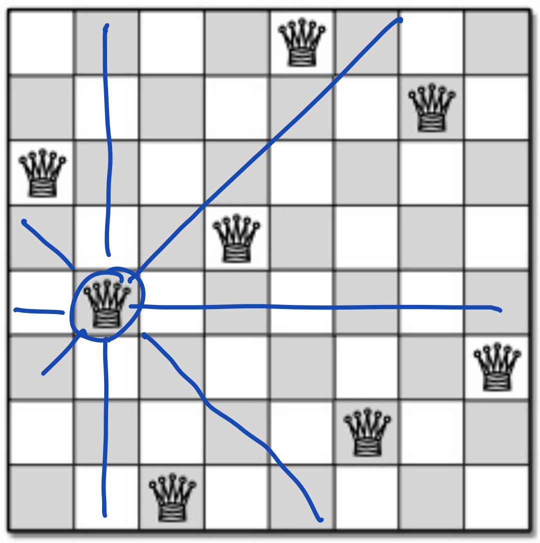
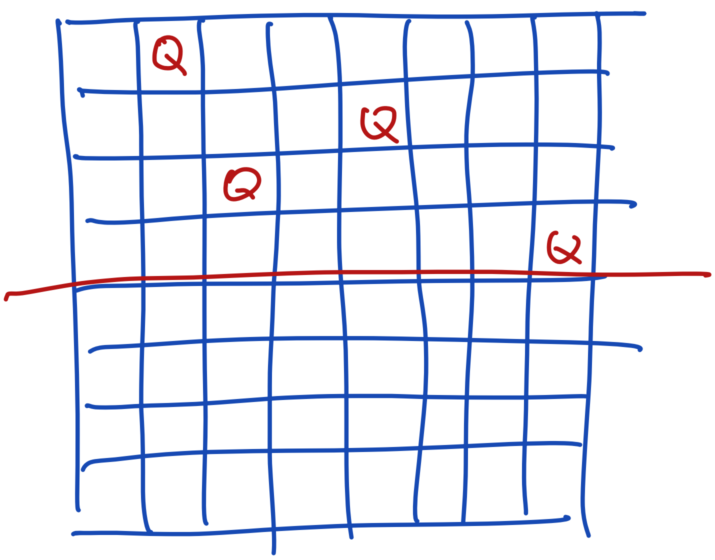
Backtracking → Dynamic Programming

FIRST MAKE IT WORK. THEN MAKE IT FAST.

~~8~~ queens (Gauss) "methodisches Tattoniren"

