Winning World War II with constraint programming

A practical on constraint modeling

Installation

You will use miniCP for this practical (`http://www.minicp.org/`). The code provided contains the complete solver, all the constraints you need are already implemented, but feel free to develop your own constraints if you need to.
We suggest to use an IDE, you will find all recommendation for the installation and the documentation in `http://www.minicp.org/`[1].
At each step, you will have to complete piece of code. Each part of the practical corresponds to one file, and each questions to one method.
The code to be completed is in `code/src/main/java/minicp/enigma`.

During the session, you will need to type passphrases in our Discord server, so join us at `https://discord.gg/PyVPDwhaqA`.

You never used Discord before ? Don't worry, you don't need to create an account (press escape when prompted to do so). Just make sure that your username is your complete name (first name + last name). If you already have an account, please rename yourself for this server with the command `/nick first name last name`.
If you need any help, please ask on this server, we are available for questions in the voice and text channels "Général".

[1] The Javadoc is not up to date, especially if you need constraints that are not in the documentation take a look at the package `code/src/main/java/minicp/engine/constraints` for the complete list

Warmup (`Warmup.java`)

Let's start with a simple *transposition* code. A transposition code uses a **static** transposition table: a permutation of the alphabet (a character will always be encoded by the same other character). Here we pretend to be spies and we want to use constraint programming to encrypt and decrypt messages.

Exercice 1 `transposeEncodeChar`**:** Write a model with two variables, one standing for a text letter and another standing for the ciphered letter by the given transposition table. The search is already setup to find all feasible combinations.

Exercice 2 `transposeEncodeText`**:** Write a model with one variable per letter in the ciphertext. The solution will correspond to the result of encrypting the plaintext "IAMAREALSPYNOW" with the same transposition table.

Exercice 3 `transposeDecodeText`**:** In an attempt to hoard the rewards for themselves, the enemy has encrypted the next part of the hackathon with a transposition code! Fortunately, in a massive oversight, they didn't change the transposition table...

Write a model with one variable per letter in the plaintext. The solution will correspond to the message that was encrypted. The message is the content of the file `part2_encrypted.html`, and the output will be written in `part2.html`. Run your program and open `part2.html` in your favorite browser to continue the hackathon.