Lille iut Université de Lille

TP 5 : Tables de Hachage

Structures de données
DUT info 2ème année
2018–2019

Les tables de hachage (*HashMap* en Java) permettent une implémentation efficace d'une table d'association. L'idée est la suivante : pour chaque clé k, on calcule un entier de hachage h(k) compris entre 0 et n-1 (n est la taille de la table). On utilise ensuite un tableau de taille n pour stocker les couples (clés, valeurs).
Le but est ici de stocker des mots (qui constituent les clés) en leur associant une valeur entière positive représentant la fréquence d'apparition du mot dans un texte donné. La clé est donc un objet de type `String`, la valeur un entier de type `Integer`.
Pour cela, on implémentera de deux façons différentes l'interface suivante :

```
public interface HashTable {
    public Integer put(String key, Integer value);
    public Integer get(String key);
    public Integer remove(String key);
    public boolean contains(String key);
    public int size();
}
```

Q 1. Écrivez une classe [1] `HashCouple` qui contient le couple (clé, information), et qui implémente l'interface `Map.Entry<String, Integer>`. Pour l'instant, la méthode `hashCode()` retournera le HashCode usuel de la clé [2] (qui peut être négatif en Java).

Q 2. Écrivez une classe `HashTableTest` qui servira de classe de test pour vos deux implémentations de `HashTable`. Vous vous inspirerez des classes de tests fournies pour les TP précédents.

Q 3. Première implémentation : par chaînage.

1. Écrivez une classe `Chainage` qui implémente l'interface `HashTable`. Elle encapsule un tableau de `LinkedList` qui vont contenir les couples dont la clé a la même valeur de hachage. Si aucune taille n'est précisée, le tableau sera de taille `DEFAULT_SIZE`.

   ```
   public class Chainage implements HashTable {
       public static final int DEFAULT_SIZE = 16;
       LinkedList<HashCouple>[] table;
   }
   ```

2. Écrivez une méthode `toString()` qui permet l'affichage de la table de hachage comme toute collection Java : les éléments de la table séparés par des virgules et encadrés par des crochets.
3. Récupérez les mots d'un texte et insérez-les dans votre table de hachage. La valeur associée pourrait par exemple être le nombre d'occurrences du mot dans le texte.

Q 4. Deuxième implémentation : par adressage ouvert.

1. Écrivez à présent une classe `Adressage` qui implémente l'interface `HashTable`. Elle encapsule un tableau de `HashCouple`, et traite les collisions par sondage linéaire.
2. Récupérez les mots d'un texte et insérez les de la même façon dans votre table de hachage. Que constatez-vous par rapport à la première implémentation ?
3. Testez maintenant une autre méthode de hachage (par exemple la valeur numérique correspondant à la position dans l'alphabet de la première lettre du mot). Quelles sont les modifications dans la répartition des données ?

Q 5. Modifiez et augmentez votre code actuel de telle sorte que `HashTable` devienne générique et implémente `Map<K,V>`.

1. Vous pourriez aussi utiliser la classe `AbstractMap.SimpleEntry<String, Integer>`
2. Pensez à redéfinir le `equals` en conséquence.