TD7 : hash

TBD : PP

- hash
- adressage ouvert
- du nombre au dictionnaire

Indexation pseudo-aléatoire et Hash sets

Soit E un ensemble de n messages de taille quelconque. On souhaite *indexer* ces messages à l'aide d'une fonction d'encodage H qui va de $\mathcal{M}$ (l'ensemble des messages possibles) vers l'intervalle $[0,...,2^h[$, où h est le nombre de bits servant à stocker l'index.

1. On assimile pour simplifier l'ensemble des messages possibles à l'ensemble des entiers naturels. On souhaite que deux messages différents aient des valeurs d'index différentes.

- Cela n'est possible bien entendu que dans la mesure où le nombre de messages à indexer n est $< 2^h$;
- On appelle *collision* le fait que deux messages différents m_1 et m_2 produisent un code identique, i.e. $H(m_1) = H(m_2)$.

On souhaite minimiser le risque de collision. Il faut pour cela que deux entiers quelconques aient une probabilité $1/2^h$ d'entrer en collision. Donner une fonction qui répartisse de manière *uniforme* l'ensemble des entiers naturels vers l'intervalle $[0,...,2^h[$.

2. Le principe général d'une fonction de hachage est de produire un code à partir de tous les bits du message m (et non seulement les bits de poids faible). Proposez une méthode itérative permettant d'obtenir un tel code à partir d'un entier quelconque. Quelle est sa complexité?

3. On souhaite à présent que deux entiers très similaires i et j produisent des codes très différents. Par exemple, si $|i-j|=1$, on veut que $|H(i) - H(j)| >> 1$ (en conservant la propriété précédente). Comment faire?

4. On souhaite construire un tableau de taille 2^h dans lequel les n messages sont indexés par la fonction de hachage (les valeurs de l'index sont donc différentes pour chaque message).

- Comment faire?
- Montrer que le test d'appartenance du message m à l'ensemble E est alors en O(1)
- Quelle est la complexité moyenne de l'insertion d'un nouveau message dans le tableau?

Table d'allocation

On considère un tableau de taille n dans lequel $p < n$ cases sont occupées. On suppose que chaque donnée d est indexée par l'adresse $i < n$ donnant sa position dans le tableau et que l'on connaît également sa taille m. On suppose de plus que la *table d'allocation* des différentes cases du tableau est codé au format binaire dans un entier B de n bits. $$B = \underset{0}{0}\underset{1}{0}\underset{0}{1}\underset{0}{0}\underset{1}{0}\underset{0}{1}\underset{0}{0}\underset{1}{0}\underset{0}{.}\underset{.}{.}\underset{.}{.}\underset{n-1}{1}$$

On suppose qu'il existe une fonction $f(B,i)$ donnant le ième bit de B.

- Écrire un algorithme permettant d'insérer une donnée d dans le premier bloc de m cases disponible (pensez à mettre à jour la table d'allocation B).
- Peut-on faire mieux en appliquant un pré-traitement à B?

Dictionnaires

On appelle *dictionnaire* une structure de données dont les valeurs sont indexées par des *mots* et non plus par des entiers. Les mots (de taille quelconque) servant à l'indexation sont appelés des clés et les valeurs indexées sont également de taille quelconque.

- Proposez une structure de données permettant d'implémenter une structure de dictionnaire de manière efficace à partir d'une table des clés et d'une table des valeurs (recherche et insertion en O(1)).
- Écrivez une fonction qui compte le nombre d'apparitions de chaque mot dans un texte. On suppose qu'il existe une fonction permettant d'extraire les mots du texte. Vous devez utiliser une structure de dictionnaire.
- Écrivez une fonction qui compte le nombre d'apparitions de couples de mots dans un texte (Exemple : "casse" peut être suivi de "tête", "pied", "noix", "toi", "le", "la", "du" etc.). Déduisez-en une méthode permettant de générer du texte aléatoire à partir d'un texte (ou d'un corpus de textes) de taille suffisante.

From:
https://wiki.centrale-med.fr/informatique/ - **WiKi informatique**

Permanent link:
https://wiki.centrale-med.fr/informatique/tc_info:td7-alt

Last update: **2019/04/30 12:44**