Algorithmique avancée

Tables de hachage

Frédéric Guyomarch

Université de Lille1
IUT-A de Lille

2017/2018 - Semestre 3

Introduction

Table de hachage

Une table de hachage correspond à un ensemble de valeurs identifiées par des clés et spécifie plusieurs opérations :

- `recherche(K key)` : retourne l'élément de clé key
- `insert(K key, V val)` : (écrit par-dessus si key existe)
- `supprime(K key)`

C'est une implémentation d'un type de données abstrait Dictionnaire (associations clé et valeurs).

Introduction

Table de hachage en résumé

- Une structure de type dictionnaire
- Contenant des couples (clé,valeur)
- La clé identifie la valeur
- Pas de notion notion d'ordre
- Pas de doublon
- Basée sur les tableaux

Introduction

Table de hachage en résumé

- Une structure de type dictionnaire
- Contenant des couples (clé,valeur)
- La clé identifie la valeur
- Pas de notion notion d'ordre
- Pas de doublon
- Basée sur les tableaux

☺Insertion et recherche en temps quasi constant !

Motivations

- Un dictionnaire
- Base de données
- Historiquement pour les compilateurs (table des symboles)
- Vérification de mots de passe
- Maintenir une liste noire d'adresses ip

Application

Déduplication

Entrée : Un flux d'objets
But : Enlever les doublons (ne garder que des objets uniques)
Exemple : Repérer des visiteurs uniques sur une site web

Application

Problème de 2-Sum

Entrée : Un tableau d'entiers non trié T, une valeur cible z.
But : Déterminer s'il existe deux entiers x et y dans T tel que $x + y = z$.

Table de hachage : Principe général

Des couples (clé,valeur) stockés dans un tableau

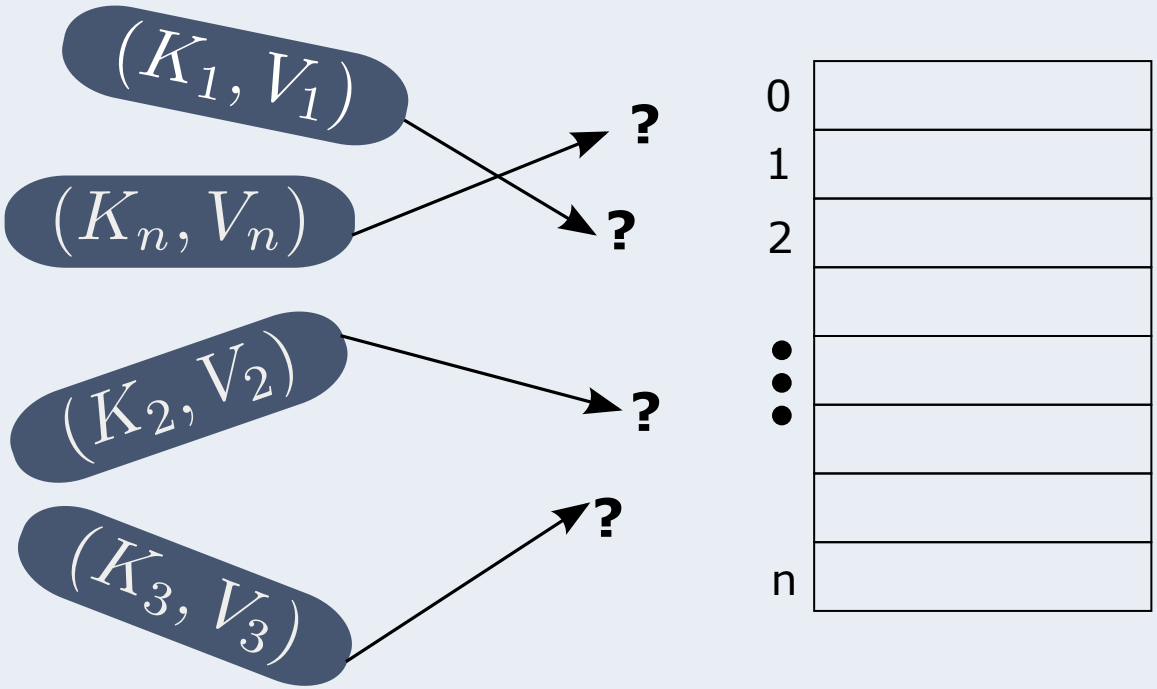
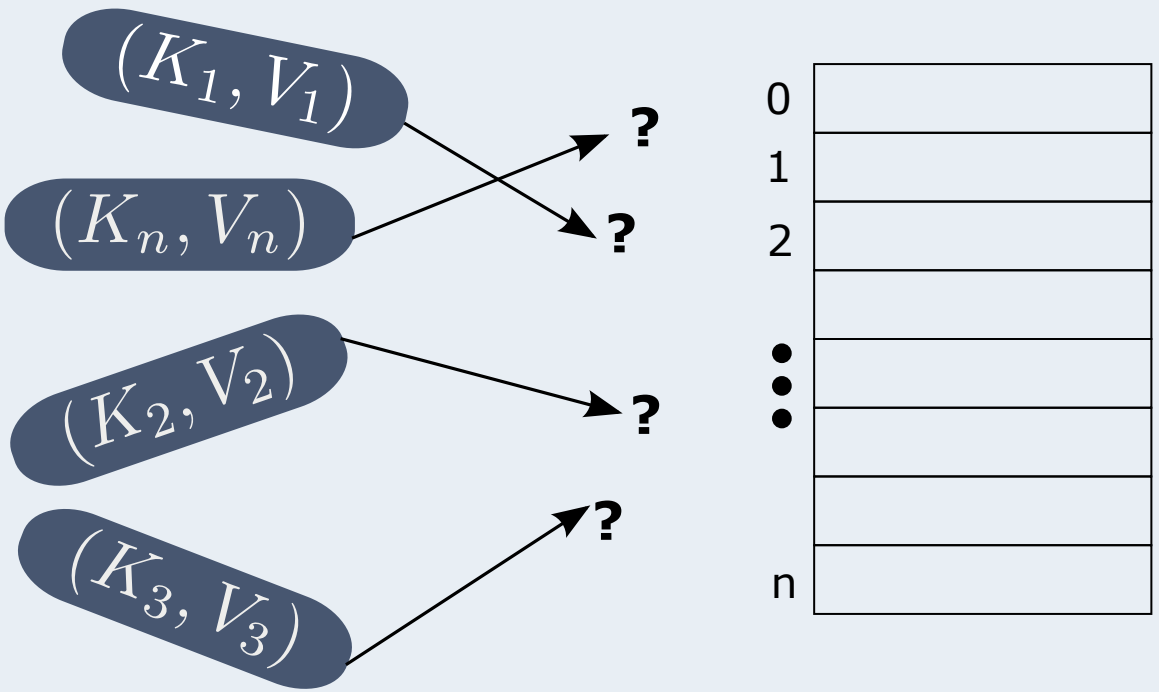


Table de hachage : Principe général

Des couples (clé,valeur) stockés dans un tableau



$\Rightarrow$ une association clé du couple/indice du tableau.

