Le sujet

Partie A

Soit U un “univers” dont les éléments sont appelés *clés*. Soit E un ensemble de clés. On suppose que l'on a une fonction $h:U\to \{0,\ldots m-1\}$, dite *fonction de hachage* (ou *hashcode*). Une *table de hachage* est un tableau $T[0 \ldots m-1]$ tel que $T[i]$ est une liste contenant les éléments x de E tels que $h(x)=i$. Si deux éléments de E ont le même hashcode, on dit qu'on a *collision*.

Exercice 0

Donnez des algorithmes pour rechercher, insérer & supprimer un élément dans une table de hachage. Donnez leur complexité dans le cas le meilleur, le pire & en moyenne.

Déduisez-en la valeur optimale (en ordre de grandeur) de m en fonction de n, ainsi qu'une contrainte sur la fonction de hachage.

Exercice 1

On considère les deux fonctions de hachage suivantes :

- La *méthode de la division* : $h(x) = x \: \mathrm{mod}\: m$.
- La *méthode de la multiplication* : $h(k) = E(m\cdot \mathrm{Frac}(x\cdot A))$, où :
  - A est un nombre de $[0\ldots1]$
  - E est la partie entière & *Frac* la partie fractionnaire $(\mathrm{Frac}(x)=x-E(x))$

Donnez, pour ces deux méthodes, des bonnes valeurs pour les paramètres m & A.

Exercice 2

Dans le *hachage cryptographique*, on veut en plus que, connaissant x (& $h(x)$), il soit impossible (à moins de ressources en temps de calcul rédhibitoires) de construire $y\ne x$ tel que $h(y) = h(x)$. Donnez des exemples d'applications du hachage cryptographique.

Exercice 3

Il arrive souvent que l'on ne sache pas à l'avance combien d'éléments contient E & que l'on mette les éléments de E dans T l'un après l'autre sans savoir quand on s'arrêtera. Donnez une ``politique" efficace de gestion de la taille de T.

Exercice 4

Soit S un ensemble de nombres à trier, on répartit S en une table de hachage tel que la fonction de hachage soit croissante ($x\leq y \Longrightarrow h(x) \leq h(y)$). On trie chaque paquet, puis on concatène. On appelle ce tri le *tri par paquets*.

- Donnez une fonction de hachage simple & croissante.
- Quelle est la complexité de cet algorithme dans le cas le meilleur, le pire & en moyenne.

Partie B

Un *dictionnaire* est une structure de données (python) qui se présente ainsi~:

```
D = {clé_1:valeur_1, clé_2:valeur_2, ..., clé_n:valeur_n}
```

Les clés pouvant être de (presque) n'importe que type (& pas seulement l'ensemble $\{0,...,$ $n-1\}$ comme avec une liste).

- On accède à `valeur_i`, la valeur associée à `clé_i` par `D[clé_i]`.
- L'opération `D[clé_p]` $\gets$ `D[valeur_p]`,
  - si `clé_p` n'est pas une clé de `D`, ajoute cette nouvelle clé à `D` & lui associe la valeur `valeur_p`
  - si `clé_p` est déjà une clé de `D`, elle change la valeur qui lui est associée en `valeur_p`.

Exercice 5

Donnez une implémentation efficace des dictionnaires. Quelle est alors la complexité (dans le cas le meilleur, le pire & en moyenne) des fonctions de base (recherche, ajout d'un élément,…) sur un dictionnaire.

Exercice 6

Utilisez un dictionnaire pour écrire un algorithme qui supprime les doublons d'une liste. Donnez sa complexité (dans le cas le pire, le meilleur & en moyenne).

Exercice 7

Utilisez un dictionnaire pour écrire un algorithme qui compte le nombre d'occurrences de chaque mot d'un texte. Donnez sa complexité (dans le cas le pire, le meilleur & en moyenne).

Partie C

Exercice 8

Table d'allocation

On considère un tableau T de taille n dans lequel

- $p<n$ cases sont occupées. Chaque donnée d est indexée par l'adresse $i<n$ donnant sa position dans le tableau & on connaît sa taille m (d occupe m cases consécutives de T).
- 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=0010010100100...01
```

- qu'il existe une fonction $f(B,i)$ donnant le i^{eme} bit de B ($f(B,i)$ vaut 1 si la i^{eme} case de T est occupée, & 0 si elle est libre).

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~?

Ancien sujet