

## 1 Problèmes de factorisation

On considère les trois problèmes suivants :

- `FINDSMALLESTFACTOR` Étant donné  $N \in \mathbb{N}^*$  renvoie le plus petit  $k \in \mathbb{N}$  avec  $k \geq 2$  tel que  $k$  divise  $N$ .
- `FINDGREATESTFACTOR` Étant donné  $N \in \mathbb{N}^*$  renvoie le plus grand  $k \in \mathbb{N}$  avec  $k < N$  tel que  $k$  divise  $N$ .
- `HASFACTOR` Étant donné  $(N, M) \in \mathbb{N}^{*2}$  décide s'il existe un facteur non trivial de  $N$  plus petit que  $M$ .

**Exercice 1.** Montrer que si l'on sait résoudre un de ces problèmes en temps polynomial, alors on sait résoudre les trois. Attention le temps polynomial se réfère à la taille de l'entrée qui est logarithmique en les nombres représentés.

**Solution 1.** Si l'on sait résoudre l'un des deux premiers alors on sait factoriser  $N$  et donc répondre aux deux autres problèmes.

Pour le dernier, on va montrer qu'on peut s'en servir pour résoudre le premier. Il s'agit de travailler par dichotomie. On commence par  $k_1 = 1$  et  $k_2 = N$  ensuite, tant que  $k_1 + 1 < k_2$  on choisit  $M = \lfloor (k_1 + k_2)/2 \rfloor$  et on regarde s'il existe un facteur de  $N$  plus petit que  $M$ , si c'est le cas on pose  $k_2 = M$  sinon on pose  $k_1 = M$ . Une fois que notre algorithme s'arrête, on a  $k_1 + 1 = k_2$  et donc  $k_1 + 1$  est le plus petit nombre qui est un facteur non trivial de  $N$ . À chaque étape  $k_2 - k_1$  perd un bit on fait donc un nombre linéaire d'appels à `HASFACTOR`.

**Exercice 2.** Montrer que `HASFACTOR` est dans  $\mathcal{NP}$ . Vous pouvez supposer que tester la primalité d'un nombre est dans  $\mathcal{P}$ .

**Solution 2.** Ce problème peut être traité en regardant le certificat  $(p_1, e_1) \dots, (p_k, e_k)$  et en vérifiant que chacun des  $p_i$  sont premiers, que  $N = \prod_i p_i^{e_i}$  et que le plus petit  $p_i$  est bien plus petit que  $M$ .

**Exercice 3.** Montrer que `HASFACTOR` est dans  $\text{co-}\mathcal{NP}$ .

**Solution 3.** De la même manière que pour  $\mathcal{NP}$ , on regarde les mêmes certificats, comme ils donnent une décomposition parfaite, il suffit de vérifier que le certificat est bien tel que chaque  $p_i$  soit plus grand que  $M$ .

## 2 Propriétés de clôture

**Exercice 4.** Montrer que les langages décidables en temps polynomial sont stables par intersection, union, complémentation et étoile (i.e. le langage  $L^*$ ).

## 3 Fonctions à sens unique

Supposons que :

- on a une bijection  $f$  des entiers sur  $n$  bits vers les entiers sur  $n$  bits, pour tout  $n$  (i.e., sur une entrée  $x$  de  $n$  bits,  $f(x)$  est un entier sur  $n$  bits tel que  $f(x) = f(y) \Rightarrow x = y$ ).
- la fonction  $f$  se calcule en temps polynomial.
- la fonction inverse de  $f$  ne peut pas se calculer en temps polynomial. (On dit que c'est une fonction à sens unique.)

**Exercice 5.** Montrer que si une telle bijection existe, alors  $\mathcal{P} \neq \mathcal{NP}$ .

*Suggestion : Montrer que le langage  $L = \{(x, f(y)) : x \leq y\}$  appartient à  $\mathcal{NP} \setminus \mathcal{P}$ .*

**Solution 5.**  $L$  est  $\mathcal{NP}$  en considérant le certificat étant  $x, y$ .

Si  $L$  est dans  $\mathcal{P}$  alors étant donné  $x$  tel qu'il existe  $y$  avec  $x = f(y)$  on cherche par dichotomie sur  $z$  le plus petit  $z$  tel que  $(z, f(y))$  est dans le langage  $L$ . On trouve alors la fonction inverse de  $f$  en temps polynomial ce qui était supposé impossible.

**Exercice 6.** Montrer de plus, que si elle existe, alors  $\mathcal{NP} \cap \text{co-}\mathcal{NP} \neq \mathcal{P}$ .

**Solution 6.** On fonctionne de la même manière, il suffit d'exhiber un  $y$  qui nous donne le contre-exemple. Et par dichotomie sur le langage complémentaire de  $L$  on peut aussi inverser  $f$  en un nombre linéaire d'appels à  $\bar{L}$ .

## 4 Quines

Pour deux MT  $A$  et  $B$  on note  $A \cdot B$  une MT qui exécute  $B$  après avoir exécuté  $A$ .

Pour chaque mot  $w \in \Sigma^*$ , soit  $P(w)$  un MT sur  $\Sigma$  qui écrit le mot  $w$  sur le ruban.

**Exercice 7.** Expliquer pourquoi la fonction  $q : \mathbb{N} \rightarrow \mathbb{N}$  est récursive (i.e. calculable).

$$q(n) = \begin{cases} \langle P(w) \rangle & \text{s'il existe } w \in \Sigma^* \text{ tel que } n = \langle w \rangle \\ \perp & \text{sinon} \end{cases}$$

**Solution 7.**  $q(n)$  peut être calculé en calculant le  $w$  tel que  $n = \langle w \rangle$  s'il existe puis affiche  $\langle P(w) \rangle$ . Le calcul de  $w$  est récursif par la forme du codage tandis que  $P(w)$  est aussi calculable car il suffit de mettre  $|w|$  états, l'état  $i$  affichant  $w_i$  puis calculer le code de  $\langle P(w) \rangle$ .

**Exercice 8.** Expliquer pourquoi la fonction  $s_2(m, n) : \mathbb{N}^2 \rightarrow \mathbb{N}$  est récursive.

$$s_2(m, n) = \begin{cases} \langle A \cdot B \rangle & \text{s'il existe } A, B \text{ telles que } \langle A \rangle = m \wedge \langle B \rangle = n \\ \perp & \text{sinon} \end{cases}$$

**Solution 8.** Étant les codes  $\langle A \rangle$  et  $\langle B \rangle$  on retrouve  $A$  et  $B$  puis on concatène les deux calculs.

**Exercice 9.** En déduire qu'il existe une MT  $M$  qui termine en affichant  $\langle M \rangle$  sur la bande.

**Solution 9.**

Soit  $s$  qui calcule  $x \rightarrow s_2(x, q(\langle x \rangle))$  on regarde  $M$  la MT qui calcule  $s \cdot P(\langle s \rangle)$  donc  $\langle M \rangle = \langle s \cdot P(\langle s \rangle) \rangle$

Une exécution de  $M$  donne sur le ruban :  $s \cdot P(\langle s \rangle) \rightarrow s(P(\langle s \rangle)) \rightarrow s(\langle s \rangle) = s_2(\langle s \rangle, q(\langle \langle s \rangle \rangle)) \rightarrow \langle s \cdot P(\langle s \rangle) \rangle = \langle M \rangle$ .