Table de hachage : Principe général

Problématique

Comment ranger les éléments dans le tableau de manière à avoir une recherche en accès direct ?

Table de hachage : Principe général

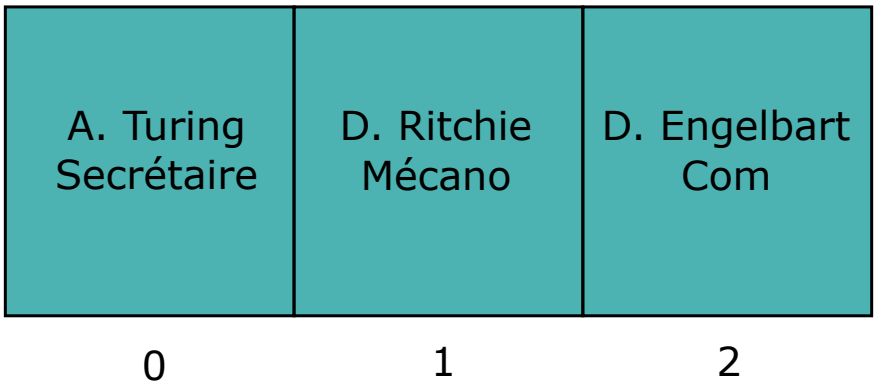
Problématique

Comment ranger les éléments dans le tableau de manière à avoir une recherche en accès direct ?

- Facile a priori si ma clé est un indice (idem tableau)

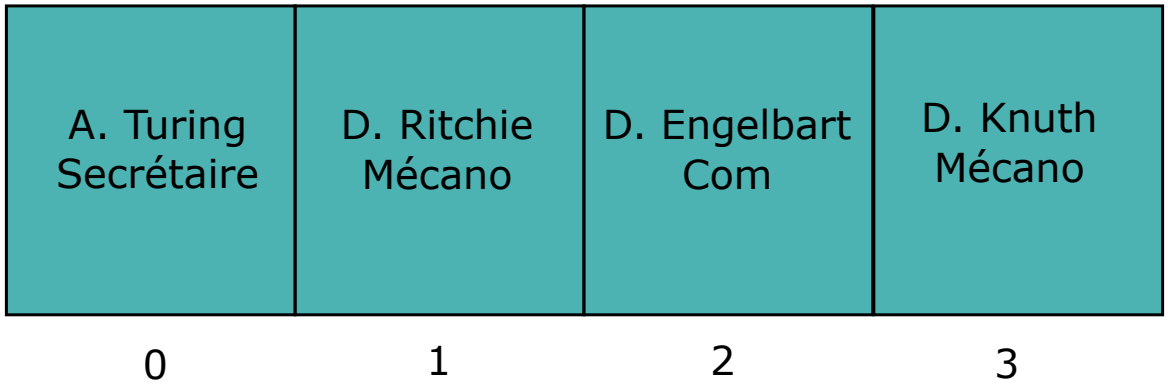
Exemple

Exemple : Un registre d'employés



Exemple

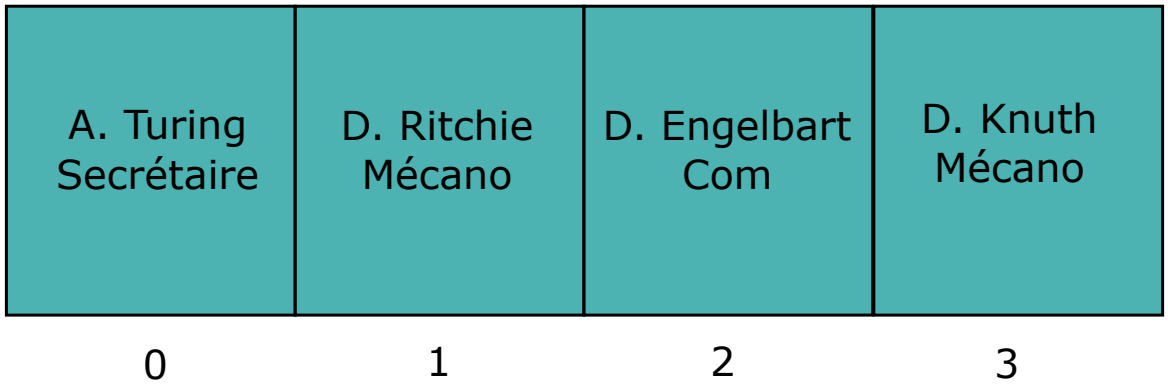
Exemple : Un registre d'employés



A chaque embauche, on incrémente la clé.

Exemple

Exemple : Un registre d'employés

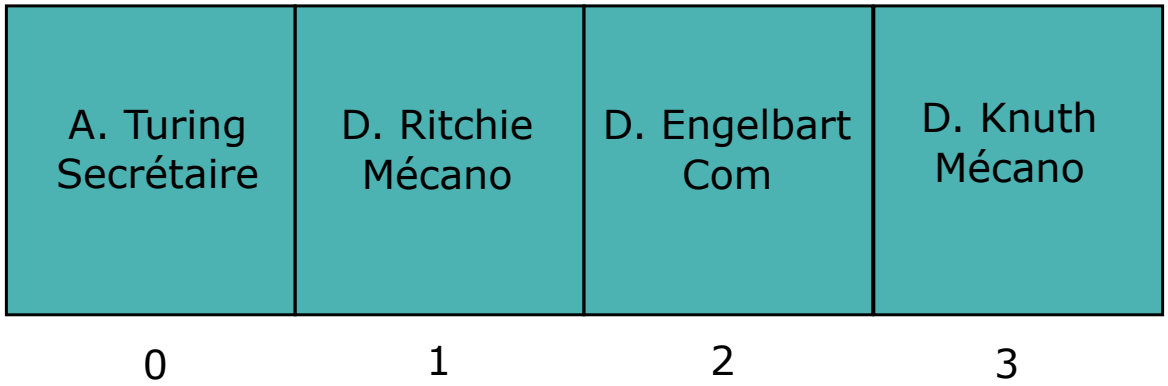


A chaque embauche, on incrémente la clé.

