N-Queens Problem

(Algorithmic Problems)

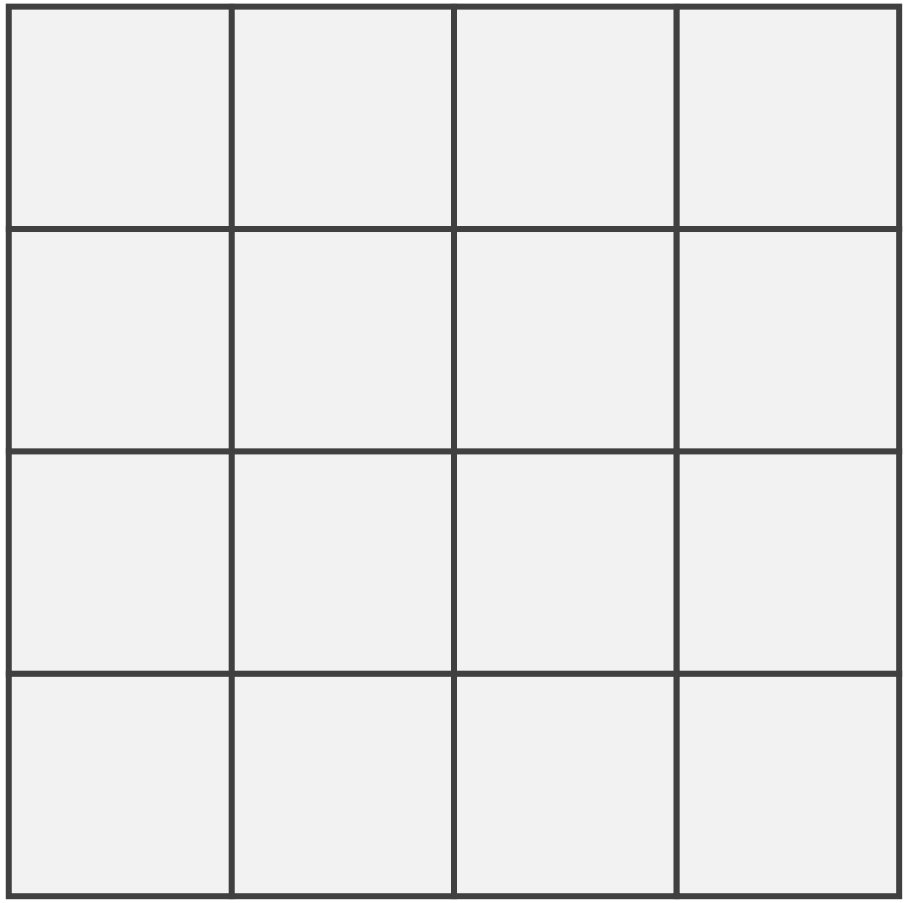
N-Queens Problem

- the problem of placing **N** chess queens on an **N×N** chessboard so that no two queens threaten each other
- queens can attack horizontally, vetically and we have to consider the diagonals too
- the original problem was designed for **8** queens (so **N=8**)
- **Gauss** worked on this problem and **Dijkstra** used this problem to illustrate the power of what he called structured programming

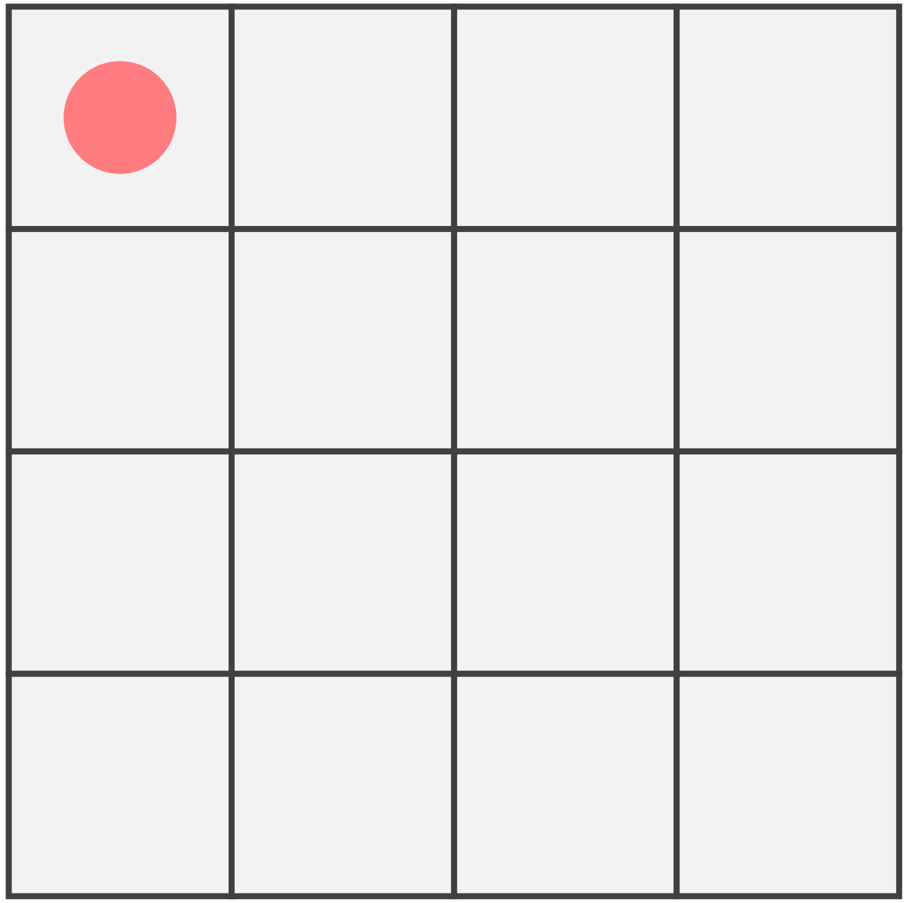
N-Queens Problem

- the problem of placing **N** chess queens on an **N×N** chessboard so that no two queens threaten each other
- how many possible states are there?
- there are **O(N^N)** possible states – that is **O(N!)** factorial running time complexity with brute-force approach
- **THERE ARE AN EXTREMELY HUGE AMOUNT OF STATES TO CONSIDER**
- we can use backtracking and eliminate bad states but the result will be **O(2^N)** which is still quite slow for large **N** values

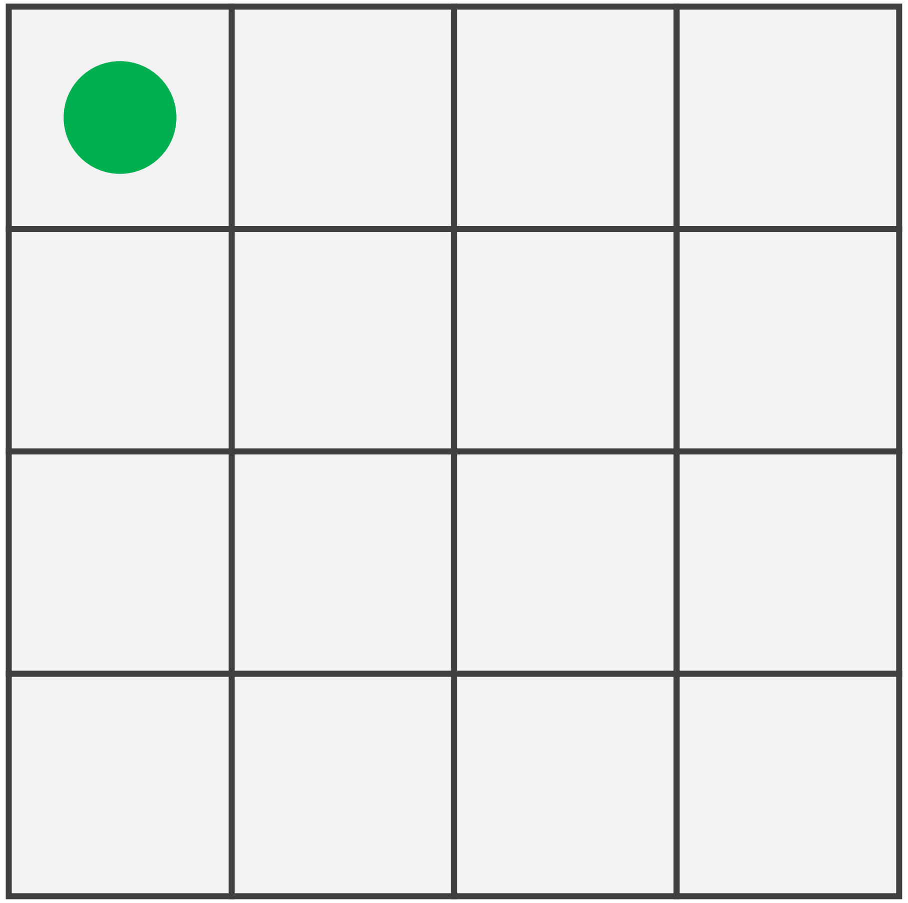
N-Queens Problem



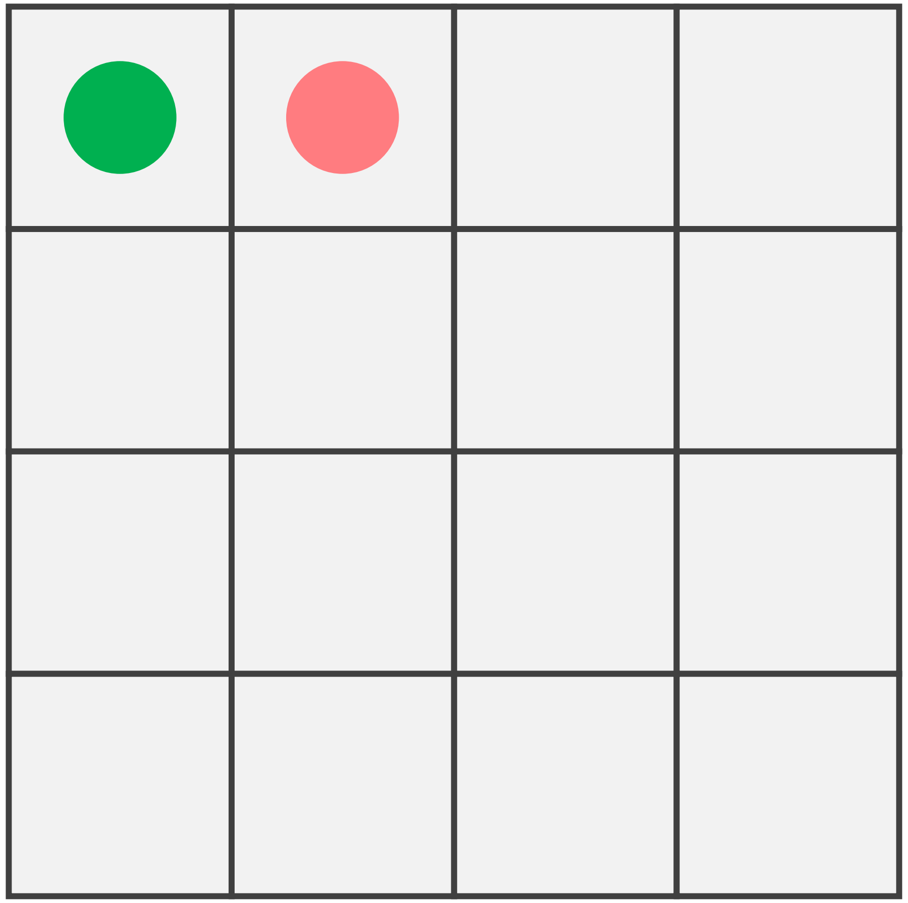
N-Queens Problem



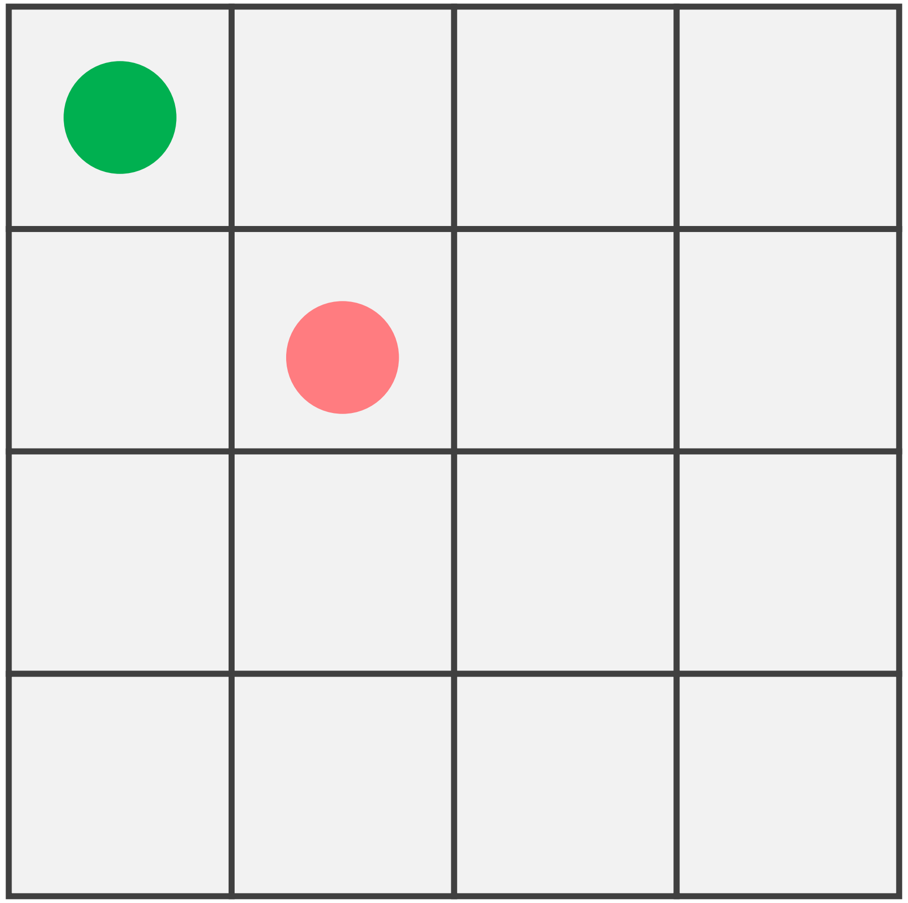
N-Queens Problem



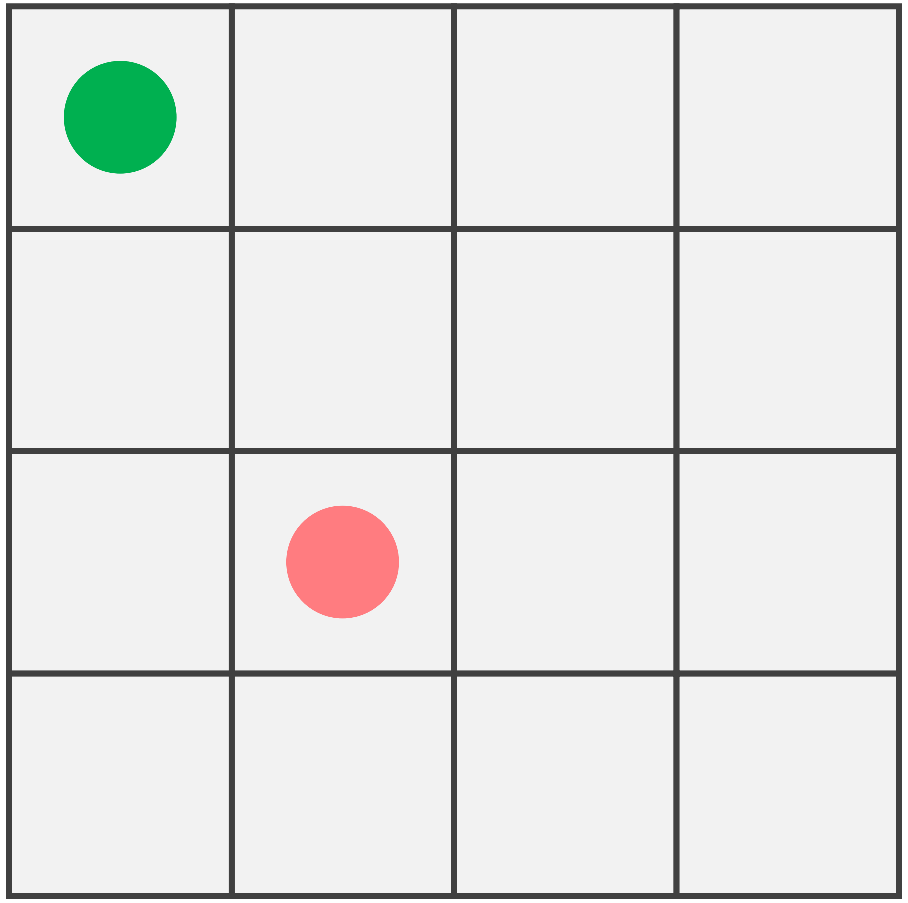
N-Queens Problem



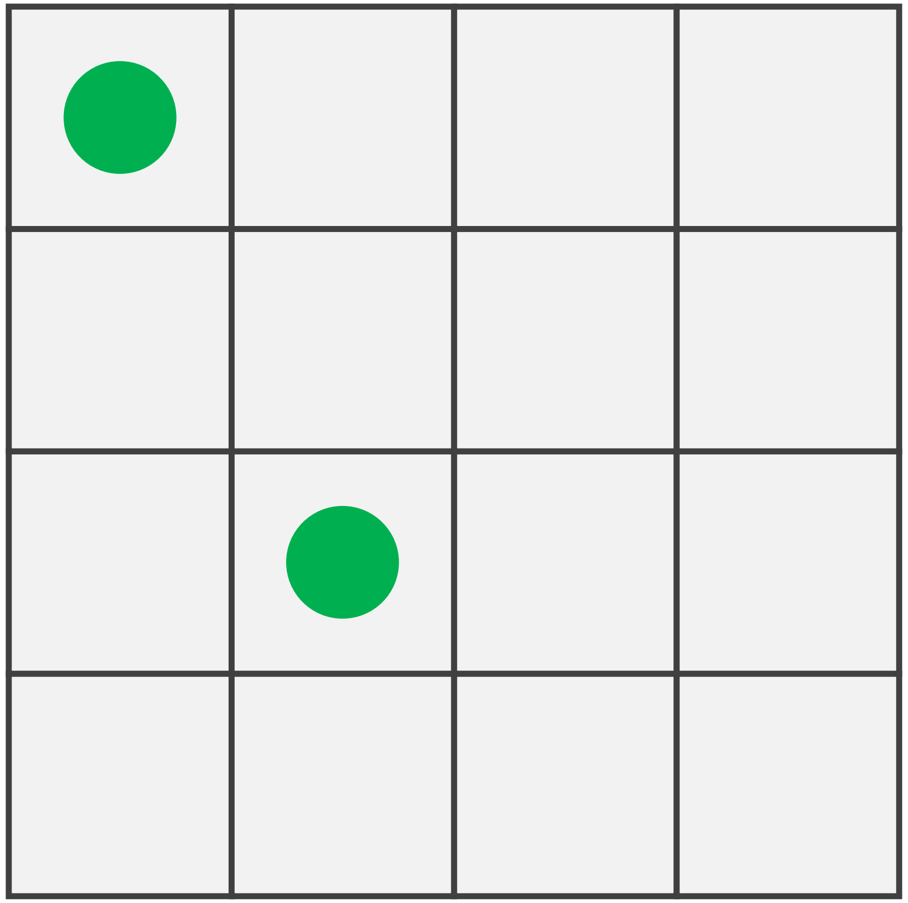
N-Queens Problem



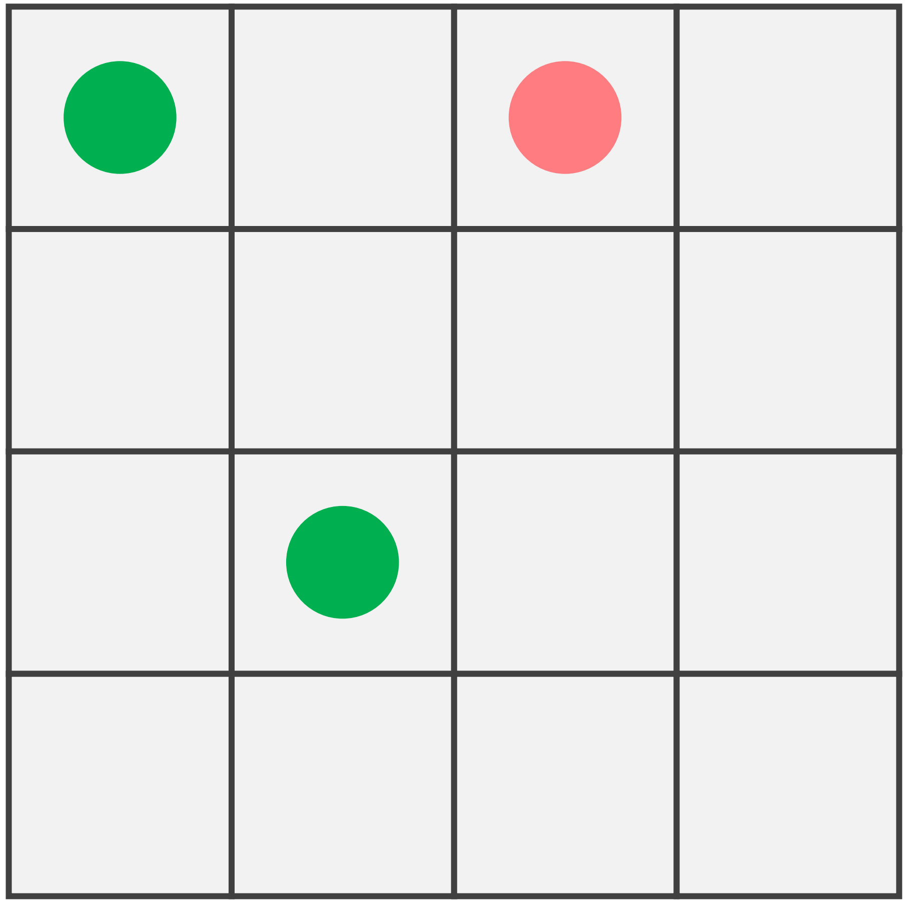
N-Queens Problem



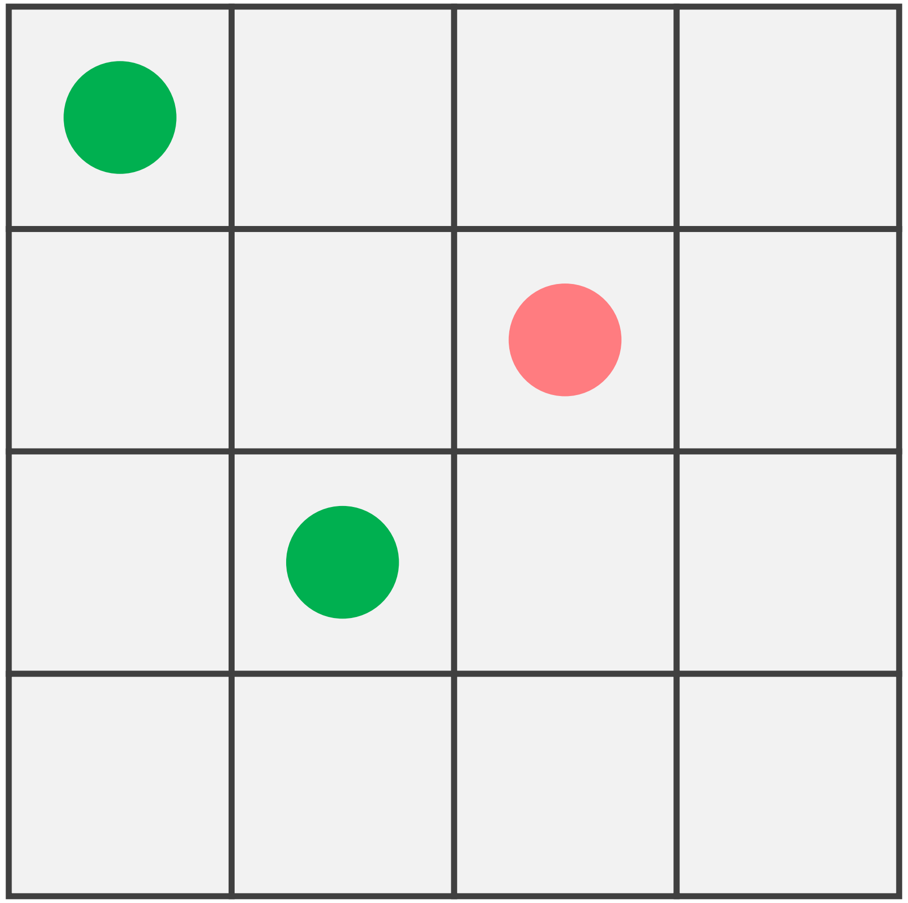
N-Queens Problem



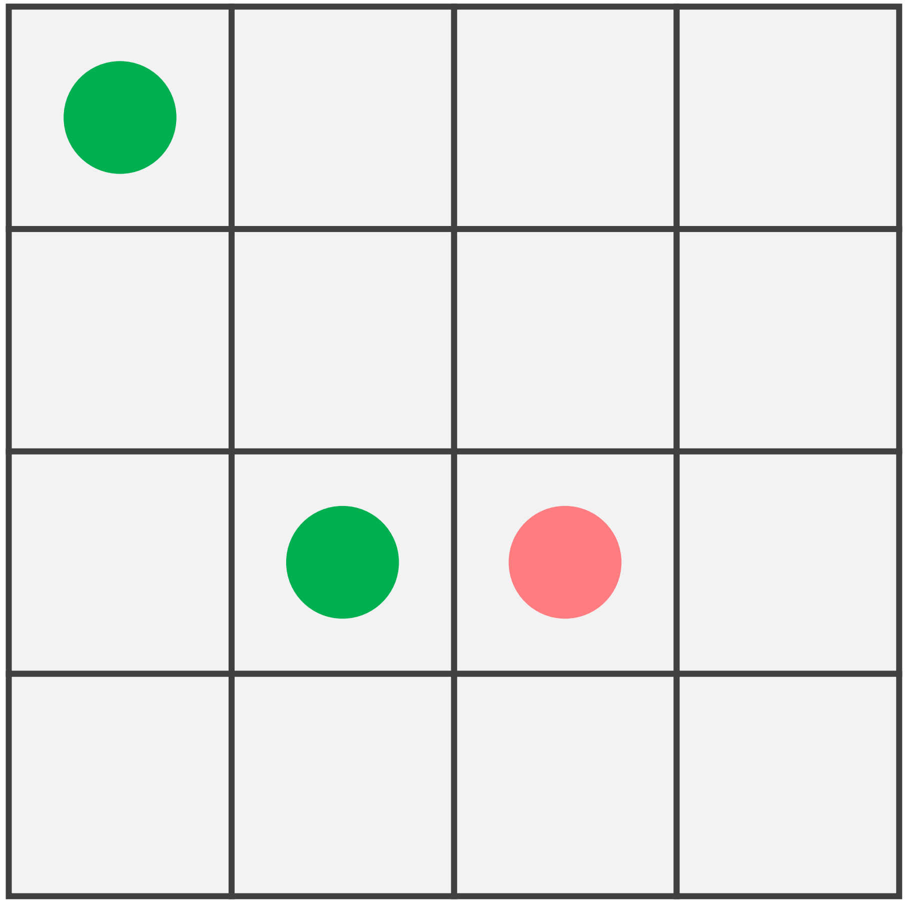
N-Queens Problem



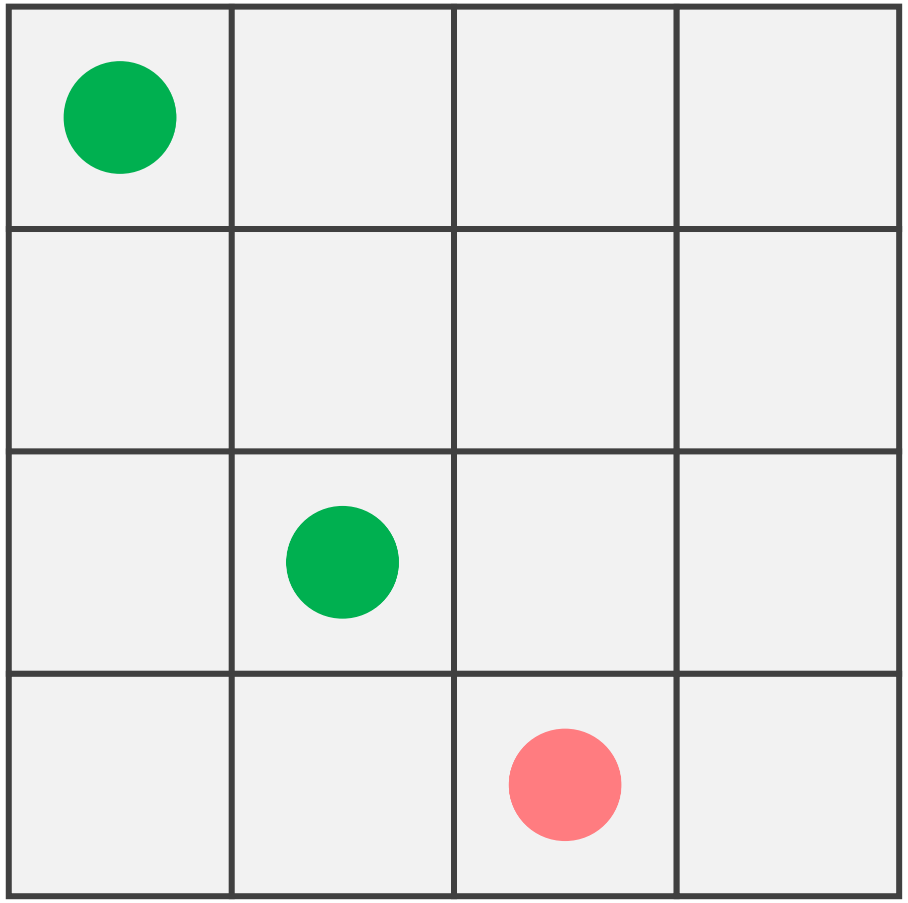
N-Queens Problem



N-Queens Problem



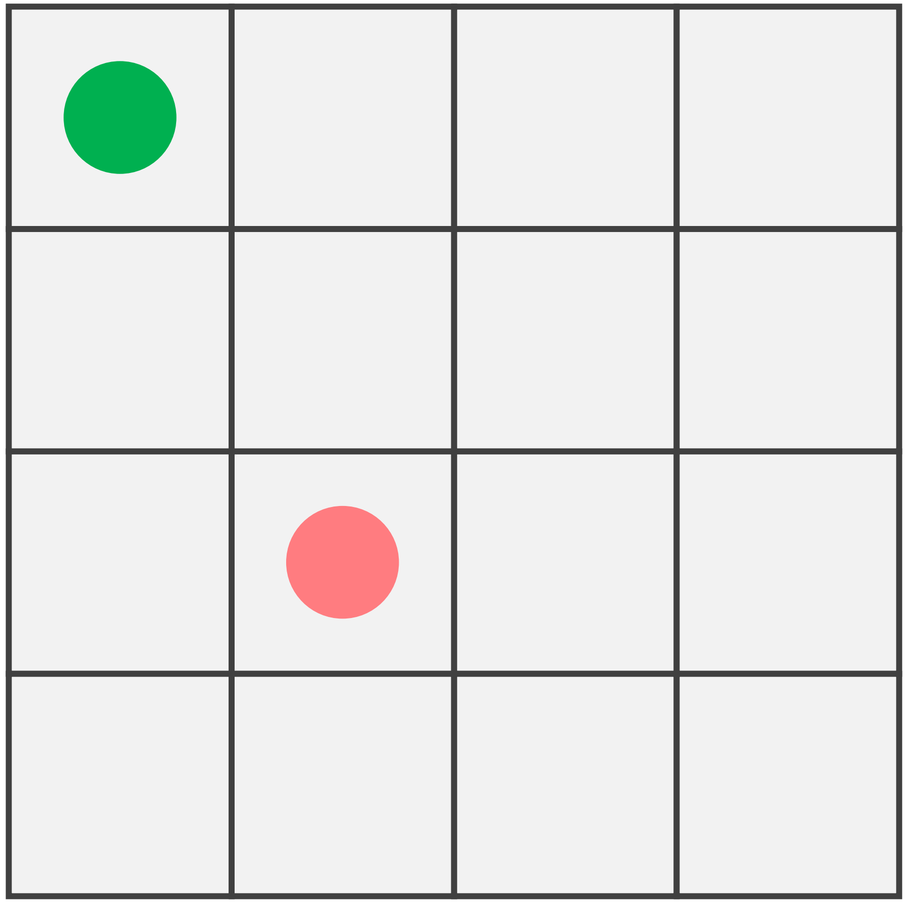
N-Queens Problem



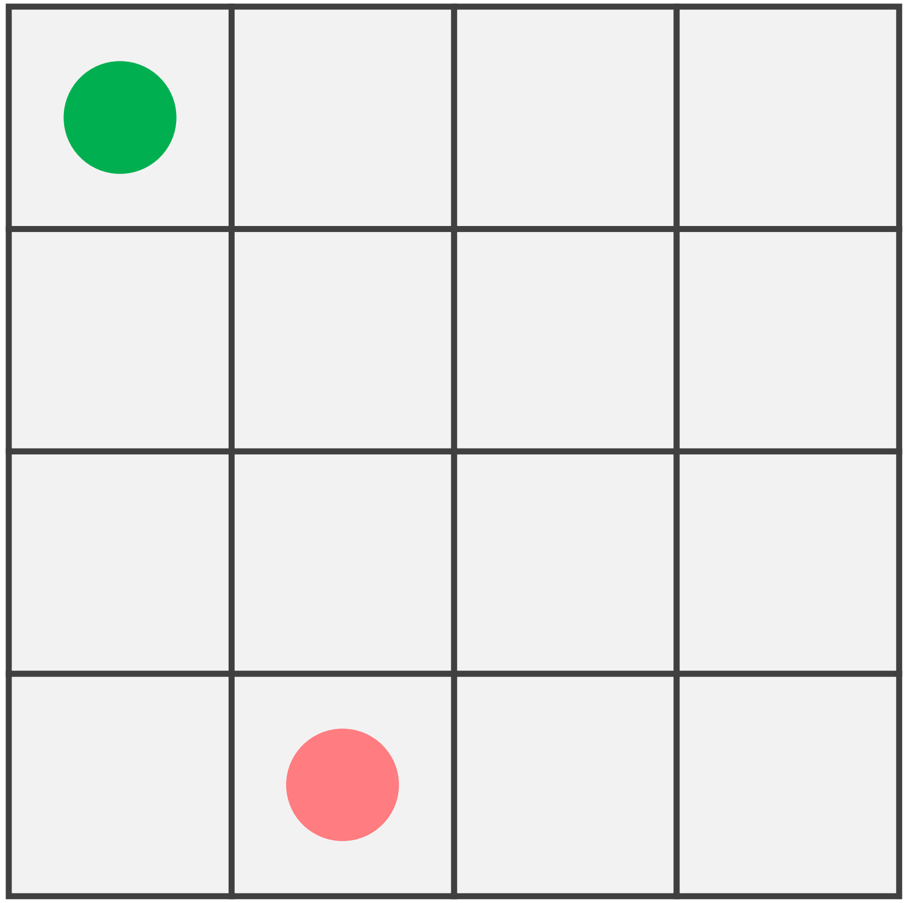
there is no feasible solution: we have to step back to the ***previous column*** *and increment the position of the queen there*

