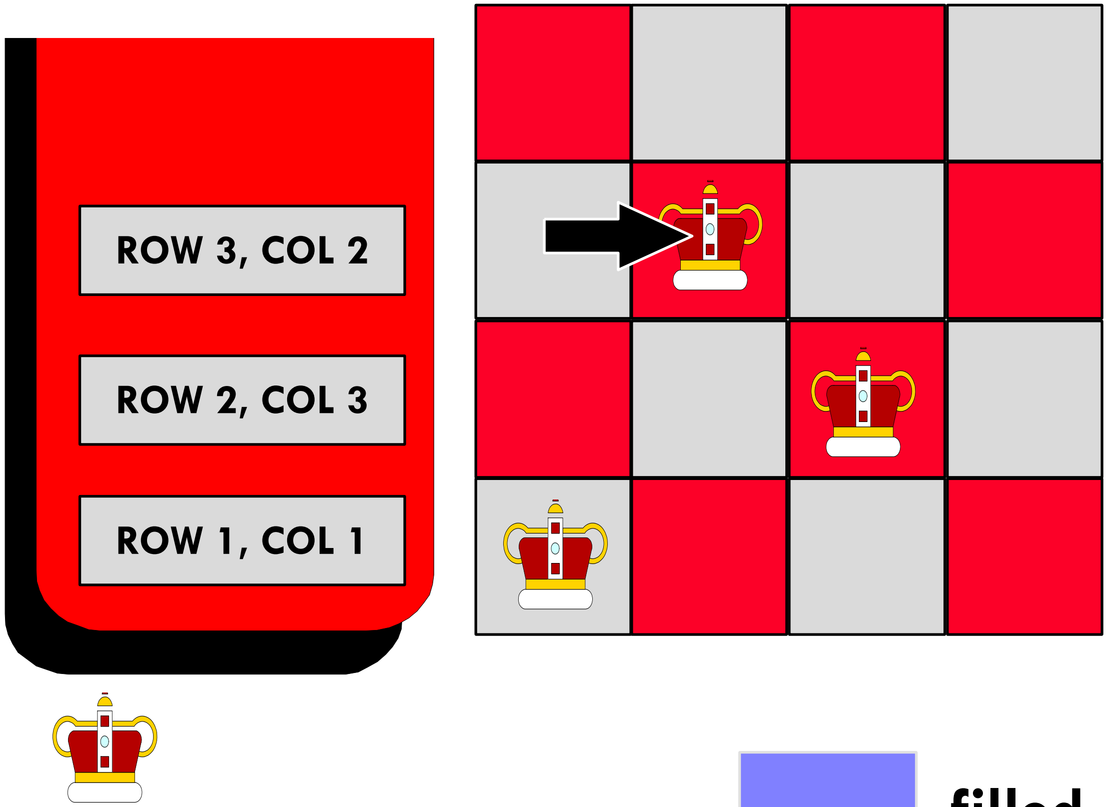
How the program works

Let's look at the third row. The first position we try has a conflict...



How the program works

...so we shift to column 2. But another conflict arises...

