# Partie 4: Tables de hachage

Marie-Pierre Béal, Laurent Bulteau

Université Paris-Est Marne-la-Vallée

# Motivation

## Objectif

Stocker un ensemble d'éléments, identifiés par des **clés** distinctes.

- Ajout
- Suppression
- Accès / modification d'un élément (pas de sa clé)

# Motivation

## Objectif

Stocker un ensemble d'éléments, identifiés par des **clés** distinctes.

- Ajout
- Suppression
- Accès / modification d'un élément (pas de sa clé)

## Exemples

| | | |
|---|---|---|
| Détails d'un film | $\leftrightarrow$ | Titre |
| Définition | $\leftrightarrow$ | Mot |
| Fiche d'un étudiant | $\leftrightarrow$ | N$^o$ d'étudiant |
| Facture | $\leftrightarrow$ | N$^o$ de facture |
| Page d'un livre | $\leftrightarrow$ | N$^o$ de page |

$\ldots$

# Motivation

## Objectif

Stocker un ensemble d'éléments, identifiés par des **clés** distinctes.

- Ajout
- Suppression
- Accès / modification d'un élément (pas de sa clé)

## Exemples

| | | |
|---|---|---|
| Détails d'un film | $\leftrightarrow$ | Titre |
| Définition | $\leftrightarrow$ | Mot |
| Fiche d'un étudiant | $\leftrightarrow$ | N$^o$ d'étudiant |
| Facture | $\leftrightarrow$ | N$^o$ de facture |
| Page d'un livre | $\leftrightarrow$ | N$^o$ de page |
| ... | | |

Si la clé est un entier inférieur au nombre d'éléments $\rightarrow$ **Tableau**

# Motivation

## Objectif

Stocker un ensemble d'éléments, identifiés par des **clés** distinctes.

- Ajout
- Suppression
- Accès / modification d'un élément (pas de sa clé)

## Exemples

| | | |
|---|---|---|
| Détails d'un film | $\leftrightarrow$ | Titre |
| Définition | $\leftrightarrow$ | Mot |
| Fiche d'un étudiant | $\leftrightarrow$ | N$^o$ d'étudiant |
| Facture | $\leftrightarrow$ | N$^o$ de facture |
| Page d'un livre | $\leftrightarrow$ | N$^o$ de page |
| ... | | |

Si la clé est un entier inférieur au nombre d'éléments $\rightarrow$ **Tableau**

Et pour les autres types de clés .... ?

# Fonction de hachage

Idée de base : associer un nombre à chaque clé possible, pour revenir au cas des tableaux.
$\rightarrow$ une **valeur de hachage**, définie par une **fonction de hachage**.
En mémoire, on utilise un tableau de 0 à la valeur maximum.

# Fonction de hachage

Idée de base : associer un nombre à chaque clé possible, pour revenir au cas des tableaux.
$\rightarrow$ une **valeur de hachage**, définie par une **fonction de hachage**.
En mémoire, on utilise un tableau de 0 à la valeur maximum.

## Exemple : Films

| **Clé** | **Élément** |
|---|---|
| Matrix | Wachowski, 1999 |
| The Dark Knight | Nolan, 2008 |
| Inception | Nolan, 2010 |
| Fight Club | Fincher, 1999 |
| Usual Suspects | Singer, 1995 |
| Les Aventuriers de l'arche perdue | Spielberg, 1981 |

**Fonction de hachage :** nombre de caractères de la clé.

# Fonction de hachage

Idée de base : associer un nombre à chaque clé possible, pour revenir au cas des tableaux.
$\rightarrow$ une **valeur de hachage**, définie par une **fonction de hachage**.
En mémoire, on utilise un tableau de 0 à la valeur maximum.

## Exemple : Films

| | **Clé** | **Élément** |
|---|---|---|
| **6** | Matrix | Wachowski, 1999 |
| **15** | The Dark Knight | Nolan, 2008 |
| **9** | Inception | Nolan, 2010 |
| **10** | Fight Club | Fincher, 1999 |
| **14** | Usual Suspects | Singer, 1995 |
| **33** | Les Aventuriers de l'arche perdue | Spielberg, 1981 |

**Fonction de hachage :** nombre de caractères de la clé.

# Fonction de hachage

Idée de base : associer un nombre à chaque clé possible, pour revenir au cas des tableaux.
$\rightarrow$ une **valeur de hachage**, définie par une **fonction de hachage**.
En mémoire, on utilise un tableau de 0 à la valeur maximum.

## Exemple : Films

| | **Clé** | **Élément** |
|---|---|---|
| **6** | Matrix | Wachowski, 1999 |
| **15** | The Dark Knight | Nolan, 2008 |
| **9** | Inception | Nolan, 2010 |
| **10** | Fight Club | Fincher, 1999 |
| **14** | Usual Suspects | Singer, 1995 |
| **33** | Les Aventuriers de l'arche perdue | Spielberg, 1981 |
| | Le Parrain | Coppola, 1972 |

**Fonction de hachage :** nombre de caractères de la clé.

# Fonction de hachage

Idée de base : associer un nombre à chaque clé possible, pour revenir au cas des tableaux.
$\rightarrow$ une **valeur de hachage**, définie par une **fonction de hachage**.
En mémoire, on utilise un tableau de 0 à la valeur maximum.

## Exemple : Films

| | **Clé** | **Élément** |
|---|---|---|
| **6** | Matrix | Wachowski, 1999 |
| **15** | The Dark Knight | Nolan, 2008 |
| **9** | Inception | Nolan, 2010 |
| **10** | Fight Club | Fincher, 1999 |
| **14** | Usual Suspects | Singer, 1995 |
| **33** | Les Aventuriers de l'arche perdue | Spielberg, 1981 |
| **10** | Le Parrain | Coppola, 1972 |

**Fonction de hachage :** nombre de caractères de la clé.

# Fonction de hachage - conflits

Un **conflit** apparait quand deux clés distinctes ont la même valeur de hachage. Dans ce cas un tableau simple ne suffit pas.

# Fonction de hachage - conflits

Un **conflit** apparait quand deux clés distinctes ont la même valeur de hachage. Dans ce cas un tableau simple ne suffit pas.