- La clé correspond à l'indice
- Les clés sont en séquences

Exemple

Exemple : Un registre d'employés



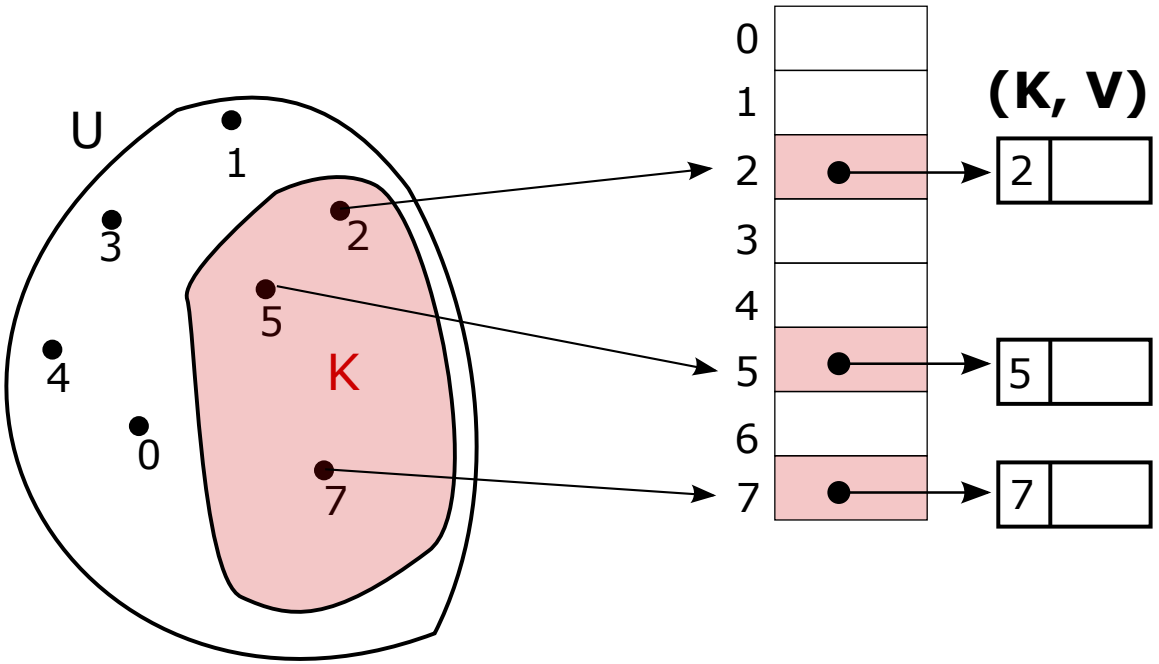
A chaque embauche, on incrémente la clé.

- La clé correspond à l'indice
- Les clés sont en séquences

$\Rightarrow$ Situation idéale

Adressage direct

Formellement, dans le cas général où les clés ne sont pas nécessairement en séquence...



- L'univers des clés possibles est U
- K est l'ensemble des clés présentes ($K \subset U$)

Problème 1 !

Problème 1 !

Si mon univers est très grand...

Problème 1 !

Si mon univers est très grand...

...par rapport au nombre de clés potentielles

Problème 1 !

Si mon univers est très grand...

...par rapport au nombre de clés potentielles

Gaspillage de mémoire énorme !

Exemple : Si j'ai des clés bornées entre 0 et 10000 mais a priori jamais plus de 50 valeurs dans ma table : 9950 cases vides.

Problème 2 !

Et si ma clé est une chaîne de caractères ? ou un objet ?

Problème 2 !

Et si ma clé est une chaîne de caractères ? ou un objet ?
Solution : utilisation d'une fonction de hachage

Fonction de hachage

Définition

Une fonction de hachage est une fonction qui transforme une donnée (la clé) en un entier (l'indice dans le tableau).

Formellement : $h(k) = i$

Fonction de hachage

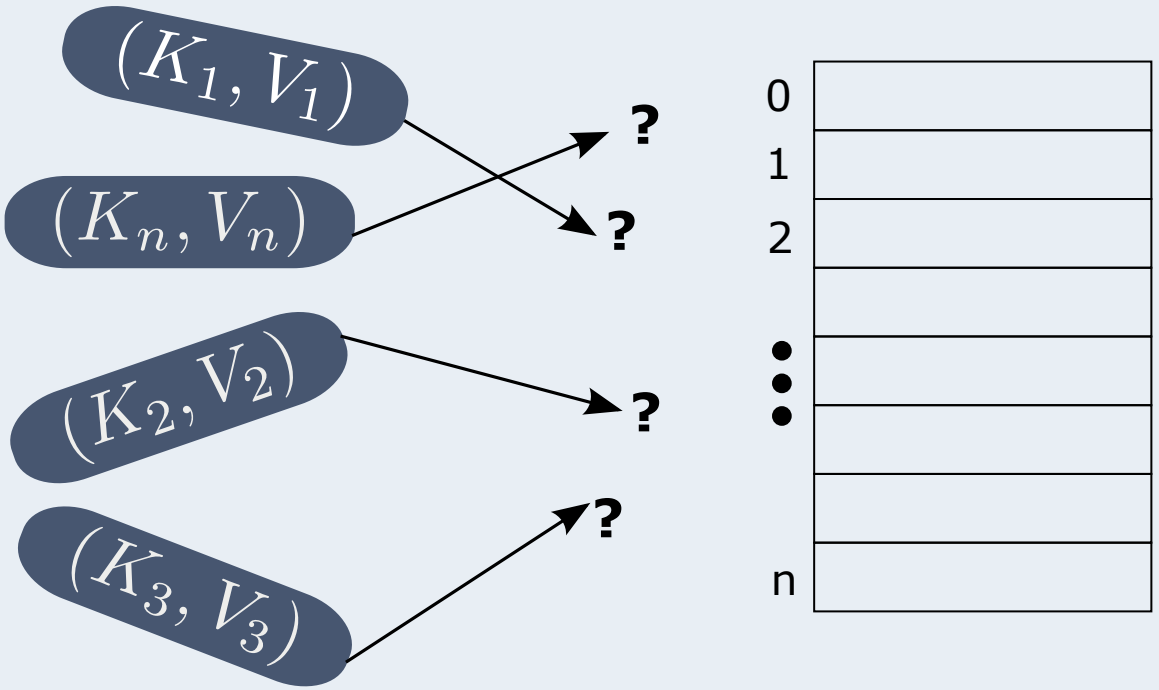
Définition

Une fonction de hachage est une fonction qui transforme une donnée (la clé) en un entier (l'indice dans le tableau).