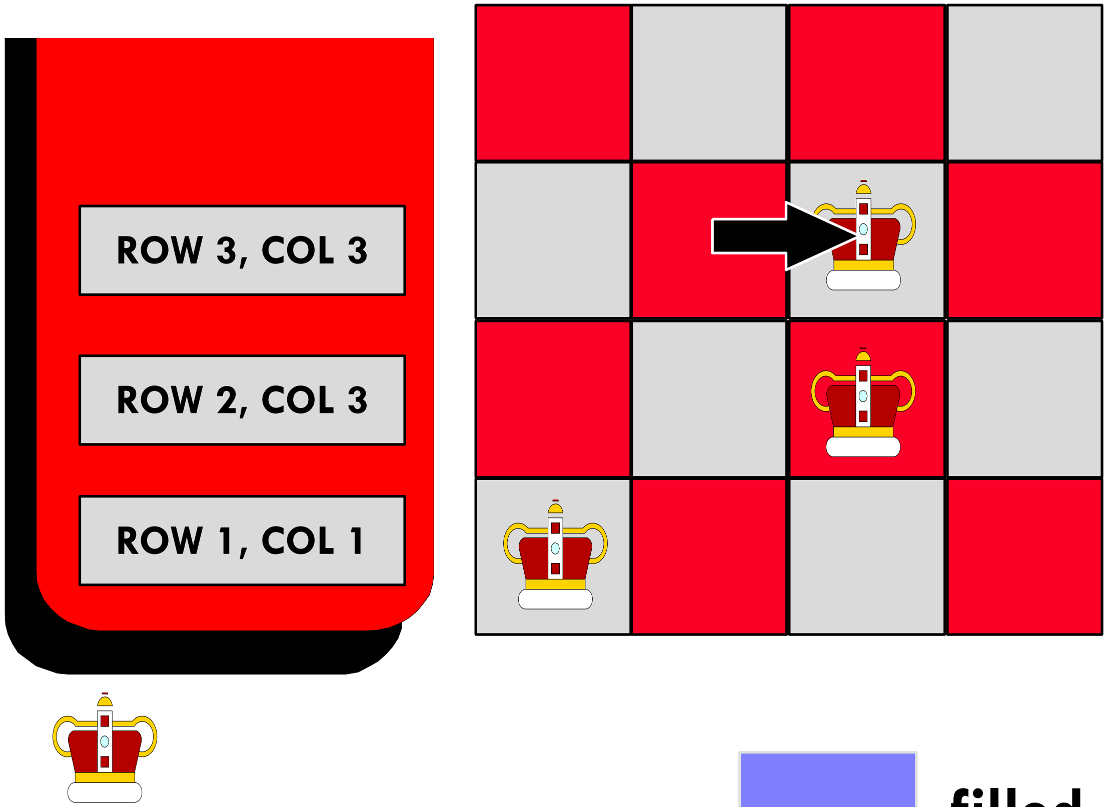


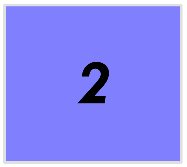
2 filled

How the program works

...and we shift to the third column.

Yet another conflict arises...

