

# Algorithmes et techniques probabilistes PC 6.

## Correction

### 1 Lemme de Schwartz-Zippel

Soit  $\mathbb{F}$  un corps, et  $S \subset \mathbb{F}$  un sous-ensemble arbitraire. Soit  $p(\mathbf{x})$  un polynôme non nul des  $n$  variables  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  de degré total  $d$  à coefficients dans  $\mathbb{F}$ . Montrer que l'équation  $p(\mathbf{x}) = 0$  possède au plus  $d|S|^{n-1}$  solutions dans  $S^n$ .

### 2 Application 1 : Isomorphisme d'arbres

Un arbre orienté est dit *ordonné* si l'on a fixé un ordre sur les fils de chacun des sommets.

Proposer un algorithme probabiliste pour déterminer si deux arbres orientés non-ordonnés de hauteur  $h$  et de taille  $n$  sont isomorphes.

On associera pour cela à chaque tel arbre un polynôme et on ramènera le problème à celui de l'égalité de polynômes.

Correction: *On associe à chaque sommet  $v$  un polynôme  $f_v$  en les variables  $x_0, x_1, \dots, x_h$  inductivement comme suit. Pour chaque feuille, on pose  $f_v = x_0$ . Pour chaque sommet interne  $v$  de hauteur  $k$  avec les fils  $v_1, \dots, v_m$ , on pose*

$$f_v = (x_k - f_{v_1})(x_k - f_{v_2}) \cdots (x_k - f_{v_m}).$$

*Le degré de  $f_v$  est égal au nombre de feuilles du sous-arbre enraciné en  $v$ . En utilisant l'unicité de la factorisation d'un polynôme, on montre que deux arbres sont isomorphes si et seulement si leurs polynômes sont égaux.*

*L'algorithme consiste à tester s'ils sont égaux en les évaluant sur des entrées arbitraires.*

### 3 Application 2 : Isomorphisme de programmes

Un *programme branchant* est un graphe acyclique orienté où chaque sommet est étiqueté par une variable, à l'exception de deux *sommets de sortie*, étiquetés par  $\mathbf{0}$  ou  $\mathbf{1}$ . Les sommets étiquetés par une variable sont appelés des sommets de *requêtes*. Chaque tel sommet possède deux arcs sortants, l'un étiqueté par  $\mathbf{0}$ ,

l'autre par **1**. Les sommets de sortie n'ont pas d'arc sortant. Un des sommets est appelé le *sommet d'entrée*.

Un programme branchant définit une fonction booléenne de la façon suivante : on prend une affectation des variables apparaissant sur ses sommets de requêtes, et on suit le chemin qui correspond à prendre l'arc correspondant en chaque sommet de requête selon la valeur de la variable en ce sommet, jusqu'à atteindre un sommet de sortie. On lit alors en le sommet de sortie la valeur de la fonction booléenne.

Un programme branchant est à *lecture unique* si chaque variable de requête apparaît au plus une fois sur chaque sommet orienté entre le sommet d'entrée et un sommet de sortie.

On veut produire un algorithme pour déterminer si deux programmes branchants à lecture unique sont équivalents, avec une probabilité d'erreur  $\leq 1/3$ .

1. Expliquer pourquoi l'idée de prendre aléatoirement des valeurs aux variables  $x_1, \dots, x_m$  qui apparaissent dans les deux programmes et évaluer ces programmes sur ces valeurs ne suffit pas.
2. Proposer une technique basée sur l'égalité de polynômes.

Correction: *correction à compléter.*

## 4 Coupures de grandes tailles

1. Etant donné un graphe orienté  $G$  avec  $m$  arêtes, montrer qu'il existe une partition des sommets  $V$  en deux ensembles disjoints  $A$  et  $B$  telle que au moins  $m/2$  arêtes connectent un sommet de  $A$  à un sommet de  $B$ . Autrement dit, qu'il existe une coupure de taille au moins  $m/2$ .

On pourra considérer les ensembles  $A$  et  $B$  obtenus en affectant aléatoirement et uniformément chaque sommet à l'un des deux, et on s'intéressera à l'espérance de la variable aléatoire qui dénote la valeur de la coupure correspondante aux ensembles  $A$  et  $B$ .

Correction: *On construit les ensembles  $A$  et  $B$  en affectant aléatoirement et uniformément chaque sommet à l'un des deux. Soit  $e_1, e_2, \dots, e_m$  une énumération des arêtes de  $G$ . Pour  $i = 1, 2, \dots, m$ , on pose  $X_i$  qui vaut 1 si l'arête  $i$  connecte  $A$  à  $B$ , et 0 sinon. La probabilité qu'une arête  $e_i$  connecte  $A$  à  $B$  est  $1/2$ , et donc,  $E[X_i] = 1/2$ . Soit  $C(A, B)$  la variable aléatoire qui dénote la valeur de la coupure correspondante aux ensembles  $A$  et  $B$ . Alors*

$$E[C(A, B)] = E\left[\sum_{i=1}^m X_i\right] = \sum_{i=1}^m E[X_i] = m/2.$$

*Puisque la moyenne vaut  $m/2$ , il doit y avoir une partition  $A$  et  $B$  avec au moins  $m/2$  arêtes connectant  $A$  à  $B$ .*

2. Proposer un algorithme de Las Vegas qui produit une coupure de taille au moins  $m/2$ .

Correction: Soit  $p = \Pr(C(A, B) \geq m/2)$ . On a  $C(A, B) \leq m$ . Donc

$$\begin{aligned} m/2 = E[C(A, B)] &= \sum_{i \leq m/2-1} i \Pr(C(A, B) = i) + \sum_{i \geq m/2} i \Pr(C(A, B) = i) \\ &\leq (1-p)(m/2-1) + pm \end{aligned}$$

ce qui implique que  $p \geq \frac{1}{m/2+1}$ . En tirant ainsi au hasard  $A$  et  $B$ , on tombe donc avec probabilité au moins cette quantité sur une coupure de taille  $\geq m/2$ . Tester si  $A$  et  $B$  donne bien une coupure sur une coupure de taille  $\geq m/2$  se fait en temps polynomial. On peut répéter l'expérience jusqu'à tomber sur une coupure de taille  $\geq m/2$ . le nombre moyen de répétitions est donc de l'ordre de  $m/2 + 1$ .

## 5 Algorithme Quicksort

Montrer que le tri Quicksort fonctionne en temps  $\mathcal{O}(n \log n)$  avec grande probabilité.

On pourra borner le nombre de pivots auquel chaque élément est comparé.