Recherche rapide

Tables de hachage

Jean-Marc.Vincent@univ-grenoble-alpes.fr[1]

[1] Laboratoire LIG
Équipe-Projet INRIA POLARIS
Université Grenoble-Alpes

Algorithmique et Modélisation
L3-INFO

I. LE PROBLÈME : Recherche d'un élément dans une collection
II. STRUCTURE ADAPTÉE : adressage fermé/ouvert
III. COMPLEXITÉ
IV. UTILISATION : structure auxiliaire
V. FONCTIONS DE HACHAGE
VI. UTILISATION : Sécurité

STRUCTURES DE DONNÉES : COLLECTIONS

Notations

- $\mathcal{E} = \{e_1, e_2, \cdots, e_n\}$ ensemble de n éléments (distincts)
  dossiers des étudiants de L3-INFO
- Chaque élément est référencé par une clé
  numéro de la carte d'étudiant
- Les éléments de $\mathcal{E}$ sont stockés dans une structure `S`
- Les opérations effectuées sur la structure sont
  - `Rechercher(e.cle)`
  - `Inserer(e)`
  - `Supprimer(e.cle)`

Structure	Recherche	Insertion	Suppression
Tableau			
Tableau tassé			
Tableau trié			
Liste chaînée			
ARR			
ABR équilibré			

PRINCIPE

L'accès à l'élément e utilise la valeur de la clé de e

Univers des clés

Structure de stockage (table)

PRINCIPE

L'accès à l'élément *e* utilise la valeur de la clé de *e*

Univers des clés

Structure de stockage (table)

Univers des clés (gigantesque !)
h fonction de hachage

$h(cle)$ = adresse
Taille de la table m taux de remplissage $\frac{m}{n}$

EXERCICE

Prendre le numéro de sa carte d'étudiant, le multiplier par $\pi = 3.141592653589793...$
considérer le premier chiffre et le deuxième chiffre après la virgule

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

ADRESSAGE OUVERT

L'accès à la structure (liste) contenant *e* utilise la valeur de la clé de *e*

Univers des clés

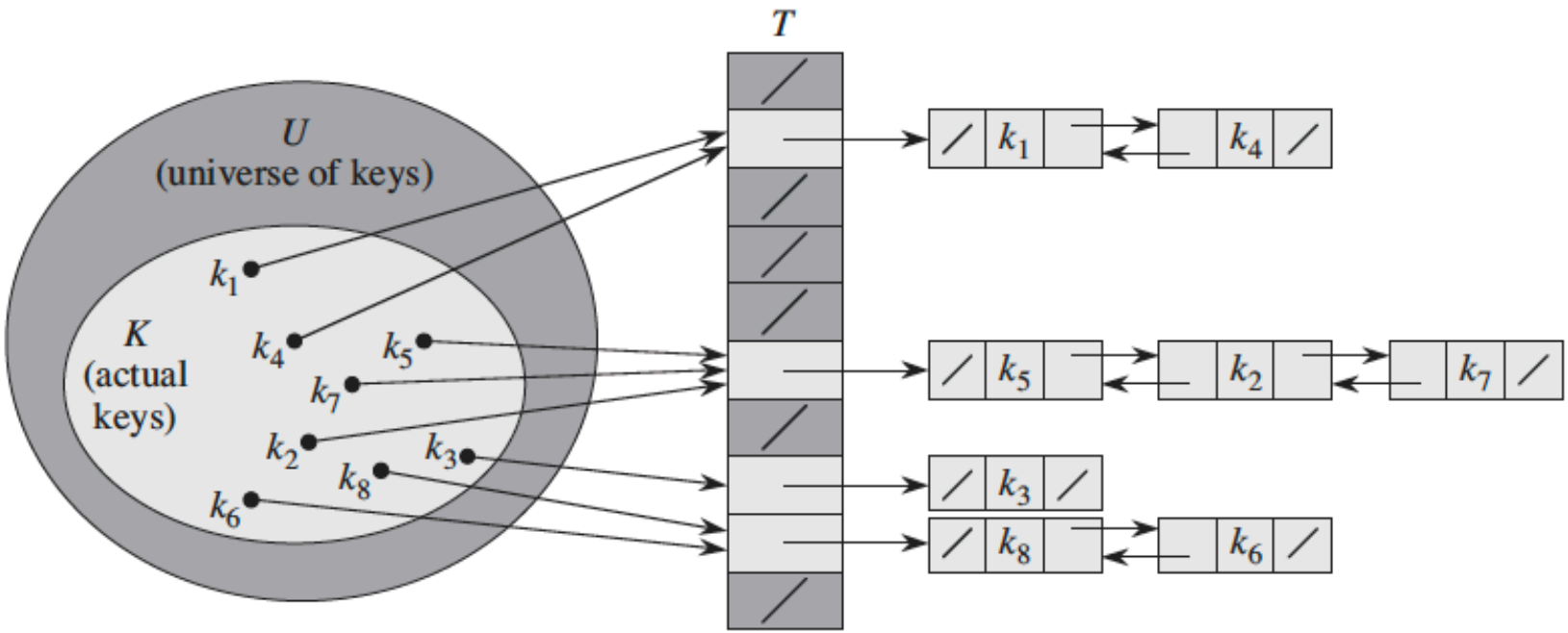
Structure de stockage (table de liste)