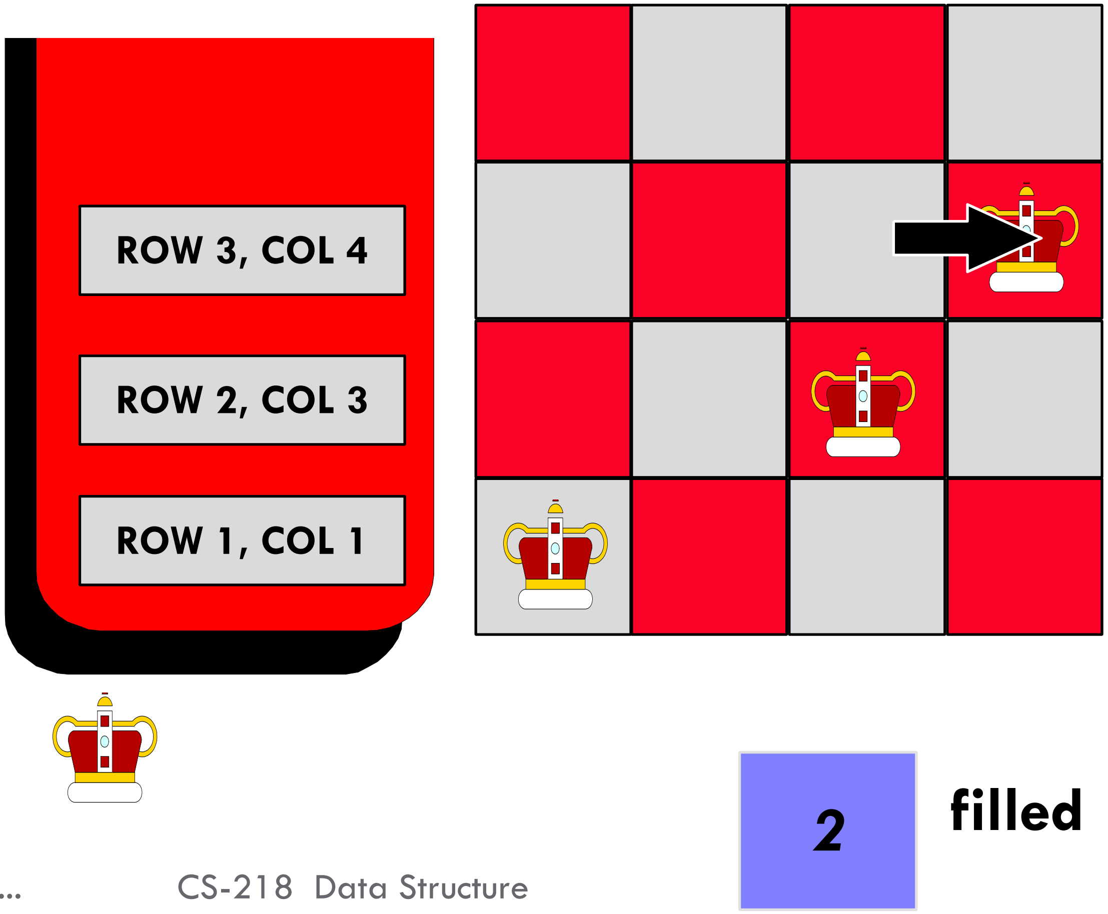




filled

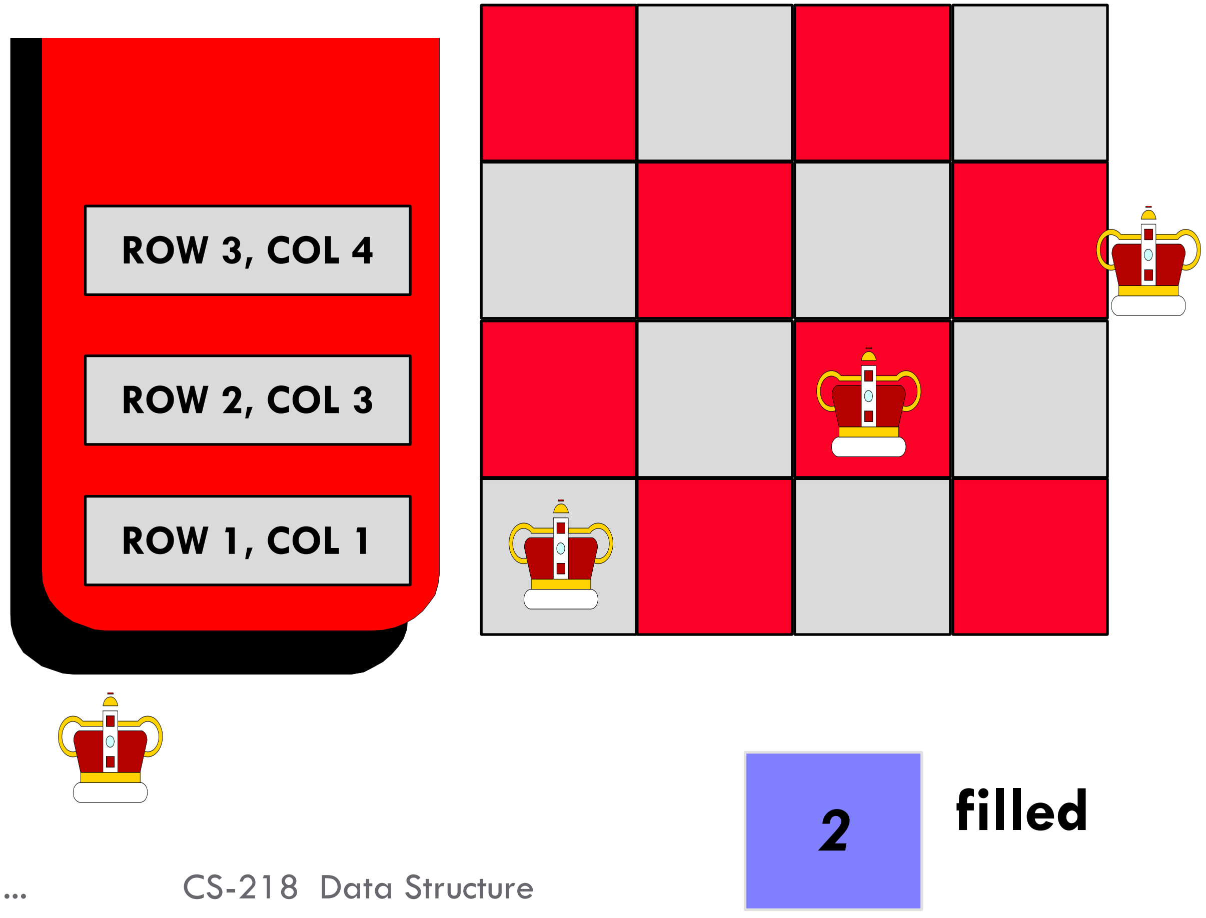
How the program works

...and we shift to column 4. There's still a conflict in column 4, so we try to shift rightward again...



