Introduction aux tables de hachage

Hugo Demaret

August 2022

hugo.demaret@polytechnique.edu

0 Plan

0 Notes de cours

Ce cours est accompagné d'exercices et de notes de cours :

- Sur le site de Nicolas Loménie (Cours d'algo avancée)
- Sur mon site : `https://hugodemaret.fr/teachings`

Vous retrouverez aussi ces slides sur mon site.

1 Problème du tableau associatif

Problème : Comment représenter une collection de paires clef-valeur ?

1 Problème du tableau associatif

Problème : Comment représenter une collection de paires clef-valeur ?
Exemple : Un groupe de personnes, dont on connait les noms et l'âge respectif.

Problème du tableau associatif

Problème : Comment représenter une collection de paires clef-valeur ?
Exemple : Un groupe de personnes, dont on connait les noms et l'âge respectif.

Definition (Tableau associatif)

Soit K l'ensemble des clefs, V l'ensemble des valeurs. Soit T un tableau. Un tableau associatif est l'association entre une fonction Φ et T, tel que $\Phi : K \times V \rightarrow T$.

Problème du tableau associatif

Plusieurs méthodes :

- ~~Tout mettre dans une liste~~ : pas efficace !
- Structures arborescentes (ABR, AVL, B-Tree, Trie...)
- **Tables de hachage**

Problème du tableau associatif

Plusieurs méthodes :

- ~~Tout mettre dans une liste~~ : pas efficace !
- Structures arborescentes (ABR, AVL, B-Tree, Trie...)
- **Tables de hachage**

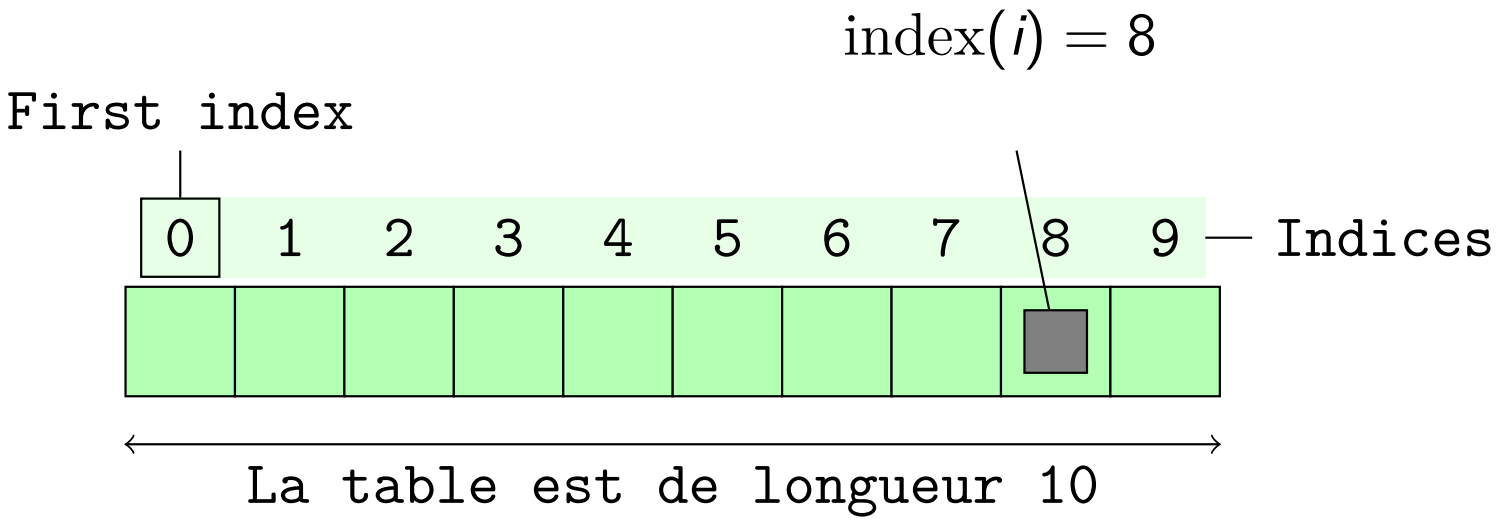
Plusieurs méthodes :

- ~~Tout mettre dans une liste~~ : pas efficace !
- Structures arborescentes (ABR, AVL, B-Tree, Trie...)
- Tables de hachage

Plusieurs méthodes :

- ~~Tout mettre dans une liste~~ : pas efficace !
- Structures arborescentes (ABR, AVL, B-Tree, Trie...)
- **Tables de hachage**

Un tableau T, et une fonction index, qui donne l'indice d'une clef (unique) dans le tableau.



Facteur de charge

On appelle facteur de charge le ratio du nombre d'éléments dans la table sur la taille de la table.

Definition (Facteur de charge)

Soit T une table de taille n. Si T contient m valeurs, le facteur de charge de T, noté α_T, est égal à :

$$\alpha_T = \frac{m}{n}$$

Utilité

Structure de données efficace pour :

- tester l'appartenance d'une clef
- récupérer une donnée

Utilité

Structure de données efficace pour :

- tester l'appartenance d'une clef
- récupérer une donnée

Utilité

Structure de données efficace pour :

- tester l'appartenance d'une clef
- récupérer une donnée

Utilité

Très utilisée en pratique :

- environ 10% de la charge serveur chez Google
- implémentation de graphes, arbres...
- implémentation de structures ensemblistes
- bases de données !
- compilateurs
- un peu partout en fait

Utilité

Très utilisée en pratique :

- environ 10% de la charge serveur chez Google
- implémentation de graphes, arbres...
- implémentation de structures ensemblistes
- bases de données !
- compilateurs
- un peu partout en fait

Utilité

Très utilisée en pratique :

- environ 10% de la charge serveur chez Google
- implémentation de graphes, arbres...
- implémentation de structures ensemblistes
- bases de données !
- compilateurs
- un peu partout en fait

Utilité

Très utilisée en pratique :

- environ 10% de la charge serveur chez Google
- implémentation de graphes, arbres...
- implémentation de structures ensemblistes
- bases de données !
- compilateurs
- un peu partout en fait

Utilité

Très utilisée en pratique :

- environ 10% de la charge serveur chez Google
- implémentation de graphes, arbres...
- implémentation de structures ensemblistes
- bases de données !
- compilateurs
- un peu partout en fait

Utilité

Très utilisée en pratique :

- environ 10% de la charge serveur chez Google
- implémentation de graphes, arbres...
- implémentation de structures ensemblistes
- bases de données !
- compilateurs
- un peu partout en fait

Utilité

Très utilisée en pratique :

- environ 10% de la charge serveur chez Google
- implémentation de graphes, arbres...
- implémentation de structures ensemblistes
- bases de données !
- compilateurs
- un peu partout en fait

Hachage ?

