ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
Section d'Informatique et de Systèmes de Communication

Corrigé de la série 12 20 Décembre 2010

1. Langages

a) Pour encoder la paire $(a, b) \in \mathbb{N}^2$, on a envie de juste juxtaposer les bits de a et ceux de b. Mais cette approche ne marchera pas parce qu'on ne sait pas où les chiffres de b commencent. Une autre idée: mettre autant de zéros que a a de chiffres, ensuite les chiffres de a et ensuite les chiffres de b:

$$(\underbrace{0,\ldots,0}_{|a| \text{ fois}}, \underbrace{*,\ldots,*}_{\text{Les chiffres de } a}, \underbrace{*,\ldots,*}_{\text{Les chiffres de } b}).$$

Étant donné un couple (a, b) encodé de cette façon, on pourra récupérer a et b en d'abord déterminant le nombre de chiffres de a, et ensuite la variable a elle-même, et puis b: il suffit de compter le nombre de 0 dans le préfixe du mot pour connaître le nombre de chiffres de a.

Ce codage aura une longueur de $|a| + |a| + |b| = 2|a| + |b|$. Ce n'est pas suffisant pour nous à cause du facteur 2 qui apparaît.

L'idée qui marchera sera de d'abord coder la longueur de a de manière plus efficace, ensuite de rajouter a et ensuite b à la fin du mot. Comme $|a| \ll a$, on peut coder la longueur de a en très peu de bits. En utilisant l'idée de tout à l'heure, on peut par exemple coder la longueur de a comme suit:

$$(\underbrace{0,\ldots,0}_{\lfloor \log(|a|) \rfloor \text{ fois}}, \underbrace{*,\ldots,*}_{\text{chiffres de } |a|})$$

En total, on aura donc

$$(\underbrace{0,\ldots,0}_{\lfloor \log(|a|) \rfloor \text{ fois}}, \underbrace{*,\ldots,*}_{\text{chiffres de } |a|}, \underbrace{*,\ldots,*}_{\text{chiffres de } a}, \underbrace{*,\ldots,*}_{\text{chiffres de } b}),$$

et ce codage utilisera $2\lfloor \log(\lfloor \log(a) \rfloor) \rfloor + \log(a) + \log(b)$ bits, ce qui est bien $|a| + |b| + O(\log(\log(a)))$, comme voulu. On a utilisé que $|x| = \lfloor \log(x) \rfloor$.

(Dans la donnée de l'exercice, $n = |a|$ et $m = |b|$.)

b) Le langage L consiste en tous les entiers impairs. En effet, un entier est impair si et seulement si son dernier chiffre (binaire) significatif est un 1. Donc le problème de décision correspondant est

L'entier x, est-il impair?

2. SAT

a) Non.

b) $x_1 = \text{FALSE}$, $x_2 = \text{FALSE}$, $x_3 = \text{TRUE}$.

c) Non.

d) Un input consiste en m clauses. La $j^{\text{éme}}$ clause contient k_j littéraux, elle aura donc comme taille $O(k_j)$: pour représenter $\lambda_1 \vee \ldots \vee \lambda_{k_j}$ il faut juste indiquer ce que valent chacun des k_j littéraux (les valeurs possibles sont $x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}$).

Donc la taille totale de l'input est

$$O\big(\sum_{j=1}^{m} k_j \log n\big) = O\big(m \cdot k \log n\big), \quad \text{où } k = \max_j k_j.$$

e) Il faut trouver une preuve vérifiable en temps polynomial. La preuve est une attribution satisfaisante, il est clair qu'on peut vérifier qu'une attribution est satisfaisante en temps polynomial: il faut faire $\leq m \cdot (k-1)$ opérations OR ($\vee$), et $m-1$ opérations AND ($\wedge$).

f) Puisque SAT $\in$ NP (question précédente), si P=NP on aura SAT $\in$ P. Donc il existe un algorithme qui résoud SAT en temps polynomial. Appelons cet algorithme $\mathcal{A}$. Donc on donne à $\mathcal{A}$ comme input une formule CNF F, et il retournera TRUE si F est satisfaisable et FALSE sinon (et son temps est polynomial en la taille de F). Nous allons utiliser $\mathcal{A}$ pour construire l'algorithme dont nous avons besoin.

On commence par donner comme input à $\mathcal{A}$ la formule F originale. Si $\mathcal{A}$ retourne FALSE nous savons que F n'est pas satisfaisable. Sinon nous savons qu'il existe une attribution satisfaisante, il nous faut donc la trouver. L'idée est d'utilise $\mathcal{A}$ pour verifier (en temps polynomial) s'il existe une attribution satisfaisante avec $x_1 = \text{TRUE}$ ou avec $x_1 = \text{FALSE}$. $\mathcal{A}$ retournera forcément TRUE dans au moins un des 2 cas (puisque nous savons que F est satisfaisable). Si par exemple $\mathcal{A}$ retourne TRUE quand $x_1 = \text{TRUE}$, nous posons $x_1 = \text{TRUE}$ et faisons de même avec $x_2, x_3, \ldots, x_n$.