How the program works

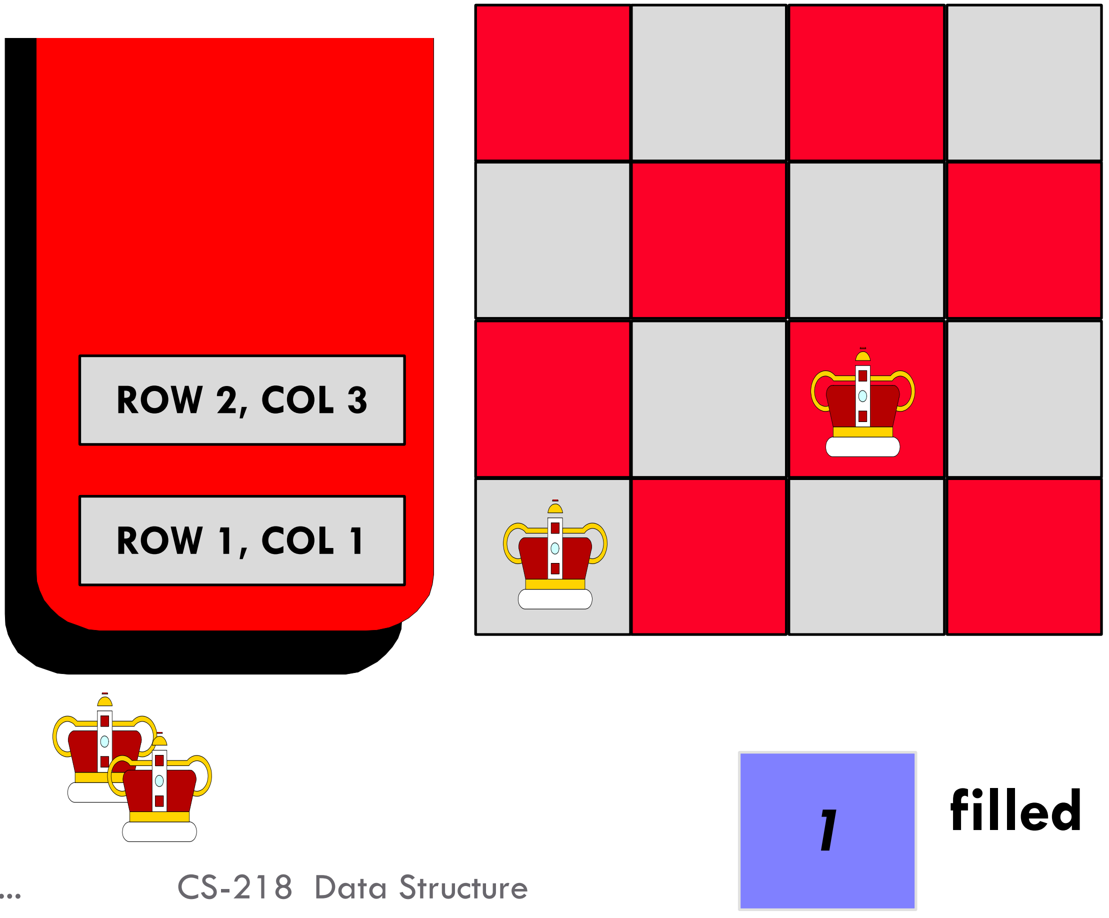
...but there's nowhere else to go.



