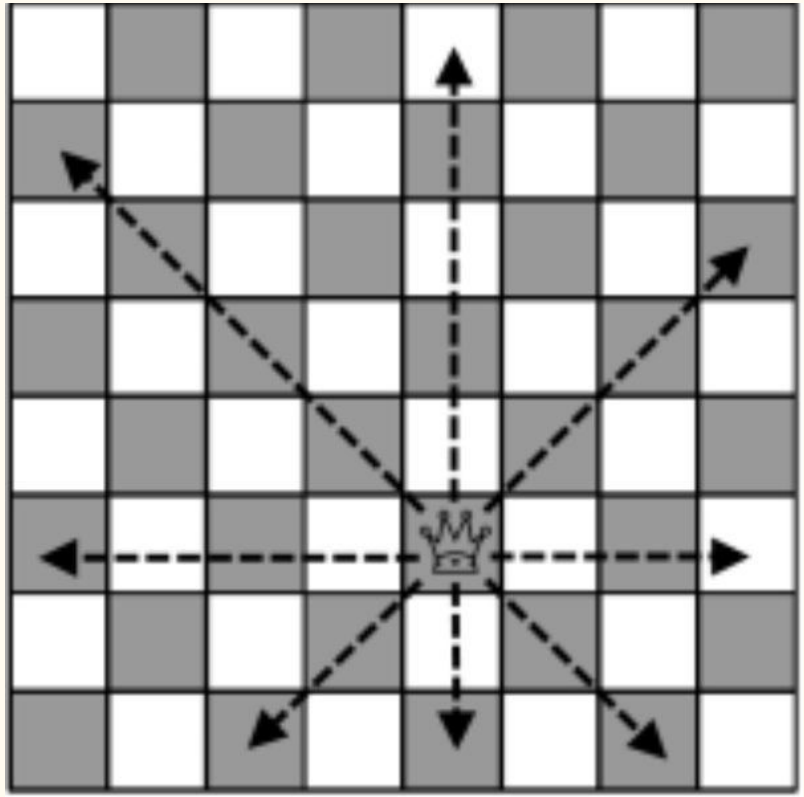
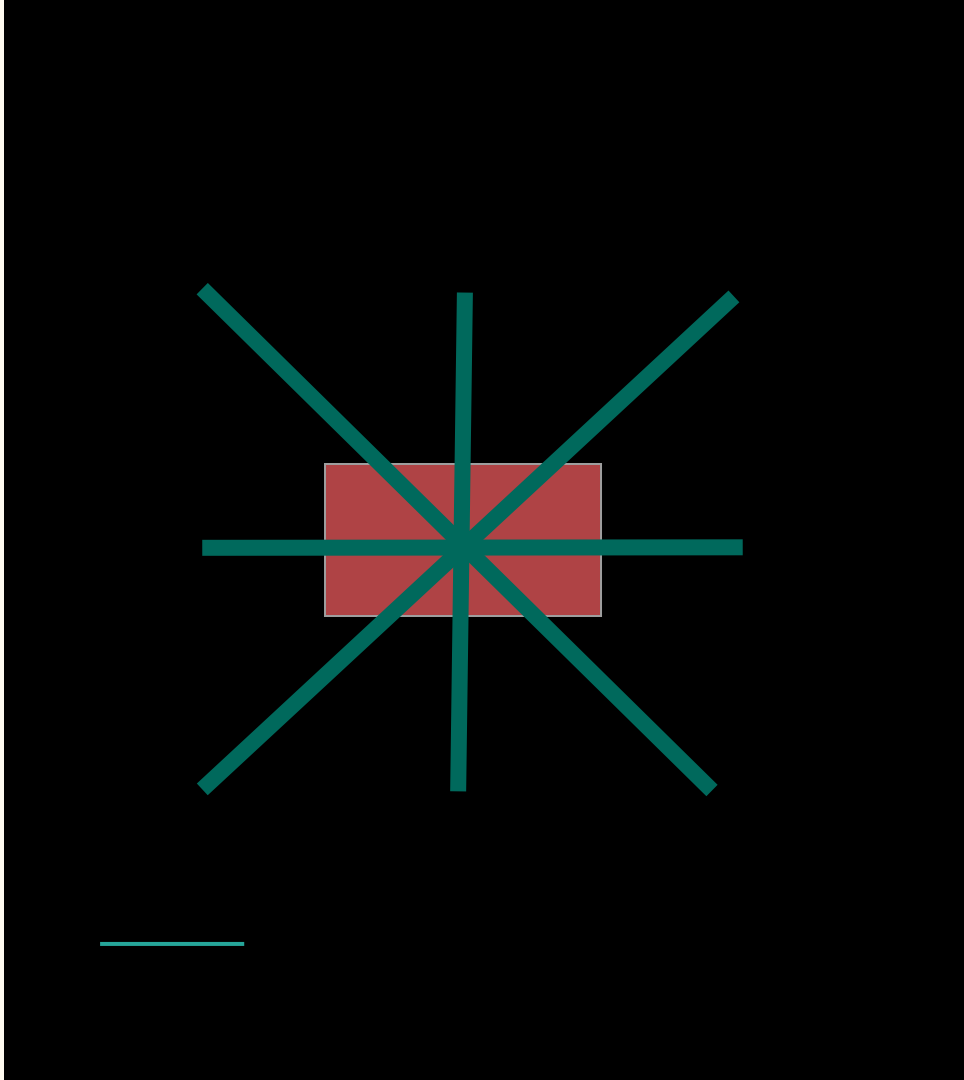
‘8’ Queens problem

