cin.ufpe.br

Algoritmos e Estruturas de Dados

Backtracking

Busca completa

vs

Backtracking

Busca completa

Gera TODAS as soluções possíveis,
verifica se existe uma válida

Backtracking

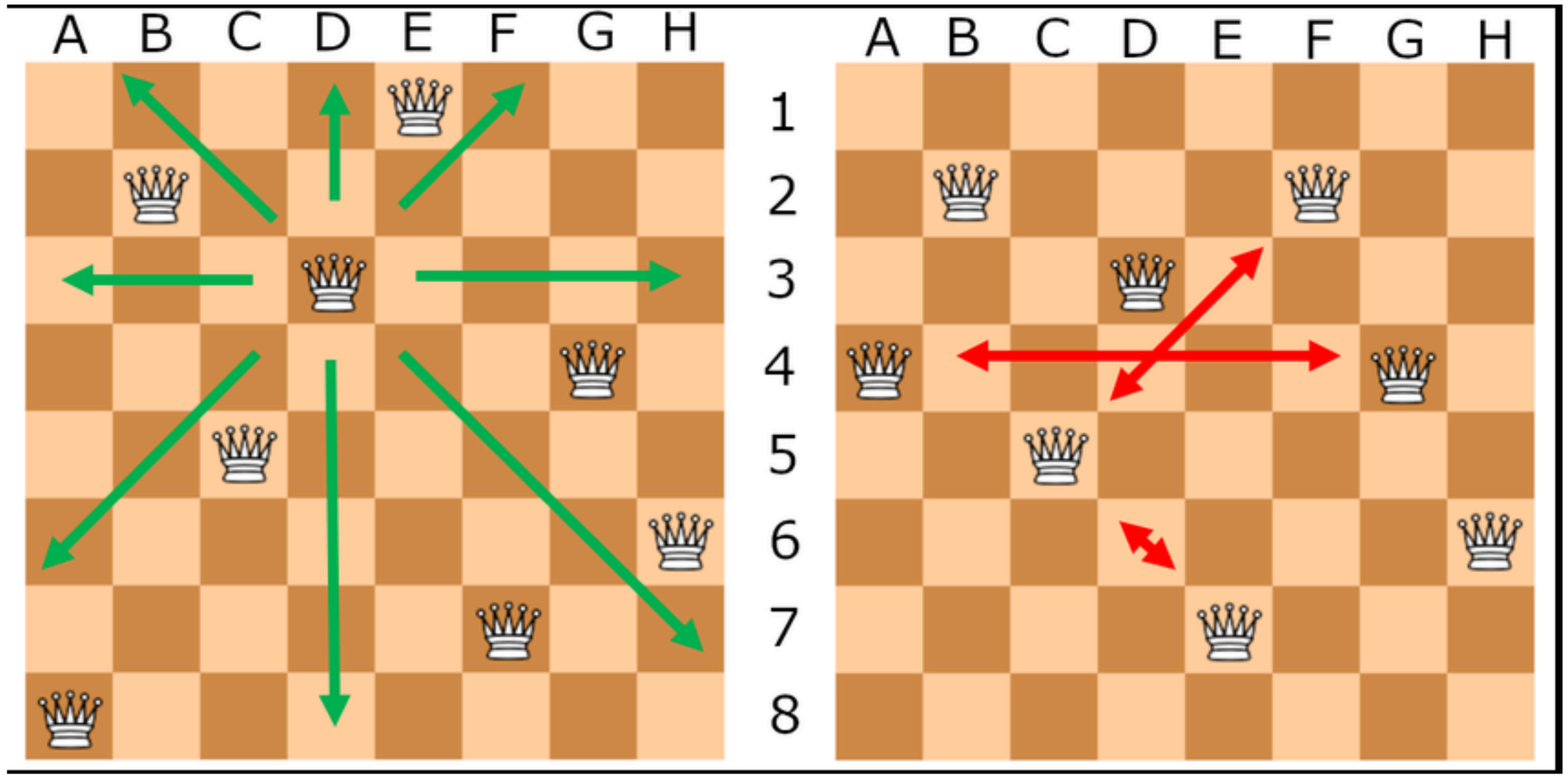
Tenta gerar todas as soluções possíveis, não vai até o final quando percebe que determinada solução não vai funcionar

