MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy

Rapport de projet INF402

Le futoshiki

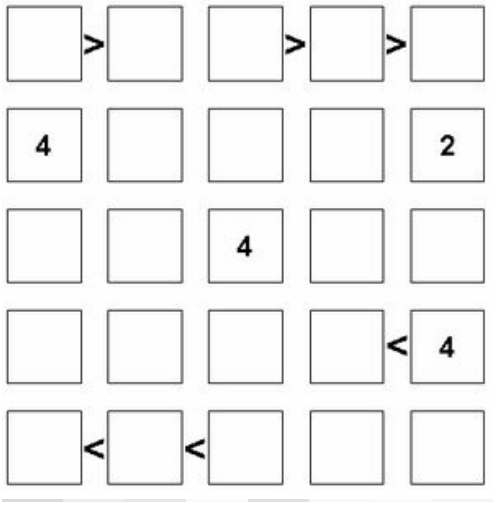
MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy

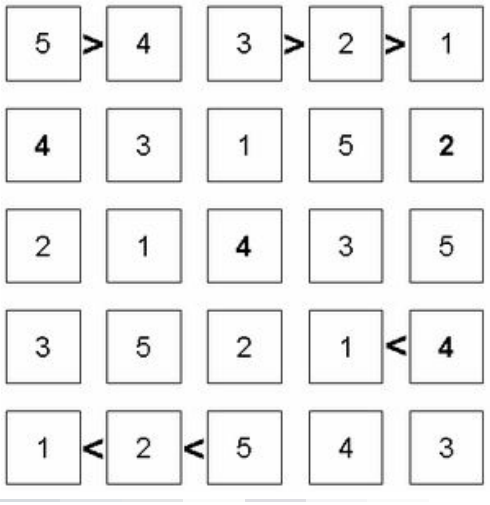
Le futoshiki

Le futoshiki, qui signifie « non égal » en Japonais, est un casse-tête japonais introduit en Europe au début des années 2000.

Son principe est simple, le jeu est composé d'une grille carrée de taille largeur N (2<=N<=9). Le joueur doit réussir à placer tous les nombre de 1 à N dans chaque ligne et chaque colonne tout en respectant les signes « inférieur à » et « supérieur à ».

Exemple : (Grille 5x5)





Une des contraintes principales de ce projet est d'arriver à modéliser la gestion des signes dans le remplissage des lignes et colonnes

Traduction logique du problème

Pour commencer, nous nous restreignons à une grille de taille 4x4.

MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy

Nous avons tout d'abord traduit le fait de vérifier que les colonnes et lignes de la grille doivent contenir une seule occurrence d'un nombre n et que chaque une case de la grille contient un unique nombre n.

Nous utilisons le littéral $x_{i,j,n}$ avec i,j les indices des lignes et colonnes comme pour une matrice carrée.

Règle 1 :

$$\forall (i,j,n) \in \{1,..,4\},\ x_{i,j,n} \Leftrightarrow (\wedge_{j1\in[1,4]\setminus j}\overline{x_{i,j1,n}}) \wedge (\wedge_{i1\in[1,4]\setminus i}\overline{x_{i1,j,n}}) \wedge (\wedge_{n'\in[1,4]\setminus n}\overline{x_{i,j,n'}})$$

ce qui s'écrit en clauses :

$$\forall (i,j,n) \in \{1,..,4\},\ x_{i,j,n} \vee (\vee_{j1\in[1,4]\setminus j}\ x_{i,j1,n}) \vee (\vee_{i1\in[1,4]\setminus i}\ x_{i1,j,n}) \vee (\vee_{n'\in[1,4]\setminus n}\ x_{i,j,n'})$$

et

$$\forall (i,j,n) \in \{1,..,4\},(\ \overline{x_{i,j,n}} \vee \overline{x_{i,j1,n}}) \text{ avec } j1 \in \{1,..,4\} \setminus j$$

$$\forall (i,j,n) \in \{1,..,4\},(\ \overline{x_{i,j,n}} \vee \overline{x_{i1,j,n}}) \text{ avec } i1 \in \{1,..,4\} \setminus i$$

$$\forall (i,j,n) \in \{1,..,4\},(\ \overline{x_{i,j,n}} \vee \overline{x_{i,j,n'}}) \text{ avec } n' \in \{1,..,4\} \setminus n$$

Nous traduisons ensuite, la contrainte du signes « inférieur ».
Nous considérons deux couples de coordonnées, (i1,j1) et (i2,j2) tel que (i1,j1)≠(i2,j2) avec soit i1=i2 ou j1=j2.
Si on a la case{i1,j1} < case{i2,j2} , on obtient alors la clause suivante.

