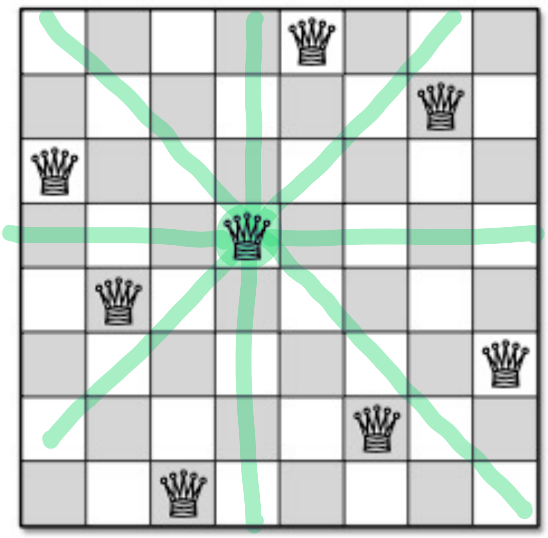
HW5 due today 8pm
HW6 due in one week
GPS6 due Monday

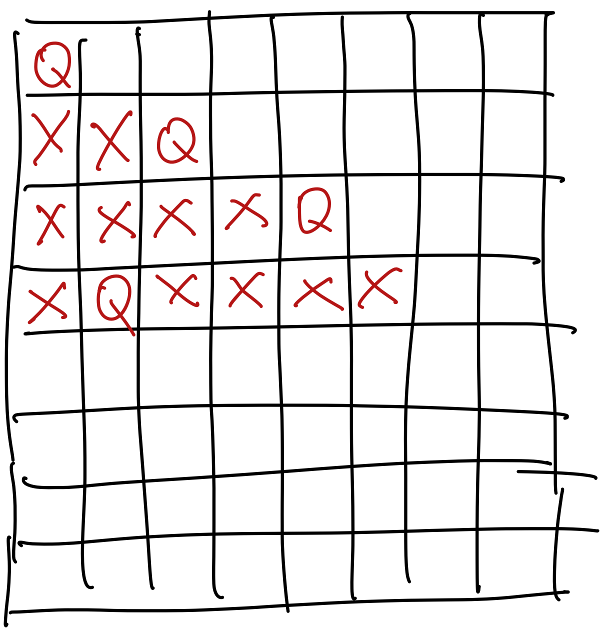
Expect to release MT1 Monday
Drop deadline next Friday

Gauss ⟶ 92 solutions to 8 Queens

"methodisches Tatonirren"