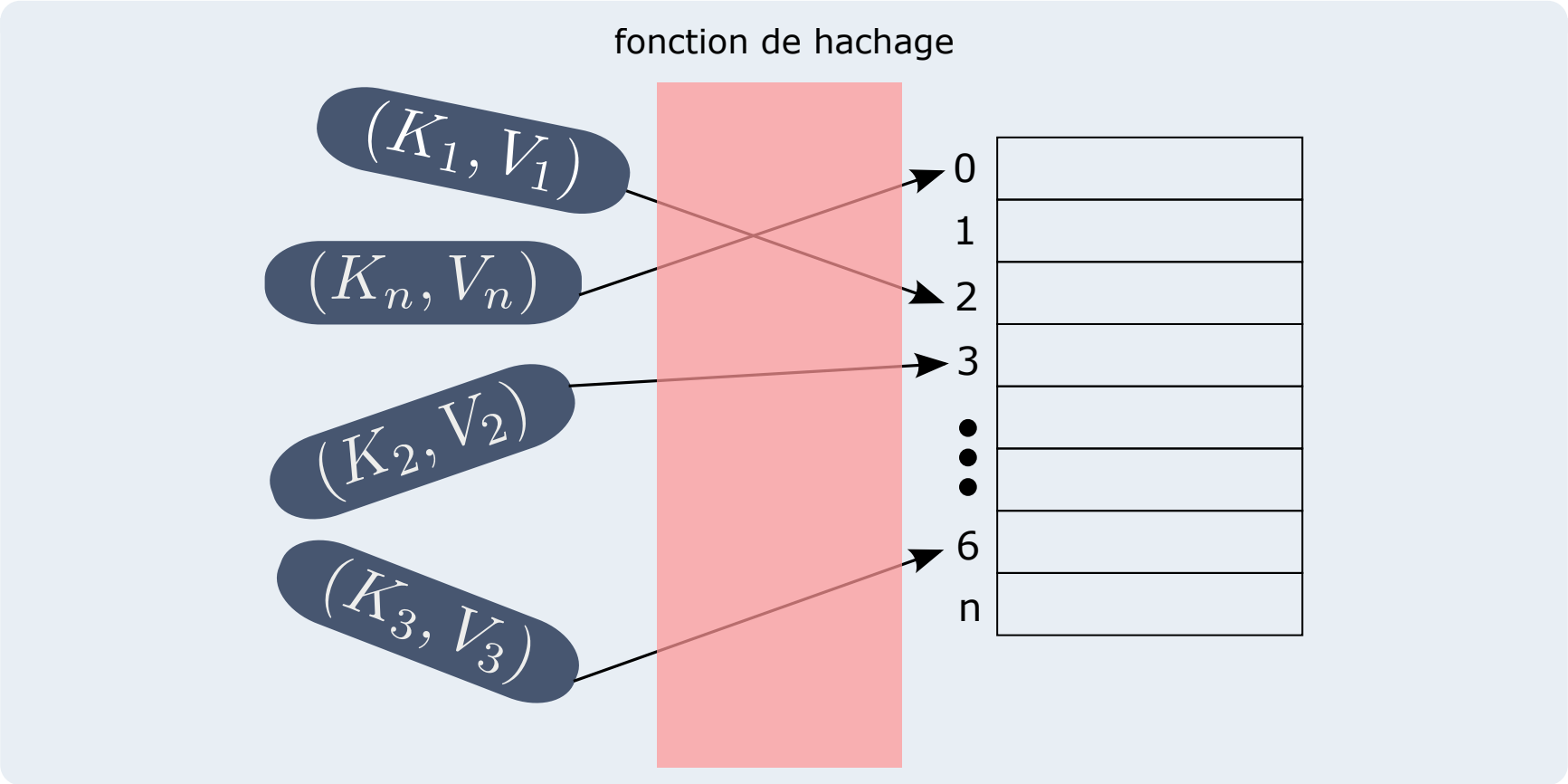
Formellement : $h(k) = i$
Elle établit une correspondance entre un couple(clé,valeur) et un indice du tableau. Le couple se trouvera à $T[i]$.

Fonction de hachage

Des couples (clé,valeur) stockés dans un tableau



Fonction de hachage

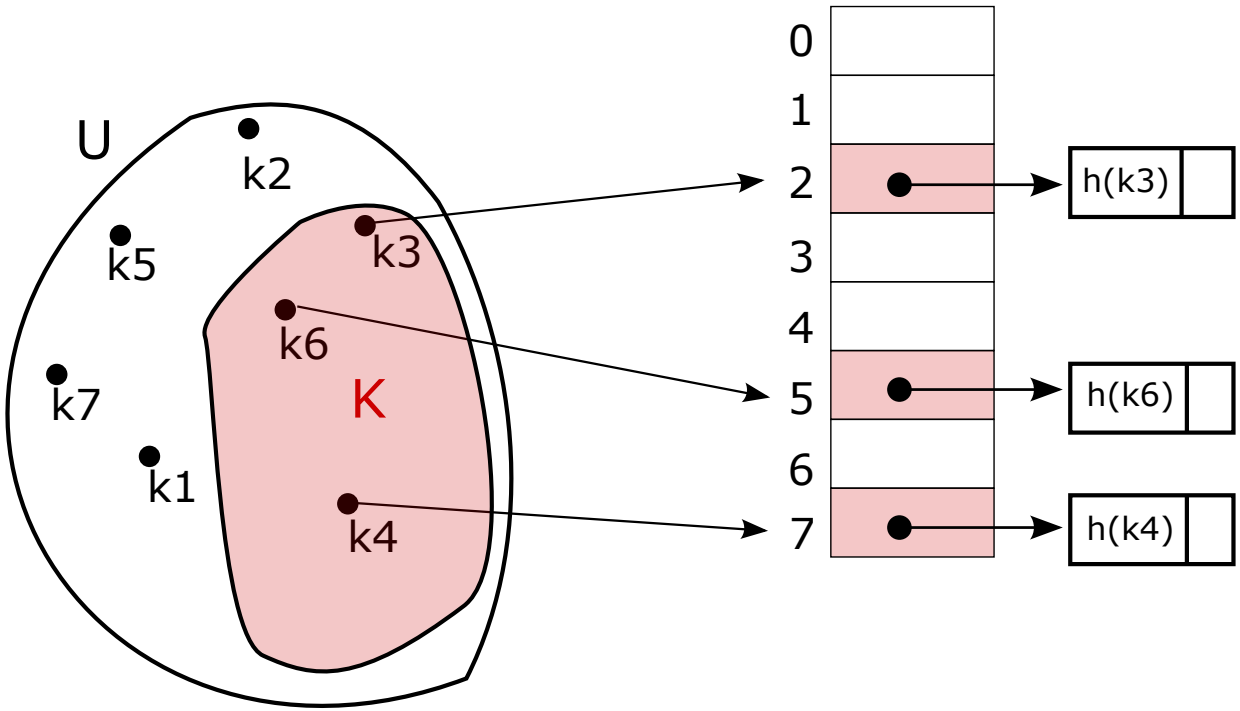


Fonction de hachage

Choix d'une bonne fonction de hachage :

- Elle doit être cohérente : la même clé produit la même valeur de hachage
- Elle doit être calculée efficacement
- Elle doit distribuer uniformément les clés

Adressage direct avec fonction de hachage



Avec h(k3) = 2, h(k4) = 7 et h(k6) = 5.

Exemple de fonction de hachage

Dictionnaire anglais-français

Convertir un mot anglais (la clé) en indice :

- Utiliser la table ascii ?

Exemple de fonction de hachage

Dictionnaire anglais-français

Convertir un mot anglais (la clé) en indice :

- Utiliser la table ascii ? **NON !**

Exemple de fonction de hachage

Dictionnaire anglais-français

Convertir un mot anglais (la clé) en indice :

- Utiliser la table ascii ? **NON !**
- Coder une lettre par un nombre entre 0 et 27 ?

Exemple de fonction de hachage

Dictionnaire anglais-français

Convertir un mot anglais (la clé) en indice :

- Utiliser la table ascii ? **NON !**
- Coder une lettre par un nombre entre 0 et 27 ? **NON !**

Exemple de fonction de hachage

Dictionnaire anglais-français

Convertir un mot anglais (la clé) en indice :

- Utiliser la table ascii ? **NON !**
- Coder une lettre par un nombre entre 0 et 27 ? **NON !**
- Utiliser une base 27 ?