BACKTRACKING *!!!*

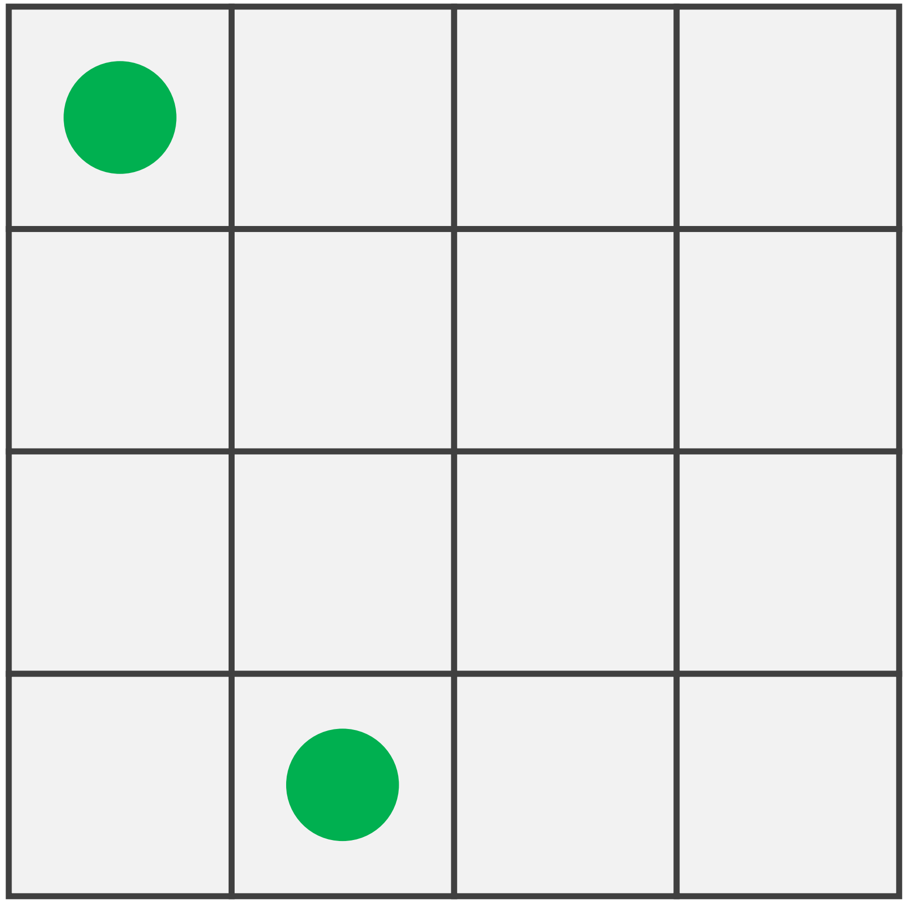
N-Queens Problem



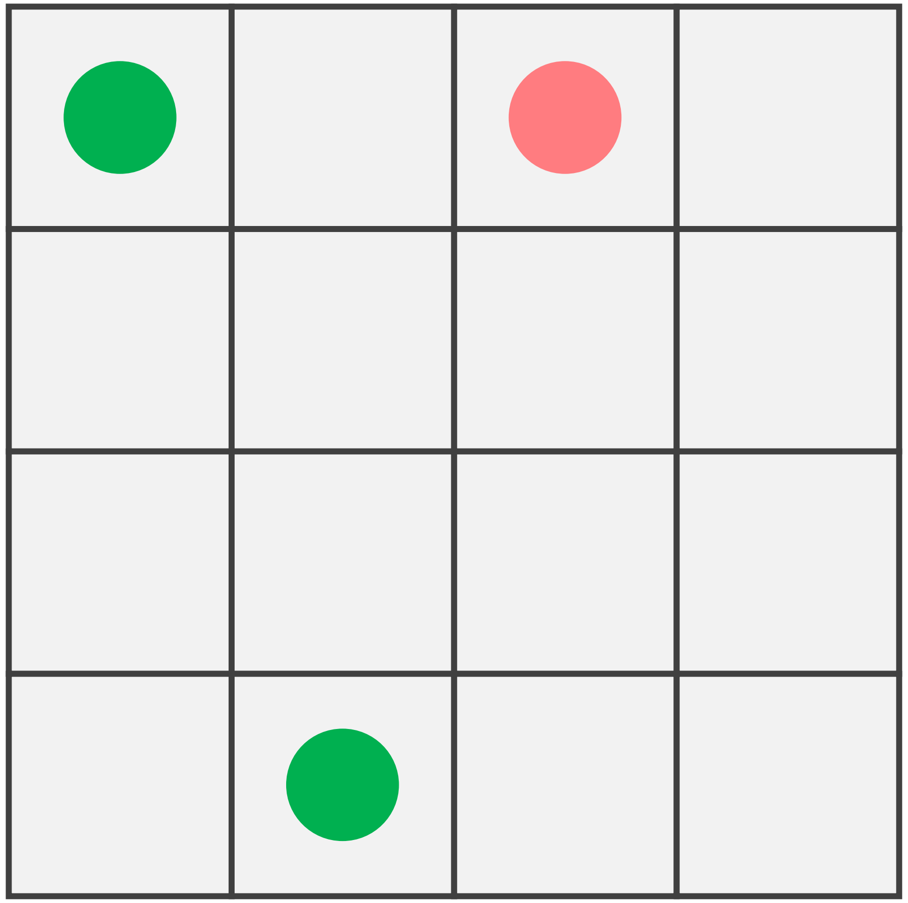
N-Queens Problem



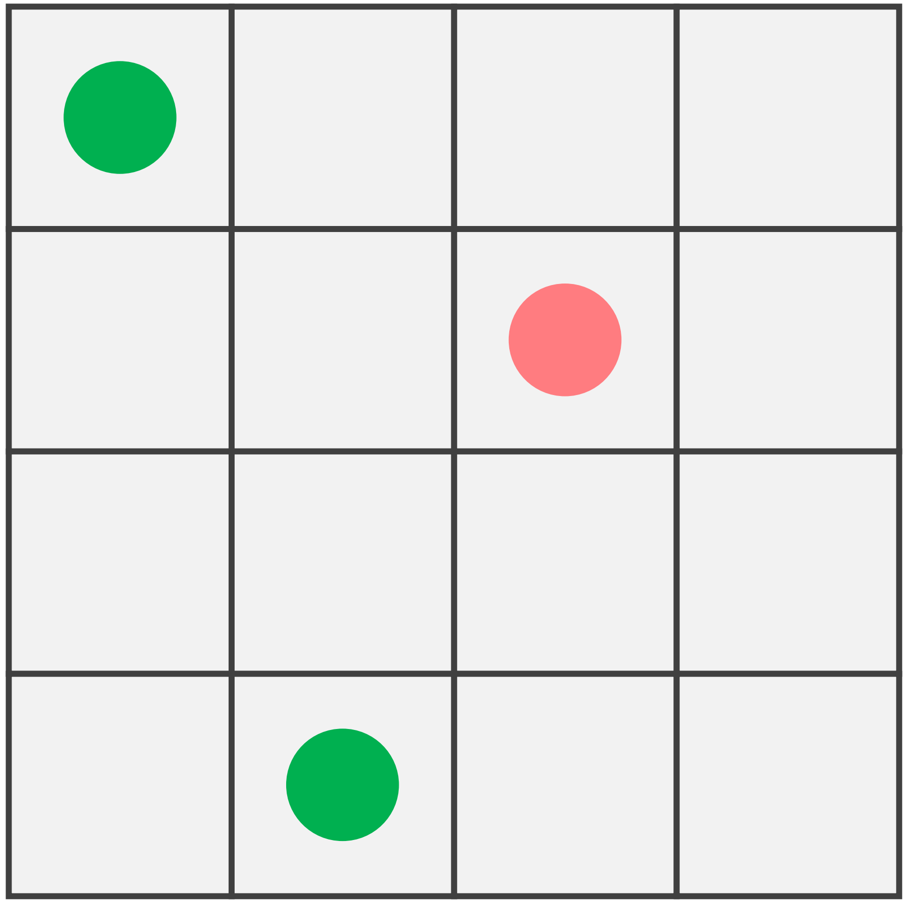
N-Queens Problem



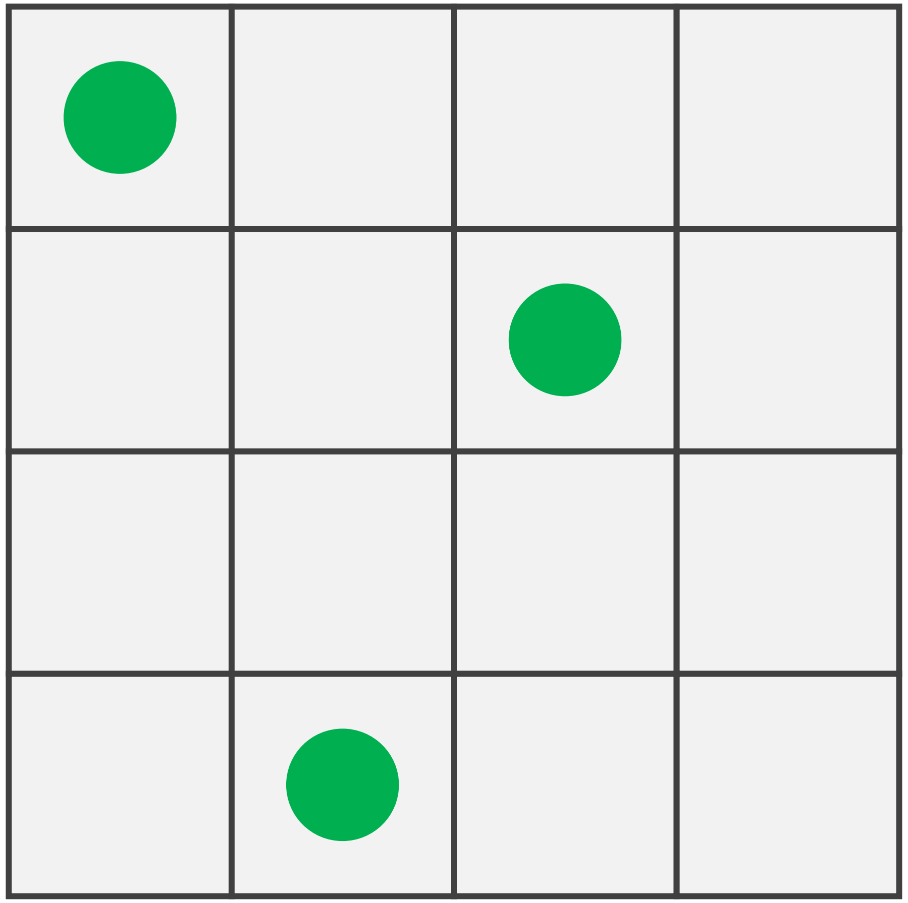
N-Queens Problem



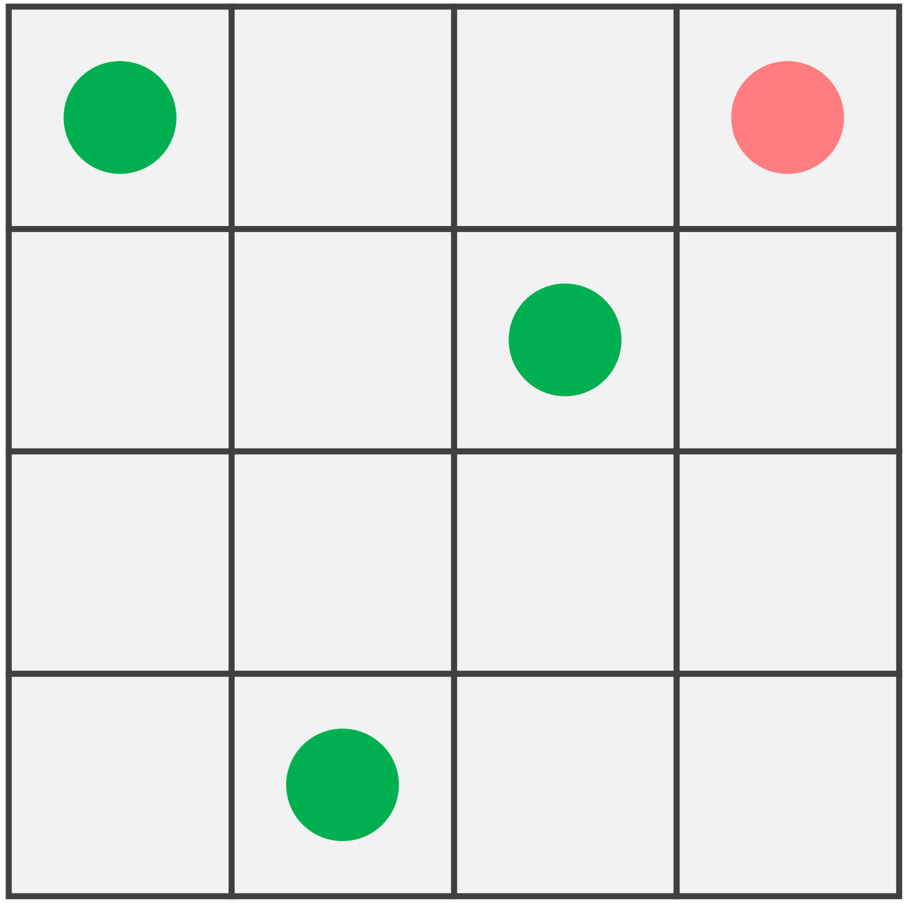
N-Queens Problem



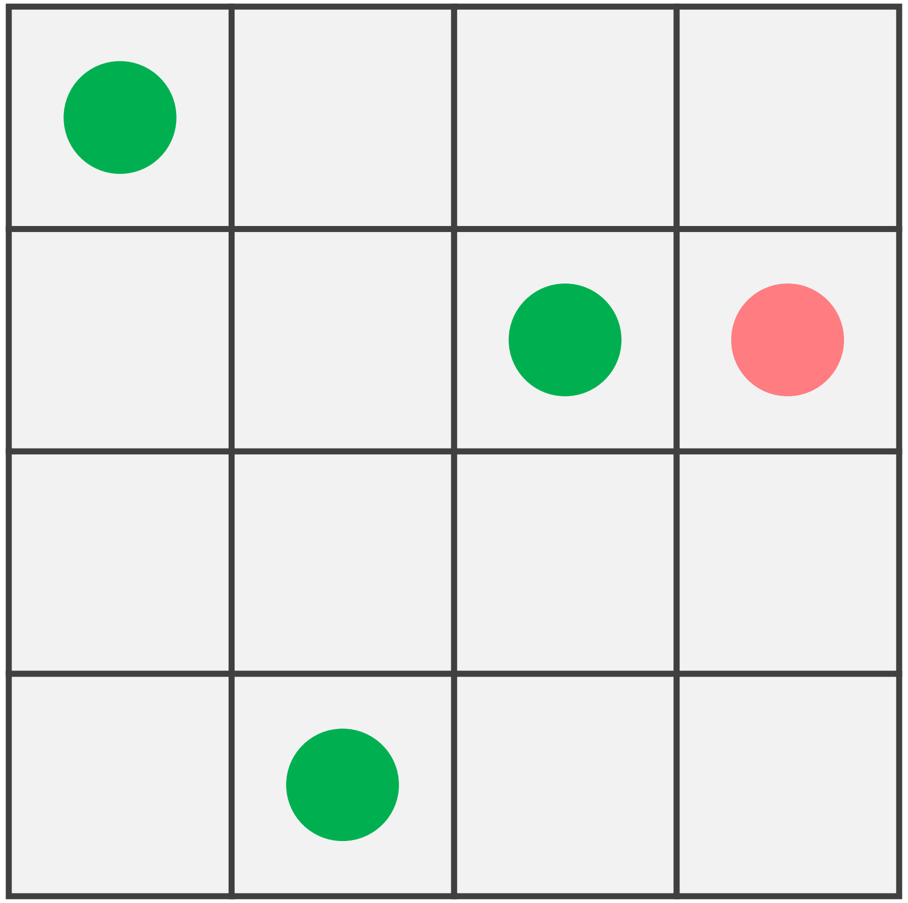
N-Queens Problem



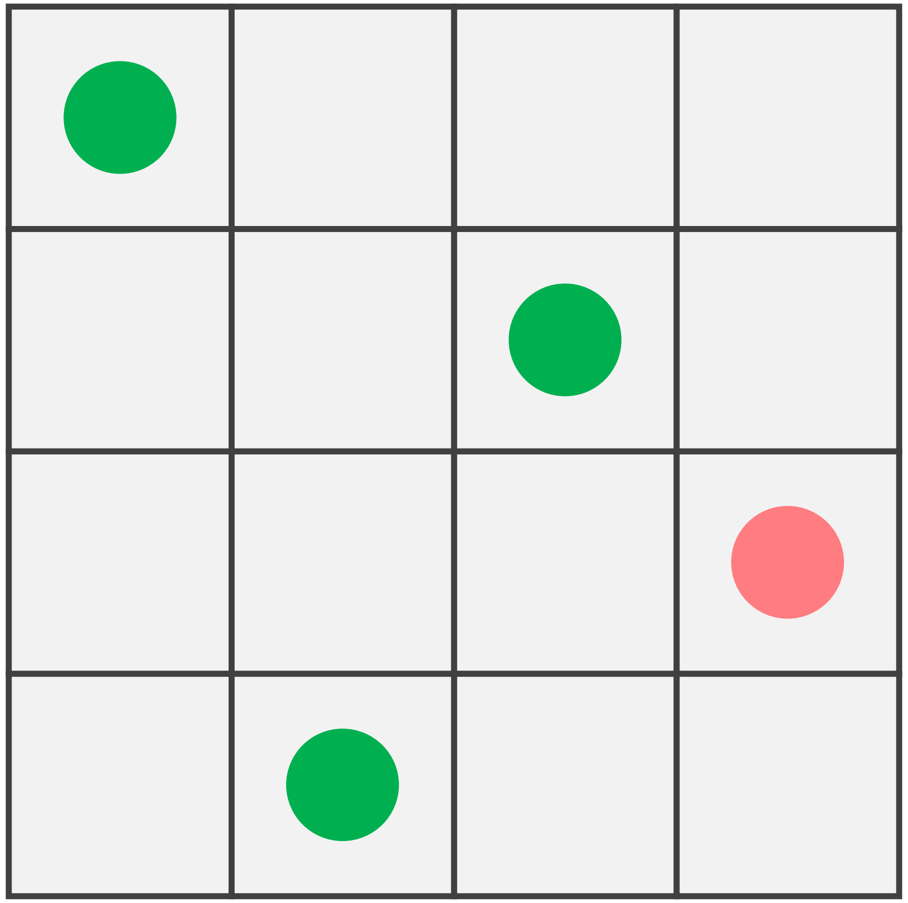
N-Queens Problem



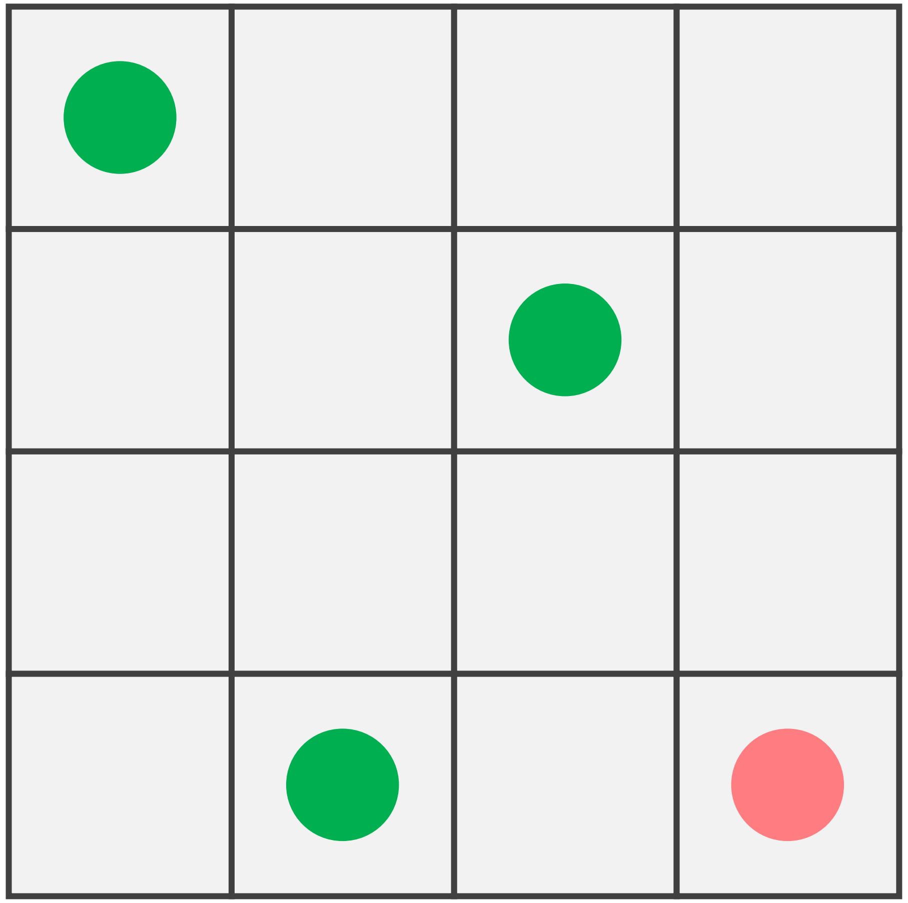
N-Queens Problem



N-Queens Problem

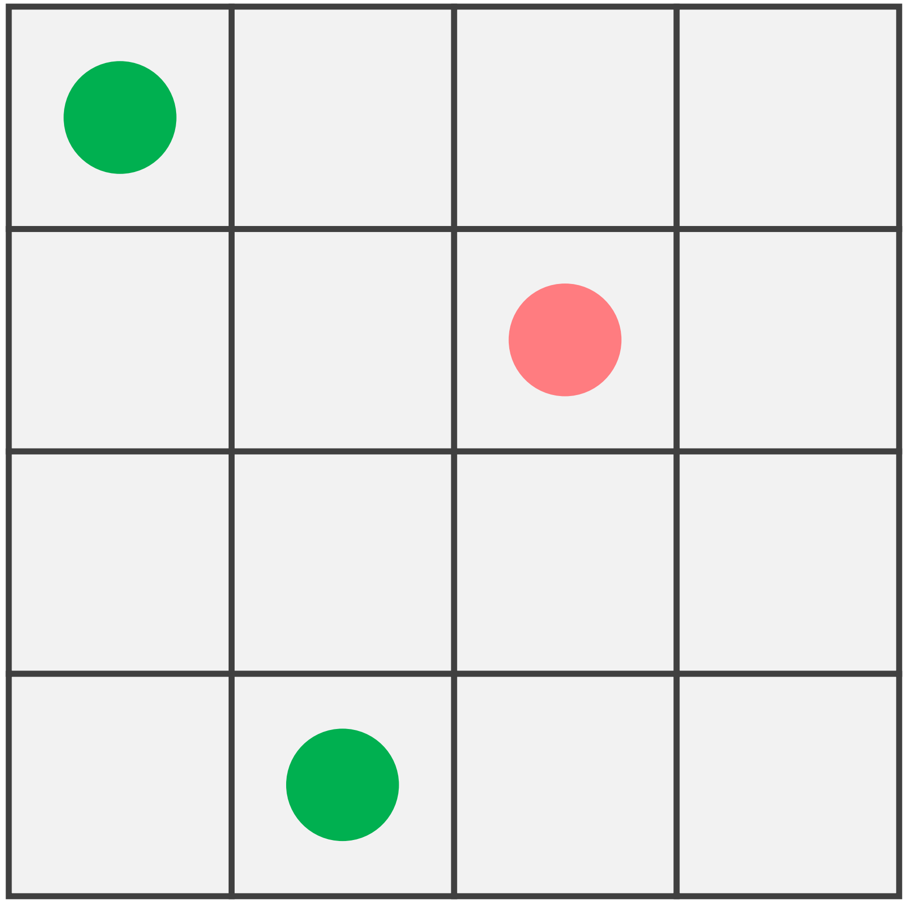


N-Queens Problem

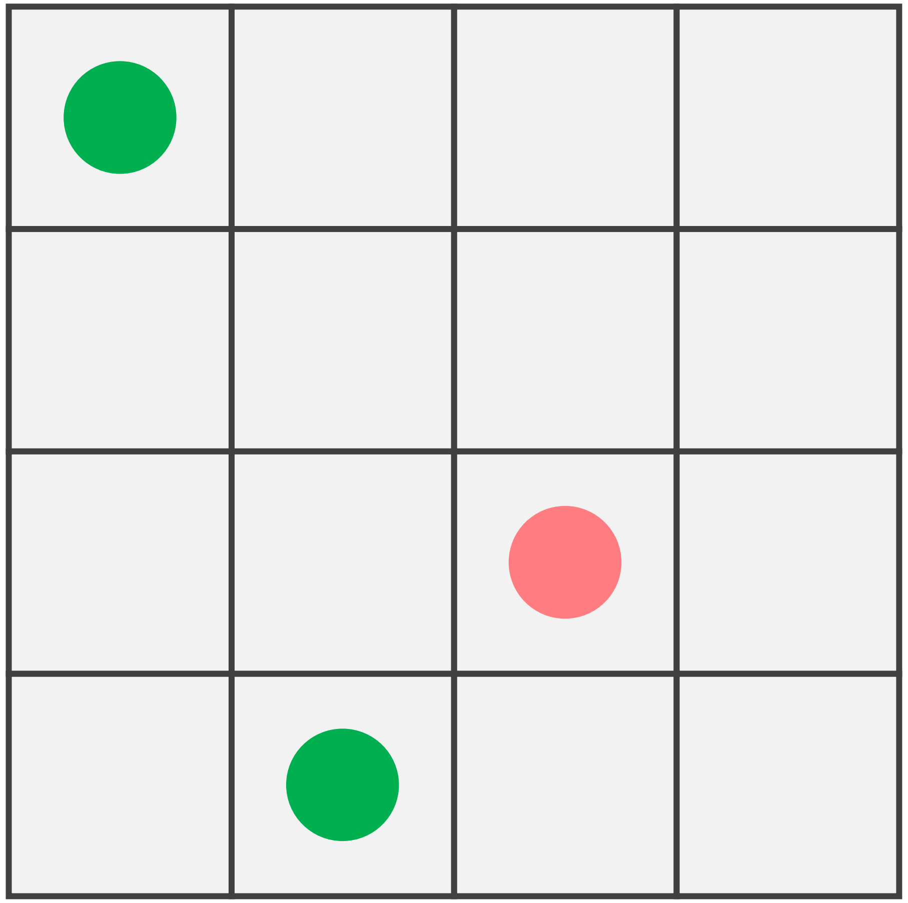


there is no feasible solution:
we have to step back to the
***previous column** and increment*
the position of the queen there
***BACKTRACKING** !!!*

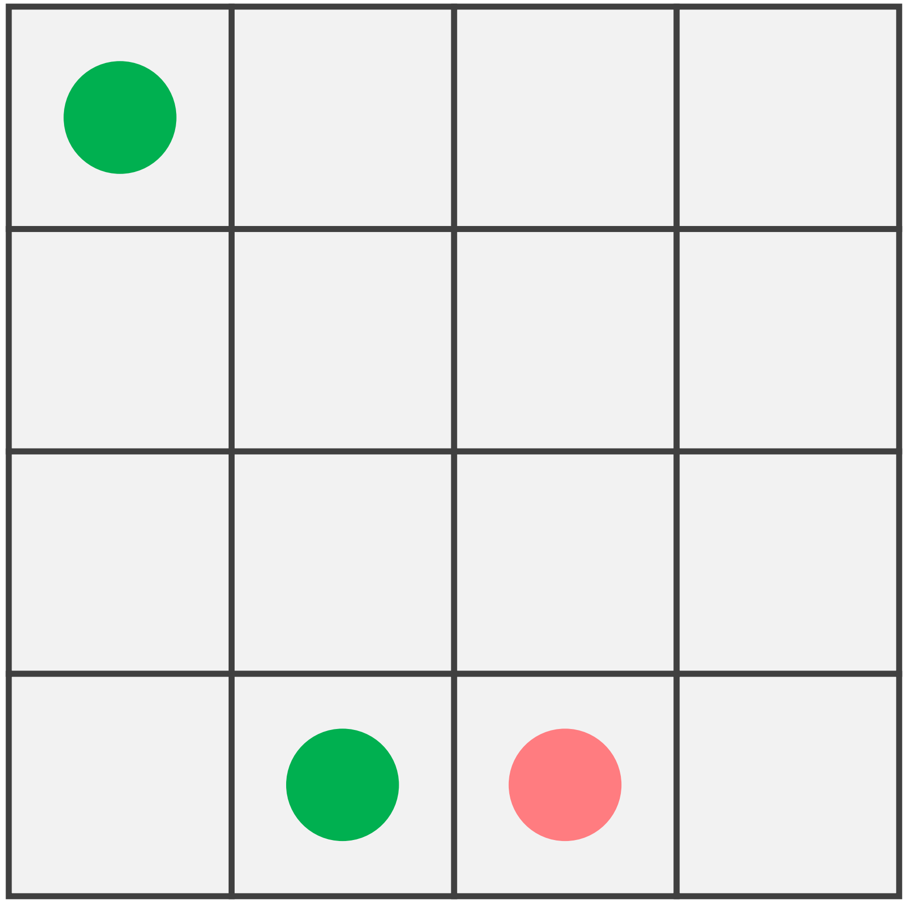
N-Queens Problem



N-Queens Problem

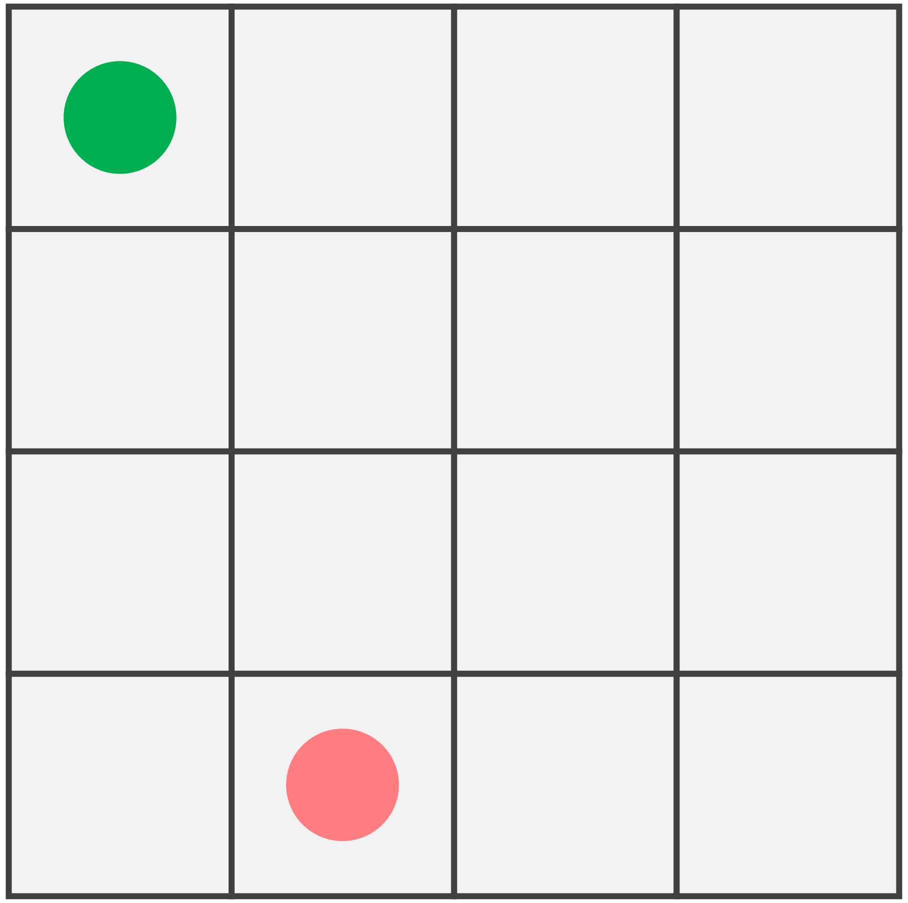


N-Queens Problem

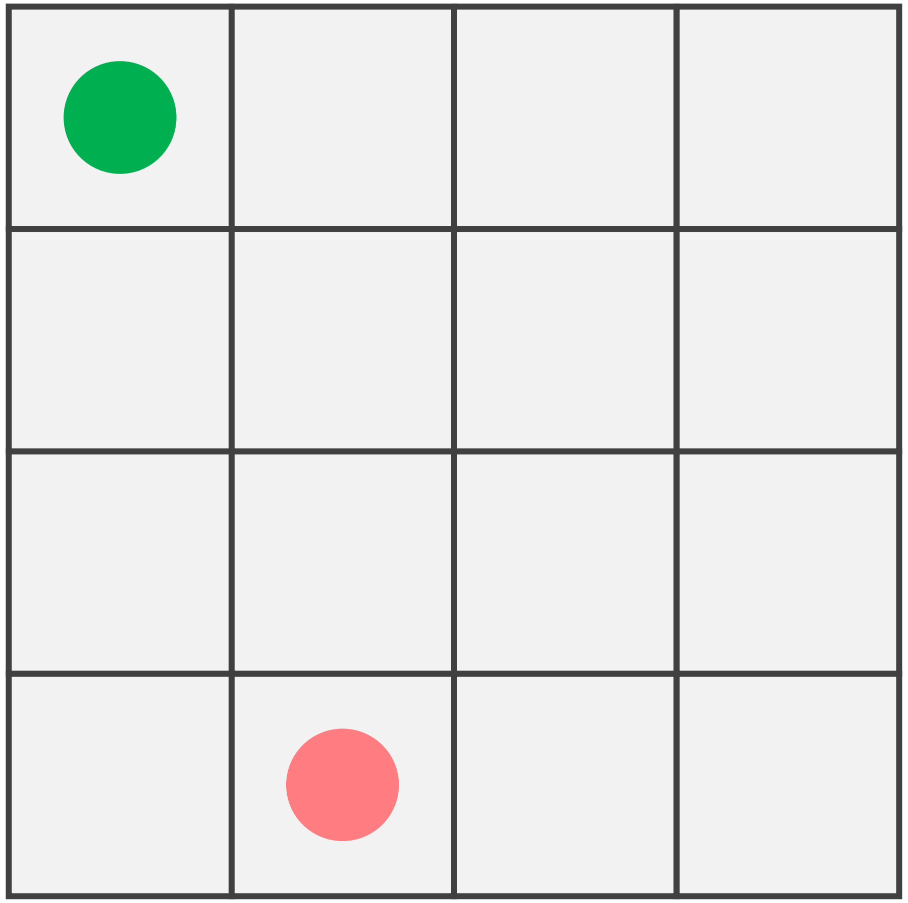


there is no feasible solution:
we have to step back to the
***previous column** and increment*
the position of the queen there
***BACKTRACKING** !!!*

