"Não jogue xadrez com pombos!"

# Backtracking

Paulo Ricardo Lisboa de Almeida

DSBD

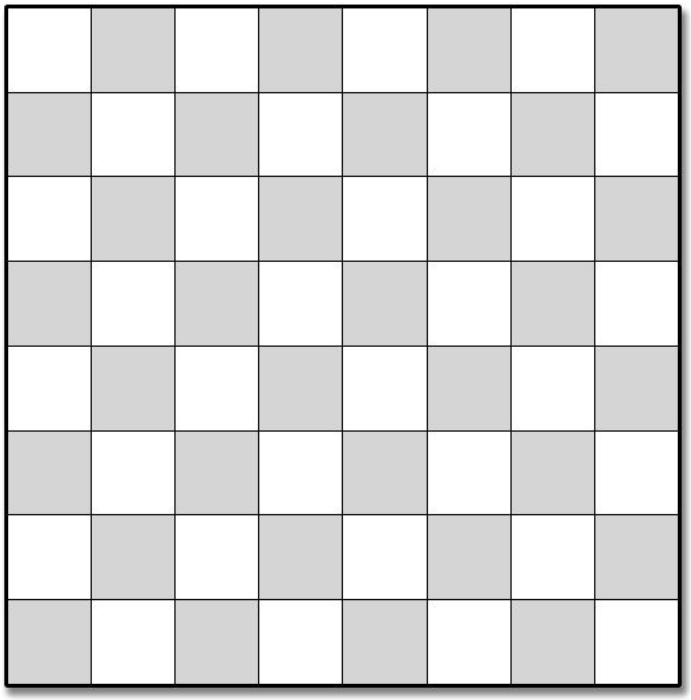
UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

# Os problema das 8 rainhas

Problema proposto por Max Bezzel em 1848

Considerando um tabuleiro de xadrez de 8x8, como dispor 8 rainhas de forma que uma rainha não possa atacar a outra?

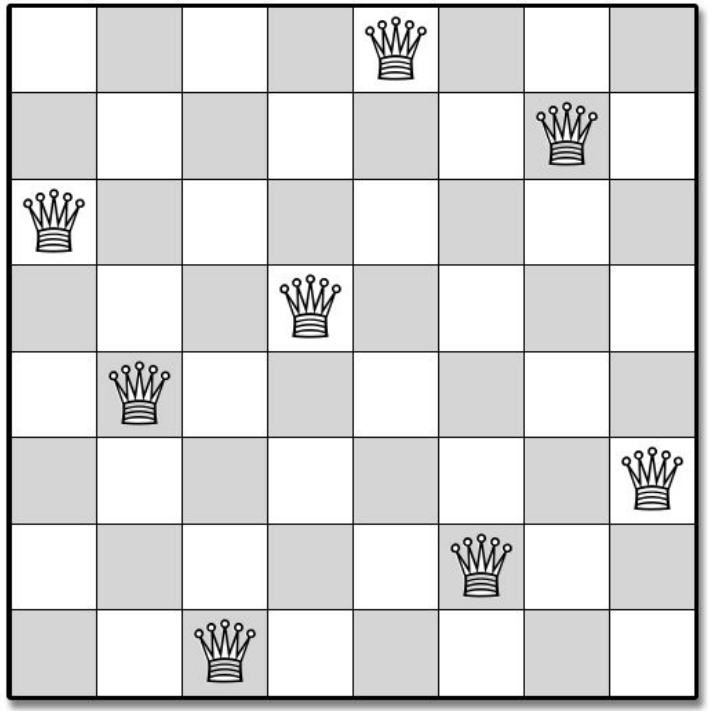
**Faça você mesmo**



# Os problema das 8 rainhas

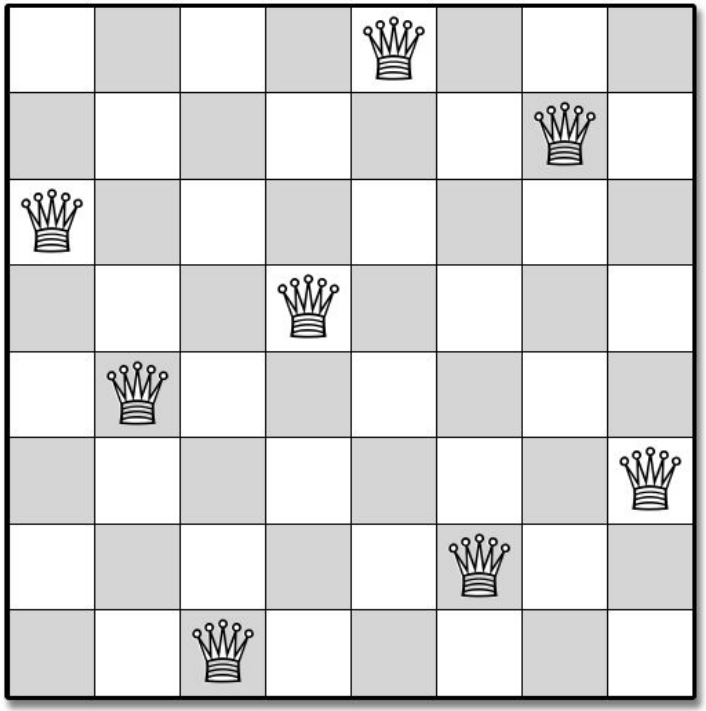
Problema proposto por Max Bezzel em 1848

Considerando um tabuleiro de xadrez de 8x8, como dispor 8 rainhas de forma que uma rainha não possa atacar a outra?



# Os problema das 8 rainhas

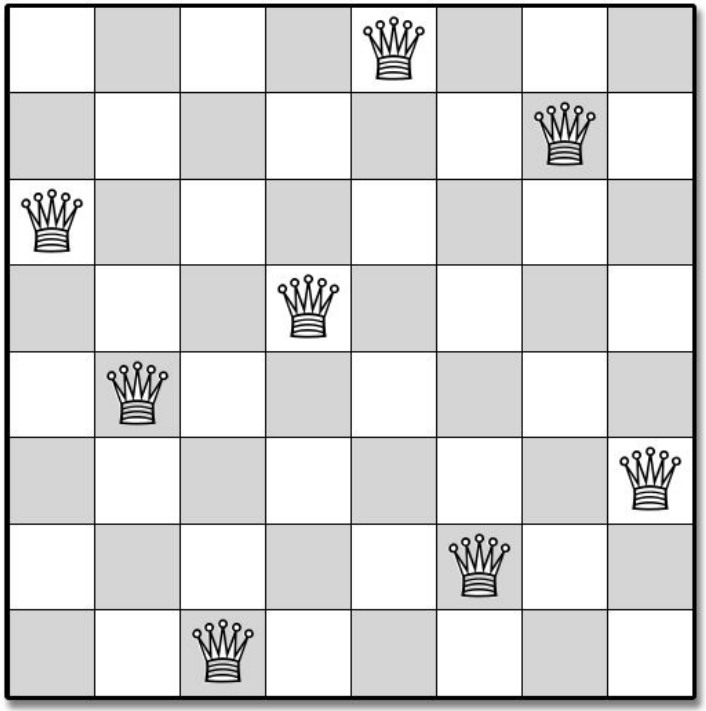
Existe mais de uma solução?



# Os problema das 8 rainhas

Existe mais de uma solução?

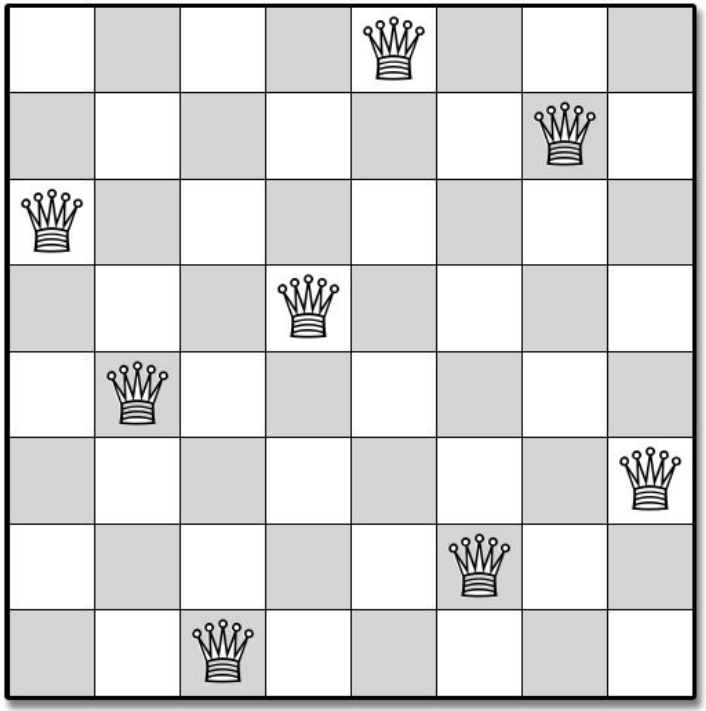
Para o tabuleiro de 8x8, existem 92 soluções.



# Os problema das 8 rainhas

Dada uma solução candidata para o problema, como verificar se ela é válida?