Definition (Fonction de hachage (générale))

Soit $F = \{0, 1, ..., n-1\}$. Soit E un ensemble (de clefs par exemple). La fonction de hachage $\mathrm{hash} : E \to F$ associe à un élément $e \in E$ un entier dans F.

Exemple d'utilisation : en Java, la méthode `hashCode` spécifie comment calculer le hash d'un objet.

Fonction d'indice

Où insérer la clef dans la table ?
On utilise une fonction d'indice !

Definition (Fonction d'indice)

Soit $T = \{0, 1, ..., n-1\}$ un tableau. Soit K un ensemble de clefs. La fonction d'indice $\mathrm{index} : K \to T$ associe une clef à un indice dans le tableau.

Remarque : la fonction d'indice est une fonction de hachage !

En pratique ?

Plusieurs familles de fonctions de hachage :

- Hachage par multiplication
- **Hachage par division**
- ...

Hachage par division

Idée : utiliser le reste de la division euclidienne pour déterminer l'indice dans le tableau.

Definition (Hachage par division)

Soit T un tableau de taille n. Soit K l'ensemble des clefs. Soit hash une **fonction de hachage de clefs**. La fonction d'indice par division index est la suivante :

$$\mathrm{index} : K \to T$$
$$\mathrm{index} : k \mapsto \mathrm{hash}(k) \mod n$$

La fonction de hachage de clefs transforme k en entier, si c'est un `string` par exemple.

Definition (Lemme des tiroirs)

Soit E et F des ensembles finis, tels que $|E| > |F|$. Alors il n'existe pas d'application injective de E dans F.

Autrement dit, si vous avez $n+1$ chaussettes et n tiroirs, il existe un tiroir qui contient au moins deux chaussettes.

Definition (Lemme des tiroirs)

Soit E et F des ensembles finis, tels que $|E| > |F|$. Alors il n'existe pas d'application injective de E dans F.

Autrement dit, si vous avez $n+1$ chaussettes et n tiroirs, il existe un tiroir qui contient au moins deux chaussettes.
L'ensemble des clefs est plus grand que T !

Definition (Lemme des tiroirs)

Soit E et F des ensembles finis, tels que $|E| > |F|$. Alors il n'existe pas d'application injective de E dans F.

Autrement dit, si vous avez $n+1$ chaussettes et n tiroirs, il existe un tiroir qui contient au moins deux chaussettes.
L'ensemble des clefs est plus grand que T ! Il existe donc des clefs qui ont le même indice...

Collision

2 Paradoxe des anniversaires

Les collisions ne sont pas rares !

- Problème des anniversaires

Dans un groupe de 23 personnes, 50% de chance qu'il y ait deux personnes avec le même anniversaire

Comment résoudre le problème des collisions ?

Deux grandes familles de solutions :

1. Tables de hachage par chainage séparé
2. Tables de hachage par adressage ouvert

Comment résoudre le problème des collisions ?

Deux grandes familles de solutions :

1. Tables de hachage par chainage séparé
2. Tables de hachage par adressage ouvert

Comment résoudre le problème des collisions ?

Deux grandes familles de solutions :

1. Tables de hachage par chainage séparé
2. Tables de hachage par adressage ouvert

Comment résoudre le problème des collisions ?

Deux grandes familles de solutions :

1. Tables de hachage par chainage séparé
2. Tables de hachage par adressage ouvert

Chainage séparé

Idée : ajouter les clefs en collision dans une structure de données annexe.

- Listes chaînées
- Arbres équilibrés
- ...

Chainage séparé

Idée : ajouter les clefs en collision dans une structure de données annexe.

- Listes chaînées
- Arbres équilibrés
- ...

Chainage séparé

Idée : ajouter les clefs en collision dans une structure de données annexe.

- Listes chaînées
- Arbres équilibrés
- ...

3 Chainage séparé

Idée : ajouter les clefs en collision dans une structure de données annexe.

- Listes chaînées
- Arbres équilibrés
- ...

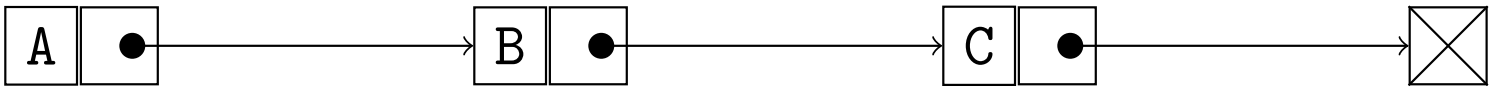
Chainage séparé

Idée : ajouter les clefs en collision dans une structure de données annexe.

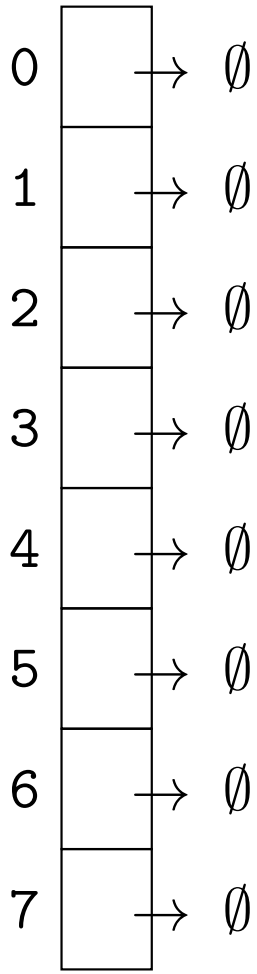
- Listes chaînées
- Arbres équilibrés
- ...

Le facteur de charge peut être supérieur à 1 !

