TD 08 – NP-Complétude, encore (corrigé)

(SatN) **Exercice 1.** *Echauffement*

On appelle SAT-N le problème SAT restreint aux formules qui n'ont pas plus de N occurrences de la même variable.

1. Montrer que SAT-3 est au moins aussi dur que SAT. En déduire que pour tout $N \geq 3$, SAT-N est $\mathcal{NP}$-complet.

   ☞ SAT-3 est trivialement NP.

   Pour montrer qu'il est NP-complet, on part d'un problème de SAT et pour chaque littéral x_i apparaissant k fois avec k avec $k > 3$, on applique l'algorithme suivant : on remplace les occurrences de x_i par k nouveaux littéraux $y_{i_1}, \ldots, y_{i_k}$, et on rajoute les clauses $y_{i_1} \vee \overline{y_{i_2}}, \ldots, y_{i_{k-1}} \vee \overline{y_{i_k}}$ et $y_{i_k} \vee \overline{y_{i_1}}$.

   Les nouveaux littéraux apparaissent alors 3 fois chacun (une fois à la place de x_i, et deux fois dans les nouvelles clauses). De plus les nouvelles clauses entraînent que les y_{i_j} prennent tous la même valeur : si y_{i_j} est vrai, alors $y_{i_{j-1}}$ également, et inversement, si y_{i_j} est faux, alors $y_{i_{j+1}}$ doit l'être aussi (en toute rigueur on devrait rajouter des modulos k), et comme les clauses sont cycliques, on obtient que tous ces littéraux ont la même valeur.

   Enfin, si n est le nombre de littéraux, et m le nombre de clauses, le nouveau problème obtenu comporte au plus nm littéraux et $(m + nm)$ clauses, donc une instance de SAT se réduit polynomialement en une instance SAT-3 équivalente, et comme SAT-3 est NP, SAT-3 est NP-complet.

   Et comme pour tout $N \geq 3$, SAT-3 est un cas particulier de SAT-N, SAT-N est NP-complet.

2. Soit x une variable apparaissant dans une formule F de SAT-2. Trouver une formule équivalente à F et dans laquelle la variable x n'apparaisse plus. En déduire un algorithme polynomial pour SAT-2.

   ☞ Soit x une variable d'un système de clauses F de SAT-2, on transforme F selon les cas de sorte que le système obtenu soit satisfiable si et seulement si le système initial l'était :
   - si x (ou $\overline{x}$) n'apparaît qu'une seule fois, on supprime la clause dans laquelle il apparaît, car on est libre de mettre x (ou $\overline{x}$), donc toute la clause à vrai.
   - si x (ou bien $\overline{x}$) apparaît 2 fois dans une même clause, ou dans 2 clauses distinctes, on supprime la ou les clauses, pour la même raison.
   - si x et $\overline{x}$ apparaissent dans la même clause, on supprime la clause car elle est toujours à vrai.
   - si x et $\overline{x}$ apparaissent dans 2 clauses distinctes, $x \vee C_1$ et $\overline{x} \vee C_2$, alors le système de clauses est satisfiable si et seulement si au moins une des deux clauses C_1 ou C_2 peut être mise à vrai en même temps que les autres clauses (pour l'autre on choisit x pour compléter), ainsi :
     - → si $C_1 = C_2 = \emptyset$ alors le système n'est pas satisfiable.
     - → sinon on remplace les 2 clauses par la nouvelle clause $C_1 \vee C_2$.

   On a ainsi défini un algorithme en n étapes (où n est le nombre de littéraux), pour lequel on décide à chaque étape de la valeur d'un littéral avec une complexité m (où m est le nombre de clauses) : on peut donc résoudre SAT-2 en $O(nm)$.

(2PartEtAll) **Exercice 2.** *2 Partitions et ses variantes*

On définit les deux problèmes suivants :

SUBSET-SUM
Instance : Un ensemble fini S d'entiers positifs et un entier objectif t.
Question : Existe-t-il un sous-ensemble $S' \subseteq S$ tel que $\sum_{x \in S'} x = t$?

2-PARTITION
Instance : $S = \{a_1, \ldots, a_n\}$ un ensemble d'entier.
Question : Existe-t-il $I \subset S$ tel que $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

On suppose pour l'instant que ces deux problèmes sont NP-Complets. Montrer la NP-Complétude des problèmes suivants.

1. Etant donnés n entiers $a_1, a_2, \ldots, a_n$, peut-on trouver un sous-ensemble $I \subset [1..n]$ tel que $|\sum_{i \in I} a_i - \sum_{i \notin I} a_i| \leq 1$ ☞ $\in$ NP : trivial.

