ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Section d'Informatique et de Systèmes de Communication

Corrigé de la série 12 14 Decembre 2007

1. La taille de l'input

a) $\log_2(n)$.

b) n^2.

c) $\lceil \log_2(1000) \rceil n = 10n$. (On peut aussi faire légèrment mieux, en représentant les entiers $a_1, \ldots, a_n$ comme un grand entier $\sum_{i=1}^{n} 1000^{i-1} a_i$, ce qui donne $\lceil \log_2(1000) n \rceil$.)

d) $n^2 \cdot \log_2(201+1) = O(n^2)$.
Explication : On a n^2 valeurs dans la matrice d'adjacence, pour chaque valeur on a besoin de $\log_2(201+1)$, bits : 201 pour les valeurs possibles de -100 à 100 et une valeur de plus pour codifier le fait qu'il n'y a pas d'arête entre les deux sommets.

2. 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 :

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

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 ?

3. Complexité

a) Puisque $A \in P$ il existe un algorithme $\mathcal{S}_A(x)$, avec comme ensemble d'output $\{\text{TRUE}, \text{FALSE}\}$, un temps de parcours polynomial en $|x|$, et la propriété que

$$\mathcal{S}_A(x) = \text{TRUE} \iff x \in A.$$

Puisque $B \in P$, il existe un algorithme $\mathcal{S}_B$ qui fait de même pour B. (Nous rappelons que $|x|$ dénote la longueur du mot x).
En définissant un algorithme $\mathcal{S}_1$ comme suit :

$$\mathcal{S}_1(x) = \mathcal{S}_A(x) \wedge \mathcal{S}_B(x), \tag{1}$$

nous voyons que

$$\begin{aligned} \mathcal{S}_1(x) = \text{TRUE} &\iff \big(\mathcal{S}_A(x) = \text{TRUE}\big) \wedge \big(\mathcal{S}_B(x) = \text{TRUE}\big) \\ &\iff \big(x \in A\big) \wedge \big(x \in B\big) \\ &\iff x \in A \cap B. \end{aligned}$$

De plus, son temps de parcours est égal à la somme des temps de parcours de $\mathcal{S}_A$ et $\mathcal{S}_B$, il est donc polynomial en $|x|$. Ainsi d'après la définition de la classe P, nous avons $A \cap B \in P$

Remarque : Nous avons dit plus haut que (1) définissait un algorithme. En fait cette ligne définit plutôt la fonction implémentée par l'algorithme. Il est cependant évident comment implémenter l'algorithme $\mathcal{S}_1$ à partir d'implémentations de $\mathcal{S}_A$ et $\mathcal{S}_B$: On execute $\mathcal{S}_A(x)$, puis $\mathcal{S}_B(x)$ et on fait le AND des deux résultats.

b) Nous définissons un algorithme $\mathcal{S}_2$ comme suit :

$$\mathcal{S}_2(x) = \mathcal{S}_A(x) \vee \mathcal{S}_B(x),$$

nous avons

$$\begin{aligned} \mathcal{S}_2(x) = \text{TRUE} &\iff \big(\mathcal{S}_A(x) = \text{TRUE}\big) \vee \big(\mathcal{S}_B(x) = \text{TRUE}\big) \\ &\iff \big(x \in A\big) \vee \big(x \in B\big) \\ &\iff x \in A \cup B. \end{aligned}$$

et comme ci-dessus, son temps de parcours est polynomials en $|x|$. Donc d'après la définition de la classe P, nous avons $A \cup B \in P$.

c) Nous voulons construire un algorithme $\mathcal{S}_3$ qui décide si un input $x \in (A : B)$, c'est à dire s'il existe $a \in A$ et $b \in B$ avec $x = a : b$. Soit $\ell = |x|$. La "coupure" entre a et b pourrait se trouver n'importe où dans x, il faut donc examiner toutes les possibilités :

$$\begin{array}{ll} a = x_1, & b = x_2 \dots x_\ell \\ a = x_1 x_2, & b = x_3 \dots x_\ell \\ a = x_1 x_2 x_3, & b = x_4 \dots x_\ell \\ \vdots & \vdots \\ a = x_1 \dots x_{\ell-1}, & b = x_\ell \end{array}$$

Nous devons donc executer $\mathcal{S}_A$ et $\mathcal{S}_B$ $\ell - 1$ fois, sur des inputs de taille $\leq |x|$. Le temps de parcours de $\mathcal{S}_3(x)$ est donc polynomial en $\ell = |x|$. Plus formellement, $\mathcal{S}_3(x)$ est defini comme suit :

Call : $\mathcal{S}_3(x)$
1: $\ell \leftarrow |x|$
2: **for** $i = 1, \dots, \ell - 1$ **do**
3: **if** $\mathcal{S}_A(x_1 \dots x_i) = \text{TRUE} \wedge \mathcal{S}_B(x_{i+1} \dots x_\ell) = \text{TRUE}$ **then**
4: **return** TRUE
5: **return** FALSE

d) Puisque $A \leq_P B$, il existe une fonction f qui satisfait
- $f(x)$ peut être calculé avec un temps de parcours polynomial en $|x|$.
- $x \in A \iff f(x) \in B$.