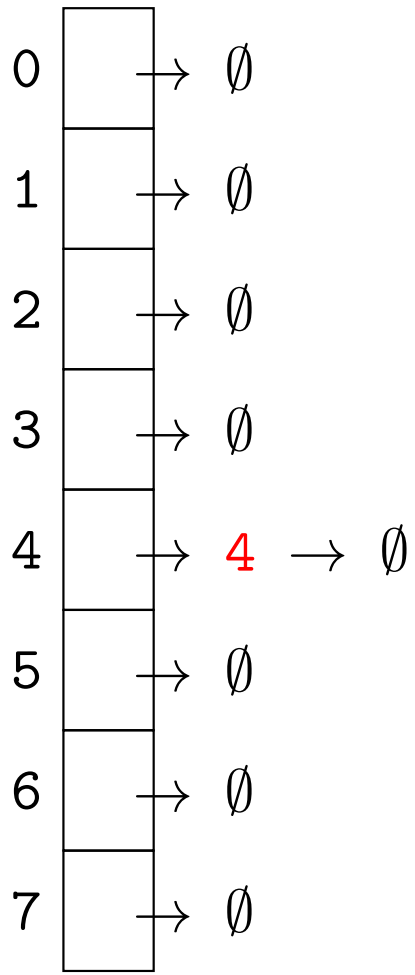
Listes chaînées



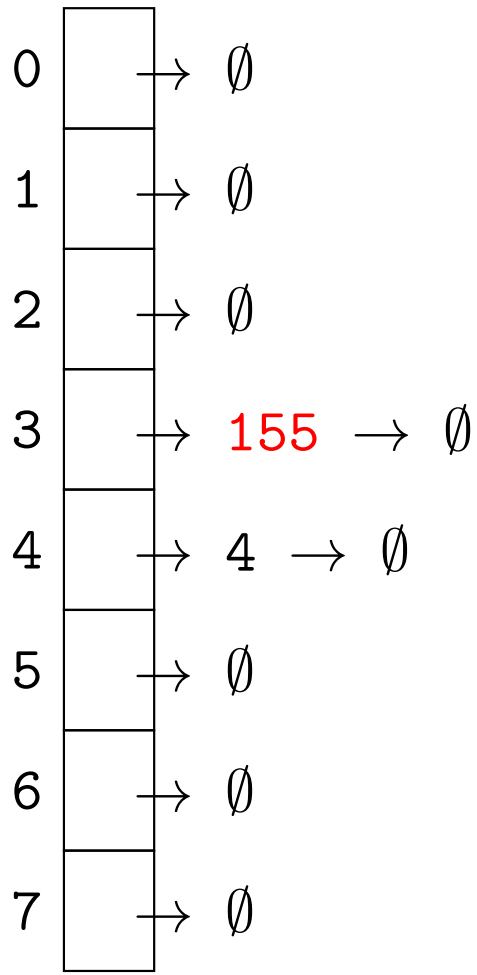
On utilise la table suivante, avec $\text{index}(k) = k \mod 8$



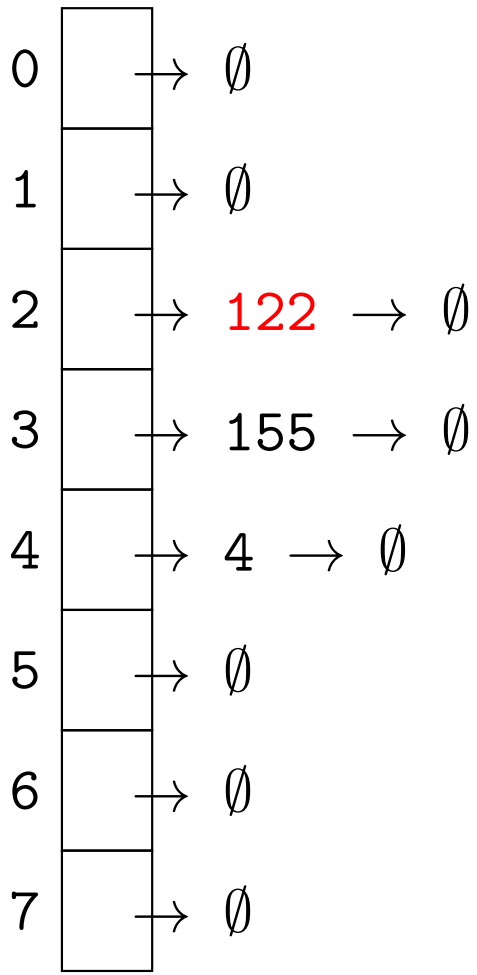
$\text{index}(4) = 4$



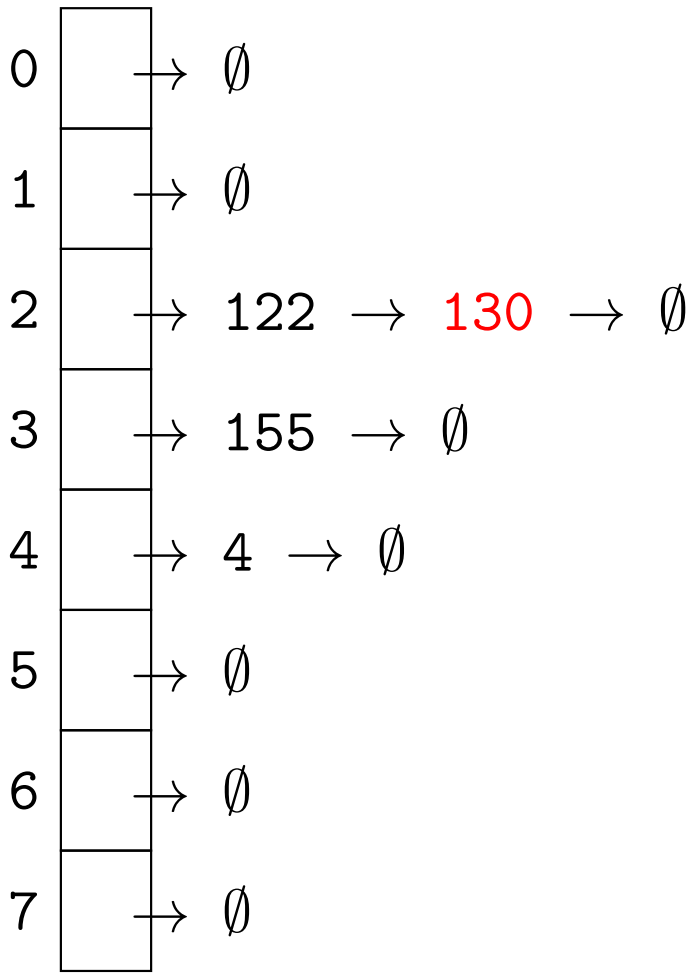
$\text{index}(155) = 3$



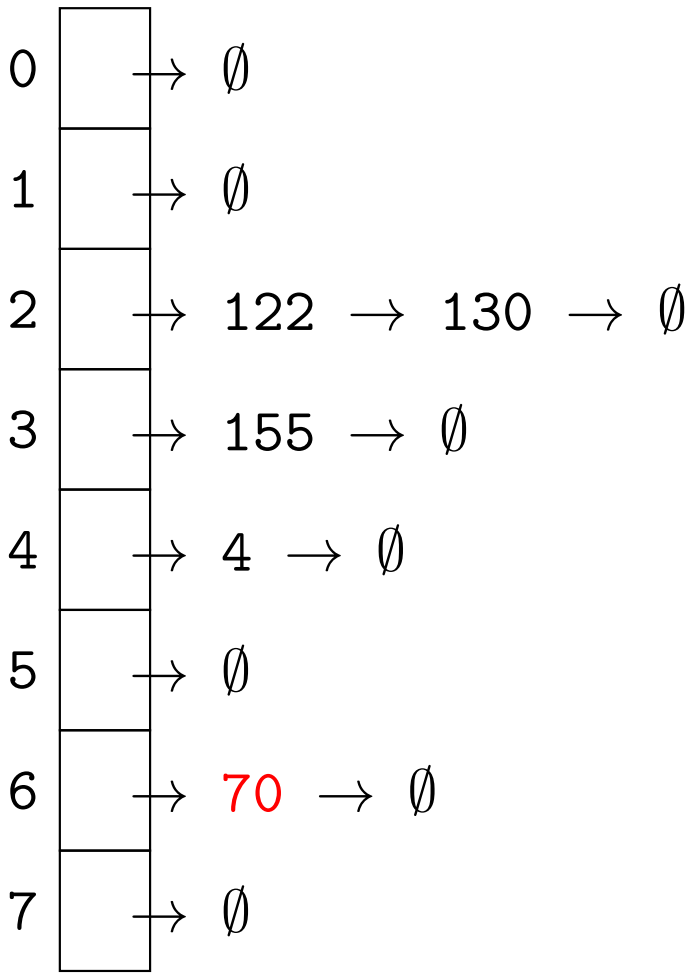
$\text{index}(122) = 2$



$\text{index}(130) = 2$



$\text{index}(70) = 6$



3 Suppression

La suppression se fait de manière classique pour une liste chaînée :

- suppression de l'élément
- raccord entre l'élément précédent et suivant

3 Redimensionnement

Plus la table est remplie, moins elle est efficace.