Règle 2 :

$$\forall n \in \{2,..,4\}\ x_{i2,j2,n} \Leftrightarrow \vee_{n'<n} x_{i1,j1,n'}$$

$$\forall n \in \{1,..,3\}\ x_{i1,j1,n} \Leftrightarrow \vee_{n'>n} x_{i2,j2,n'}$$

$$\overline{x_{i2,j2,1}}$$

$$\overline{x_{i1,j1,4}}$$

ce qui se traduit en clauses :

$$\forall n \in \{2,..,4\},\qquad \overline{x_{i2,j2,n}} \vee (\vee_{n'<n} x_{i1,j1,n'})$$

et $\forall n' < n \qquad x_{i2,j2,n} \vee \overline{x_{i1j1,n'}}$

MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy

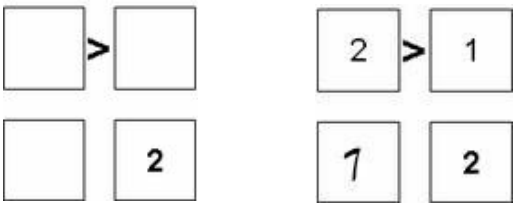
$\forall n \in \{1, .., 3\}$, $\quad \overline{x_{i1,j1,n}} \vee (\vee_{n'>n} x_{i2,j2,n'})$

et $\forall n' > n \quad x_{i1,j1,n} \vee \overline{x_{i2j2,n'}}$

$\overline{x_{i2j2,1}}$

et $\overline{x_{i1,j1,4}}$

Exemple sur une matrice 2x2:



Règle 1 :

x111 = -x112.-x121.-x211

:= (-x111+-x112) . (x111+x112+x121+x211) . (-x111+-x121) . (-x111+-x211)

x112 = -x111.-x122.-x212

:= (-x111+-x112) . (x111+x112+x122+x212) . (-x112+-x122) . (-x112+-x212)

x121 = -x122.-x111.-x221

:= (-x111+-x121) . (x111+x121+x122+x221) . (-x121+-x122) . (-x121+-x221)

x122 = -x121.-x112.-x222

:= (-x112+-x122) . (x112+x121+x122+x222) . (-x121+-x122) . (-x122+-x222)

x211 = -x212.-x221.-x111

MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy

:= (-x111+-x211) . (x111+x211+x212+x221) . (-x211+-x212) . (-x211+-x221)

x212 = -x211.-x112.-x222 :

= (-x112+-x212) . (x112+x211+x212+x222) . (-x211+-x212) . (-x212+-x222)

x221 = -x222.-x211.-x121

:= (-x121+-x221) . (x121+x211+x221+x222) . (-x211+-x221) . (-x221+-x222)

x222 = -x221.-x212.-x122

:= (-x122+-x222) . (x122+x212+x221+x222) . (-x212+-x222) . (-x221+-x222)

Règle 2 :

Case {1,2} < case {1,1} donc :

x121 <=> x112 := (-x121+x112) . (x121+-x112)

x112 = x121 := (-x121+x112) . (x121+-x112)

(ce sont les mêmes car cas particulier)

-x111

-x122

et on a par hypothèse : x222

Forme Conjonctive :

(-x111+-x112) . (x111+x112+x121+x211) . (-x111+-x121) . (-x111+-x211) . (-x111+-x112) .
(x111+x112+x122+x212) . (-x112+-x122) .(-x112+-x212).(-x111+-x121) .
(x111+x121+x122+x221) . (-x121+-x122) . (-x121+-x221) . (-x112+-x122) .
(x112+x121+x122+x222) . (-x121+-x122) . (-x122+-x222) .(-x111+-x211) .
(x111+x211+x212+x221) . (-x211+-x212) . (-x211+-x221) .(-x112+-x212) .
(x112+x211+x212+x222) . (-x211+-x212) . (-x212+-x222) . (-x121+-x221) .
(x121+x211+x221+x222) . (-x211+-x221) . (-x221+-x222) . (-x122+-x222) .
(x122+x212+x221+x222) . (-x212+-x222) . (-x221+-x222) .(-x121+x112) . (x121+-x112) .
-x111 .-x122. x222

