Algorithmique et Programmation

TD n° 10 : $\mathcal{NP}$-complétude et réductions

École normale supérieure – Département d'informatique

algoL3@di.ens.fr

2015-2016

Exercice 1. Une *clause de Horn* est une formule logique en forme normale conjonctive où chaque clause contient au maximum un seul littéral positif. En d'autres termes une clause de Horn est ou bien une implication :

$$(X_1 \wedge X_2 \wedge \ldots \wedge X_k) \Rightarrow X_{k+1}$$

où les X_i sont des variables propositionnelles, ou bien une clause purement négative :

$$\overline{X_1} \vee \overline{X_2} \vee \ldots \vee \overline{X_k}$$

Notons qu'une implication avec $k = 0$ revient à affirmer que la variable propositionnelle à droite du signe $\Rightarrow$ est vraie de façon inconditionnelle (c'est un *fait*).

1. Donner un algorithme pour la satisfaisabilité (de la conjonction) d'un ensemble de clauses de Horn.
2. Expliquer comment l'implanter en temps linéaire en la taille de son entrée.

Exercice 2. Le problème de l'ensemble indépendant (*independent set*, en anglais) est le suivant : étant donné un graphe $G = (V, E)$ et un entier k, déterminer s'il existe un sous-ensemble $V' \subseteq V$ de cardinal k tel que si $(u, v) \in V'^2$ alors $(u, v) \notin E$. Montrer que ce problème est $\mathcal{NP}$-complet.

Exercice 3. Le problème 2-SAT exige que toutes les clauses soient satisfaites. Il est naturel de se demander s'il existe une assignation qui satisfait non nécessairement toutes les clauses, mais un grand nombre d'entre elles. Le problème Max-2-SAT consiste à décider, étant donnés un ensemble de clauses, chacune avec au plus deux littéraux et un entier k, s'il existe une assignation qui satisfait au moins k clauses.

1. Montrer que toute assignation qui satisfait $(x \vee y \vee z)$ peut être étendue en une assignation qui satisfait exactement 7 clauses parmi les 10 clauses suivantes et pas plus, alors que toutes les assignations restantes peuvent être étendues pour satisfaire au plus 6 clauses.

$$(x) \quad (y) \quad (z) \quad (w) \quad (\overline{x} \vee \overline{y}) \quad (\overline{y} \vee \overline{z}) \quad (\overline{z} \vee \overline{x}) \quad (x \vee \overline{w}) \quad (y \vee \overline{w}) \quad (z \vee \overline{w})$$

2. Montrer que Max-2-SAT est $\mathcal{NP}$-complet.

Exercice 4. Nous nous intéressons au problème 3-SAT, qui consiste à déterminer si une formule logique en forme normale conjonctive avec au plus trois littéraux par clause est satisfaisable. L'instance a n variables et m contraintes.

1. Démontrer que si $\psi \wedge x$ est insatisfaisable, alors toute assignation des variables satisfaisant ψ contiendrait $\overline{x}$.

2. Démontrer si x est un littéral, alors ψ et $(\psi \wedge x) \vee (\psi \wedge \overline{x})$ sont équivalentes (c.-à.-d. qu'ils ont les mêmes solutions). En déduire un algorithme récursif pour 3-SAT de complexité $O(Poly(m,n) \cdot 2^n)$.

Nous allons maintenant chercher à obtenir un algorithme asymptotiquement plus rapide. Si ψ n'est pas vide, alors elle s'écrit $\psi = (x \vee y \vee z) \wedge \psi'$, où x, y et z sont des littéraux.

3. Montrer que ψ est équivalente à

$$(x \wedge \psi') \vee (y \wedge \psi') \vee (z \wedge \psi').$$

   En déduire un algorithme récursif de complexité $O(Poly(m,n) \cdot 1.8393^n)$.

4. Un littéral x est dit pur si $\overline{x}$ n'apparaît pas dans ψ. Montrer que si ψ est satisfaisable, alors il existe une assignation des variables dans laquelle tous les littéraux purs sont vrais. Cela permet de se ramener au cas où aucun littéral n'est pur. Cela signifie que si ψ est non-triviale, on peut trouver deux clauses, l'une contenant x et l'autre $\bar{x}$.

   En déduire un algorithme récursif de complexité $O(Poly(m,n) \cdot 1.7692^n)$.

5. Justifier que dans chaque appel récursif, ou bien on a fait apparaître un littéral pur (qu'on peut donc éliminer), ou bien on fait apparaître une clause à deux littéraux. En déduire un algorithme de complexité $O(Poly(m,n) \cdot 1.6181^n)$.