Programação Funcional

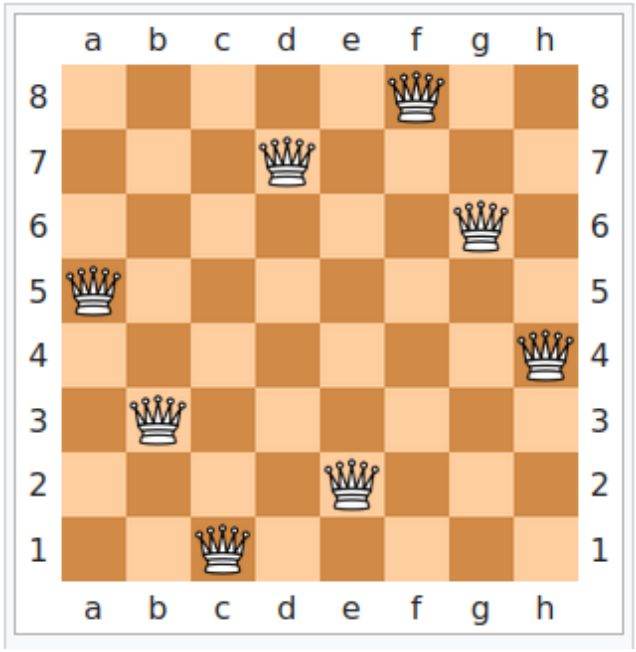
Aula 12 — O *puzzle* das 8 rainhas

Pedro Vasconcelos, DCC/FCUP

abril 2021

O *puzzle* das 8 rainhas

Colocar 8 rainhas num tabuleiro de Xadrez (8×8) de forma a que estas não se ataquem entre si.



https://en.wikipedia.org/wiki/Eight_queens_puzzle

Objetivo

- Escrever um programa para encontrar todas as soluções deste *puzzle*
- Um problema de *enumeração e pesquisa*
  - espaço finito de alternativas (posições das rainhas)
  - procuramos soluções que satisfazem uma condição (não há ataques)

Algoritmo

Algoritmo clássico: colocar *n* rainhas por *backtracking*

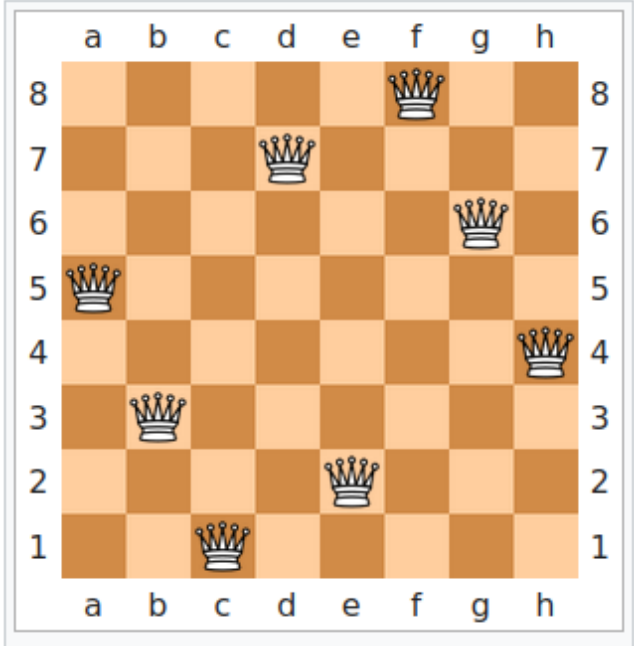
- gerar uma posição para a próxima rainha
- se há ataques: retrocer e tentar outra posição
- repetir até gerar posições válidas para todas as rainhas

Em Haskell: em vez de pensar imperativamente (*avançar/retrocer*) vamos *gerar a lista de todas as soluções*.