Plus la table est remplie, moins elle est efficace.
Comment résoudre ce problème ?

Plus la table est remplie, moins elle est efficace.
Comment résoudre ce problème ?
Redimensionnement de la table !

Redimensionnement de la table :

1. création d'une nouvelle table plus grande (par exemple, $\times 2$)
2. transfert des données dans la nouvelle table et rehachage de celles-ci

Redimensionnement de la table :

1. création d'une nouvelle table plus grande (par exemple, $\times 2$)
2. transfert des données dans la nouvelle table et rehachage de celles-ci

Redimensionnement de la table :

1. création d'une nouvelle table plus grande (par exemple, $\times 2$)
2. transfert des données dans la nouvelle table et rehachage de celles-ci

Complexité en pire cas : $\mathcal{O}(n)$
Complexité moyenne : $\mathcal{O}(1)$

Proof.

Voir les notes du cours pour les preuves

3 Autres structures de données

D'autres structures de données :

- Arbres binaires équilibrés (en Java, quand il y a beaucoup de collisions)
- BTree
- B+Tree
- Trie
- ...

3 Autres structures de données

D'autres structures de données :

- Arbres binaires équilibrés (en Java, quand il y a beaucoup de collisions)
- BTree
- B+Tree
- Trie
- ...

3 Autres structures de données

D'autres structures de données :

- Arbres binaires équilibrés (en Java, quand il y a beaucoup de collisions)
- BTree
- B+Tree
- Trie
- ...

3 Autres structures de données

D'autres structures de données :

- Arbres binaires équilibrés (en Java, quand il y a beaucoup de collisions)
- BTree
- B+Tree
- Trie
- ...

3 Autres structures de données

D'autres structures de données :

- Arbres binaires équilibrés (en Java, quand il y a beaucoup de collisions)
- BTree
- B+Tree
- Trie
- ...

4 Adressage ouvert

En adressage ouvert : on stocke les valeurs en collision **dans la table**.

4 Adressage ouvert

En adressage ouvert : on stocke les valeurs en collision **dans la table**.

$\Longrightarrow$ Il faut trouver un endroit où insérer !

4 Adressage ouvert

En adressage ouvert : on stocke les valeurs en collision **dans la table**.

$\Longrightarrow$ Il faut trouver un endroit où insérer !

Méthode de **probing** : on fait des “sauts” pour trouver un emplacement

Adressage ouvert

En adressage ouvert : on stocke les valeurs en collision **dans la table**.

$\Longrightarrow$ Il faut trouver un endroit où insérer !

Méthode de **probing** : on fait des “sauts” pour trouver un emplacement

Les sauts suivent des règles (à nous de les définir !)

4 Adressage ouvert

En adressage ouvert : on stocke les valeurs en collision **dans la table**.

$\Longrightarrow$ Il faut trouver un endroit où insérer !

Méthode de **probing** : on fait des “sauts” pour trouver un emplacement

Les sauts suivent des règles (à nous de les définir !)

Conséquence : le facteur de charge α doit être plus petit que 1 !

4 Adressage ouvert

En adressage ouvert : on stocke les valeurs en collision **dans la table**.

$\Longrightarrow$ Il faut trouver un endroit où insérer !

Méthode de **probing** : on fait des “sauts” pour trouver un emplacement

Les sauts suivent des règles (à nous de les définir !)

Conséquence : le facteur de charge α doit être plus petit que 1 !

En pratique : entre 0.65 et 0.75

4 Linear probing

Idée du **linear probing** : on fait des sauts selon une fonction linéaire.

Linear probing

Idée du **linear probing** : on fait des sauts selon une fonction linéaire.
De manière générale, la fonction d'indice est de la forme suivante :

$$\mathrm{index}(k, i) = (a\mathrm{hash}(k) + bi) \mod n$$

Linear probing

Idée du **linear probing** : on fait des sauts selon une fonction linéaire.
De manière générale, la fonction d'indice est de la forme suivante :

$$\mathrm{index}(k, i) = (a\mathrm{hash}(k) + bi) \mod n$$

Par défaut, $i = 0$. Si il y a collision pour $i = 0$, on essaie $i = 1$, puis $i = 2$, et ainsi de suite en incrémentant de 1 à chaque fois.

4 Linear probing : exemple

La table est de taille 7. La fonction d'insertion est la suivante :

$$\text{index}(k, i) = (\text{hash}(k) + i) \mod 7$$

4 Linear probing : exemple

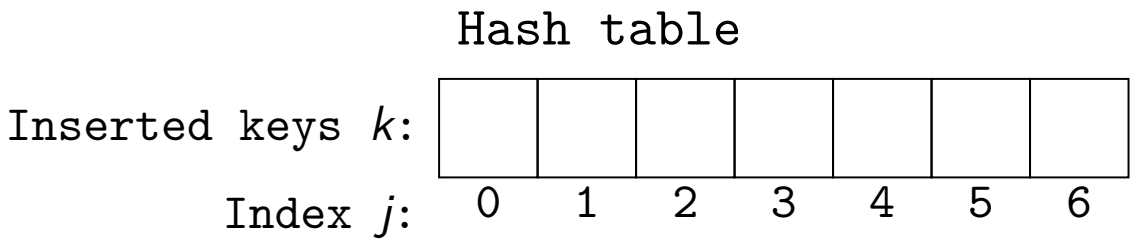
La table est de taille 7. La fonction d'insertion est la suivante :

