TP1 : Constraint Programming

Exercise 1

Consider the following addition problem :

```
  SEND
+ MORE
-------
= MONEY
```

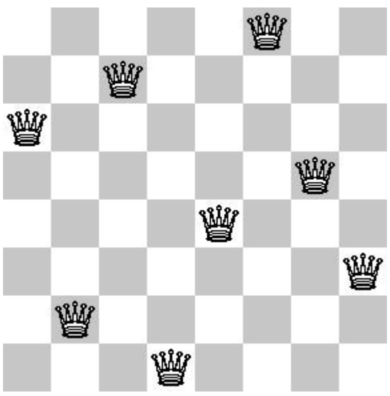
Each letter represents a distinct digit between 0 and 9. We need to determine the value of each letter, keeping in mind that the first letter of each word is different from zero.

Question 1 • Model the problem as a constraint network $N = \langle X, D, C \rangle$.

Question 2 • How large is the search space for this problem ?

Exercise 2

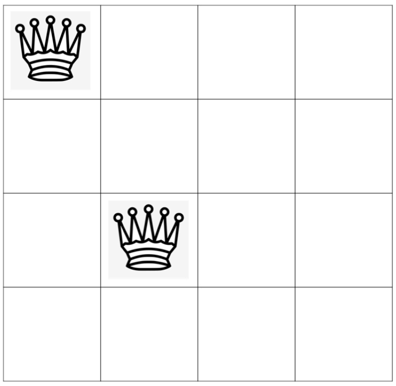
Consider a chessboard of size $(N \times N)$. The N-Queens problem consists of placing N queens such that none of them can attack each other.



Question 1 • Model the problem as a constraint satisfaction problem (CSP) $N = \langle X, D, C \rangle$.

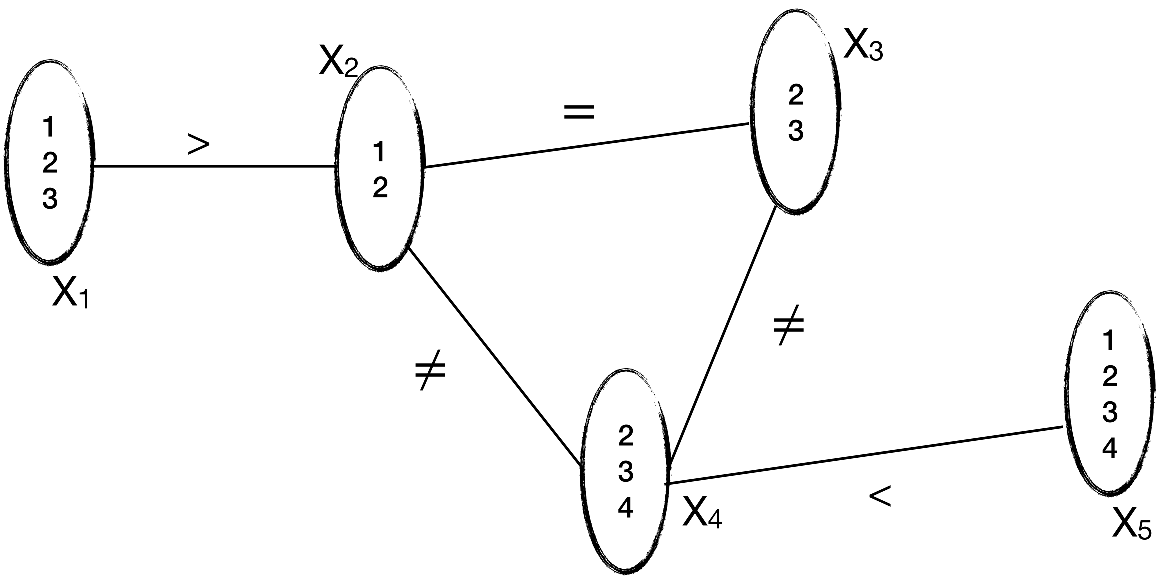
Question 2 • What is the size of the search space for this problem on a $N \times N$ chessboard ?

Question 3 • Run the Backtracking (BT), Forward Checking (FC), and Maintaining Arc Consistency (MAC) algorithms on the partial instantiation given bellow.



Exercise 3

Consider the constraint network described below :



Question 1 • Write the initial propagation queue (list of arcs) that AC3 will start with. Remember to include **both directions** for each constraint. For example, for $x_1 \neq x_2$, include $\overrightarrow{x_1 \neq x_2}$ and $\overrightarrow{x_2 \neq x_1}$.

Question 2 • Using the AC-3 algorithm, manually process the propagation queue step by step. For each step :

— Indicate which arc is being checked.
— Show any changes made to the domains of the variables.
— If a domain changes, add the necessary arcs back to the queue.

Question 3 • At the end of the process, write the final domains of all variables.