Exemple de fonction de hachage

Dictionnaire anglais-français

Convertir un mot anglais (la clé) en indice :

- Utiliser la table ascii ? **NON !**
- Coder une lettre par un nombre entre 0 et 27 ? **NON !**
- Utiliser une base 27 ? **C'est mieux !**

Exemple de fonction de hachage

Dictionnaire anglais-français

Convertir un mot anglais (la clé) en indice :

- Utiliser la table ascii ? **NON !**
- Coder une lettre par un nombre entre 0 et 27 ? **NON !**
- Utiliser une base 27 ? **C'est mieux !**

On a résolu le problème 2, mais pas le 1 !

Exemple

Dictionnaire anglais-français

On va avoir un tableau de taille faramineuse et beaucoup de cases réservées pour des mots qui n'existent pas !

clé	fira	firb	firc	ird	**fire**	firf
indice	125146	125147	125148	125149	125150	125151

Hachage

- Quelle taille de tableau souhaite-t-on avoir ?

Hachage

- Quelle taille de tableau souhaite-t-on avoir ?
- Idéalement, une taille proche du nombre d'éléments qu'il y aura effectivement dedans.

Hachage

- Quelle taille de tableau souhaite-t-on avoir ?
- Idéalement, une taille proche du nombre d'éléments qu'il y aura effectivement dedans.
- Comment compresser les valeurs ?

Hachage

- Quelle taille de tableau souhaite-t-on avoir ?
- Idéalement, une taille proche du nombre d'éléments qu'il y aura effectivement dedans.
- Comment compresser les valeurs ?

$\Rightarrow$ **Utiliser le modulo**

La méthode de la division

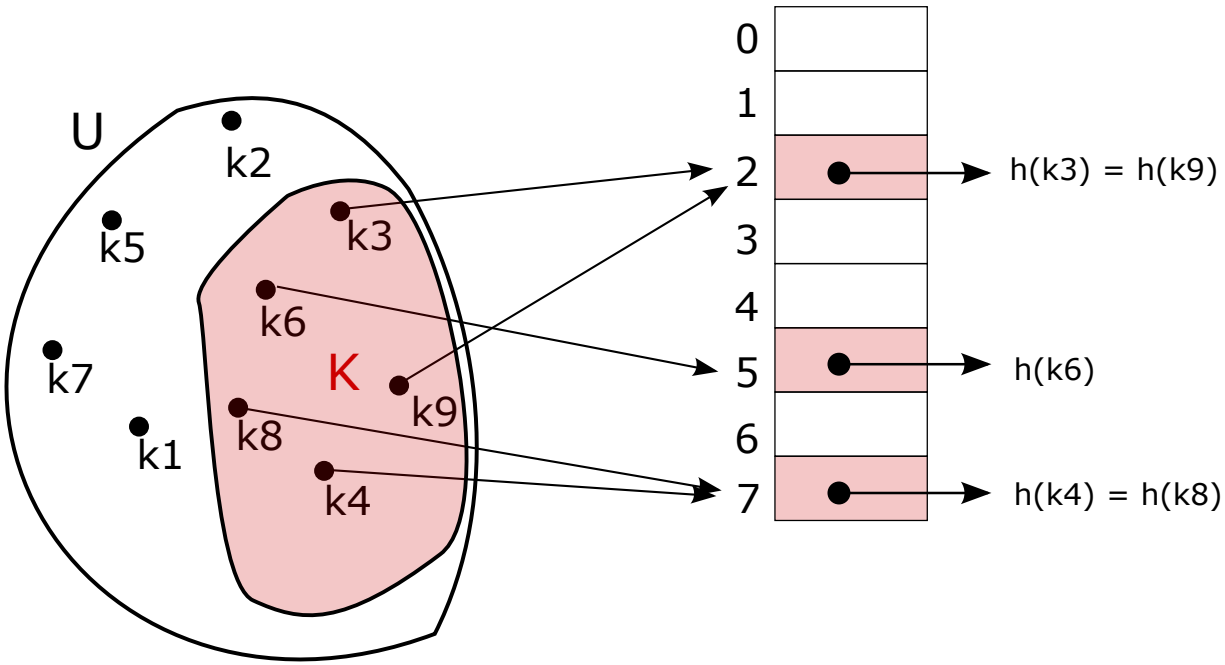
$h(k) = k$ mod m correspond à *la méthode de la division*.

- Méthode simple (rapide)
- Il faut bien choisir la taille m de la table. (nb premier de préférence)

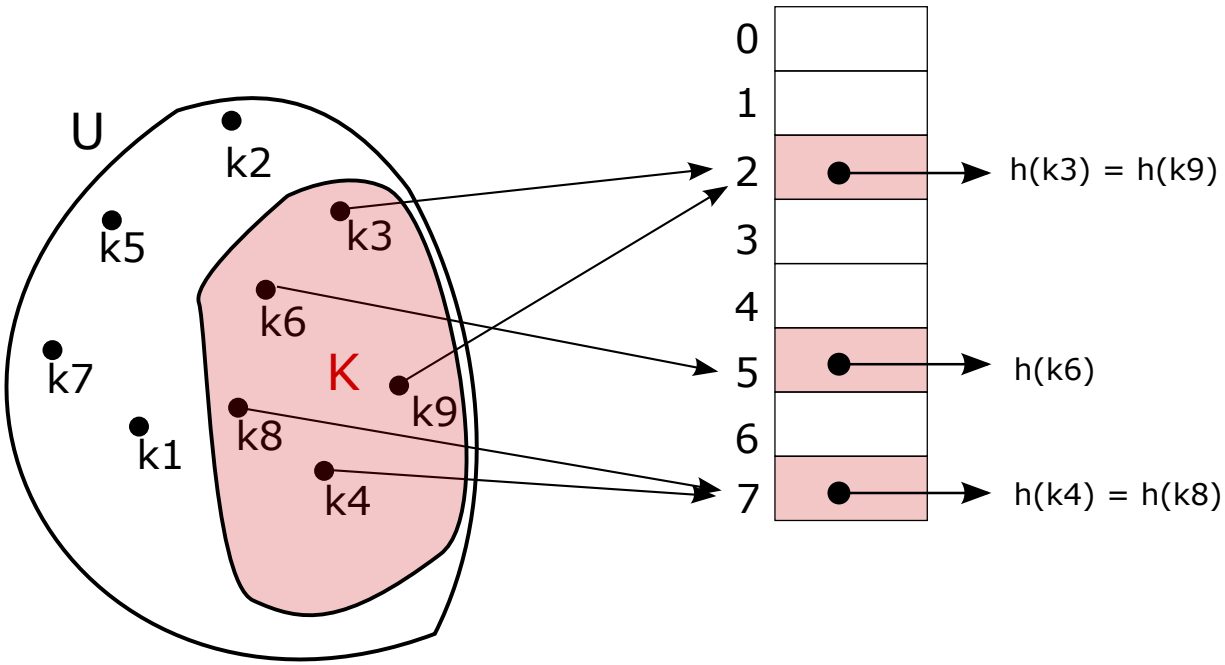
$U = 2000$ et $m = 701$, une moyenne de 3 échecs.