How the program works

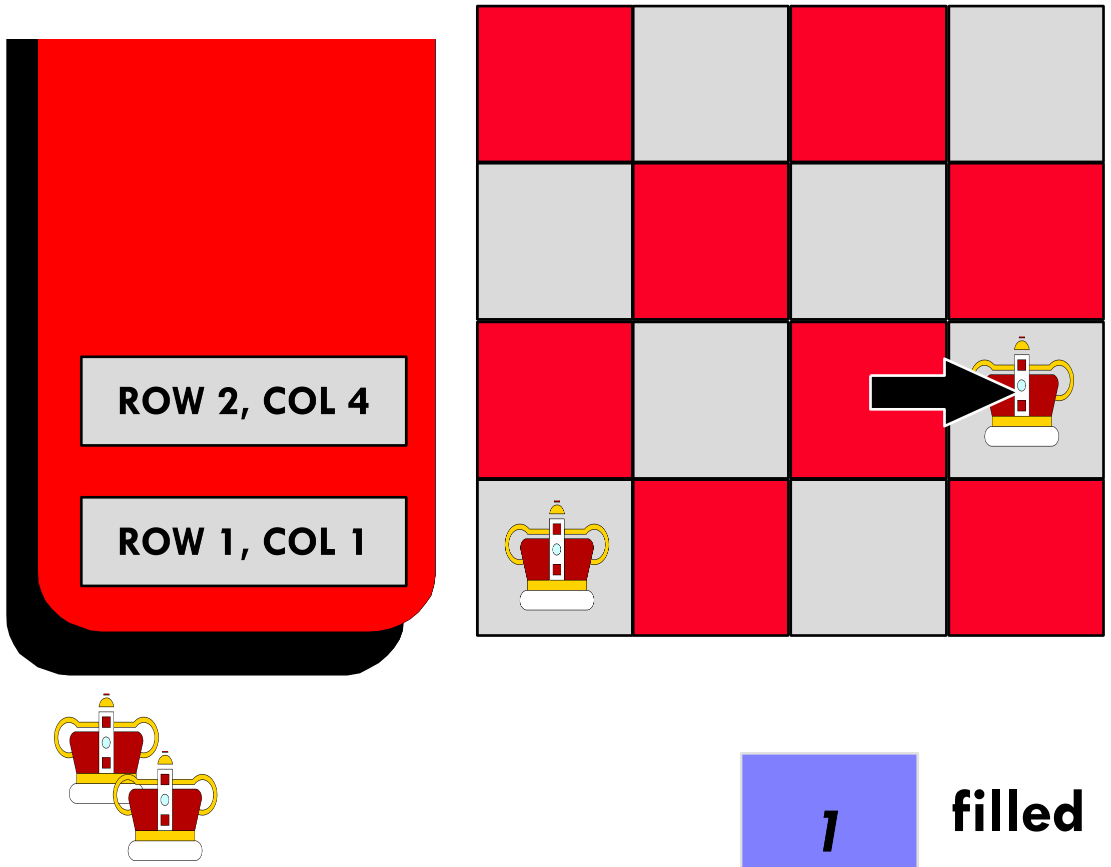
<u>When we run out of room in a row:</u>

- pop the stack,
- reduce filled by 1
- and continue working on the previous row.



