TP 03 – Le réflexe `minisat`

Que faire quand on doit, malgré tous nous efforts pour éviter cela, résoudre des instances de problèmes NP-complets ? On utilise un solveur de problèmes NP-complets !

D'après la définition de NP-complétude, un programme qui résout (le plus efficacement que l'on sache faire) les instances de n'importe quel problème NP-complets résout aussi (le plus efficacement que l'on sache faire) les instances de n'importe quel autre problème NP-complet (via les réductions !). Puisque **SAT** est le plus « naturel/simple/élémentaire/souple » vers lequel réduire, c'est celui-là qui est en général choisi pour écrire des **solveur**.

Le solveur `minisat` permet de décider la satisfaisabilité de formules en forme normale conjonctive (CNF). De tels solveurs partagent le même format pour écrire les formules CNF : `dimacs`.

doc `minisat` : `http://www.dwheeler.com/essays/minisat-user-guide.html`
installer `minisat` chez soi : `http://minisat.se/`
doc `dimacs` : `http://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html`

Exercice 1. *Première utilisation de `minisat`*

1. Écrire la formule $(a \vee b \vee \neg c \vee d) \wedge (\neg b \vee c) \wedge (\neg a \vee \neg d)$ au format `dimacs`.
2. Tester la satisfaisabilité de cette formule en utilisant le solveur `minisat`.
3. On considère la formule :

$$\phi = (\neg a \wedge b \wedge d) \vee (a \to (c \vee (b \wedge \neg c \wedge d)))$$

   (a) Mettre ϕ sous forme normale conjonctive.
   (b) Écrire la formule CNF obtenue au format `dimacs`.
   (c) Utiliser le solveur `minisat` pour décider si ϕ est satisfaisable.
   (d) Si oui, décider si elle admet au moins deux valuations satisfaisantes.
4. Proposer une procédure qui permettrait de décider si une formule est une tautologie.

Exercice 2. *Réduction de* **3-Color** *à* **SAT**

Le problème **3-Color** consiste à décider si un graphe est 3-coloriable. Une *3-coloration* est une assignation d'une couleur, parmi trois couleurs, à chaque sommet du graph. Une coloration est *valide* si deux sommets adjacents n'ont pas la même couleur.

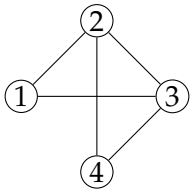
3-Color
entrée : Un graphe non-orienté $G = (V, E)$
question : G admet-il une 3-coloration valide ?

1. Montrer que **3-Color** $\leq_m^p$ **SAT**.

Un graphe sera représenté par une suite de nombres de la façon suivante. Le premier entier indique le nombre de sommets n, le second le nombre d'arêtes p. Viennent ensuite $2\,p$ entiers compris entre 1 et n, chacun des p couple décrivant les extrémités d'une arête. Ainsi,

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

représente le graphe à 4 sommets et 5 arêtes suivant.



2. Dessiner le graphe représenté par la séquence

   `10 15 1 2 2 3 3 4 4 5 5 1 1 6 2 7 3 8 4 9 5 10 6 8 7 9 8 10 9 6 10 7.`

3. Implémenter votre réduction de la question 1, prenant en entrée un graphe selon ce format, et produisant en sortie une formule au format `dimacs`.
4. En faire un programme qui décide si un graphe donné est 3-coloriable (en utilisant `minisat`).
5. Améliorer le programme précédent de telle sorte que si le graphe est 3-coloriable alors un 3-coloriage est obtenu. Ce programme doit indiquer la couleur de chaque sommet.
6. Tester votre algorithme sur des exemples simples et sur des graphes aléatoires. Que se passe-t-il quand on fait grossir la taille des graphes ?

Exercice 3. *Réduction de* **Sudoku** *à* **SAT**

La grille de jeu est un carré de neuf cases de côté, subdivisé en autant de sous-grilles carrées identiques, appelées "régions". Le but du jeu est de remplir cette grille avec des chiffres allant de 1 à 9 en veillant toujours à ce que chaque ligne, chaque colonne et chaque région ne contiennent qu'une seule fois tous les chiffres allant de 1 à 9. Au début du jeu, un certain nombre de chiffres sont déjà placés.
Étant donnée une grille de Sudoku, nous souhaitons savoir s'il existe une solution. Nous allons pour cela modéliser ce problème par une formule propositionnelle. Pour cela, on pourra utiliser $9 \times 9 \times 9$ variables propositionnelles $c_{i,j,k}$ avec $i, j, k \in \{1, ..., 9\}$, qui codent le fait que la case (i, j) contient le chiffre k.

1. Écrire l'ensemble des contraintes modélisant le problème sous forme d'une formule CNF :
   - Au moins une valeur par case : $\bigwedge_{i,j \in \{1,...,9\}} \bigvee_{k \in \{1,...,9\}} c_{i,j,k}$.
   - Au moins une fois chaque chiffre sur chaque ligne.
   - Au moins une fois chaque chiffre sur chaque colonne.
   - Au moins une fois chaque chiffre dans chaque région.
   - Au plus une valeur par case.
   - Au plus une fois chaque chiffre sur chaque ligne.
   - Au plus une fois chaque chiffre sur chaque colonne.
   - Au plus une fois chaque chiffre dans chaque région.

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

Représentation textuelle :

```
...81...5....2..3.8....5.49
4..1...63.2.....9.37...2..8
78.2....6.4..5....6...98...
```

2. **Bonus :** Écrire un programme qui lit sur l'entrée standard une grille de Sudoku au format textuelle et produit l'ensemble des contraintes modélisant le problème au format `dimacs`. Ne pas oublier d'ajouter les contraintes pour les chiffres initialement placés.
   Vous trouverez des grilles de test avec les mots clés `top95` et `top1465`.