## Un peu de math...

Avec une mémoire limitée, les conflits sont **inévitables** :
Exemple des titres : <50 caractères, chacun parmis $\simeq$40 symboles (lettres, nombres, poctuation) : tableau de taille $40^{50} \simeq 10^{80}$.
Si on est limités à $10^{12}$ valeurs de hachage différentes (un tera), deux clés distinctes auront nécessairement la même valeur.

# Fonction de hachage - conflits

Un **conflit** apparait quand deux clés distinctes ont la même valeur de hachage. Dans ce cas un tableau simple ne suffit pas.

**Un peu de math...**

Avec une mémoire limitée, les conflits sont **inévitables** :
Exemple des titres : <50 caractères, chacun parmis $\simeq$40 symboles (lettres, nombres, poctuation) : tableau de taille $40^{50} \simeq 10^{80}$.
Si on est limités à $10^{12}$ valeurs de hachage différentes (un tera), deux clés distinctes auront nécessairement la même valeur.

Deux stratégies pour gérer les conflits

- Hachage fermé
- Hachage ouvert

# Résumé

On veut stocker $n$ éléments :

- à l'aide d'un tableau de taille $B$
- chaque élément dispose d'une clé unique (type quelconque)
- à chaque clé correspond une valeur de hachage (pas nécessairement unique)

Idéalement : les opérations sur la table (ajout, accès, . . . ) s'effectuent en temps constant ($O(1)$).

# Exercice 1 - Fonction de hachage

On souhaite stocker les informations relatives à une liste d'étudiants identifiés par un numéro unique.

On utilise la fonction de hachage qui calcule le reste modulo 9 de la clé. (Donc $B = 9$.)

| **Clé** | **Élément** |
|---|---|
| 233 | David |
| 311 | Matt |
| 674 | Chris |
| 649 | Peter |
| 452 | Jodie |
| 152 | Tom |
| 324 | Jon |
| 963 | Bill |

**Calculer la valeur de hachage pour chaque élément.**

**Encore un peu de math...**

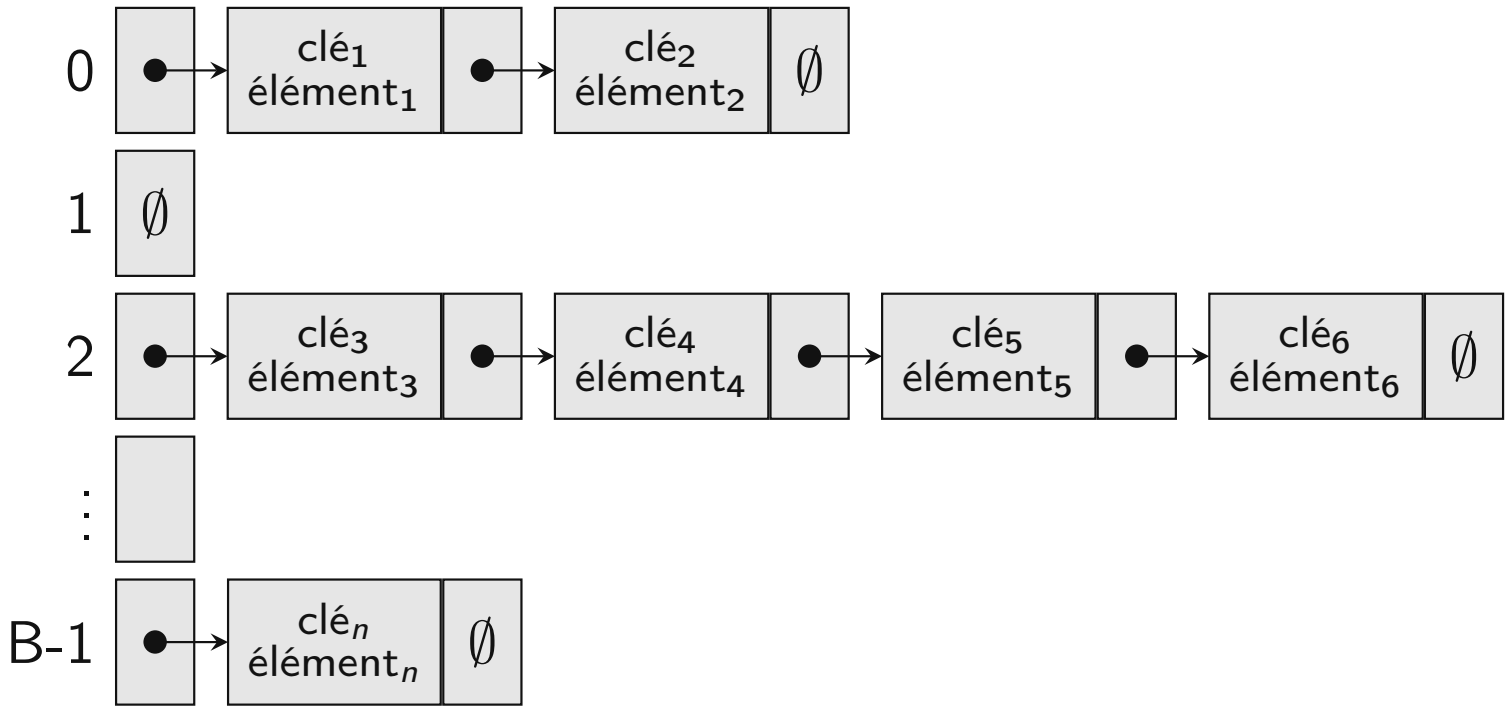
$233 \,\%\, 9 = (2 + 3 + 3) \,\%\, 9$

# Hachage fermé

# Hachage fermé

## Gestion des conflits

Tous les éléments avec la même valeur de hachage sont stockés **ensembles** dans une **liste chaînée**.



# Hachage fermé

## Gestion des conflits

Tous les éléments avec la même valeur de hachage sont stockés **ensembles** dans une **liste chaînée**.

