TP n°7 - Tables de hachage [1]

1 Fonctions de hachage

On rappelle qu'une fonction de hachage est une fonction qui à toute clé d'un ensemble C associe un élément d'un ensemble N fini de cardinal raisonnable.

Q1 Le *hachage par division* de largeur m est une fonction de hachage définie sur les entiers et à valeurs dans $[\![0, m-1]\!]$, qui à tout entier k associe le reste de la division euclidienne de k par m.

Écrire une fonction `hash_int : int -> int -> int` qui prend en argument deux entiers m et k et qui renvoie l'image de k pour la fonction de hachage par division de largeur m de k.

Q2 Dans le cas d'une chaîne de caractères, on va d'abord coder la chaîne sous forme d'entier avant de lui appliquer la fonction de hachage : la chaîne $s_0 \dots s_{n-1}$ pourra être codée par l'entier $\sum_{k=0}^{n-1} a_k 256^k$ où a_k est le code ASCII du caractère s_k.

Écrire une fonction `int_of_str : string -> int` qui prend en argument une chaîne de caractères et qui renvoie l'entier décrit ci-dessus.

Q3 En déduire une fonction `hash_string : int -> string -> int` de hachage pour les chaînes de caractères.

2 Tables de hachage de taille fixe

On représente une table de hachage par le type

```
type ('a, 'b) table = {
    h : 'a -> int ;
    contenu : ('a * 'b) list array
  };;
```

Q4 Écrire une fonction `creer_table : ('a -> int) -> int -> ('a, 'b) table` prenant en argument une fonction de hachage et la largeur de la table et qui renvoie une table de hachage vide.

Q5 Écrire une fonction `contient : ('a, 'b) table -> 'a -> 'b` qui prend en argument une table et une clé et qui renvoie `true` si la table contient la clé, `false` sinon.

Q6 Écrire une fonction `trouver : ('a, 'b) table -> 'a -> 'b` qui prend en argument une table et une clé et qui renvoie la valeur associée à la clé si elle existe.

Q7 Écrire une fonction `ajouter : ('a, 'b) table -> 'a -> 'b -> unit` qui prend en argument une table, une clé c et une valeur v et qui ajoute le couple (c, v). On précisera le comportement de la fonction si la clé est déjà présente dans la table.

Q8 Écrire une fonction `supprimer : ('a, 'b) table -> 'a -> unit` qui prend en argument une table et une clé c et qui supprime l'entrée associée à la clé c dans la table.

1. D'après des TP de Vincent Simonet et Judicaël Courant.

3 Tables de hachage de taille dynamique

3.1 Une première approche

Lorsque le nombre de collisions est important, la taille des les listes contenues dans le tableau augmente. Cela sera notamment le cas si on ajoute de nombreuses entrées dans la table.

Or la partie coûteuse de la recherche d'une clé est le parcours de la liste correspondant à l'image de la clé par la fonction de hachage.

On se propose d'améliorer ce point en utilisant des tables de hachage de taille dynamique : lorsque le nombre d'éléments devient trop important, on augmente la taille du tableau (ce qui impose de rajouter un paramètre à la fonction de hachage).
On utilisera donc le type

```
type ('a, 'b) table_dyn = {
    h : int -> 'a -> int ;
    mutable nb : int ;
    mutable contenu : ('a * 'b) list array
  };;
```

où `nb` est le nombre d'éléments présents dans la table.

Q9 Écrire une fonction `creer_table_dyn : (int -> 'a -> int) -> int -> ('a, 'b) table_dyn` prenant en argument une fonction de hachage et la largeur initiale de la table et qui renvoie une table de hachage vide.

Q10 Écrire une fonction `contient_dyn : ('a, 'b) table_dyn -> 'a -> 'b` qui prend en argument une table et une clé et qui renvoie `true` si la table contient la clé, `false` sinon.

Q11 Écrire une fonction `trouver_dyn : ('a, 'b) table_dyn -> 'a -> 'b` qui prend en argument une table et une clé et qui renvoie la valeur associée à la clé si elle existe.

Q12 Si le nombre d'éléments de la table dépasse le double de la largeur de la table, alors on double la largeur de la table. Écrire une fonction `redimensionne_dyn : ('a, 'b) table_dyn -> unit` qui réarrange la table passée en argument en doublant sa largeur.

Q13 En déduire une fonction `ajouter_dyn : ('a, 'b) table_dyn -> 'a -> 'b -> unit` qui prend en argument une table, une clé c et une valeur v et qui ajoute le couple (c, v) en redimensionnant la table si nécessaire.

Q14 Écrire une fonction `supprimer_dyn : ('a, 'b) table_dyn -> 'a -> unit` qui prend en argument une table et une clé c et qui supprime l'entrée associée à la clé c dans la table.

3.2 Largeur et nombre premier

Les tables de hachage sont réputées mieux fonctionner si la taille du tableau utilisé est un nombre premier.
On va donc construire un tableau de nombres premiers proches de puissances de 2. Pour cela, on utilisera le tableau `t` de longueur $N = 40$ défini dans le fichier `tp07_premiers.ml`, qui vérifie : pour tout $i \in [\![0, N[\![$, $2^i - t.(i)$ est un nombre premier.

Q15 Construire un tableau `premiers` tel que pour tout $i \in [\![0, N[\![$, `premiers.(i)` contient $2^i - t.(i)$.

Q16 Reprendre les questions de la partie précédente de sorte que les largeurs de tables appartiennent à la liste `premiers`. On pourra modifier le type `table_dyn` pour y ajouter toute donnée pertinente.