N-Queens Problem

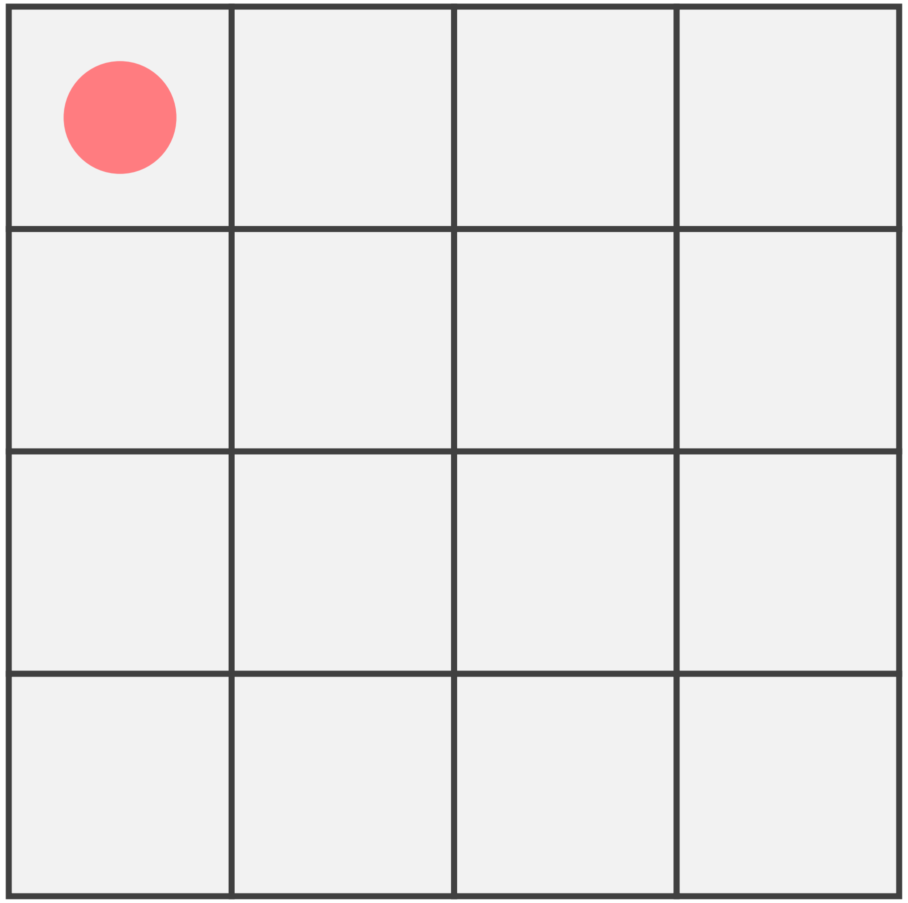


N-Queens Problem

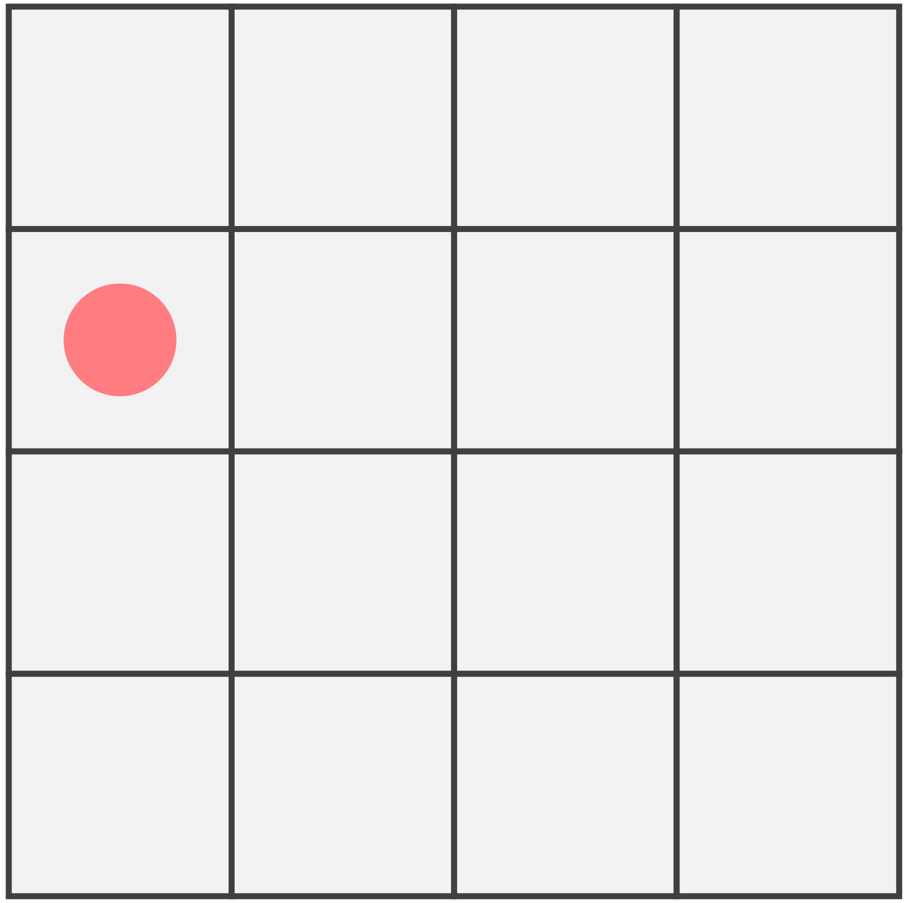


*there is no feasible solution: we have to step back to the **previous column** and increment the position of the queen there **BACKTRACKING** !!!*

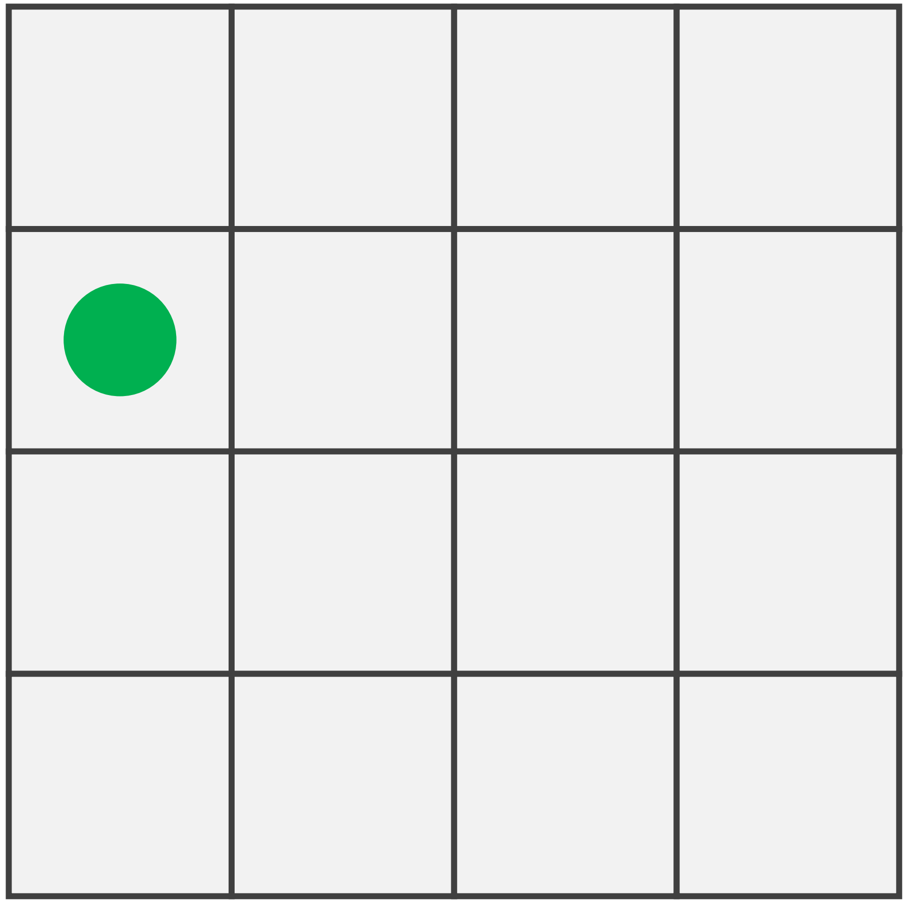
N-Queens Problem



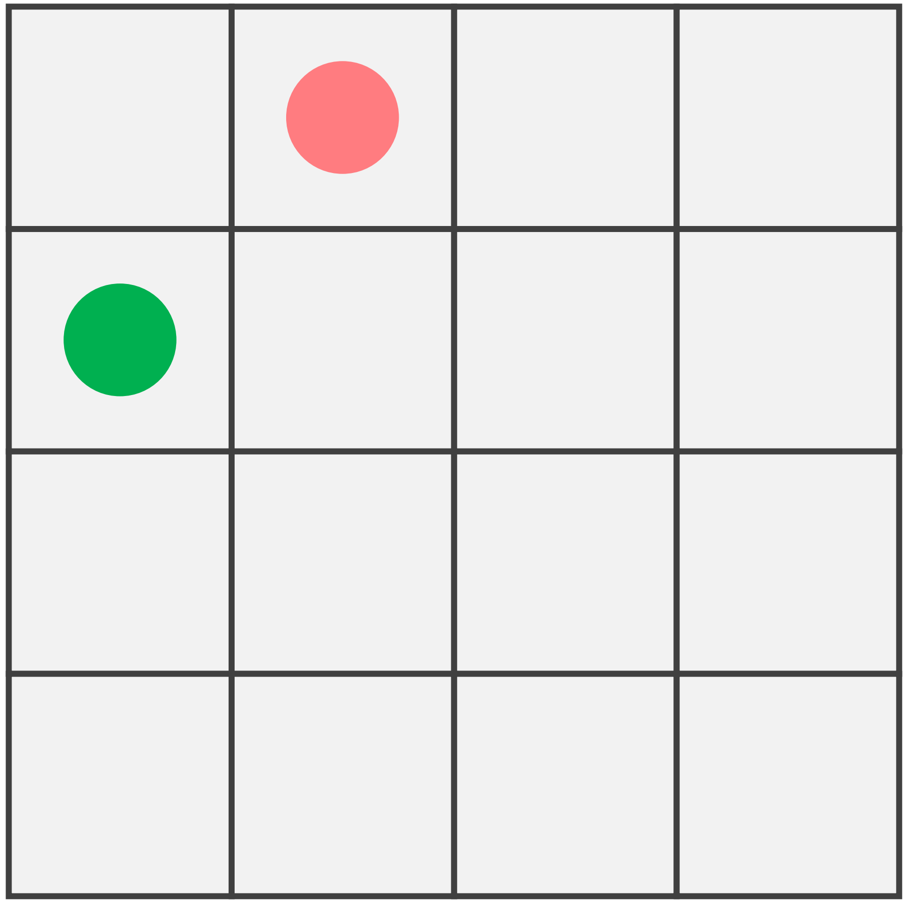
N-Queens Problem



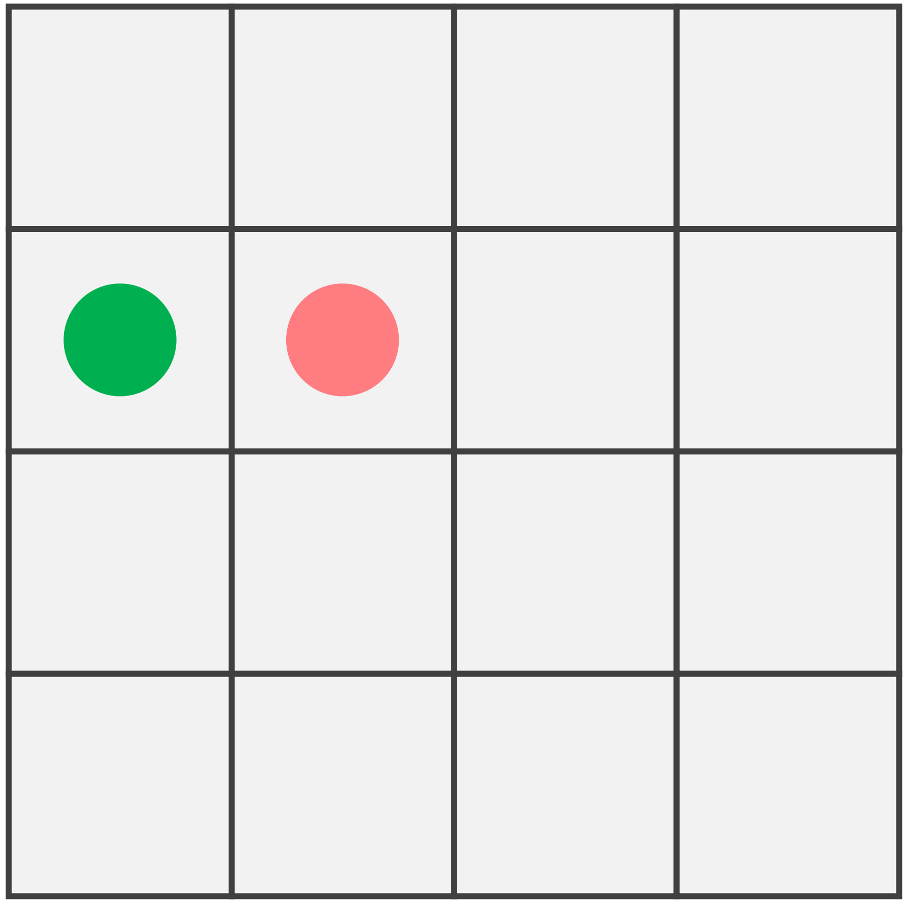
N-Queens Problem



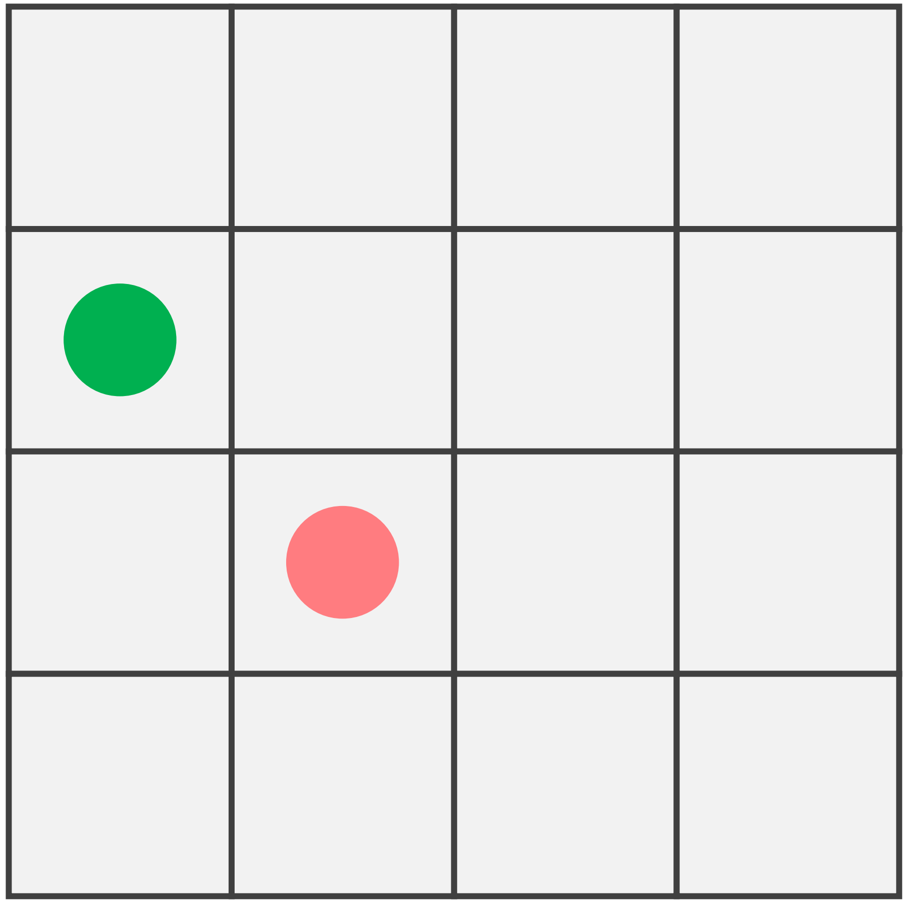
N-Queens Problem



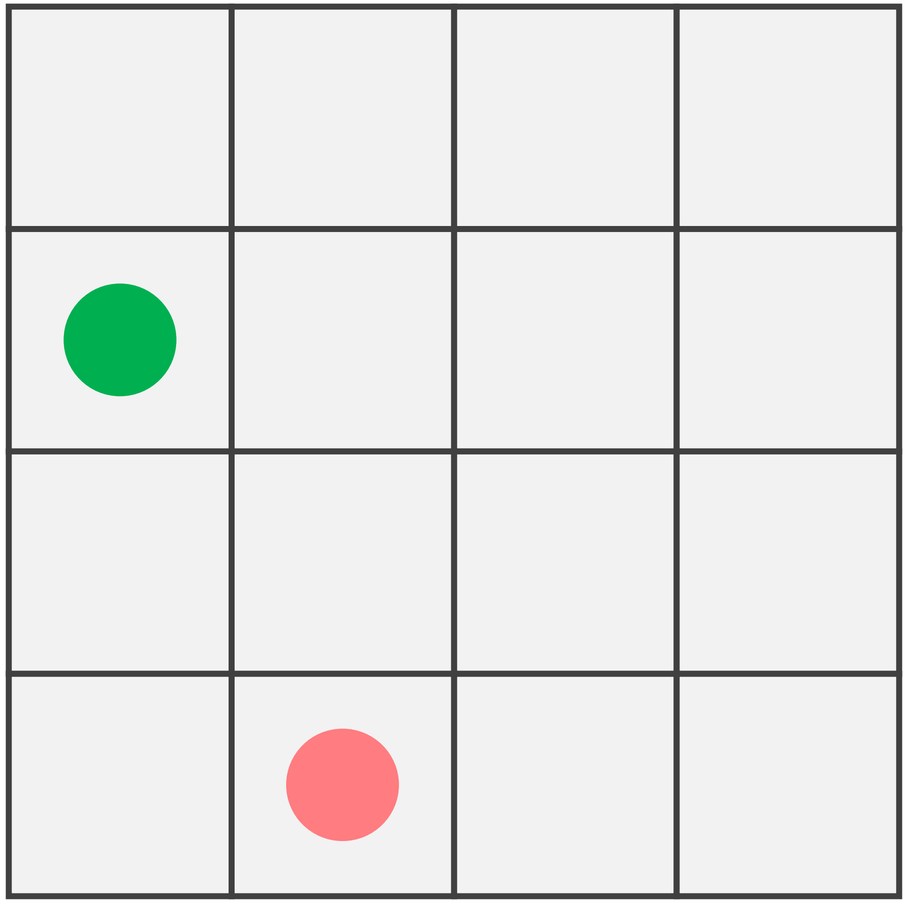
N-Queens Problem



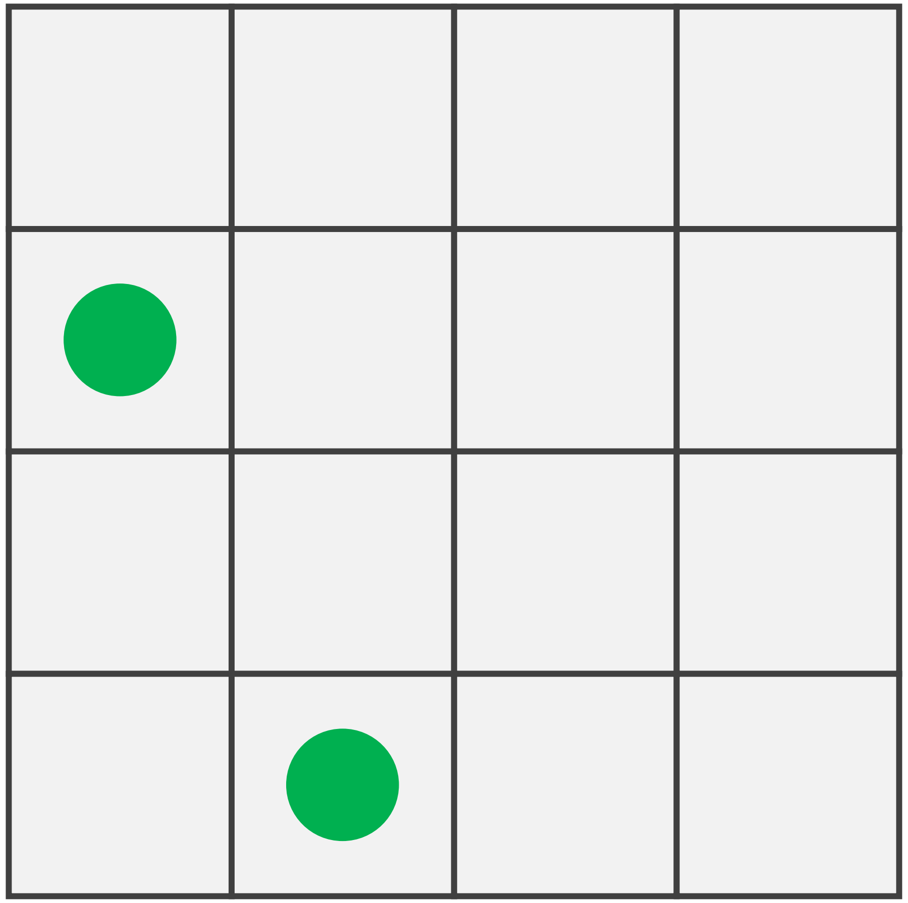
N-Queens Problem



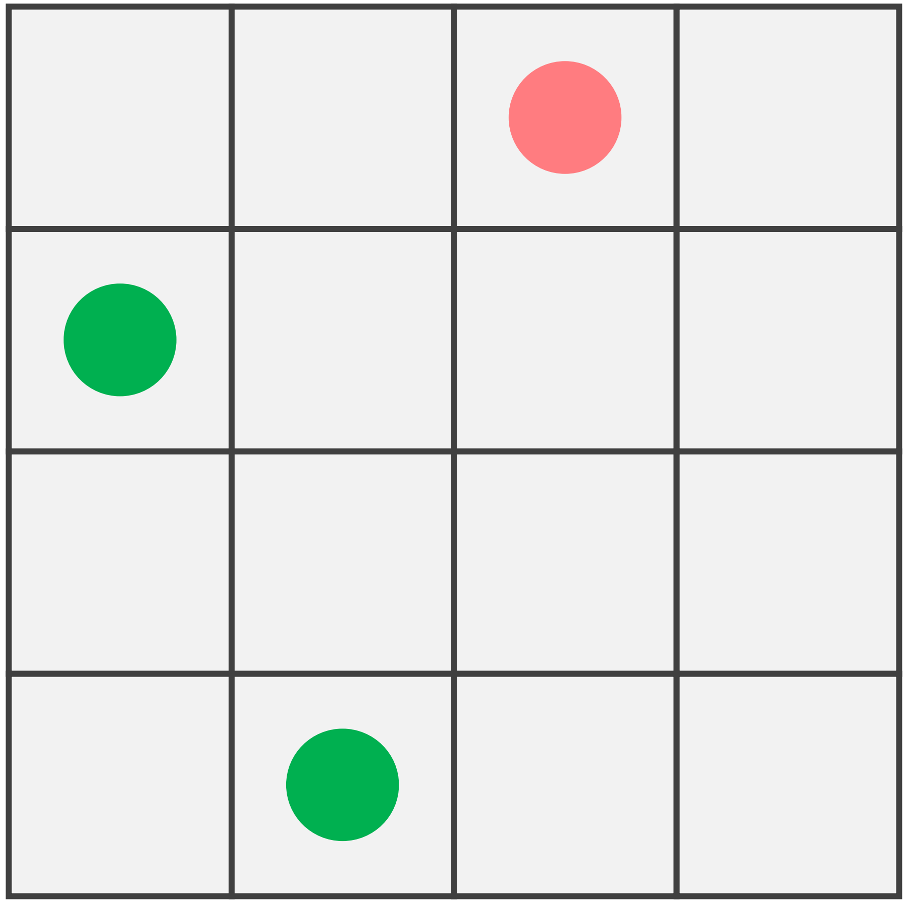
N-Queens Problem



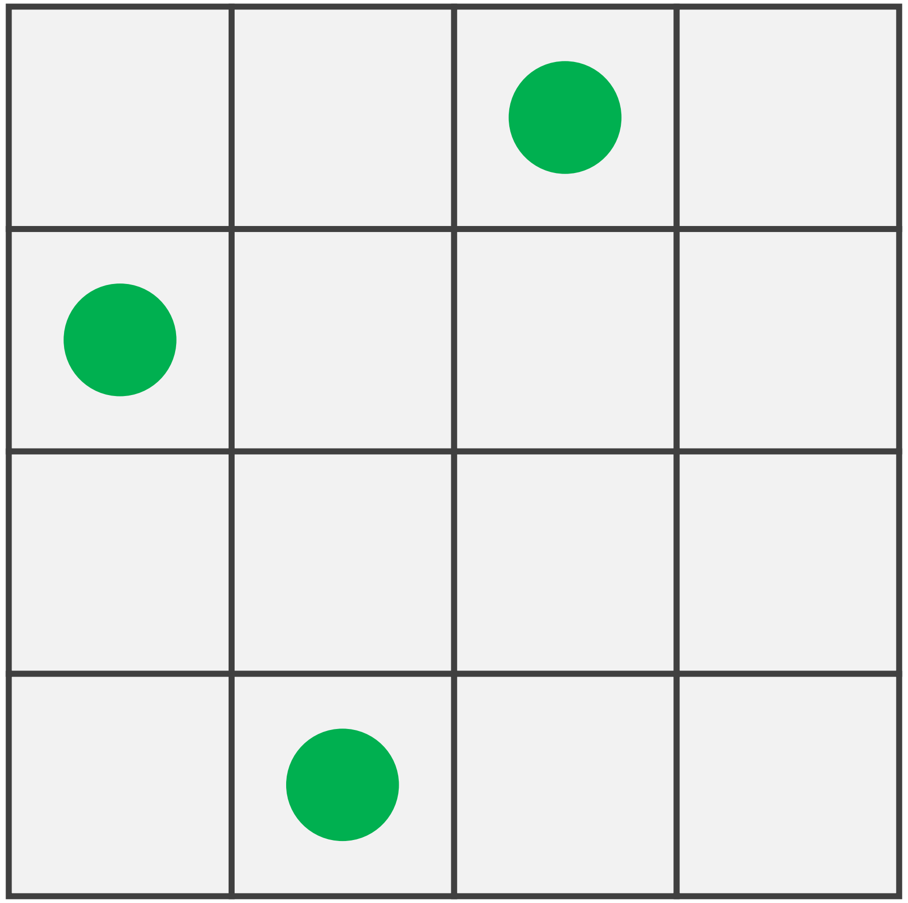
N-Queens Problem



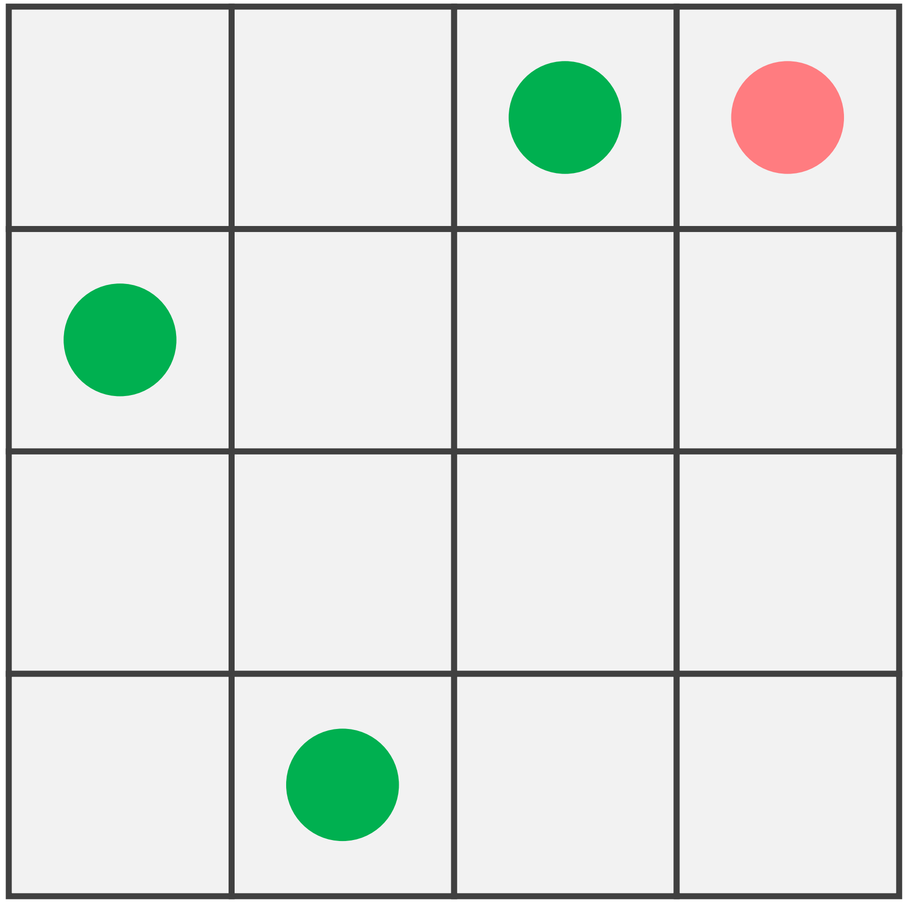
N-Queens Problem



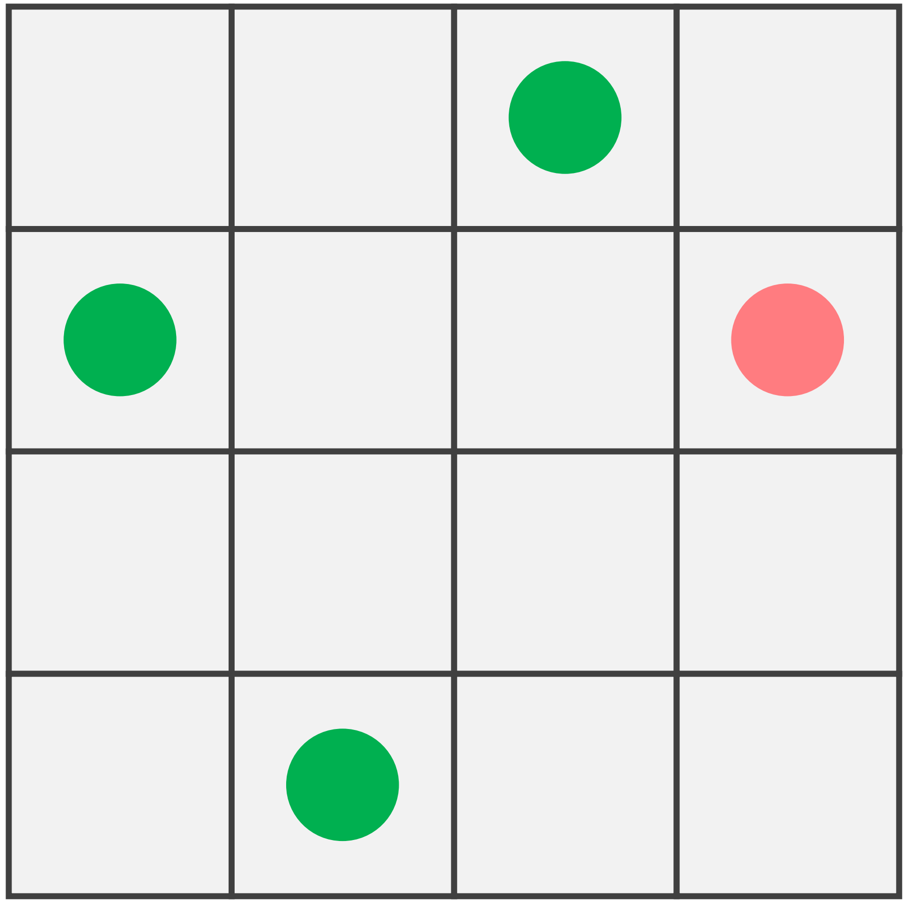
N-Queens Problem



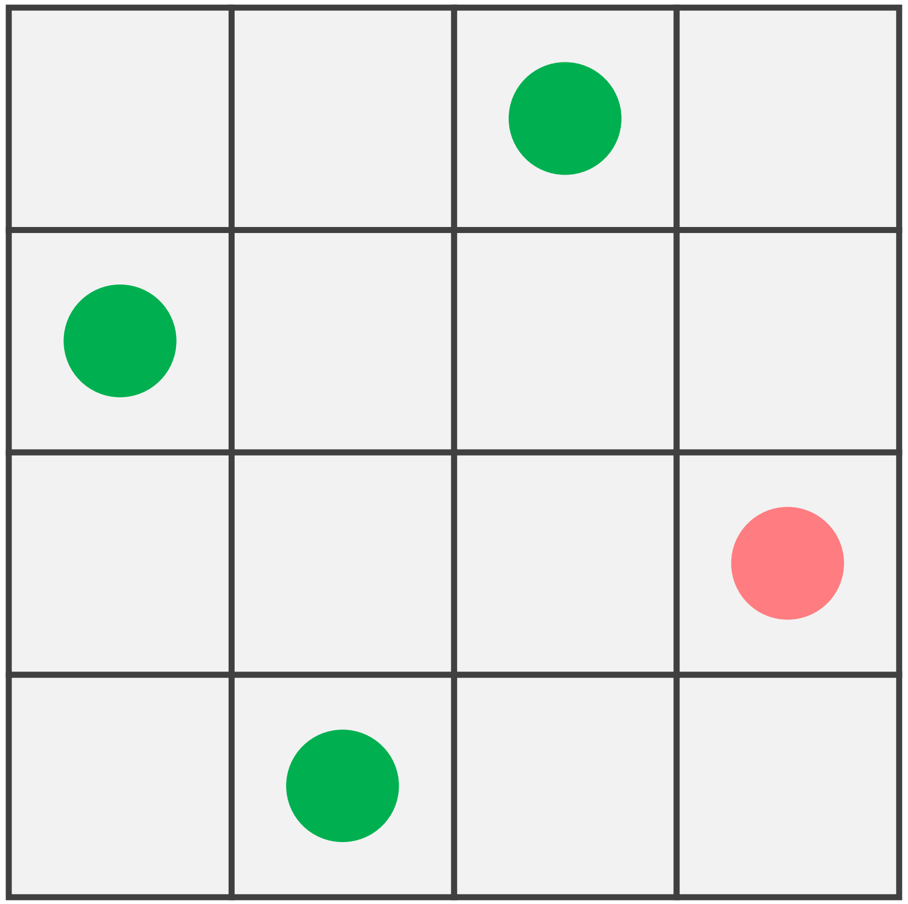
N-Queens Problem



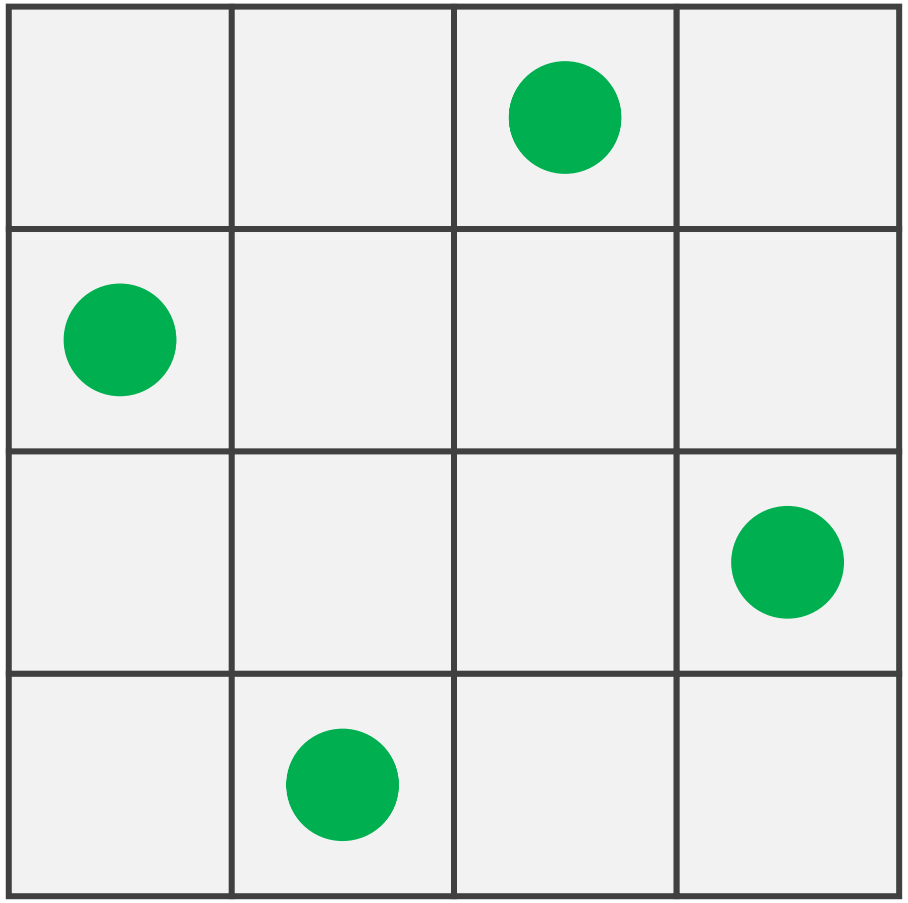
N-Queens Problem



N-Queens Problem

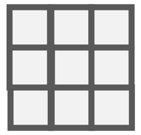


N-Queens Problem



N-Queens Problem
Search Tree Visualization
(Algorithmic Problems)

N-Queens Problem



ROOT

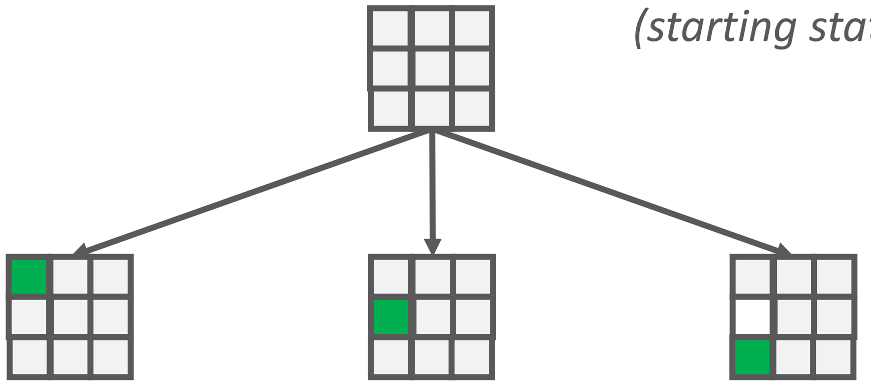
this is the empty state

(starting state)