   Instance I1 de départ : $a_1, \ldots a_n$ de *2-partition*.
   Instance I2 créée : $(1664a_1, \ldots 1664a_n)$.
   Equivalence : S'il existe I avec $\sum_{i \in I} a_i = \sum_{i \in [1,n] \setminus I} a_i$, alors $|\sum_{i \in I} a_i - \sum_{i \in [1,n] \setminus I} a_i| = 0 \leq 1$

   Réciproquement, s'il existe I avec $|\sum_{i \in I} 1664a_i - \sum_{i \in [1,n] \setminus I} 1664a_i| \leq 1$, alors on a $|\sum_{i \in I} a_i - \sum_{i \in [1,n] \setminus I} a_i| \leq 1/1664$. Or on travaille dans les entiers, ainsi on aboutit à $\sum_{i \in I} a_i = \sum_{i \in [1,n] \setminus I} a_i$.

2. Soient $n = 2p$ un entier pair, et n entiers strictement positifs $a_1, a_2, \ldots, a_n$. Existe-t-il une partition de $\{1, 2, \ldots n\}$ en deux ensembles I et I' de même cardinal p et tels que $\sum_{i\in I} a_i = \sum_{i\in I'} a_i$? ☞ $\in$ NP : trivial.

Instance I1 de départ : $a_1, ...a_n$ de *2-partition*.

Instance I2 créée : $(a_1 + 1, ...a_n + 1, 1, ...1)$ de cardinal 2n.

Equivalence : Si $\sum_{i\in I} a_i = \sum_{i\in[1,n]\setminus I} a_i$ alors $\sum_{i\in I}(a_i + 1) + (n - |I|) = \sum_{i\in[1,n]\setminus I}(a_i + 1) + |I|$.

Réciproquement, une solution est de la forme $\sum_{i\in I}(a_i + 1) + (n - |I|) = \sum_{i\in[1,n]\setminus I}(a_i + 1) + |I|$. En effet il faut avoir n termes à droite et à gauche donc il faut rajouter un $(n - |I|)$ à gauche, et $|I| \leq n$ sinon au n'aurait jamais l'égalité, car tous les a_i sont positifs. Ainsi on a bien $\sum_{i\in I} a_i = \sum_{i\in[1,n]\setminus I} a_i$

3. Étant donnés n entiers $S\{a_1, a_2, \ldots, a_n\}$, peut-on trouver trois sous-ensembles I_1, I_2 et I_3 partitionnant $[1..n]$ et tels que $\sum_{i\in I_1} a_i = \sum_{i\in I_2} a_i = \sum_{i\in I_3} a_i$?

☞ 3-partition appartient à NP : étant donné un certificat on vérifie en temps linéaire que $\sum_{i\in I_1} a_i = \sum_{i\in I_2} a_i = \sum_{i\in I_3} a_i$.

On va faire une réduction à partir de... 2-Partition (étonnant non ?). Pour construire une instance de 3-Partition on prend donc une instance de 2-Partition avec n entiers $S = \{a_1, \ldots, a_n\}$, avec $P = \sum_{i=1}^{n} a_i$ et tels que $\forall i, a_i \leq P/2$ (sinon il n'y a pas de solution), et on rajoute un nouvel entier $a_{n+1} = P/2$. Si 2-Partition a une solution I avec l'ensemble initial S, alors il suffit de prendre comme solution pour 3-Partition : $\sum_{i\in I} a_i = \sum_{i\notin I} a_i = a_{n+1}$, soit $I_1 = I$, $I_2 = S\setminus I$ et $I_3 = a_{n+1}$.

Réciproquement, si 3-Partition a un solution, alors nécessairement on a I_1 ou I_2 ou I_3 qui est égal à $I' \cup a_{n+1}$, avec $\sum_{i\in I'} a_i = 0$, et il suffit de prendre les sous-ensembles restant comme solution de 2-Partition.

Donc 3-Partition est NP-complet.

4. S'agit-il de NP-complétude au sens faible ou au sens fort ?

☞ Au sens faible. Prendre l'algo du cours pour 2-Partition et l'adapter pour 3-Partition.

On va maintenant s'intéresser à la NP-Complétude des deux problèmes définis au début de l'exercice.

5. Montrer que 2-Partition est NP-complet. (On pourra supposer la NP-Complétude de **SUBSET-SUM**).

☞ 2-Partition est trivialement dans NP : on vérifie en temps linéaire qu'un certificat nous donne bien $\sum_{i\in I} a_i = \sum_{i\notin I} a_i$.