ADRESSAGE OUVERT

L'accès à la structure (liste) contenant *e* utilise la valeur de la clé de *e*

Univers des clés

Structure de stockage (table de liste)



Extrait de *Introduction to Algorithms*
Cormen, Leiserson, Rivest, Stein (2009).
(il y a une version française)

h(*cle*) = adresse
consultation par parcours
Chaînage simple/double

ADRESSAGE FERMÉ

L'accès à t *e* utilise la valeur de la clé de *e* et parcourt la structure jusqu'à trouver *e* (sondage)

Univers des clés

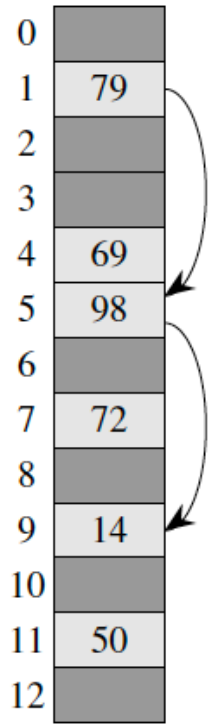
Structure de stockage (table)

ADRESSAGE FERMÉ

L'accès à t *e* utilise la valeur de la clé de *e* et parcourt la structure jusqu'à trouver *e* (sondage)

Univers des clés

Structure de stockage (table)

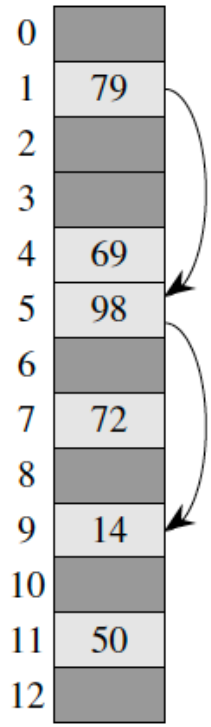


ADRESSAGE FERMÉ

L'accès à t e utilise la valeur de la clé de e et parcourt la structure jusqu'à trouver e (sondage)

Univers des clés

Structure de stockage (table)



$h(cle)$ = adresse,
consultation par parcours

- sondage linéaire $h(k, i) = (h'(k) + i)\ mod\ m$
- sondage quadratique $h(k, i) = (h'(k) + c_1.i + c_2.i^2)\ mod\ m$
- hachage double $h(k, i) = (h_1(k) + i.h_2(k))\ mod\ m$