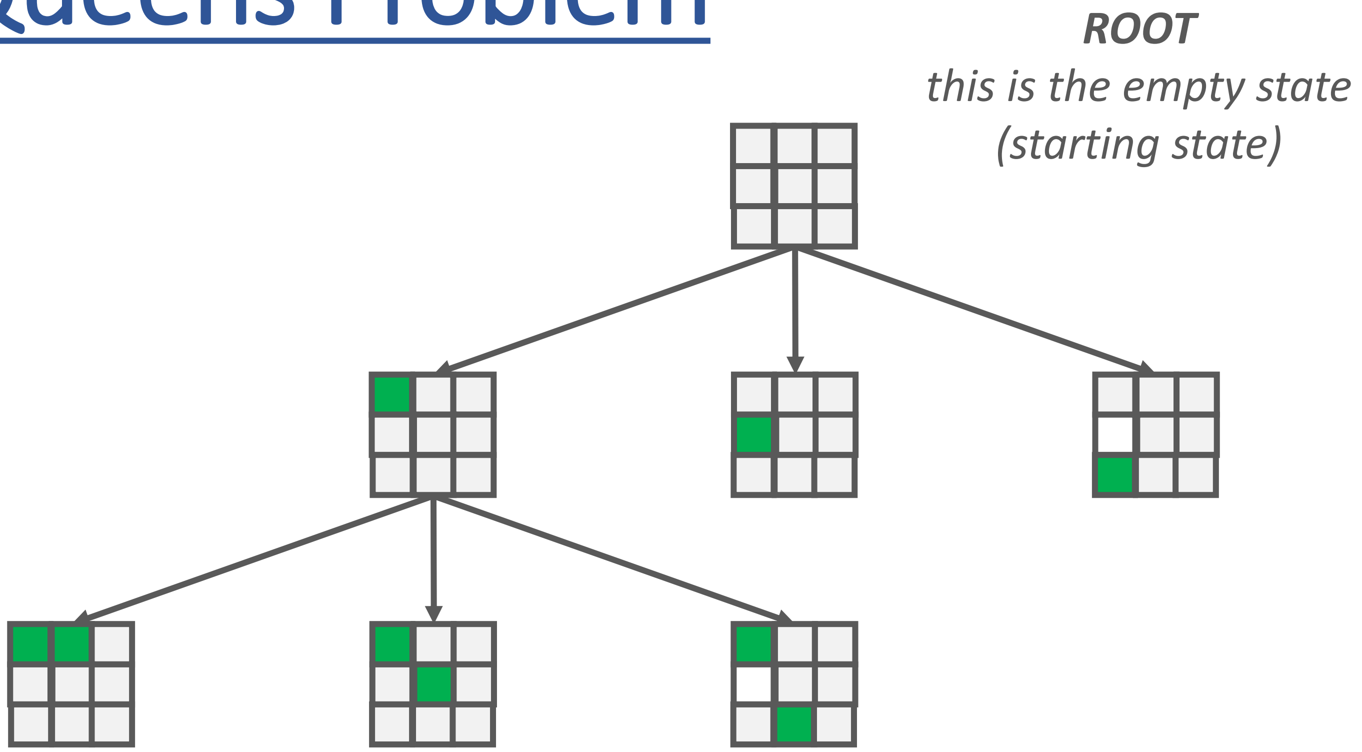
N-Queens Problem

ROOT

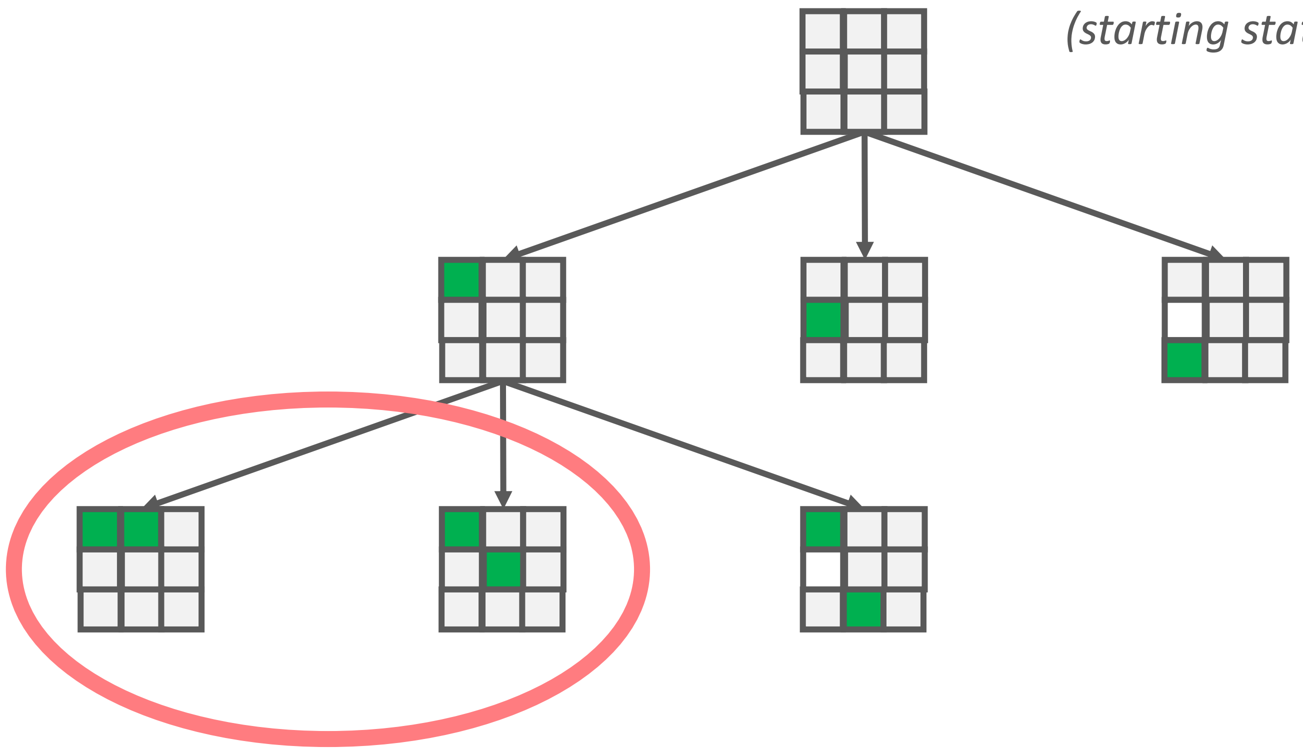
this is the empty state
(starting state)



N-Queens Problem



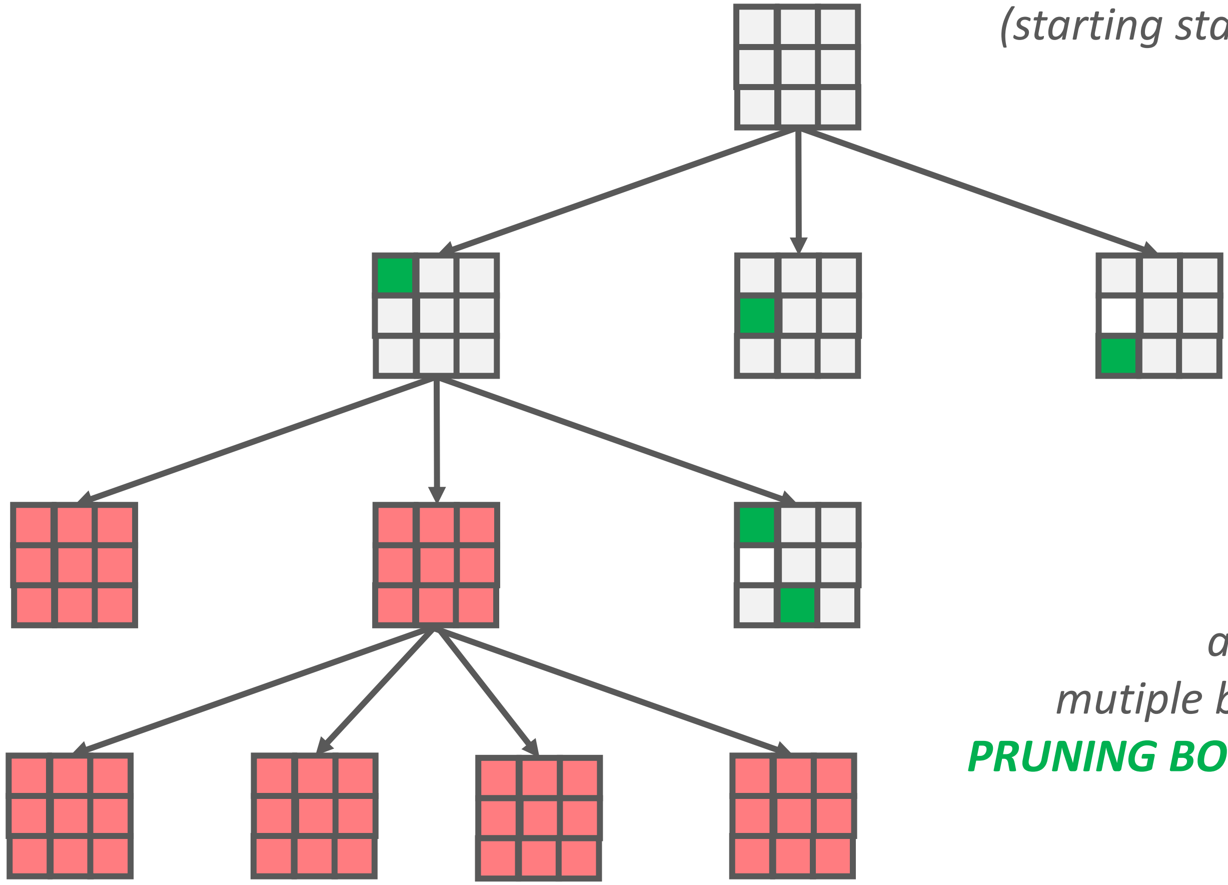
N-Queens Problem



ROOT
this is the empty state
(starting state)

we can prune these branches
as we know for certain
that the queens can attack each other

N-Queens Problem

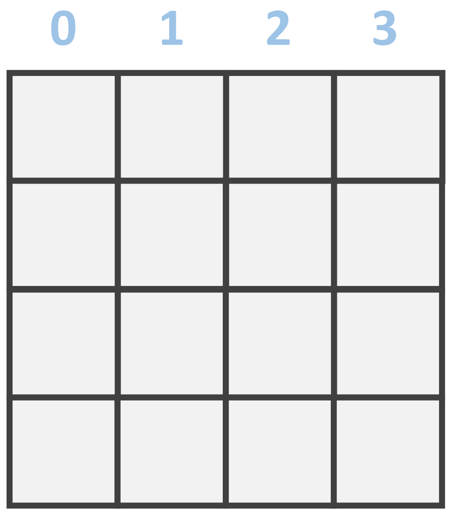


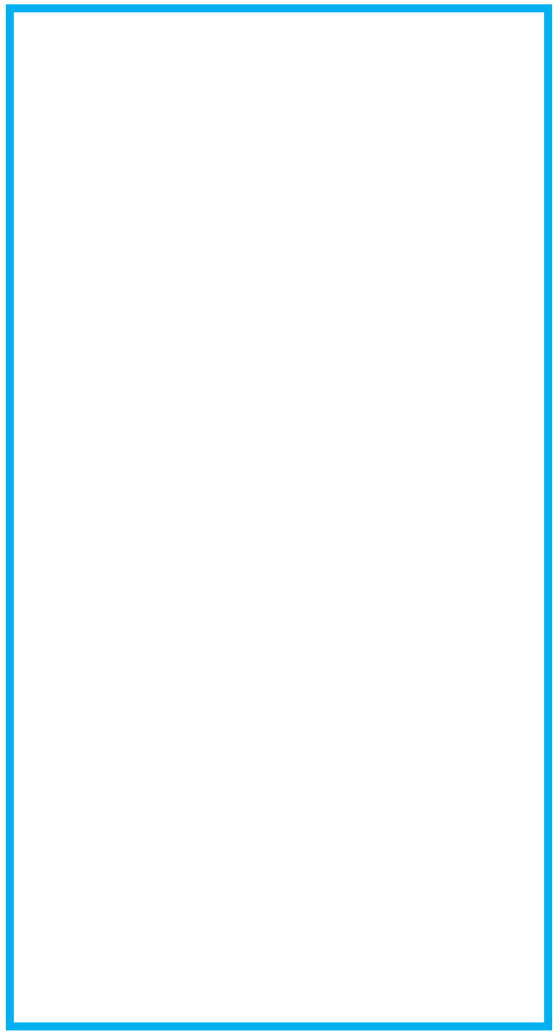
ROOT
this is the empty state
(starting state)

we can prune these branches
and beause of that we can discard
mutiple bad states (subtrees) in the same iteration
PRUNING BOOSTS THE RUNNING TIME OF BACKTRACKING

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

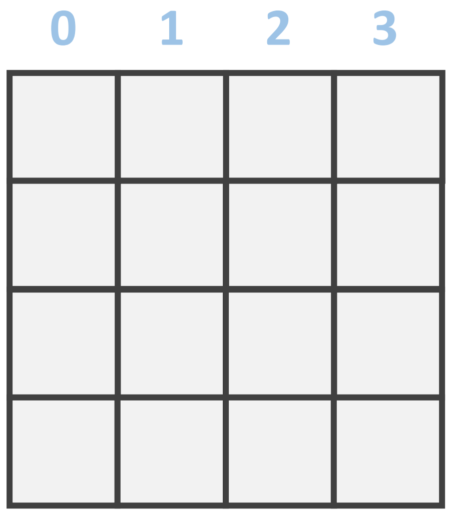


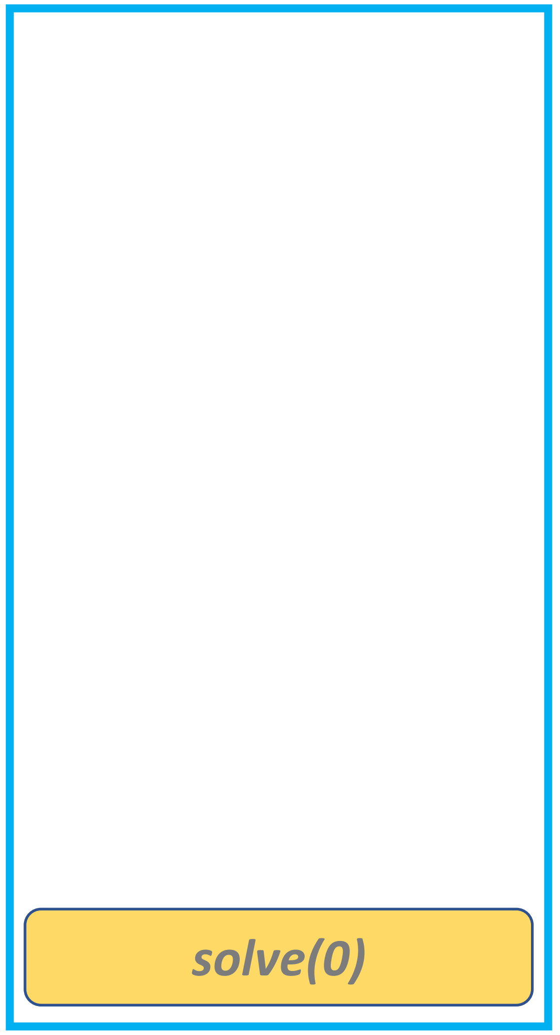


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

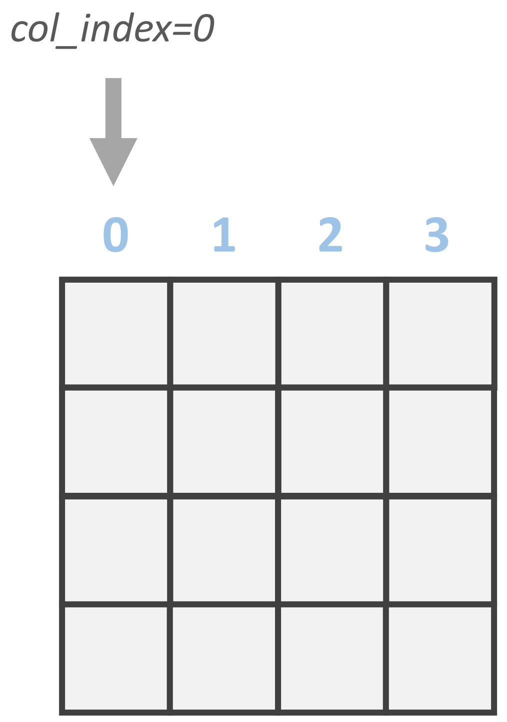


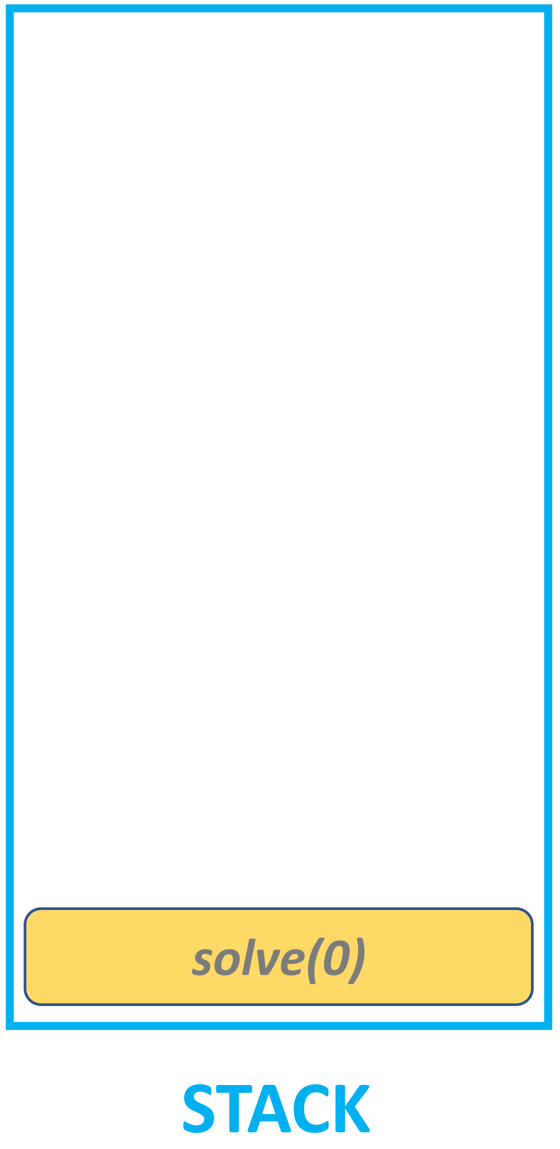


STACK

N-Queens Problem

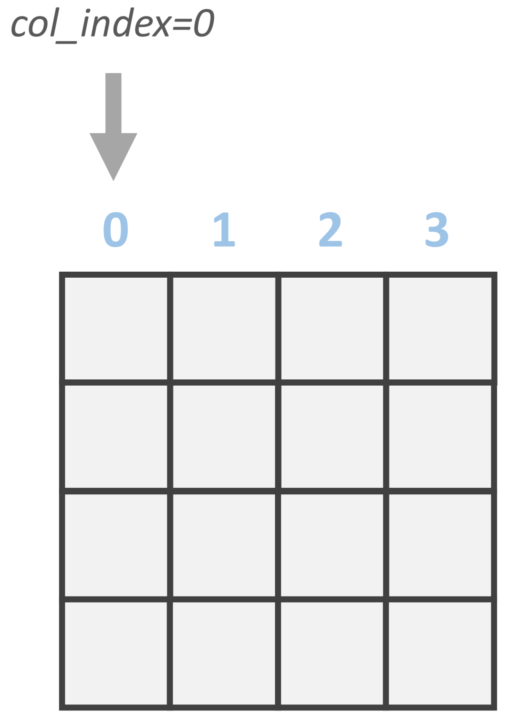
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

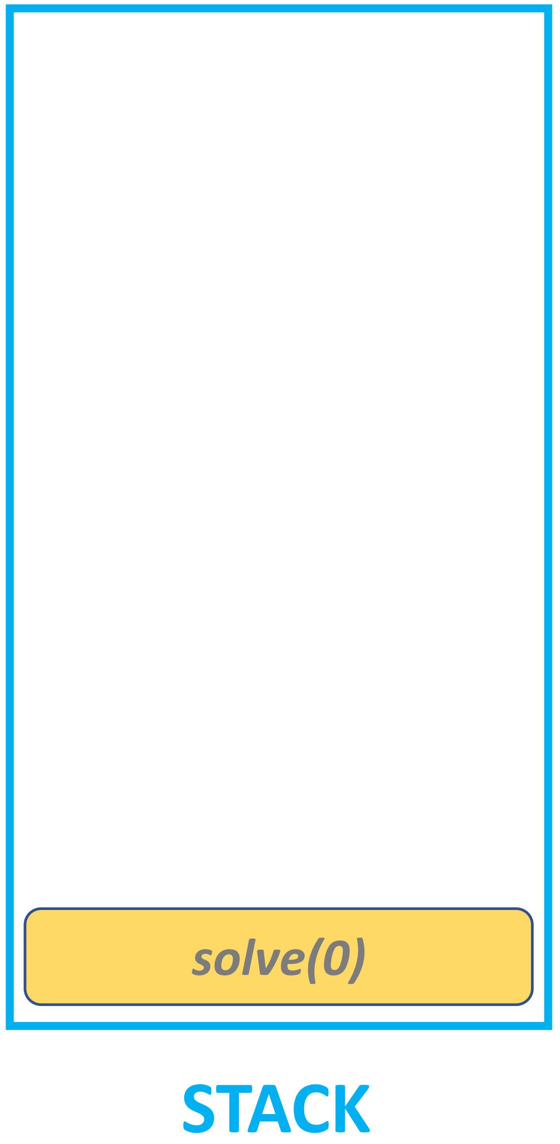




N-Queens Problem

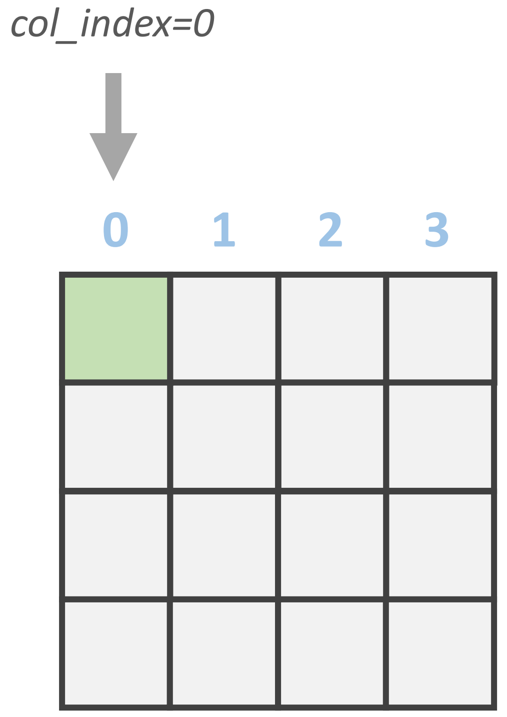
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

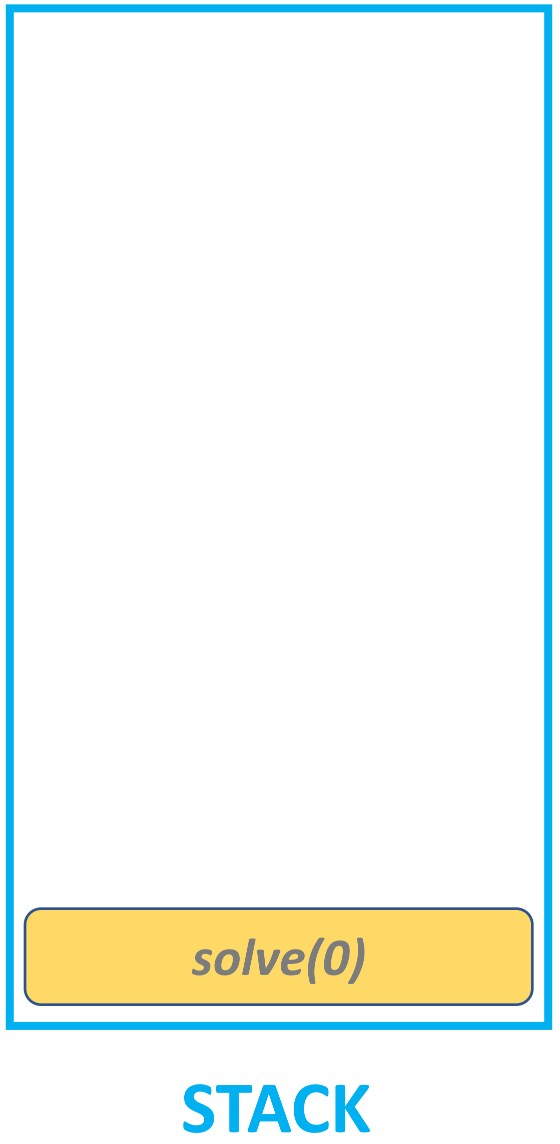




N-Queens Problem

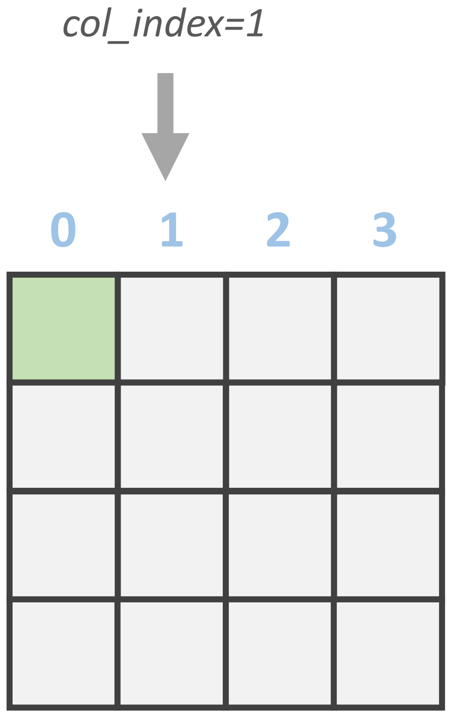
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

