Option Informatique

TP Option informatique $n°04$
Tables de hachage

On souhaite implémenter un type `('a,'b) dict` implémentant une version impérative des dictionnaires avec clés de type `'a` et enregistrements de type `'b`. Les opérations sur ce type seront :

- `creer_dict_vide : unit→('a, 'b) dict`
- `contient : 'a→('a, 'b) dict→bool`
- `valeur : 'a→('a, 'b) dict→'b`
- `inserer : 'a→'b→('a, 'b) dict→unit`
- `supprimer : 'a→dict→unit`
- `modifier : 'a→'b→('a, 'b) dict→unit`

On va ici implémenter nos dictionnaires par des `('a*'b) list array` (tableau de listes de couples (clé,enregistrement)). Chaque case du tableau est appelée une *alvéole* et la liste qu'elle contient donne tous les couples (clé,enregistrement) d'une même famille. Voyons maintenant à quoi correspondent ces familles : on fixe un entier n correspondant à la longueur du tableau et une fonction `h : 'a →int` qui prend ses valeurs dans $\{0, 1, \dots, n-1\}$. La fonction h est appellée *fonction de hachage* et indique, pour toute valeur de clé, dans quelle alvéole on devra stocker le couple correspondant.

Les points essentiels pour s'assurer que cette implémentation soit pertinente sont :

- que n ne soit ni trop grand ni trop petit ;
- que les évaluations de la fonction de hachage h soient *rapides* ;
- que les valeurs prises par la fonction de hachage le soient le plus uniformément possible réparties dans $\{0, \dots, n-1\}$.

Ceci dépend évidemment du type de dictionnaire que l'on souhaite manipuler, on ne s'en préoccupera donc pas dans la suite, et on se contentera d'un exemple extrêmement naïf.

Q1. Un exemple : on suppose que `'a` est le type `string` et qu'on a `n=26`.

Écrire une fonction de hachage `h_exemple : string→int` qui donne l'indice de la première lettre de son argument (si `s` commence par un `a`, alors `h_exemple s` doit retourner `0`, ..., si `s` commence par un `z`, alors `h_exemple s` doit retourner `25`, et comportement arbitraire si `s` est la chaîne vide ou une chaine qui ne commence pas par une lettre ; et si vous gérez les majuscules c'est top).

On définit le type `dict` par

```
type ('a, 'b) dict = {
        hache: 'a -> int;
        table_hachage: ('a * 'b) list array;
        largeur: int};;
```

Q2. Écrire la fonction `creer_dict_vide`. Plutôt que de considérer la fonction de hachage et la longueur de la table comme des paramètres, on considèrera que `creer_dict_vide` est de type `int→('a→int)→('a, 'b) dict`.

Application : créez un `string*string dict` vide avec une table de hachage de longueur 26 et la fonction de hachage `h_exemple`. Appelez-le `d_exemple`.

Q3. Écrire la fonction `contient : 'a→('a, 'b) dict→bool`.

Q4. Écrire la fonction `inserer : 'a→'b→('a, 'b) dict→unit`.
La fonction devra provoquer une exception si la clé est déjà présente dans le dictionnaire.
Application : mettez à jour `d_exemple` *avec quelques mots de la langue française et leur traduction en anglais. Mettez au moins deux mots commençant par la même lettre.*

Q5. Écrire la fonction `valeur : 'a→('a, 'b) dict→'b`.
Application : tester sur `d_exemple`.

Q6. Écrire les fonctions `supprimer : 'a→('a, 'b) dict→unit` et `modifier : 'a→'b→('a, 'b) dict→unit`.
Application : tester sur `d_exemple`, *ainsi que* `contient`.

Q7. Écrire une fonction `frequences_mots : string → (string,float) dict` qui prend comme argument une chaîne de caractères et donne comme résultat un dictionnaire associant à chaque mot présent dans la chaîne de caractères sa fréquence d'apparition (on utilisera la table de hachage `h_exemple` et on considèrera que toutes les séquences de caractères séparées par une espace typographique sont des mots).

Et en vrai ?

Des tables de hachage bien plus pertinentes sont déjà implémentées en OCaml. Il s'agit du module `Hashtbl`.

Q8. En utilisant le module `Hashtbl`, réécrire la fonction `fréquences_mots` mais qui soit de type `string → (string,float) Hashtbl.t`