DIFFÉRENTS SONDAGES

COMPLEXITÉ (PREUVE EN TD)

Sur une structure de table fermée (table de liste)
$\alpha = \frac{n}{m}$ **taux de remplissage de la table**

Recherche

Le coût moyen d'une recherche fructueuse (l'élément est dans la table) ou infructueuse

$$\mathcal{O}(\alpha)$$

en pratique si $m \geqslant n$ la recherche est en temps constant $\mathcal{O}(1)$

Insertion

Le coût moyen d'une insertion (l'élément n'est pas dans la table) est

$$\mathcal{O}(1)$$

insertion en tête de liste

Suppression

Le coût moyen d'une suppression (l'élément est dans la table) correspond au coût de la recherche d'un élément puis de la modification du chaînage

$$\mathcal{O}(\alpha)$$

EXERCICE

Opérations sur les ensembles

On se donne deux tableaux T et U de tailles n et m, on note $N = \max\{m, n\}$. On veut produire un tableau V contenant tous les éléments apparaissant dans T mais pas dans U.

1. Proposer un algorithme de coût $\mathcal{O}(N^2)$ pour effectuer cette opération.
2. On peut améliorer l'opération précédente en commençant par trier les tableaux, calculer la complexité d'une telle solution ?
3. Peut-on calculer le tableau V en coût $\mathcal{O}(N)$? Si oui, écrire l'algorithme.

ALGORITHME SUR LES CHAÎNES DE CARACTÈRE

Algorithme de Rabin-Karp (1987)

SOUS-CHAINE(x,n)

Données: T texte de taille n dans lequel le motif M de taille m est recherché (tableaux de caractères)

Résultat: Les positions du motif M s'il est dans le texte

```
hmotif=h (M[1..m])
for i = 1 to n − m
    hcourant = h (T[i..i+m-1])
    if (hcourant == hmotif)
        if égalité (T[i..i+m-1],M[1..m])
            write i
```

- Calcul de h en $\mathcal{O}(1)$
- $h(A[1..m]) = (a_1 + a_2 + \cdots + a_m)$ modulo p, avec p un entier donné.
- $h_1(A[1..m]) = (a_1.d^{m-1} + a_2.d^{m-2} + \cdots + a_{m-1}.d^1 + a_m.d^0)$ modulo p, avec d un entier donné (base).

FONCTION DE HACHAGE

Spécification

- calcul en $\mathcal{O}(1)$ (lecture de la clé)
- hypothèse d'uniformité et d'indépendance
  difficile à assurer en pratique,
  NE JAMAIS IMPROVISER SA FONCTION DE HACHAGE