n





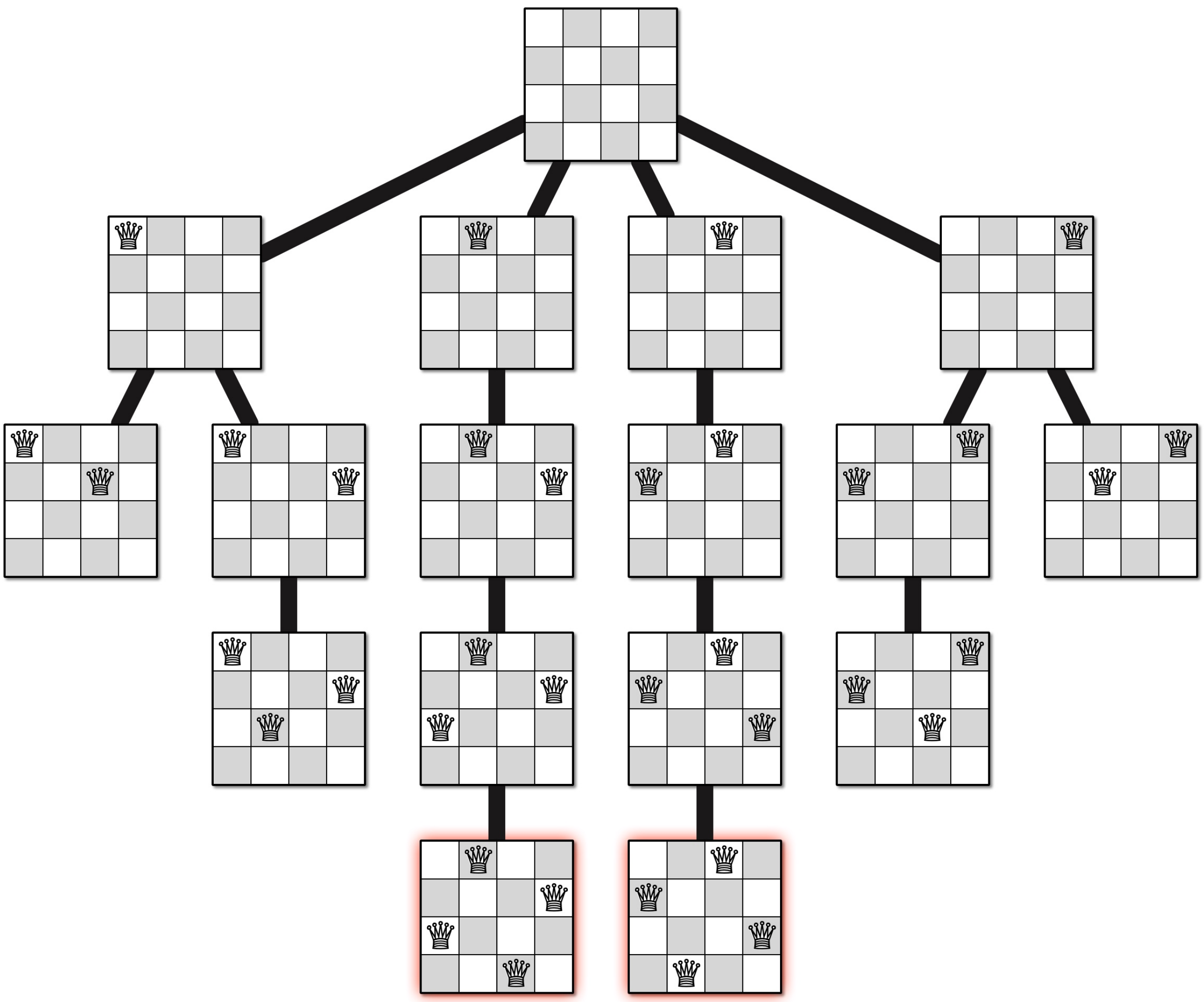
$r = \cancel{4}\ 5$

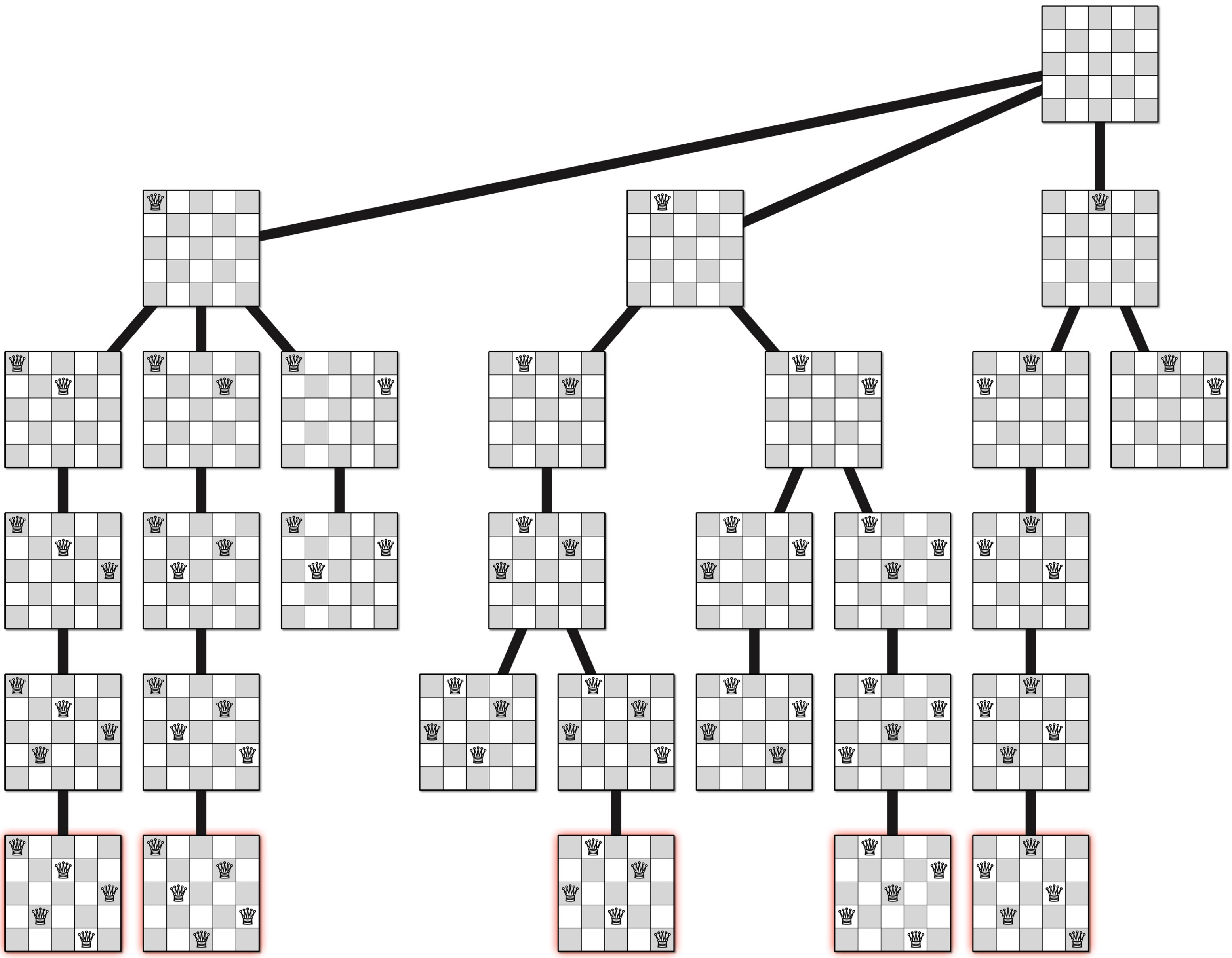
$Q = [2, 5, 3, 8]$

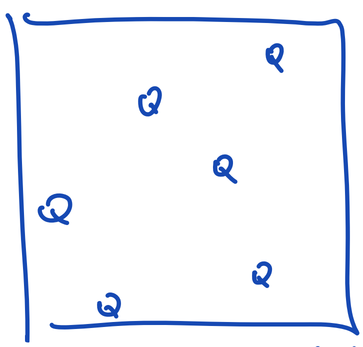
How many ways are there to place n queens on an $n \times n$ board, if queens are already on first $r-1$ rows at positions $Q[1..r-1]$?

```
PLACEQUEENS(Q[1..n], r):
  if r = n + 1
      print Q[1..n]
  else
      for j ← 1 to n        ← column index
          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!⟩⟩
```

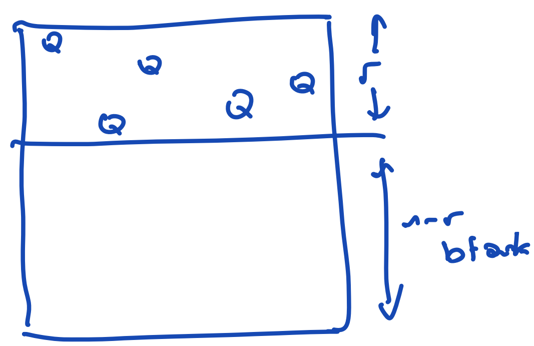
Figure 2.2. Gauss and Laquière's backtracking algorithm for the n queens problem.