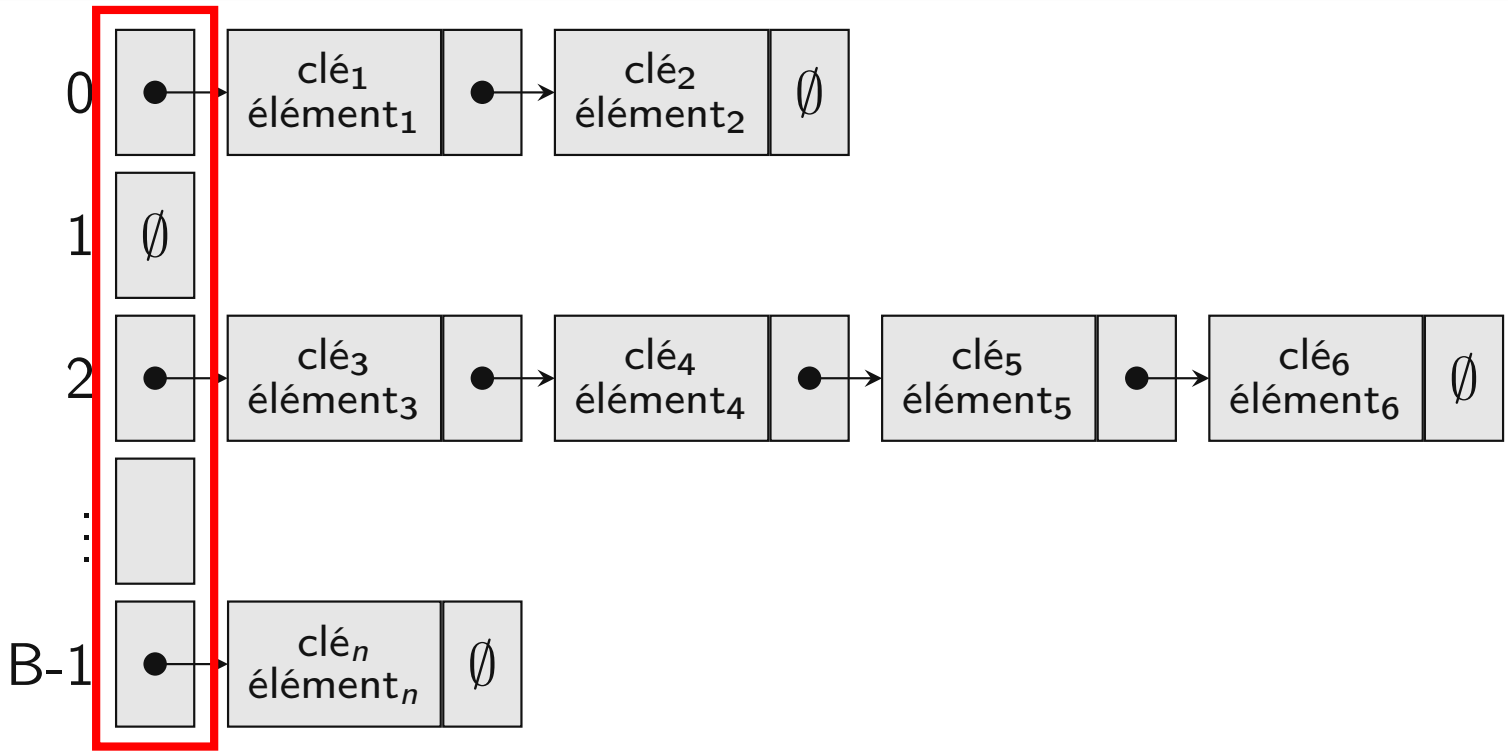
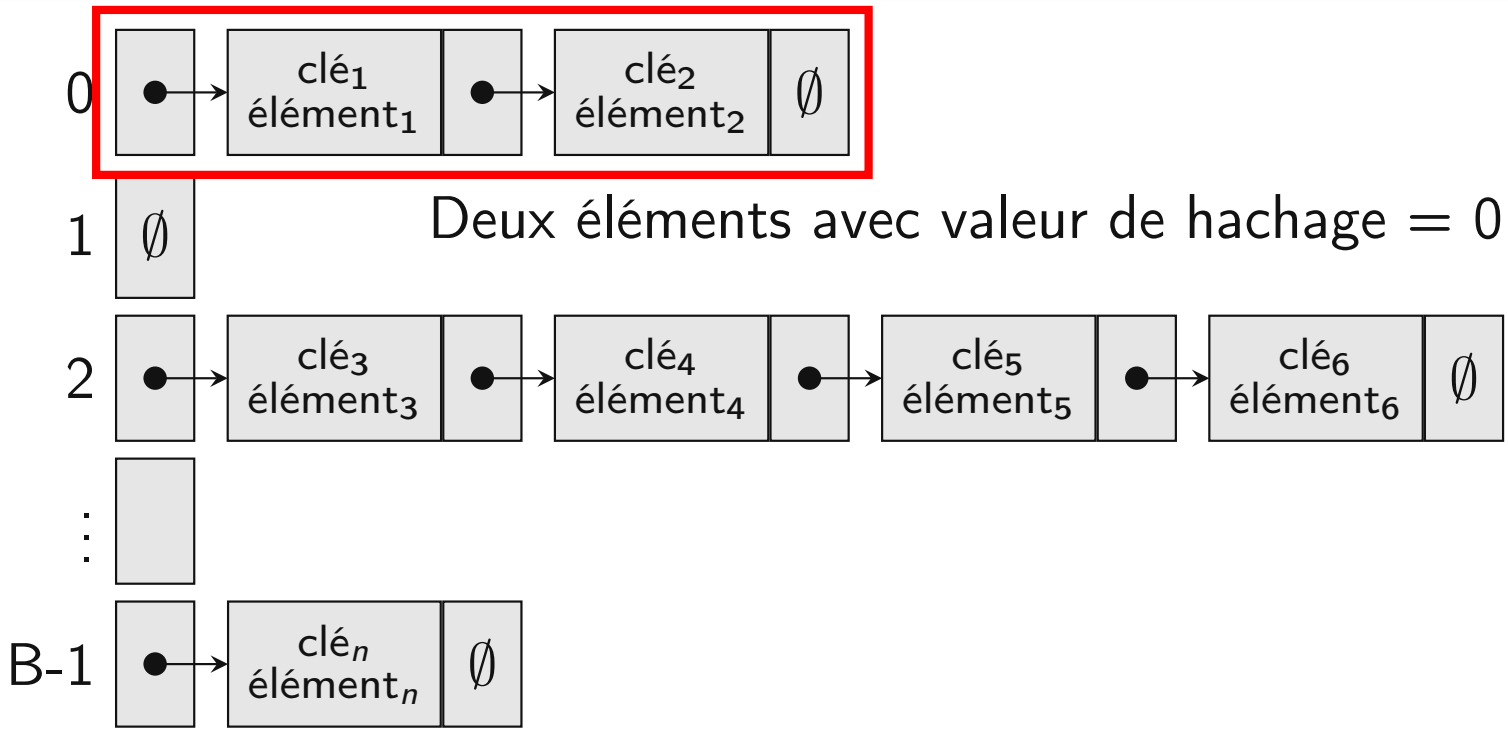