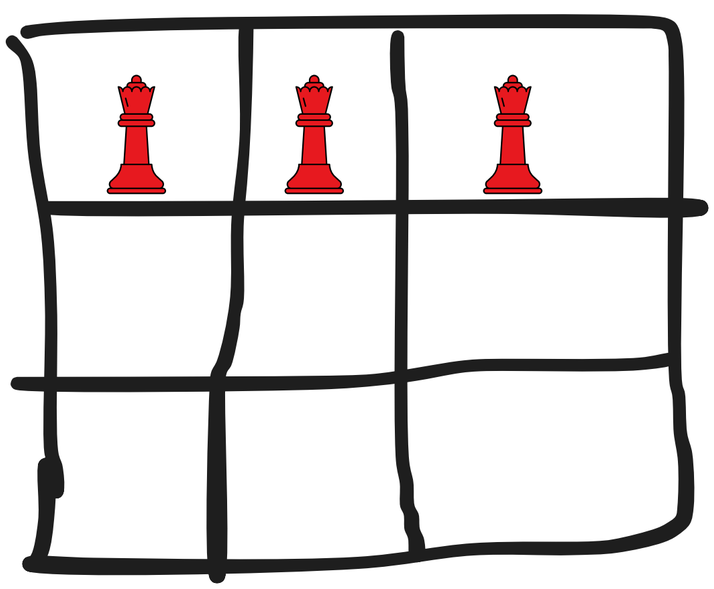
cin.ufpe.br

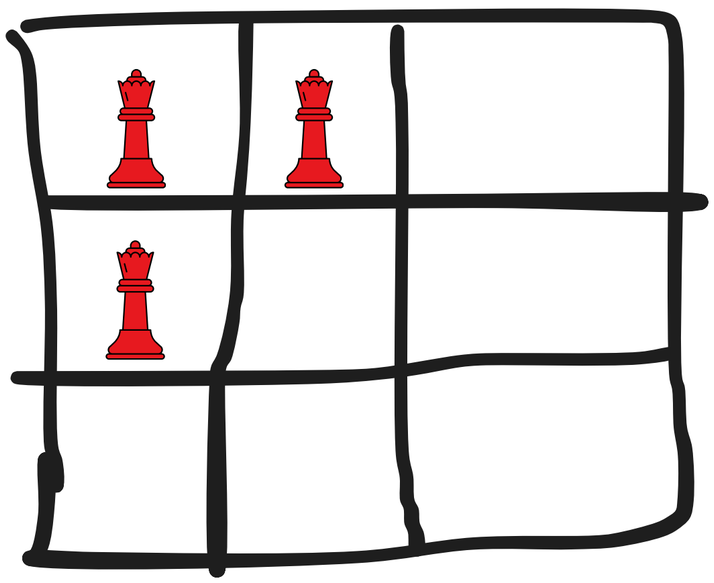
Problema das n rainhas

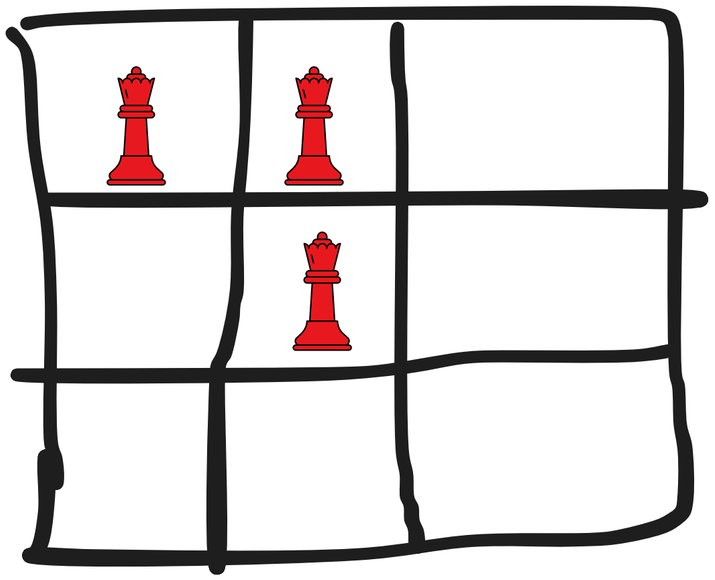
Dado um tabuleiro de dimensões **n** x **n**,

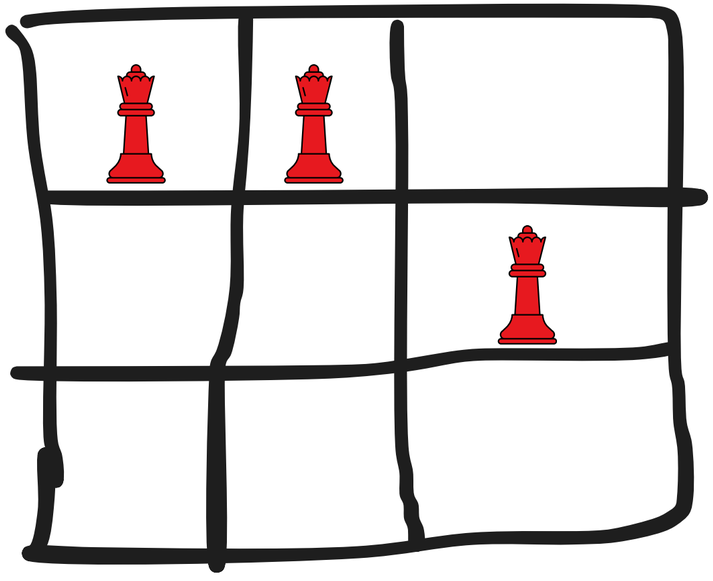
De quantas formas posso posicionar **n** rainhas tal que elas não se atacam?

