[IA02] Sujet de TP 2 : Utilisation de solveurs SAT

Information	Valeur
Auteur	Sylvain Lagrue (sylvain.lagrue@utc.fr (mailto:sylvain.lagrue@utc.fr))
Licence	Creative Common CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)
Version document	1.5.1

Exercice 1 : prise en main d'un solveur SAT

Le format DIMACS

La plupart des solveurs SAT utilisent comme format de représentation de CNF le format DIMACS. C'est par exemple le format d'entrée pour la compétition SAT (https://www.satcompetition.org/) dont les benchmarks de l'édition 2018 sont accessibles ici (http://sat2018.forsyte.tuwien.ac.at/benchmarks/).

Un fichier DIMACS est composé de caractères ASCII et se veut être très simple à *parser* (attention, les fins de ligne sont au format Unix). Il est composé de 3 principales parties :

- Une première partie de commentaires. Un ligne de commentaire commence par le caractère `c` .
- Une deuxième partie détaillant le contenu du fichier. Il est de la forme `p FORMAT #VARIABLES #CLAUSES`
- L'ensemble des clauses. Chacune de ces lignes de clauses est composée d'entiers (positifs ou négatifs) et *devrait* se terminer par `0` . Un nombre négatif représente un littéral négatifs, tandis qu'un littéral positif est représenté par un nombre positif.

Exemple

Si l'on considère le vocabulaire $\{A, B, C\}$ (dans cet ordre), la formule $(\neg C \vee A) \wedge (B \vee \neg A \vee C)$ est représentée par le fichier suivant :

```
c
c Exemple de fichier .cnf file.
c
p cnf 3 2
-3 1 0
2 -1 3 0
```

gophersat

Le solveur **gophersat** est un programme open-source écrit en Go (https://golang.org/). Ses sources sont accessibles sur https://github.com/crillab/gophersat (https://github.com/crillab/gophersat). Une version compilée est disponible sur le cours Moodle. C'est un solveur SAT, MAX-SAT et pseudo-booléen. Il est également capable de faire du comptage de modèles.

Question

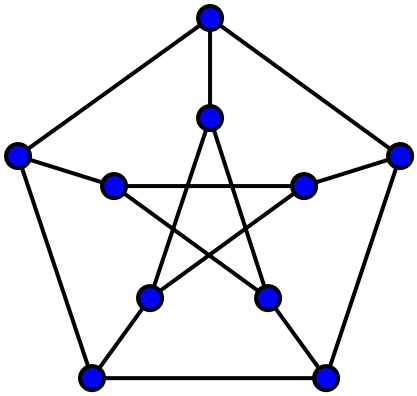
Tester le solveur et ses différentes options sur les différents fichiers `.cnf` disponibles sur l'ENT. Ces fichiers proviennent du site https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html (https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html) et du site de gophersat. Que pouvez-vous dire de la sortie ?

Exercice 2 : la licorne

1. Écrire (à la main) le fichier DIMACS correspondant au problème de la licorne de la feuille de TD n°2.
2. Montrer que que votre modélistation est cohérente.
3. Utiliser le solveur SAT pour répondre aux questions suivantes :
   - Peut-on déduire que la licorne a une corne ? Qu'elle n'a pas de corne ?
   - Peut-on déduire qu'elle est mythique ? Qu'elle n'est pas mythique ?

Exercice 3 : problème de coloration

Écrire un programme python permettant de générer le fichier `.cnf` permettant de résoudre le problème de coloration de graphe (https://fr.wikipedia.org/wiki/Coloration_de_graphe) à 3 couleurs suivant (schéma issu de wikipedia (https://fr.wikipedia.org/wiki/Fichier:Petersen1_tiny.svg)) :



1. Pour cela, créer une fonction qui, étant donné un graphe génère la chaîne de caractères contenant le problème sous format DIMACS associé.
2. Convertir l'éventuelle solution en une solution lisible.

Quelques liens python utiles :

- str.split (https://docs.python.org/3/library/stdtypes.html#str.split) et plus généralement les actions de base sur les chaînes de caractères `str` (https://docs.python.org/3/library/stdtypes.html#str)
- Les fstrings (https://www.python.org/dev/peps/pep-0498/)
- la bibliotèque itertools (https://docs.python.org/3/library/itertools.html) et en particulier combinations (https://docs.python.org/3/library/itertools.html#itertools.combinations)
- lecture/écriture de fichiers sous python (https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files)
- la bibliothèque subprocess (https://docs.python.org/fr/3/library/subprocess.html) permet d'appeler des programmes externes…