Tableau de $B$ listes

# Hachage fermé

## Gestion des conflits

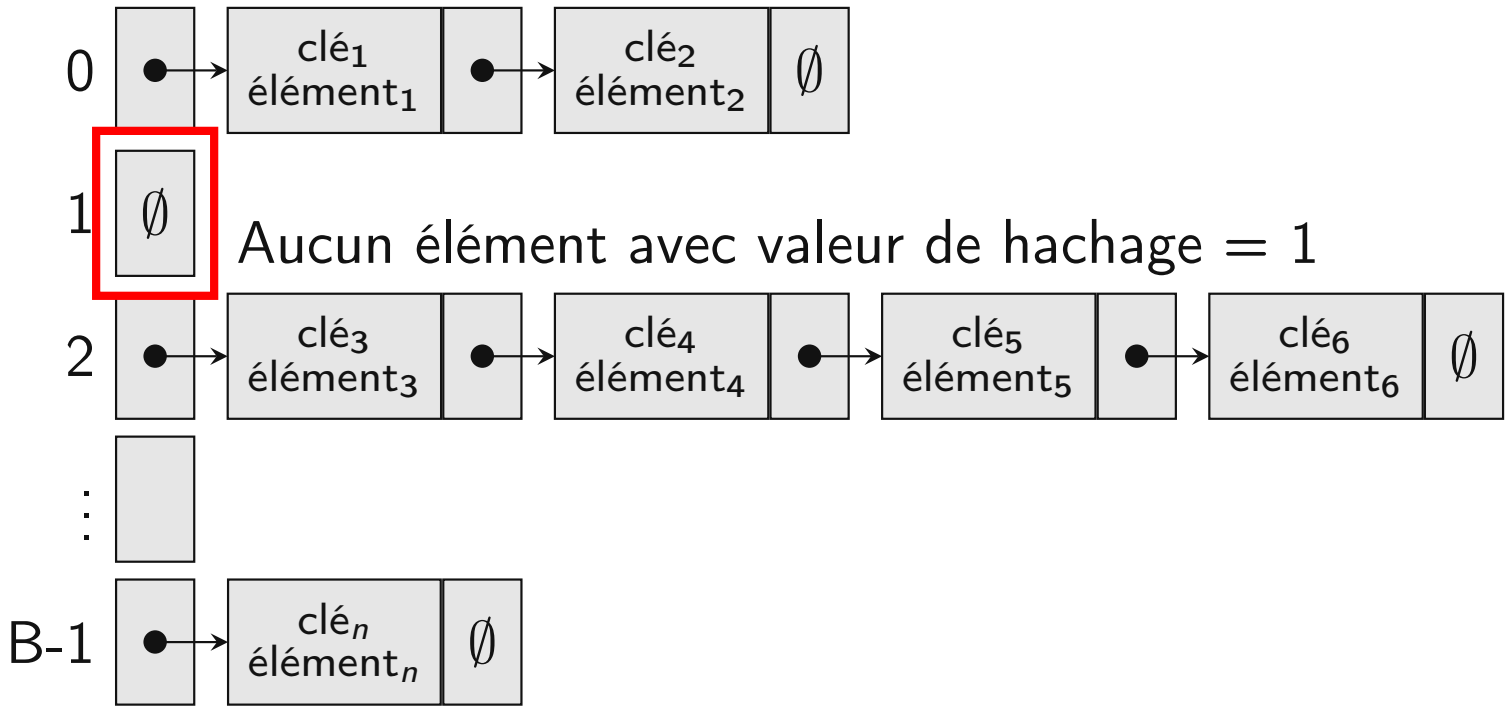
Tous les éléments avec la même valeur de hachage sont stockés **ensembles** dans une **liste chaînée**.