▷ **Il existe d'autres méthodes...**

Hachage et modulo



Hachage et modulo



Plusieurs clés correspondent au même indice dans le tableau : c'est une **collision**.

Collisions

Pourquoi y a-t-il des collisions ?

- Cas évident : principe des tiroirs
  - Si n chaussettes et m tiroirs et $n > m$
    $\Rightarrow$ Forcément plus d'une chaussette par tiroir
- Si $n < m$, risque de collision tout de même
  - $\Rightarrow$ paradoxe des anniversaires
    $\Rightarrow$ Dans un groupe de 23 personnes, 50% de chance qu'il y ait deux personnes avec la même date de naissance.

Résoudre les collisions

- Par adressage ouvert
- Par chaînage

Principe

En adressage ouvert, quand un élément ne peut être placé à l'endroit calculé par hachage, on essaie de le placer dans une autre case vide.

Nous allons étudier 3 méthodes :

- Le sondage linéaire
- Le sondage quadratique
- Le hachage double

Sondage linéaire

Soit $i = h(k)$:

1. On accède à $T[i]$ par le retour de la fonction de hachage
2. On regarde si la clé n'est pas dans la case $h(k) + 1$:
3. **En cas d'insertion** : on ajoute dans la première case vide trouvée
   En cas de recherche : Tant qu'on a pas trouvé la valeur on répète 2.
   Si on trouve une case vide, la donnée n'existe pas.

On regarde aux indices $h(k) + 2, h(k) + 3, h(k) + 4...$

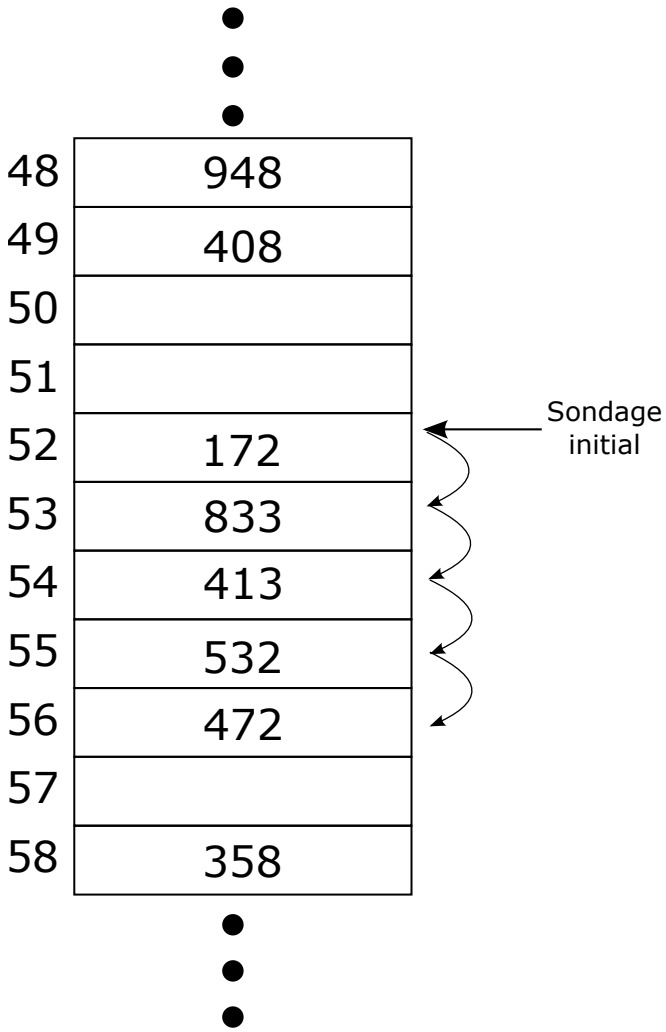
Sondage linéaire

Soit $i = h(k)$:

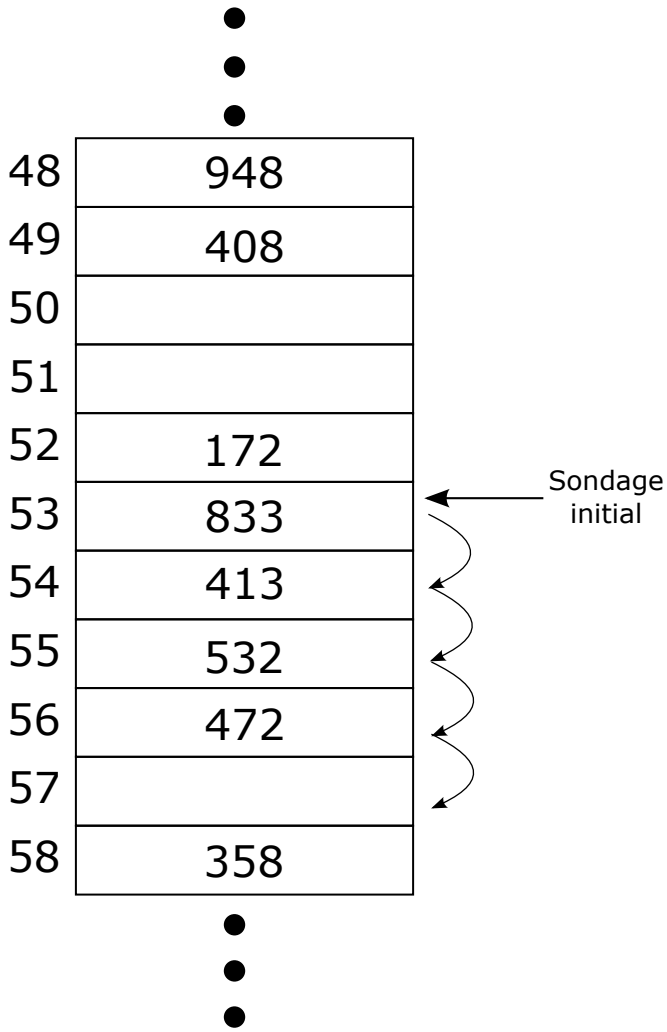
1. On accède à $T[i]$ par le retour de la fonction de hachage
2. On regarde si la clé n'est pas dans la case $h(k) + 1$:
3. **En cas d'insertion** : on ajoute dans la première case vide trouvée
   En cas de recherche : Tant qu'on a pas trouvé la valeur on répète 2.
   Si on trouve une case vide, la donnée n'existe pas.

