

PROGRAMMING ASSIGNMENT #6
CS 2223 B-TERM 2024
BACKTRACKING
AND
THE n -QUEENS PROBLEM

ONE HUNDRED POINTS
DUE: THURSDAY, DECEMBER 12, 2024 11PM

We crown the term with the n -Queens Problem.

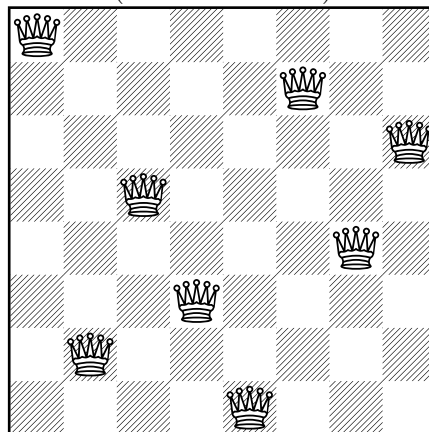
The challenge is to place n Queens on an $n \times n$ board (rectangular array?), so that no two attack each other, i.e. no two Queens may be on the same rank (row), file (column), or diagonal (????).

1. (20 Points) `ISLEGALPOSITION(BOARD, n)`

Write a method `ISLEGALPOSITION(BOARD, n)` that takes a (possibly partial) position and n as arguments and returns `TRUE` if and only if no two Queens attack each other.

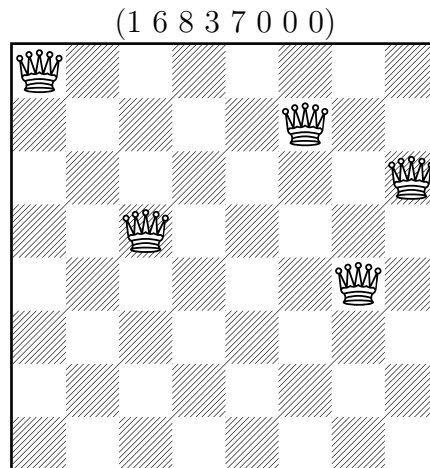
Here is a solution to the 8-Queens Problem:

(1 6 8 3 7 4 2 5)

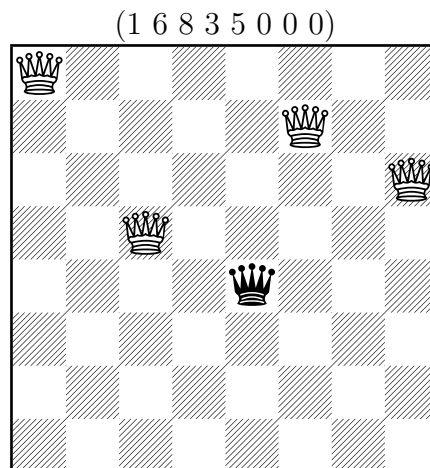


Thus, `ISLEGALPOSITION((1 6 8 3 7 4 2 5),8)` should return `TRUE`.

Because we are implementing a backtracking algorithm, we will restrict ourselves to positions which fill from the top of the board. We will insist then that the first $k \leq n$ positions be filled, i.e. non-zero, but the remaining $n - k$ positions may be zeroes. So the partial solution:



should also have `ISLEGALPOSITION((1 6 8 3 7 0 0 0),8)` return `TRUE`, while



should cause `ISLEGALPOSITION((1 6 8 3 5 0 0 0),8)` to return `FALSE`.

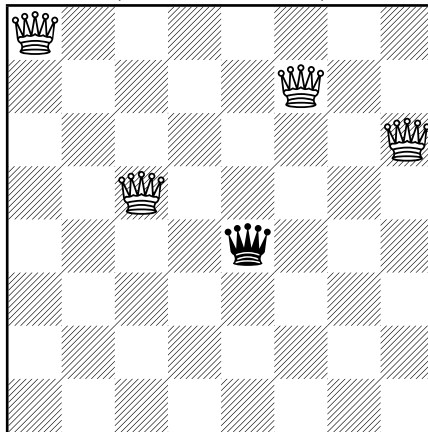
Why?

Do you see an elegant way to check that?

2. (20 Points) `NEXTLEGALPOSITION(BOARD,n)`

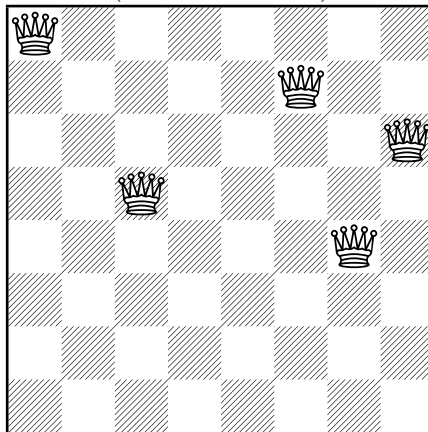
From any (possibly partial) position, we need to be able to find the *next* legal position. There are, perhaps, three cases here. First, the next legal position from an illegal partial position; second, the next legal position from a *legal* partial position, and third, the next legal position after a full-fledged solution. We will fill our board from the top down and from left to right, so the next legal position after (illegal) partial position:

(1 6 8 3 5 0 0 0)