On fait une réduction à partir de SUBSET-SUM. Soit une instance de SUBSET-SUM : on a n entiers $S = \{a_1, \ldots, a_n\}$, et un objectif t. On construit une instance de 2-Partition de la façon suivante :

— si $t \leq \sum_{i=1}^{n} a_i < 2t$: on pose $a_{n+1} = 2t - \sum_{i=1}^{n} a_i$, et on construit un ensemble $S' = S \cup \{a_{n+1}\}$ pour 2-Partition. On a donc $\sum_{i=1}^{n+1} a_i = 2t$. Si on a une solution I_1 à SUBSET-SUM telle que $\sum_{i\in I_1} a_i = t$, alors on a une solution pour 2-Partition en prenant $\sum_{i\in I_1} a_i = \sum_{i\in S'\setminus I_1} a_i = t$. Inversement si 2-Partition a une solution, alors nécessairement $a_{n+1} \in I$ ou $a_{n+1} \in S'\setminus I$, et il suffit de prendre le sous ensemble de S' qui ne contient pas a_{n+1}.

— si $2t \leq \sum_{i=1}^{n} a_i$: on pose $a_{n+1} = \sum_{i=1}^{n} a_i - 2t$, et on construit l'ensemble S' comme précédemment. On a donc $\sum_{i=1}^{n+1} a_i = 2\sum_{i=1}^{n} a_i - 2t$. Comme précédemment on a l'équivalence des solutions avec $\sum_{i\in I_1} a_i = \sum_{i\in S'\setminus I_1} a_i = \sum_{i=1}^{n} a_i - t$, et donc nécessairement on a soit dans I_1 soit dans $S'\setminus I_1$: $\sum_{i=1}^{n} a_i - 2t + \sum_{i\in J} a_i$, et donc l'ensemble J permet d'avoir $\sum_{i\in J} a_i = t$ la solution à SUBSET-SUM.

Donc **2-PARTITION** est NP-complet.

6. Montrer que **SUBSET-SUM** est NP-complet.

Indication : vous pouvez par exemple effectuer une réduction à partir de 3-SAT. A partir d'un ensemble de clauses $C_0, ..., C_{m-1}$ sur les variables $x_0, ..., x_{n-1}$, considérer S l'ensemble des entiers $v_i = 10^{m+i} + \sum_{j=0}^{m-1} b_{ij}10^j$ et $v'_i = 10^{m+i} + \sum_{j=0}^{m-1} b'_{ij}10^j$, $0 \leq i \leq n-1$, où b_{ij} (resp. b'_{ij}) vaut 1 si le littéral x_i (resp. $\overline{x_i}$) apparaît dans C_j et 0 sinon, et des entiers $s_j = 10^j$ et $s'_j = 2.10^j$, $0 \leq j \leq m-1$. Trouver alors un entier objectif t tel qu'il existe un sous-ensemble $S' \subseteq S$ de somme t si et seulement si l'ensemble initial de clauses est satisfiable. Conclure. Quels autres entiers auraient aussi marché ?

☞ SUBSET-SUM est évidemment NP.

Pour montrer qu'il est NP-complet nous allons suivre l'indication et effectuer une réduction à partir de 3-SAT. Pour ce faire nous allons commencer en transformant les données de 3-SAT en chiffres.

Soit $C_0, \ldots, C_{m-1}$ les clauses et $x_0, \ldots, x_{n-1}$ les variables de 3-SAT. On construit v_i et v'_i de la manière suivante :

$$v_i = 10^{m+i} + \sum_{j=0}^{m-1}(b_{ij}10^j) \text{ et } v'_i = 10^{m+i} + \sum_{j=0}^{m-1}(b'_{ij}10^j),\ 1 \leq i \leq n-1 \text{ avec}$$

$$b_{ij} = \begin{cases} 1 & \text{si } x_i \text{ apparaît dans } C_j \\ 0 & \text{sinon} \end{cases} \quad \text{et } b'_{ij} = \begin{cases} 1 & \text{si } \overline{x_i} \text{ apparaît dans } C_j \\ 0 & \text{sinon} \end{cases}$$

Pour mieux comprendre ce qu'on fait, on peut imaginer qu'on crée des $(m+n)$-upplets dont les n premières cases il n'y à que des zéros sauf à un endroit i qui permet de savoir de quelle variable on parle, que ce soit x_i ou $\overline{x_i}$, et les m suivants permettent de savoir si la variable est dans les clauses correspondantes.

