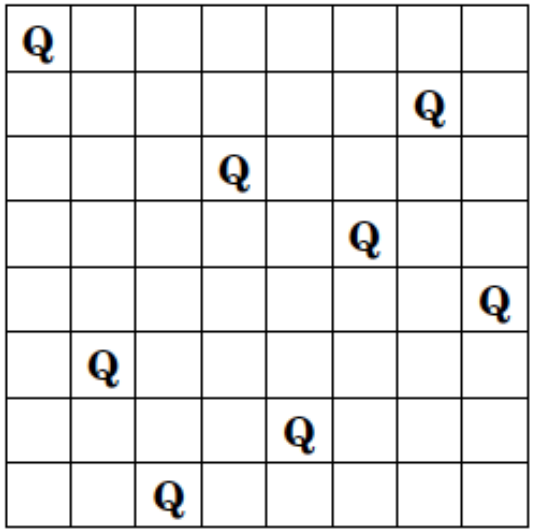
Modelling and Solving Exercises in MiniZinc -1

Before Starting

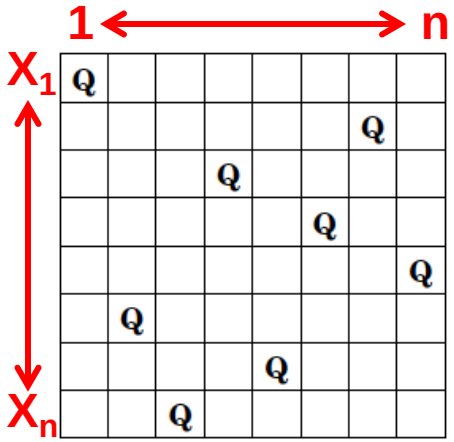
- Download MiniZinc.
- Download the exercise descriptions and the necessary files from the Virtuale page.
  - Module 1: Material for Exercise Sessions.
- Use a separate folder for each problem.
- Configure the solver to obtain the solution statistics, to search for one or all solutions, and to set a time limit when needed.

N-Queens

- Place n queens in an nxn board so that no two queens can attack each other.



Row Model



- Variables and Domains
  - A variable for each row $[X_1, X_2, \ldots, X_n]$ → no row attack
  - Domain values [1..n] represent the columns:
    - $X_i = j$ means that the queen in row i is in column j
- Constraints
  - alldifferent($[X_1, X_2, \ldots, X_n]$) → no column attack
  - for all $i<j$ $|X_i - X_j| \neq |i - j|$ → no diagonal attack

Alldifferent Model

- Variables
  - $[X_1, X_2, \ldots, X_n] \in [1..n]$
- Constraints
  - alldifferent($[X_1, X_2, \ldots, X_n]$)
  - alldifferent($[X_1 + 1, X_2 + 2, \ldots, X_n + n]$)
  - alldifferent($[X_1 - 1, X_2 - 2, \ldots, X_n - n]$)

Combined Alldifferent and Symmetry Breaking Model

- Variables
  - for all i, $X_i \in [1..n]$, for all i, j $B_{ij} \in [0..1]$
- Constraints
  - alldifferent($[X_1, X_2, \ldots, X_n]$)
  - alldifferent($[X_1 + 1, X_2 + 2, \ldots, X_n + n]$)
  - alldifferent($[X_1 - 1, X_2 - 2, \ldots, X_n - n]$)
  - lex$\leq$(B , π(B)) for all π
    - Study Section 2.7.6 of the MiniZinc Tutorial.
- Channeling Constraints
  - for all i,j $X_i = j \leftrightarrow B_{ij} = 1$

N-Queens 1

- Implement the 3 models with GAC on alldifferent.
- Search for **all solutions** for n = 8, 9, 10, 12 using the default search of Gecode.
- Compare the number of solutions and the failures.

N-Queens 2

- Focus on the alldifferent model.
- Post the alldifferent constraints either by using global constraints or by decomposing them.
- Search for **one solution** for N = 28, 29 and 30 using the **input order of the variables and the values** with Gecode.

```
solve :: int_search(q, input_order, indomain_min)  satisfy;
```

- Compare the number of failures and the total time.