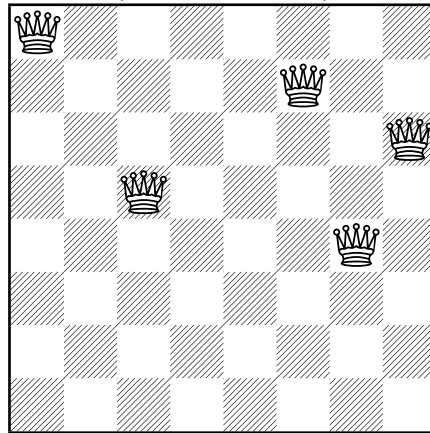
is

(1 6 8 3 7 0 0 0)



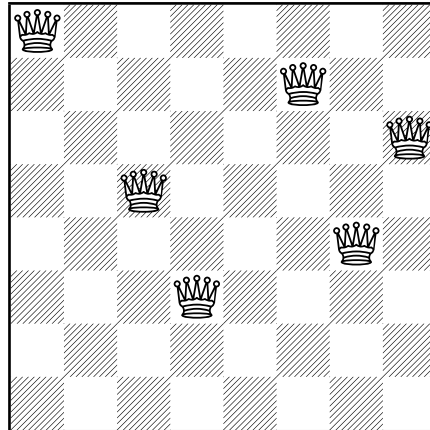
And the next legal position after (legal!) partial position:

(1 6 8 3 7 0 0 0)



is

(1 6 8 3 7 4 0 0)

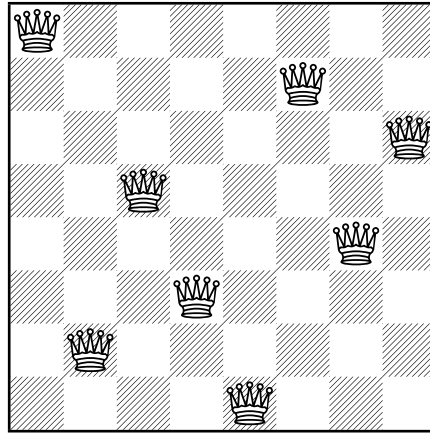


Will the next legal position *from* a legal position *always* add a Queen to the next rank?

Why? / Why not?

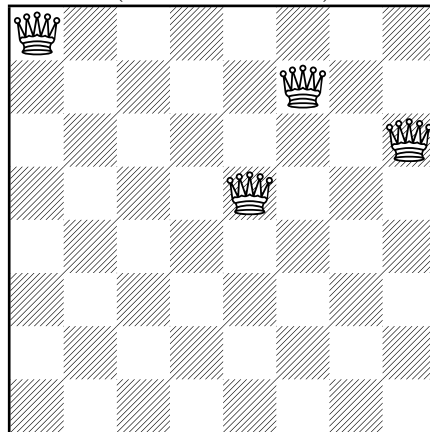
Lastly, the next legal position after our solution:

(1 6 8 3 7 4 2 5)



is

(1 6 8 5 0 0 0 0)



(Understanding this is understanding the backtracking—and then the forwarding—we are doing. This is the crux of the method.)

Write a method `NEXTLEGALPOSITION(BOARD, n)` that takes a (possibly partial) position and n as arguments and returns a board/position/array that represents the next legal position, or $(0_1, 0_2, \dots, 0_n)$ if no legal position succeeds “board”.

Hint: It may be useful to write another method `SUCCESSOR(BOARD, n)` that returns the next position to “board”, whether legal or not.

3. (30 Points) Find the “first” solution to the n -Queens Problem for $n = 4 \dots 100^\dagger$.

With `ISLEGALPOSITION(BOARD,n)` and `NEXTLEGALPOSITION(BOARD,n)` in your hip pocket, write a program which solves the n -Queens problem for all values between 4 and 100, inclusive.

Your output should give a single solution to each instance of the problem, and it should be the *first* solution lexicographically.

We saw that the 4-Queens problem has solutions (2, 4, 1, 3) and (3, 1, 4, 2) as its distinct solutions. Your output should be the first of these.

Is our solution to the 8-Queens problem the first one?

4. (30 Points) Find all solutions to the n -Queens Problem for a particular n .

With `ISLEGALPOSITION(BOARD,n)` and `NEXTLEGALPOSITION(BOARD,n)` in your hip pocket, write a program/method which finds (counts) all solutions to the n -Queens problem for each instance of the problem with $4 \leq n \leq 20^\ddagger$. Your output should be:

There are 2 solutions to the 4-Queens Problem.
There are 10 solutions to the 5-Queens Problem. ...
...
...

(What constitutes a “solution”?)

For parts 2-4, you can get some gains in efficiency by modifying `ISLEGALPOSITION(BOARD,n)`. We will *build* positions from *legal* positions. This means that only the last Queen, the last non-zero entry, can cause a position to be illegal. Do you see?

Why are you going to want increased efficiency?

You may find the Wikipedia entry on the “8-Queens puzzle” to be helpful. . . maybe even interesting.

[†]OK, you will NOT be able to go this high – we’re searching and pruning an n^n tree so we can do only so much. See how high you can go!

[‡]This is probably out of reach, too; there are more than 2 million solutions to $n = 15$.