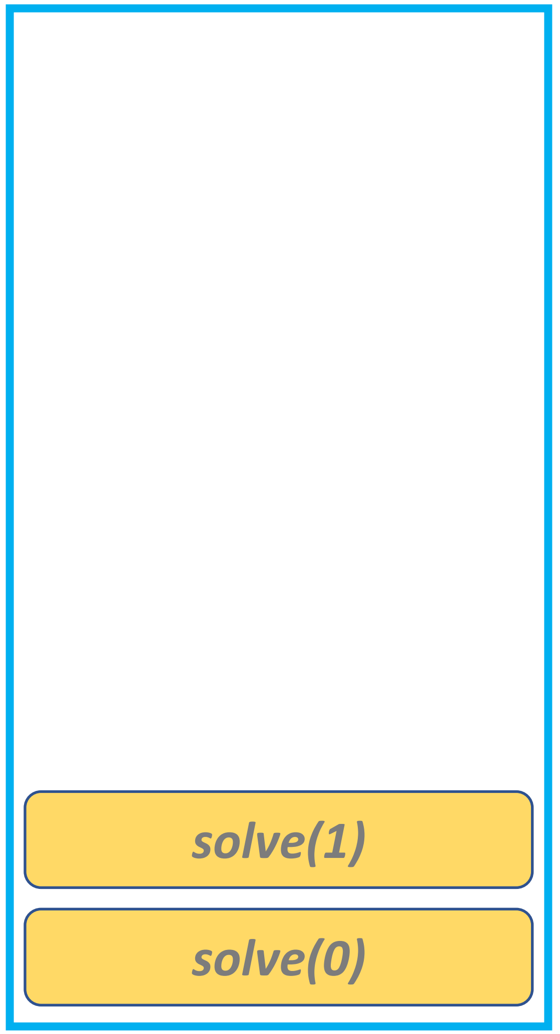




N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

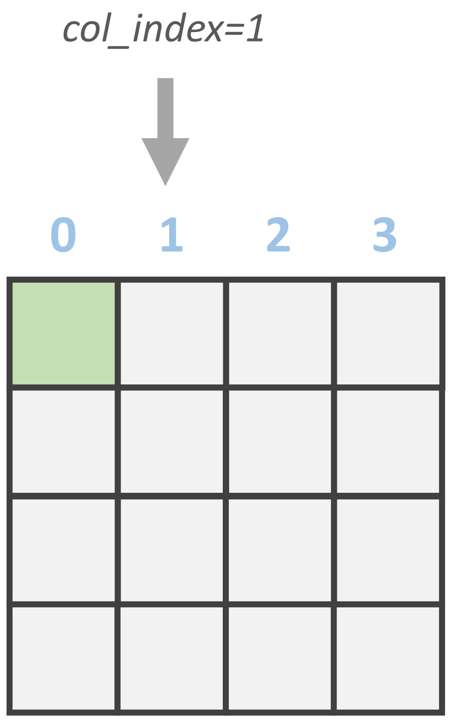


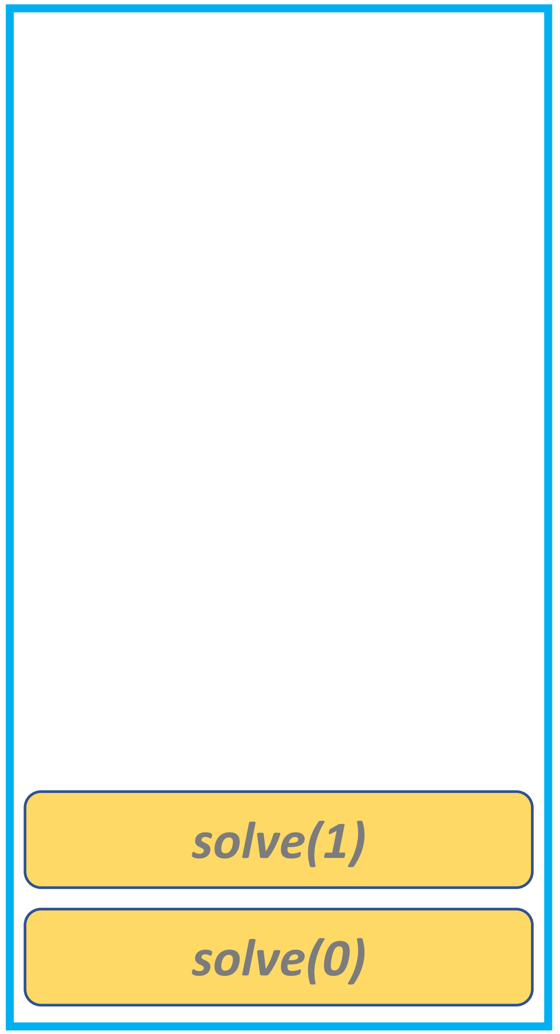


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

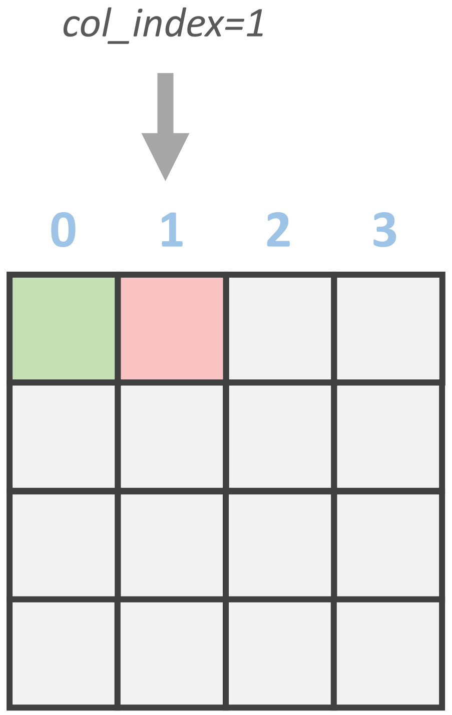


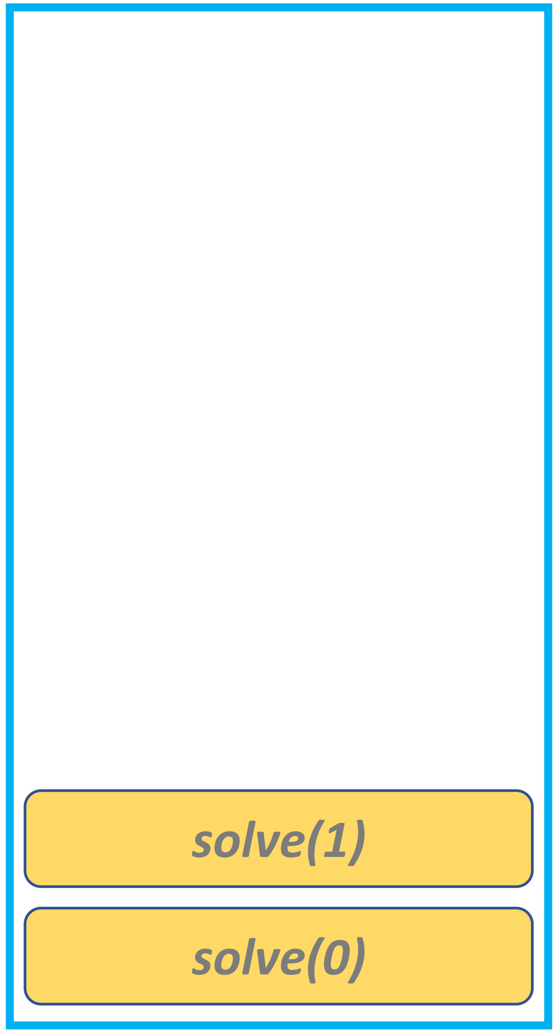


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

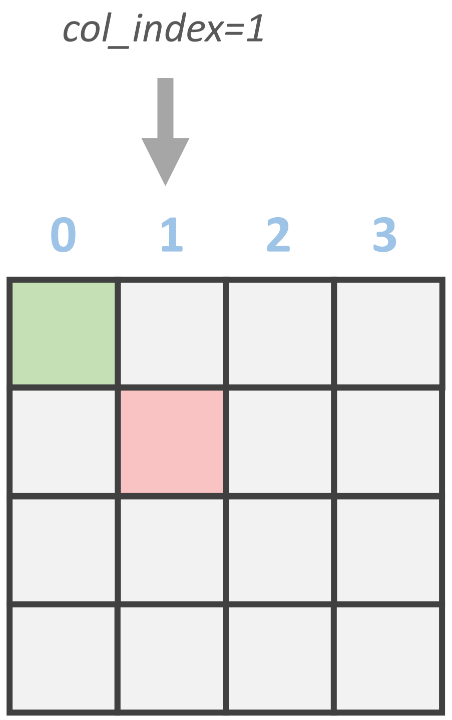


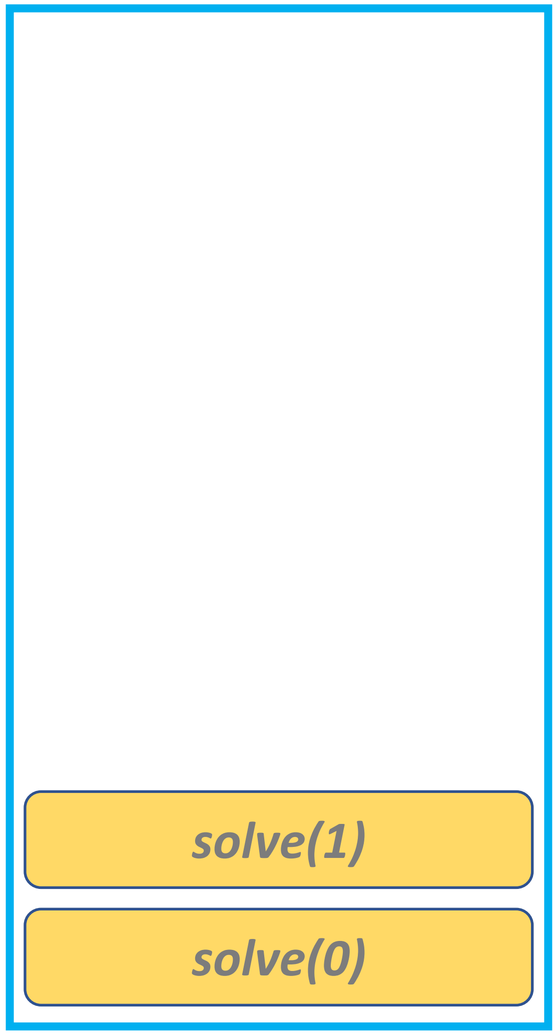


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

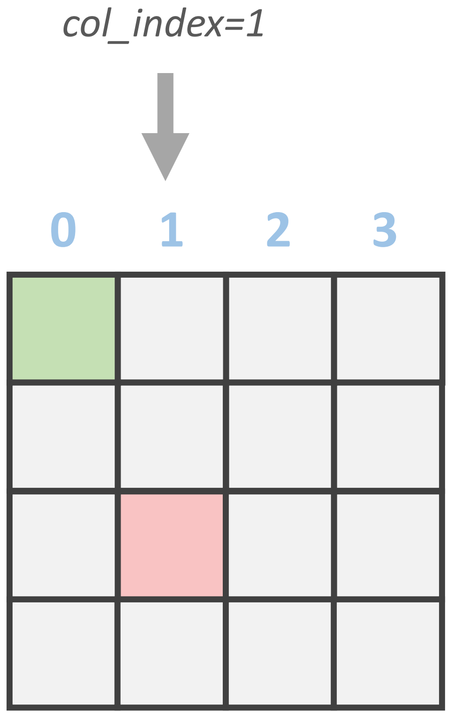


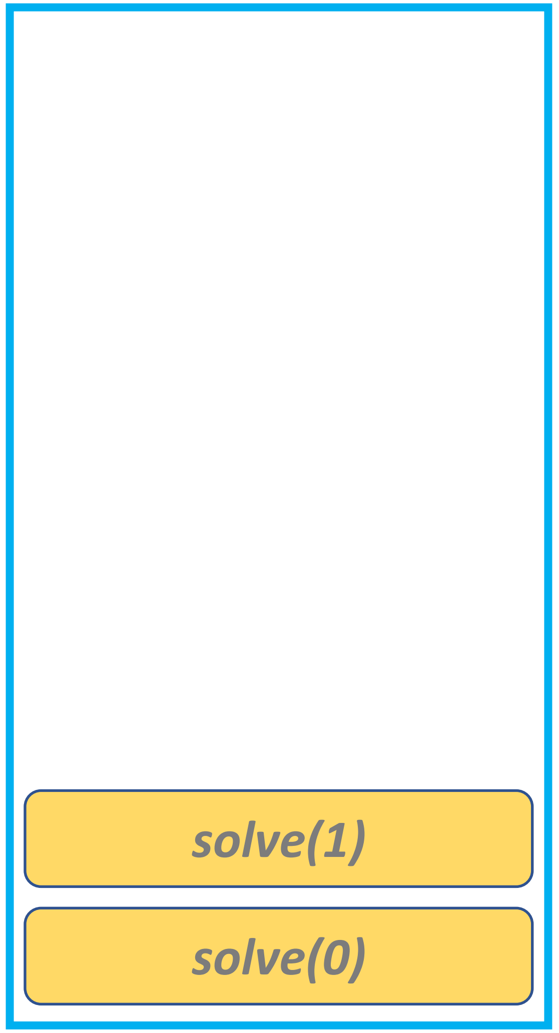


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

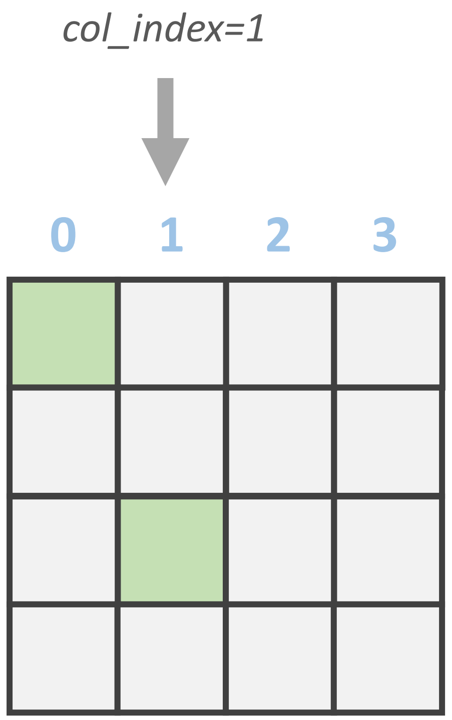


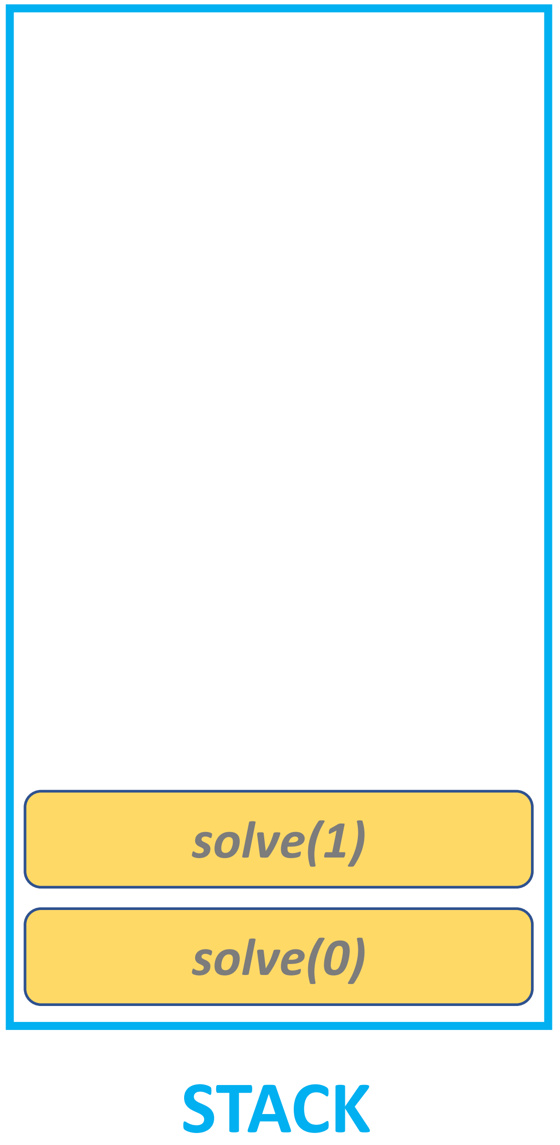


STACK

N-Queens Problem

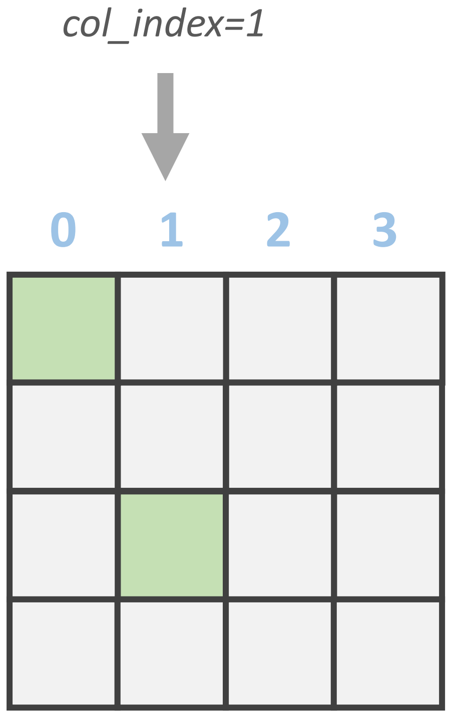
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

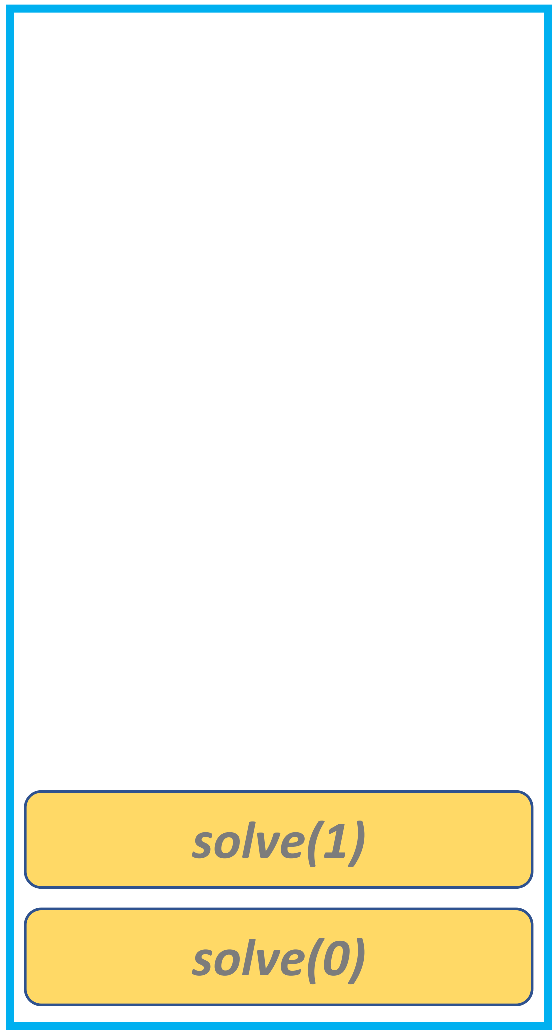




N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

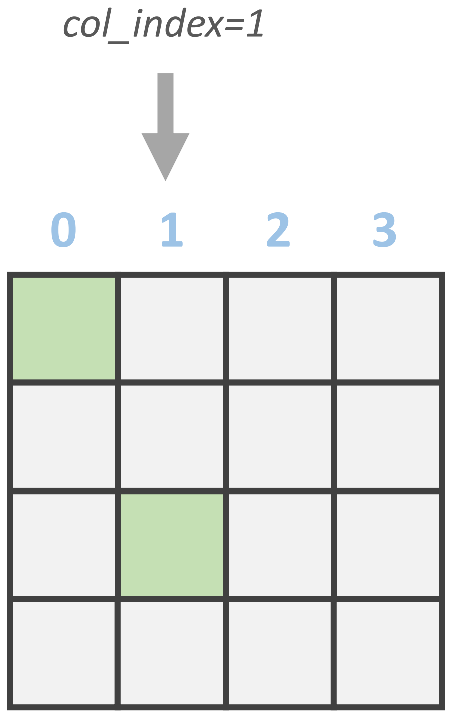


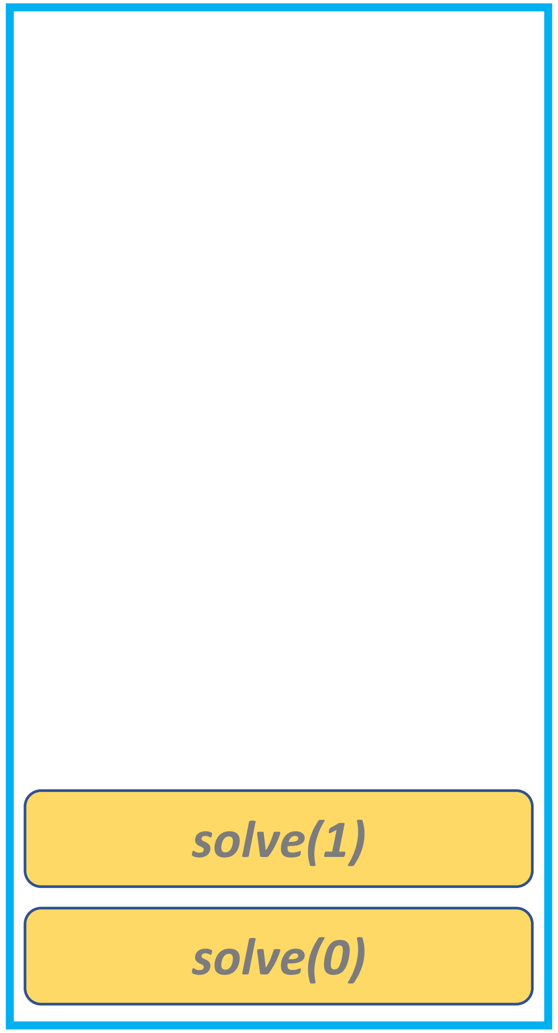


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

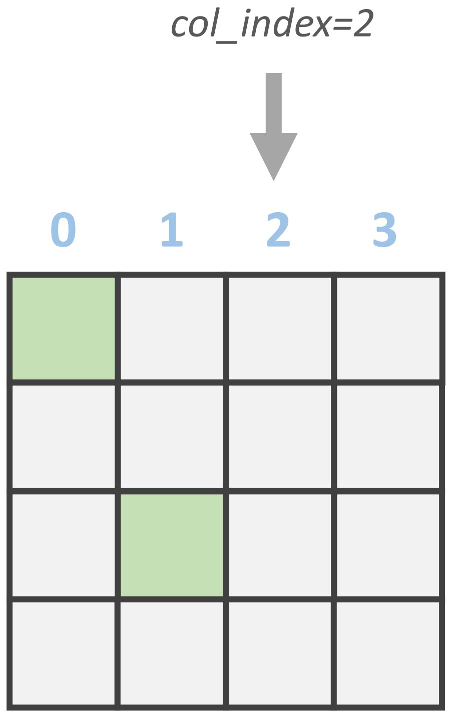


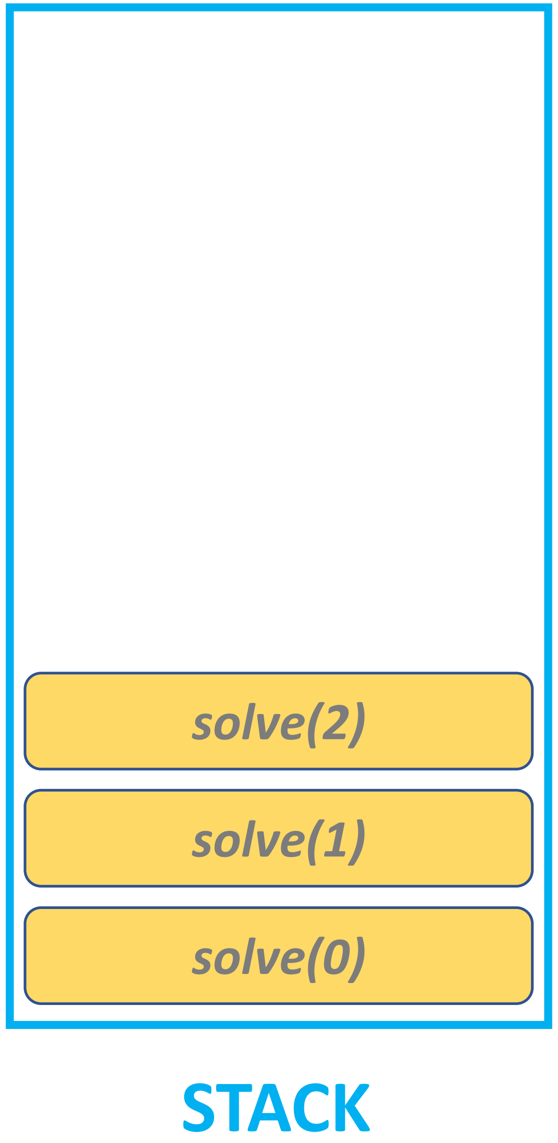


STACK

N-Queens Problem

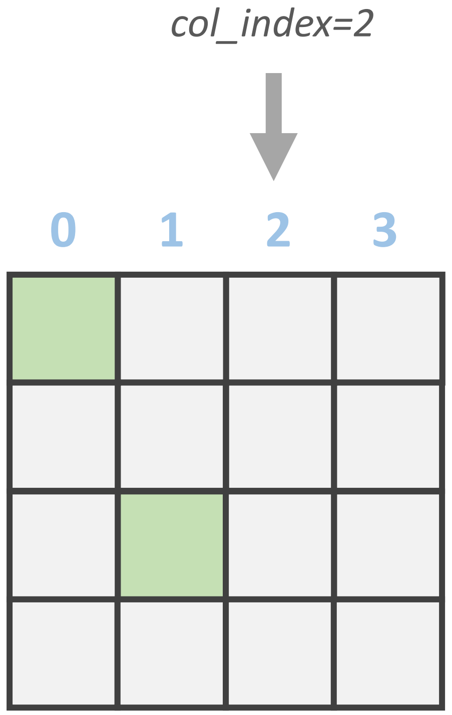
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

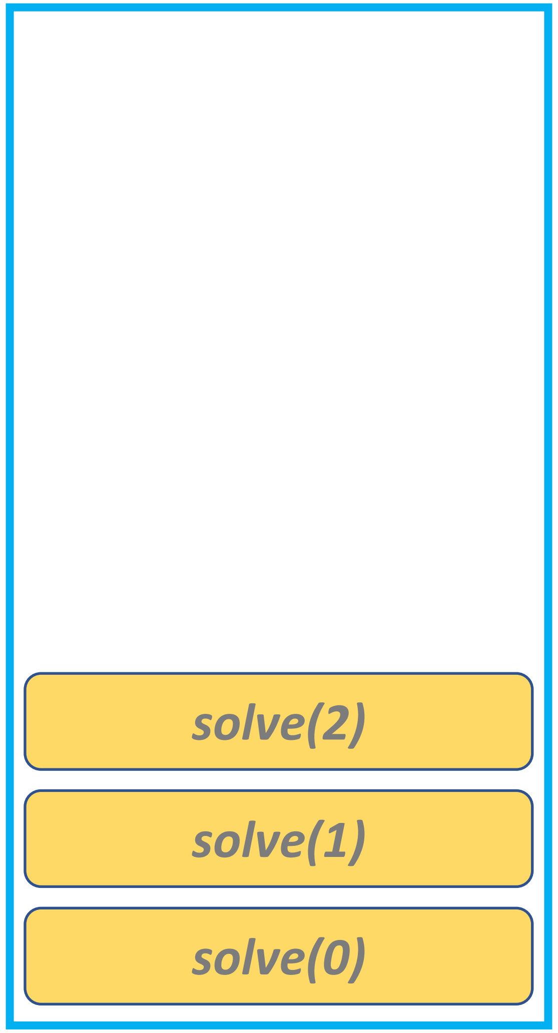




N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

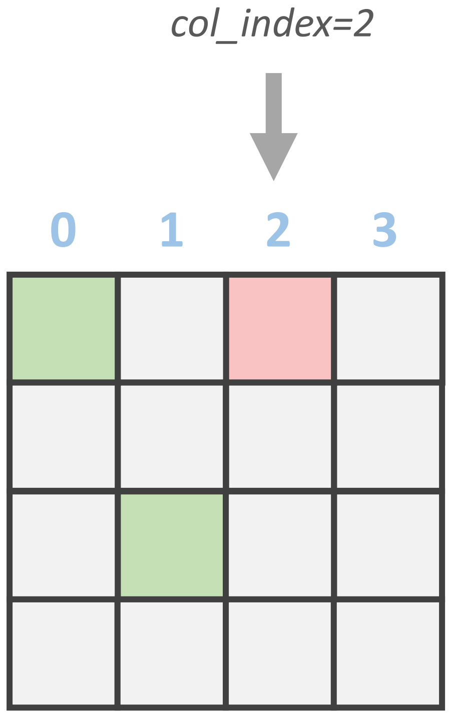


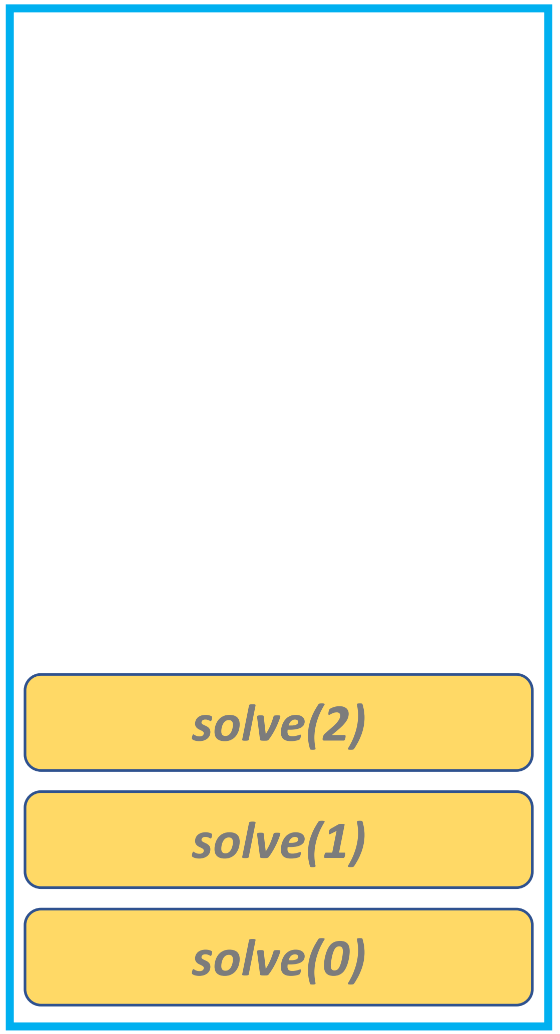


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

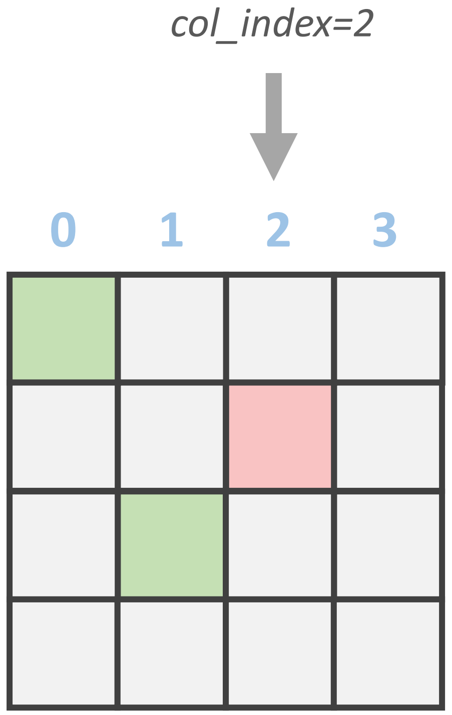


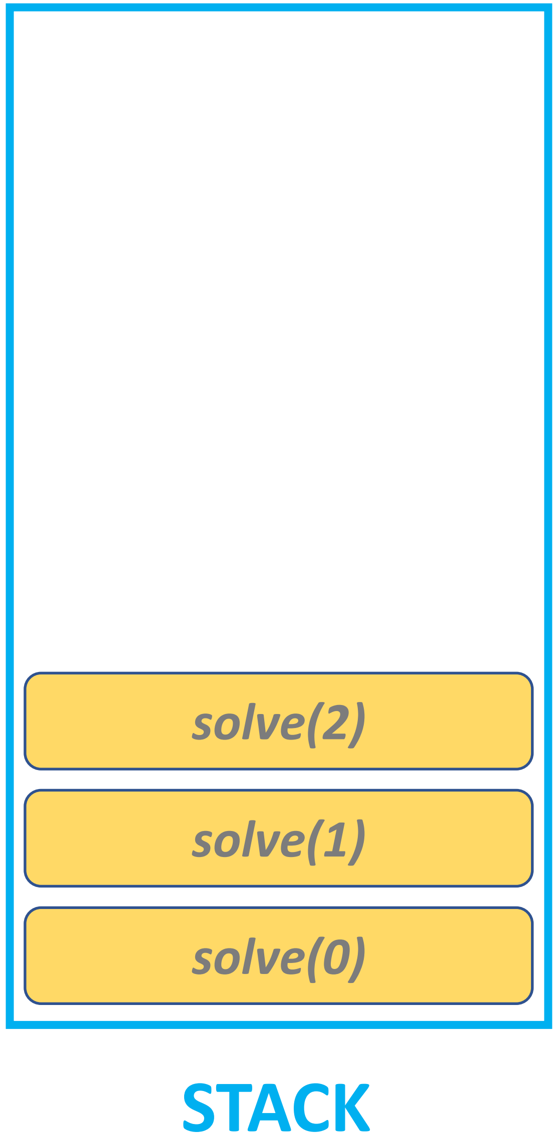


STACK

N-Queens Problem

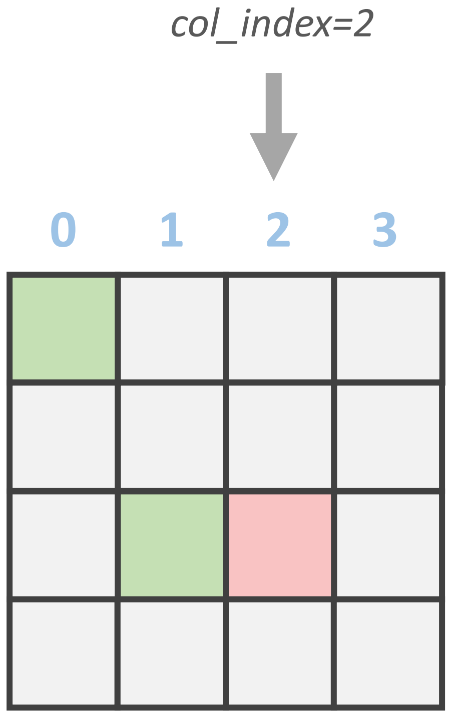
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

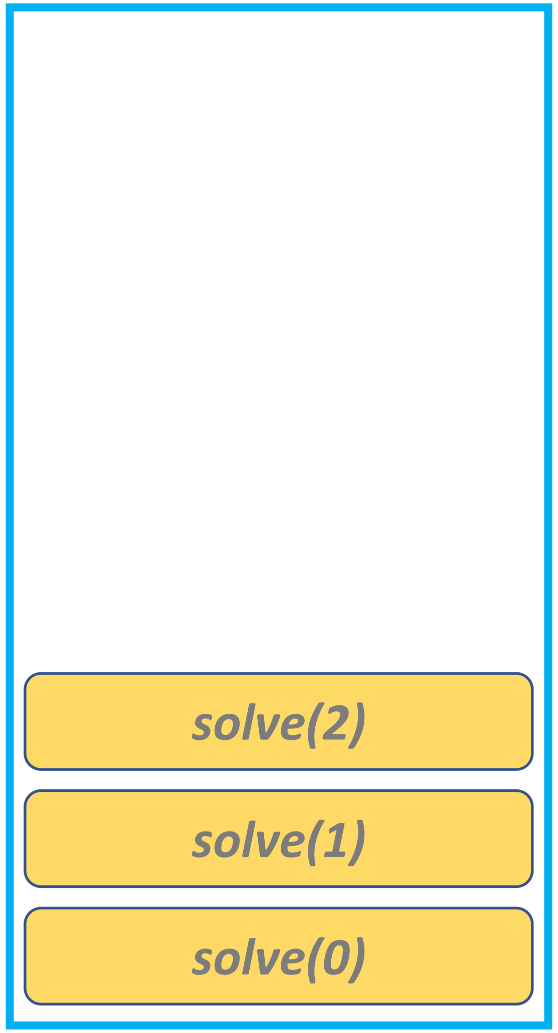




N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

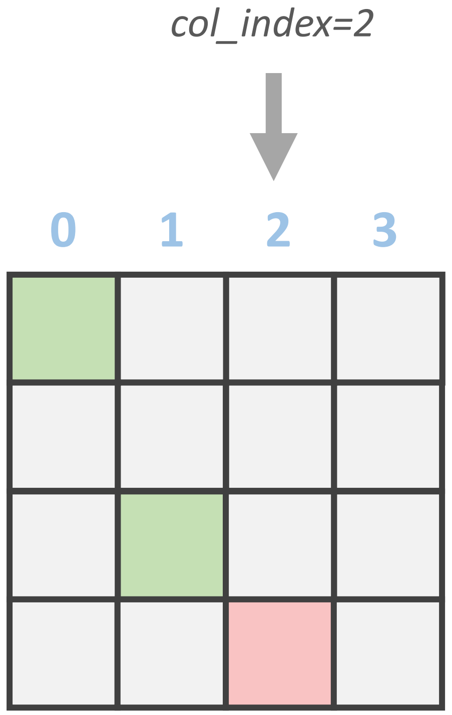


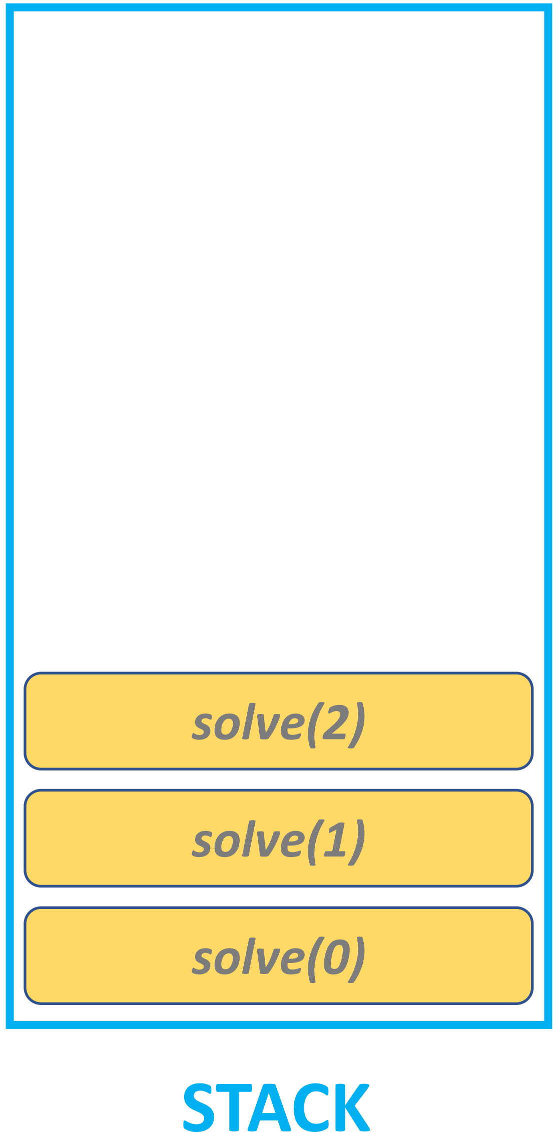


STACK

N-Queens Problem

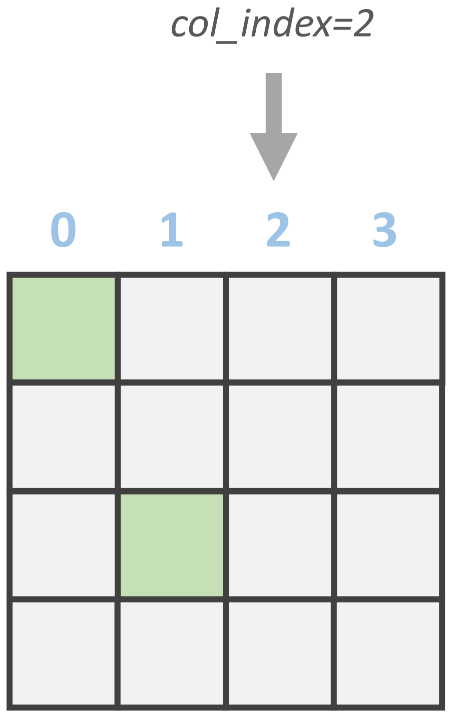
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

