1 Un problème dans BPP

1.1 Définitions

Définition 1. *On rappelle que la classe BPP peut se définir comme la classe des langages L tels qu'il existe un polynôme P et un langage A reconnu en temps polynomial par une machine de Turing tels que :*

$$x \in L \Rightarrow \mathop{\mathbb{P}}_{\varepsilon \in \{0,1\}^{p(|x|)}}\Big((x,\varepsilon) \in A\Big) \geq \frac{2}{3} \qquad x \notin L \Rightarrow \mathop{\mathbb{P}}_{\varepsilon \in \{0,1\}^{p(|x|)}}\Big((x,\varepsilon) \in A\Big) \leq \frac{1}{3}$$

Définition 2. *On rappelle que le* degré d'un monôme $X_1^{d_1} \dots X_n^{d_n}$ *est* $\sum_j d_j$ *et que le* degré d'un polynôme *est le maximum des degrés des monômes le composant. Par exemple $XXXXY + XYZ + Z + ZY$ est de degré 5 (car $XXXXY$ est de degré 5).*

1.2 Polynômes en forme générale

Définition 3. *Un polynôme peut toujours s'écrire sous la forme de sommes de monôme. Un polynôme est en* forme générale *quand il est donné comme variable simple (i.e. X_i), produit, somme ou différence de polynômes en forme générale. Par exemple* $((X_1 + X_2) \times (X_1 - X_2) + X_1) \times X_1$.

On ignore ici volontairement les détails de l'encodage mais on suppose qu'il est non ambigu et que la taille d'un encodage est $O(ln(n))$ pour décrire la n-ième variable, et $O(1)+n_1+n_2$ pour décrire la somme (ou le produit) de deux polynômes de taille n_1 et n_2.

Question 1. Montrer que la fonction qui transforme un polynôme donné en forme générale en un polynôme sous forme de monômes ne peut pas être calculée en temps polynomial.

Question 2. Quelle est la complexité de la fonction qui teste l'égalité de deux polynômes en forme générale en utilisant l'expansion en somme de monômes ?

1.3 Lemme de Schwartz-Zippel

Lemme 1. *Soit $P \in \mathbb{Z}[X_1, \dots, X_n]$ non nul de degré d et soit S une partie finie de $\mathbb{Z}$. Si $x_1, \dots, x_n$ sont choisis uniformément dans S alors* $\mathbb{P}\Big(P(x_1, \dots, x_n) = 0\Big) \leq \dfrac{d}{|S|}$

Question 3. Montrer ou admettre ce lemme.

Question 4. En déduire un algorithme BPP pour tester l'égalité de deux polynômes en forme générale.

2 Révision : Théorème de Rice

Définition 4. *Un état* inutile *d'une machine de Turing est un état qui n'est jamais visité pendant un calcul.*

Question 5. Montrer que le problème de décider si une machine de Turing a des états inutiles est indécidable.

Question 6. Expliquer pourquoi on ne peut pas appliquer le théorème de Rice pour résoudre ce problème.

3 Révision : Castor affairé

Le problème des « busy beavers » (en français : « castors affairés ») a été introduit par Radó, dans le but de définir « simplement » une fonction non calculable. Le modèle de machines de Turing considéré est le suivant : on suppose les machines déterministes, possédant un ruban bi-infini, un alphabet réduit

à $\{0, 1\}$ (on ne distingue pas les 0 du symbole blanc). On suppose de plus que les machines possèdent un unique état final, duquel aucune transition ne sort, et qui n'est pas compté parmi les états de la machine. Enfin, on considérera toujours un ruban initialement complètement blanc et l'on suppose qu'il existe un état d'arrêt duquel ne part aucune transition.
La fonction du castor affairé, notée $\Sigma(n)$, est définie comme le nombre maximum de 1 écrits sur la bande (pas nécessairement consécutifs) après qu'une machine à $n + 1$ états (n états d'opération et un état d'arrêt) arrive dans l'état d'arrêt (la machine doit donc s'arrêter).

Question 7. Justifier l'existence de $\Sigma(n)$ pour tout $n \in \mathbb{N}$.

Question 8. Que valent $\Sigma(0)$, $\Sigma(1)$, $\Sigma(2)$ (ou plus simplement montrer que $\Sigma(2) \geq 4$) ?

Question 9. Montrer que la fonction Σ est strictement croissante.

Le modèle de fonction calculable considéré est le suivant : f est calculable si et seulement s'il existe une machine de Turing qui, sur un ruban contenant initialement n symboles 1 consécutifs à droite du symbole blanc de départ, s'arrête après un nombre fini de déplacements en produisant un bloc de $f(n)$ symboles 1 consécutifs.

Question 10. Montrer que la fonction Σ du castor affairé croit strictement plus vite que toute fonction calculable f (i.e. $f(n) = o(\Sigma(n))$).