IFT2015 Hiver 2011 — Devoir 6

Miklós Csűrös

7 avril 2011

À remettre avant 20 :15 le 14 avril. Remettez un rapport écrit par courriel (à csuros@iro. . .) en format PDF.

6.1 Hachage par multiplication (4 points)

Hachage par multiplication se fait à l'aide d'un paramètre γ :

$$h_\gamma(x) = \Big\lfloor M \cdot \Big\{ \gamma \cdot x \Big\} \Big\rfloor .$$

(Rappelez que $\{z\} = z - \lfloor z \rfloor$.)

Exemple (2 points). Les clés $10, 22, 31, 4, 15, 28, 17, 88, 65$ sont inserés (dans cet ordre) dans un tableau de hachage $T[0..M-1]$ en utilisant la méthode de la multiplication, avec $\gamma = \frac{\sqrt{5}-1}{2}$. Montrez le placement des clés dans le tableau quand $M = 12$ et on utilise l'adressage ouvert avec sondage linéaire (séquence de sondage : $h(x), h(x)+1, h(x)+2, \ldots \pmod M$).

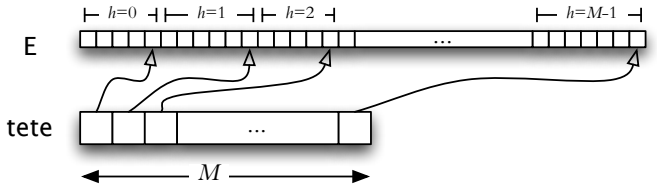
Arithmétique entière (2 points). On veut calculer h_γ par arithméthique entière sur $w = 16$ bits. Soit $M = 4096$ et $\gamma = 1/e = 0.367879\cdots$. Identifiez quelles valeurs *entières* A, d on doit utiliser dans le code suivant[1] pour calculer $h_{1/e}$ sur 16 bits. L'opération $\ggg$ dénote décalage vers le droit (*right bitshift*, `>>>` en Java).

$$h \leftarrow \big((A \cdot x) \bmod 2^w\big) \ggg d$$

[1] Notez que $\bmod 2^w$ est «automatique» quand on travaille avec des entières de w bits. En Java, la conversion de type `(short)` garde les derniers 16 bits, ce qui correspond à $\bmod 2^w$ avec $w = 16$.

6.2 Chaînage par tableau (6 points)

On a une fonction de hachage $h\colon \mathcal{X} \mapsto \{0, 1, 2, \ldots, M-1\}$. Considérez la structure suivante qui implante le TA dictionnaire avec recherche seulement (mais ne permet ni suppression, ni insertion). L'idée est d'utiliser chaînage séparé, mais au lieu de mettre les éléments dans des listes chaînées, on utilise un seul tableau $\mathsf{E}[0..n-1]$ pour stocker tous les éléments. Le tableau est trié selon la valeur de hachage : $h(\mathsf{E}[0]) \leq h(\mathsf{E}[1]) \leq \cdots \leq h(\mathsf{E}[n-1])$. La structure maintient un tableau tete qui permet de trouver rapidement les indices pour lesquelles $h(E[i]) = h(x)$. Par exemple, $\mathsf{tete}[j] = \max\{i = 0, \ldots, n-1\colon h(E[i]) \leq j\}$ pour chaque $j = 0, \ldots, M-1$.



Initialisation. (3 points) La structure est initialisée avec $\mathsf{E}[\,]$ (déjà trié dans l'ordre croissant de $h(E[i])$) comme argument. Donnez un algorithme (INITTETE) pour remplir le tableau $\mathsf{tete}[\,]$ à l'initialisation. L'algorithme doit prendre $O(n)$, et utiliser une espace de travail $O(1)$ (à l'addition des tableaux $\mathsf{E}[\,]$ et $\mathsf{tete}[\,]$). Vous avez le droit de changer la définition de tete (mais il doit rester un tableau de taille $M + O(1)$ au plus) si cela convient à votre implantation de search.

Recherche. (3 points) Donnez une implantation de l'opération $\mathsf{search}(x)$ (en utilisant tete proprement initialisé) qui retourne l'indice de x dans E s'il est là, sinon -1. Caractérisez le temps de calcul de votre implantation comme une fonction de $\alpha = n/M$, en notation asymptotique. Comparez le temps moyen de recherche de cette structure de chaînage, et celui d'une structure simple avec $E[\,]$ trié selon les clés (où search oeut utiliser la recherche binaire) : pour quelles valeurs de M, n est la première plus efficace que la seconde ?