Quel est l'input exact qu'il faut donner à $\mathcal{A}$ pour savoir s'il existe une attribution satisfaisante avec $x_1 = \text{TRUE}$? Il faut donner une version modifiée de F: Toutes les clauses qui contiennent x_1 peuvent être enlevées (puisqu'elles sont toutes vraies quelles que soitent les valeurs de $x_2, \ldots, x_n$), et nous pouvons enlever $\overline{x_1}$ de toutes les clauses qui le contiennent, puisque

$$\lambda_1 \vee \ldots \vee \lambda_\ell \vee \overline{x_1} \quad = \quad \lambda_1 \vee \ldots \vee \lambda_\ell \quad \text{si } \overline{x_1} = \text{FALSE}.$$

Ainsi pour obtenir cette version modifiée F' de F on enlève certaines clauses, et on enlève certains litteraux dans d'autres clauses, la taille de F' sera donc $\leq$ à la taille de F.

$$\begin{array}{lcl} \#\text{clauses dans } F' & \leq & \#\text{clauses dans } F \\ \text{taille des clauses dans } F' & \leq & \text{taille des clauses dans } F \end{array}$$

3. Révision: Knapsack 0/1

a) Nous completons la matrice selon l'algorithme du cours, obtenant ainsi:

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&3&3&3&3&3&3&3&3&3&3&3&3&3&3&3&3&3&3&3\\
0&0&3&3&7&7&10&10&10&10&10&10&10&10&10&10&10&10&10&10&10\\
0&0&3&3&7&7&12&12&15&15&19&19&22&22&22&22&22&22&22&22&22\\
0&0&3&3&7&7&12&12&15&15&19&19&22&22&23&23&26&26&30&30&33\\
0&0&3&3&7&7&12&12&15&15&19&19&22&22&23&23&26&26&30&30&33\\
0&0&3&3&7&7&12&12&15&15&19&19&22&22&25&25&28&28&30&30&33\\
0&0&3&6&7&9&12&13&15&18&19&21&22&25&25&28&28&31&31&34&34
\end{pmatrix}.$$

Ainsi la valeur maximale atteignable est $C_{7,20} = 34$ (la valeur en bas à droite de la matrice).

Pour trouver un ensemble d'objets satisfaisants, il nous faut regarder comment cette valeur a été obtenue lors de la construction de cette matrice. De manière générale nous avons

$$C_{i,j} = \max\big(C_{i-1,j},\ C_{i-1,j-w_i} + v_i\big). \tag{1}$$

Si $C_{i,j} = C_{i-1,j}$ alors l'objet i ne sera pas utilisé, alors que si $C_{i,j} = C_{i-1,j-w_i} + v_i$ il le sera (et si les deux valeurs sont égales il y a deux solutions possibles).

Ainsi puisque

$$34 = 28 + 6 = C_{6,20-4} + 6 = C_{6,20-w_6} + v_6,$$

nous déduisons que l'objet A_7 sera bien utilisé. Par le même raisonnement nous voyons (voir dessin ci-dessous) que A_6, A_3, A_2 et A_1 seront utilisés alors que A_5 et A_4 ne le seront pas. La solution optimale est donc

$$\{A_1, A_2, A_3, A_6, A_7\}.$$

$$\begin{pmatrix}
0&\boxed{0}&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&3&\boxed{3}&3&3&3&3&3&3&3&3&3&3&3&3&3&3&3&3&3\\
0&0&3&3&7&7&10&\boxed{10}&10&10&10&10&10&10&10&10&10&10&10&10&10\\
0&0&3&3&7&7&12&12&15&15&19&19&22&\boxed{22}&22&22&22&22&22&22&22\\
0&0&3&3&7&7&12&12&15&15&19&19&22&\boxed{22}&23&23&26&26&30&30&33\\
0&0&3&3&7&7&12&12&15&15&19&19&22&\boxed{22}&23&23&26&26&30&30&33\\
0&0&3&3&7&7&12&12&15&15&19&19&22&22&25&25&28&\boxed{28}&30&30&33\\
0&0&3&6&7&9&12&13&15&18&19&21&22&25&25&28&28&31&31&34&\boxed{34}
\end{pmatrix}.$$

b) En appliquant (1) nous obtenons

$$\begin{aligned}
C_{7,20} &= \max\big(C_{6,20},\ C_{6,20-y} + x\big)\\
&= \max\big(C_{6,20},\ C_{6,13} + x\big)\\
&= \max\big(33, 22 + x\big).
\end{aligned}$$

c) Il faut que

$$\begin{aligned} C_{6,20} = C_{6,20-y} + 11 \quad &\iff \quad 33 = C_{6,20-y} + 11 \\ &\iff \quad C_{6,20-y} = 22. \end{aligned}$$

Nous voyons que $C_{6,12} = C_{6,13} = 22$, donc en choisissant $y = 7$ ou $y = 8$ il y a aura bien deux sous-sensembles différents qui donnent la valeur optimale (qui sera 33).

Une des solutions utilisera l'objet A_7, et l'autre ne l'utilisera pas.