Algorithmique – Programmation Objet – Python
TD n° 10
Tables de hachage

Licence Informatique 2ème année
Université de Nice Sophia Antipolis

Introduction

Une table de hachage est une structure de données classique qui permet une association clé-élément, où chaque élément est associé à une clé et chaque clé correspond à un seul élément. Elle consiste en un tableau (la table de hachage), dont les cases sont appelées *alvéoles*, et une fonction (la fonction de hachage). La fonction de hachage associe à une clé une valeur de hachage. Elle fait donc correspondre la clé d'un élément à une valeur qui est utilisée comme index dans la table de hachage pour cet élément.
Le but de cette structure de données est d'accéder le plus rapidement possible à un élément à partir de sa clé.
Les opérations permises par une table de hachage H sont les suivantes :

- INSÈRE(*élt*, *clé*) : insère dans la table de hachage un élément *élt* dont la clef est *clé*.
- ÉLÉMENT RECHERCHE(*clé*) : recherche dans la table de hachage si un élément est associé à la clef *clé* et renvoie cet élément ; le cas échéant, renvoie **nil**.
- **booléen** APPARTIENT(*clé*) : renvoye **vrai** si un élément est associé à la clef *clé* dans la table de hachage.

Lorsque deux clés distinctes ont la même valeur de hachage, on dit qu'on a une collision. Ces clés ne peuvent être stockées à la même position, on doit alors employer une stratégie de résolution des collisions.
Il existe deux méthodes principales pour gérer le problème des collisions :

1. par chaînage;
2. par adressage ouvert.

1 Résolution des collisions par chaînage

Nous allons utiliser pour résoudre les conflits par chaînage un tableau annexe à deux dimensions : la première contient des éléments (et donc leur clé), alors que la seconde contient l'indice du prochain élément ayant la même valeur de hachage que la clé courante et 0 s'il n'y a pas de valeur suivante. La valeur de la table de hachage pour une entrée donnée contient un indice dans cette table de débordement.
Par exemple, on considère que la fonction de hachage renvoie la valeur de la clé modulo 10. Si on insère successivement les paires (élément,clé) suivantes $(e_1, 12)$, $(e_2, 32)$, $(e_3, 43)$, $(e_4, 55)$, $(e_5, 42)$ et $(e_6, 53)$, alors la table de hachage et la table de débordement seront :

table de hachage	
valeur hachage	indice débordement
0	0
1	0
2	1
3	3
4	0
5	4
6	0
7	0
8	0
9	0

table de débordement			
indice	clé	élément	suivant
1	12	e_1	2
2	32	e_2	5
3	43	e_3	6
4	55	e_4	0
5	42	e_5	0
6	53	e_6	0

1. Détaillez le remplissage d'une table de taille m avec les 8 éléments suivants : $(e_1, 22)$, $(e_2, 12)$, $(e_3, 5)$, $(e_4, 8)$, $(e_5, 17)$, $(e_6, 2)$, $(e_7, 3)$, $(e_8, 10)$, en utilisant la fonction de hachage $hash(i)\{$ renvoyer $(i \bmod m)\}$, avec $m = 10$.

2. Expliquez comment l'élément correspondant à la clé 2 est retrouvé, puis comment l'élément correspondant à la clé 1 est recherché.

3. Donnez les algorithmes implémentant cette table de hachage. Il s'agit donc de donner les algorithmes correspondant aux trois fonctions de la structure de données.

2 Résolution des collisions par adressage ouvert

L'adressage ouvert consiste, dans le cas d'une collision, à stocker les valeurs de hachage dans d'autres alvéoles de la table. La position de ces alvéoles est déterminée par une méthode de « sondage ». Plus précisément, lors d'une recherche, si la case obtenue par hachage direct ne permet pas d'obtenir le bon élément, alors une recherche sur les cases obtenues par une méthode de sondage est effectuée jusqu'à trouver l'élément, ou non, ce qui indique qu'aucun élément de ce type n'appartient à la table.
Les méthodes de sondage courantes sont (pour une table de taille m) :

- Sondage linéaire (*linear probing*) : l'intervalle entre les cases est fixe, souvent 1. La position de l'alvéole suivante pour le ième appel est donnée par $next(clé, i) = (hash(clé) + i) \bmod m$. L'ordre de parcours des alvéoles est $T[hash(clé)], T[hash(clé) + 1], T[hash(clé) + 2]$...

