Programmation par contraintes

Exercice 1
On considère l'addition suivante :

```
  TWO
+ TWO
-----
 FOUR
```

où chaque lettre représente un chiffre différent (compris entre 0 et 9). On souhaite connaître la valeur de chaque lettre, sachant que la première lettre de chaque mot représente un chiffre différent de 0.

- Donnez un programme en qui résout ce problème.
- La variable F peut être contrainte d'avoir la valeur 1 en utilisant la borne consistance. Expliquez comment.
- Le domaine de la variable T peut être aussi réduit en utilisant la borne consistance. Expliquez comment.
- Donnez une solution du problème en expliquant comment vous l'avez trouvée. Indication: Il y a une solution avec $T = 7$.

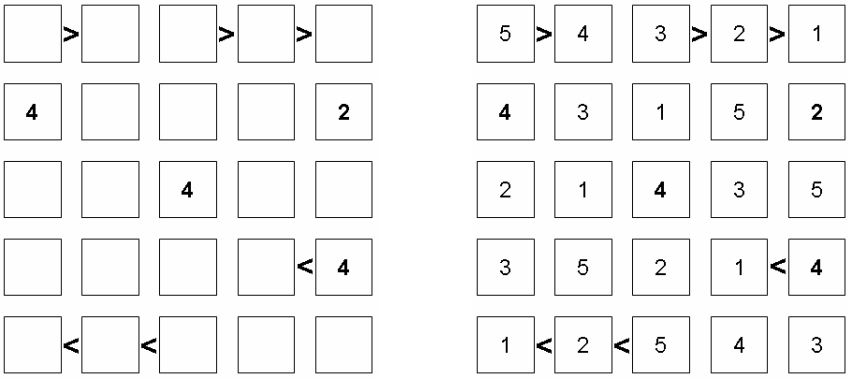
Exercice 2
En GNU Prolog les prédicats prédéfinis `fd_maximize(+Goal,?Var)` et `fd_minimize(+Goal,?Var)` prennent un but `Goal` et donnent donne la plus grande (petite) solution pour la variable `Var`. Attention: `Goal` devrait instancier `Var`. Ce qui est le cas, si `Goal` contient `fd_labeling(L)` ou `L` est une liste contenant `X`.

Écrire un programme GNU Prolog qui trouve la meilleure solution du problème du contrebandier vu en cours en utilisant `fd_maximize`.

Exercice 3
Une usine doit fabriquer deux produits sur deux machines. Les deux produits ont besoin des deux machines. La fabrication du premier produit dure 12 minutes sur la première machine et 30 minutes sur la deuxième. La fabrication du deuxième produit dure 24 minutes sur la première machine et 24 minutes sur la deuxième. La première machine peut tourner 400 heures et la deuxième 490 heures. Chaque premier produit rapporte 12 euros, chaque deuxième 20. But : maximiser le profit en produisant au moins 100 unités de chaque produit.

- Donnez un programme pour résoudre ce problème.

Exercice 4
Le Futoshiki est un jeu très simple. Il est basé sur une simple grille dans laquelle sont inscrits des nombres suivant quelques règles très simples. Sur une grille de 5x5, les nombres de un à cinq doivent être placés dans chaque ligne et chaque colonne, sans aucune répétition. Les signes "plus grand que" ou "plus petit que" entre les cases sont des indices qui doivent obligatoirement être respectés. Voici un exemple de Futoshiki et sa solution.



- Modéliser le problème **de l'exemple** comme un problème de satisfaction de contraintes (Quelles sont les variables, leur domaines et les contraintes ?)
- Donnez un programme qui résout le Futoshiki **de l'exemple**.
- Donnez un programme qui résout un Futoshiki **quelconque** de taille 5x5.

Exercice 5

Pour un entier $n > 0$ on considère des suites $x_0, x_1, \ldots x_n$, où tous les éléments sont dans $\{0, \ldots, n\}$. Une suite est appelé magique, si i apparaît exactement x_i fois dans la suite. Par exemple, pour $n = 3$, la suite $1, 2, 1, 0$ est magique car 0 apparaît 1 fois, 1 apparaît 2 fois, 2 apparaît 1 fois et 3 apparaît 0 fois.

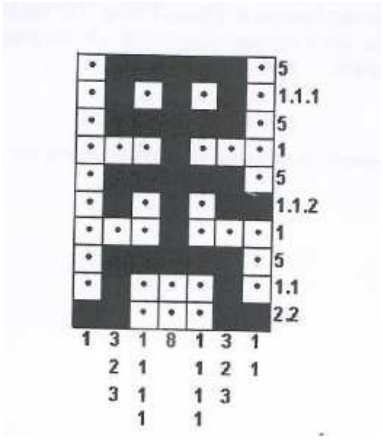
- Donnez un programme pour résoudre ce problème pour $n = 3$. Combien y-a-t-il des solutions ?
- Donnez un programme pour un n en général.

Exercice 6

(Source: wikipedia) Le but du jeu des nonogrammes consiste à retrouver les cases noires dans chaque grille. Les chiffres donnés sur le côté et en haut de la grille vous donnent des indices. Ils indiquent la taille des blocs de cases noires de la ligne ou de la colonne sur laquelle ils se trouvent.

Par exemple 3,4 à droite d'une ligne indique qu'il y a, de gauche à droite, un bloc de 3 cases noires puis un bloc de 4 cases noires sur cette ligne. Chaque grille résolue fait découvrir un dessin.

Exemple d'une grille remplie:



- Écrire un programme qui résout un nonogramme quelconque. Celui de l'exemple est donné par

```
nonogramme(10,7,[[5],[1,1,1],[5],[1],[5],[1,1,2],[1],[5],[1,1],[2,2]],
               [[1],[3,2,3],[1,1,1,1],[8],[1,1,1,1],[3,2,3],[1,1]]).
```

D'autres exemples se trouvent dans `~habermeh/clp/nonogramme.pl`

On peut commencer en donnant un programme spécifique pour le nonogramme le plus simple.