$$\mathrm{index}(k, i) = (\mathrm{hash}(k) + i) \mod 7$$

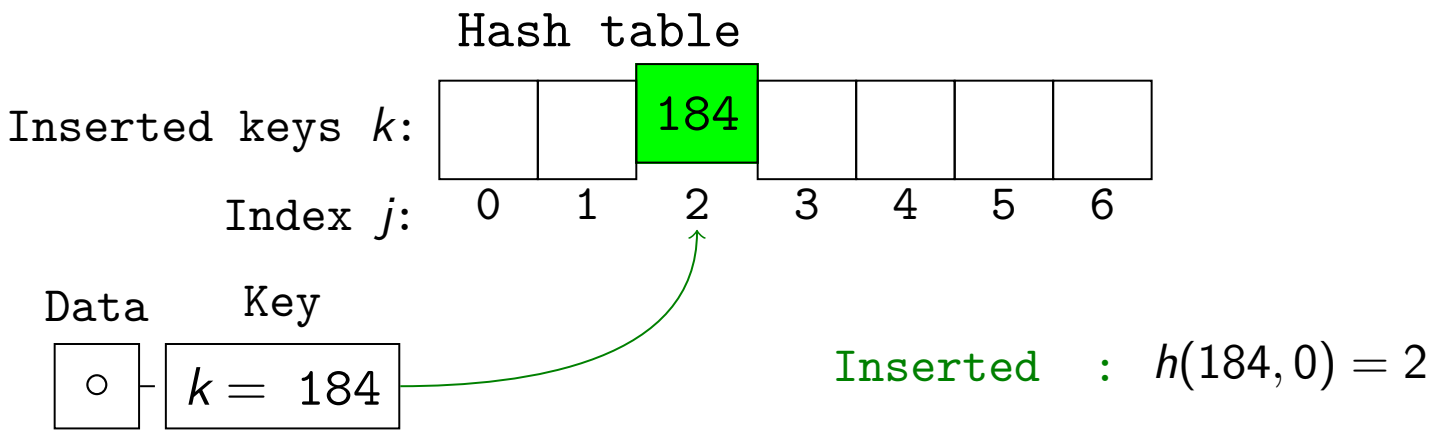
Nous allons insérer les valeurs suivantes dans la table