Puisque $x \in A \iff f(x) \in B$ nous avons aussi $x \notin A \iff f(x) \notin B$ (si deux affirmation sont équivalentes, leurs négations le sont aussi), et donc $x \in \overline{A} \iff f(x) \in \overline{B}$.
Ainsi notre fonction f vérifie :
- $f(x)$ peut être calculé avec un temps de parcours polynomial en $|x|$.
- $x \in \overline{A} \iff f(x) \in \overline{B}$.

Donc par définition nous avons $\overline{A} \leq_P \overline{B}$ (la réduction se fait par la même fonction f).

4. Problème de réduction

Nous montrerons $A \leq_{\mathsf{P}} B$. Par symétrie l'autre inégalité sera alors aussi vraie. Soient u et v deux mots d'alphabet Σ tels que $u \in B$ et $v \notin B$. (Remarquons que u et v existent puisque les inclusions $\emptyset \subset B \subset \Sigma^*$ sont strictes.) On peut alors définit la réduction de A à B comme suit :

$$f(x) := \begin{cases} u & \text{si } x \in A \\ v & \text{sinon.} \end{cases}$$

Comme $A \in \mathsf{P}$, la fonction f est calculable en temps polynomial. Par le choix de u et v nous avons aussi que $f(x) \in B$ si et seulement si $x \in A$. Donc, $A \leq_{\mathsf{P}} B$.

5. Réduction polynomiale

a) Supposons que $G = (V, E)$, et soit u le sommet ajouté pour obtenir $f_1(G)$. Ainsi $f_1(G) = (V_1, E_1)$ où $V_1 = V \cup \{u\}$, et $\forall v \in V_1 : \ (u, v_i), (v_i, u) \in E_1$.

$\Rightarrow$. Supposons que G contient une clique $S \subseteq V$ de taille $\geq k$. Puisque u est connecté à tous les autres sommet,

$$S \cup \{u\} \subseteq V_1$$

forme une clique de taille $|S| + 1 \geq k + 1$.

$\Leftarrow$. Supposons que $f_1(G)$ contient une clique $S \subseteq V_1$ de taille $\geq k + 1$. Nous distinguons 2 cas :

• **Cas 1 :** $u \notin S$. Dans ce cas nous avons $S \subseteq V$, et donc cette clique de taille $\geq k + 1$ est aussi dans G. G contient donc bien une clique de taille $\geq k$.

• **Cas 2 :** $u \in S$. Si nous retirons u de la clique nous obtenons une clique de taille $|S| - 1$, donc de taille $\geq k$. puisque tous les sommets de V_1 sauf u sont aussi dans V, $S - \{u\} \subseteq V$ forme une clique de taille $\geq k$ dans le graphe G.

b) Nous ajoutons a sommets $u_1, \ldots u_a$, qui sont connectés à tous les autres sommets (et aussi connecté entre eux). Soit $U = \{u_1, \ldots, u_a\}$. Formellement si $G = (V, E)$ alors $f_a(G) = (V_a, E_a)$ où

$$\begin{array}{rcl} V_a & = & V \cup U \\ E_a & = & E \ \cup \ (V_a \times U) \ \cup \ (U \times V_a) \end{array}$$

Nous pouvons montrer de manière identique au point a) que G contient une clique de taille $\geq k$ si et seulement si $f_a(G)$ contient une clique de taille $\geq k + a$.

c) Nous voyons d'abord que

$$n_a = |V_a| = |V| + a = m + a.$$

Nous savons que G contient une clique de taille $\geq k$ si et seulement si $f_a(G)$ contient une clique de taille $\geq k + a$. Nous voulons donc choisir a tel que $n_a/2 = k + a$.

$$\begin{array}{rcl} \frac{n_a}{2} = k + a & \iff & \frac{m+a}{2} = k + a \\ & \iff & m + a = 2k + 2a \\ & \iff & a = m - 2k \end{array}$$

Puisque $k \leq m/2$, nous voyons que cette valeur est ≥ 0.

d) Il nous reste à couvrir le cas où $k > m/2$. Nous construisons le graphe $h(G)$ en ajoutant à V $2k - m$ sommets sans aucune arête. Formellement, $h(G) = (V', E')$ avec

$$\begin{array}{rcl} V' & = & V \cup \{w_1, \ldots, w_{2k-m}\} \\ E' & = & E. \end{array}$$

Puisque les nouveaux sommets n'on par d'arêtes, il est clair que G contient une clique de taille $\geq k$ si et seulement si $h(G)$ contient une clique de taille $\geq k$.

Ainsi la réduction est la suivante :

$$f(G, k) = \begin{cases} f_{m-2k}(G) & \text{si } \ k \leq \frac{m}{2} \\ h(G) & \text{sinon} \end{cases}$$