Como verificar se determinada rainha não está atacando outra?



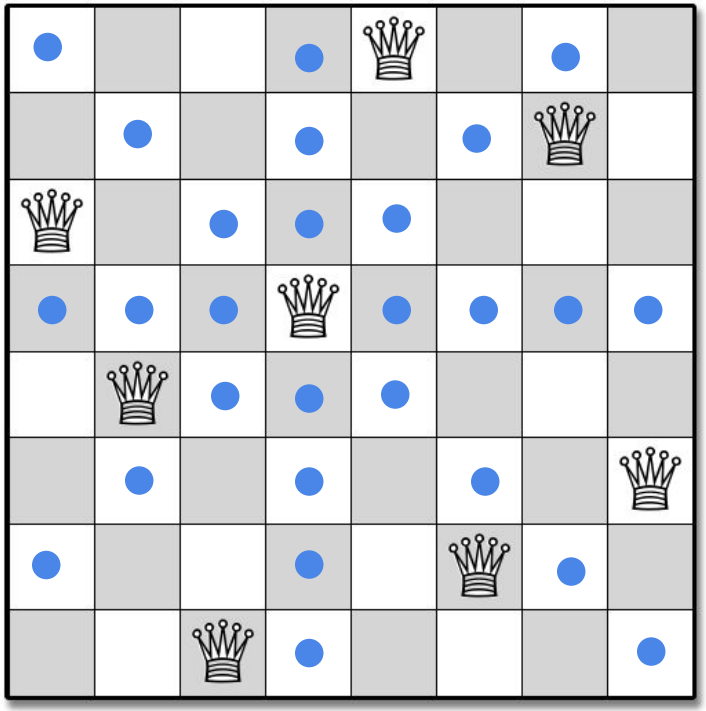
# Os problema das 8 rainhas

Dada uma solução candidata para o problema, como verificar se ela é válida?

Como verificar se determinada rainha não está atacando outra?

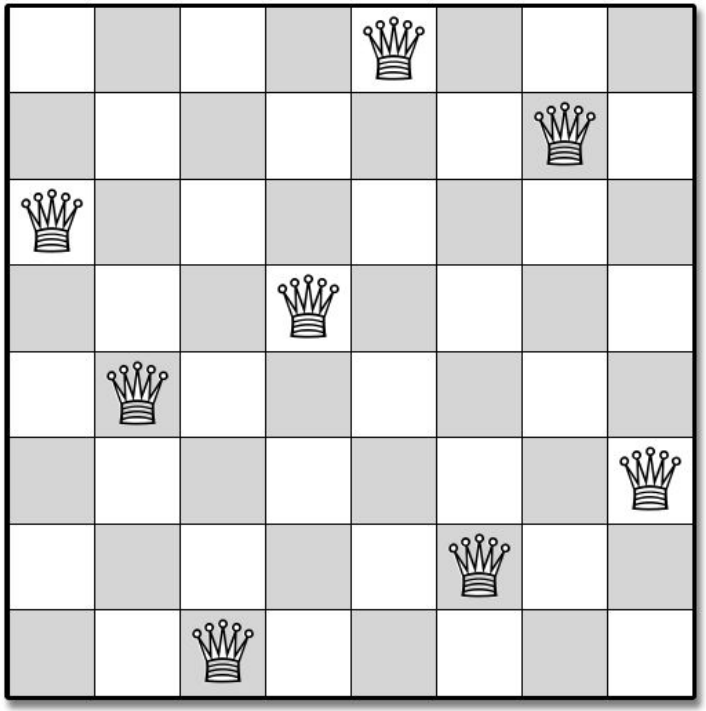
Basta procurar por outra rainha na mesma linha, coluna e diagonais da rainha atual

Se encontrar, a solução não é válida



# Os problema das 8 rainhas

Como criar um algoritmo que enumere as soluções para o problema?



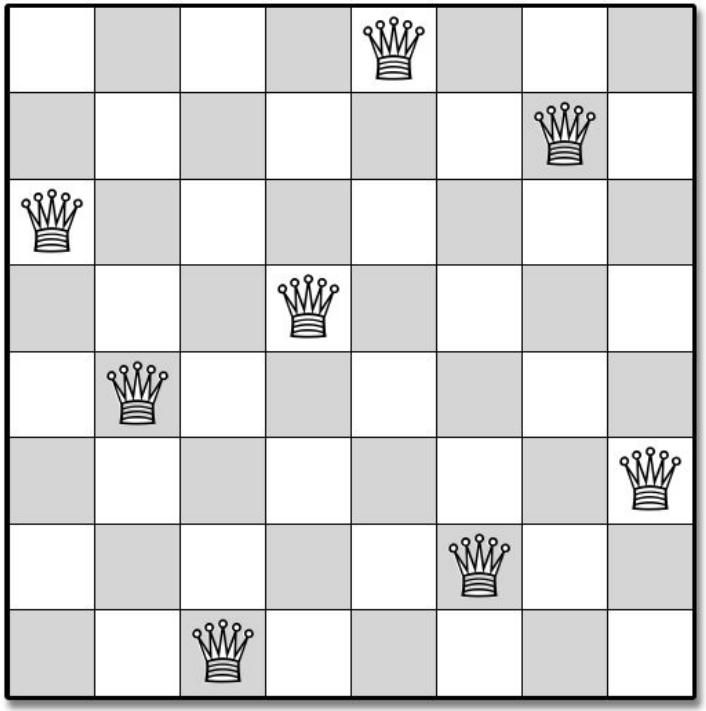
# Os problema das 8 rainhas

Como criar um algoritmo que enumere as soluções para o problema?

**Proposta:**

Gerar todos os tabuleiros 8x8 possíveis com 8 rainhas, e verificar para cada um se nenhuma rainha ataca outra.

**Qual o tamanho do problema?**



# Os problema das 8 rainhas

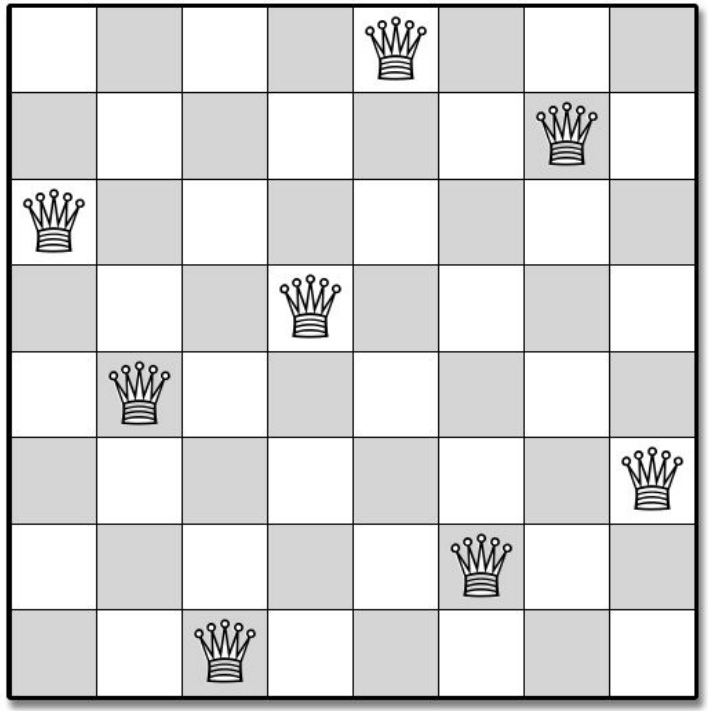
Como criar um algoritmo que enumere as soluções para o problema?

**Proposta:**

Gerar todos os tabuleiros 8x8 possíveis com 8 rainhas, e verificar para cada um se nenhuma rainha ataca outra.

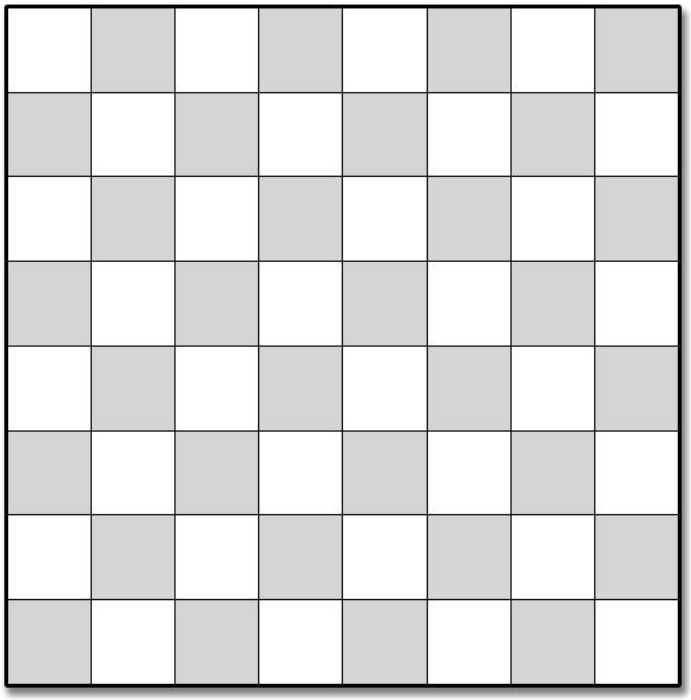
**Qual o tamanho do problema?**

Temos $\binom{64}{8} = 4.426.165.368$ combinações para explorar.