$$184, 165, 156, 81, 130$$

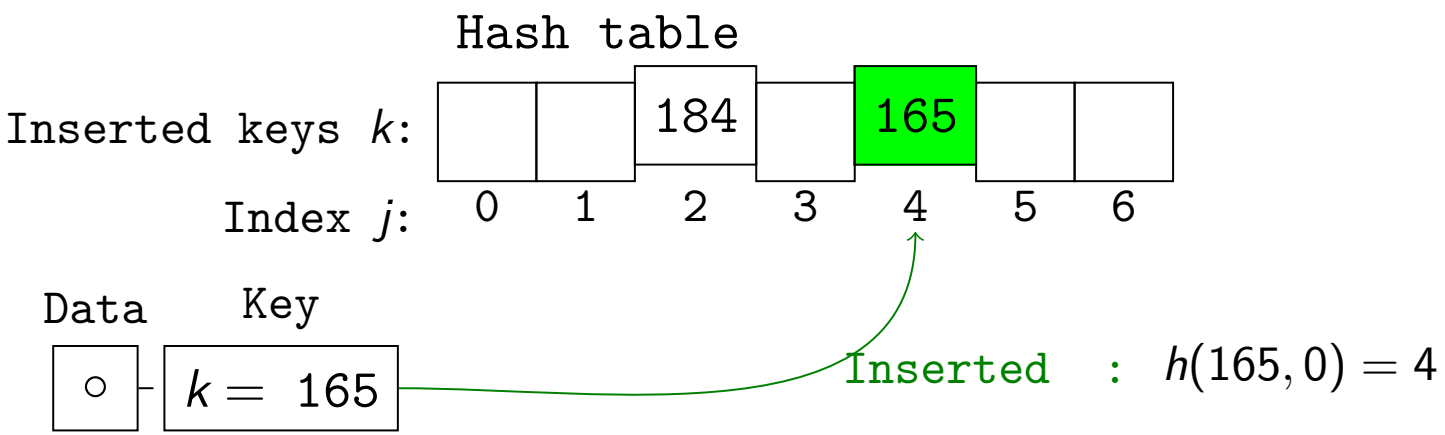
4 Linear probing : Insertion



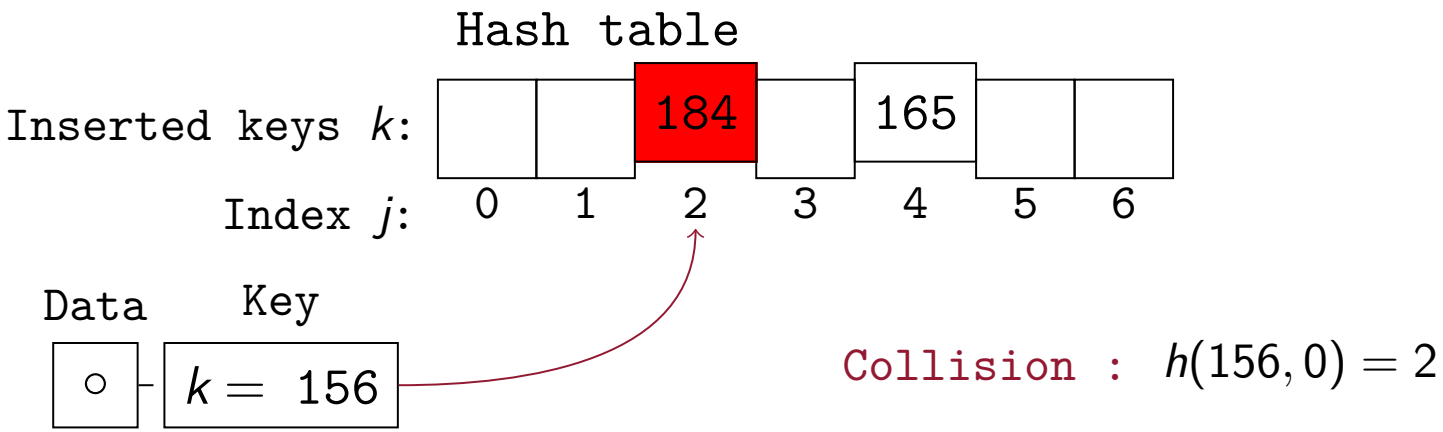
Linear probing : Insertion



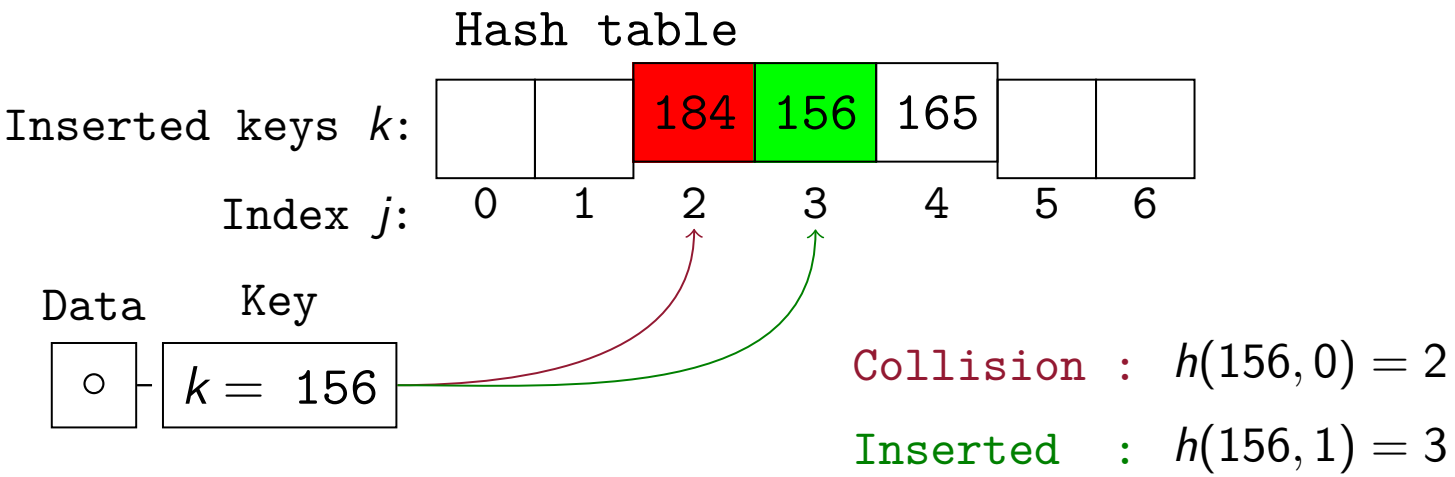
Linear probing : Insertion



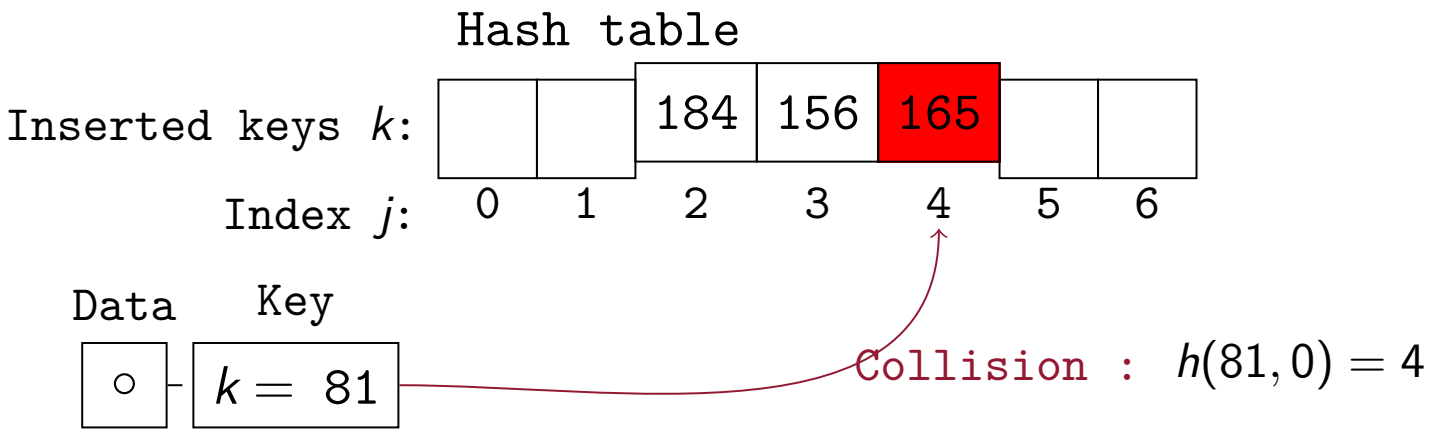
Linear probing : Insertion



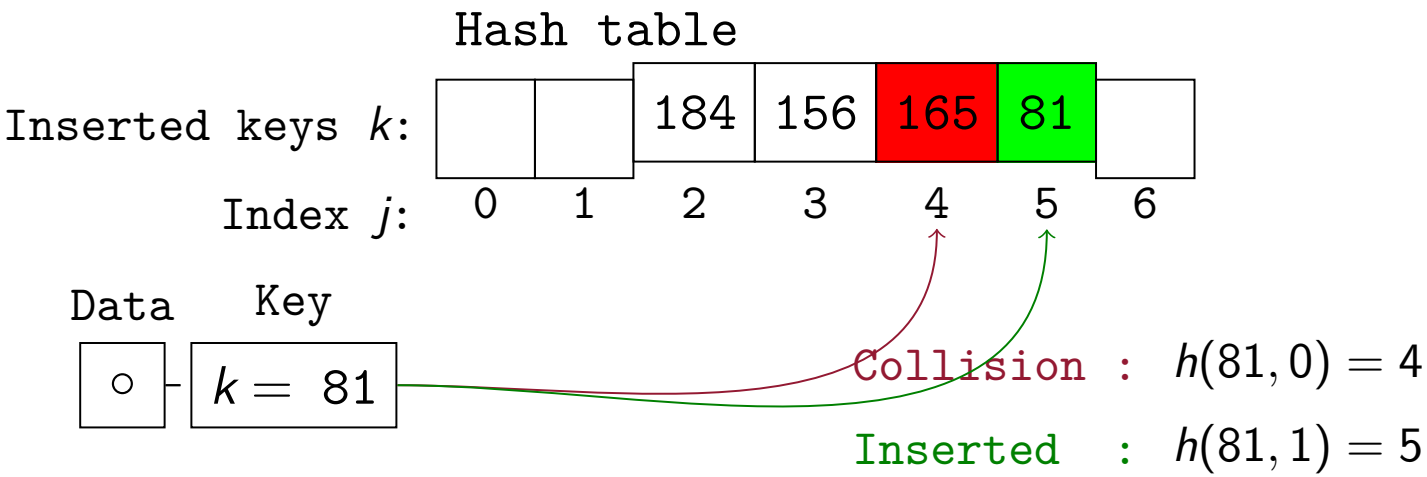
4 Linear probing : Insertion



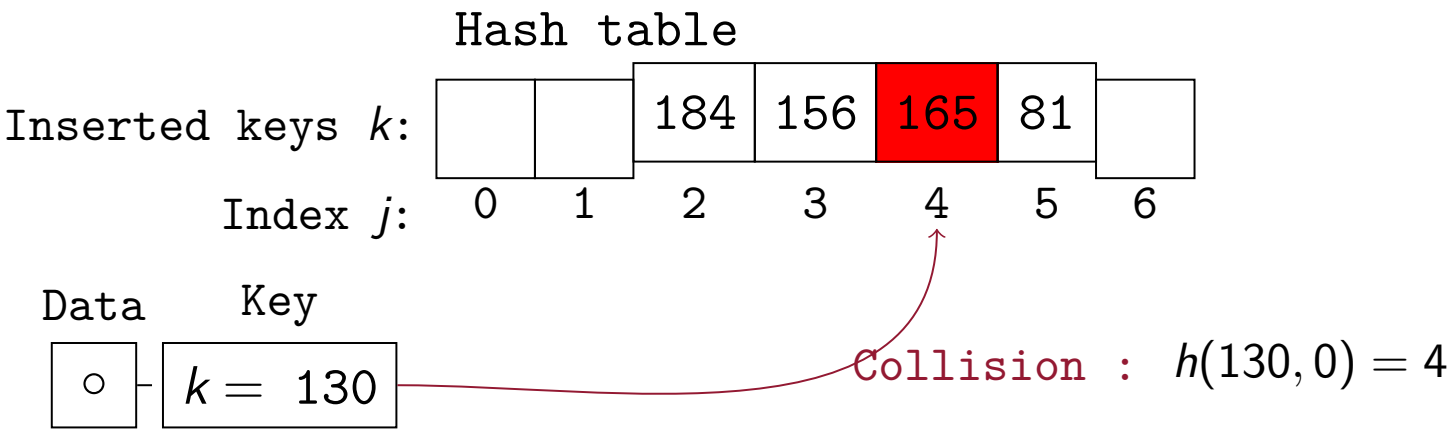
4 Linear probing : Insertion



