Cours 4 : Tables de hachage

Jean-Stéphane Varré

Université Lille 1

jean-stephane.varre@univ-lille1.fr

Des solutions ?

Quelle structure de données utiliser ?

- un dictionnaire Python, oui d'accord mais que connaît-on des complexités des opérations ?
- utiliser un tableau, oui mais :
  - si les données ne sont pas indicées par des entiers ?
  - si les données sont éparpillées ?
- utiliser une liste, oui mais :
  - la recherche est séquentielle !

On dispose de couples de données à ranger pour lesquels on souhaite faire les opérations

- **d'ajout,**
- **de recherche,**
- **(optionnellement de suppression).**

de manière très efficace

c'est-à-dire :

- aussi rapide qu'une liste pour ajouter
- aussi rapide qu'un tableau pour accéder

On dispose de couples de données à ranger pour lesquels on souhaite faire les opérations

- **d'ajout,**
- **de recherche,**
- **(optionnellement de suppression).**

de manière très efficace

c'est-à-dire :

- aussi rapide qu'une liste pour ajouter
- aussi rapide qu'un tableau pour accéder

L'exemple de l'annuaire

- Problème : on souhaite implanter un annuaire qui permette de retrouver facilement le numéro de téléphone en fonction du nom (on suppose qu'il n'y a pas d'homonymes)
- Solution proposée : avoir un tableau indicé par les noms (chaînes de caractères) et y stocker le numéro de téléphone
- Difficulté technique : on se sait pas créer des tableaux indicés par des chaînes de caractères
- Solution proposée : transformer chaque nom en nombre et stocker le numéro de téléphone dans un tableau indicé par ces nombres

$$f(\text{nom}) = ???$$

Une table de hachage est une structure de données dont le cahier des charges est le suivant :

- permet l'association d'une valeur à une clé
  dans l'exemple les valeurs sont des numéros de téléphone et les clés des noms
- permet un accès rapide à la valeur à partir de la clé (comme un tableau)
- permet l'insertion rapide (comme dans une liste)

Remarques

- Inconvénients :
  - si la longueur des noms n'est pas bornée $f(\text{nom})$ peut prendre une infinité de valeurs
  - si la longueur est bornée, le tableau devra être très grand
    pour 10 lettres, 141,167,095,653,376 valeurs possibles ! ce qui est inutile dans la plupart des cas ... et qui de toute façon ne rentre pas dans la mémoire d'un téléphone portable
- Solution :
  - se contenter d'un tableau plus petit, ne pouvant tout contenir, mais de capacité suffisante pour l'usage ciblé
  - appliquer un modulo sur le résultat de la fonction f de manière à avoir des valeurs inférieures à la longueur tableau
- Difficulté : deux noms, même non homonymes, peuvent se voir associer le même entier : il y a collision.

Vocabulaire

- clé : l'objet auquel est associé la valeur
- valeur : l'objet que l'on souhaite stocker
- table : la structure dans laquelle sont rangées les associations <clé,valeur> à des adresses
- alvéole : case qui se trouve à une adresse de la table
- la fonction de hachage : transforme une clé en une adresse dans la table

`http://groups.engin.umd.umich.edu/CIS/course.des/cis350/hashing/WEB/HashApplet.htm`

Dans l'exemple de l'annuaire :

- clé = nom
- valeur = numéro de téléphone
- fonction de hachage :

$$h(k) = f(k) \mod M$$

avec M la capacité de la table

Fonction de hachage

La fonction de hachage permet de transformer une clé en adresse :

- adressage direct :
  - la clé = l'adresse
- adressage indirect :
  - l'adresse est une fonction de la clé
- la fonction doit être simple à calculer, pour rester efficace en temps

La fonction de hachage doit être **uniforme** pour assurer un bon comportement : chaque clé doit avoir la même chance d'être rangée dans chaque alvéole.

Méthode de la division

$$h(k) = f(k) \mod M$$

- le choix de M peut dépendre de la distribution des clés
  si on insère des clés k telles que $f(k)$ est multiple de 10 on n'aura pas intérêt à choisir un M multiple de 10
- en général, choisir un nombre premier pour M donne de bons résultats
- on pourra choisir M en fonction de la performance qu'on souhaite
  si on insère 2000 mots et que l'objectif est d'examiner en moyenne 3 alvéoles dans le cas d'une recherche infructueuse, on pourra choisir $M = 673$.

Dans ce cas, la qualité du hachage dépend de la taille de la table.

Calcul du hachage pour des objets quelconques

- il faut avoir une valeur unique pour chaque objet
- dans certains langages ce calcul est implicite (souvent en utilisant l'adresse mémoire où est rangé l'objet)
- mais attention, deux objets créés identiquement n'ont pas nécessairement même hash code,
  en Java par exemple il est nécessaire de redéfinir la méthode `hashCode`, en Python la méthode `__hash__`
- mais encore attention, il peut aussi être nécessaire de redéfinir l'égalite au sens logique des objets,
  en Java la méthode `equals`, en Python la méthode `__equals__`

(voir cours de POO pour les tables de hachage en Java)

En conclusion, il faut être en capacité

- de **calculer une adresse** à partir de la clé pour ranger
- de **tester l'égalité** entre deux clés pour le prédicat de présence