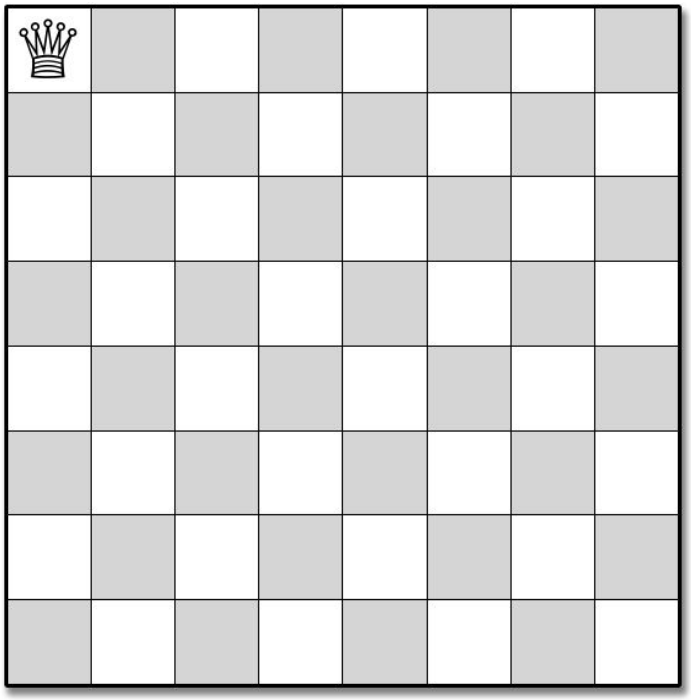
# Poda

Podemos utilizar alguma estratégia de poda.
Eliminar combinações que trivialmente sabemos que não vão gerar uma resposta válida.



# Poda

Ao inserir uma rainha em uma posição qualquer em uma linha, o que sabemos trivialmente que não podemos fazer?



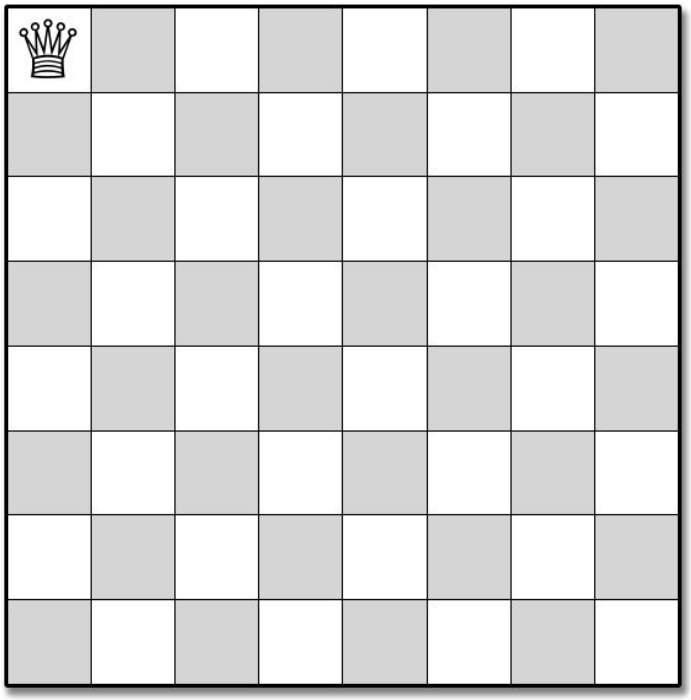
# Poda

Ao inserir uma rainha em uma posição qualquer em uma linha, o que sabemos trivialmente que não podemos fazer?

A próxima rainha precisa estar em outra linha.

Não podem existir duas ou mais rainhas na mesma linha.

**Quantas combinações?**



# Poda

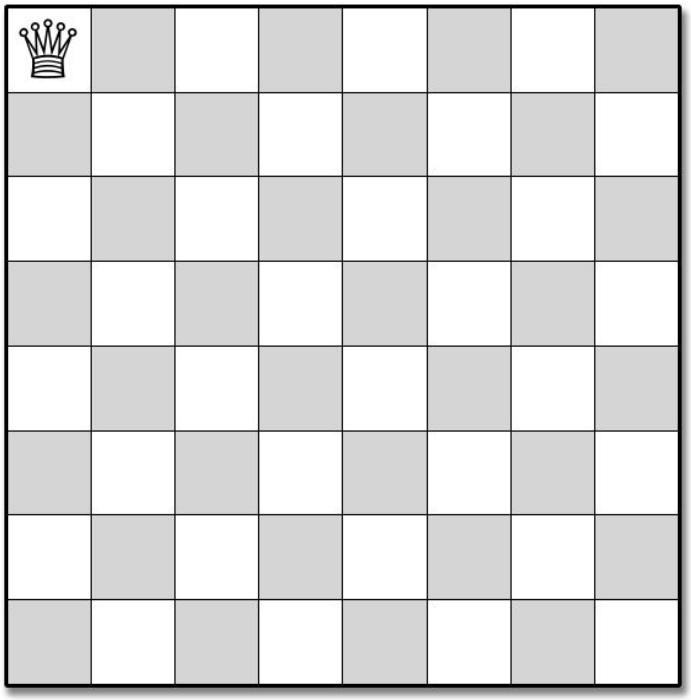
Ao inserir uma rainha em uma posição qualquer em uma linha, o que sabemos trivialmente que não podemos fazer?

A próxima rainha precisa estar em outra linha.

Não podem existir duas ou mais rainhas na mesma linha.

**Quantas combinações?**

$8^8 = 16.777.216$



# Poda

Ao inserir uma rainha em uma posição qualquer em uma linha, o que sabemos trivialmente que não podemos fazer?

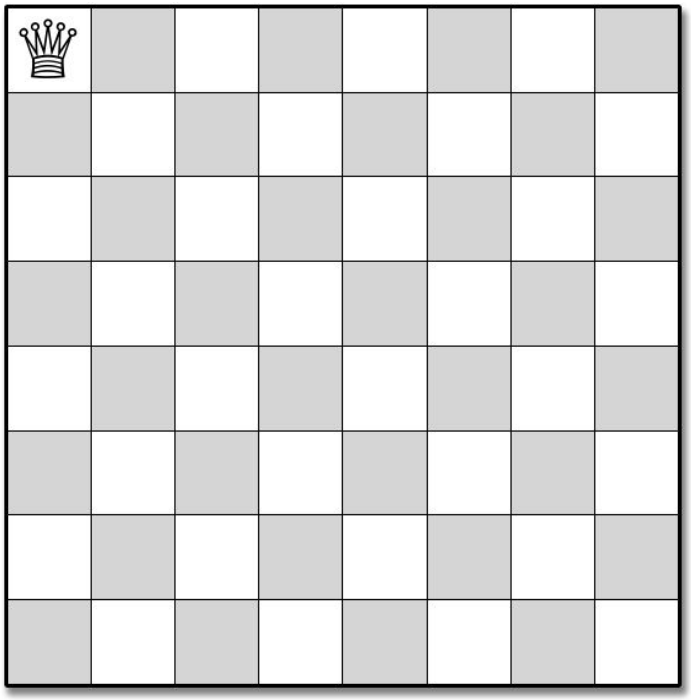
A próxima rainha precisa estar em outra linha.

Não podem existir duas ou mais rainhas na mesma linha.

**Quantas combinações?**

$8^8 = 16.777.216$

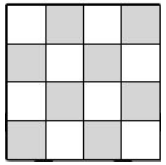
$\frac{\binom{64}{8}}{8^8} \approx 0.38\%$ do problema original.



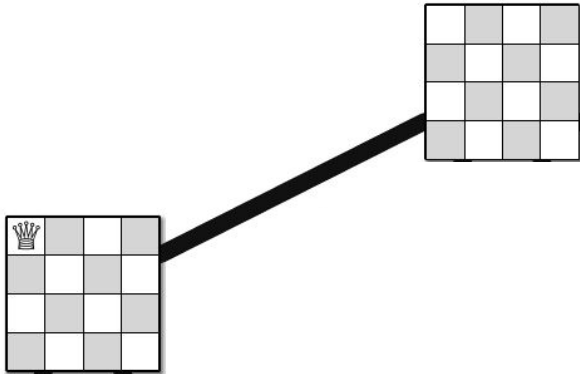
# Backtracking

“Algoritmos de *backtracking* tentam construir a solução de um problema computacional **incrementalmente**. Toda vez que o algoritmo precisa decidir entre múltiplas alternativas para um próximo componente da solução, ele avalia de maneira **recursiva** todas as alternativas, selecionando a melhor delas” (Erickson, 2019).