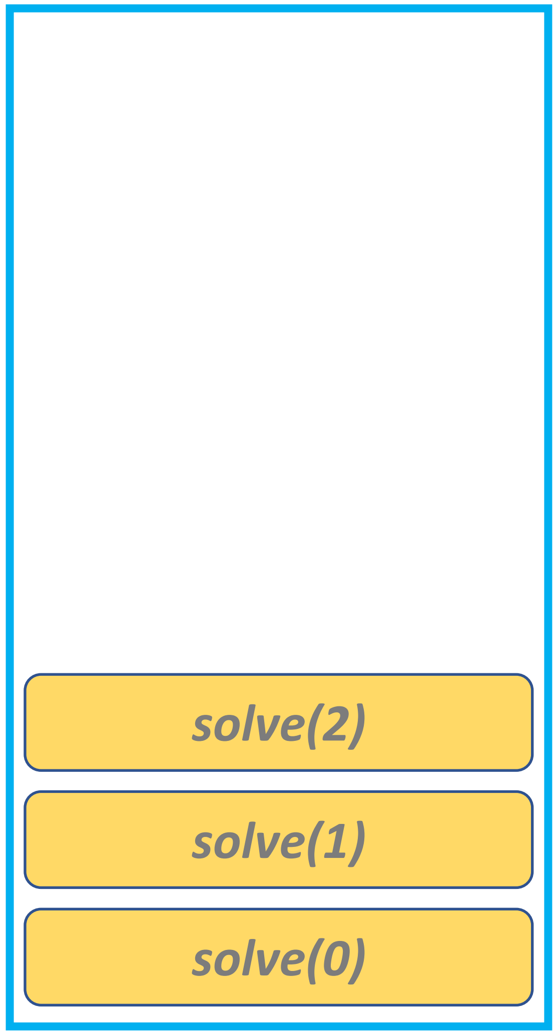




N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

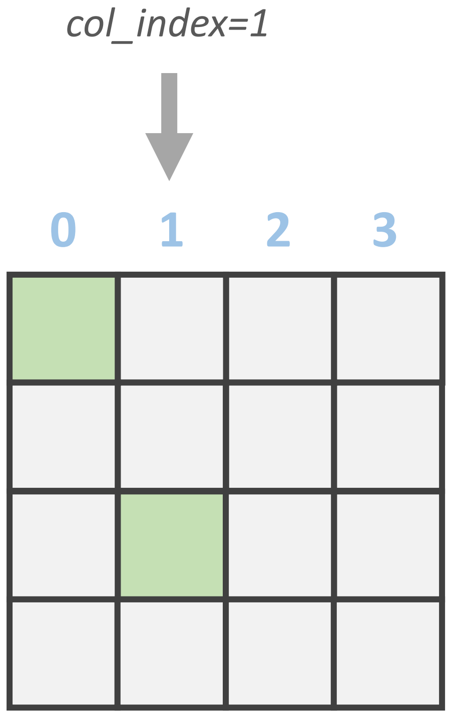


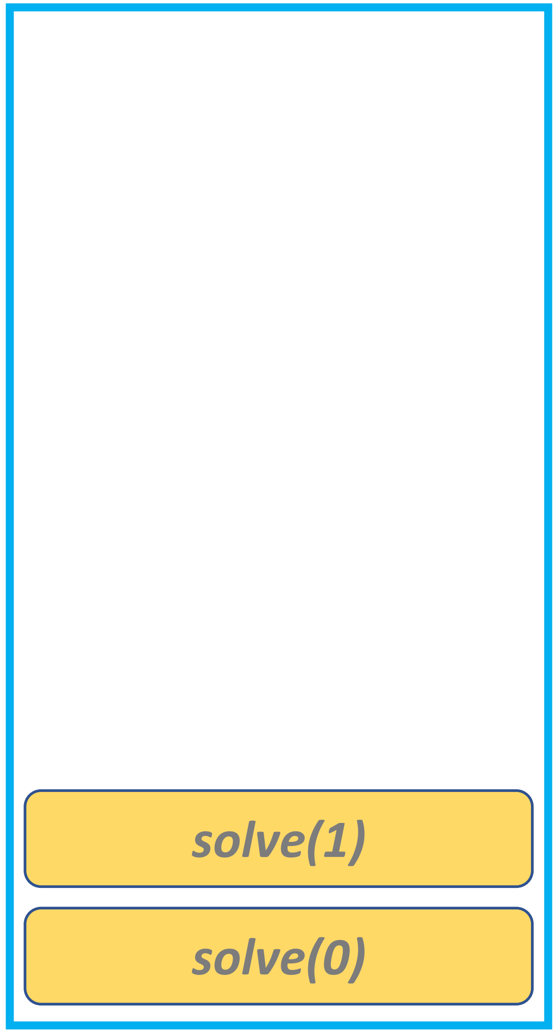


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

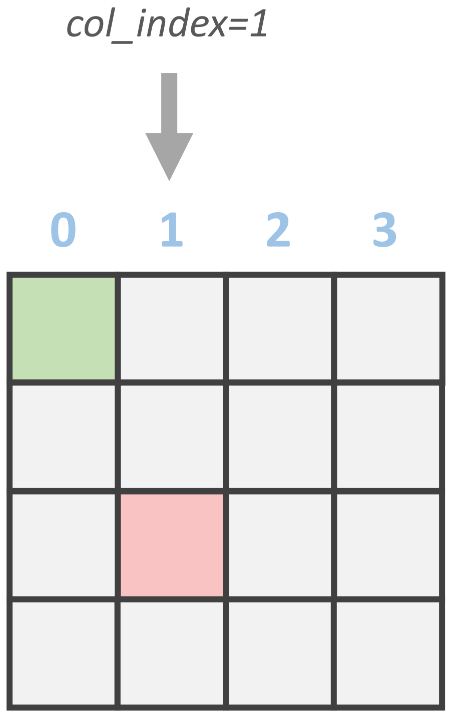


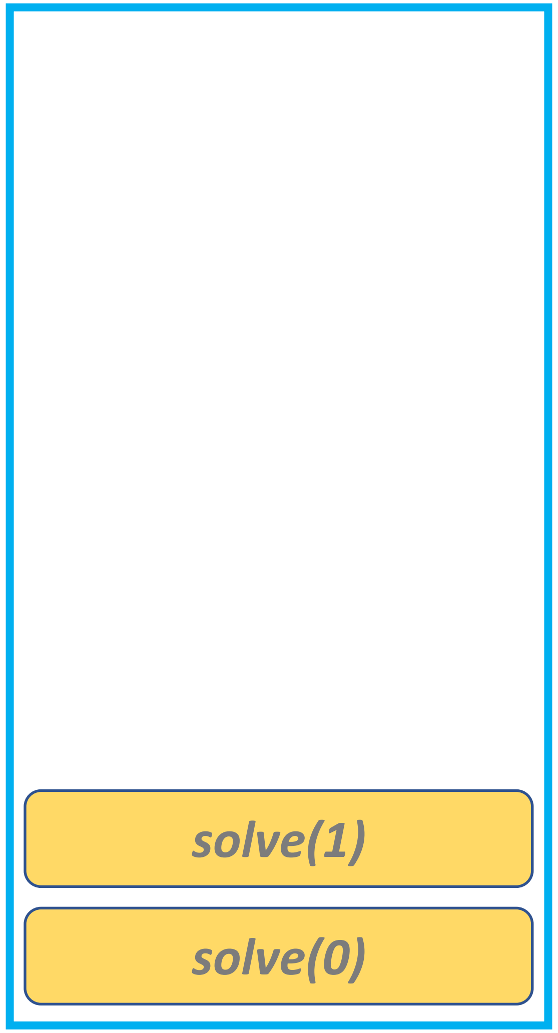


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

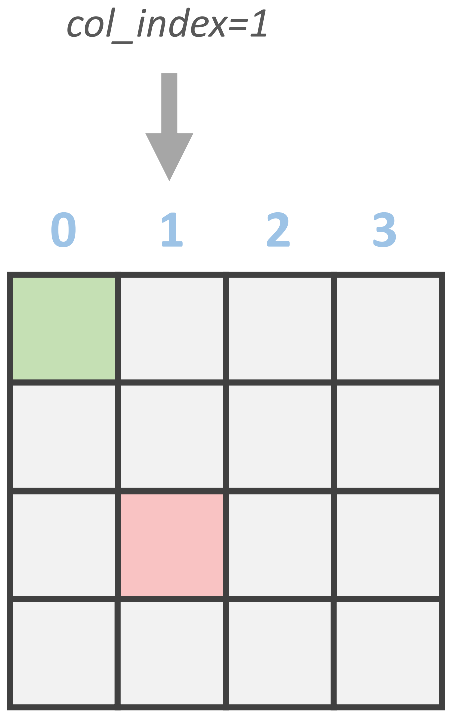


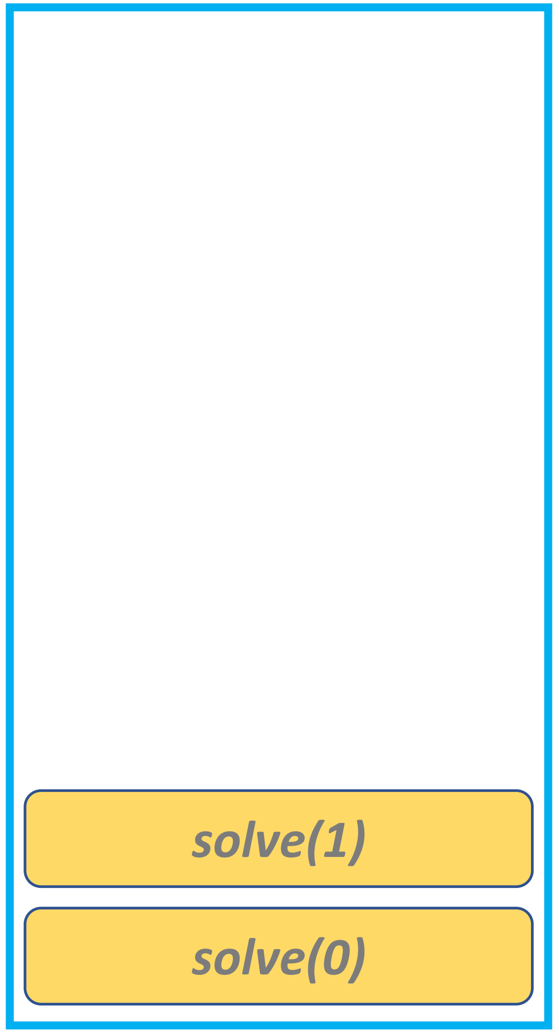


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

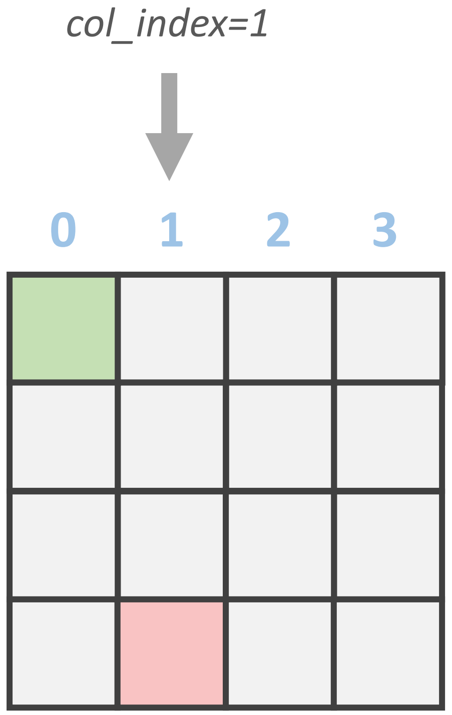


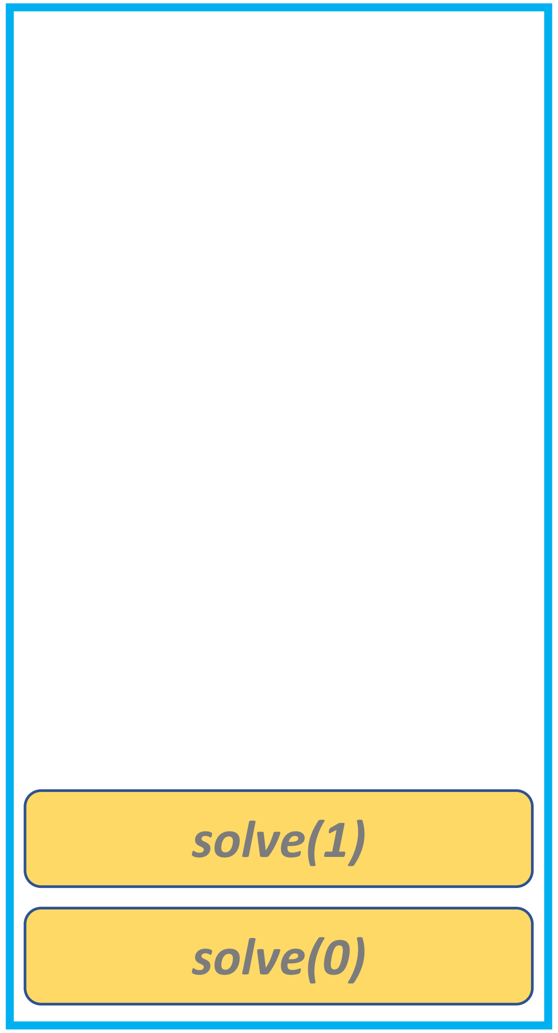


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

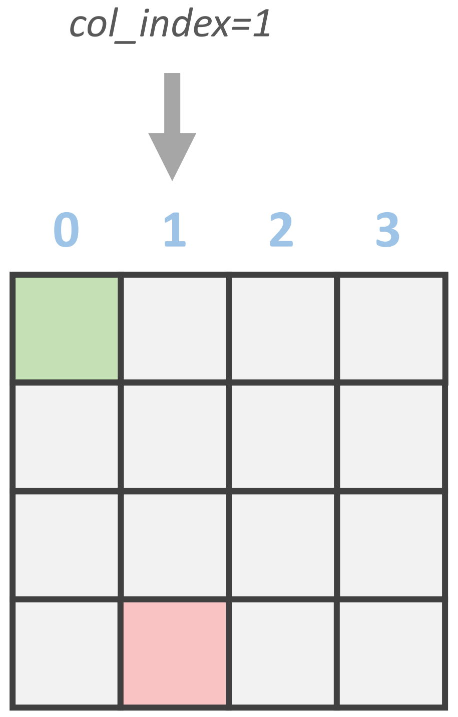


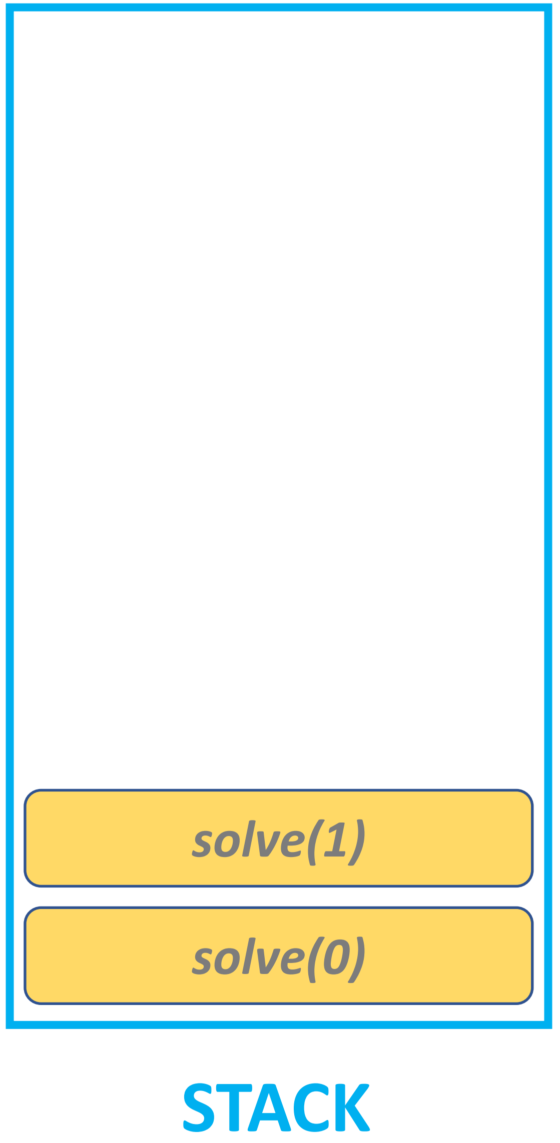


STACK

N-Queens Problem

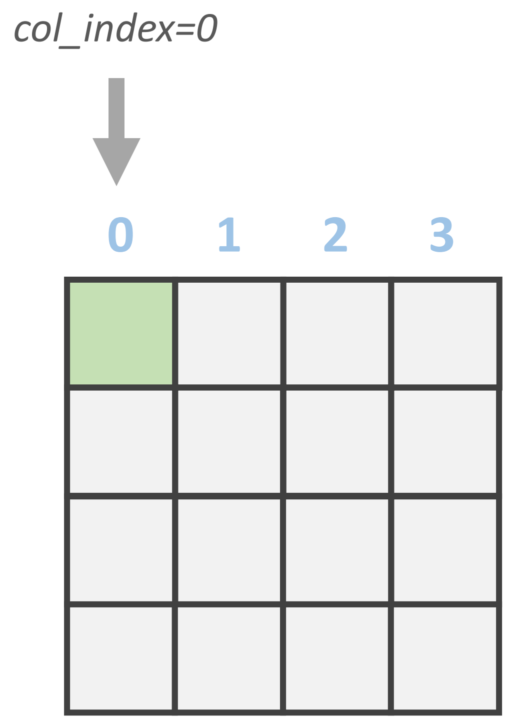
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

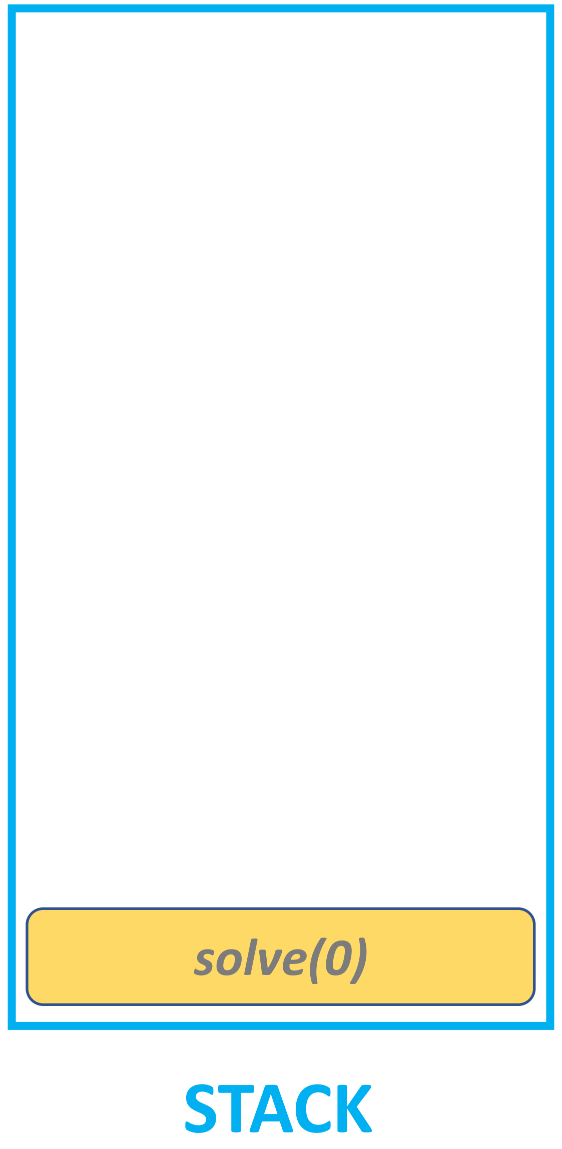




N-Queens Problem

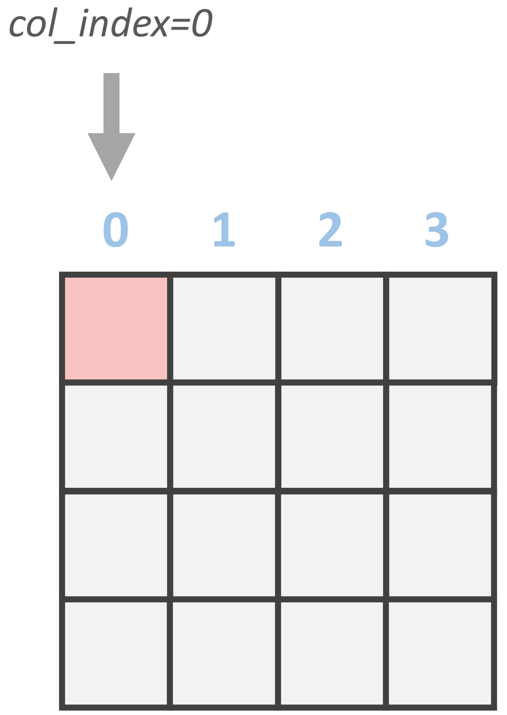
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

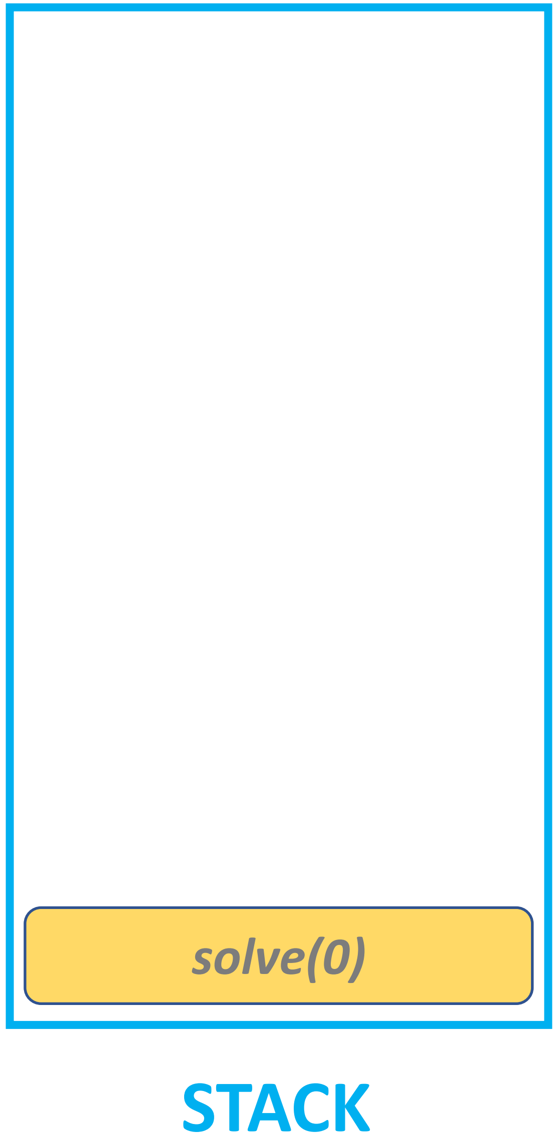




N-Queens Problem

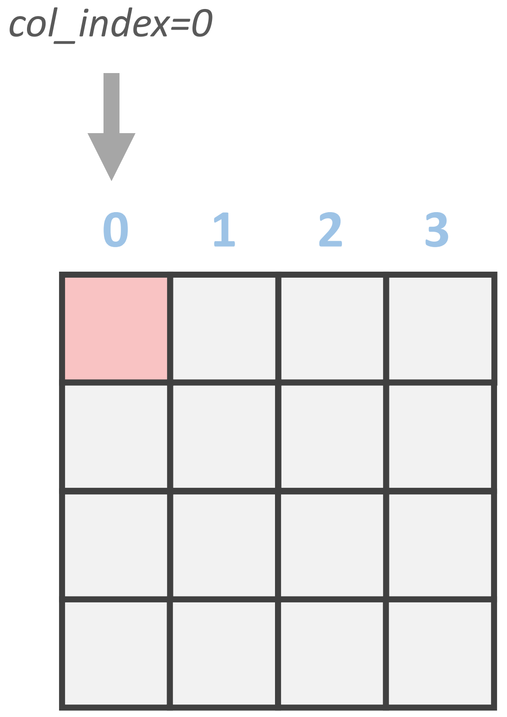
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

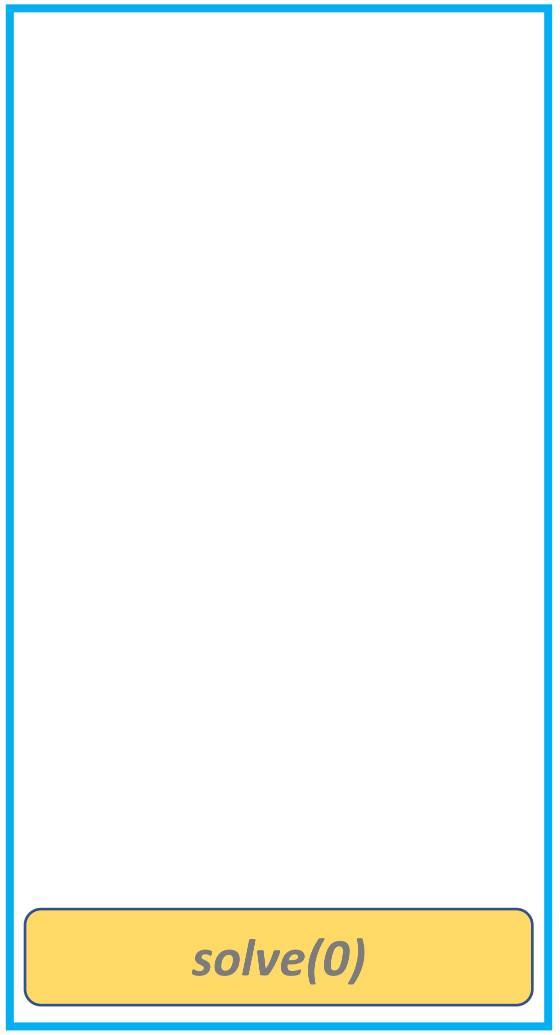




N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

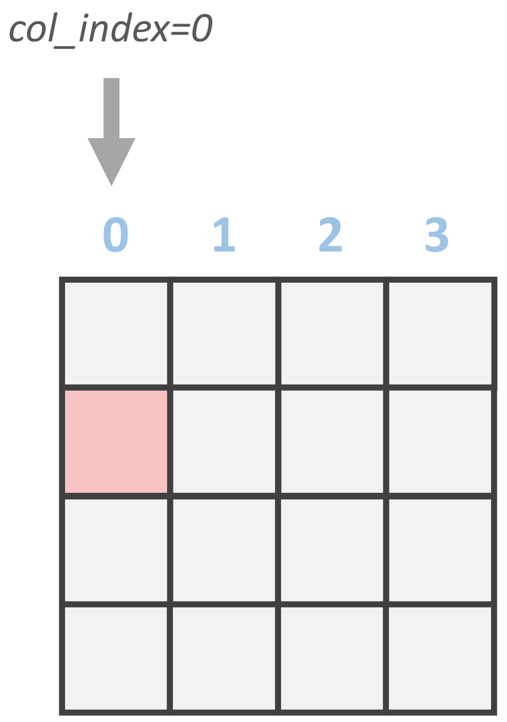


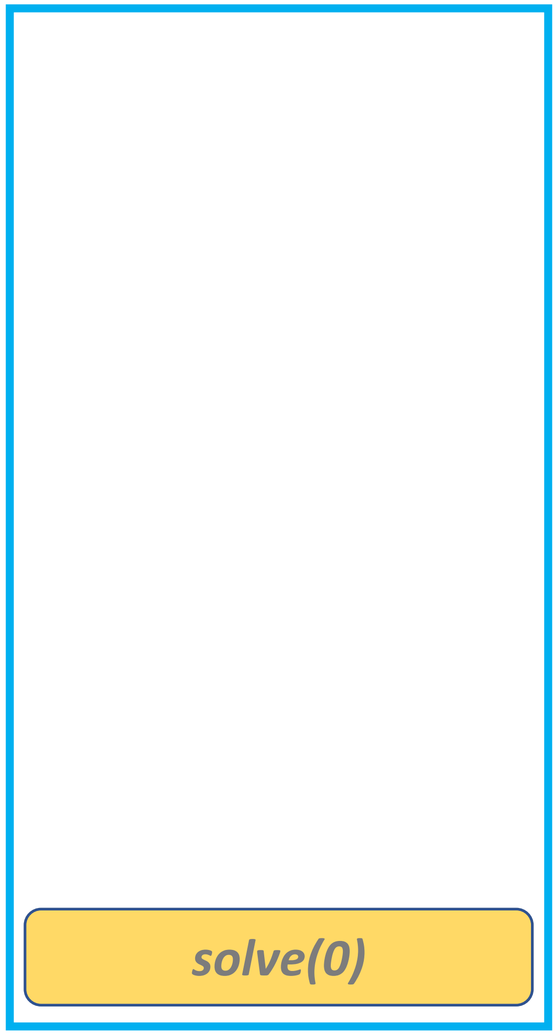


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

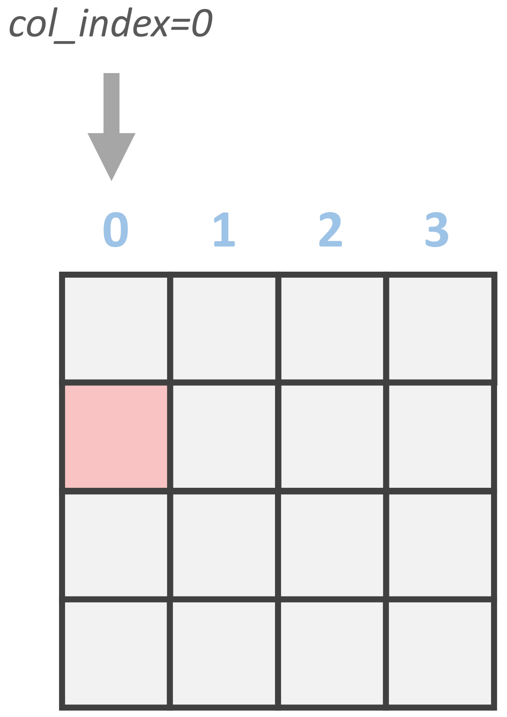


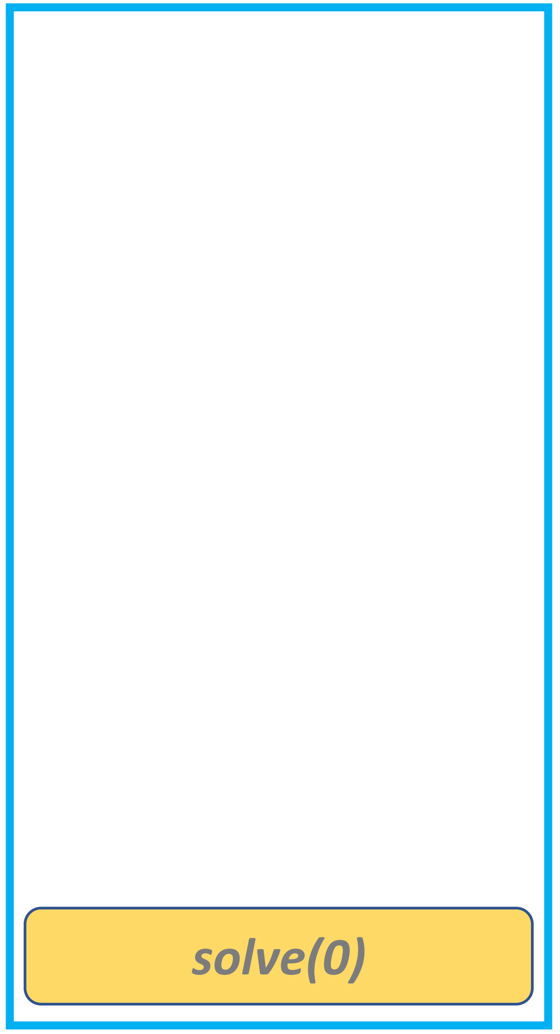


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

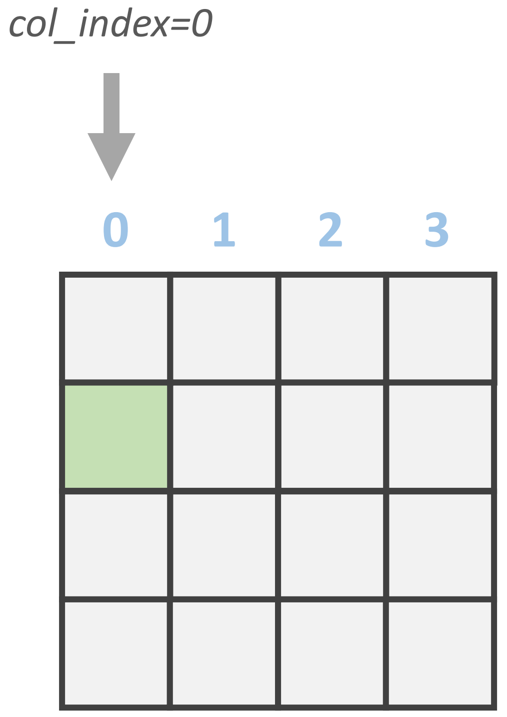


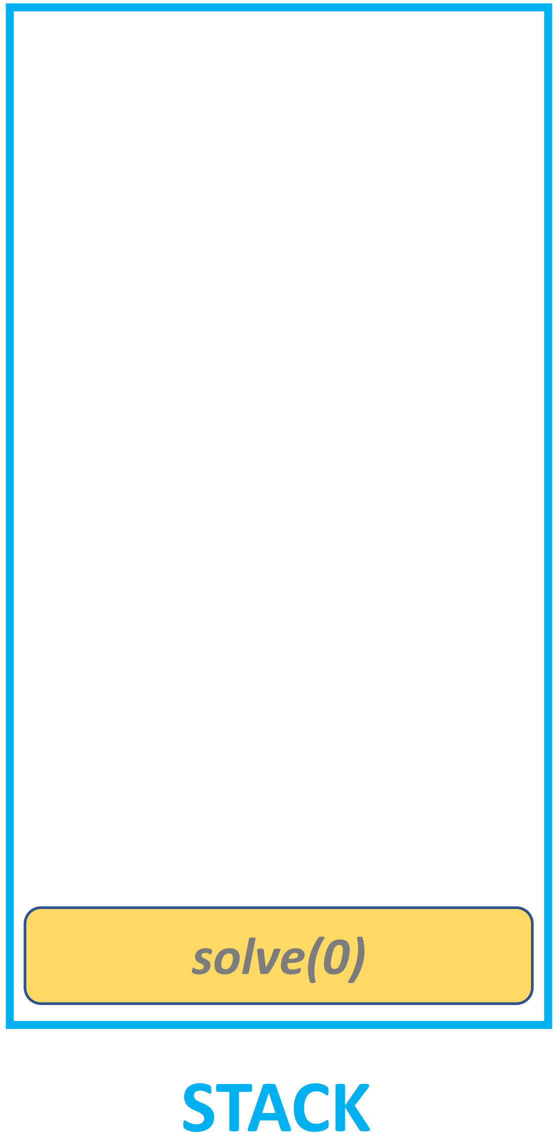


STACK

N-Queens Problem

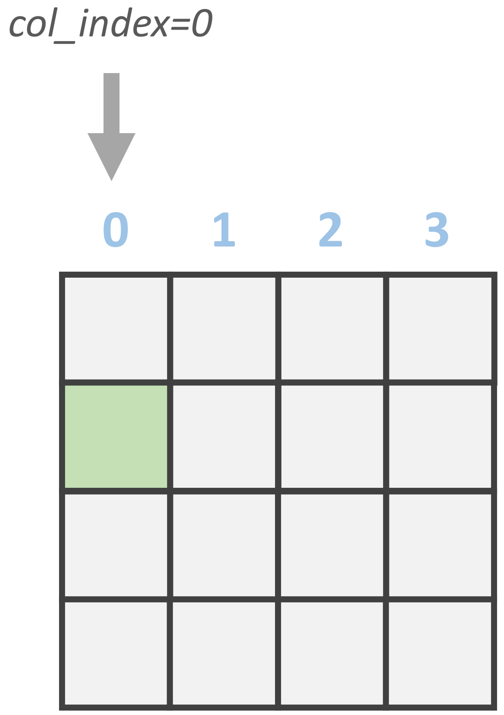
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