Question 4 • Was the constraint network arc-consistent after running AC-3 ? Explain briefly.

Exercise 4

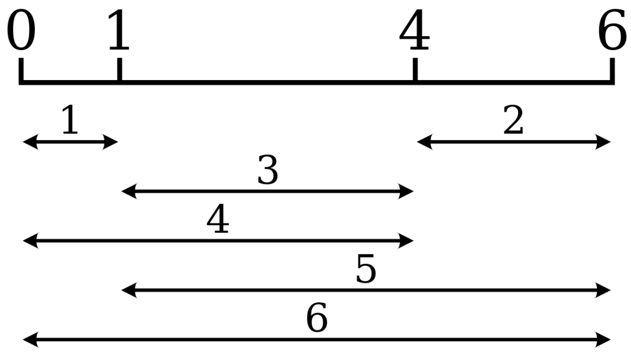
A Golomb ruler is a rule that contains marks at integer positions such that every pair of marks has a different length between them.

Question 1 • Model the problem as a constraint network $N = \langle X, D, C \rangle$.

Question 2 • Provide the optimization version of the problem, with the objective of returning the smallest Golomb ruler.

Question 3 • In pairs, conduct a comparative study of the different versions of the model, from the basic version `GolombRuler1.java` to the most refined and optimized version `GolombRuler5.java`. Write a brief report presenting the improvements made by each refinement and the gains achieved in terms of performance or accuracy.

Question 4 • In pairs, analyze whether the constraint $\texttt{tiks}[\texttt{m}-1] \geq \texttt{m}(\texttt{m}-1)/2$ can be added to the model. Discuss whether it is beneficial to include it or not, and justify your answer in a brief report.



Exercise 5

This assignment is to be completed in pairs, and the report must be submitted jointly.

Question 1 • Model the Sudoku problem on paper as a Constraint Network $N = \langle X, D, C \rangle$.

You will find in your local repository the file `Sudoku.java`, which contains a Constraint Programming (CP) model of the Sudoku problem using the Choco-Solver library.

Question 2 • Modify the provided code to return all possible solutions to the Sudoku puzzle.

Next, we will test the declarative nature of CP by applying modifications to the provided model. Figure 1 depicts one of the most challenging instances of a 9×9 Sudoku puzzle.

Question 3 • Modify the CP model in `Sudoku.java` to solve the instance shown in Figure 1.

Question 4 • Adapt your code to handle both the instance in Figure 1 and the one in Figure 2.

Greater Than Sudoku

One variation of the classic Sudoku is the `Greater Than Sudoku` (GTSudoku), an example of which is shown in Figure 3. In addition to the constraints of the classic Sudoku, GTSudoku introduces comparison symbols (`>` and `<`) in the grid. These symbols indicate inequality constraints between adjacent cells within the same sub-grid.

8								
		3	6					
	7			9		2		
	5				7			
				4	5	7		
			1				3	
		1					6	8
		8	5				1	
	9					4		

FIGURE 1 – Difficult instance of 9×9 Sudoku

	G			F	8	9	6	4	B	D	5			3	
6	C					4	E	2	7					5	9
			D			G	7	F	E			6			
		4	3	A							6	1	B		
7			5	8	F					B	E	9			G
8				9			4	D			3				2
C	1	3				6			G				F	4	5
9	D	B			G					F			7	A	6
G	B	A			2					7			5	6	D
5	6	F				A			2				8	7	4
D				6			9	5			G				F
3			C	B	5					A	4	G			1
		9	6	G							7	2	C		
			G			B	D	C	5			F			
4	3					8	2	G	F					1	7
	8			5	9	E	A	1	3	2	D			G	

FIGURE 2 – Instance of 16×16 Sudoku

Question 5 • Revise the model in `Sudoku.java` to solve the GTSudoku instance in Figure 3.

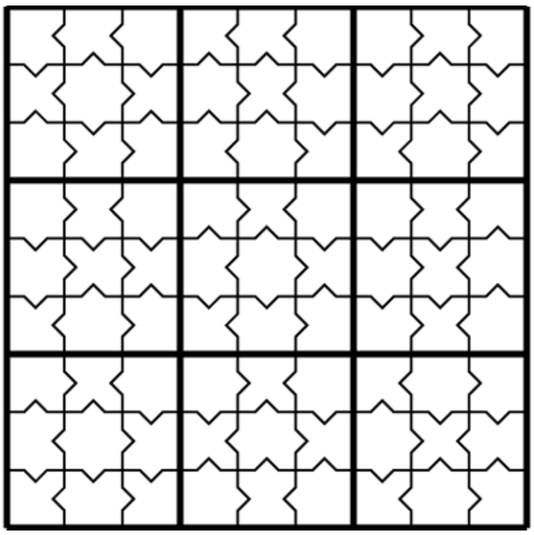


FIGURE 3 – Instance of *GT*-Sudoku (9×9)