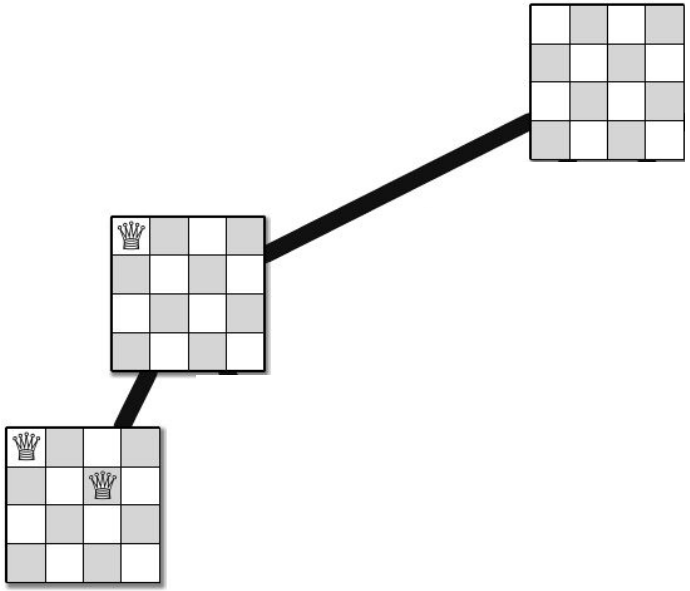
# Ideia



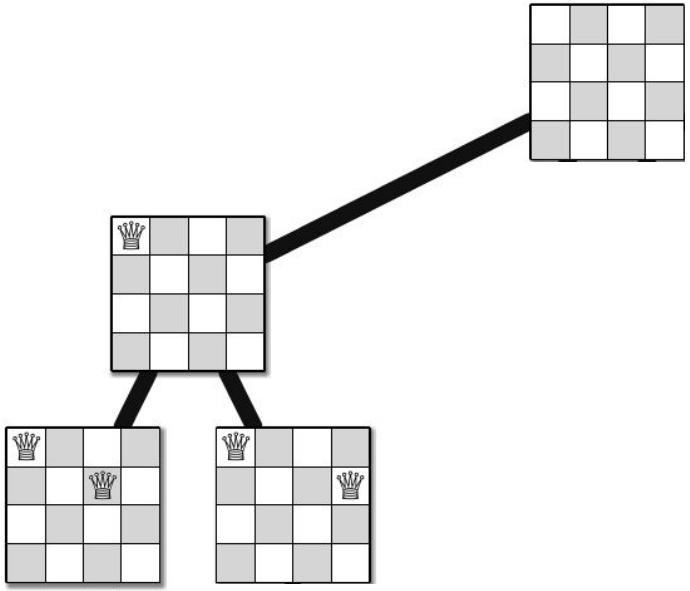
# Ideia



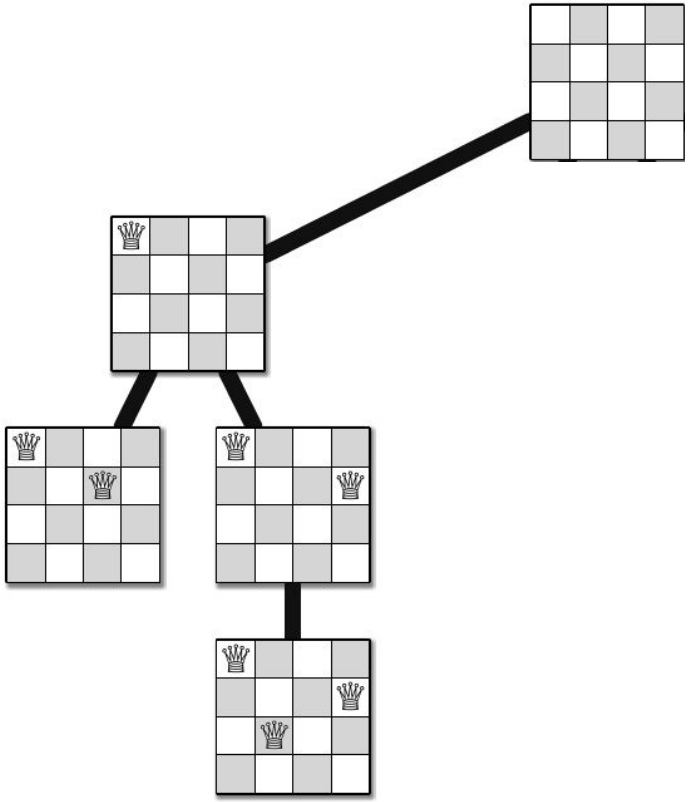
# Ideia



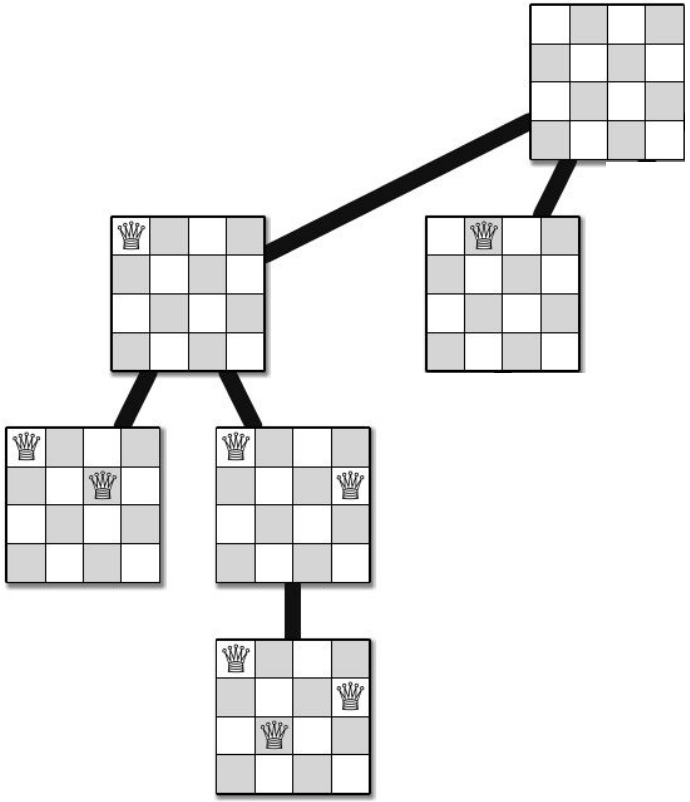
# Ideia



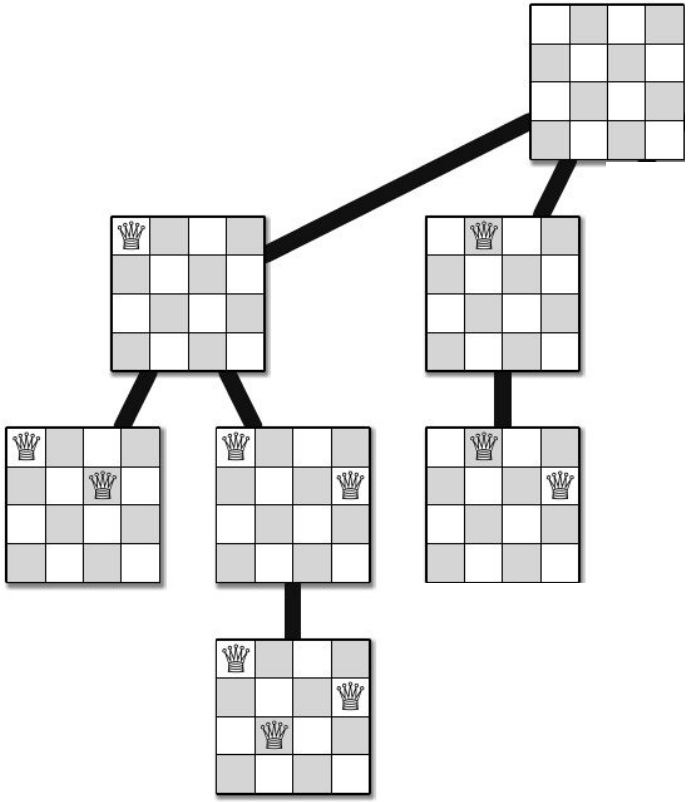
# Ideia



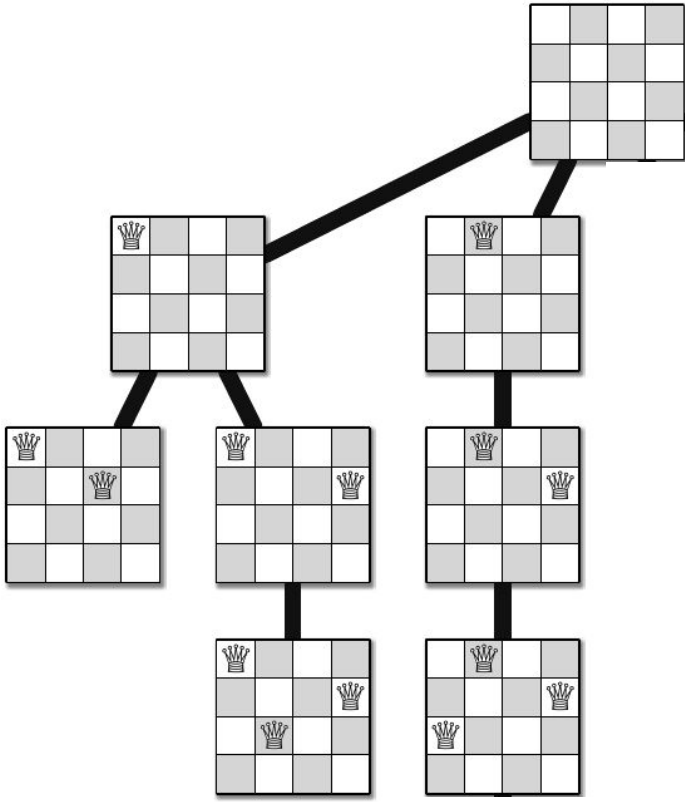
# Ideia



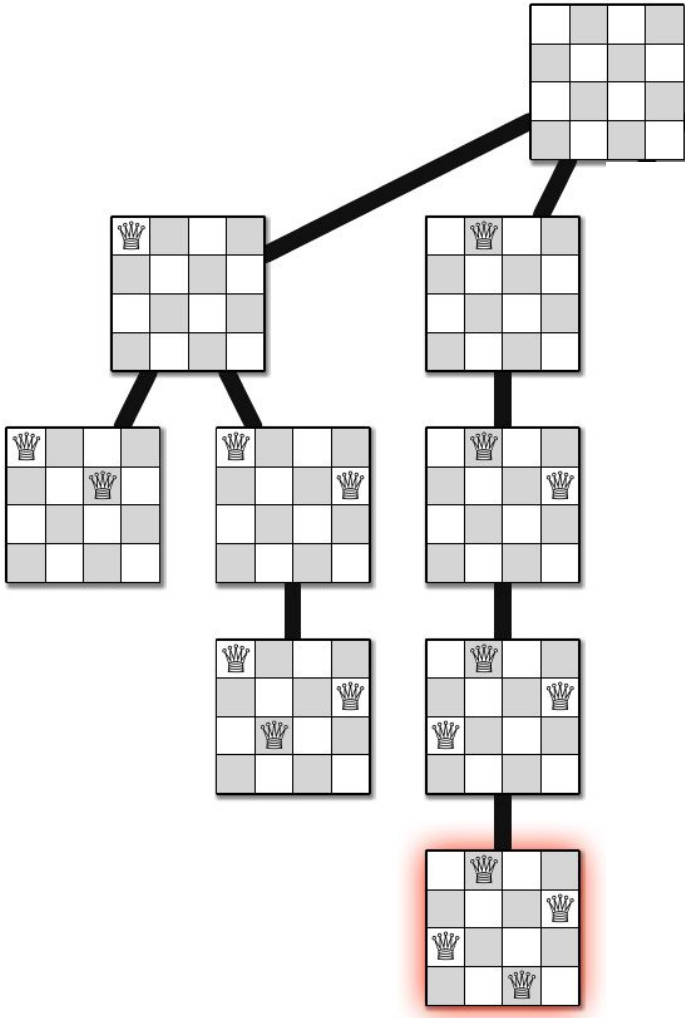
# Ideia



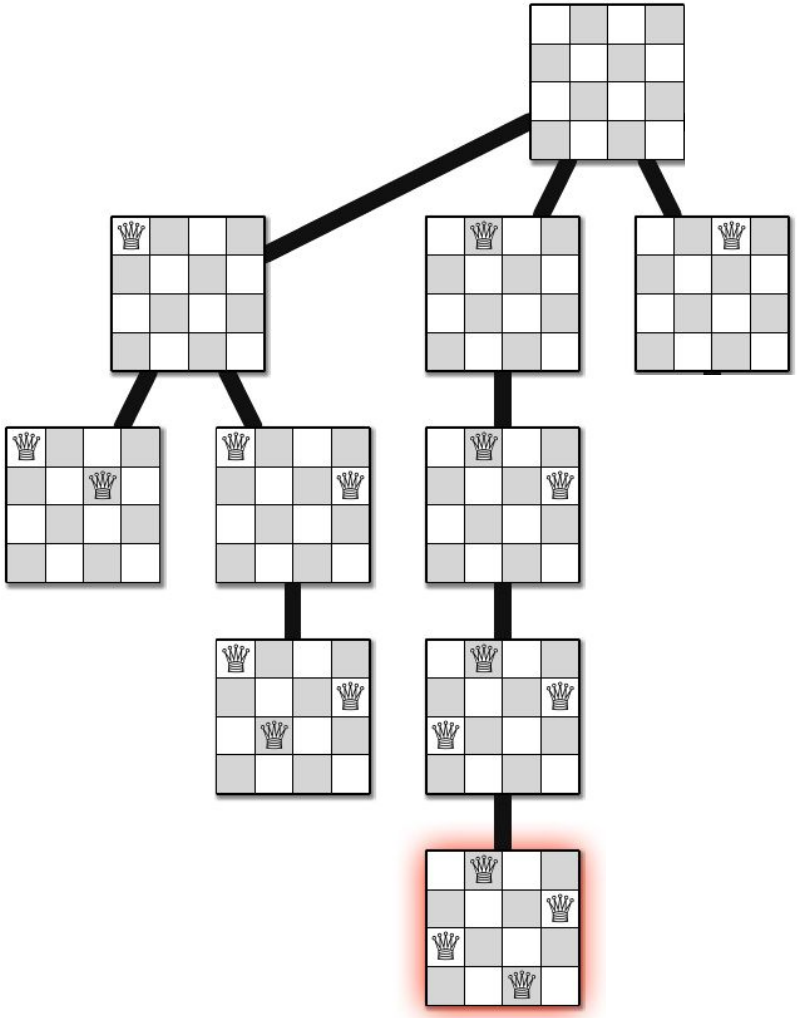
# Ideia



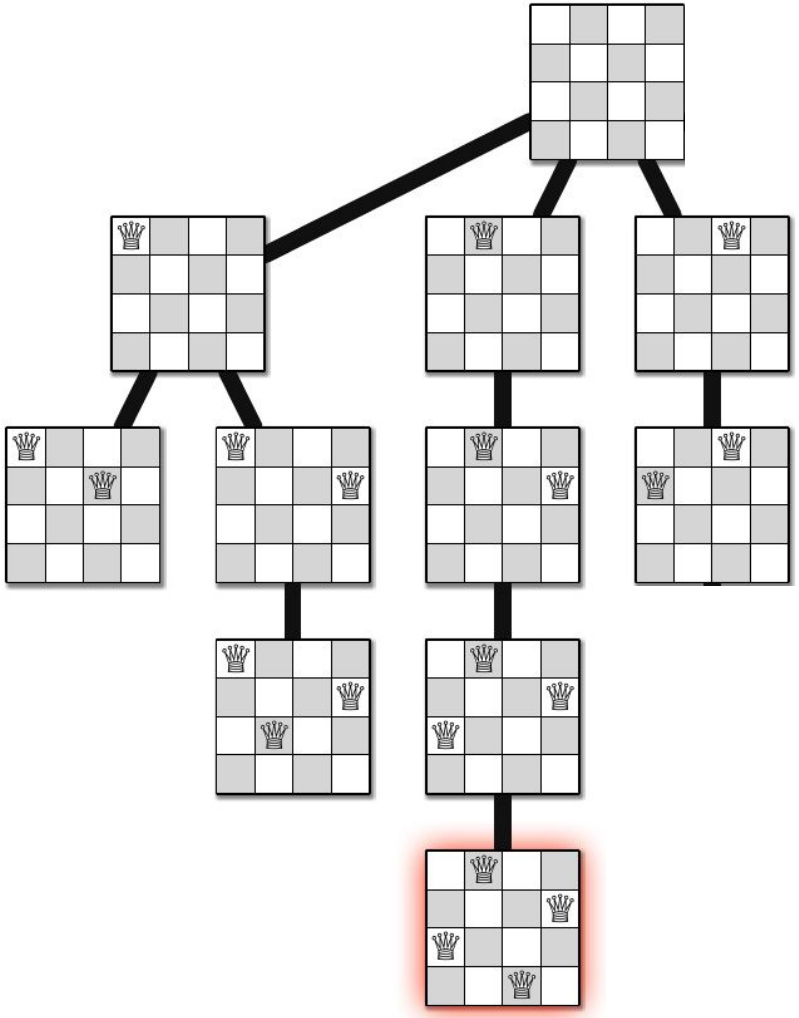
# Ideia



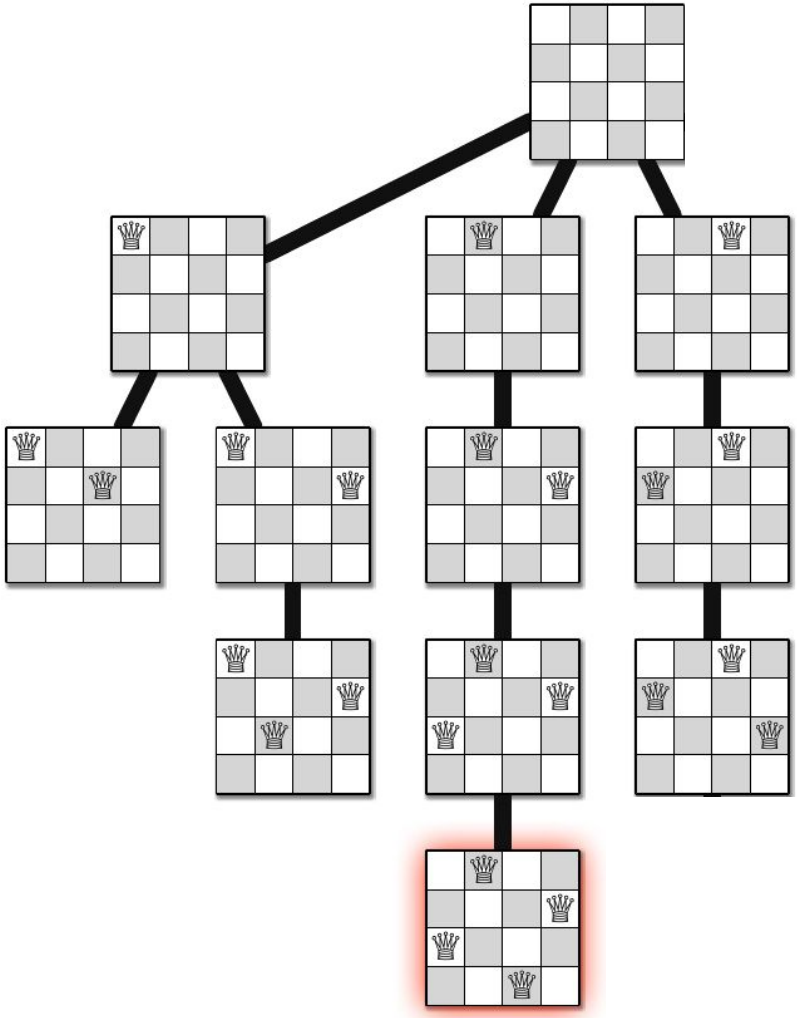
# Ideia



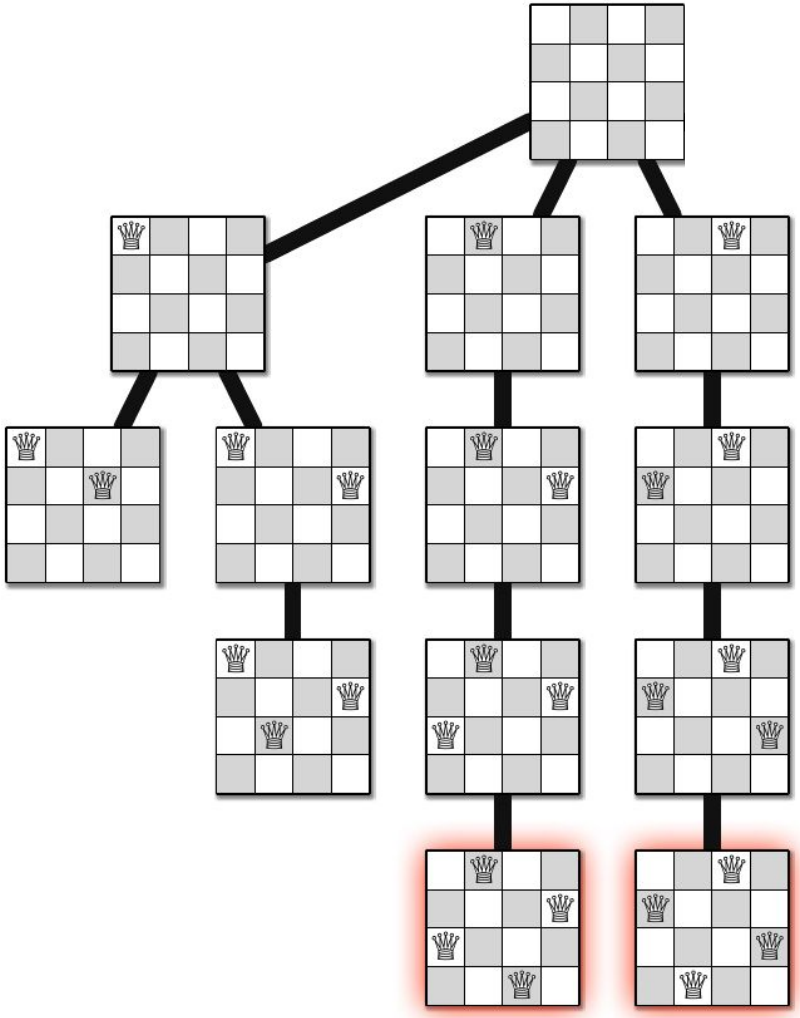
# Ideia



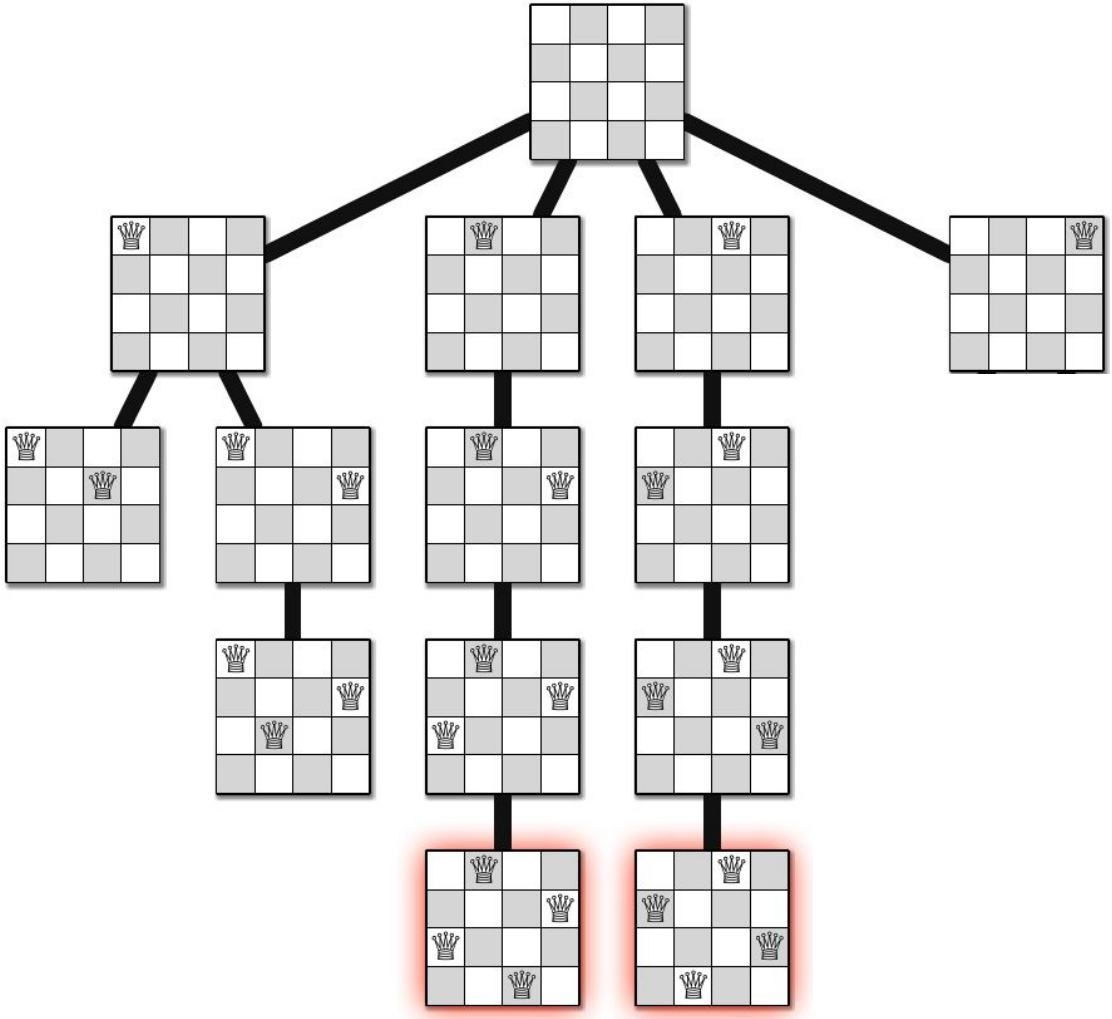
# Ideia



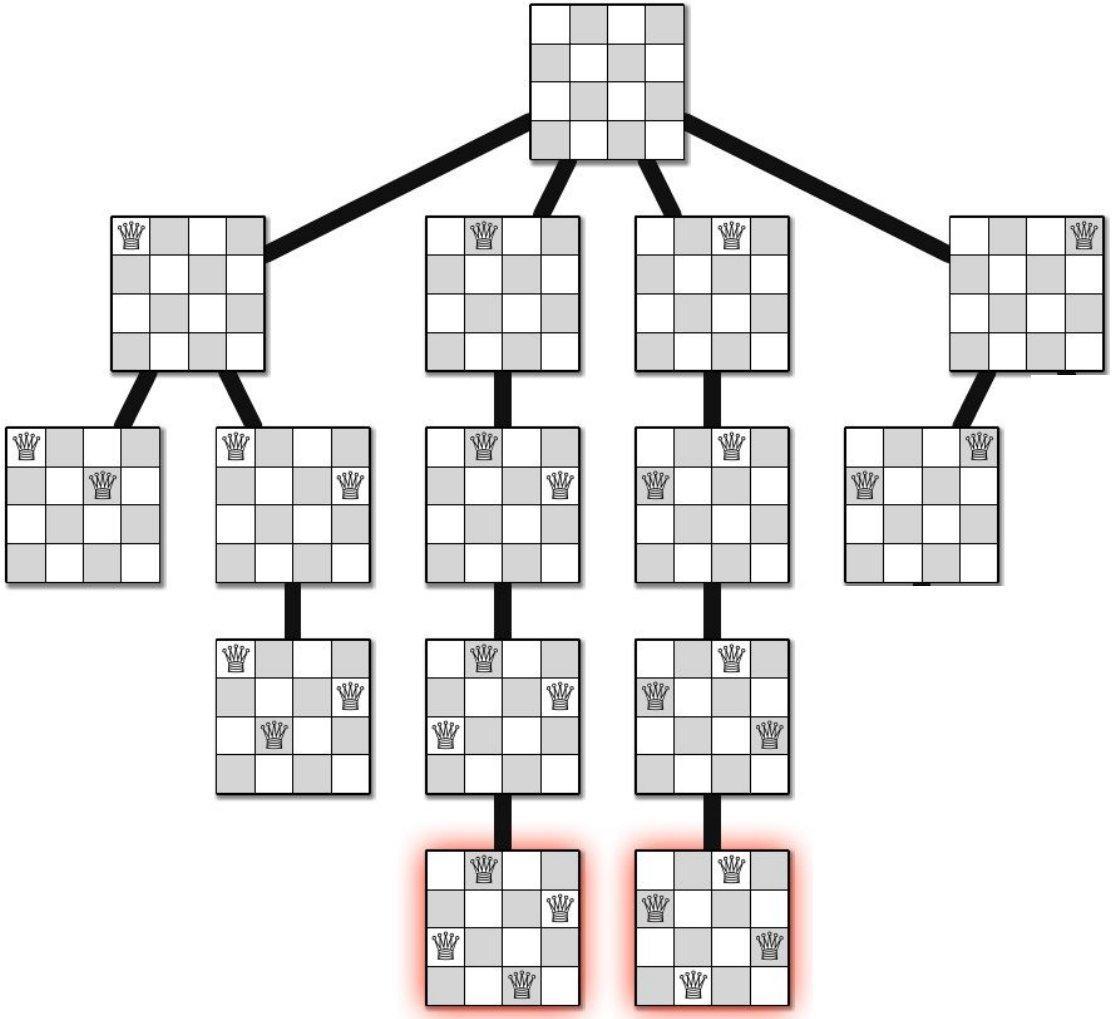
# Ideia



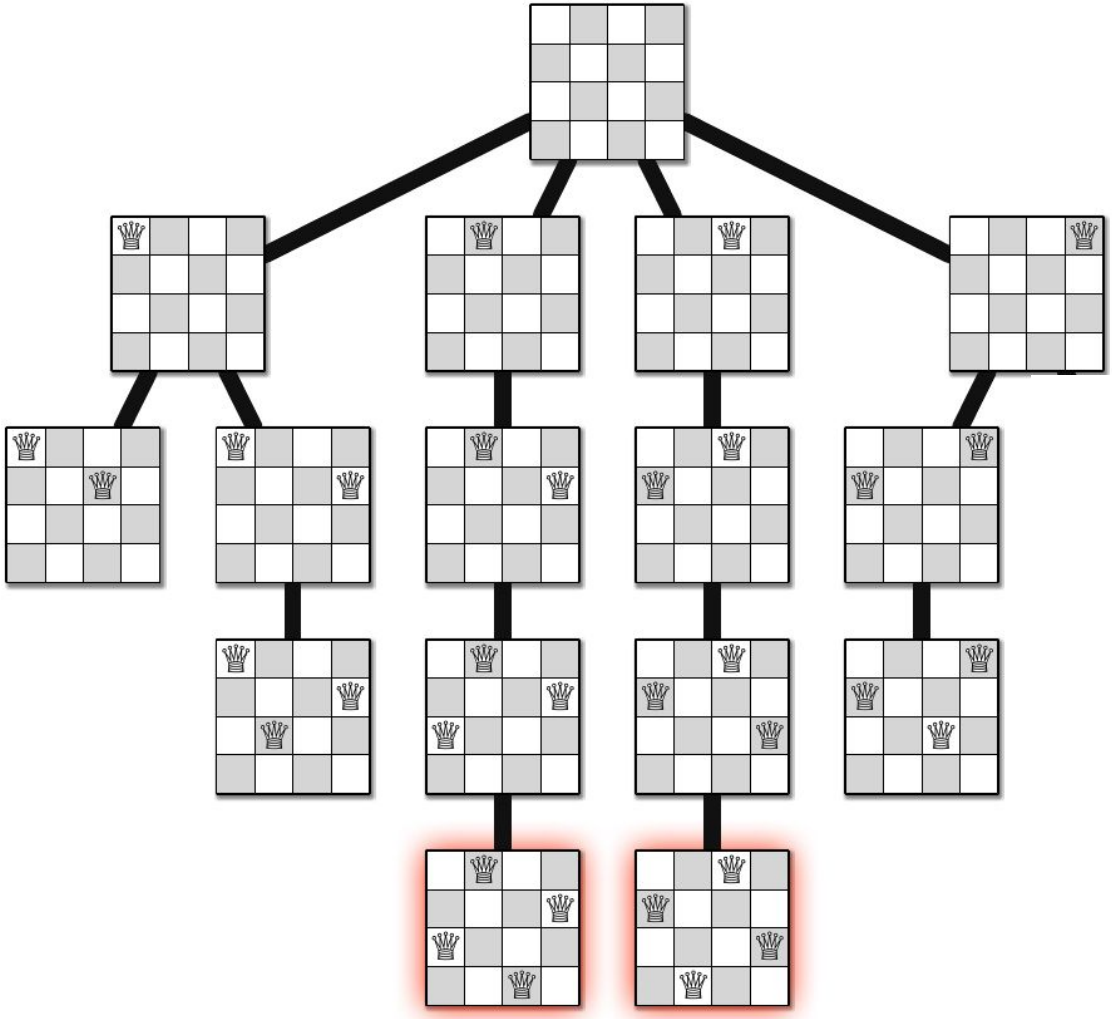
# Ideia



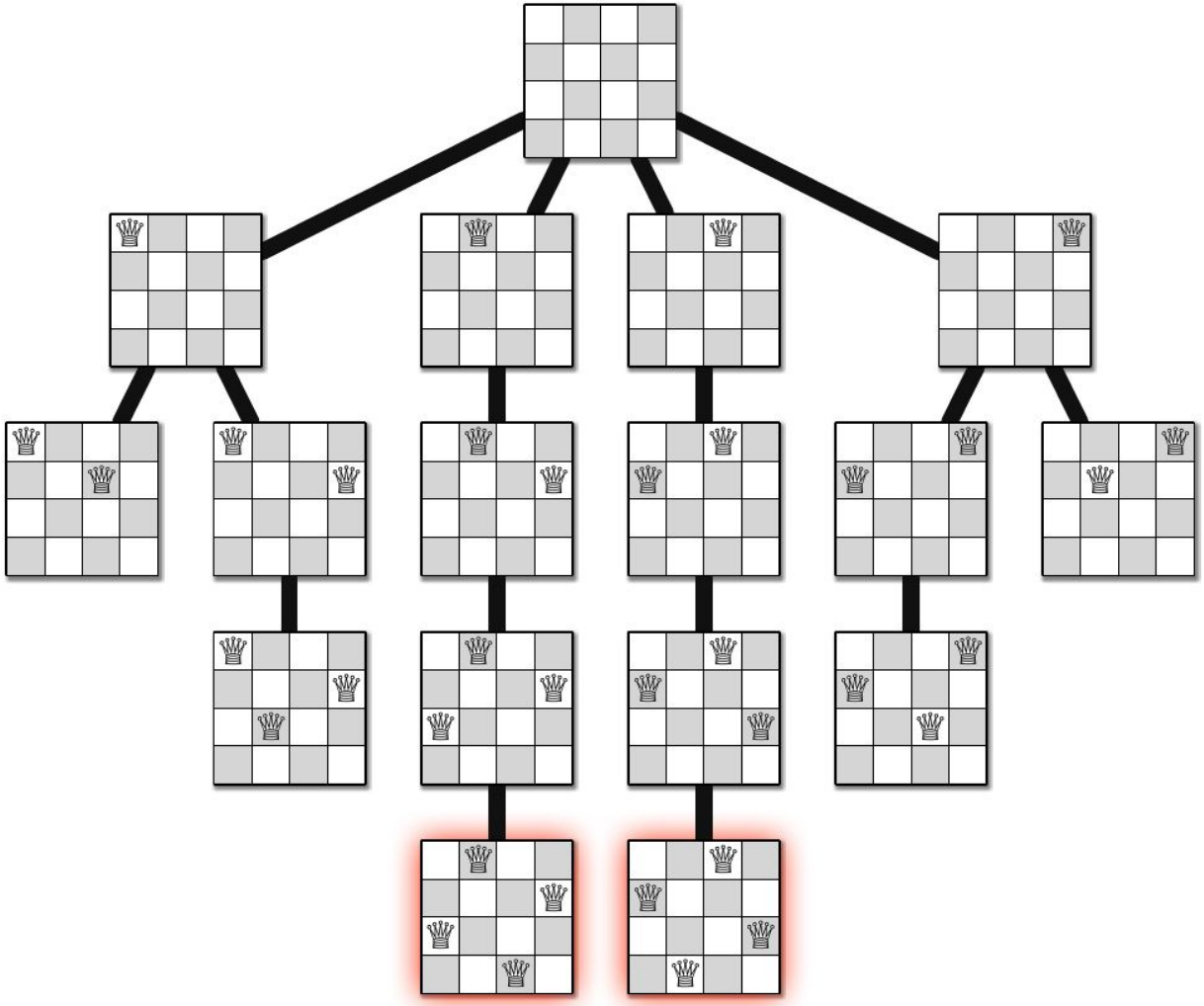
# Ideia



# Ideia



# Ideia



# Solução de Gauss

Solução proposta por Carl Friedrich Gauss em 1850

Johann Carl Friedrich Gauss (30/04/1777 - 23/02/1855).
Matemático, astrônomo e físico alemão.
-Método de Gauss-Seidel
-Lei de Gauss
-Lei de Gauss para o magnetismo
-Distribuição Normal
en.wikipedia.org/wiki/Carl_Friedrich_Gauss

# Solução de Gauss

```
função posicionarRainhas (t,n,r)
entrada: vetor t representando um tabuleiro indexado por [1..n],e a linha atual r
saída: as soluções para o problema das n rainhas são impressas

se r = n+1