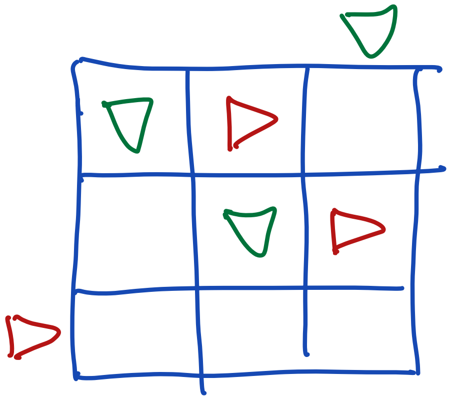


n-queens completion is NP-hard



top-down n-queens completion

OPEN



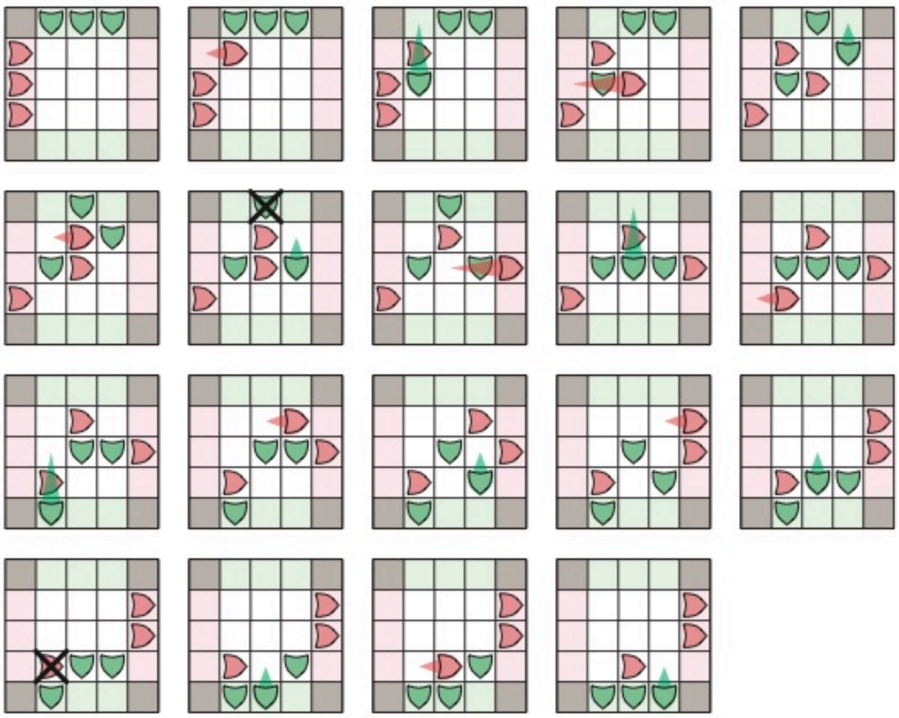
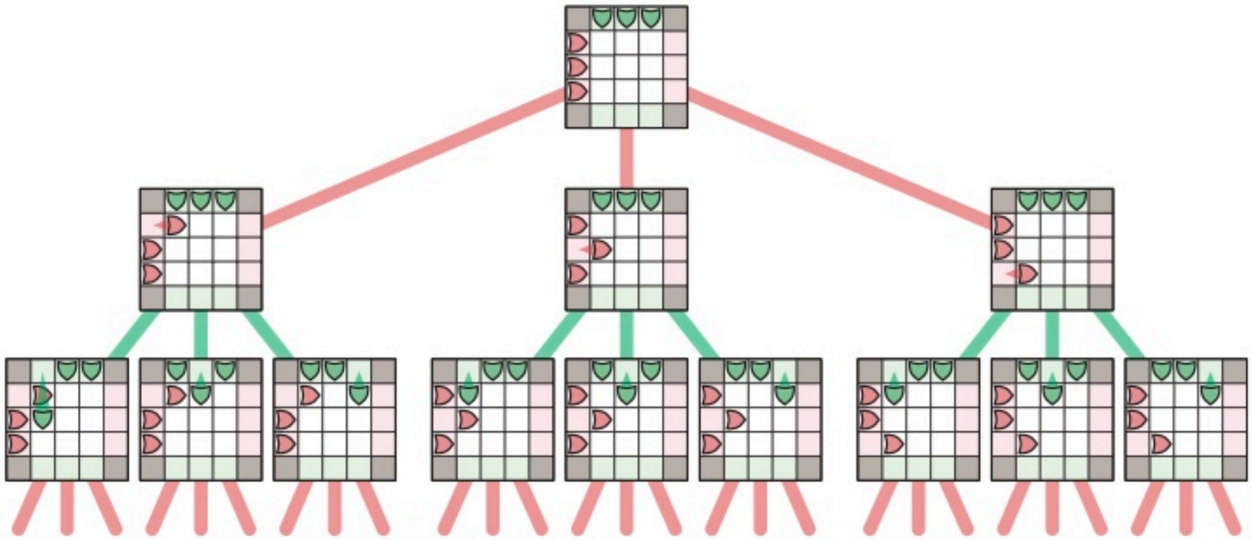


Figure 2.4. Vera wins the 3 × 3 fake-sugar-packet game.

Game tree



Game state = positions of all pieces

NOT FULL HISTORY

```
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

Given a string A[1..n], is A the concat of words?

IsWord(w) → True if w is a word
→ False o/w

Is the suffix A[i..n] the concat of words?

```
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
```

$$Splittable(i) = \begin{cases} \text{TRUE} & \text{if } i > n \\ \bigvee_{j=i}^{n} \left(\text{IsWord}(i,j) \wedge Splittable(j+1)\right) & \text{otherwise} \end{cases}$$

```
⟨⟨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
```

$O(2^n)$ time

But Only n different ways to call this function

Write down results! ⇒ $O(n^2)$ time (calls to IsWord)