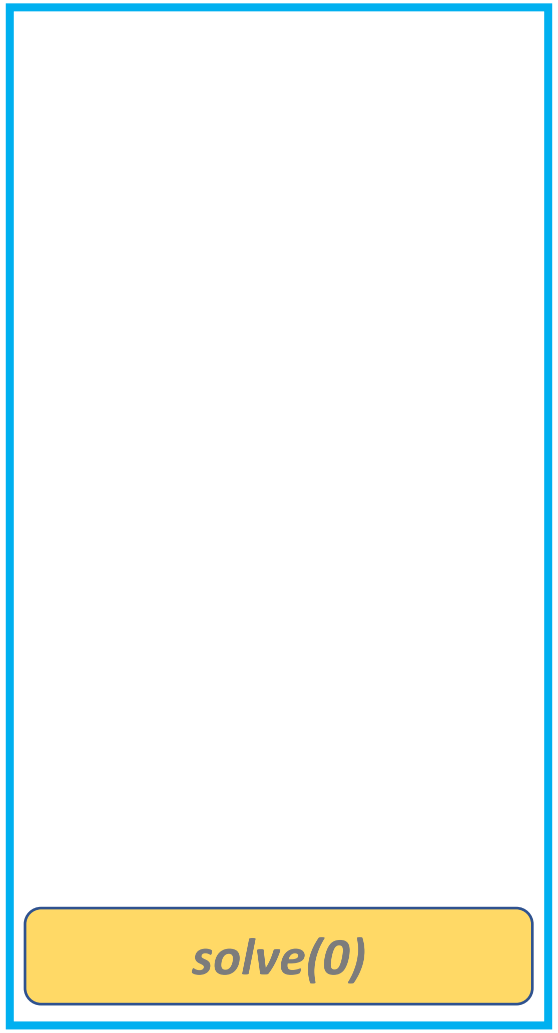




N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

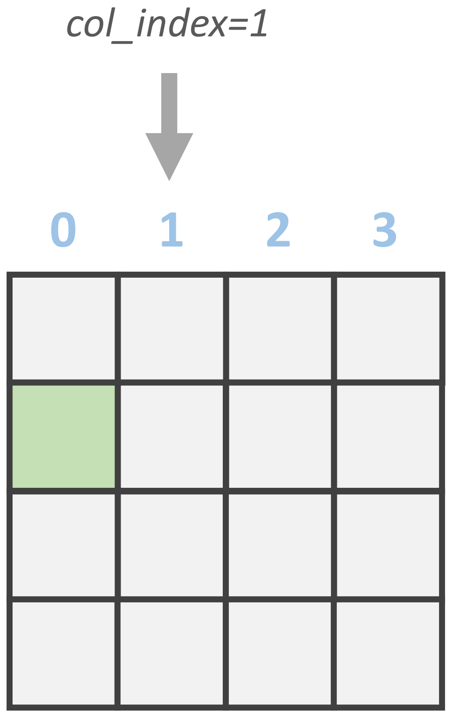


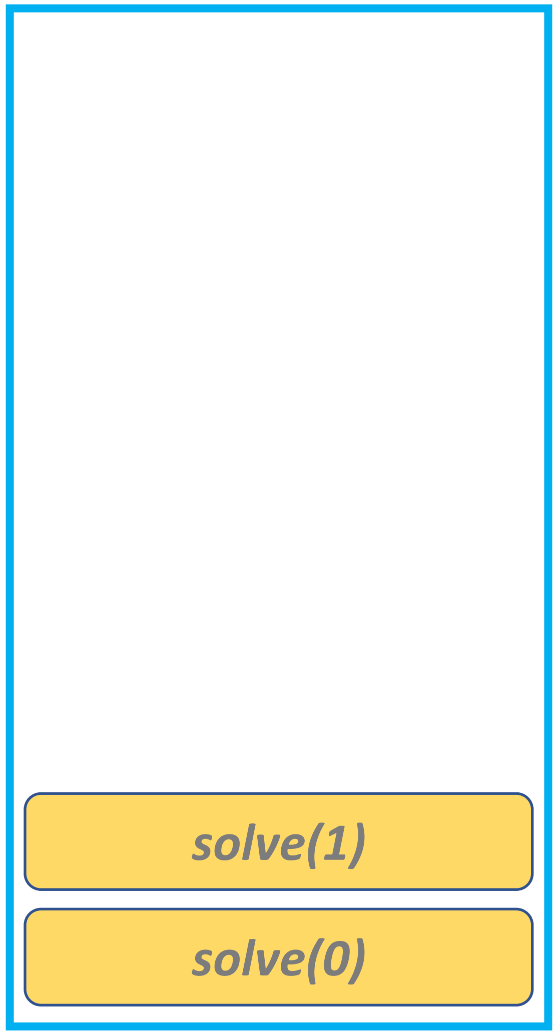


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

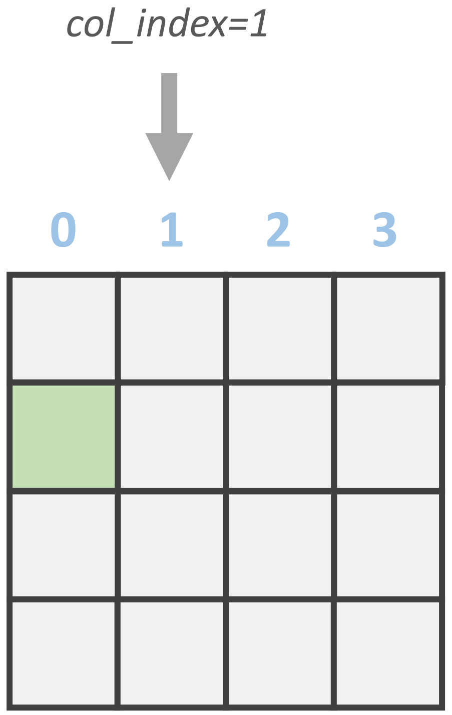


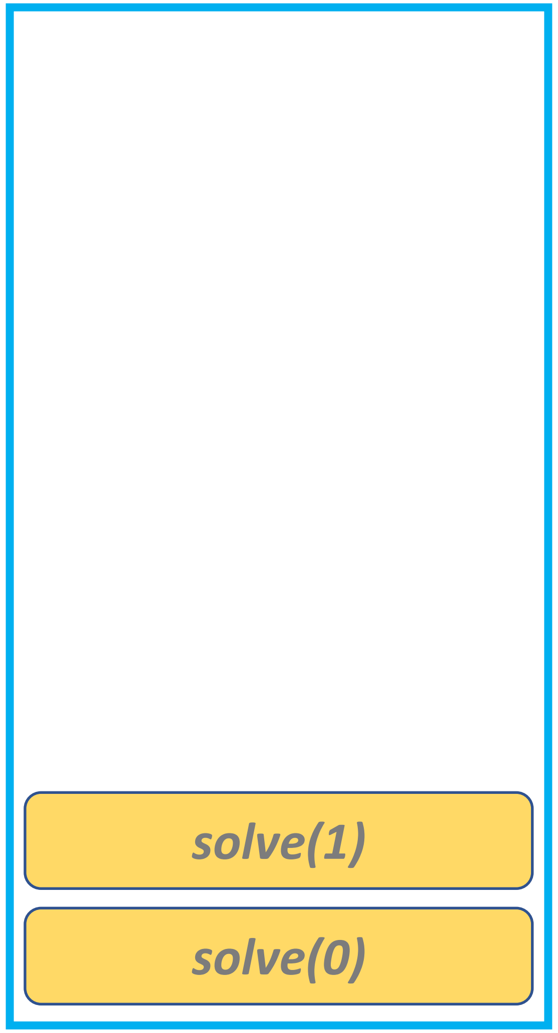


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

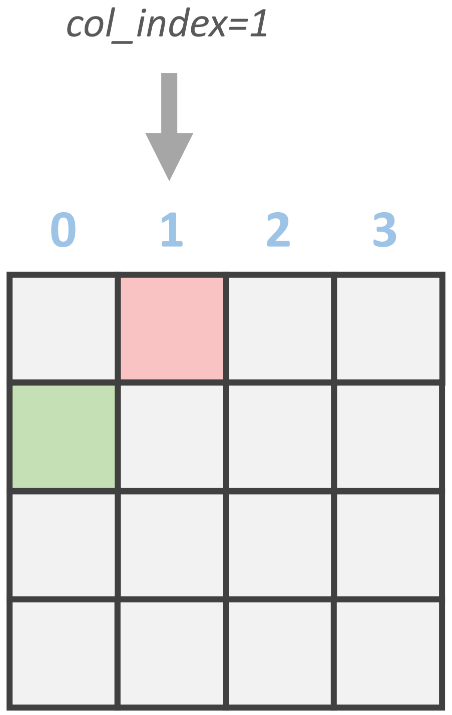


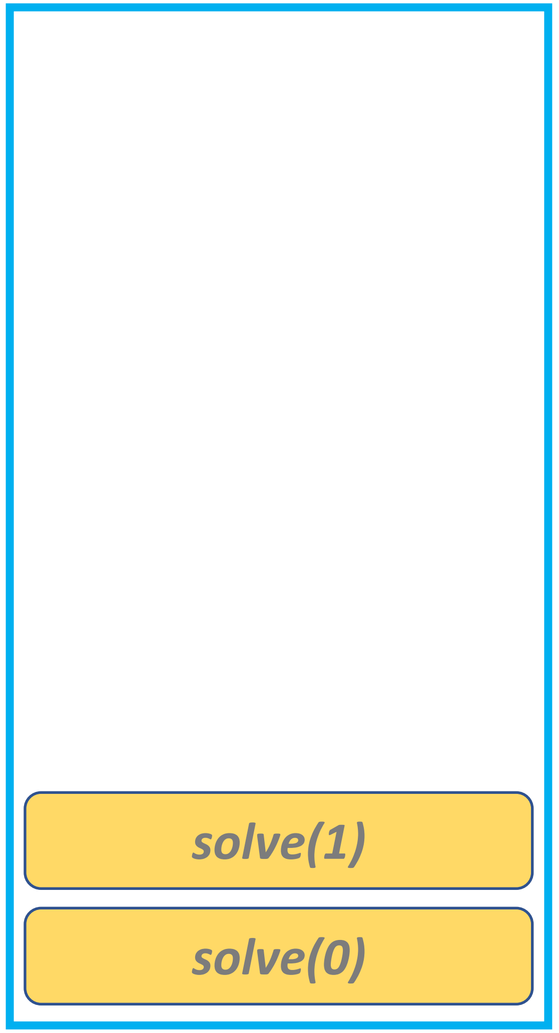


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

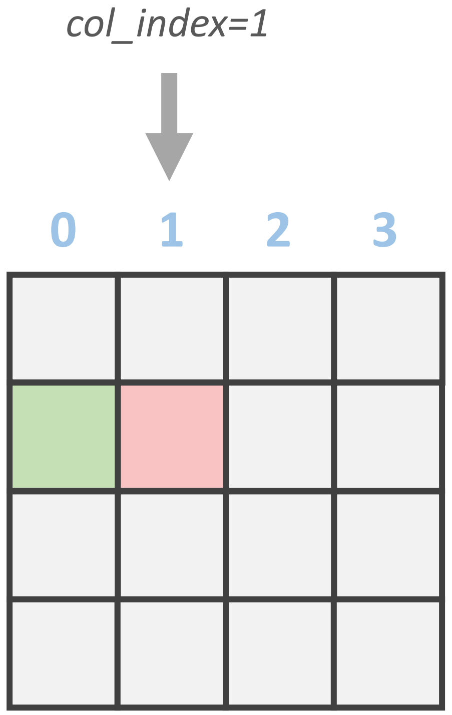


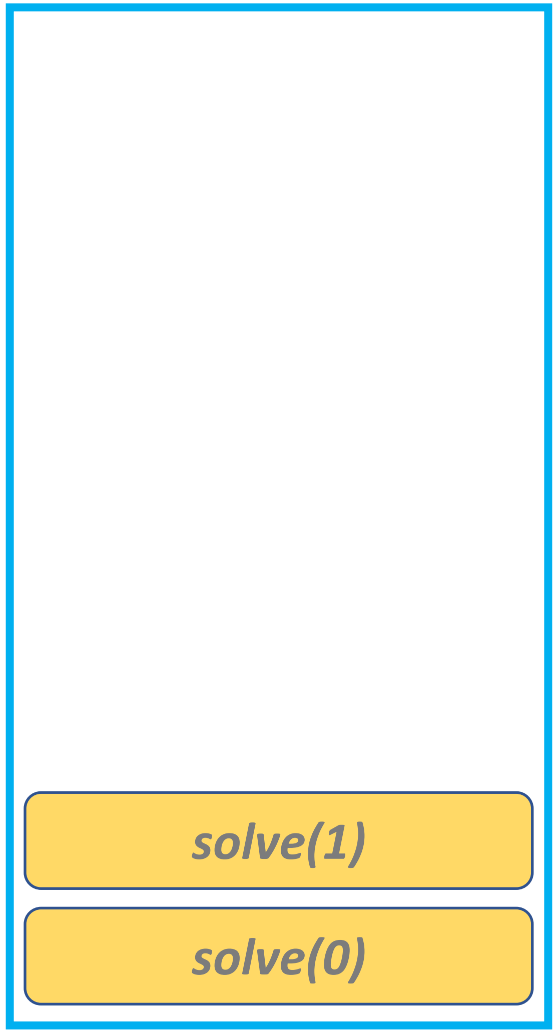


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

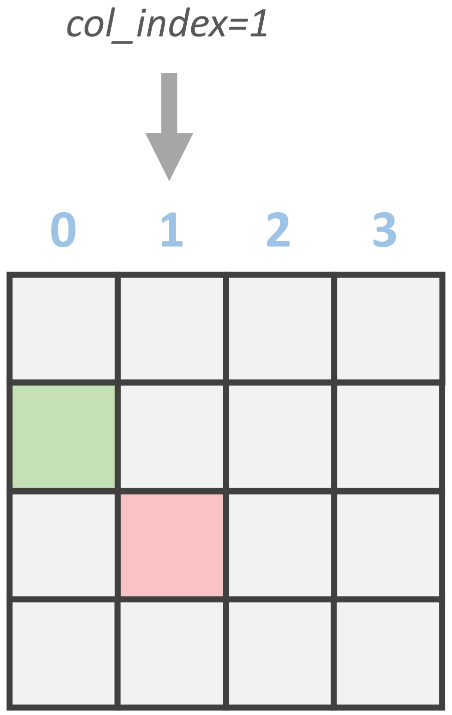


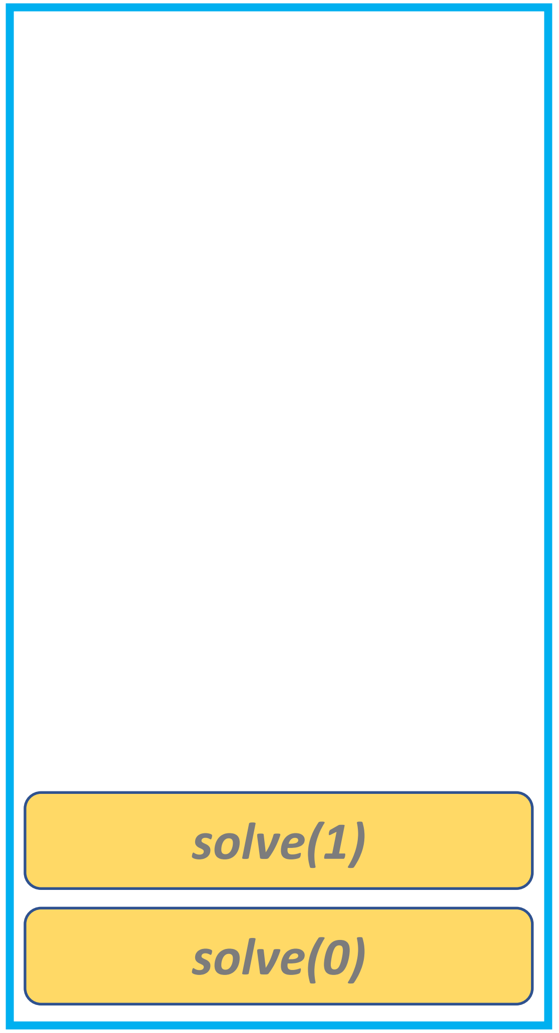


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

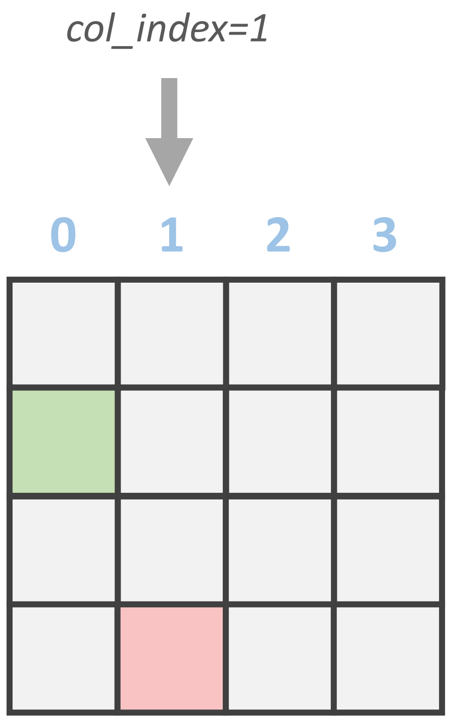


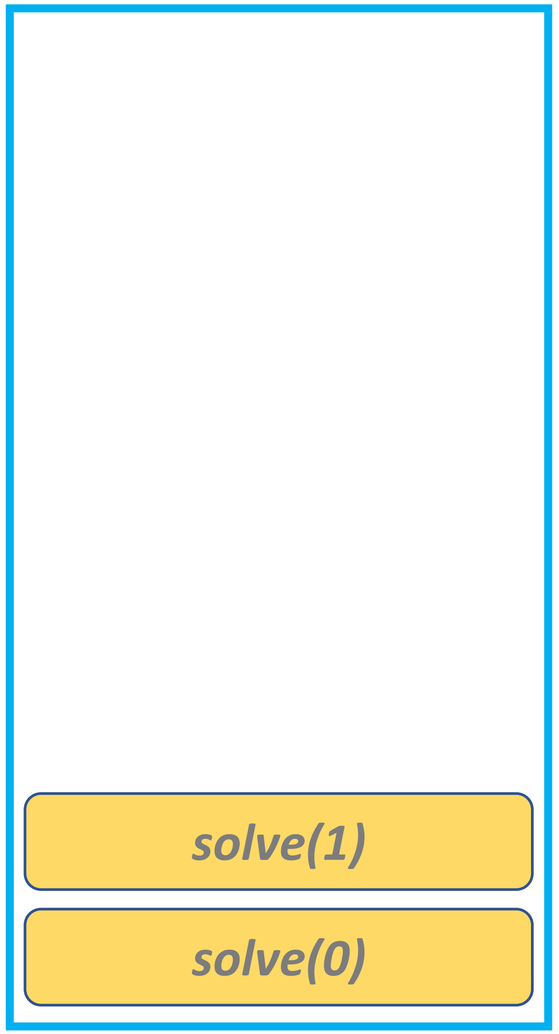


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

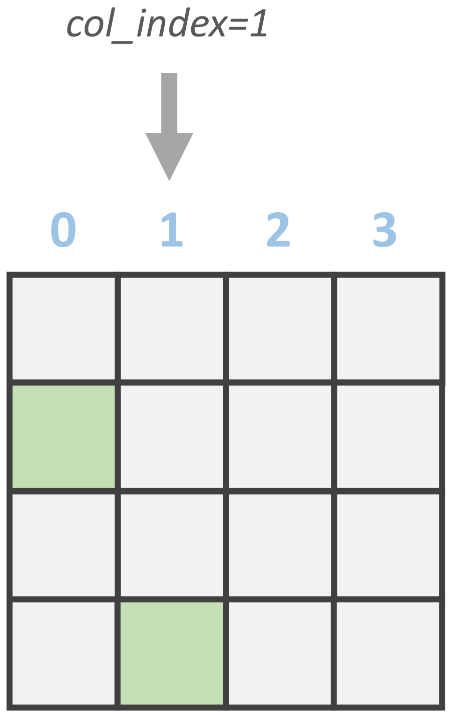


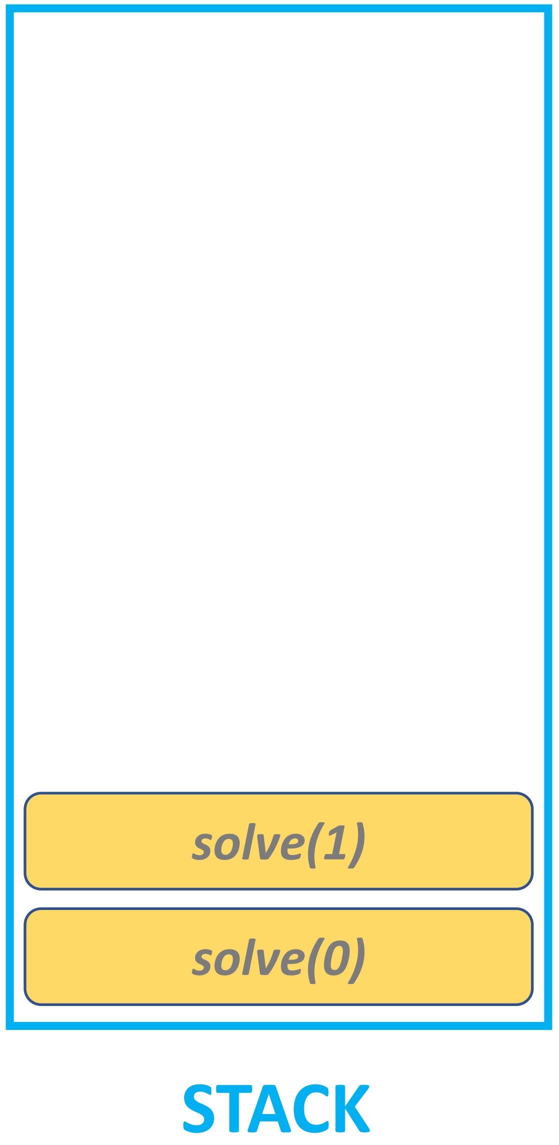


STACK

N-Queens Problem

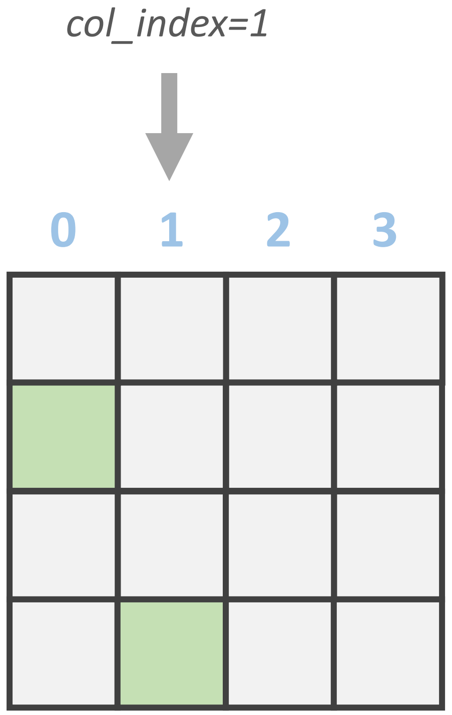
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

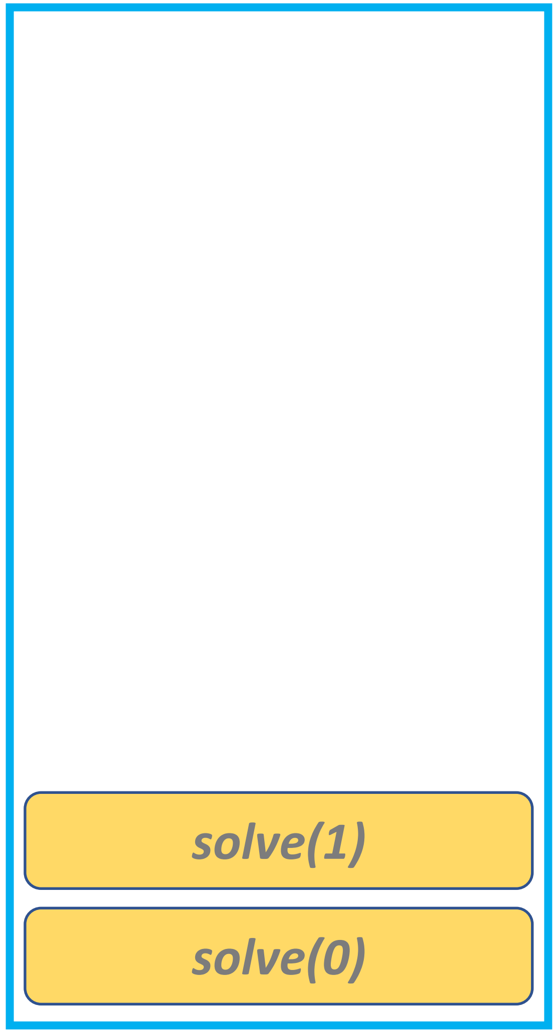




N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

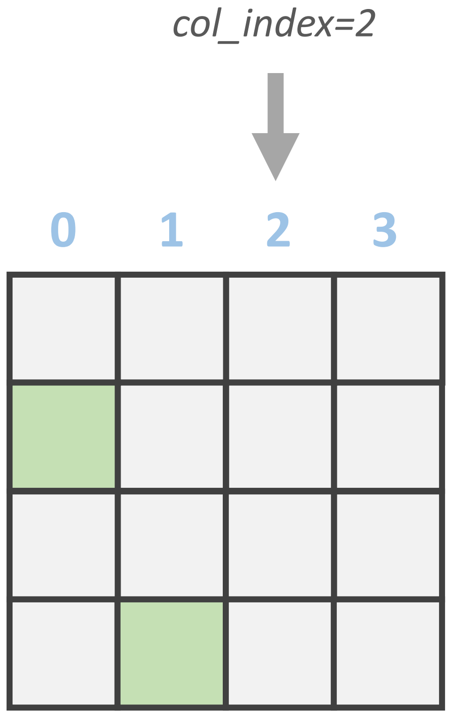


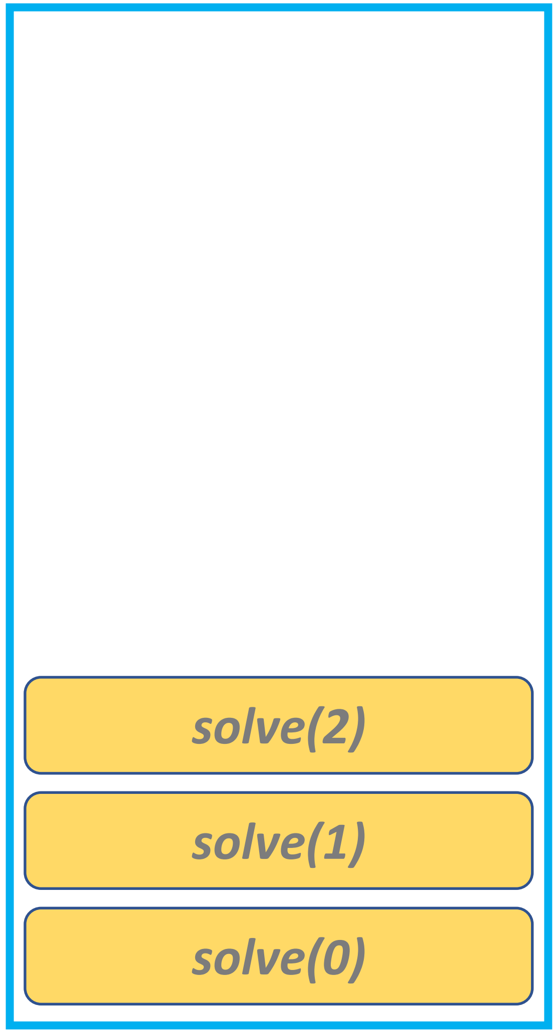


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

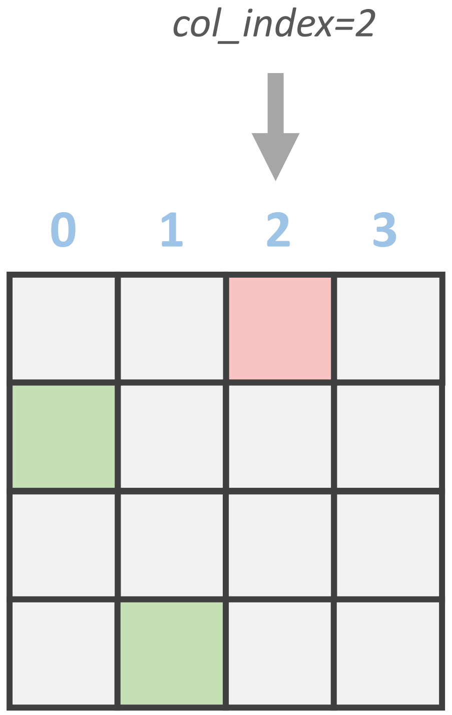


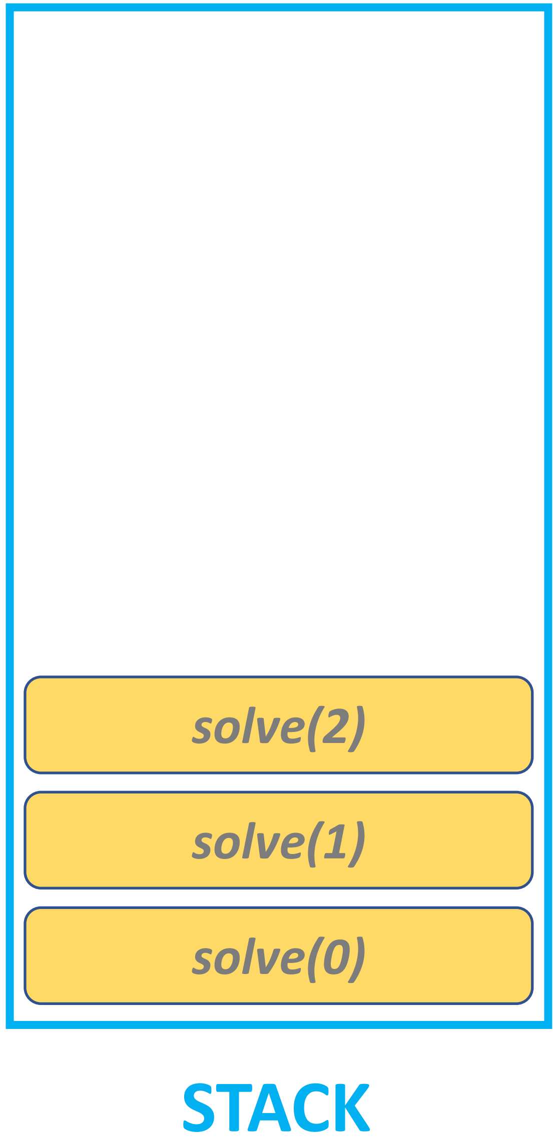


STACK

N-Queens Problem

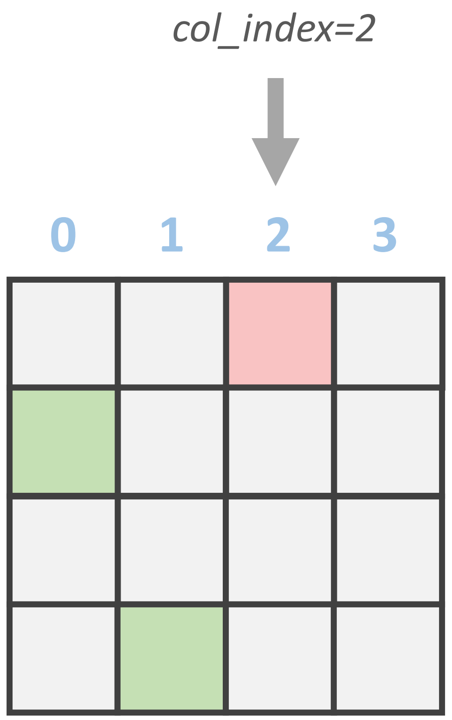
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