     imprimaSolução()
     retorne
para j ← 1 até n
     legal ← verdadeiro
     para i ← 1 até r-1
          se (t[i] = j ou t[i] = j+r-i ou t[i] = j-r+i)
               legal ← falso
     se legal
          t[r] ← j
          posicionarRainhas(t,n,r+1)
retorne
```

# Solução de Gauss

Faça um teste de mesa **com calma** para um tabuleiro de 4x4.

```
função posicionarRainhas (t,n,r)
entrada: vetor t representando um tabuleiro indexado por [1..n],e a linha atual r
saída: as soluções para o problema das n rainhas são impressas

se r = n+1
     imprimaSolução()
     retorne
para j ← 1 até n
     legal ← verdadeiro
     para i ← 1 até r-1
          se (t[i] = j ou t[i] = j+r-i ou t[i] = j-r+i)
               legal ← falso
     se legal
          t[r] ← j
          posicionarRainhas(t,n,r+1)
retorne
```

# Solução de Gauss

```
função posicionarRainhas (t,n,r)
entrada: vetor t representando um tabuleiro indexado por [1..n],e a linha atual r
saída: as soluções para o problema das n rainhas são impressas

se r = n+1
      imprimaSolução()
      retorne
para j ← 1 até n
      legal ← verdadeiro
      para i ← 1 até r-1
            se (t[i] = j ou t[i] = j+r-i ou t[i] = j-r+i)
                  legal ← falso
      se legal
            t[r] ← j
            posicionarRainhas(t,n,r+1)
retorne
```