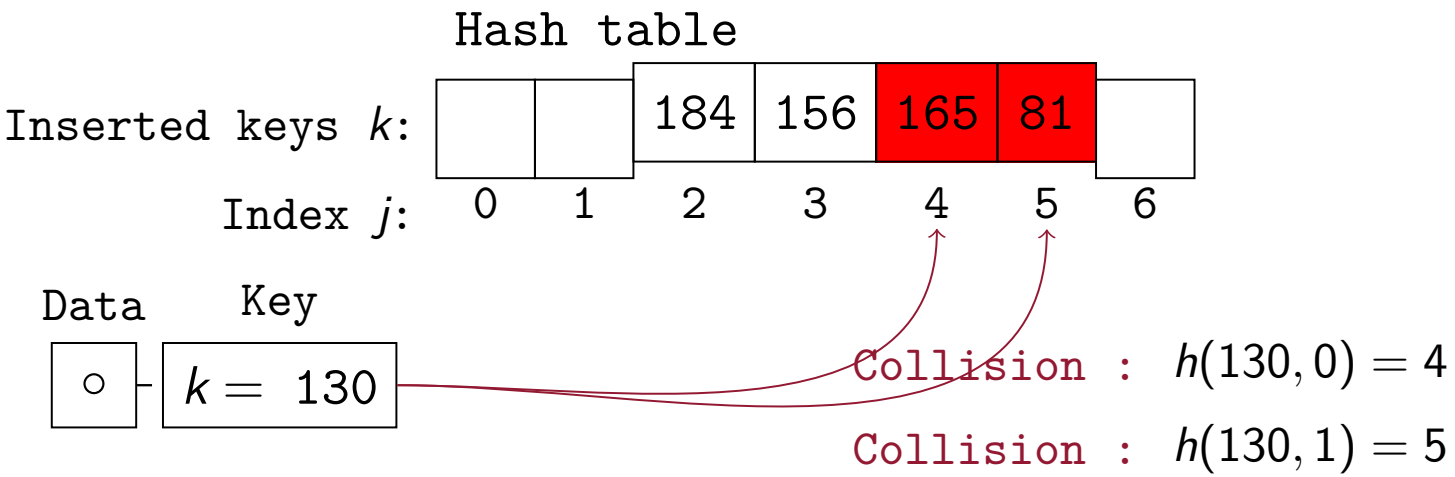
4 Linear probing : Insertion



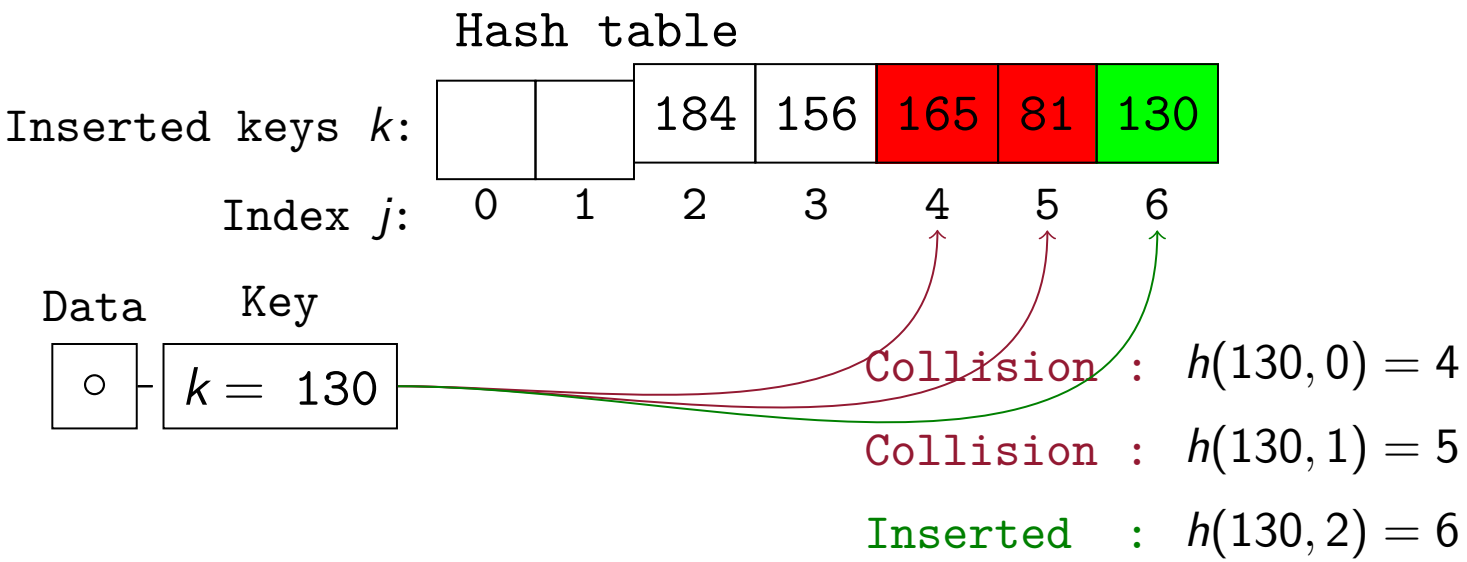
Linear probing : Insertion



4 Linear probing : Insertion



4 Linear probing : Insertion



4 Adressage ouvert : suppression

Comment supprimer des clefs ?

Adressage ouvert : suppression

Comment supprimer des clefs ?

Problème : le probing s'arrête quand il trouve un bucket vide

Adressage ouvert : suppression

Comment supprimer des clefs ?

