Feuille d'exercices n°6.2 - Classes de complexité P et NP

Notions abordées

- montrer qu'un problème est NP
- réduction polynomiale d'un problème à un autre

Exercice 1 : Coloration et couverture de graphe

Dans cet exercice on considère deux problèmes de décision associés à des problèmes de minimisation sur les graphes non orientés. Le problème de coloration consiste à colorier les sommets d'un graphe avec le moins de couleur possible de sorte que deux sommets reliés par une arête ne soient pas de la même couleur. Le problème de couverture par des cliques consiste à répartir les sommets d'un graphe en le moins de groupe possible de sorte que chacun de ces groupes forme une clique♣.

Q. 1 Formaliser les problèmes de décision ColorationMin et Couv.Clique respectivement associés au problème de coloration et au problème de couverture par des cliques.

Q. 2 Montrer que Couv.Clique $\preccurlyeq_P$ ColorationMin et ColorationMin $\preccurlyeq_P$ Couv.Clique.

Exercice 2 : Vérifier qu'une formule est satisfaite

On s'intéresse dans cet exercice au problème qui consiste à tester la satisfaction d'une formule de la logique propositionelle par un environnement propositionnel donné.

Satisfait : $\begin{cases} \textbf{Entrée :} \text{ Une formule de la logique propositionnelle } G, \text{ un env. propositionnel } \rho \\ \textbf{Sortie :} \ [\![G]\!]^\rho \end{cases}$

On considère les formules de la logique propositionnelle construites sur l'ensemble de variables $\mathscr{P} = \{p_i \mid i \in \mathbb{N}\}$. Ainsi une variable peut être identifiée par un indice, ce qui permet de représenter les formules par le type OCaml suivant. Bien qu'on ait une infinité de variables à disposition, une formule ne peut faire intervenir qu'un nombre fini de variables. Ainsi, bien qu'un environnement propositionnel complet sur $\mathscr{P}$ soit défini par une suite de $\mathbb{B}^{\mathbb{N}}$, il suffit pour interpréter une formule d'avoir une suite finie de valeurs de vérités. Plus précisément, il suffit d'avoir autant de valeurs de vérités que la formule a de variables distinctes.

```
type flp =
  | Top
  | Bot
  | Var of int
  | Not of flp
  | And of flp * flp
  | Or of flp * flp
  | Imply of flp * flp
  | Equiv of flp * flp
```

Cependant, on choisit dans cet exercice de prendre en entrée au moins autant de valeurs de vérité que la formule a d'occurrences de variables, sous la forme d'une chaîne de caractères `'V'` et `'F'`. Une telle chaîne peut être vue comme une pioche de booléens.Pour une formule donnée, on construit

♣. On rappelle qu'une **clique** est un ensemble de sommets qui induit un sous-graphe complet.

alors un environnement partiel en y piochant, à chaque occurrence d'une nouvelle variable, son interprétation.

Exemples :

- La chaîne `"VV"` pour la formule $(p_1 \wedge p_2) \vee p_2$ sera considérée comme trop courte car on a trois occurrences de variables et seulement deux caractères.
- La chaîne `"VVF"` pour la formule $(p_1 \wedge p_2)$ correspond à l'environnement prop. ($p_1 \mapsto$ V, $p_2 \mapsto$ V).
- La chaîne `"VVF"` pour la formule $(p_1 \wedge p_2) \vee p_2$ correspond aussi à l'env. prop. ($p_1 \mapsto$ V, $p_2 \mapsto$ V).
- La chaîne `"VVF"` pour la formule $(p_1 \wedge p_2) \vee p_3$ correspond à l'env. prop. ($p_1 \mapsto$ V, $p_2 \mapsto$ V, $p_3 \mapsto$ F).
- La chaîne `"VVF"` pour la formule $(p_3 \wedge p_2) \vee p_3$ correspond à l'environnement prop. ($p_3 \mapsto$ V, $p_2 \mapsto$ V).

Q. 1 Définir une fonction `compte (f: flp) : int` qui compte avec multiplicité le nombre de variables apparaissant dans `f`. Par exemple, pour la formule $p_1 \wedge p_1 \wedge p_4$ on attend 3 et pour $p_1 \wedge (\top \vee \bot)$ on attend 1.