Tenta inserir uma rainha na posição j na linha atual.

# Solução de Gauss

```
função posicionarRainhas (t,n,r)
entrada: vetor t representando um tabuleiro indexado por [1..n],e a linha atual r
saída: as soluções para o problema das n rainhas são impressas

se r = n+1
     imprimaSolução()
     retorne
para j ← 1 até n
     legal ← verdadeiro
     para i ← 1 até r-1
          se (t[i] = j ou t[i] = j+r-i ou t[i] = j-r+i)
               legal ← falso
     se legal
          t[r] ← j
          posicionarRainhas(t,n,r+1)
retorne
```

Verifica se a rainha não está em conflito com as rainhas nas linhas anteriores.

# Solução de Gauss

```
função posicionarRainhas (t,n,r)
entrada: vetor t representando um tabuleiro indexado por [1..n],e a linha atual r
saída: as soluções para o problema das n rainhas são impressas

se r = n+1
      imprimaSolução()
      retorne
para j ← 1 até n
      legal ← verdadeiro
      para i ← 1 até r-1
            se (t[i] = j ou t[i] = j+r-i ou t[i] = j-r+i)
                  legal ← falso
      se legal
            t[r] ← j
            posicionarRainhas(t,n,r+1)
retorne
```

Verifica se existe uma rainha na mesma coluna

Verifica se existe uma rainha na diagonal à direita

Verifica se existe uma rainha na diagonal à esquerda

# Solução de Gauss

```
função posicionarRainhas (t,n,r)