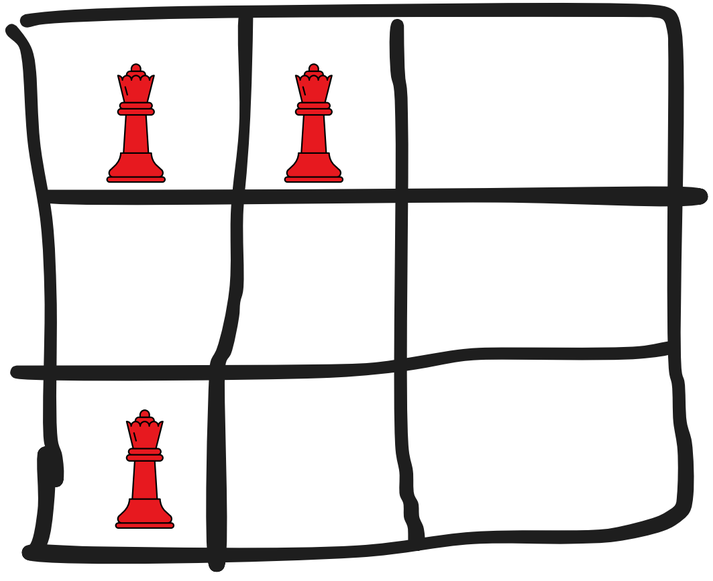

A B C D E F G H
1
2
3
4
5
6
7
8
A B C D E F G H

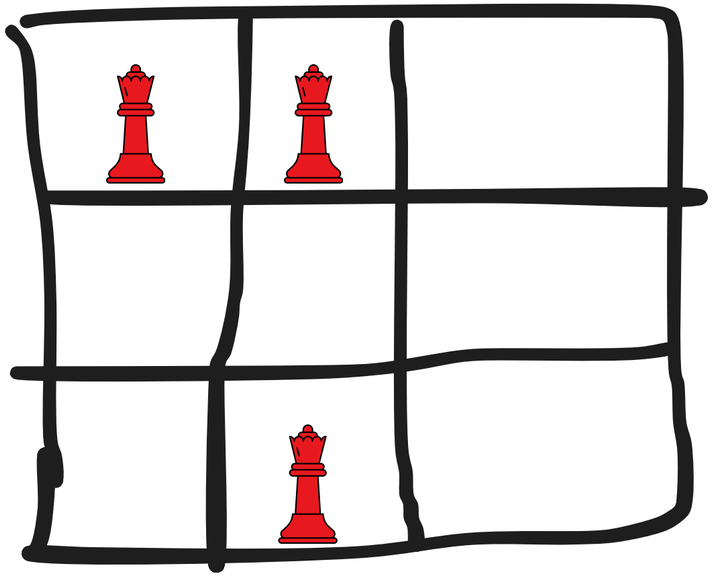


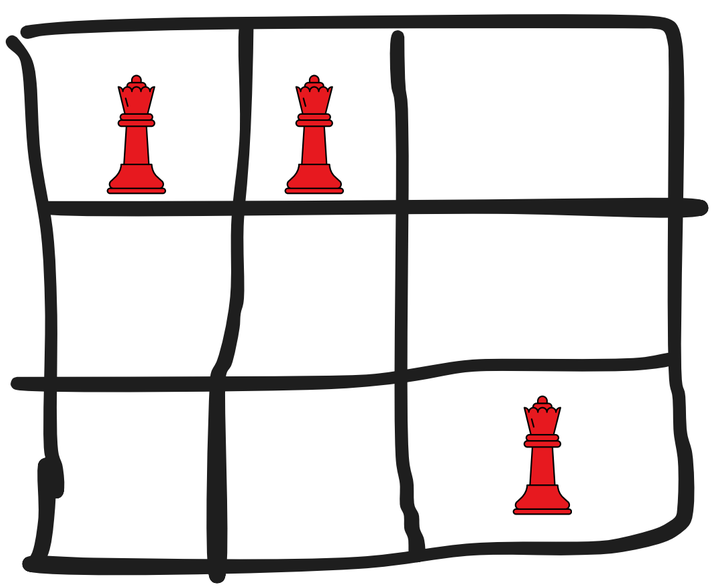