# Hachage fermé

## Gestion des conflits

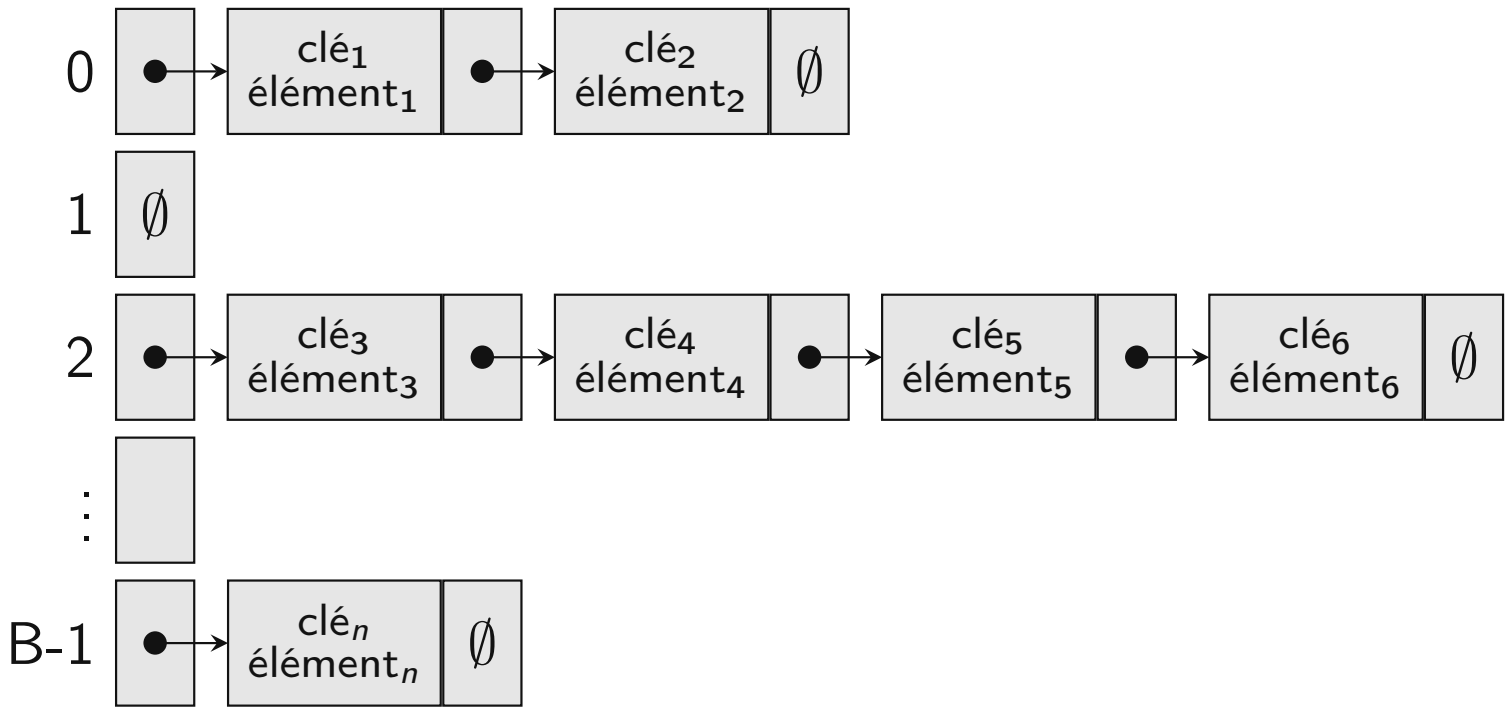
Tous les éléments avec la même valeur de hachage sont stockés **ensembles** dans une **liste chaînée**.



# Hachage fermé

## Gestion des conflits

Tous les éléments avec la même valeur de hachage sont stockés **ensembles** dans une **liste chaînée**.




Emprunte mémoire : $O(B + n)$

# Hachage fermé

- **Ajout d'un élément** (la clé doit être nouvelle)
  - Calculer la valeur de hachage
  - Extraire la liste correspondante du tableau
  - Insérer l'élément en tête de liste

# Hachage fermé

- **Ajout d'un élément** (la clé doit être nouvelle)
  - Calculer la valeur de hachage $O(1)$
  - Extraire la liste correspondante du tableau $O(1)$
  - Insérer l'élément en tête de liste $O(1)$

# Hachage fermé

- **Ajout d'un élément** (la clé doit être nouvelle)
  - Calculer la valeur de hachage $O(1)$
  - Extraire la liste correspondante du tableau $O(1)$
  - Insérer l'élément en tête de liste $O(1)$
- **Accès / édition / suppression d'un élément**
  - Calculer la valeur de hachage
  - Extraire la liste correspondante du tableau
  - Parcourir la liste en comparant les clés
  - Effectuer les modifications nécessaires

# Hachage fermé

- **Ajout d'un élément** (la clé doit être nouvelle)
  - Calculer la valeur de hachage $O(1)$
  - Extraire la liste correspondante du tableau $O(1)$
  - Insérer l'élément en tête de liste $O(1)$
- **Accès / édition / suppression d'un élément**
  - Calculer la valeur de hachage $O(1)$
  - Extraire la liste correspondante du tableau $O(1)$
  - Parcourir la liste en comparant les clés $O(\ell + 1)$
  - Effectuer les modifications nécessaires $O(1)$

# Hachage fermé

- **Ajout d'un élément** (la clé doit être nouvelle)
  - Calculer la valeur de hachage $O(1)$
  - Extraire la liste correspondante du tableau $O(1)$
  - Insérer l'élément en tête de liste $O(1)$
- **Accès / édition / suppression d'un élément**
  - Calculer la valeur de hachage $O(1)$
  - Extraire la liste correspondante du tableau $O(1)$
  - Parcourir la liste en comparant les clés $O(\ell + 1)$
  - Effectuer les modifications nécessaires $O(1)$

## Combien vaut $\ell$ ?

Si les valeurs de hachage sont équiprobables, alors *(on peut prouver que)* :

$$E[\ell] = \frac{n}{B}$$

Donc tant que $n = O(B)$, on a $\ell = O(1)$ *en moyenne*.

# Exercice 2 – Hachage fermé

1. Calculer la table de hachage fermée obtenue après insertion des éléments ci-contre, dans l'ordre indiqué.

   On utilise à nouveau $B = 9$ et la fonction de hachage qui calcule le reste modulo 9 de la clé.

| **Clé** | **Élément** |
|---|---|
| 233 | David |
| 311 | Matt |
| 674 | Chris |
| 649 | Peter |
| 452 | Jodie |
| 152 | Tom |
| 324 | Jon |
| 963 | Bill |

2. En comptant 1 opération pour calculer une valeur de hachage et 1 opération pour comparer deux clés, combien d'opérations sont nécessaires pour créer la table ? (On suppose que pour chaque insertion, on vérifie au préalable que la clé est effectivement nouvelle.)
3. Combien d'opérations auraient été nécessaires en utilisant une liste chaînée classique ?