Ce qui donne, une fois simplifiée :

(-x111) . (x112) . (x121) . (-x122) . (x211) . (-x212) . (-x221) . (x222)

MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy

On peut voir que la seule solution possible est :

['x111', 'x112', 'x121', 'x122', 'x211', 'x212', 'x221', 'x222']

= [0 , 1 , 1 , 0 , 1 , 0 , 0 , 1]

A l'aide de la traduction des clauses on peut prévoir combien de clause on a et créer le fichier DIMACs convenable .

<u>En format DIMACS (le fichier équivalent) :</u>

```
p cnf 222 36

-111 -112  0
111 112 121 211  0
-111 -121  0
-111 -211  0
-111 -112  0
111 112 122 212  0
-112 -122  0
-112 -212  0
-111 -121  0
111 121 122 221  0
-121 -122  0
-121 -221  0
-112 -122  0
112 121 122 222  0
-121 -122  0
-122 -222  0
-111 -211  0
111 211 212 221  0
-211 -212  0
-211 -221  0
-112 -212  0
112 211 212 222  0
-211 -212  0
-212 -222  0
-121 -221  0
121 211 221 222  0
-211 -221  0
-221 -222  0
-122 -222  0
122 212 221 222  0
```

MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy

```
-212 -222   0
-221 -222   0
-121 112   0
121 -112   0
-111  0
222   0
```

Présentation des programmes :

Pour l'implémentation du jeu nous avons réalisé trois programmes.

Le premier programme "dimacs" effectue la transformation des données contenues dans la grille de futoshiki initiale en un fichier DIMACS. Ce programme dépend du module "dimacs" et "fonctions". "dimacs" prend, en entrée, un fichier contenant les données initiales (dont le format sera explicité dans le README.txt) et la taille de la grille. Il produit un fichiers appelé "*dimacs.txt*" qui contient les clauses en format DIMACS.

Le deuxième programme "3_SAT" effectue la traduction du fichier "dimacs.txt" en format 3-SAT. Il écrit le résultat dans un nouveau fichier sortie appelé "dimacs.txt_3_SAT". Ce programme dépend du module "3_SAT" et du module "fonctions". Le programme prend comme unique argument "dimacs.txt".

Le dernier programme "res_dimacs" dépend du module "res_dimacs" et du module "fonctions". Il prend en arguments deux éléments : le fichier solution du SAT solveur et la taille de la grille. "res_dimacs" traduit le fichiers solution du sat solveur dans "fichier.txt" contenant les valeurs de chaques cases ligne par ligne.

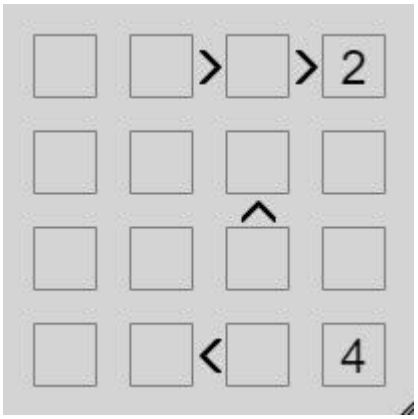
Pour finir, nous avons écrit un programme python utilisant la librairie Tkinter. Ce programme ouvre une fenêtre graphique contenant la grille finale remplie correspondant au fichier "fichier.txt" .

Afin de faciliter l'exécution de programme nous avons créé un script shell qui nous nous permet d'exécuter les programmes dans l'ordre avec les bons arguments en une seule ligne de commande. Il prend en arguments le fichier entré et la taille de la grille . Il cré le fcihier "dimacs.txt" , puis le transforme en 3-SAT appelé "dimacs.txt_3_SAT" , puis on execute le SAT solveur sur ce dernier fichier , et on affiche le résultat .