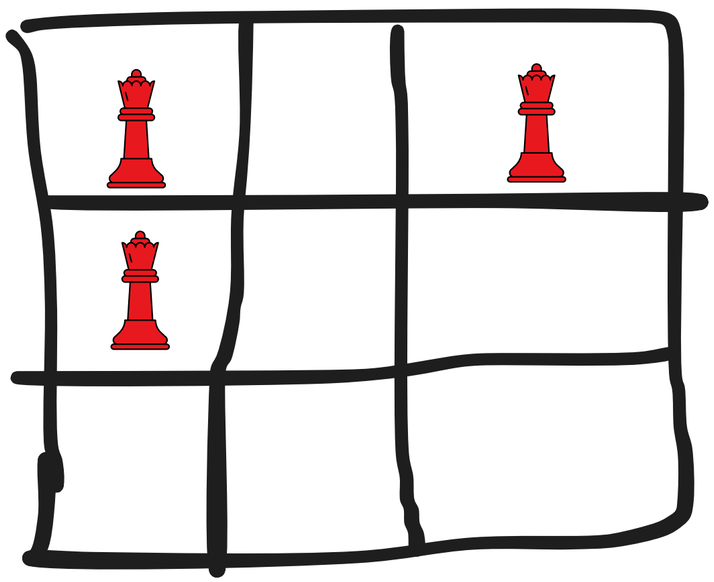


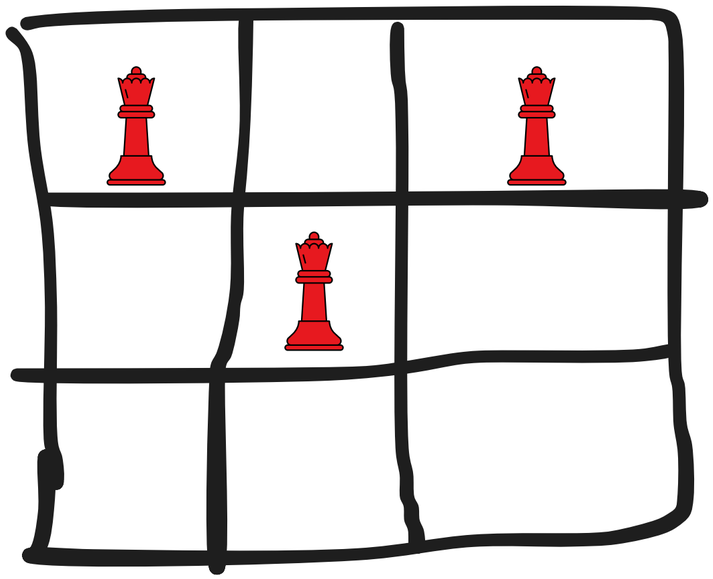


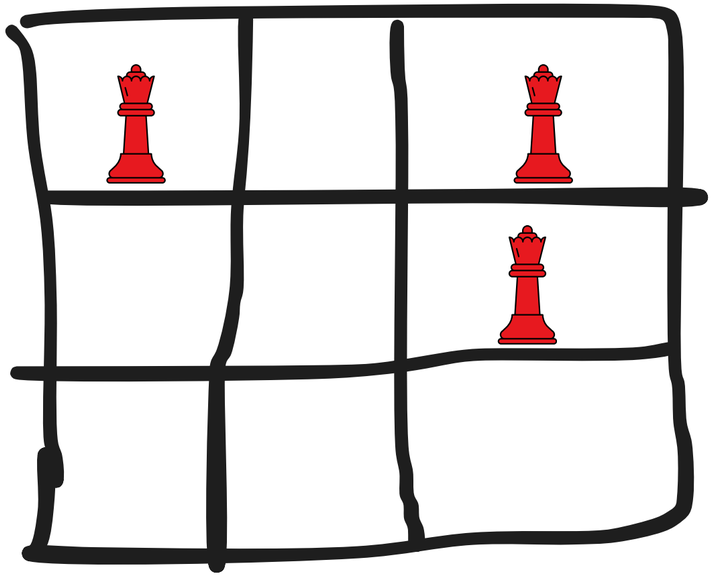