Entier	n variables x_{n-1} x_{n-2} ... x_0	m clauses C_{m-1} C_{m-2} ... C_0	Commentaires
v_0	$00\ldots1$	$\ldots1\ldots1\ldots$	1 : clauses dans lesquelles x_0 apparaît
v_{n-2}	$01\ldots0$	$\ldots1\ldots$	1 : clauses dans lesquelles x_{n-2} apparaît
v'_0	$0\ldots1$	$\ldots1\ldots1\ldots$	1 : clauses dans lesquelles $\overline{x_0}$ apparaît
v'_i	$\ldots010\ldots$	$\ldots1\ldots1\ldots$	1 : clauses dans lesquelles $\overline{x_i}$ apparaît
s_0	$00\ldots0$	$00\ldots1$	Uniquement 1 dans colonne C_0
s_i	$00\ldots0$	$\ldots1\ldots$	Uniquement 1 dans colonne C_i
s'_1	$00\ldots0$	$00\ldots02$	Uniquement 2 dans colonne C_0
s'_i	$00\ldots0$	$\ldots020\ldots$	Uniquement 2 dans colonne C_i
t	$11\ldots1$	$44\ldots4$	Entier objectif

Désormais on va sommer certains v_i et certains v'_i, en fait on somme n entiers, un par littéral, $T = \sum_{k=0}^{n-1}[x_k v_k + (1-x_k)v'_k]$. T est un nombre qui commence par n "1" et est suivit de m chiffres compris entre 0 et 3 (comme on est parti de 3-SAT il y a au plus 3 variables par clause et ces m chiffres correspondent exactement au nombre de littéraux qui mettent cette clause à vrai). Si on part d'une solution du problème 3-SAT, et qu'on prend v_i si x_i est vrai et v'_i si x_i est faux, on remarque que ces m chiffres sont compris entre 1 et 3.

A partir de là on se demande comment faire la différence entre les sous-ensembles donnant une somme qui contient un "0" de ceux donnant une somme sans "0" qui sont exactement les sous-ensembles correspondant à un certificat du problème 3-SAT par la bijection évidente $v_i \in$ S' $\leftrightarrow x_i$ est vrai et $v'_i \in$ S' $\leftrightarrow \overline{x_i}$ est vrai.

C'est à cela que servent les $s_j = 10^j$ et $s'j = 2.10^j$. En prenant $t = 1\ldots14\ldots4$ et en ne réfléchissant que sur les m chiffres de droite de T on voit que :
— à partir de 3 on peut avoir 4,$3+1$
— à partir de 2 on peut avoir 4,$2+2$
— à partir de 1 on peut avoir 4,$1+1+2$
— à partir de 0 on ne peut avoir 4.
(Cette réflection est la même qui si on considère les nombres comme des $n+m$-upplets)

Le problème 3-SAT est donc réduit en SUBSET-SUM avec $S = \{v_i | 1 \leq i \leq n-1\} \cup \{v'_i | 1 \leq i \leq n-1\} \cup \{s_j | 1 \leq j \leq m-1\} \cup \{s_j | 1 \leq j \leq m-1\}$ et $t = 1\ldots14\ldots4$

On a ainsi que, si on trouve une solution du problème 3-SAT, on a une solution du problème SUBSET-SUM en prenant la bijection précédente, et en complétant avec les $s_j = 10^j$ et $s'j = 2.10^j$.

Réciproquement, si on trouve un sous-ensemble S' de S tel que si $\sum_{x \in S'}(x) = t = 1\ldots14\ldots4$, en prenant $\tilde{\text{S}} = S' \cap (\{v_i | 1 \leq i \leq n-1\} \cup \{v'_i | 1 \leq i \leq n-1\})$ on a nécessairement $T = \sum_{x \in \tilde{S}}(x)$ qui est nombre commençant par n "1" et suivit de m chiffres compris entre 1 et 3 (ni 0 ni 4 : au plus 3 peut venir des s_i et s'_i, il y a nécessairement un 1 qui vient des v_i ou v'_i), ce qui montre que comme aucun "0" n'apparaît, les instances de 3-SAT peuvent toutes êtres mises à vrai, trouvé à l'aide de la bijection citée plus haut, donc à partir du certificat de ce problème SUBSET-SUM on retrouve un certificat du problème 3-SAT de départ.

La réduction précédente est polynomiale car on passe de n variables et m clauses à $2n+2m$ entiers de m chiffres, SUBSET-SUM est donc NP-complet.

☞ **Remarque :** On peut faire le même travail dans toutes les autres bases qui n'introduisent pas de problèmes dus aux phénomènes de retenue.

☞ **Remarque :** On peut aussi remplacer les "4" dans t par n'importe quel autre chiffre supérieur à 4, à condition de rajouter des "s" en nombre suffisant.

☞ **Remarque :** On peut aussi faire des $\tilde{s_i} = 10^{m+i}$ sans changer t, ça remplace les variables pour lesquelles on peut choisir n'importe que booléen.