⟶ Sudoku



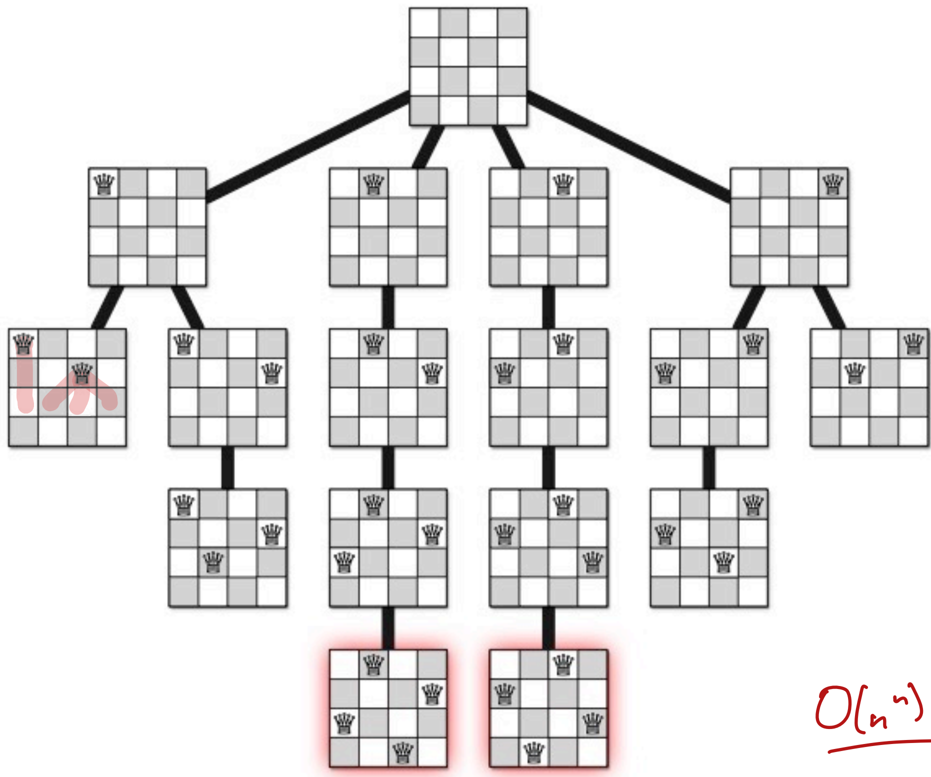
? ⟶
? ⟶

Recursive Backtracking

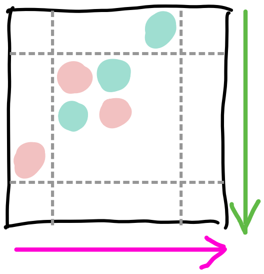
PlaceQueens(Q, r):
  Print all possible ways to
  Place queens in rows r thru n
   given locations Q[1..r] of queens
    in rows 1..r−1.

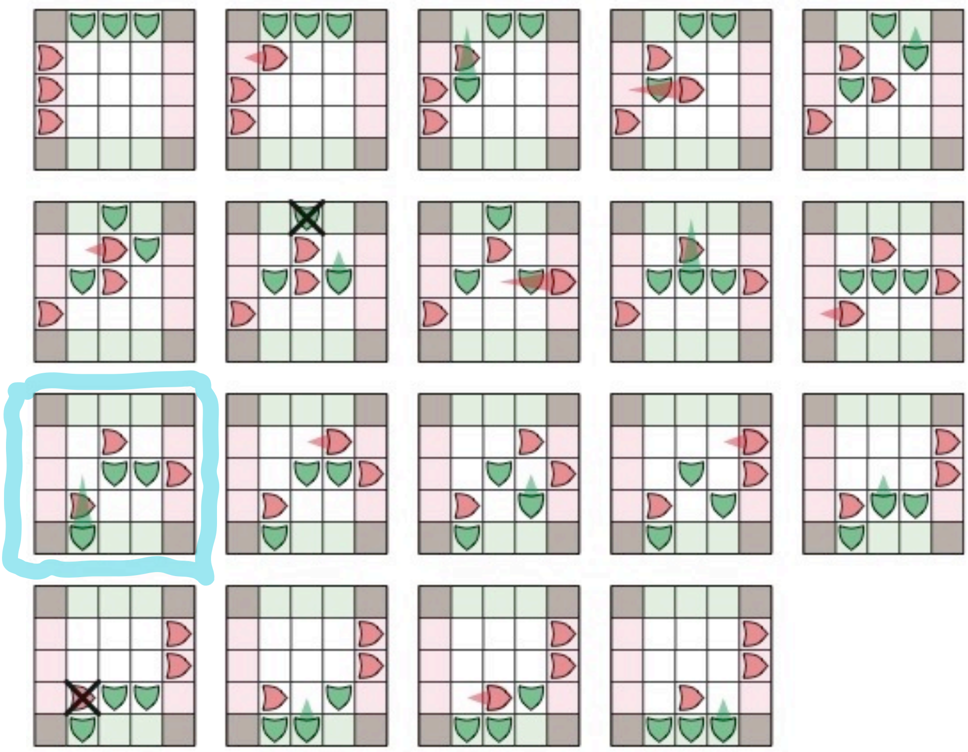
```
PLACEQUEENS(Q[1..n], r):
    if r = n + 1
        print Q[1..n]
    else
        for j ← 1 to n
            legal ← TRUE
            for i ← 1 to r − 1
                if (Q[i] = j) or (Q[i] = j + r − i) or (Q[i] = j − r + i)
                    legal ← FALSE
            if legal
                Q[r] ← j
                PLACEQUEENS(Q[1..n], r + 1)        ⟨⟨Recursion!⟩⟩
```

Figure 2.2. Gauss and Laquière's backtracking algorithm for the n queens problem.