Q. 2 Définir en OCaml deux exceptions `EnvTropPetit` et `EnvNonValide`.

Q. 3 Proposer un jeu de tests pour la fonction `deserialise` (décrite à la Q4) qui teste, entre autres, les comportements d'erreurs.

Q. 4 Définir en OCaml une fonction `deserialise (n:int) (s:string): bool array` qui désérialise la chaîne `s` en un tableau de `n` booléens si c'est possible, *i.e.* si la chaîne `s` commence par `n` caractères `'V'` ou `'F'`. Dans le cas où la chaîne `s` présente strictement moins de `n` caracatères, la fonction lèvera l'exception `EnvTropPetit`. Sinon, dans le cas où elle présente un caractère autre que `'V'` ou `'F'` parmi ses `n` premiers caractères, la fonction lèvera l'exception `EnvNonValide`. Par exemple pour l'entier 3 et la chaîne `"VVFFVVFF"`, la fonction calculera `[|true;true;false|]`.

Q. 5 Définir en OCaml une fonction `interprete (f:flp) (pre_env:bool array) : bool` qui calcule l'interprétation de la formule `f` en interprétant chaque nouvelle variable selon `pre_env` ♡. On utilisera un dictionnaire de type `(int, bool) Hashtbl.t` pour enregistrer les valeurs de vérités attribuées aux variables déjà rencontrées. *On veillera à bien parcourir toutes les variables de la formule lors de cette interprétation, malgré la possibilité d'évaluation paresseuse.*

Q. 6 Définir en OCaml une fonction `satisfait (s:string) (f:flp): bool` qui décide le problème Satisfait. En particulier cette fonction doit terminer sur toute entrée, donc elle renverra `false` dans le cas où `s` n'est pas une sérialisation convenable d'environnement propositionnel pour `f`.

♡. Comme expliqué plus haut, il s'agit d'interpréter chaque nouvelle variable rencontrée lors d'un parcours de la formule de gauche à droite selon la première valeur non utilisée de `pre_env`, et chaque variable déjà rencontrée de la même manière qu'à la précédente rencontre.

Exercice 3 : Appariement à 3 dimensions

On s'intéresse dans cet exercice à un problème de construction d'équipes, les équipes ayant ici 3 personnes. Afin de définir le problème qui nous intéresse dans cet exercice on définit d'abord ce qu'est un 3DM.

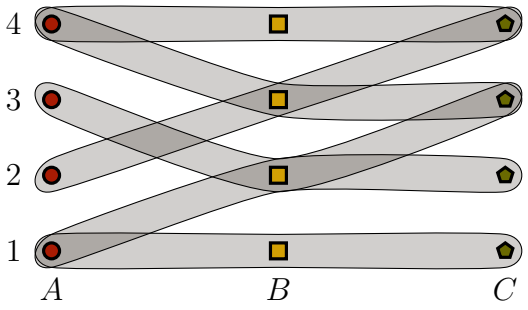
Appariement à 3 dimensions (3DM). Étant donné trois ensembles finis A, B et C et une relation ternaire $\mathcal{R} \subseteq A \times B \times C$, on appelle **appariement à 3 dimensions** (abrégé en **3DM**♣ dans la suite) un sous-ensemble $\mathcal{M}$ de $\mathcal{R}$ tel que les projections π_1, π_2 et π_3 sur chacune des composantes sont injectives sur $\mathcal{M}$, autrement dit deux triplets de $\mathcal{M}$ n'ont aucune composante en commun, à moins d'êtres égaux. Si A, B et C sont disjoints, un 3DM est un sous-ensemble des triplets de $\mathcal{R}$ tel que chaque élément de $A \cup B \cup C$ apparaît dans au plus un triplet. On appelle **taille** d'un 3DM son cardinal, c'est-à-dire le nombre de triplets qu'il contient.