Tests d'exécution du programme :

Grille 4x4:

MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy



Fichier entrée :

2

4

1 4 2

4 4 4

1 3 1 2

1 4 1 3

2 3 3 3

4 2 4 3

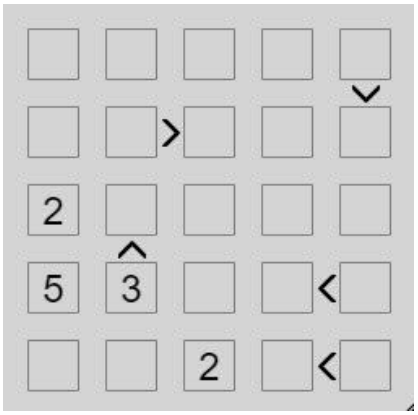
ligne de commande: ***./monsat.sh entree_grille_4.txt 4***

sortie

1	4	3	2
4	2	1	3
2	3	4	1
3	1	2	4

Grille 5x5:

MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy



Fichier entrée :

4

5

3 1 2

4 1 5

4 2 3

5 3 2

2 3 2 2

2 5 1 5

3 2 4 2

4 4 4 5

5 4 5 5

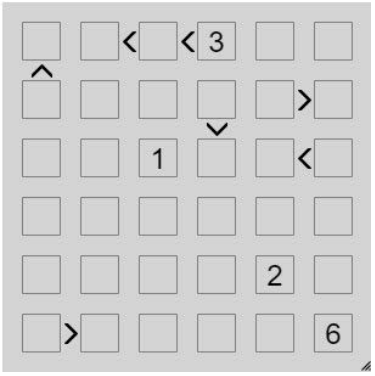
ligne de commande: ***./monsat.sh entree_grille_5.txt 5***

Sortie:

3	2	1	5	4
4	5	3	2	1
2	1	5	4	3
5	3	4	1	2
1	4	2	3	5

MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy

Grille 6x6:



Fichier entrée :

4

7

1 4 3

3 3 1

5 5 2

6 6 6

1 2 1 3

1 3 1 4

1 1 2 1

2 6 2 5

3 4 2 4

3 5 3 6

6 2 6 1

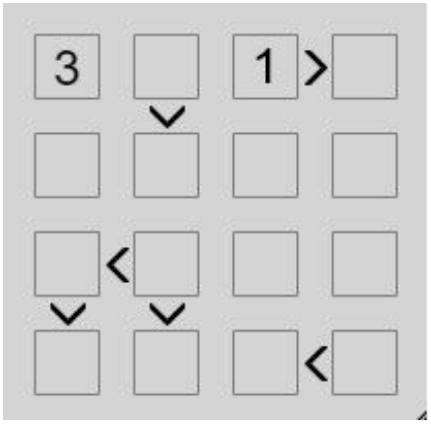
ligne de commande: ***./monsat.sh entree_grille_6.txt 6***

Sortie:

5	1	2	3	6	4
6	2	3	5	4	1
2	6	1	4	3	5
3	4	6	1	5	2
1	5	4	6	2	3
4	3	5	2	1	6

MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy

Grille 4x4 insatisfaisable:



Fichier entrée :

2

6

1 1 3

1 3 1

1 4 1 3

2 2 1 2

3 1 3 2

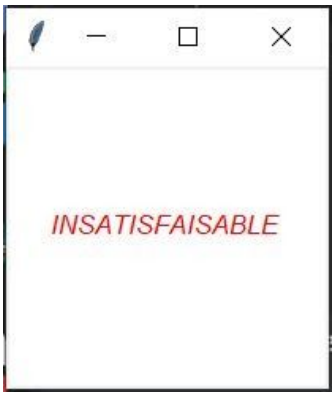
4 1 3 1

4 2 3 2

4 3 4 4

ligne de commande: ***./monsat.sh entree_grille_faux_4.txt 4***

Sortie:



MAURY Norman
ABU ZAKI Teimur
MIN1
SEBAN Jérémy

Conclusion:

Ce projet nous a permis d'avoir une approche plus concrète de la logique en appliquant celle-ci à un exemple de casse tête.

Pour la réalisation de nos tests, Nous avons utilisé ce site pour générer des exemples et vérifier nos résultats. Chacun des test que nous avons effectués était concluant.