entrada: vetor t representando um tabuleiro indexado por [1..n],e a linha atual r
saída: as soluções para o problema das n rainhas são impressas

se r = n+1
      imprimaSolução()
      retorne
para j ← 1 até n
      legal ← verdadeiro
      para i ← 1 até r-1
            se (t[i] = j ou t[i] = j+r-i ou t[i] = j-r+i)
                  legal ← falso
      se legal
            t[r] ← j
            posicionarRainhas(t,n,r+1)
retorne
```

Se a rainha não está em conflito, insira na posição.
Considere a solução atual como uma **solução parcial**.
Chame recursivamente para inserir a próxima rainha.

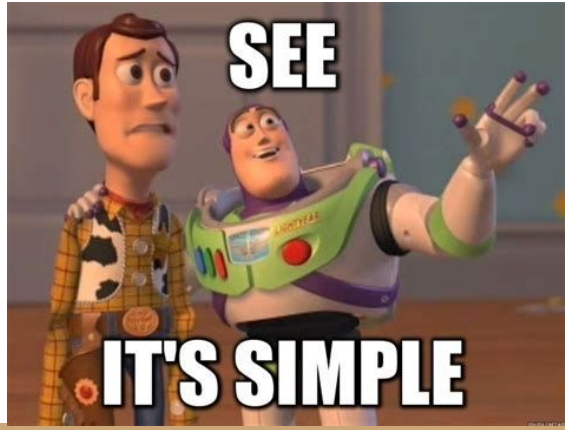
# Solução de Gauss

```
função posicionarRainhas (t,n,r)
entrada: vetor t representando um tabuleiro indexado por [1..n],e a linha atual r
saída: as soluções para o problema das n rainhas são impressas

se r = n+1
      imprimaSolução()
      retorne
para j ← 1 até n
      legal ← verdadeiro
      para i ← 1 até r-1
            se (t[i] = j ou t[i] = j+r-i ou t[i] = j-r+i)
                  legal ← falso
      se legal
            t[r] ← j
            posicionarRainhas(t,n,r+1)
retorne
```

**Backtracking**. Se a chamada recursiva não encontrar uma solução (ou nesse problema, mesmo que encontre), voltamos para soluções passadas e as modificamos em busca de novas soluções.

# Solução de Gauss

```