Representação do tabuleiro

- Lista de posições das rainhas
- Nunca há duas rainhas na mesma coluna
- Vamos representar pela *lista das linhas* de cada coluna

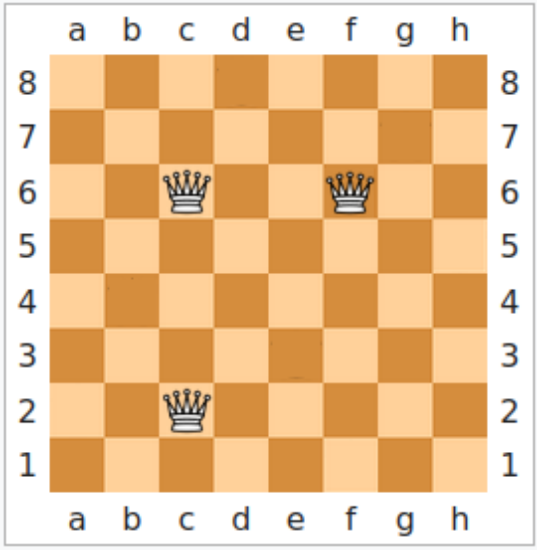
```
type Rainhas = [Int]
```

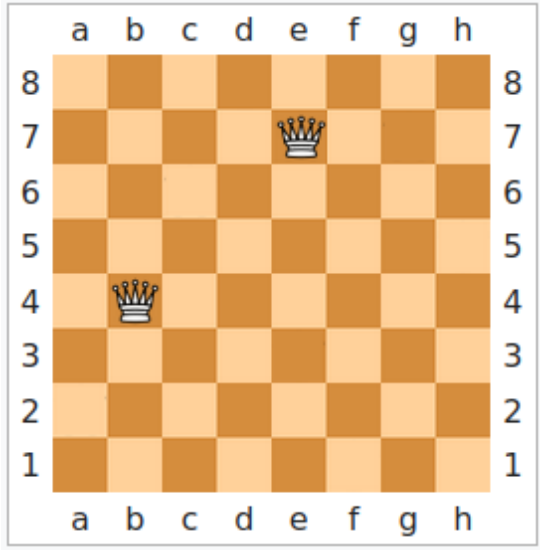


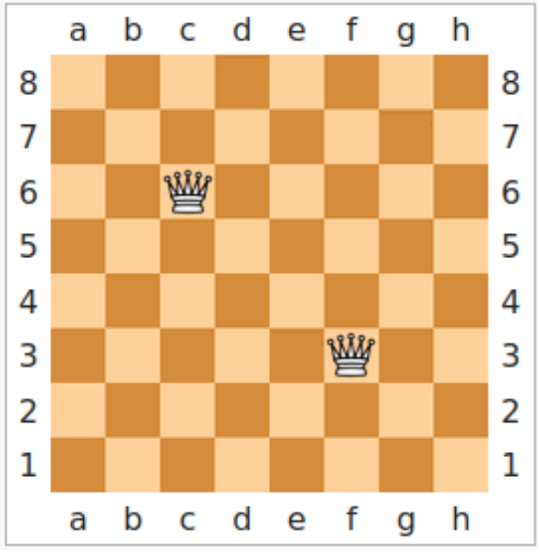

```
[5, 3, 1, 7, 2, 8, 6, 4] :: Rainhas
```

Ataques

- Duas rainhas atacam-se se estão na mesma linha, coluna, diagonal principal ou secundária
- Testar ataques é mais fácil entre pares de coordenadas (x, y) e (x', y')







$$x = x' \vee y = y' \qquad x - y = x' - y' \qquad x + y = x' + y'$$

Em Haskell

```haskell
ataque :: (Int,Int) -> (Int,Int) -> Bool
ataque (x,y) (x',y')
     = x==x' ||          -- coluna igual
       y==y' ||          -- linha igual
       x-y == x'-y' ||   -- diagonal 1 igual
       x+y == x'+y'      -- diagonal 2 igual
```

Simplificação

Podemos omitir o teste de igualdade de colunas: o nosso programa vai sempre gerar uma só rainha por coluna

```
ataque :: (Int,Int) -> (Int,Int) -> Bool
ataque (x,y) (x',y')
     = -- x==x'          -- sempre False
       y==y' ||          -- linha igual
       x-y == x'-y' ||   -- diagonal 1 igual
       x+y == x'+y'      -- diagonal 2 igual
```