Codage

travail sur des nombres
conversion de chaînes de caractères en nombres

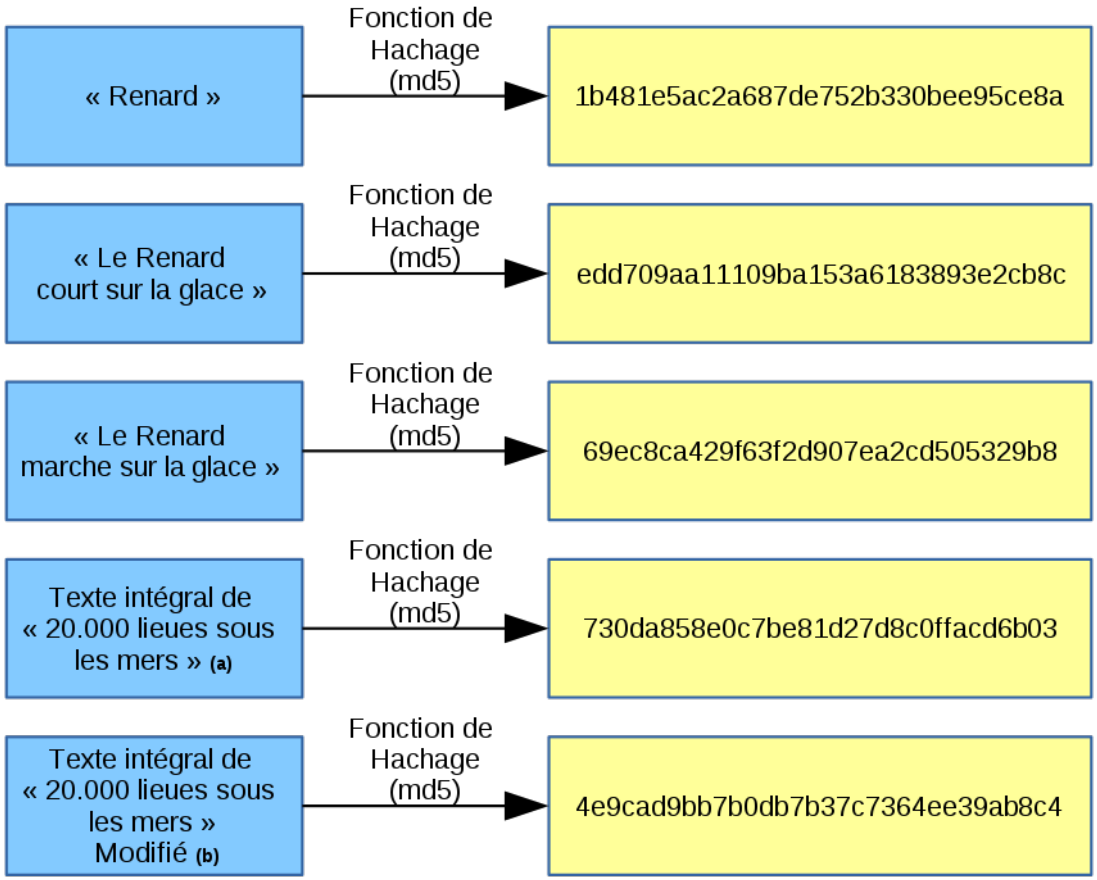
Quelques méthodes

- Méthode de division $h(k) = k mod m$ (faible) choix de m important
- Méthode de multiplication $h(k) = \lfloor M.(A \times k \ mod \ 1) \rfloor$
- Méthodes natives dans le langage de programmation (perl, python, Java,...)

UGA IM²AG UFR Université Grenoble Alpes

LA FONCTION DE HACHAGE MD5

Message Digest *Rivest(1991)* 128 bits (faille grave détectée en 1996)



SECURE HASH ALGORITHM

Un peu de lecture

- SHA-0 (1993) abandon
- SHA-1 (1995) abandon
- SHA-2 (2002) fortement suspecte
- SHA-3 (2012) ? ?

De manière générale, les fonctions de hachage sont utilisées en cryptographie car elles sont très difficiles à inverser et qu'elles devraient avoir une probabilité de collision extrêmement faible

UTILISATIONS DES FONCTIONS DE HACHAGE EN SÉCURITÉ

Les bonnes propriétés de ces fonctions sont utiles au delà de la manipulation de tables de hachage et de dictionnaires :

- **Vérification d'intégrité :** si un fichier est corrompu, le code de hachage de son contenu sera modifié de façon significative, donc fournir le code de hachage d'un fichier avec le contenu permet de détecter les erreurs de transmission.
- **Authentification :** il est essentiellement impossible de retrouver la donnée en connaissant son code de hachage, ainsi si la donnée contient une information secrète (clé de chiffrement), recalculer le code permet de vérifier l'authenticité de la donnée.
- **Identification unique :** Un code de hachage assez long permet en pratique d'identifier une donnée (contenu d'un fichier) de façon unique au moyen d'un identifiant de taille fixe, ce qui permet par exemple d'éliminer les doublons dans une base de données.

Différentes fonctions de hachage standard servent pour ces usages :
MD5, SHA-1, SHA-256...