função posicionarRainhas (t,n,r)
entrada: vetor t representando um tabuleiro indexado por [1..n],e a linha atual r
saída: as soluções para o problema das n rainhas são impressas

se r = n+1
      imprimaSolução()
      retorne
para j ← 1 até n
      legal ← verdadeiro
      para i ← 1 até r-1
            se (t[i] = j ou t[i] = j+r-i ou t[i] = j-r+i)
                  legal ← falso
      se legal
            t[r] ← j
            posicionarRainhas(t,n,r+1)
retorne
```

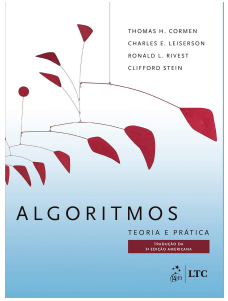


# Exercícios

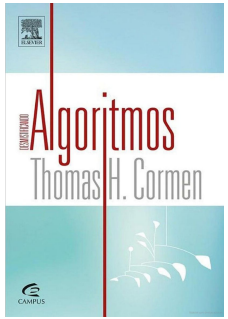
1. Implemente o algoritmo em C
2. Modifique a função *posicionarRainhas(t,n,r)* para a função retorne ao encontrar a primeira solução. A solução deve ser retornada.
3. Modifique a função para que sejam **retornadas** todas as soluções possíveis
    a. Faça de forma genérica, para que a função retorne todas as soluções de um problema de nxn (não faça uma matriz de tamanho fixo para conter os resultados)

# Referências


T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 3a ed. 2012



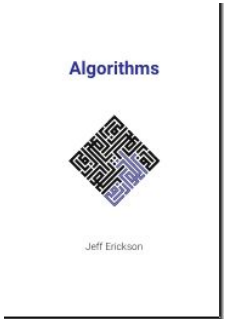

T. Cormen. Desmistificando algoritmos. 2017.




Renato Carmo. Algoritmos e Estruturas de Dados. www.inf.ufpr.br/renato

R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2014


J. Erickson. Algorithms. 2019.



# Licença