Les tableaux associatifs

Nous allons maintenant étudier un autre type abstrait de données: les dictionnaires, aussi appelés tableaux associatifs.

Un tableau associatif est un type abstrait qui associe des valeurs à des clés, nous verrons plus bas, les opérations disponible sur ce type.

On retrouve une structure qui ressemble, à première vue, beaucoup à un tableau (à chaque élément on associe un indice de position).

Mais au lieu d'associer chaque élément à un indice de position, dans un dictionnaire, on associe chaque élément (on parle de valeur dans un dictionnaire) à une clef, on dit qu'un dictionnaire contient des couples clef: valeur (chaque clé est associée à une valeur).

On pourrait, par exemple, associer les valeurs "prénom" et "nom" aux clés "numéros de sécurité sociale", mais la contrainte principale est que chaque clé doit être unique dans le tableau associatif.

Exemples de couples clé: valeur:

prenom:Kevin, nom:Durand, date-naissance:17-05-2005.

prenom, nom et date sont des clés

Kevin, Durand et 17-05-2005 sont des valeurs.

Voici les opérations que l'on peut effectuer sur le type abstrait dictionnaire :

•ajout: on associe une nouvelle valeur à une nouvelle clé

•modif: on modifie un couple clé:valeur en remplaçant la valeur courante par une autre valeur (la clé restant identique)

•suppr: on supprime une clé (et donc la valeur qui lui est associée)

•rech: on recherche une valeur à l'aide de la clé associée à cette valeur.

Exemples:

Soit le dictionnaire D composé des couples clé: valeur suivants

prenom:Kevin, nom:Durand, date-naissance:17-05-2005

Pour chaque exemple ci-dessous on repart du dictionnaire d'origine:

•ajout(D,tel:06060606)

- le dictionnaire D est maintenant composé des couples suivants:
    - prenom: Kevin, nom: Durand, date-naissance: 17-05-2005, tel: 06060606

•modif(D,nom:Dupont)

- le dictionnaire D est maintenant composé des couples suivants:
    - prenom: Kevin, nom: Dupont, date-naissance: 17-05-2005

•suppr(D,date-naissance)

- le dictionnaire D est maintenant composé des couples suivants:
    - prenom: Kevin, nom: Durand

•rech(D,prenom)

- la fonction retourne Kevin

Le type *dict* de Python est une implémentation du type tableau associatif.

Une caractéristique essentielle des dictionnaires est que la récupération d'une valeur associée à une clé se fait en temps constant, indépendant de la taille du dictionnaire.

De même, savoir si une clé fait partie du dictionnaire prend un temps constant (alors que vérifier si un élément est dans une liste prend un temps proportionnel à la taille de la liste).

L'implémentation des dictionnaires correspond à une table de hachage, ce qui signifie que la valeur est stockée dans un tableau et que la position dans ce tableau dépend du résultat d'une fonction de hachage appliquée à la clé.

Comme dit plus haut, en un temps indépendant du nombre de valeurs stockées dans le dictionnaire, Python peut retrouver la valeur associéé à n'importe quelle clé:

pour cela il calcule un indice à partir de la valeur de la clé (ce qui doit donc être hachable, c'est à dire, récursivement non-mutable) et récupère la valeur stockée à cet indice dans un tableau.

Les tables de hachages ainsi que les fonctions de hachages qui sont utilisées pour construire les tables de hachages, ne sont pas au programme de NSI.

Cependant, l'utilisation des fonctions de hachages est omniprésente en informatique, il serait donc bon, pour votre "culture générale informatique", de connaitre le principe des fonctions de hachages.

Voici un texte qui vous permettra de comprendre le principe des fonctions de hachages: c'est quoi le hachage .

Pour avoir quelques idées sur le principe des tables de hachages, je vous recommande le visionnage de cette vidéo: wandida : les tables de hachage

Si vous avez visionné la vidéo de wandida, vous avez déjà compris que l'algorithme de recherche dans une table de hachage a une complexité O(1) (le temps de recherche ne dépend pas du nombre d'éléments présents dans la table de hachage), alors que la complexité de l'algorithme de recherche dans un tableau non trié est O(n).

Comme l'implémentation des dictionnaires s'appuie sur les tables de hachage, on peut dire que l'algorithme de recherche d'un élément dans un dictionnaire a une complexité O(1) alors que l'algorithme de recherche d'un élément dans un tableau non trié a une complexité O(n).

Python propose une implémentation des dictionnaires, nous avons déjà étudié cette implémentation l'année dernière, n'hésitez pas à vous référer à la ressource proposée l'an passé: les dictionnaires en Python

Sources:

- https://pixees.fr/informatiquelycee/n_site/nsi_term_structDo_dico.html
- Prépabac Tle générale spécialité NSI, édition Hatier