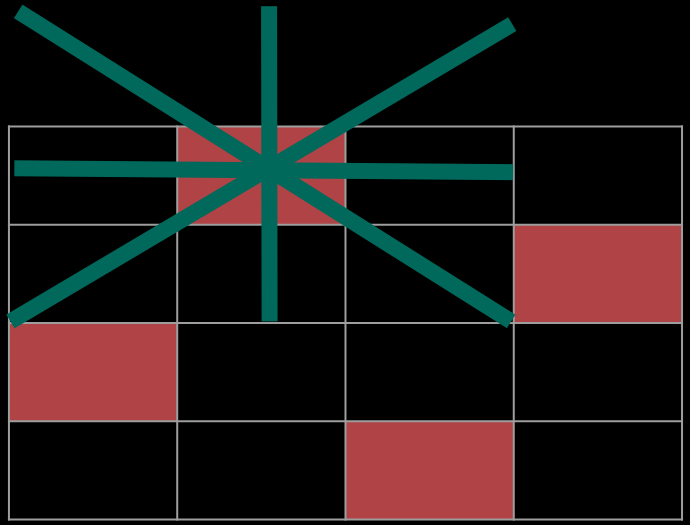
Queen move



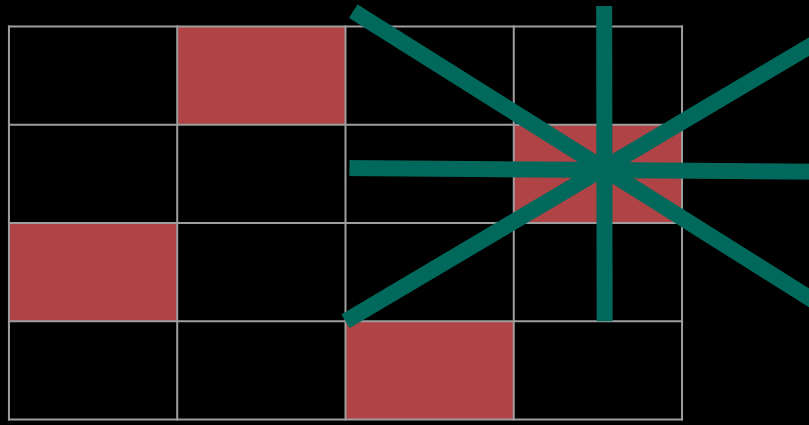
```
queens 1
[[1]]
```



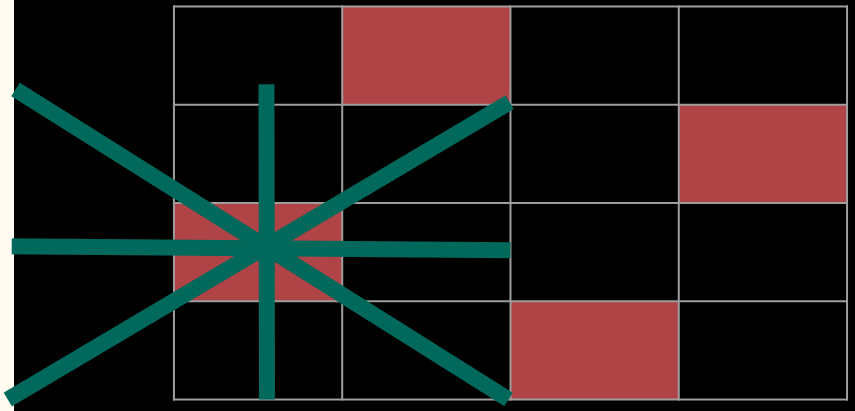
```
queens 4
[[2, 4, 1, 3], [3, 1, 4, 2]]
```



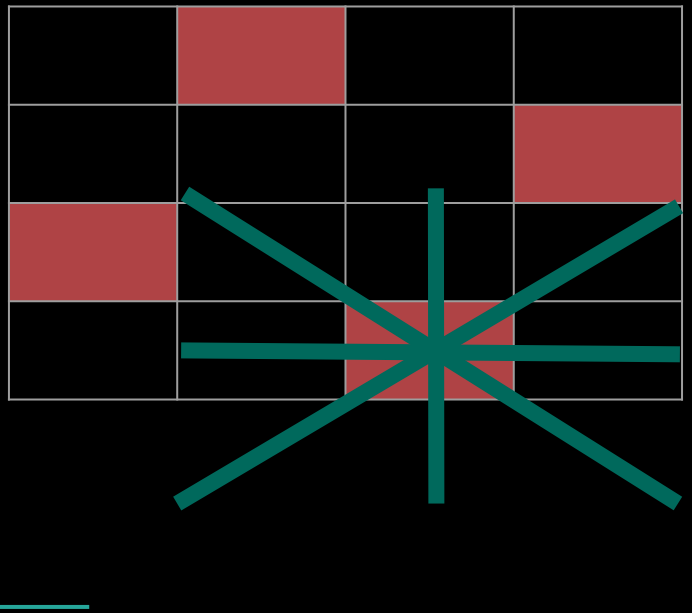
```
queens 4
[[2, 4, 1, 3], [3, 1, 4, 2]]
```



```
queens 4
[[2, 4, 1, 3], [3, 1, 4, 2]]
```



```
queens 4
[[2, 4, 1, 3], [3, 1, 4, 2]]
```



Sabemos que...

- El output son listas dentro de otra lista.

- Sabemos que las listas que están adentro, no pueden tener mayor longitud de la que colocamos.
  - Porque solo hay n filas.

```
type Solution = [Int]

queens::Int -> [Solution]
queens = error "Implement It"
```

¿Qué necesitamos?

- Recursión entre las distintas soluciones que puede haber
- Recursión para recorrer el tablero
- i > size
  - **i** = fila en la que estamos parados
  - **size** = valor que recibimos como parámetro
- Tenemos que empezar desde la primera fila

```haskell
type Solution = [Int]

queens::Int -> [Solution]
queens size = solve 1 [[]]
```

```haskell
type Solution = [Int]

queens::Int -> [Solution]
queens size = solve 1 [[]]
  where solve i solutions
          | i > size  = solutions
          | otherwise = solve (i+1) [j:solution | j <- [1..size], solution <- solutions, ok j 1 solution]
```


Una lista con todos los valores de la columna

```haskell
type Solution = [Int]

queens::Int -> [Solution]
queens size = solve 1 [[]]
  where solve i solutions
          | i > size  = solutions
          | otherwise = solve (i+1) [j:solution | j <- [1..size], solution <- solutions, ok j 1 solution]
```

Para empezar en el primer valor

```haskell
queens::Int -> [Solution]
queens size = solve 1 [[]]
  where solve i solutions
          | i > size  = solutions
          | otherwise = solve (i+1) [j:solution | j <- [1..size], solution <- solutions, ok j 1 solution]
        ok _ _ []           = True
        ok j offset (c:cs) = j /= c && j /= c-offset && j /= c + offset && ok j (offset+1) cs
```