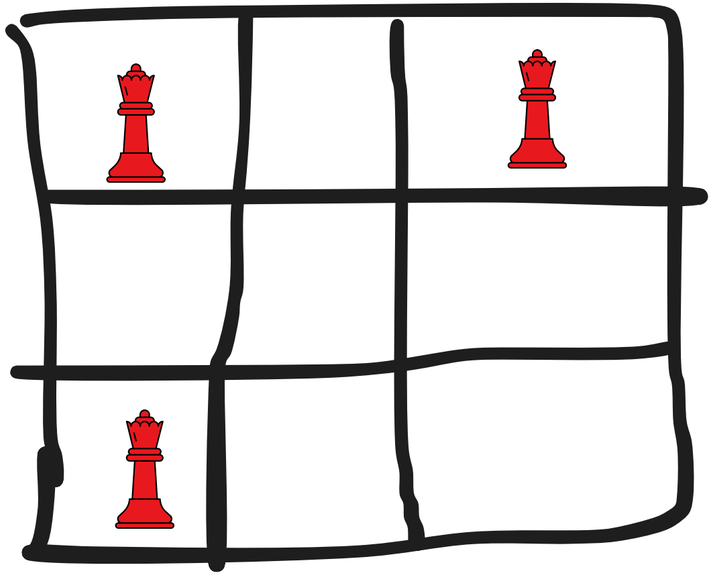


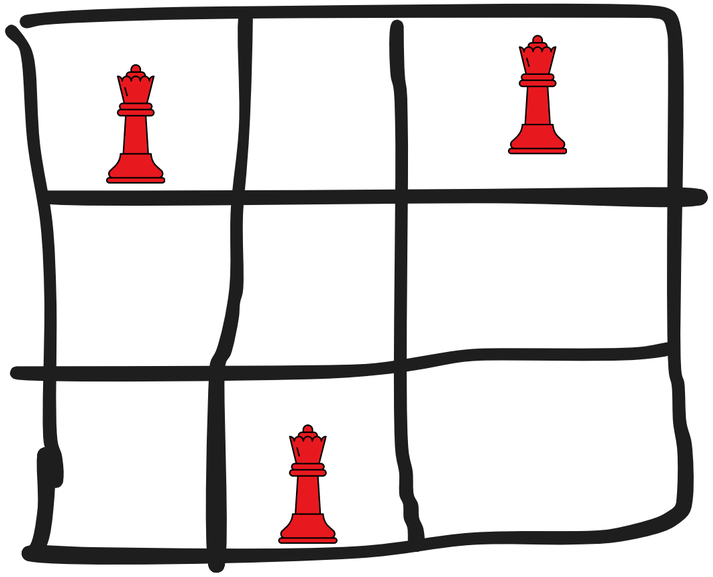












cin.ufpe.br

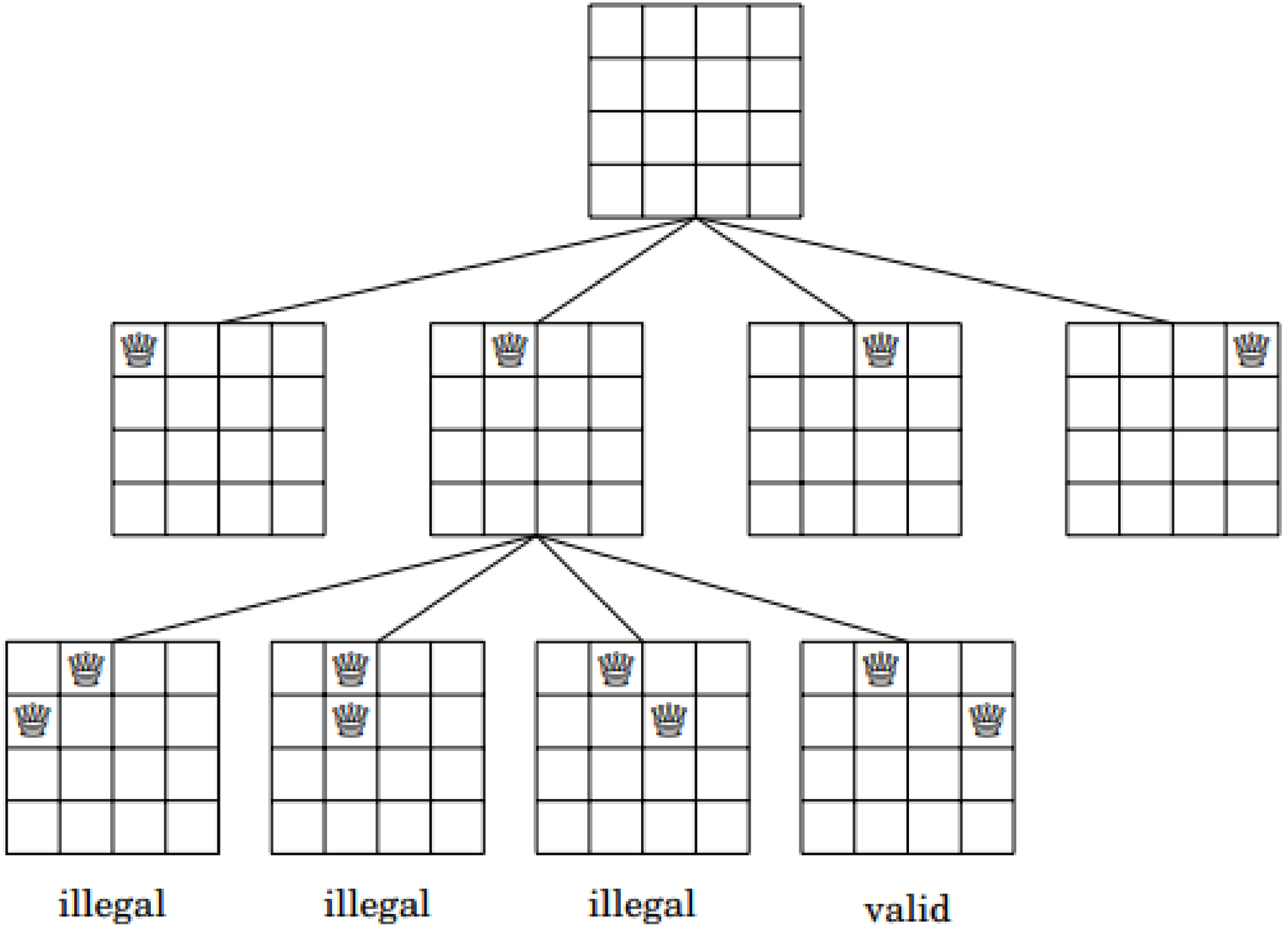
Complexidade?