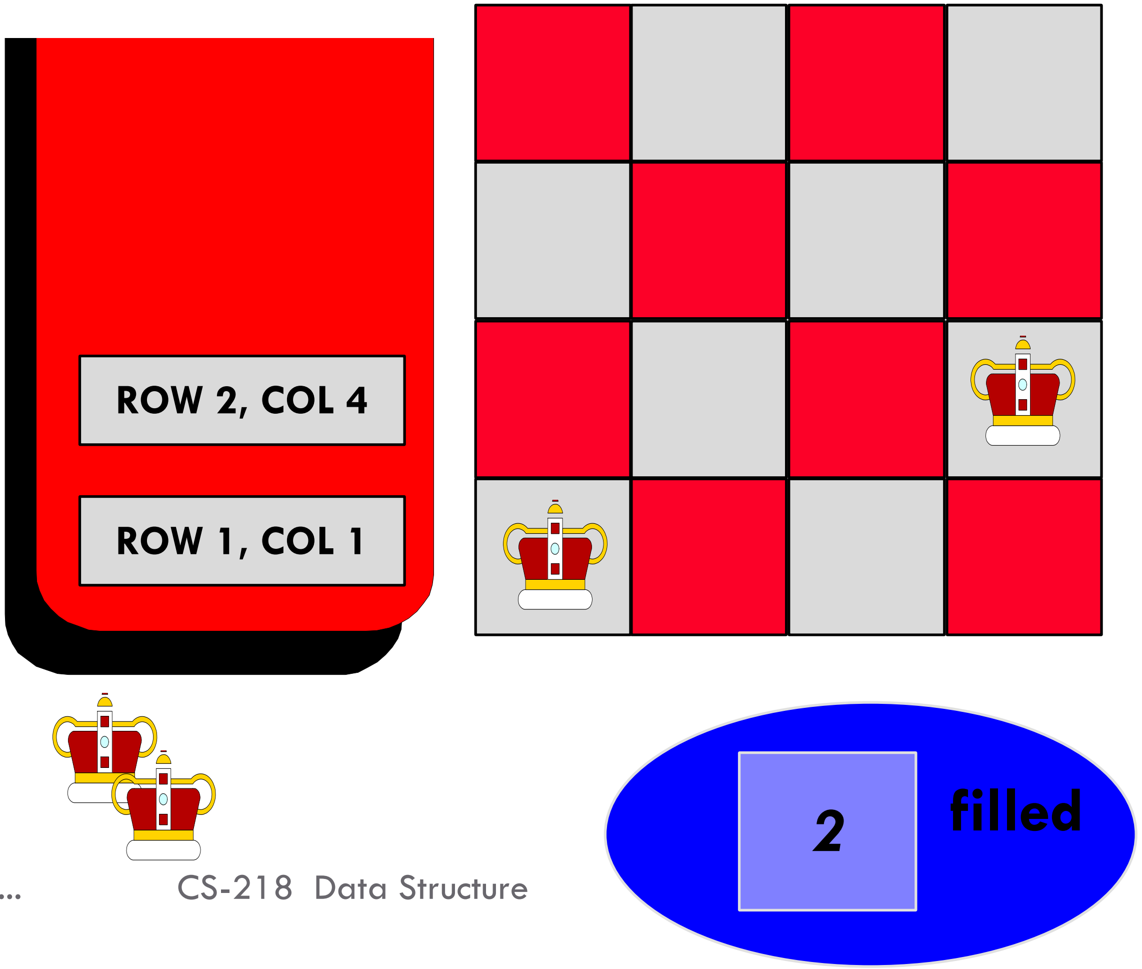
How the program works

Now we continue working on row 2, shifting the queen to the right.