# Implémentation du hachage fermé

```
#define table_size 9
typedef int Key_t;
typedef char* Value_t;
int hash(Key_t k){ return k % table_size; }

typedef struct cellule {
    Key_t key;
    Value_t val;
    struct cellule* succ;
} Cellule;
typedef Cellule* Liste;

typedef struct {
    int size;
    Liste tab[table_size];
} Hashtable;
```

# Exercice 3 – Implémentation du hachage fermé

Implémenter les fonctions de manipulation sur les tables de hachage fermé (penser à maintenir `size` à jour) :

```
Value_t get(Hashtable* t, Key_t k);
void set(Hashtable* t, Key_t k, Value_t v);
void delete(Hashtable* t, Key_t k);
```

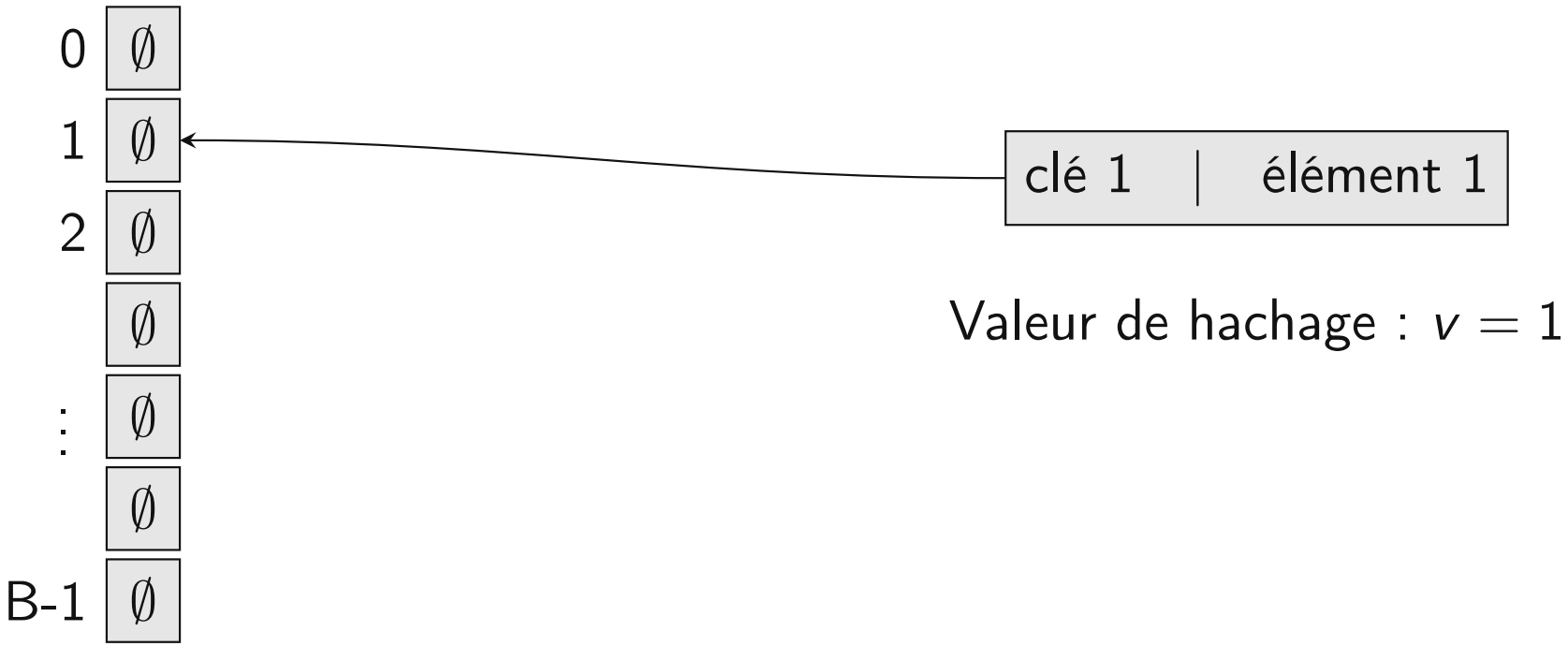
Pour `get`, on utilisera la valeur `null` si la clé n'est pas présente.
`set` peut soit insérer une nouvelle valeur si la clé n'était pas présente, soit modifier la valeur existante.
*À terminer en TP*

# Hachage ouvert

# Hachage ouvert

## Principe

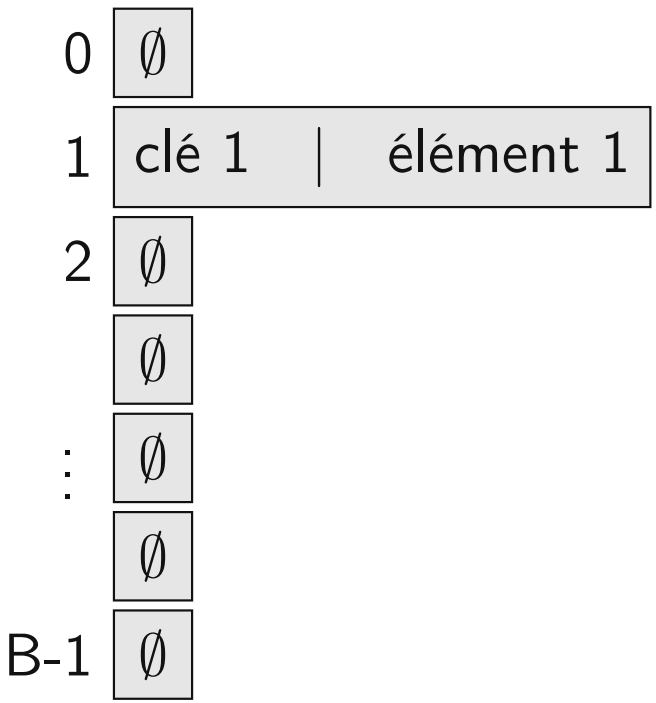
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