$$\binom{n^2}{n}$$

n	$f(n) = \binom{n^2}{n}$	tempo
3	84	0 ms
5	53.130	0.5 ms
8	4.426.165.368	44 s
20	2.788.360.983.670.896.737. 872.851.072.994.080	$7 * 10^7$ * idade do universo

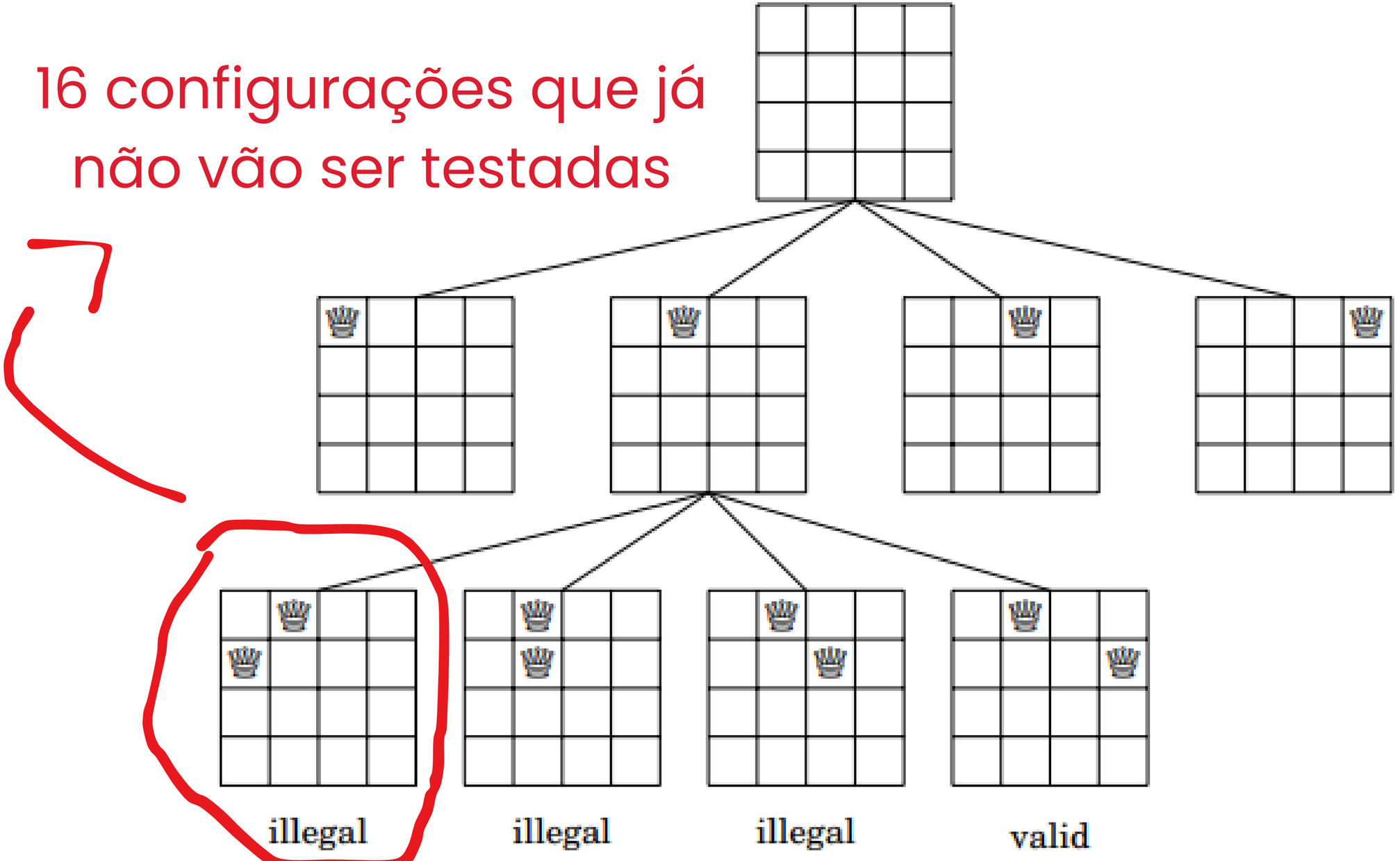
Abordagem backtracking

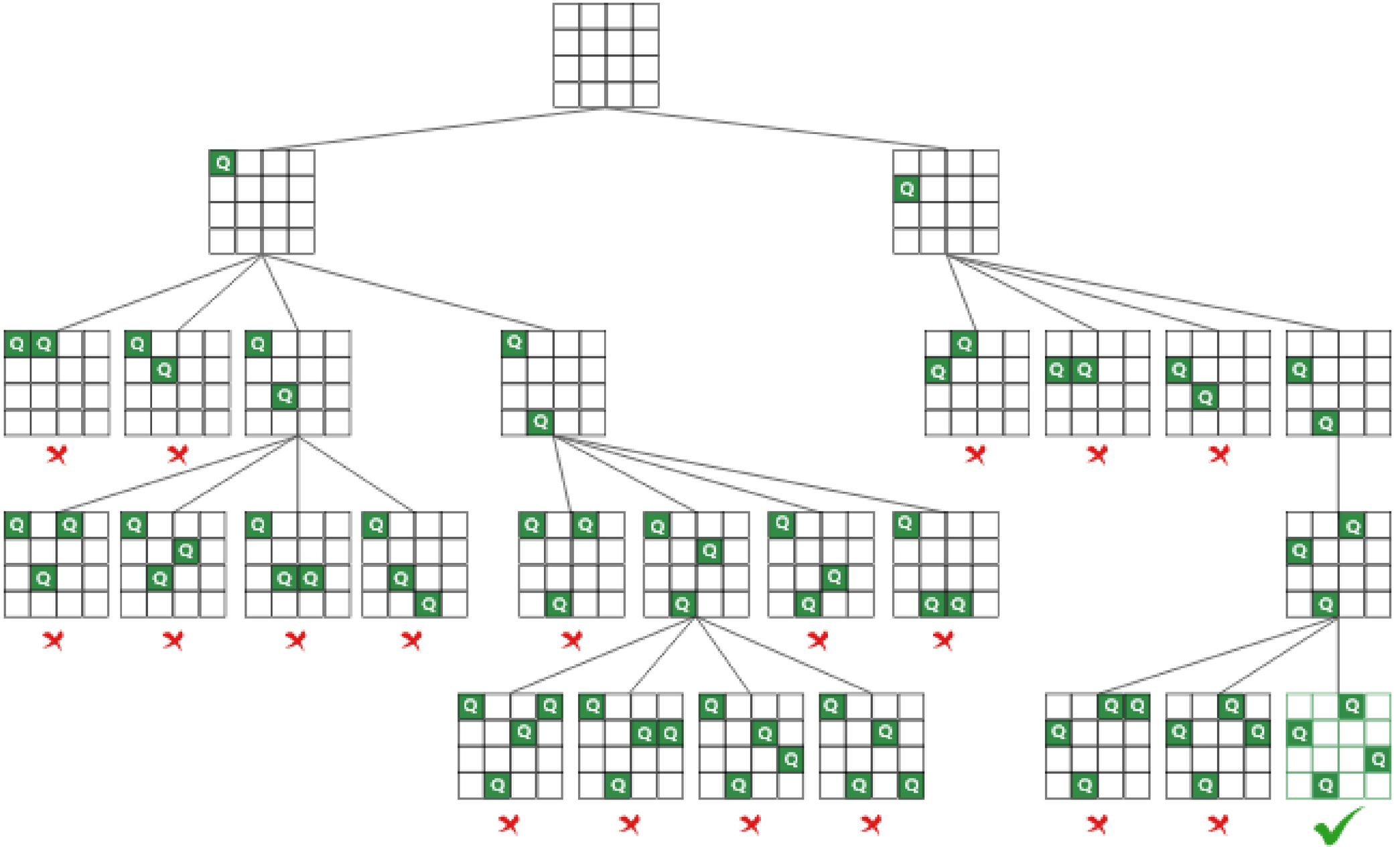
Abordagem backtracking



Abordagem backtracking

16 configurações que já não vão ser testadas





Testes de posições para n = 8:

n	busca completa	backtracking
8	4.426.165.368	1792

...Continua ineficiente

Mission Accomplished

This FPGA computation has raised the bar:

Q(27) = 234907967154122528

Seven years after the computation of Q(26), this project lets FPGAs prevail again.

News

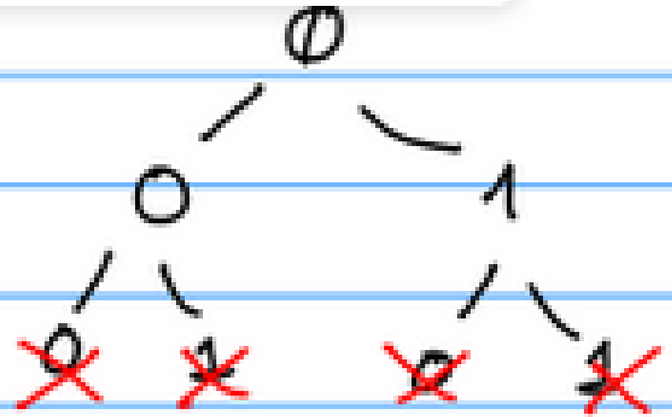
Sep 19, 2016: Mission completed - there are **234907967154122528** solutions of the 27-Queens Puzzle. 29363791967678199, i.e. very slightly more than an eighth, of them had actually to be discovered by the computation running for slightly more than a year.

