TD d'algorithmique numéro 10 - Tables de hachage
Olivier Bodini et Natacha Portier
lundi 26 novembre 2001

Question 1 ⋆ Tables de hachage avec chaînage ⋆

On cherche à implémenter un dictionnaire dynamique sur lequel on désire pouvoir effectuer efficacement les trois opérations de base suivantes : INSERER(mot), RECHERCHER(mot), SUPPRIMER(mot) et tout ceci en occupant un espace mémoire raisonnable. Soit Σ un alphabet et Σ^* l'ensemble des mots sur cet alphabet. On appelle *fonction de hachage* une application h de Σ^* dans $\{0, ..., m-1\}$ où m est un entier donné. Grâce à cette fonction, nous allons construire un tableau $T[0..m-1]$ de listes chaînées, appelé *table de hachage* du dictionnaire D de la manière suivante :

- $\forall i$, $0 \leq i \leq m-1$, $T[i] := Nil$.
- Pour tout mot w dans D, insérer w en tête de la liste $T[h(w)]$.

Question 1.1 ⋆ Insérer,Rechercher,Supprimer ⋆

Donner un pseudo-code pour les opérations INSERER(T, w), RECHERCHER(T, w), SUPPRIMER(T, w).

Question 1.2 ⋆ Complexité ⋆

Considérons que le dictionnaire contienne n mots. On note $\alpha = n/m$ le *facteur de remplissage*, c'est à dire la longueur moyenne des listes chaînées. De plus, faisons l'hypothèse (appelée *hypothèse de hachage uniforme simple*) que les listes chaînées de la table de hachage de D ont toutes α termes (même si α n'est pas un entier !), et supposons que la valeur $h(w)$ peut être calculée en $O(1)$. Déterminer sous ces hypothèses la complexité de ces trois algorithmes.

Remarque : En pratique, la difficulté est de trouver une bonne fonction de hachage, c'est à dire une fonction de hachage qui vérifie (à peu près) l'hypothèse de hachage uniforme pour les dictionnaires que l'on risque de rencontrer. Voici quelque exemple de fonctions qui marchent bien en pratique (ceci reste très empirique !).
A chaque mot w, on associe un entier naturel, sa *clef* notée clef(w), on peut définir alors la fonction de hachage suivante $h(w) = clef(w) \mod m$ où m est un entier judicieusement choisi.
Une autre possibilité est de choisir un réel A, $0 < A < 1$ et de définir

$$h(w) = \lfloor m \times (clef(w) \times A \bmod 1) \rfloor .$$

Question 1.3 ⋆ Hachage universel ⋆

Pour éviter qu'un dictionnaire ne provoque le pire comportement qu'un algorithmicien puisse imaginer, (c'est à dire que la table contient une unique liste chainée non vide de longueur n), nous allons introduire une part de hasard dans notre algorithme. On va pour cela tirer aléatoirement la fonction de hachage pendant l'exécution parmi une classe appropriée de fonctions. Plus explicitement, soit H une collection finie de fonctions de hachage de Σ^* dans $\{0, ..., m-1\}$, H est *universelle* si pour chaque paire de mots distincts w,w', le nombre de fonctions de hachage $h \in H$ pour lesquelles $h(w) = h(w')$ vaut exactement $|H|/m$.

On se propose maintenant de construire une collection universelle de fonctions de hachage. On suppose dorénavant que m est premier. A tout mot w, on associe bijectivement un $(r+1)$-uplet $(x_0, ..., x_r)$ d'entiers inférieurs à m. Pour tout $a \in \{0, ..., m-1\}^{r+1}$, on définit la fonction de hachage :

$$h_a(w) = \sum_{i=0}^{r} a_i x_i \bmod m.$$

Montrer que $\bigcup_a \{h_a\}$ est une collection universelle de fonctions de hachage.

Question 1.4 ⋆ Adressage ouvert ⋆

On cherche dorénavant à ne plus utiliser de listes chaînées mais à pouvoir stocker injectivement n'importe quel dictionnaire de moins de m mots dans une table $T[0, ..., m-1]$. Pour cela, on définit une *séquence de sondage* :

$$h : \Sigma^* \times \{0, ..., m-1\} \to \{0, ..., m-1\}$$

qui vérifie que pour tout w, $(h(w, 0), ..., h(w, m-1))$ est une permutation de $(0, .., m-1)$.
Le pseudo-code de l'algorithme INSERER[T,w] sera alors le suivant :

INSERER(T,w)
$i := 0$;
répéter $j := h(w, i)$
  si $T[j]$ =NIL ou $T[j]$ =OTÉ
    alors $T[j] := w$; retourner j;
    sinon $i := i + 1$;
jusqu'à $i = m$;
erreur "table trop petite"

Expliquer ce que fait le pseudo-code ci-dessus.
Ecrire un pseudo-code pour les opérations RECHERCHER(T, w) et SUPPRIMER(T, w).