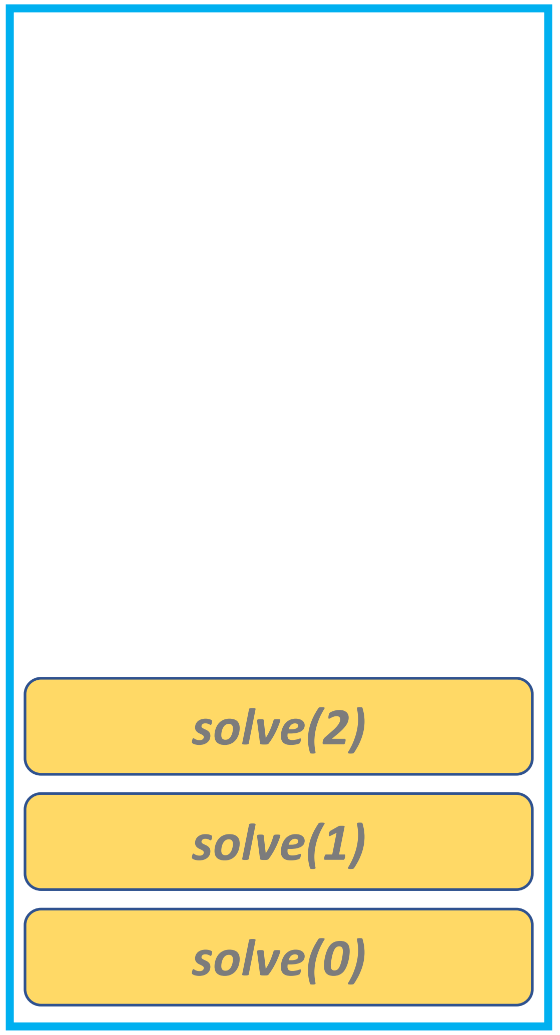




N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

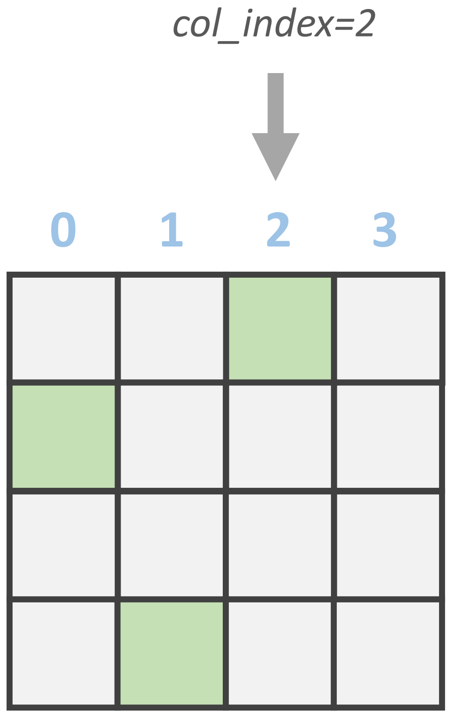


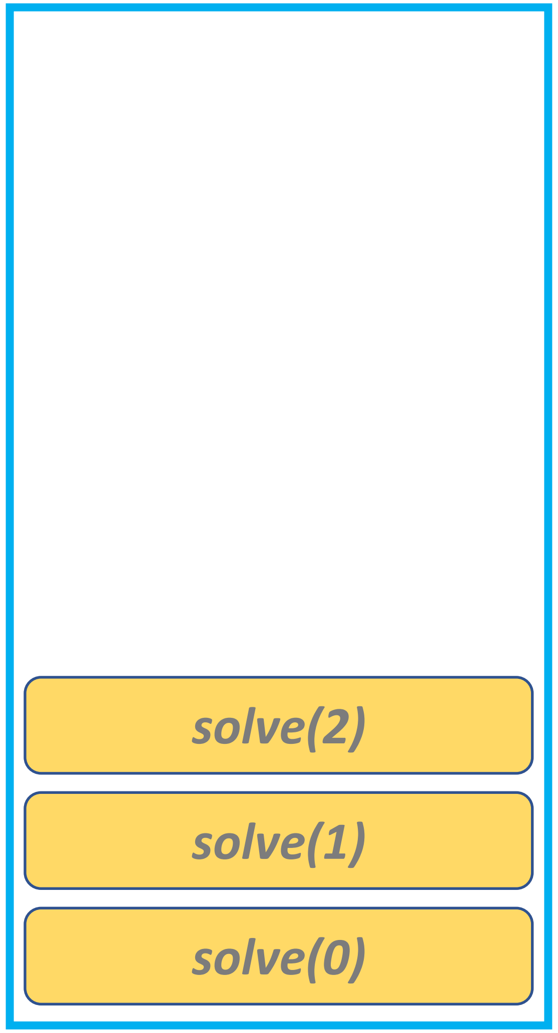


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

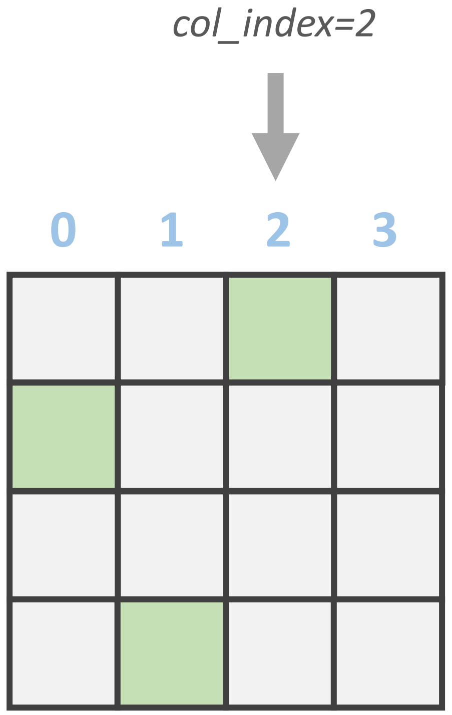


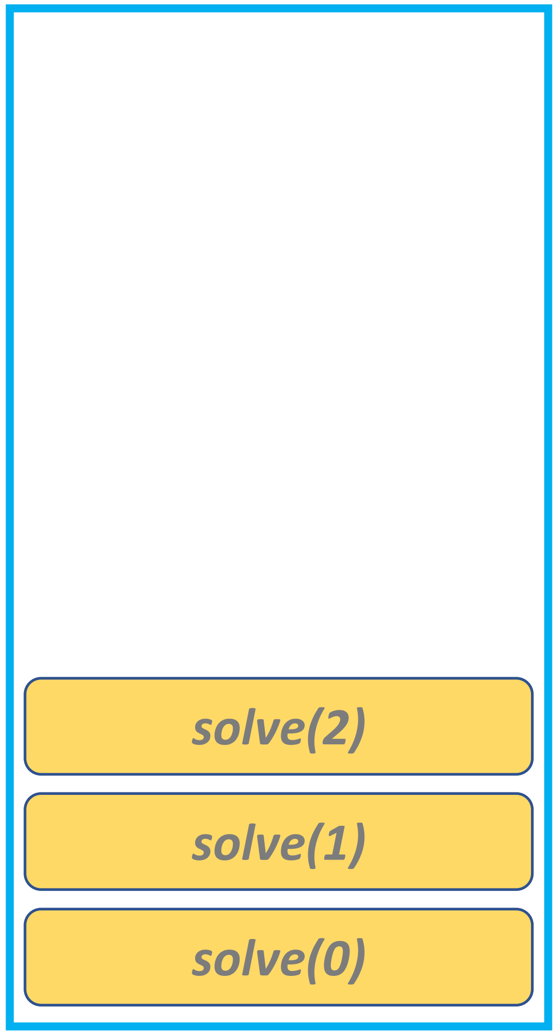


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

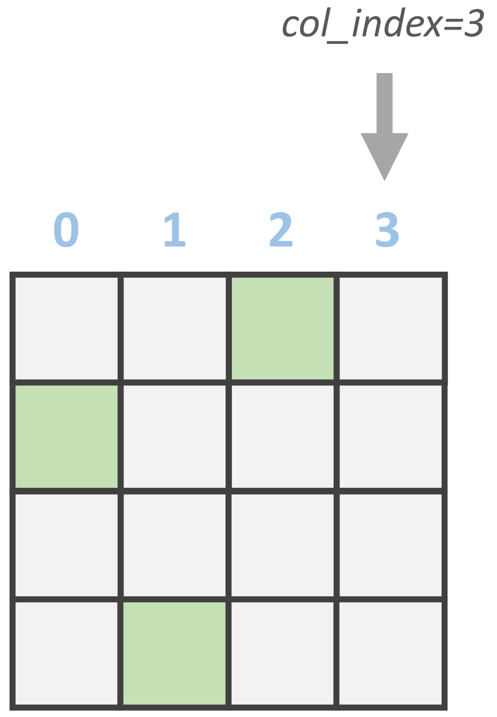


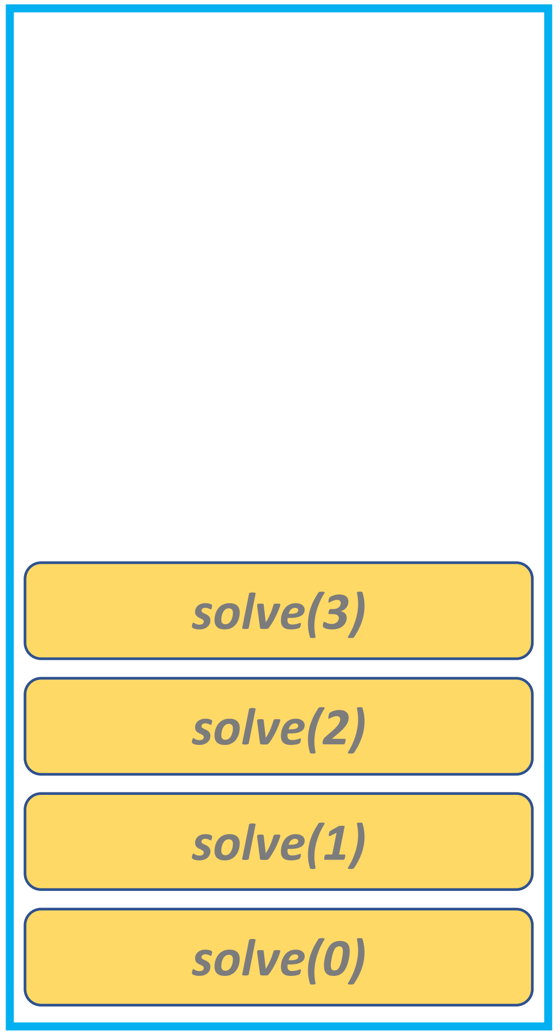


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

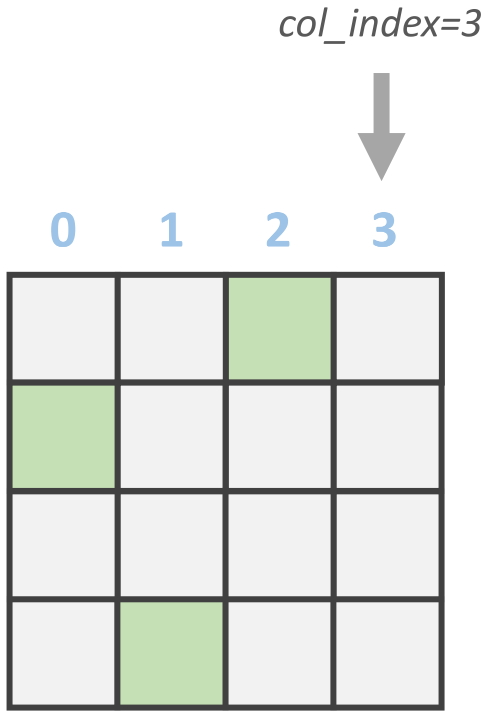


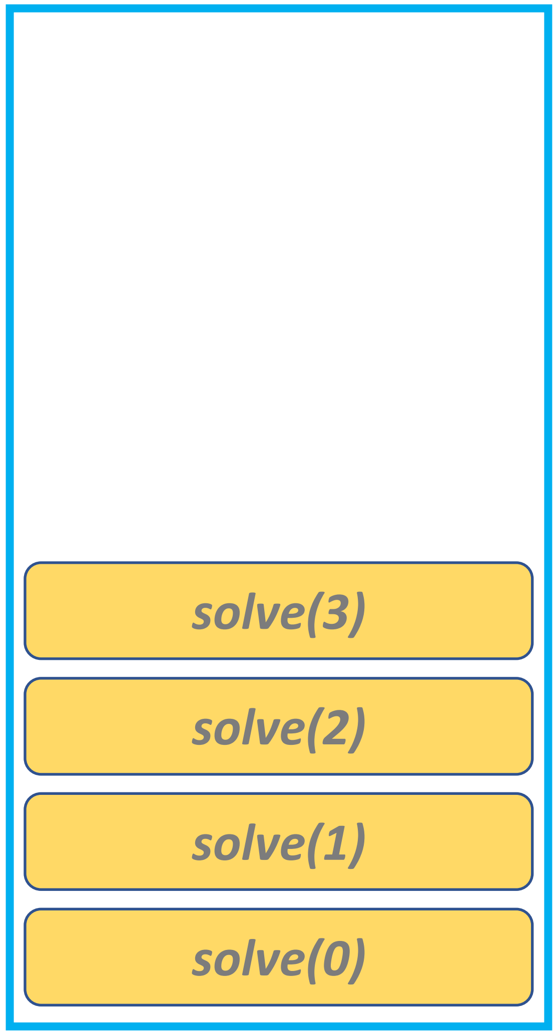


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

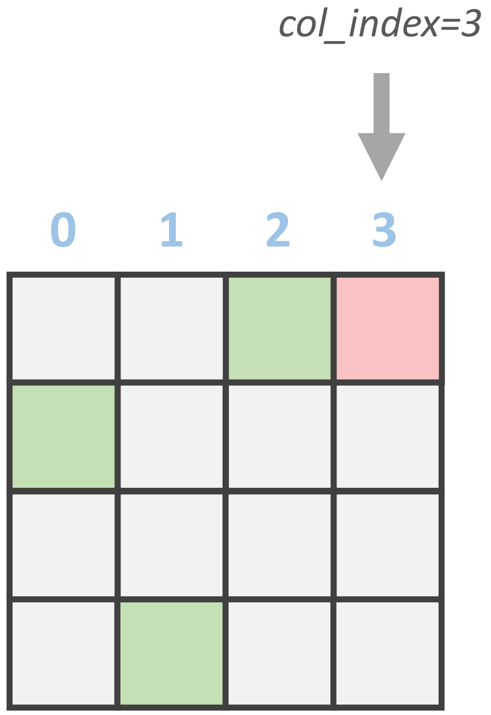


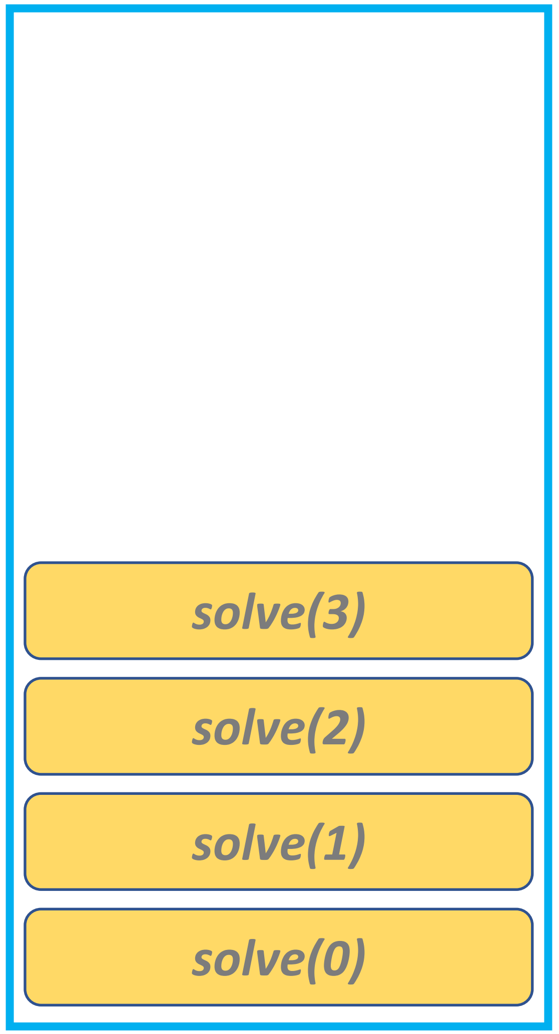


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

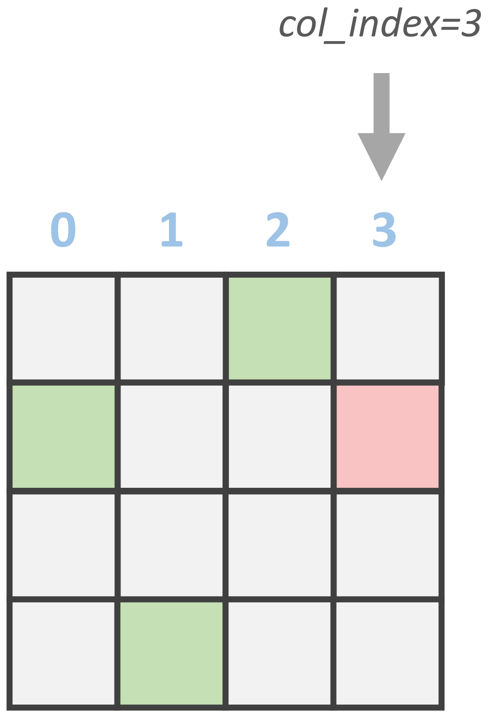


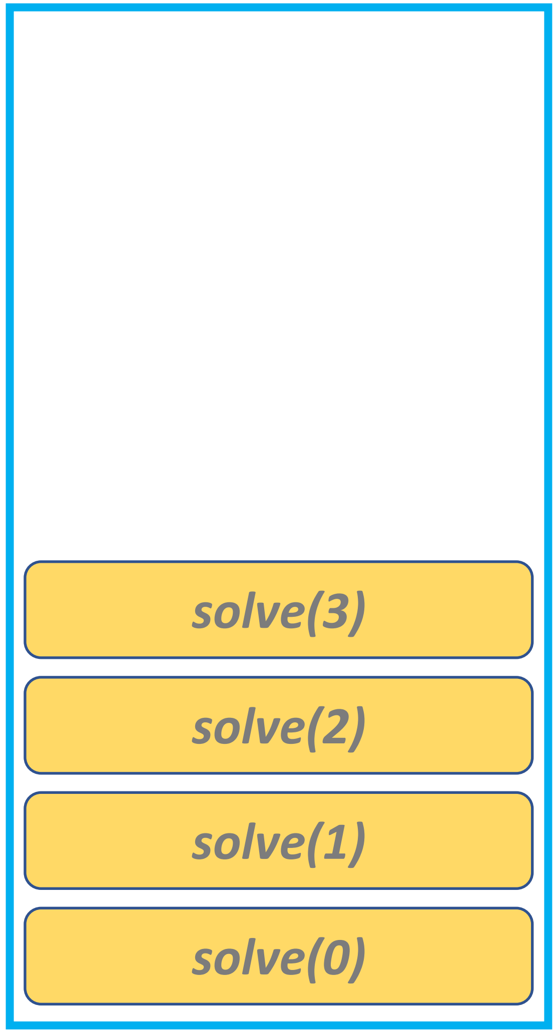


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

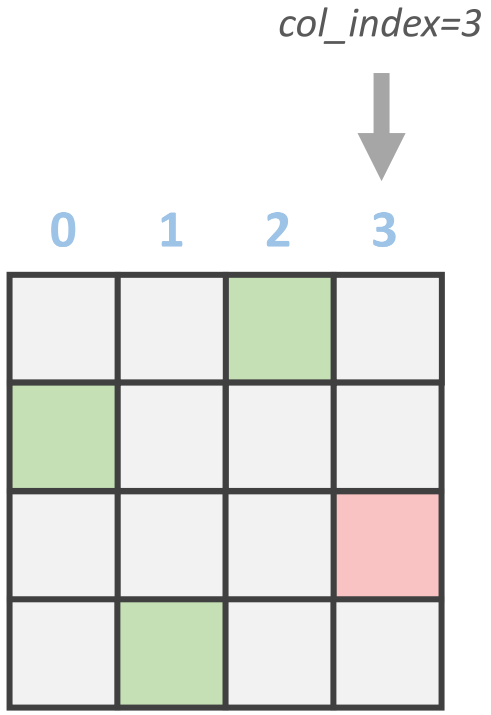


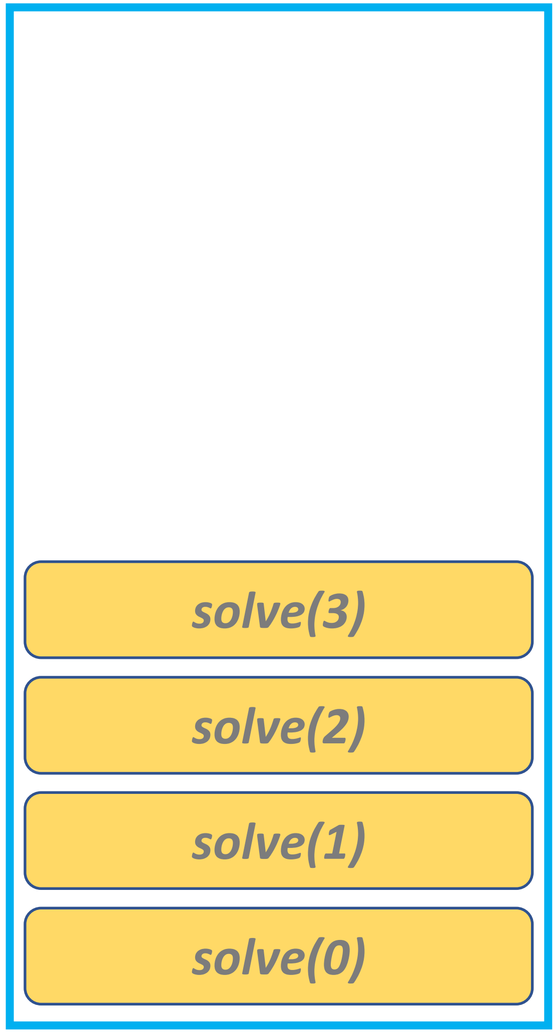


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

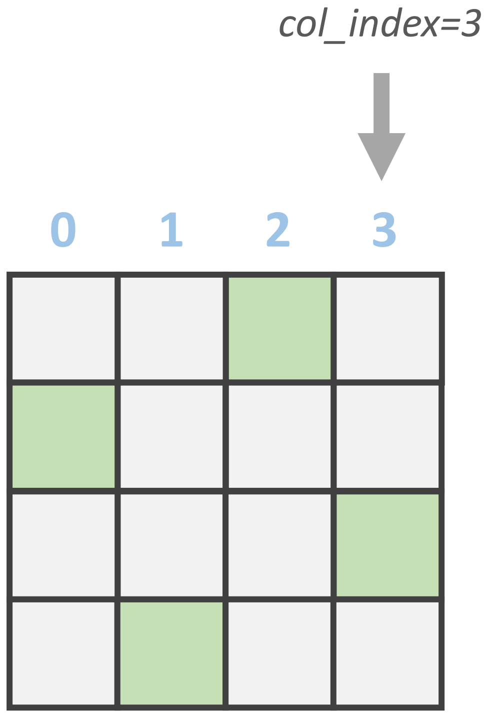


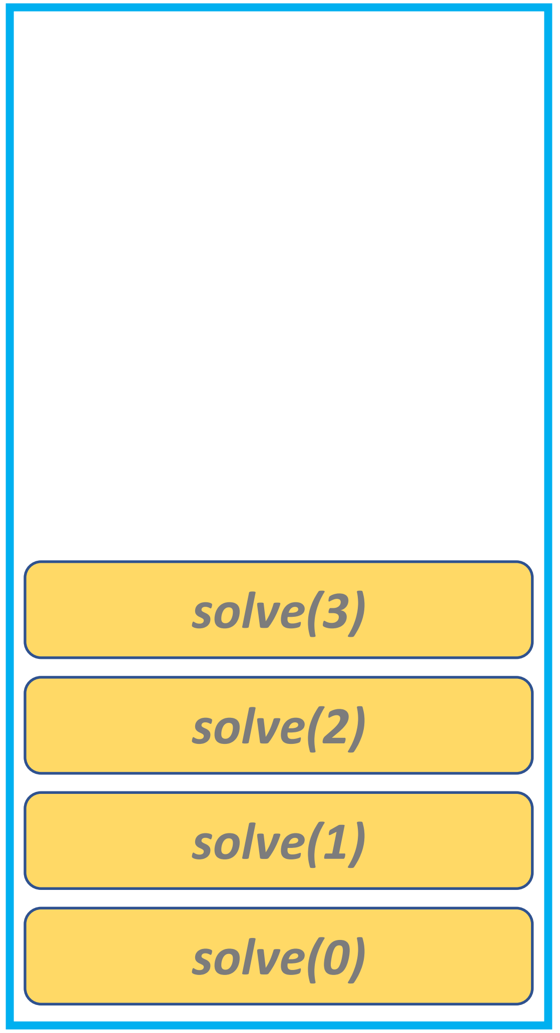


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

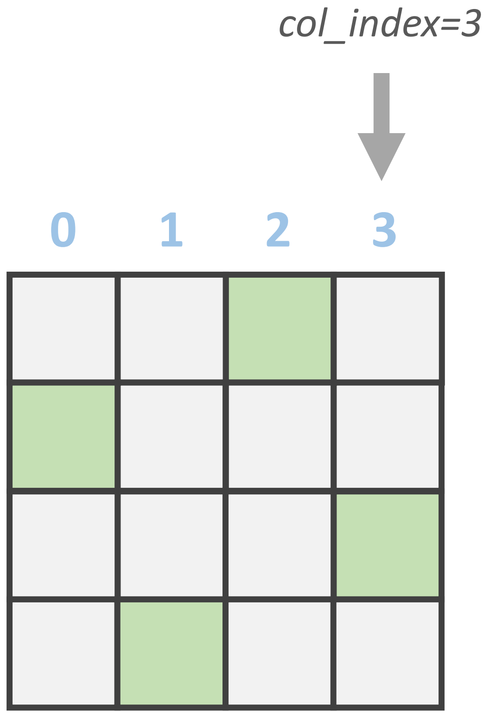


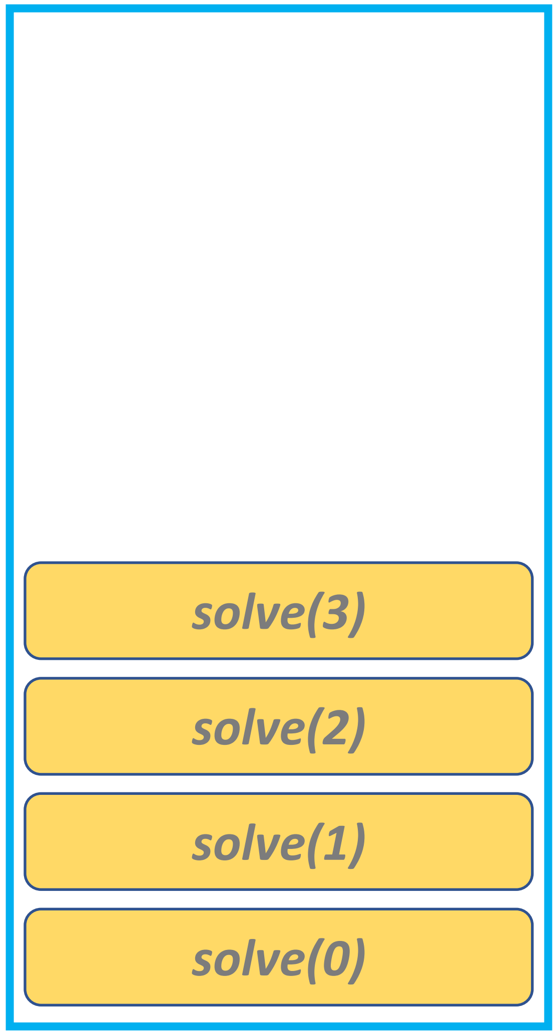


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

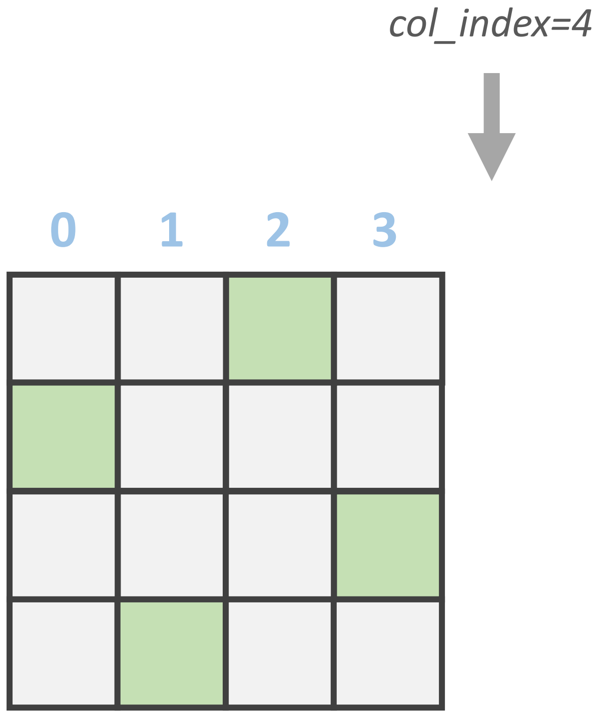


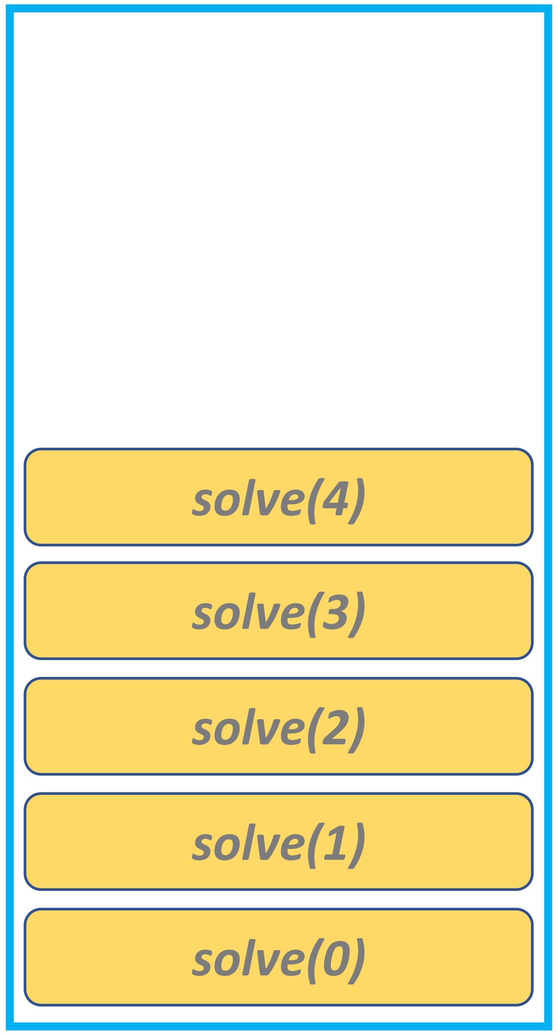


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

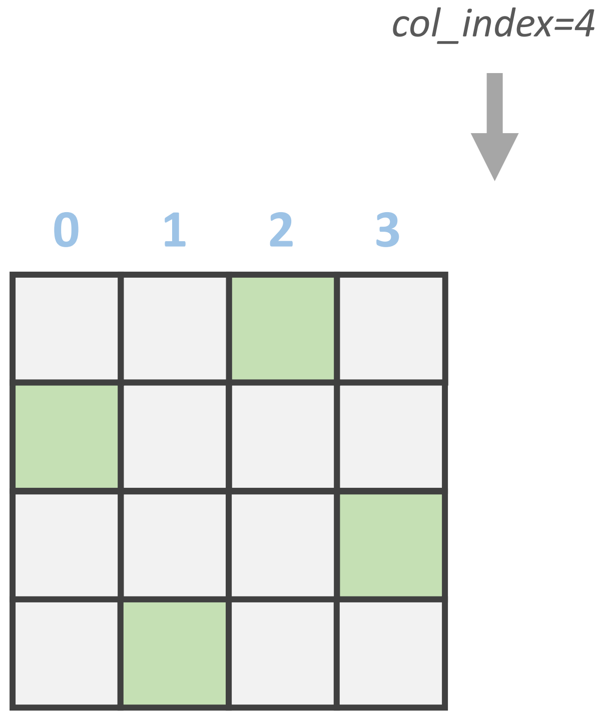


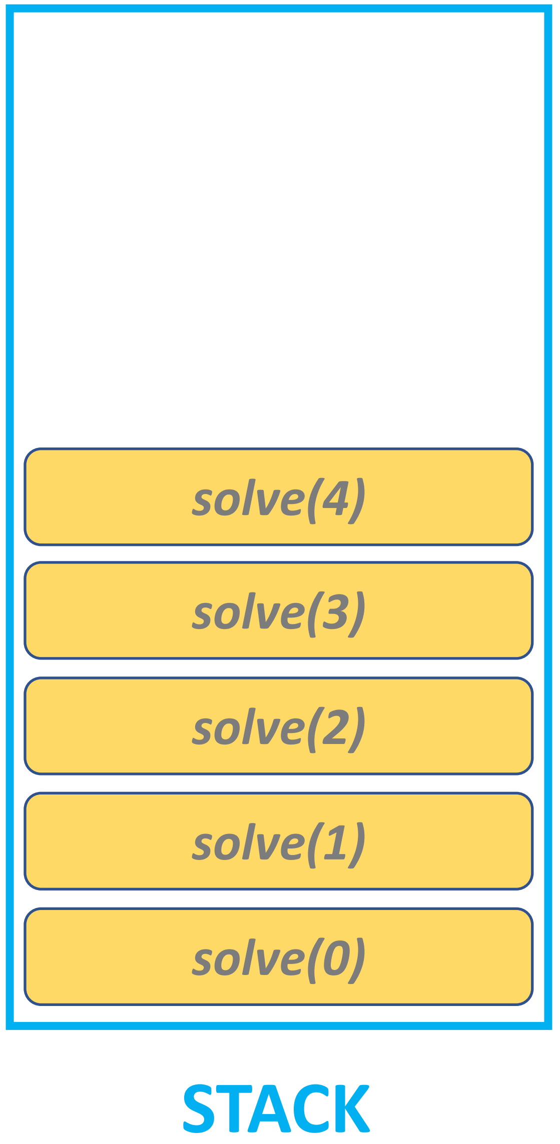


STACK

N-Queens Problem

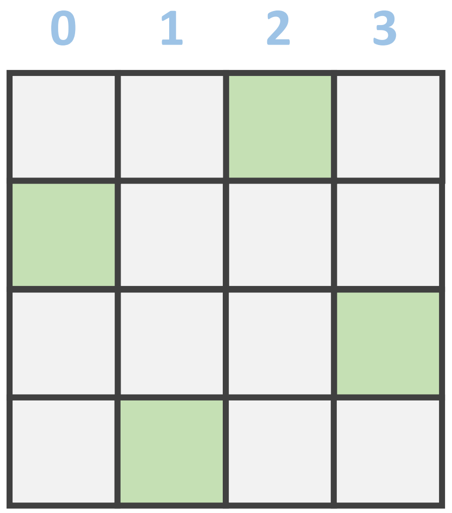
```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

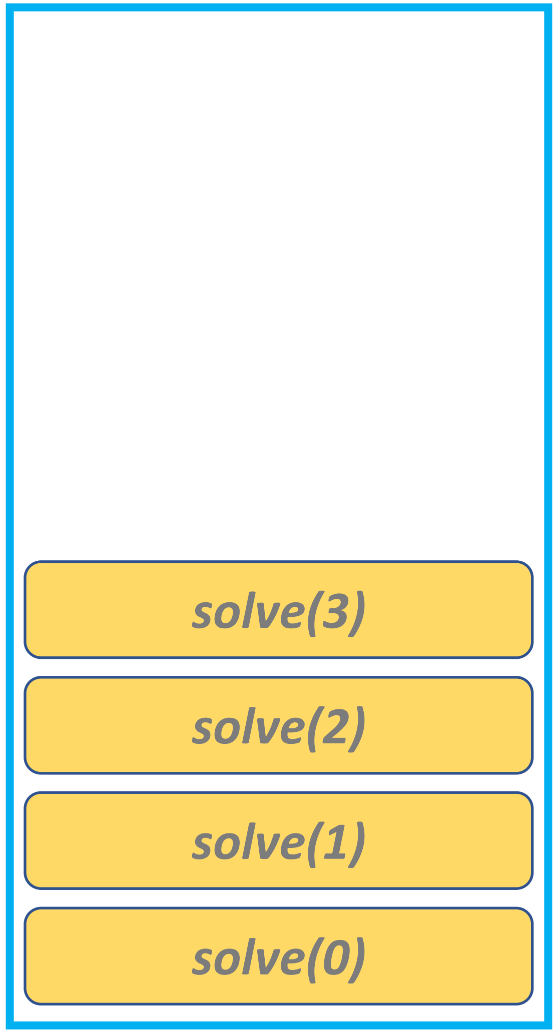




N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

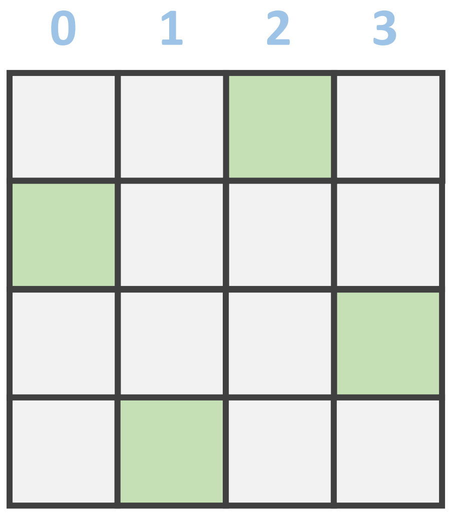


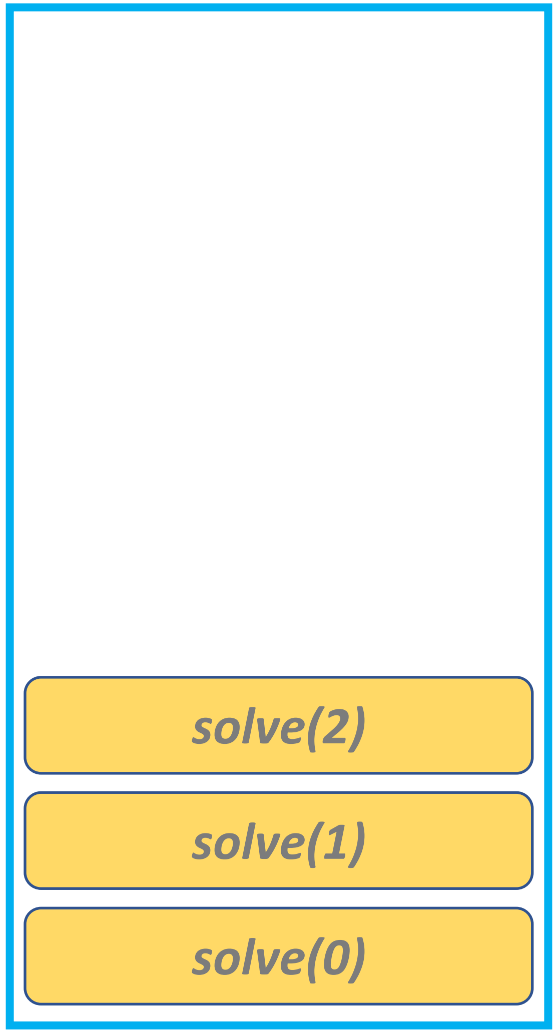


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

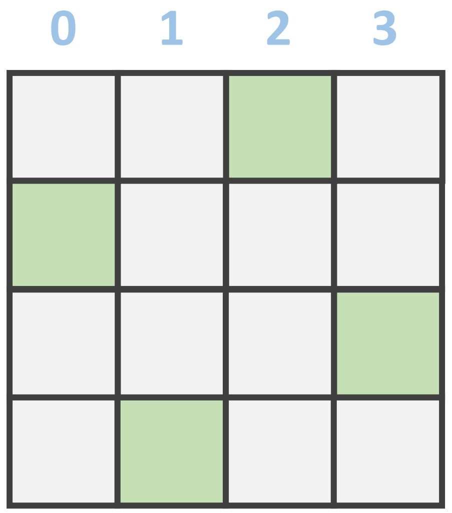


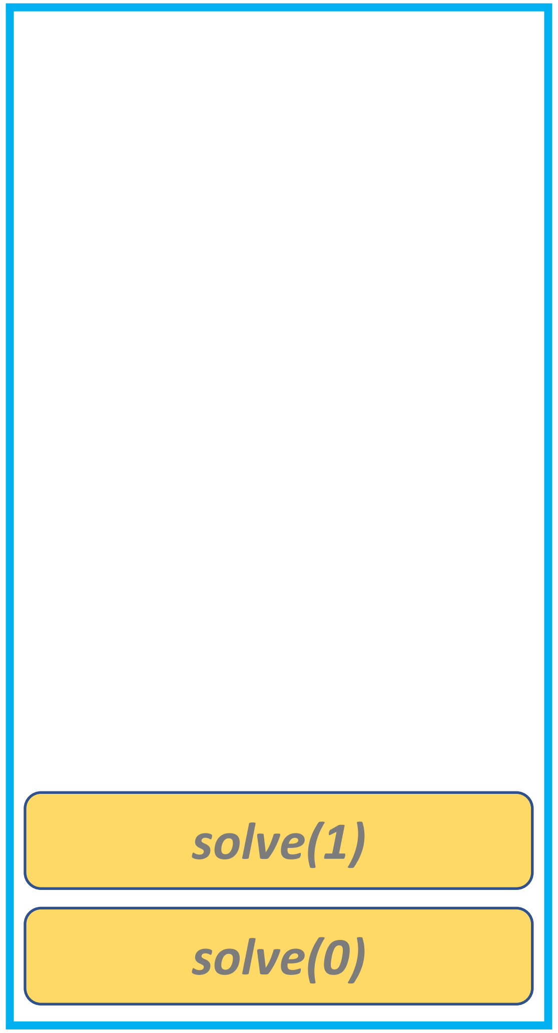


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```

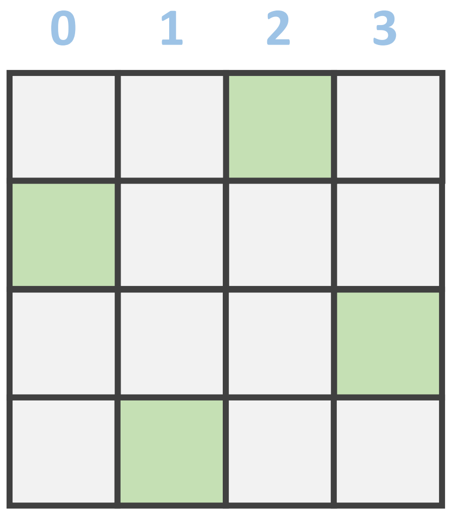


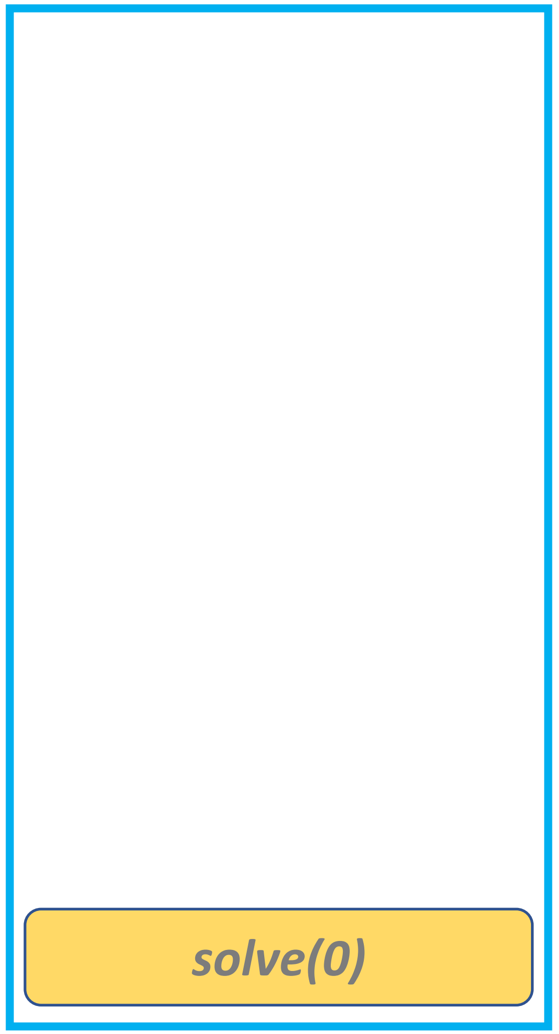


STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```





STACK

N-Queens Problem

```
solve(col_index)

    if col_index == number of queens
        return True

    for row_index in given column
        if is_place_valid(row_index, col_index)
            set cell green

            if solve(col_index+1)
                return True

            set cell empty again

    return False
```