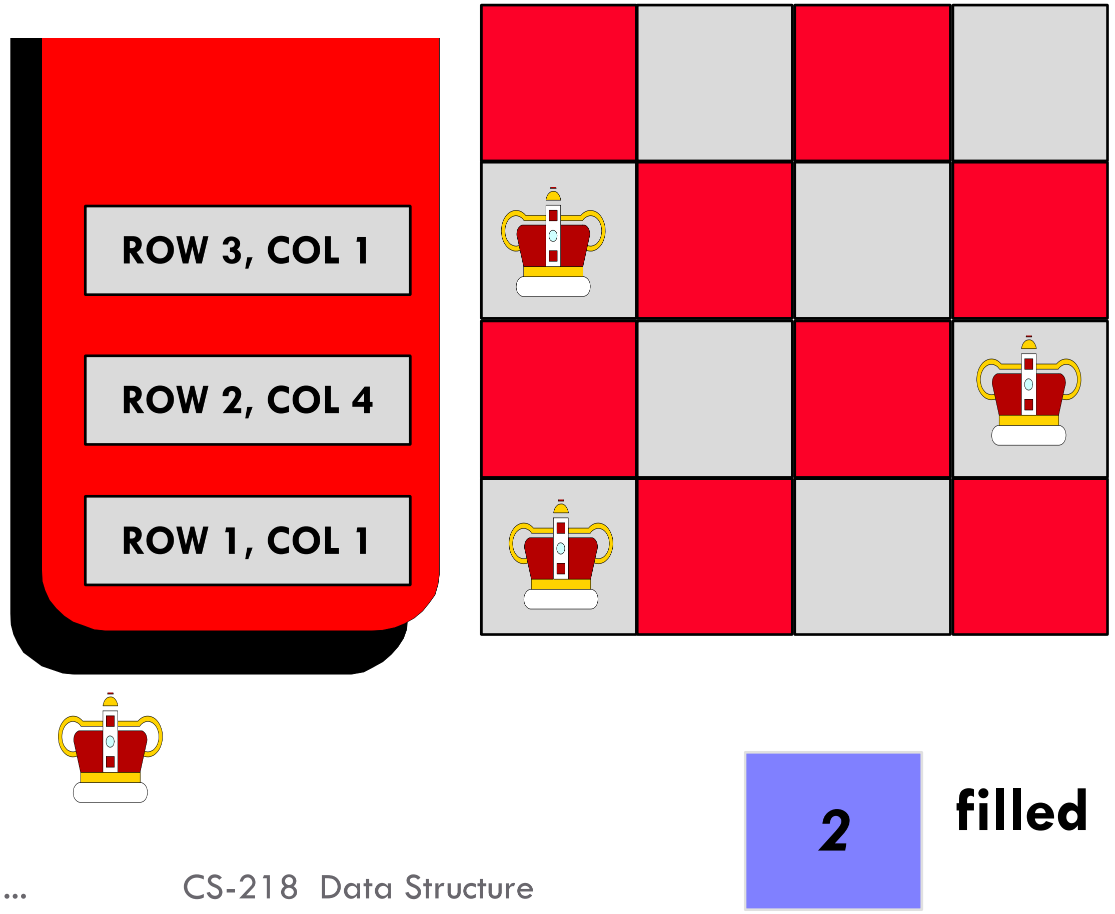
How the program works

This position has no conflicts, so we can increase filled by 1, and move to row 3.