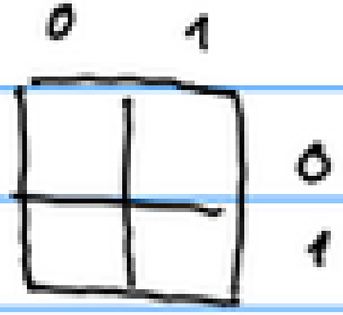
No ***Problema das n-rainhas***, temos um tabuleiro de xadrez de tamanho $n \times n$, sobre o qual desejamos posicionar n rainhas de forma que nenhuma delas seja "ameaçada" por nenhuma das demais, ou seja, não pode haver duas peças na mesma linha, coluna ou diagonal. Para um tabuleiro padrão 8×8, existem mais de $4,4$ bilhões de possibilidades de posicionamento das peças, porém a maioria delas não é válida. Entretanto, supondo as linhas do tabuleiro numeradas $0, \ldots, n-1$ de cima para baixo, e as colunas numeradas no mesmo intervalo da esquerda para a direita, esse problema pode ser facilmente resolvido com um algoritmo de ***backtracking***, escolhendo progressivamente as colunas da rainhas das linhas $i = 0, \ldots, n-1$ (cada linha deve ter exatamente uma rainha), tentando cada uma das possibilidades $j = 0, \ldots, n-1$, e retrocedendo assim que um conflito é encontrado.

Determine o menor valor $n_{min} \geq 2$ para o qual o problema tem uma solução, ilustrando a execução do algoritmo acima através da sua árvore de busca para $n = 2, 3, \ldots, n_{min}$

N = 2)

Impossible





Branch and Bound

A cada passo, ela calcula um limite superior ou inferior para a função objetivo, permitindo descartar subproblemas inteiros que não podem melhorar a solução ótima atual.

Considere o problema da mochila (***0-1 Knapsack***) para a seguinte entrada:

Item	1	2	3	4	5
Peso (w)	4	3	1	2	2
Valor (v)	40	25	10	20	15

Capacidade da mochila: $K = 7$

■ **QUESTÃO 5** (2,0pt)

Ilustre a execução do algoritmo ***branch&bound*** para o problema da mochila da Questão 4, exibindo a árvore de solução com as cotas superiores em cada nó, e indicando claramente os pontos de backtracking. A árvore de solução é uma árvore binária em que cada nível corresponde à decisão sobre incluir ou não um item. Uma cota superior para um nó pode ser estimada considerando-se a soma parcial correspondente a esse nó mais os valores de todos os itens ainda não considerados.