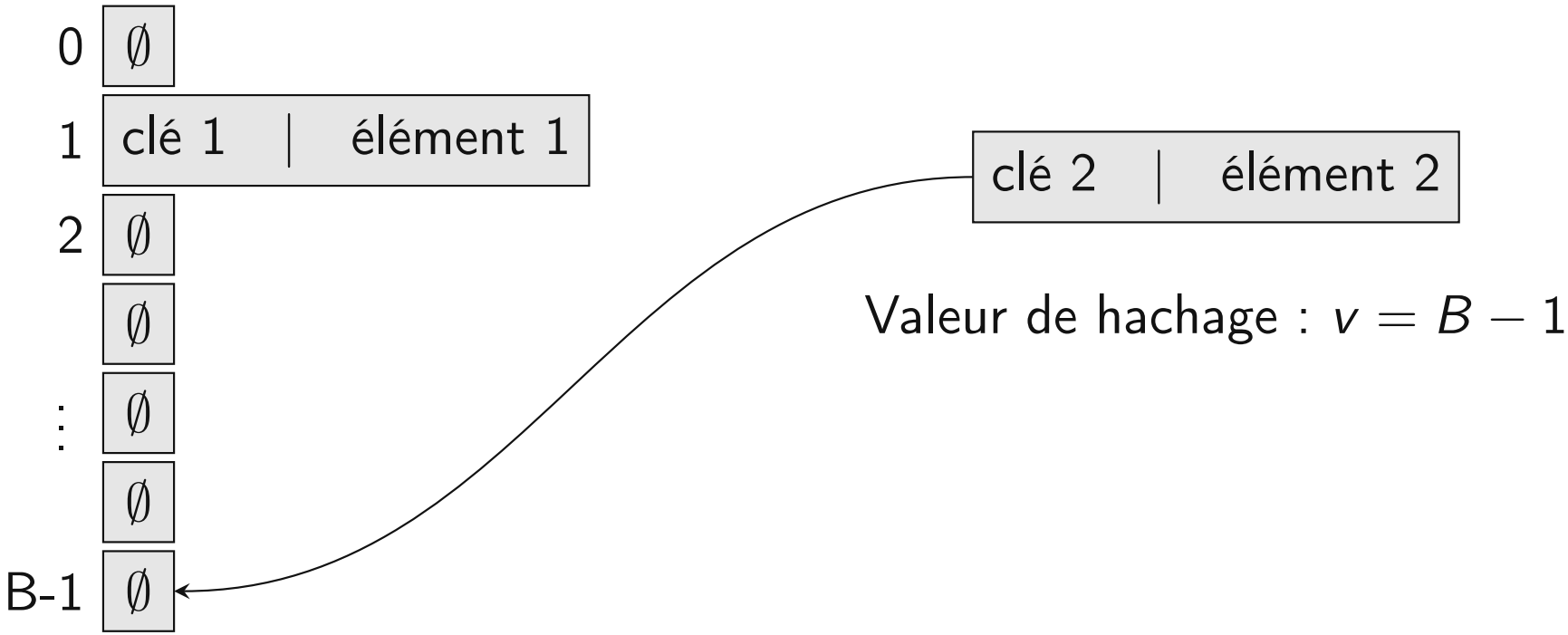
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

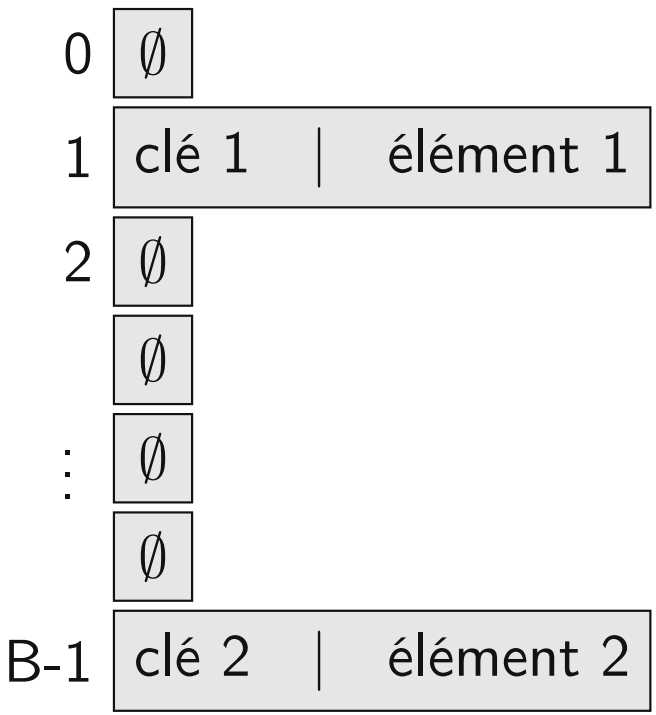
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

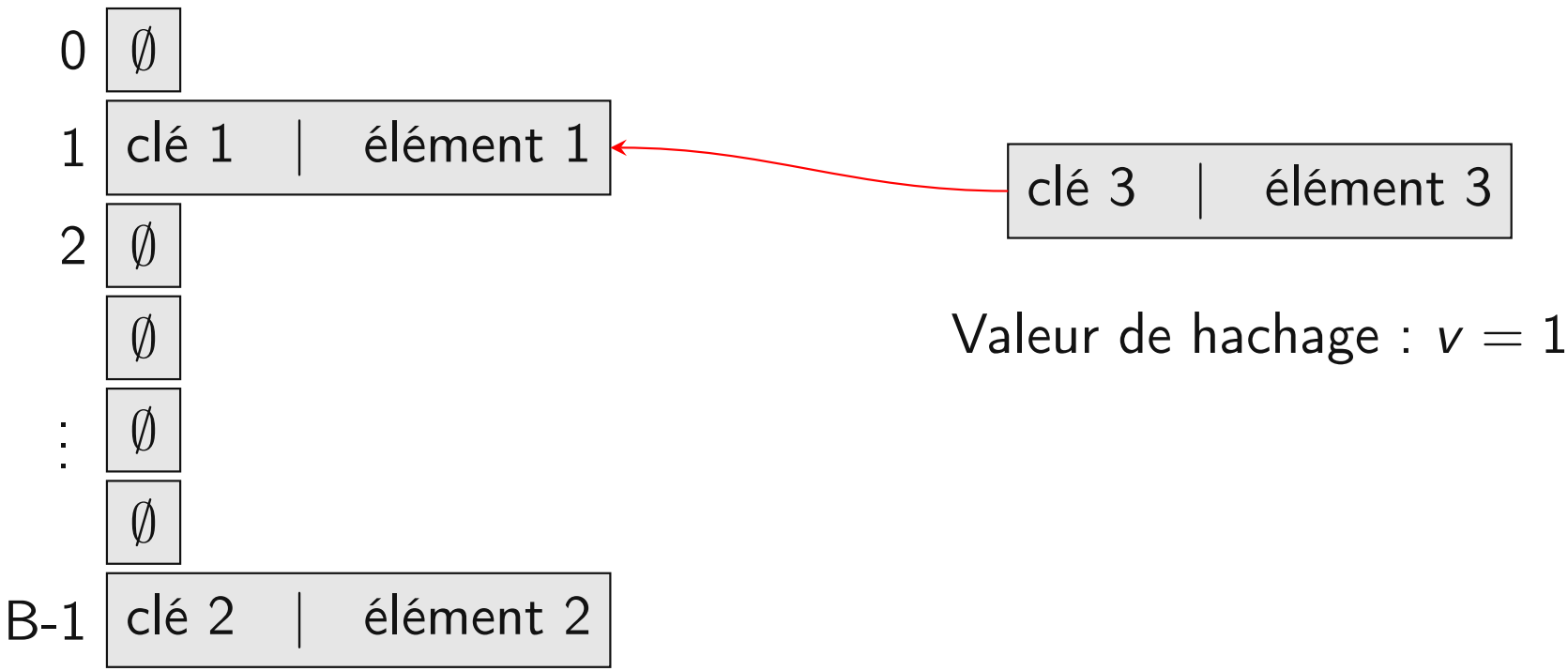
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

