Rabin-Karp and hashtables

Florian Bourse

Dans les deux parties de ce TP, nous allons utiliser la même fonction de hachage

$$h : u_0 \dots u_{m-1} \mapsto \sum_{i=0}^{m-1} u_i B^{m-1-i} \mod k$$

avec $B = 256$ et k un nombre premier.

1 Rabin-Karp

Question 1. Écrire une fonction

```
is_occurence (text : string) (pat : string) (i : int) : bool
```

qui prend en entrée t et p et renvoie `true` si $t_i \dots t_{i+|p|-1} = p$ et `false` sinon.

Question 2. Écrire une fonction `hash (u : string) (i : int) (m : int) : int` qui prend en entrée u, i et m et calcule $h(u_i \dots u_{i+m-1})$.

Question 3. Écrire une fonction `update : (h : int) (u0 : char) (um : char) : int` qui prend en entrée $h(u_0 \dots u_{m-1})$, u_0 et u_m et qui renvoie $h(u_1 \dots u_m)$.

Question 4. Écrire une fonction `search : (text : string) (pat : string) : int` qui affiche les indices des occurences de p dans t et qui renvoi leur nombre.

2 Création de tableaux associatifs avec une liste d'associations

On souhaite à présent créer un tableau associatif, permettant d'associer des clés de type `'a` à des valeurs de type quelconque `'b`.

On représentera ces tableaux par des listes de couples d'associations, de type `('a * 'b) list`. Le tableau vide sera donc représenté par la liste vide `[]`.

Question 5. Écrire une fonction

```
add (tab : ('a * 'b) list) (k : 'a) (v : 'b) : ('a * 'b) list
```

qui prend en entrée un tableau associatif t, une clé k et une valeur v et qui renvoie un nouveau tableau associatif contenant les associations de t et l'association (k, v).

Question 6. Écrire une fonction

```
find (tab : ('a * 'b) list) (k : 'a) : 'b option
```

qui prend en entrée un tableau associatif t et une clé k et et qui renvoie `Some x` où la dernière valeur v associée à k dans t, ou `None` si v n'existe pas.

Question 7. Écrire une fonction

```
remove (tab : ('a * 'b) list) (k : 'a) : ('a * 'b) list
```

qui prend en entrée un tableau associatif t et une clé k et renvoie un nouveau tableau associatif contenant les associations de t sauf la dernière association ajoutée avec la clé k.

Question 8. Écrire une fonction

```
iter (f : 'a -> 'b -> unit) (tab : ('a * 'b) list)  : unit
```

qui prend en entrée une fonction f et un tableau associatif t et qui applique `f k v` pour chaque association (k, v) du tableau.

Question 9. En déduire une fonction pour afficher le contenu d'un tableau associatif.

3 Implémentation d'une table de hachage

Pour éviter la complexité $O(|t|)$ dans le pire des cas pour la recherche, on préfère utiliser un tableau. On va maintenant créer une table de hachage qui servira de tableau associatif pour des clés de type `string` et des valeurs d'un type quelconque `'a`.

Question 10. Écrire une fonction `create (n : int) : (string * 'a) list array` qui créé une table de hachage de taille n initialement vide.

Question 11. Écrire une fonction

```
add (t : (string * 'a) list array) (k : string) (v : 'a) : unit
```

qui ajoute à la table de hachage t l'association (k, v).

Question 12. Écrire une fonction

```
find (t : (string * 'a) list array) (k : string) : 'a
```

qui prend en entrée un tableau associatif t et une clé k et et qui renvoie `Some x` où la dernière valeur v associée à k dans t, ou `None` si v n'existe pas.

Question 13. Écrire une fonction

```
remove (t : (string * 'a) list array) (k : string) : unit
```

qui supprime de la table de hachage t la dernière association ayant pour clé k.