


TP n° 8


Tables de hachage

Dans ces travaux pratiques, nous implantons le TAD Dictionnaire grâce à notre propre table de hachage. Nous proposons d'implémenter deux mécanismes de gestion des collisions : le chaînage et le sondage linéaire.


Exercice 1 – Table de hachage sans collision


Dans ce 1^{er} exercice, nous proposons une implémentation simple sans gestion des collisions.

 Dans un nouveau projet Eclipse, télécharger et intégrer l'interface **Dictionnaire**. Créer une nouvelle classe **TableHachage**<Clé,Valeur> qui implémente cette interface.


 Dans la classe **TableHachage**, ajouter :

- Une classe **Couple** qui possède deux attributs, une clé de type **Clé** et une valeur de type **Valeur**.
- un tableau d'objets **données** qui contiendra les couples de la table et qui sera initialisé avec une taille de 15.

 Écrire les algorithmes des opérateurs **ajouter**, **est_présent**, **rechercher** et **supprimer** de manière à rendre opérationnelle la table de hachage, mais sans prendre en compte les éventuelles collisions qui pourraient se présenter lors de l'insertion. On prendra soin toutefois d'implémenter les préconditions du TAD.

 Dans le programme principal, créer un dictionnaire dans lequel les clés et les valeurs sont des chaînes de caractères. Remplir ce dictionnaire avec les couples ci-contre (l'ordre d'ajout n'a pas d'importance) :


clé	valeur
USB	Universal Serial Bus
BIOS	Basic Input-Output System
IP	Internet Protocol
BYTE	A byte is a storage unit for data
PC	Personal Computer
MAC	Apple Macintosh
ROM	Read-Only Memory
CPU	Central Processing Unit

 Écrire la méthode `toString` permettant d'afficher la table de hachage de la question précédente sous la forme suivante dans la console :

```

0 -
1 - <BIOS,Basic Input-Output System>
2 -
3 - <IP,Internet Protocol>
4 -
5 -
6 -
7 - <CPU,Central Processing Unit>
8 - <BYTE,A byte is a storage unit for data.>
9 - <USB,Universal Serial Bus>
10 -
11 -
12 - <PC,Personal Computer>
13 - <ROM,Read-Only Memory>
14 - <MAC,Apple Macintosh>


```


 Tester votre implémentation en effectuant les tests suivants sur un dictionnaire contenant les couples listés dans la question précédente :


- Vérifier que la clé *BIOS* est présente, vérifier la valeur associée.
- Vérifier que la clé *WYSIWYG* n'est pas présente.
- Vérifier que la recherche de *WYSIWYG* lève l'exception de clé introuvable.
- Vérifier que la clé *PDF* n'est pas présente.
- Vérifier que si on supprime le couple dont la clé est *BYTE*, ce couple n'est plus présent dans la table.
- Vérifier que si on désire insérer un nouveau couple *<MAC, (adress) Media Access Control>*, l'exception de clé unique est levée.

Exercice 2 – Résolution des collisions par sondage linéaire


 Dupliquer la classe `TableHachage` dans une nouvelle classe `TableHachageSondage`.

 Modifier l'opérateur `ajouter` pour prendre en compte une éventuelle collision, c'est à dire une tentative d'insertion d'un couple dans un emplacement déjà occupé par un autre couple dont la clé est différente. La collision est résolue par une recherche du prochain emplacement libre dans la table.

 Modifier les opérateurs `est_présent` et `rechercher` pour s'adapter à la règle de gestion des collisions. Lors de la recherche d'une clé, on parcourt les cellules depuis l'emplacement correspondant à sa valeur de hachage jusqu'à trouver la clé (la recherche est alors positive) ou une cellule vide (la recherche est alors négative).

 Dans une nouvelle procédure de test, créer un dictionnaire dont les clés et les valeurs sont des chaînes de caractères. Ajouter les couples suivants (dans l'ordre alphabétique des clés) et vérifier pour chacun la valeur de hachage et l'indice occupé dans le tableau :

clé	valeur	valeur de hachage	indice dans le tableau
BIOS	Basic Input-Output System	1	1
BYTE	A byte is a storage unit for data	8	8
CPU	Central Processing Unit	7	7
IP	Internet Protocol	3	3
MAC	Apple Macintosh	14	14
OS	Operating System	12	12
PC	Personal Computer	12	13
PDF	Portable Document Format	8	9
RAM	Random Access Memory	14	0
ROM	Read-Only Memory	13	2
RTFM	Read The F***** Manual	13	4
USB	Universal Serial Bus	9	10
VGA	Video Graphics Array	12	5
WYSIWYG	What You See Is What You Get	10	11


 Modifier l'opérateur **supprimer** pour s'adapter au sondage linéaire. Puisqu'une cellule vide rencontrée lors du parcours rendrait la recherche invalide pour les clés situées en aval, on décide de ne pas supprimer le couple lors de la suppression mais de simplement le "marquer" comme étant supprimé. Pour cela, on modifiera la classe **Couple** pour ajouter un booléen **supprimé** permettant d'indiquer pour un couple s'il a été supprimé (True) ou s'il fait encore partie des couples de la table (False). On pensera aussi à modifier à nouveau les opérateurs **ajouter**, **est_présent** et **rechercher** pour prendre en compte cette nouvelle propriété.

 Ajouter à votre procédure de tests unitaires le test qui consiste en les opérations suivantes :

1. Supprimer les clés *PC* et *ROM*,
2. Vérifier que *RTFM* est bien présent dans votre table,
3. Ajouter les couples suivants : *<GPU, Graphics Processing Unit>* et *<GIF, Graphics Interchange Format>*.

Exercice 3 – Résolution des collisions par chaînage

Dans cet exercice, on propose une implémentation alternative de la table de hachage dans laquelle les collisions potentielles sont résolues par un chaînage des couples dans chaque emplacement.

 Dans une nouvelle classe **TableHachageChaînage**, implémentez l'interface **Dictionnaire** et ses opérateurs en tenant compte du fait que les emplacements du tableau contiennent des listes de couples.

 Utiliser les tests des exercices précédents pour valider l'implémentation par chaînage.