# TD 11 : Structures de données aléatoires

## 1 Tables de hachage

Un dictionnaire est une structure de données dont la fréquence de modifications est très faible devant la fréquence des consultations. Par exemple, les correcteurs orthographiques des traitements de texte ont recours à un dictionnaire.

Dans la suite, on s'intéresse á des dictionnaires implementés par des tables de hachage. Dans la mesure où les données du dictionnaire sont pratiquement inchangées durant une longue période, il est intéressant de choisir la fonction de hachage parmi un ensemble de fonctions de telle sorte qu'il n'y a pas de collisions.

Supposons par la suite que les identifiants sont $n$ valeurs numériques comprises entre 0 et $p-1$, avec $p$ premier. Nous désirons stocker ce dictionnaire dans une table de hachage à $m$ entrées, avec $m \leq p$.

La fonction de hachage que nous recherchons se trouve parmi l'ensemble $H = \{h_{a,b} \mid 0 < a < p \wedge 0 \leq b < p\}$ où

$$h_{a,b}(x) = ((ax + b \mod p) \mod m) + 1$$

On identifiera dans la suite la fonction $h_{a,b}$ à la paire $(a, b)$.

**Exercice 1.**

1. *Soient $i$ et $j$ deux identifiants différents et $h_{a,b} \in H$. Montrer que $ai + b \not\equiv aj + b \mod p$.*
2. *Soient $i$ et $j$ deux identifiants différents. Montrer que*

   $$\forall 0 \leq u \neq v < p,\ \exists! h_{a,b} \in H \text{ t.q. } (ai+b \equiv u \mod p) \wedge (aj+b \equiv v \mod p)$$

3. *On suppose toujours $i \neq j$. Montrer que $|\{h_{a,b} \mid h_{a,b}(i) = h_{a,b}(j)\}| \leq \frac{p(p-1)}{m}$*
4. *En déduire que $q = \sum_{i<i'} |\{(a,b) \mid h_{a,b}(i) = h_{a,b}(i')\}|$ le nombre total des collisions vérifie $q \leq \frac{n(n-1)}{2} \frac{p(p-1)}{m}$.*
5. *Soit $m = n^2$. Montrer que $|H'| < \frac{1}{2}|H|$, où $H'$ est le sous-ensemble de fonctions qui produisent au moins une collision.*
6. *Donner un algorithme probabiliste pour trouver une fonction de hachage sans collisions. Analyser l'algorithme.*

**Exercice 2.** *La table de hachage dans l'exercice précèdent a $n^2$ entrées. Nous allons obtenir un meilleur résultat avec un* double *hachage.*

*Le double hachage opère de la façon suivante:*

- *Chaque entrée de la table de hachage* primaire *est une référence vers une table de hachage* secondaire.
- *Chaque table de hachage secondaire possède sa fonction de hachage propre. On note $h_i$ la fonction de hachage associée à l'entrée $i$ de la table primaire et $m_i$ la taille de la table de l'entrée $i$.*
- *Lorsqu'un nouvel identifiant id est ajouté à la table on calcule $i = h(id)$ puis $j = h_i(id)$ afin de déterminer dans quelle entrée de la table de hachage référencée depuis l'entrée $i$, il doit être inséré. L'insertion se fait comme pour un table de hachage simple.*
- *La recherche et la suppression suivent un schéma similaire.*

1. *Soit $H'' \subseteq H$ le sous-ensemble des fonctions de hachage qui produisent au moins $n$ collisions. Montrer que, lorsque $m = n$, $|H''| \leq \frac{1}{2}|H|$.*
2. *On choisit une fonction $h \notin H''$ comme fonction de hachage primaire. Notons $n_i$ le nombre d'identifiants id du dictionnaire t.q. $h(id) = i$. En déduire que $\sum_i \frac{n_i(n_i-1)}{2} < n$.*
3. *Comment peut-on choisir les fonctions de hachage secondaires? Quelle est la complexité en espace de ce double hachage?*

