TP 11 : Tables de Hachage

20 mars 2024

1 Principe des Tables de Hachage

Rappel : La structure de dictionnaire (ou tableau associatif) est une structure dans laquelle on stocke un ensemble de couples clé/valeur, dans laquelle on peut rechercher la présence d'une clé, ajouter ou supprimer une clé, modifier la valeur associée à une clé...

Une implémentation possible des dictionnaires est l'utilisation d'arbres binaires de recherche : on stocke le couple clé/valeur dans l'étiquette. Les clés appartiennent à un ensemble totalement ordonné et sont comparées pour déterminer leur position dans l'arbre binaire de recherche (et pouvoir être ajoutées, supprimées, recherchées en temps $O(h)$).

Une autre implémentation possible est la **table de hachage**. Le principe de base consiste à utiliser la structure de tableau pour tirer parti de la possibilité d'accéder à une case du tableau en temps constant.

On considère un ensemble de couples clés/valeurs avec les clés dans un ensemble E, dont on pourra considérer qu'il s'agit d'une partie de $[\![0; N-1]\!]$. On peut envisager de représenter le dictionnaire par un tableau de taille N, où les clés correspondent aux indices, et la présence d'un couple clé/valeur (c, v) est représentée par la présence de la valeur v à la case $t[c]$ (on peut utiliser une valeur spéciale pour indiquer que la clé n'est pas dans le dictionnaire).

L'accès, l'ajout, la suppression ou la modification de couples clés/valeurs se font alors en temps $O(1)$. Si les clés ne sont pas dans l'ensemble des entiers (chaînes de caractère par exemple), mais dans un ensemble de cardinal N, on peut créer une bijection entre cet ensemble et $[\![0; N-1]\!]$.

On peut avoir plusieurs problèmes :

- Les clés peuvent ne pas être des entiers ;
- L'ensemble dans lequel se situe les clés peut être de cardinal largement supérieur à celui de l'ensemble des couples clés/valeurs

Pour pallier à ce problème, on utilisera un tableau de taille m comparable au nombre d'éléments dans la structure, et on associe à chaque clé possible un nombre entre 0 et $m-1$ via une **fonction de hachage** h. Ce nombre correspondra à l'indice du tableau associé à la clé, et auquel on stockera la valeur qui lui est associée.

Comme les clés sont potentiellement dans un ensemble théoriquement plus grand, h n'est pas nécessairement injective : plusieurs clés peuvent avoir une même image, et donc correspondre à un même indice dans le tableau. On parle de **collision**.

Il existe plusieurs méthodes pour résoudre le problème des collisions. L'une d'entre elles est la **résolution par chaînage**, où l'on ne stocke pas de clés mais des listes de clés à chaque case du tableau (appelée *alvéole*). La présence d'une clé c dans un tableau est indiquée par la présence du couple clé/valeur (c, v) dans la liste à l'alvéole $h(c)$ associée par la table de hachage.

2 Exercices

Exercice 1

Pour des clés entières de valeur largement supérieure au nombre d'éléments du tableau, on peut choisir d'associer à une clé c une valeur comprise entre 0 et $m-1$ pour un m donné en prenant le reste dans la division euclidienne de c par m.

Ecrire une fonction `reste` de type `int -> int -> int` telle que `reste m c` renvoie le reste dans la division euclidienne de c par m. (En particulier, `reste m` sera une fonction de type `int -> int`)

Exercice 2

Pour effectuer un hachage à partir d'une clé c de type quelconque vers un entier entre 0 et $m-1$, on peut d'abord associer un entier n à la clé c, puis calculer le reste de la division euclidienne de n par m. Par exemple, pour une chaîne de caractères ASCII, on peut associer à chaque caractère l'entier entre 0 et 127 associé : une séquence de caractère peut être associée à une séquence de chiffres en base 128, et donc au nombre correspondant.

Ecrire une fonction `hachage_chaine` qui prend en entrée une séquence de caractères $s_0 s_1 ... s_{n-1}$ et renvoie le nombre $\sum_{i=0}^{n-1} code(s_i) \times 128^i$.

Exercice 3

On définit un type table de hachage avec la donnée :

- D'une fonction de hachage qui prend en entrée des clés de type `'k` et renvoie l'entier correspondant à l'alvéole associée à la clé ;
- D'un tableau de listes d'éléments de type `('k * 'v)`

```
type ('k, 'v) table_hachage = { hache: 'k -> int; donnees: ('k * 'v) list array };;
```

1) Définir une fonction `creer_table` qui prend en entrée une fonction `h` et une longueur `m` et renvoie une table de hachage de taille `m` utilisant la fonction de hachage `h`.

2) Ecrire une fonction `recherche` qui prend en entrée une table de hachage `t` et une clé `k` et renvoie Vrai si la clé est dans la table de hachage `t`.

3) Ecrire une fonction `element` qui prend en entrée une table de hachage `t` et une clé `k` et renvoie la valeur associée à la clé est dans la table de hachage `t` (avec une exception si la clé n'est pas dans la table).

4) Ecrire une fonction `ajout` qui prend en entrée un tableau `t` et un couple `(k,v)` et qui ajoute ou modifie la valeur `v` associée à la clé `k` dans le tableau.

5) Ecrire une fonction `suppression` qui prend en entrée un tableau `t` et une clé `k` et supprime la clé du tableau et sa valeur associée.