On regarde aux indices $h(k) + 2, h(k) + 3, h(k) + 4...$

☹Problème de clustering (pire si la table est trop remplie).



a)Recherche réussie de 472



b)Recherche infructueuse de 893

Sondage quadratique

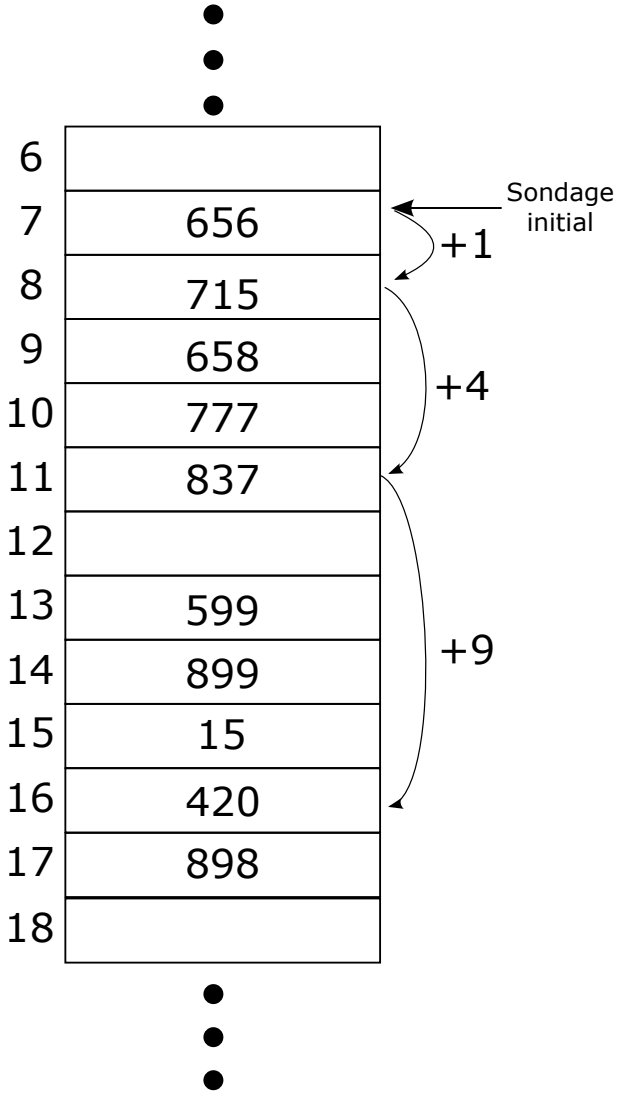
- On ne regarde plus juste à la case suivante.
- On applique une fonction quadratique à pour passer d'une case à l'autre

On regarde aux indices $h + 1^2, h + 2^2, h + 3^2, h + 4^2, ...h + i^2...$ ou plus généralement aux indices $h(k) + \alpha i + \beta i^2$.

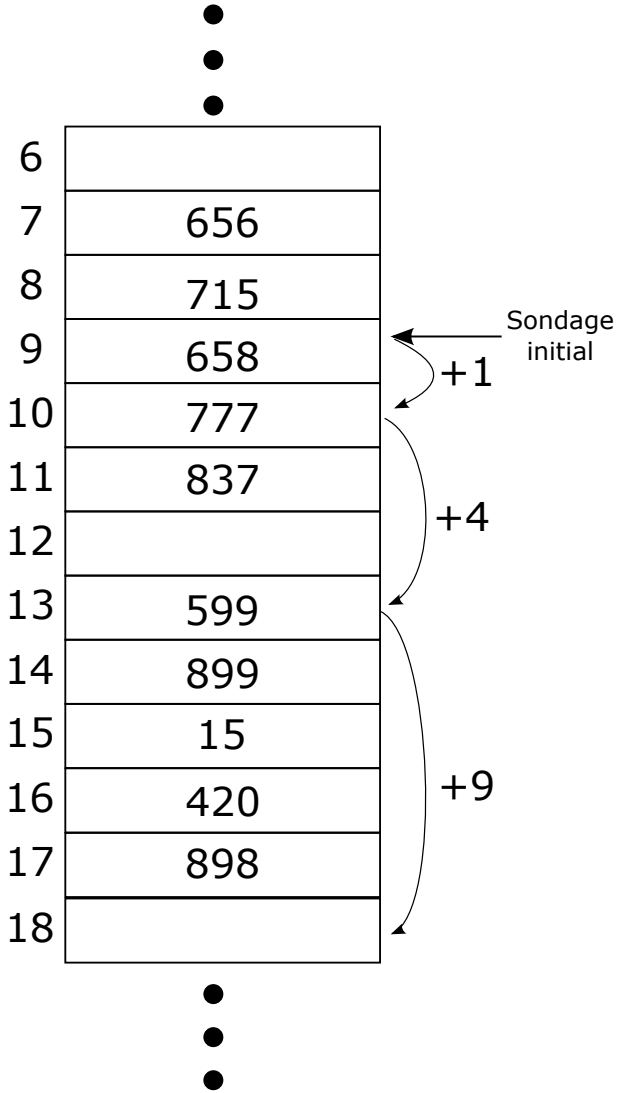
Sondage quadratique

- On ne regarde plus juste à la case suivante.
- On applique une fonction quadratique à pour passer d'une case à l'autre

On regarde aux indices $h + 1^2, h + 2^2, h + 3^2, h + 4^2, ...h + i^2...$
ou plus généralement aux indices $h(k) + \alpha i + \beta i^2$.
☹Problème de clustering secondaire.



a)Recherche réussie de 420



b)Recherche infructueuse de 480

Pour que le sondage soit efficace :

- Le facteur de charge doit être limité (pas plus 2/3)
  $facteurdecharge = nbElements / tailleTable$
- On ne peut avoir plus d'éléments que de cases du tableau

Double Hachage

Les sondages linéaire et quadratique posent des problèmes de clustering car on suit toujours la même séquence.

Double Hachage

Les sondages linéaire et quadratique posent des problèmes de clustering car on suit toujours la même séquence.