# 2 Arbres binaires de recherche

*Rappel* : un arbre binaire de recherche (ABR) est un arbre binaire dans lequel chaque nœud $n$ possède un identifiant *id* et tel que

- chaque nœud du sous-arbre gauche de $n$ a un identifiant inférieur à *id*
- chaque nœud du sous-arbre droit de $n$ a un identifiant supérieur à *id*

**Exercice 3.** *Pour apprécier l'intérêt des arbres, il faut procéder à une analyse probabiliste de la profondeur des nœuds, vue comme une variable aléatoire. L'hypothèse probabiliste que nous faisons est :*

*« L'arbre est obtenu par ajout successif de $n$ nœuds, où le nœud à ajouter est à chaque fois choisi de manière équiprobable parmi les nœuds restants. »*

*Dans la suite nous notons les identifiants des $n$ nœuds par $id_1 < \ldots < id_n$. Soit $1 \leq i < j \leq n$.*

- *$X_{i,j}$ désigne la v.a. qui vaut 1 si $id_i$ et $id_j$ ont été comparés et 0 sinon.*
- *$p(i)$ désigne la v.a. correspondant à la profondeur de $id_i$ (on suppose que la profondeur de la racine est 1, la profondeur de ses fils est 2 et ainsi de suite).*
- *$p_m = \frac{1}{n}\sum_{i=1}^{n} p(i)$ désigne la v.a. correspondant à la profondeur moyenne d'un identifiant dans un arbre aléatoire.*

- $p_{max} = max(\{p(i) \mid 1 \leq i \leq n\})$ *désigne la v.a. correspondant à la profondeur d'un arbre aléatoire.*

1. *Soit* $1 \leq i < j \leq n$. *Montrer que les identifiants* $id_i$ *et* $id_j$ *seront comparés si et seulement si l'un des deux est le premier à être inséré parmi l'ensemble des identifiants compris entre* $id_i$ *et* $id_j$ *(i.e. parmi* $ID(i,j) = \{id_i, id_{i+1}, \ldots, id_{j-1}, id_j\}$*).*
2. *Montrer que* $p_m = 1 + \frac{1}{n}\sum_{i<j} X_{i,j}$.
3. *Calculer* $\mathbf{E}(X_{i,j})$.
4. *Montrer que* $\mathbf{E}(p_m) = -3 + 2(1+\frac{1}{n})\sum_{k=1}^{n}\frac{1}{k}$.
5. *Montrer que* $2(1+\frac{1}{n})\ln(n+1) - 3 \leq \mathbf{E}(p_m) \leq 2(1+\frac{1}{n})(1+\ln(n)) - 3$.
6. *Quelle est la complexité d'une recherche fructueuse (en supposant que les identifiants sont recherchés avec la même probabilité)?*
7. *Quelle est la complexité d'une recherche infructueuse (identique au cas de l'ajout)? (Indication: supposons que l'on recherche un identifiant placé entre* $id_i$ *et* $id_{i+1}$*)?*
8. *Dans ce qui suit, on note* $p_n$ *la v.a.* $p_{max}$ *d'un arbre obtenu par ajout aléatoire de n identifiants et* $f_n = 2^{p_n}$ *la profondeur exponentielle de l'arbre. Notons* $f_{n,i}$ *la v.a. représentant la profondeur exponentielle sachant que le premier identifiant inséré est* $id_i$*. Montrer que*

$$f_{n,i} = 2^{1+max(p_{i-1}, p_{n-i})}$$

*et que*

$$f_{n,i} \leq 2(f_{i-1} + f_{n-i}) \ .$$

9. *Montrer que* $\mathbf{E}(f_n) \leq \frac{4}{n}\sum_{i=0}^{n-1}\mathbf{E}(f_i)$.
10. *Conclure que* $\forall n, \mathbf{E}(f_n) \leq n^3 + 1$
11. *Conclure que* $\forall n > 1, \mathbf{E}(p_n) \leq 3log(n) + 1$