Problème : le probing s'arrête quand il trouve un bucket vide

Solution : utilisation de tombstones, signalée par $\perp$

Adressage ouvert : suppression

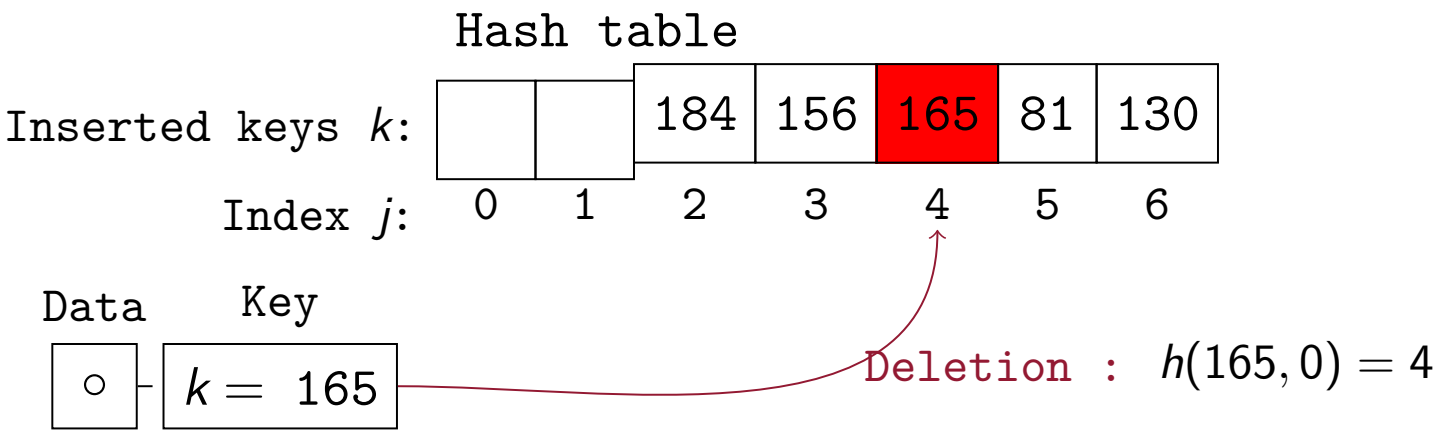
Comment supprimer des clefs ?

Problème : le probing s'arrête quand il trouve un bucket vide

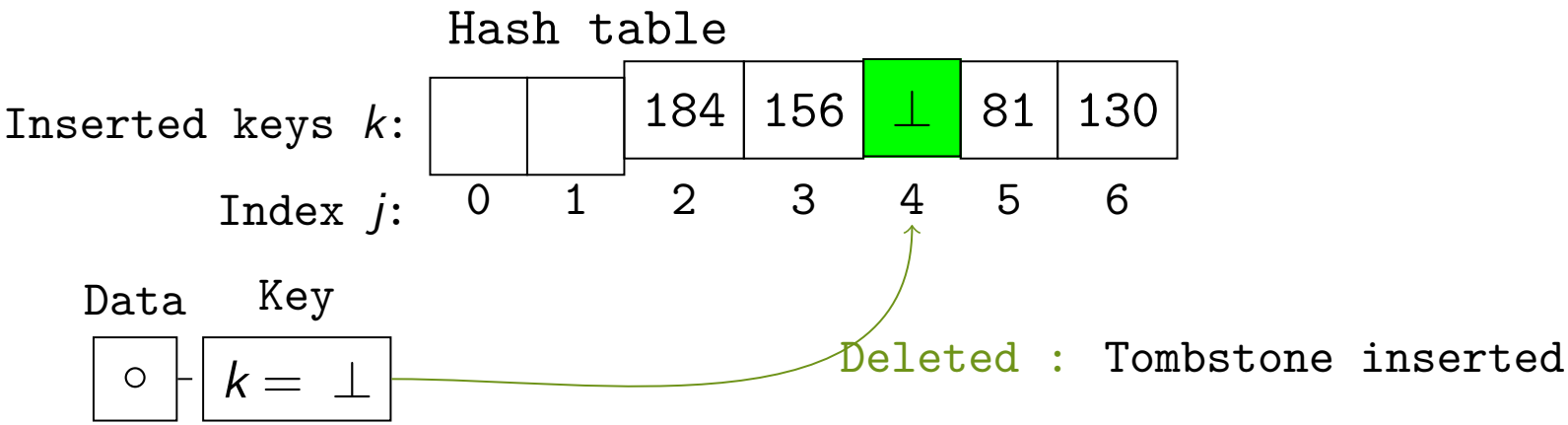
Solution : utilisation de tombstones, signalée par $\perp$

Exemple avec le linear probing : suppression de la clef 165.

Linear probing : suppression



4 Linear probing : suppression



4 Redimensionnement

Plus la table est remplie, moins elle est efficace.

4 Redimensionnement

Plus la table est remplie, moins elle est efficace.
Comment résoudre ce problème ?

4 Redimensionnement

Plus la table est remplie, moins elle est efficace.
Comment résoudre ce problème ?
Redimensionnement de la table !

Redimensionnement de la table :

1. création d'une nouvelle table plus grande (par exemple, $\times 2$)
2. transfert des données dans la nouvelle table et rehachage de celles-ci

Redimensionnement de la table :

1. création d'une nouvelle table plus grande (par exemple, $\times 2$)
2. transfert des données dans la nouvelle table et rehachage de celles-ci

Redimensionnement de la table :

1. création d'une nouvelle table plus grande (par exemple, $\times 2$)
2. transfert des données dans la nouvelle table et rehachage de celles-ci

Complexité en pire cas : $\mathcal{O}(n)$
Complexité moyenne : $\mathcal{O}(1)$

Proof.

Voir les notes du cours pour les preuves

4 Primary clustering

Avantage du linear probing : localité mémoire.
Désavantage du linear probing : primary clustering.

4 Primary clustering

Avantage du linear probing : localité mémoire.
Désavantage du linear probing : primary clustering.
Primary clustering : tendance à former de longues séquences successives remplies.

4 Autres méthodes

D'autres méthodes existent :

- Quadratic Probing
- Random hashing
- Robing Hood hashing
- Cuckoo hashing
- ...

4 Autres méthodes

D'autres méthodes existent :

- Quadratic Probing
- Random hashing
- Robing Hood hashing
- Cuckoo hashing
- ...

4 Autres méthodes

D'autres méthodes existent :

- Quadratic Probing
- Random hashing
- Robing Hood hashing
- Cuckoo hashing
- ...

4 Autres méthodes

D'autres méthodes existent :

- Quadratic Probing
- Random hashing
- Robing Hood hashing
- Cuckoo hashing
- ...

4 Autres méthodes

D'autres méthodes existent :

- Quadratic Probing
- Random hashing
- Robing Hood hashing
- Cuckoo hashing
- ...

5 Conclusion

Table de hachage = structure de données pour le problème du dictionnaire.
Complexité moyenne **constante**.
Plusieurs implémentations différentes.
Très utiles et utilisées en pratique.