Une solution : Le double hachage

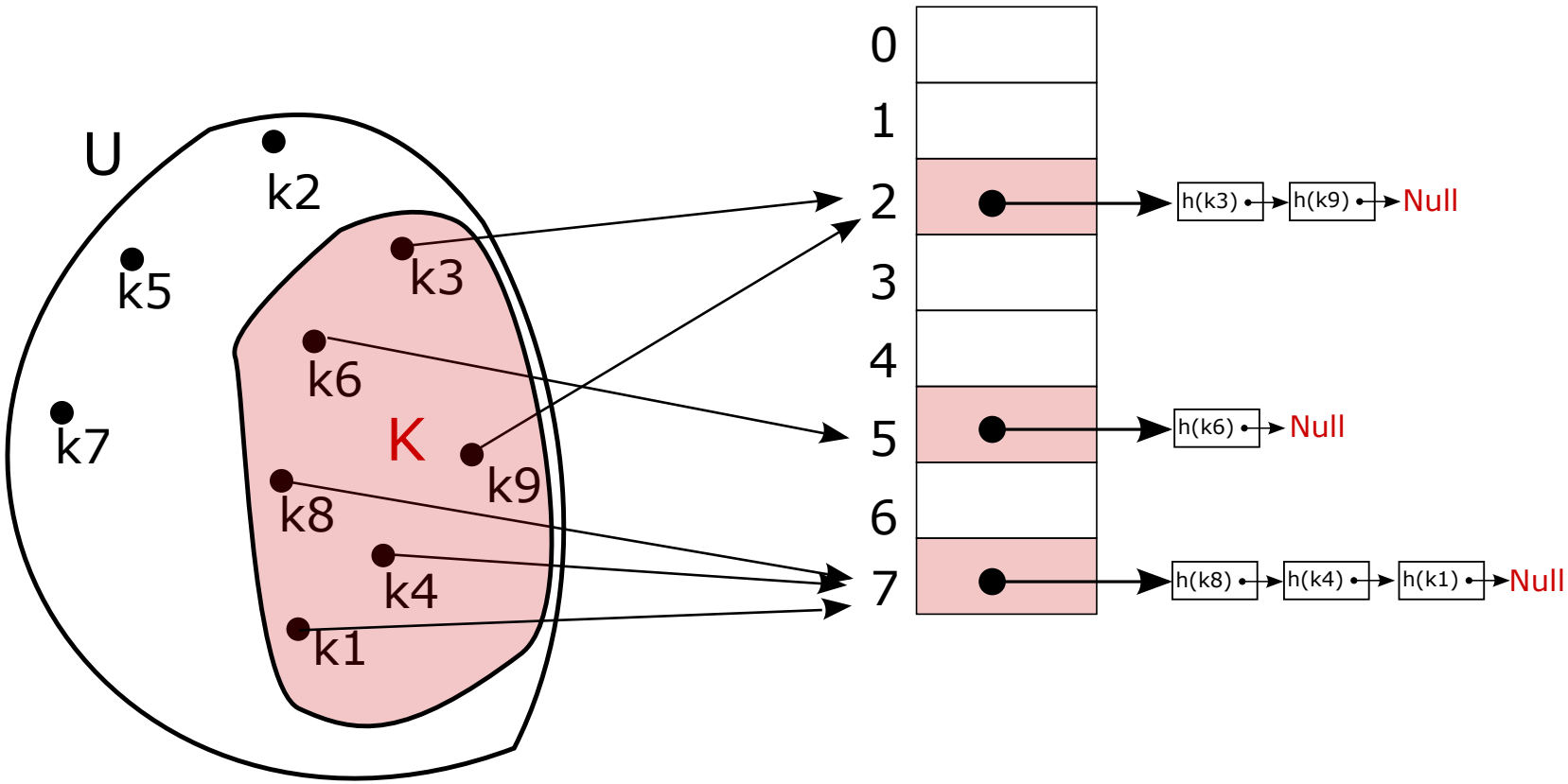
Principe

On va utiliser une seconde fonction de hachage sur les deux clés qui va déterminer un pas (probablement) différent.

Jusque qu'ici, au maximum autant d'objets que la taille de la table.

Principe

On va ranger tous les couples dont les clés ont la même valeur de hachage dans une liste chaînée


U
k2
k5
k3
k6
K
k7
k9
k8
k4
k1
0
1
2
3
4
5
6
7
h(k3)
h(k9)
Null
h(k6)
Null
h(k8)
h(k4)
h(k1)
Null

Nous utiliserons le plus souvent la classe `HashMap` de l'interface `Map`.
Exemple d'utilisation :

```
public static void main(String[] args) {
    Map <String ,Integer> iMap
    = new HashMap<String ,Integer >();
    iMap.put("one", new Integer(1));
    iMap.put("two",new Integer(3));
    iMap.put(null, new Integer(45));
    iMap.put(null, new Integer(32));

    Integer i = iMap.get("three"); //Retourne null
    Integer j = iMap.get("one"); // j = 1
    System.out.println(iMap.get(null)); //affiche 32
}
```

Egalité d'objets

L'opérateur == compare l'égalité des références des objets et la méthode `equals()` leur égalité tout court.

- Mais par défaut `equals()` se base sur les références.
- Il faut donc la redéfinir !
- Sinon deux instances aux mêmes propriétés seront considérées comme différentes !

La méthode `hashcode`

En Java, la valeur de la fonction de hachage est implicite à chaque objet :

- La méthode `hashcode` de `Object` renvoie la valeur de hachage.
- Par défaut, elle fait référence à l'emplacement mémoire comme `equals`.

Si a.equals(b) => a.hashcode() == b.hashcode
Mais la réciproque est fausse !

La méthode `hashcode`

En Java, la valeur de la fonction de hachage est implicite à chaque objet :

- La méthode `hashcode` de `Object` renvoie la valeur de hachage.
- Par défaut, elle fait référence à l'emplacement mémoire comme `equals`.

Si a.equals(b) => a.hashcode() == b.hashcode
Mais la réciproque est fausse !

Il est donc important de redéfinir conjointement ces méthodes !

Set

Les Set sont des ensembles d'objets :

- Uniquement des objets (pas de couple(clé,valeur))
- Pas de doublon
- Pas d'ordre
- Relatif à la notion d'ensemble en mathématiques.

En Java, l'implémentation de l'interface `Set` `HashSet` est basée sur l'utilisation d'une `HashMap`.

```
HashSet<String> etatsSet = new HashSet<String>();
etatsSet.add("FR");
etatsSet.add("AG");
etatsSet.add("CU");

if(etatsSet.contains("MX")) {
   System.out.println("Deja trouve");
}else{
   etatsSet.add("MX");
}
```

Entry(K,V) de Map

- La méthode `Set< Map.Entry<K,V> > entrySet()` retourne un Set de Map.Entry<K,V> qui correspond à un couple (clé,valeur).
- Le `Set` retourné correspond à une vue sur la `Map` à l'instant t.
- On peut appeler les méthodes `getKey()` and `getValue()` pour accéder respectivement aux clés et aux valeurs.

```
public static void main(String[] args) {
    Map<String,Integer> iMap =
    new HashMap<String,Integer>();
    iMap.put("one", new Integer(1));
    iMap.put("two",new Integer(3));
    iMap.put(null, new Integer(45));
    iMap.put(null, new Integer(32));

    Set<Entry<String, Integer>>  set = iMap.entrySet();
    System.out.println("valeur d'un set: " + set);
    //valeur d'un set: [null=32, two=3, one=1]
}
```

Au delà du cours : Empreinte MD5

Empreinte MD5 : fonction de hachage cryptographique calculée sur un fichier.

$\Rightarrow$ Permet de vérifier l'intégrité d'un fichier.

Scénario exemple :

- Je télécharge une version d'ubuntu
- Je veux être certain que mon fichier téléchargé n'est pas corrompu ou vérolé
- Je récupère l'empreinte MD5 sur le site de source officielle
- Je calcule cette empreinte avec un utilitaire comme `md5sum`
- Je compare cette empreinte avec l'officielle
- Si elles correspondent, je peux faire confiance à mon fichier.

Dans ce cours, nous avons vu :

- Le fonctionnement des tables de hachages,
- La résolution des collisions,
- Comment on les utilise en pratique (c'est simple) !