- Sondage quadratique (*quadratic probing*) : l'intervalle entre les cases augmente linéairement. Les indices des cases augmentent donc quadratiquement : +2 , +6, +12, +20 ... La position de l'alvéole suivante pour le ième appel est donnée par $next(clé, i) = ((hash(clé) + i + i^2) \bmod m$. L'ordre de parcours des alvéoles est $T[hash(clé)], T[hash(clé) + 2], T[hash(clé) + 6]$...

- Double hachage : l'adresse de la case est donnée par une deuxième fonction de hachage, ou hachage secondaire. La position de l'alvéole suivante pour le ième appel est donnée par $next(clé, i) = (hash_1(clé) + i \times hash_2(clé))$ $\bmod m$.

1. Détaillez le remplissage de la table avec les 8 éléments suivants : $(e_1, 22)$, $(e_2, 12)$, $(e_3, 5)$, $(e_4, 8)$, $(e_5, 17)$, $(e_6, 2)$, $(e_7, 3)$, $(e_8, 10)$ munie de la fonction de hachage $hash(i)\{$ renvoyer $(i \bmod m)\}$, avec $m = 10$ dans le cas d'un sondage linéraire. Expliquez comment l'élément correspondant à la clé 2 est retrouvé, puis comment l'élément correspondant à la clé 1 est recherché.

2. Répondez aux mêmes questions avec un sondage quadratique.

3. Donnez les algorithmes implémentant une table de hachage dans tous les cas de sondage. Quel problème observe-t-on avec le sondage quadratique ? Comment peut-on éviter cela ? Il s'agit donc de donner les algorithmes correspondant aux trois fonctions de la structure de données.

3 Suppression dans une table à adressage ouvert

La suppression d'un élément dans une table de hachage ouverte n'est pas une chose aisée. Cependant, voici une méthode possible.
Au lieu d'avoir deux statuts possibles pour une case qui sont « vide » et « contenant un élément », on introduit un troisième type de valeur possible : « supprimé ». Une valeur insérée peut prendre la place d'une valeur « supprimée », et les valeurs « supprimée » doivent être ignorées pendant la recherche d'élément. Elles ne doivent surtout pas être considérées comme « vides ». Puis, lorsque le nombre de cases « supprimée » devient trop important, on recrée la table de hachage.

1. Discutez des avantages et inconvénients de cette méthode.
2. Knuth a proposé une autre méthode beaucoup plus subtile. Un intervalle d'indices $[i, j]$ d'une table de hachage H est appelé « bloc » si $H[j]$ est vide et toutes les entrées de $H[i]$ à $H[j-1]$ ne sont pas vides. On veut supprimer l'élément qui est situé en $H[i]$. On considère que $[i, j]$ est un bloc.
   (a) Montrez que si on a pour chaque élément dont la clé est *clé*, se trouvant dans une case entre $i+1$ et $j-1$, $hash(clé) > i$, alors on peut positionner $H[i]$ à « vide » sans crainte.
   (b) Soit k un indice avec $i < k < j$ tel que l'élément contenu dans $H[k]$ vérifie pour sa clé c $hash(c) \leq i$ et pour chaque élément dans une case entre $i+1$ et $k-1$ dont la clé est *clé* on a $hash(clé) > i$. Montrez que l'on peut mettre l'élément contenu dans la case $H[k]$ dans la case $H[i]$ et considérer que l'on veut maintenant vider la case $H[k]$.
3. A partir de ces deux propriétés, essayez d'écrire un algorithme qui gère la suppression d'un élément.