

**IFT2810 – Structures de données**  
**Devoir 2, session automne 2009**  
**À remettre le mardi 24 novembre 2009**

---

**1. Dictionnaires et tables de hachage**

**1.1 (20 points)** Supposons que l'université veut garder en mémoire l'information sur chacun de ces étudiants dans une table de hachage et quelle décide de prendre comme clé l'âge de chaque étudiant. Elle décide d'utiliser une table de 100 cellules avec la fonction de hachage  $f(x) = x \bmod 100$ .

- a) Expliquer pourquoi le choix de cette clé et fonction de hachage n'est pas pertinent dans le contexte.
- b) Pouvez-vous me dire, malgré tout, quelle serait la meilleure méthode de résolution de collisions dans ce contexte (chaînage ou sondage linéaire) et pourquoi ?
- c) Étant donné un fichier contenant l'information sur un étudiant (nom, adresse, téléphone, code permanent, dossier universitaire, date de naissance) trouvez une meilleur façon (meilleur clé, meilleur fonction) de garder l'information sous forme de table de hachage. Expliquez pourquoi votre façon est meilleure.

**1.2 (20 points)** La compagnie de téléphone "DRING" utilise une table de hachage qui garde en mémoire leur banque de données de paires (noms,no.de téléphone). Ils utilisent pour éviter les collisions, une approche de résolution de collisions par chaînage. Depuis quelque temps, un certain Mr. Fido essaie de les convaincre de changer leur méthode de résolution de collisions à la méthode par sondage linéaire. Un employé de la compagnie "DRING" vous appelle, comme consultant, et vous demande de répondre aux questions suivantes :

- a) Dessinez, le résultat de l'insertion des numéros suivantes, dans une table de hachage composée de 13 cellules : # 6374, 7019, 5606, 7662, 1609, 5074 et 5110, lorsque la méthode de compression utilisée est la division modulo 13.
- b) Qu'arrive-t-il si j'enlève la clé "7662" de la table? Est-ce que ça cause des problèmes ?
- c) Pouvez-vous me dire quelle est la meilleure méthode (chaînage ou sondage linéaire) et pourquoi ?

**1.3 (20 points)** Une fonction de hachage  $h(x) = h_2(h_1(x))$  est définie par la composition du code de hachage  $h_1(x)$  et de la fonction de compression  $h_2(x)$  suivants :

$h_1$  : séquence de lettres  $\rightarrow$  Somme des positions de chaque lettre dans l'alphabet  
(a =1, b = 2, c=3, ...)

$h_2$  : entier  $x$   $\rightarrow$   $x \bmod 7$

Insérez les éléments A, B, C, D et E, dans cet ordre, dans une table de hachage de taille 7 et utilisant la fonction de hachage  $h(x)$  et les clés suivantes :

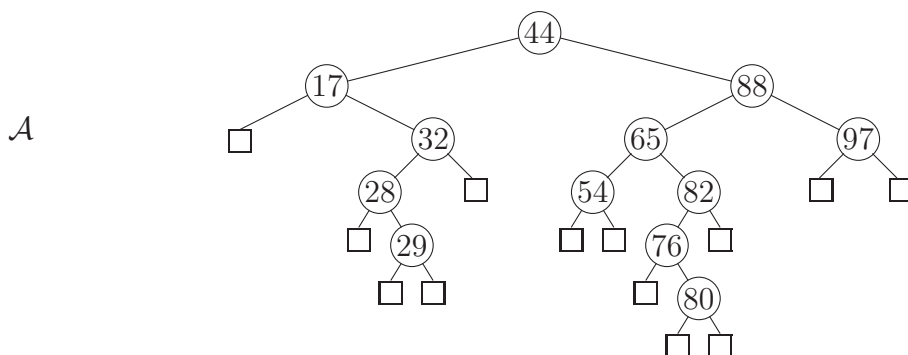
Éléments	A	B	C	D	E
Clés	la	fg	be	al	cd

- En utilisant la méthode de résolution de collisions par chaînage.
- En utilisant la méthode de résolution de collisions par sondage linéaire.
- En utilisant la méthode de résolution de collisions par sondage quadratique

## 2. Arbres Binaires de Recherche

### 2.1 (20 points)

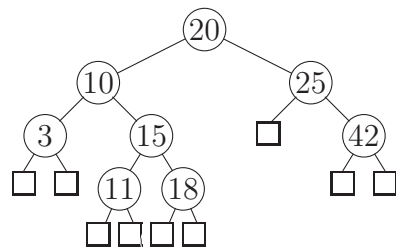
- Insérer dans un arbre binaire de recherche initialement vide les clés suivantes, dans cet ordre : 30, 40, 23, 58, 48, 26, 11, 13. Dessiner l'arbre après chacune des insertions.
- Supprimer dans l'arbre binaire de recherche  $\mathcal{A}$  les clés suivantes, dans l'ordre : 32, 65, 76, 88, 97. Dessiner l'arbre après chaque suppression :



- Si on essaie d'insérer une séquence d'éléments dans un arbre binaire de recherche initialement vide, dans deux ordres différents, il est possible que l'on obtienne deux arbres différents. Construisez un exemple de ce phénomène avec une séquence d'au moins 5 clés.

**2.2 (10 points)** Soit  $T$  un arbre binaire de recherche et soit  $x$  une clé. Donnez un algorithme efficace pour trouver la plus petite clé  $y$  dans  $T$  telle que  $y > x$ . Notez que  $y$  peut être ou ne pas être dans  $T$ . Donnez la complexité en temps de votre algorithme et expliquez.

**2.3 (10 points)** Lorsqu'on fait la recherche d'une clé  $x$  dans un arbre binaire de recherche, on appelle la liste des noeuds visités, la **séquence de clés** de la recherche. Par exemple, lors de la recherche de 18 dans l'arbre suivant, la **séquence de clés** est 20, 10, 15, 18 :



Supposons que nous avons un arbre binaire de recherche gardant en mémoire les clés  $1, 2, 3, \dots, 1000$ , et que l'on veuille faire la recherche de la clé 363. Expliquer pourquoi la séquence suivante NE PEUT PAS être la séquence de clés de votre recherche : 925, 202, 911, 240, 920, 245, 363.