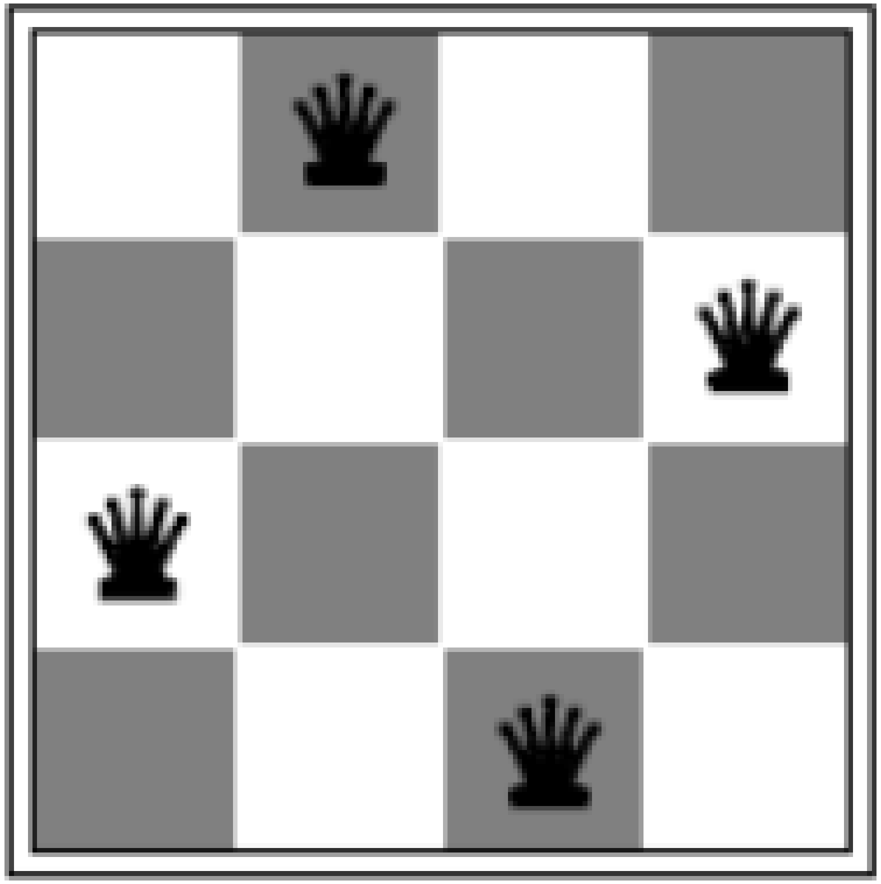


How the program works

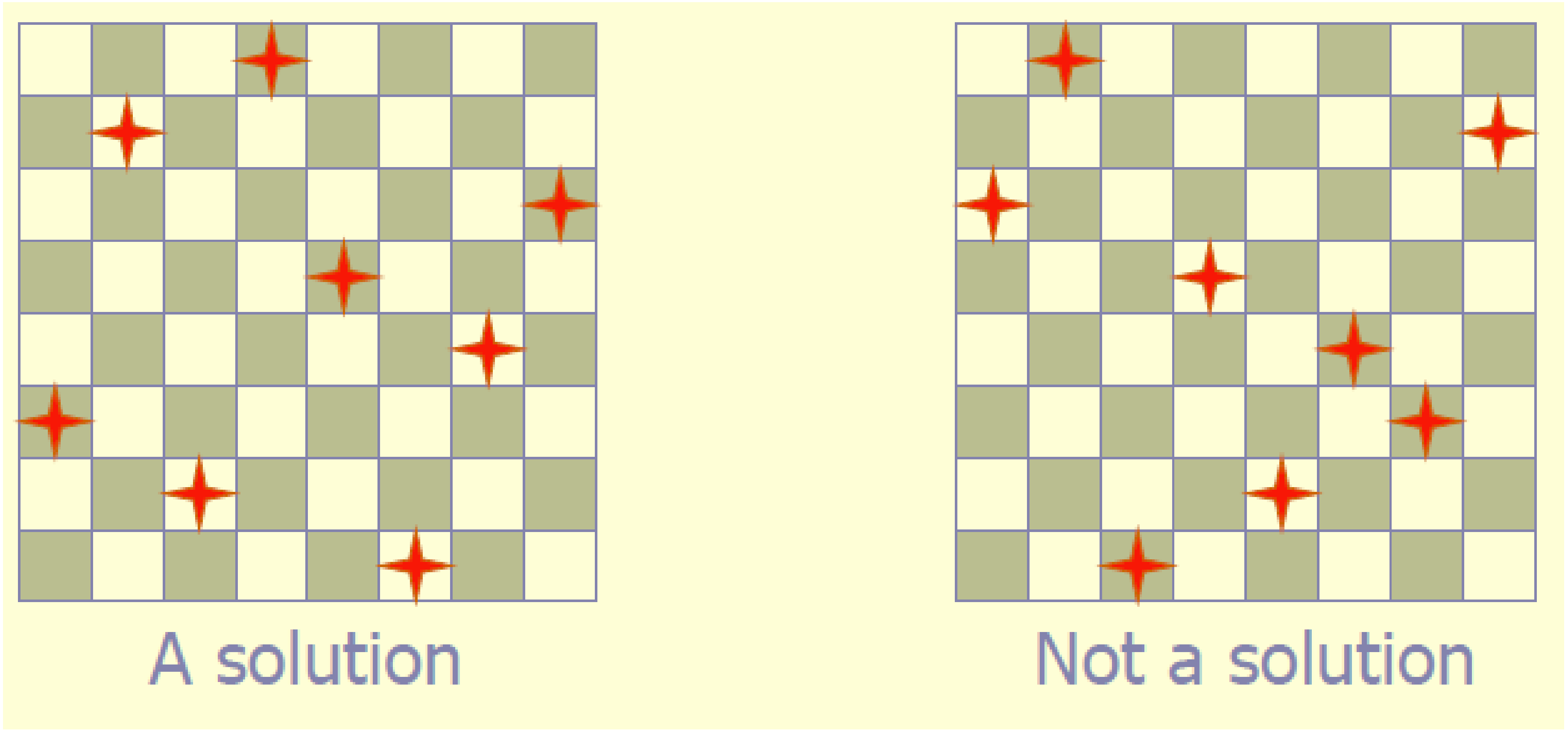
In row 3, we start again at the first column.



Solution



8-Queen Problem



PSEUDOCODE FOR N-QUEEN PROBLEM

Pseudocode for N-Queens

- **Initialize a stack** where we can keep track of our decisions and setting `filled` to 0.
- Place the first queen, pushing its position onto the stack.
- Repeat these steps:
  - if there are no conflicts with the queens...
  - else if there is a conflict and there is room to shift the current queen rightward...
  - else if there is a conflict and there is NO room to shift the current queen rightward...

Pseudocode for N-Queens

- Repeat these steps
  - if there are no conflicts with the queens...

Increase `filled` by 1. *If `filled` is now N, then the algorithm is done.* Otherwise, move to the next row and place a queen in the first column.

Pseudocode for N-Queens

- Repeat these steps
  - if there are no conflicts with the queens...
  - else if there is a conflict and there is room to shift the current queen rightward...

Move the current queen rightward, adjusting the record on top of the stack to indicate the new position.

Pseudocode for N-Queens

- Repeat these steps
  - if there are no conflicts with the queens...
  - else if there is a conflict and there is room to shift the current queen rightward...
  - else if there is a conflict and there is no room to shift the current queen rightward...

Backtrack!
Keep popping the stack, and reducing filled by 1, **until you reach a row where the queen can be shifted rightward**. Shift this queen right.

Reading Materials

- Schaum's Outlines: Chapter # 6
- D. S. Malik: Chapter # 7
- Nell Dale: Chapter # 4
- Mark A. Weiss: Chapter # 3
- Chapter 7, ADT, Data structures and problem-solving using C++ , Larry Nyhoff.