Question 1.5 ⋆ Complexité ⋆

Sous l'hypothèse de hachage uniforme simple, montrer qu'étant donnée une table de hachage en adressage ouvert avec un facteur de remplissage $\alpha = n/m < 1$, le nombre moyen attendu de sondages lors d'une recherche infructueuse vaut au plus $1/(1-\alpha)$.
Montrer qu'insérer un élément demande au plus $1/(1-\alpha)$ sondages en moyenne.
Montrer que si la recherche est fructueuse alors le nombre moyen vaut au plus :

$$\frac{1}{\alpha} \ln\left(\frac{1}{1-\alpha}\right)$$

Question 1.6 ⋆ Exemples classiques de sondage ⋆

On considère l'insertion des mots $10, 22, 31, 4, 15, 28, 17, 88, 59$ dans une table de hachage de longueur $m = 11$ en utilisant l'adressage ouvert avec pour fonction de hachage principale $h'(w) = w \mod m$. Illustrer le résultat de l'insertion de ces mots par *sondage linéaire*, c'est à dire avec $h(w, i) = (h'(w) + i) \mod m$, par *sondage quadratique* avec $c_1 = 1$ et $c_2 = 3$, c'est à dire avec $h(w, i) = (h'(w) + c_1 i + c_2 i^2) \mod m$ et par *double hachage* avec $h_2(k) = 1 + (k \mod (m - 1))$, c'est à dire avec $h(w, i) = (h'(w) + ih_2(w)) \mod m$.

Question 2 ⋆ Sur le nombre de racines réelles d'un polynôme de $\mathbb{R}[X]$ dans un intervalle ⋆

Soit f un polynôme de $\mathbb{R}[X]$ et soient a et b deux réels, $a < b$. On dit qu'un suite de polynômes $f_0 = f, f_1, ..., f_s$ est une *suite de Sturm* pour f sur $[a, b]$ lorsque les conditions suivantes sont vérifiées :

1. $f(a) \times f(b) \neq 0$,
2. la fonction polynôme f_s ne s'annule pas sur $[a, b]$,
3. si c est un réel tel que $a < c < b$ et $f_j(c) = 0$ pour un indice j vérifiant $0 < j < s$, alors on a $f_{j-1}(c) \times f_{j+1}(c) < 0$,
4. si c est un réel tel que $a < c < b$ et $f(c) = 0$, alors le polynôme produit $f(c) \times f_1(c)$ est du signe de $x - c$ au voisinage du point c.

A une telle suite et à un point $x \in]a, b[$, on associe le *nombre de variations de signe de la suite au point* x, noté $V(f_0 = f, f_1, ..., f_s; x)$ (ou plus simplement $V(x)$), défini par la formule :

$$V(x) = |\{(i, j)\,; 0 \leq i < j \leq s; f_i(x) \times f_j(x) < 0 \text{ } et \text{ } f_k(x) = 0 \text{ } si \text{ } i < k < j\}|$$

Question 2.1 ⋆ Théorème de Sturm ⋆

Montrer le théorème suivant : Soit f un polynôme à coefficient réels et soient a et b deux réels, $a < b$. Si $f_0 = f, f_1, ..., f_s$ est une suite de Sturm pour f sur $[a, b]$, alors le nombre de racines distintes de f sur l'intervalle $[a, b]$ est égal à $V(a) - V(b)$.

Question 2.2 ⋆ construction d'une suite de Sturm ⋆

Soit f un polynôme à coefficient réels et soient a et b deux réels, $a < b$. Considérons la suite $f_0 = f, f_1, ..., f_s$ définie par : $f_0 = f, f_1 = f', f_2 = -f \mod f', ..., f_{i+1} = -f_{i-1} \mod f_i, ..., f_{s+1} = 0$, alors la suite $g_i = f_i/f_s$, $0 \leq i \leq s$, est une suite de Sturm pour g_0 sur $[a, b]$, pourvu que l'on ait $g_0(a) \times g_0(b) \neq 0$.

Question 2.3 ⋆ algorithme ⋆

Montrer le théorème suivant : Soit f un polynôme à coefficient réels et soient a et b deux réels, $a < b$. Considérons la suite $f_0 = f, f_1, ..., f_s$ définie par : $f_0 = f, f_1 = f', f_2 = -f \mod f', ..., f_{i+1} = -f_{i-1} \mod f_i, ..., f_{s+1} = 0$, alors si $f(a)$ et $f(b)$ sont non nuls, le nombre de racines distintes de f sur l'intervalle $[a, b]$ est égal à $V(a) - V(b)$.
En déduire un algorithme donnant le nombre de racines réelles de f dans l'intervalle $[a, b]$.