Listar todas as soluções

```
rainhas :: Int -> [Rainhas]
```

- Colocar n rainhas nas n primeiras colunas
- Definição recursiva:
  - se $n = 0$: solução trivial (vazia)
  - se $n > 0$: resolvemos para $n - 1$ e consideramos todas as 8 posições para a nova rainha
  - a posição é válida se as rainhas anteriores não atacam a rainha nova
- `rainhas 8` calcula todas as soluções do *puzzle*

Em Haskell

```haskell
rainhas :: Int -> [Rainhas]
rainhas 0 = [[]]
rainhas n | n>0
  = [ys++[y] | ys<-rainhas (n-1), y<-[1..8], y`valida`ys]

valida :: Int -> Rainhas -> Bool
valida y ys = all (not . ataque (x,y)) (zip [1..] ys)
  where x = 1+length ys -- coluna da nova rainha
```

Todas as soluções

```
> rainhas 8
[[1,5,8,6,3,7,2,4],[1,6,8,3,7,4,2,5],[1,7,4,6,8,2,5,3],
[1,7,5,8,2,4,6,3],[2,4,6,8,3,1,7,5],...

> length (rainhas 8)
92
```

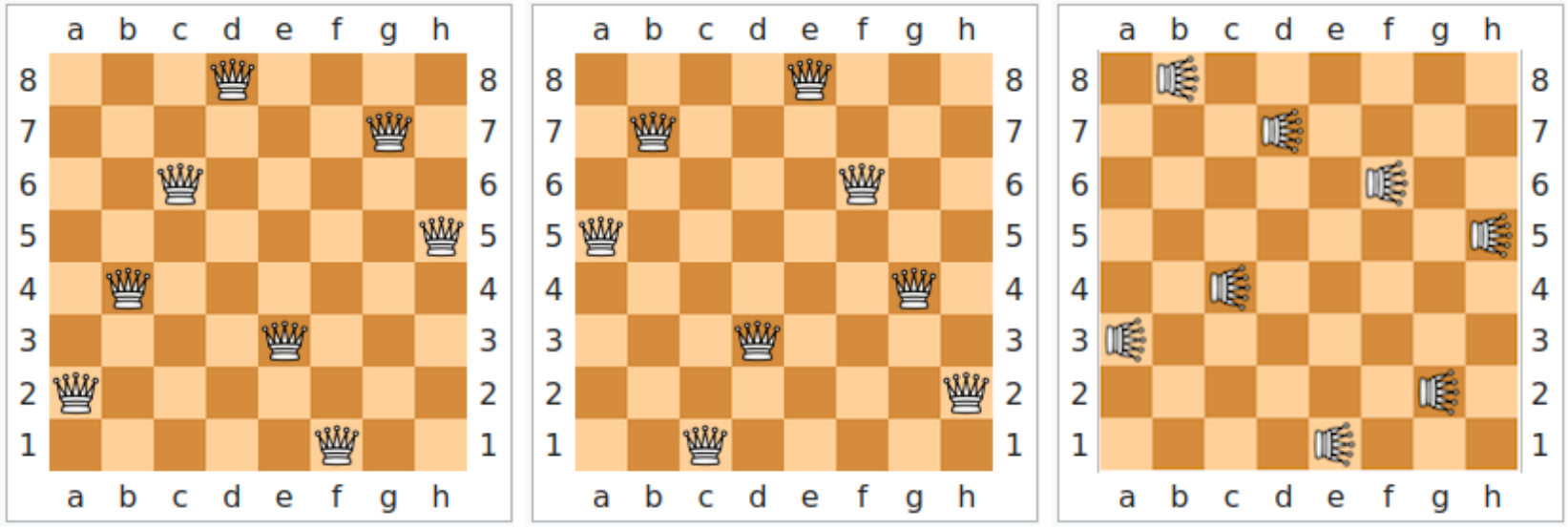
Apenas a primeira solução

```
> head (rainhas 8)
[1,5,8,6,3,7,2,4]
```

- Por causa da *lazy evaluation* esta pesquisa não necessita de construir toda a lista
- Obtemos a primeira solução mais rapidamente

Extras: Simetrias

- Há muitas soluções que são similares
- Podem ser obtidas de outras por rotações ou simetrias horizontais/verticais



- Exercício: modificar o programa para listar apenas soluções fundamentais (são 12)