Problème 3DM On s'intéresse au problème de décision associé au problème de trouver un 3DM de taille maximale, c'est-à-dire au problème 3DM défini ci-dessous.

$$\text{3DM} : \begin{cases} \textbf{Entrée :} \text{ Trois ensembles finis } A, B, C, \mathcal{R} \subseteq A \times B \times C, \text{ un seuil } K \in \mathbb{N}. \\ \textbf{Sortie :} \text{ Existe-t-il } \mathcal{M} \subseteq \mathcal{R} \text{ un 3DM de taille} \geqslant K \end{cases}$$

On représente ci-contre une instance de 3DM partiellement définie par

- $A = \{1, 2, 3, 4\}$,
- $B = \{1, 2, 3, 4\}$,
- $C = \{1, 2, 3, 4\}$ et
- $\mathcal{R} = \{(4,4,4), (4,3,3), (3,2,3), (2,3,4), (1,2,2), (1,1,1)\}$



Pour $K \leqslant 3$ cette instance est positive car $\mathcal{M} = \{(4,4,4), (3,2,3), (1,1,1)\}$ est un 3DM de taille 3.

On souhaite montrer que le problème 3DM est NP complet. On justifie d'abord qu'il est dans NP puis on montre qu'il est NP difficile par réduction depuis 3-SAT.

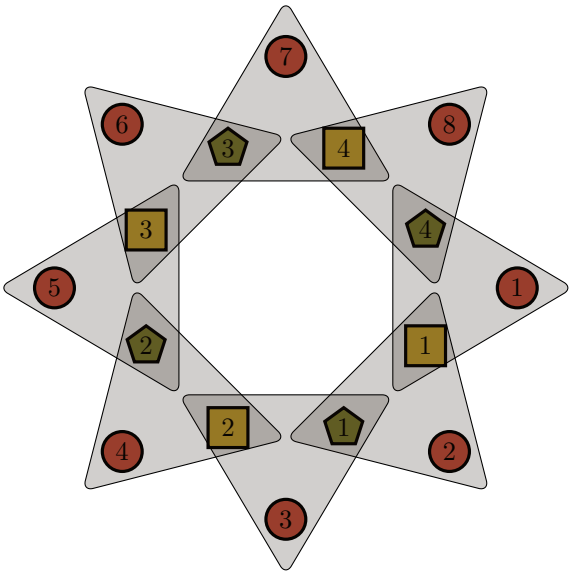
Q. 1 Justifier que 3DM $\in$ NP.

♣. DM pour *dimensionnal matching* en anglais.

Présentation du gadget pour réduction de 3SAT à 3DM.
Étant donné un entier $n \in \mathbb{N}^\star$, on considère l'instance partielle définie par les ensemble suivants avec la convention que $n+1 = 1$ et $2n+1 = 1$.

- $A = \{1, 2, \ldots, 2n\}$,
- $B = \{1, 2, \ldots, n\}$,
- $C = \{1, 2, \ldots, n\}$.
- $\mathcal{R} = \bigcup_{i=1}^{n} \{(2i, i, i), (2i+1, i+1, i)\}$

On présente ci-contre le gadget ainsi défini pour $n=4$.



Q. 2 Démontrer qu'un 3DM $\mathcal{M}$ de taille $n \in \mathbb{N}^\star$ du gadget ci-dessus est tel que :

$$\mathcal{M} = \bigcup_{i=1}^{n} \left\{ \left(2i, i, i\right) \right\} \quad \text{ou} \quad \mathcal{M} = \bigcup_{i=1}^{n} \left\{ \left(2i+1, i+1, i\right) \right\}$$

Q. 3 Proposer une réduction polynomiale de 3SAT à 3DM.
On pourra construire, pour chaque variable propositionnelle x_i de la formule, un gadget comme décrit ci-avant avec $n = n_i$ le nombre d'occurrences de x_i dans la formule, On pourra de plus ajouter des élements représéntant les clauses de la formule et les relier (dans $\mathcal{R}$) habilement aux éléments ronds des gadgets afin de représenter le fait qu'une clause est satisfaite grâce à une occurence d'une variable. Enfin on pourra ajouter des éléments "joker"♣ afin d'assurer que tous les éléments ronds puissent être couverts si la formule est statisfiable.

Q. 4 Exhiber la réduction dans le cas de l'entrée $\{x \vee y \vee z, \neg x \vee \neg x \vee \neg z\}$.

♣. ces éléments sont l'analogue des c_i et d_i introduits pour réduire 3-SAT à SUBSETSUM, ils sont utiles pour construire une solution du nouveau problème, pas pertient pour trouver quel environnement satsifait la formule instance de 3-SAT