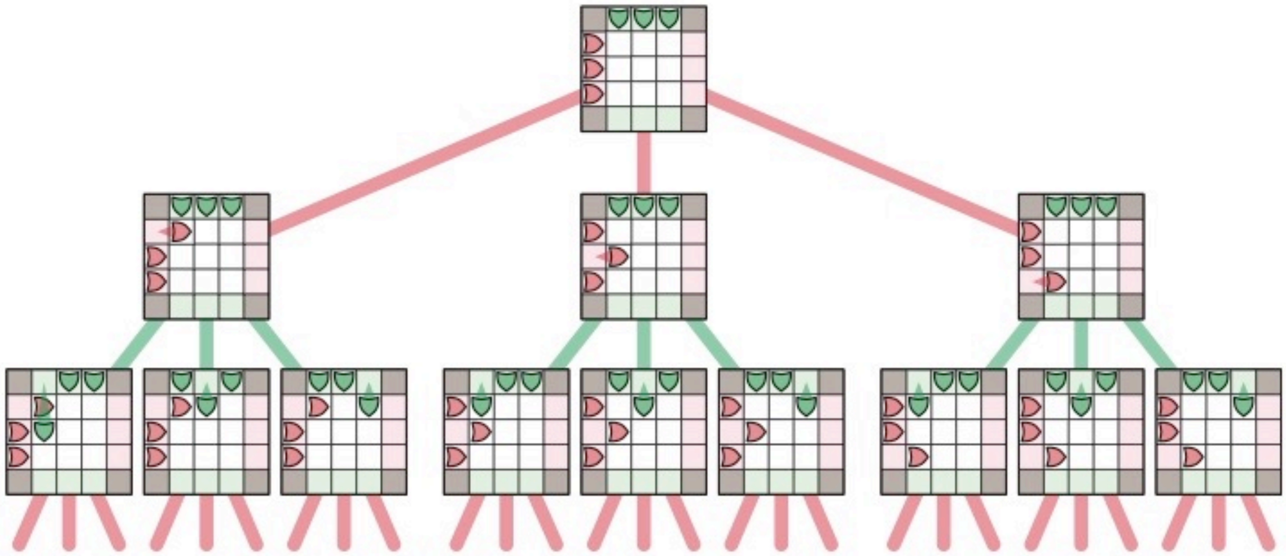
$O(n^n)$ time





A game $\underline{\text{state}}$ = positions of all pieces
+ who goes next

A game state is $\underline{\text{good}}$ iff
- current player has already won, or
- there is a move leaves opponent with a $\underline{\text{bad}}$ game state.



```
PlayAnyGame(X, player):
  if player has already won in state X
      return Good
  if player has already lost in state X
      return Bad
  for all legal moves X ⇝ Y
      if PlayAnyGame(Y, ¬player) = Bad
          return Good        ⟨⟨X ⇝ Y is a good move⟩⟩
  return Bad                 ⟨⟨There are no good moves⟩⟩
```

PRIMVSDIGNITASINTAMTENVISCIENTIANONPOTEST
ESSERESENIMSVNTPARVAEPROPEINSINGVLISLITTERIS
ATQVEINTERPVNCTIONIBUSVERBORVMOCCVPATAE

interpuncts

BLUEST | EMU | NITRO | BOT |

BLUE	STEM	UNIT	ROBOT	HEARTHANDSATURNSPIN

HE

HEAR

HEART

HEARTH

IsWord(w) → T/F

This algorithm repeats subproblems.

Memoize → remember results of each subproblem

```
SPLITTABLE(A[1..n]):
  if n = 0
      return TRUE
  for i ← 1 to n
      if ISWORD(A[1..i])
          if SPLITTABLE(A[i+1..n])
              return TRUE
  return FALSE
```

```
⟨⟨Is the suffix A[i..n] Splittable?⟩⟩
SPLITTABLE(i):
  if i > n
      return TRUE
  for j ← i to n
      if ISWORD(i, j)
          if SPLITTABLE(j+1)
              return TRUE
  return FALSE
```

IsWord(A[i..j])
O(n) time

$$T(n) = O(n^2) + \sum_{i=1}^{n-1} T(n-i)$$
$$= O(2^n)$$

How many different ways can we call this function?

$\boxed{O(n)}$

Ignoring recursion, How long does this run?

$\boxed{O(n^2)}$

$\boxed{O(n^3)}$

— What do you need to remember about the past?

— What problem are solving to make future decisions?

Recurse!