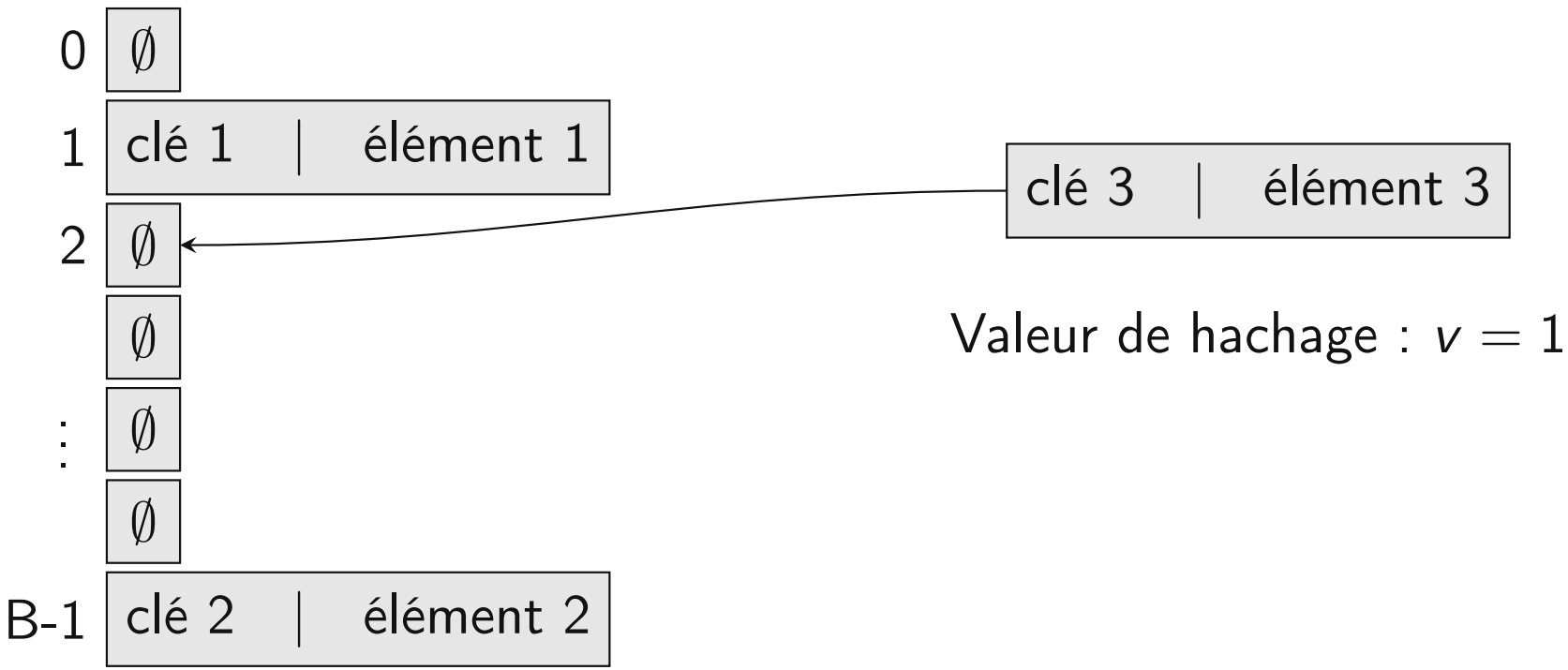
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

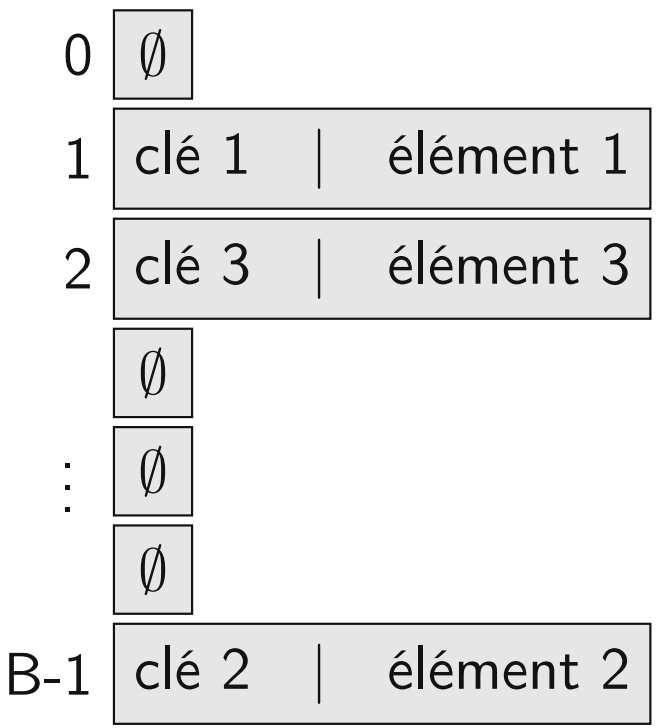
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

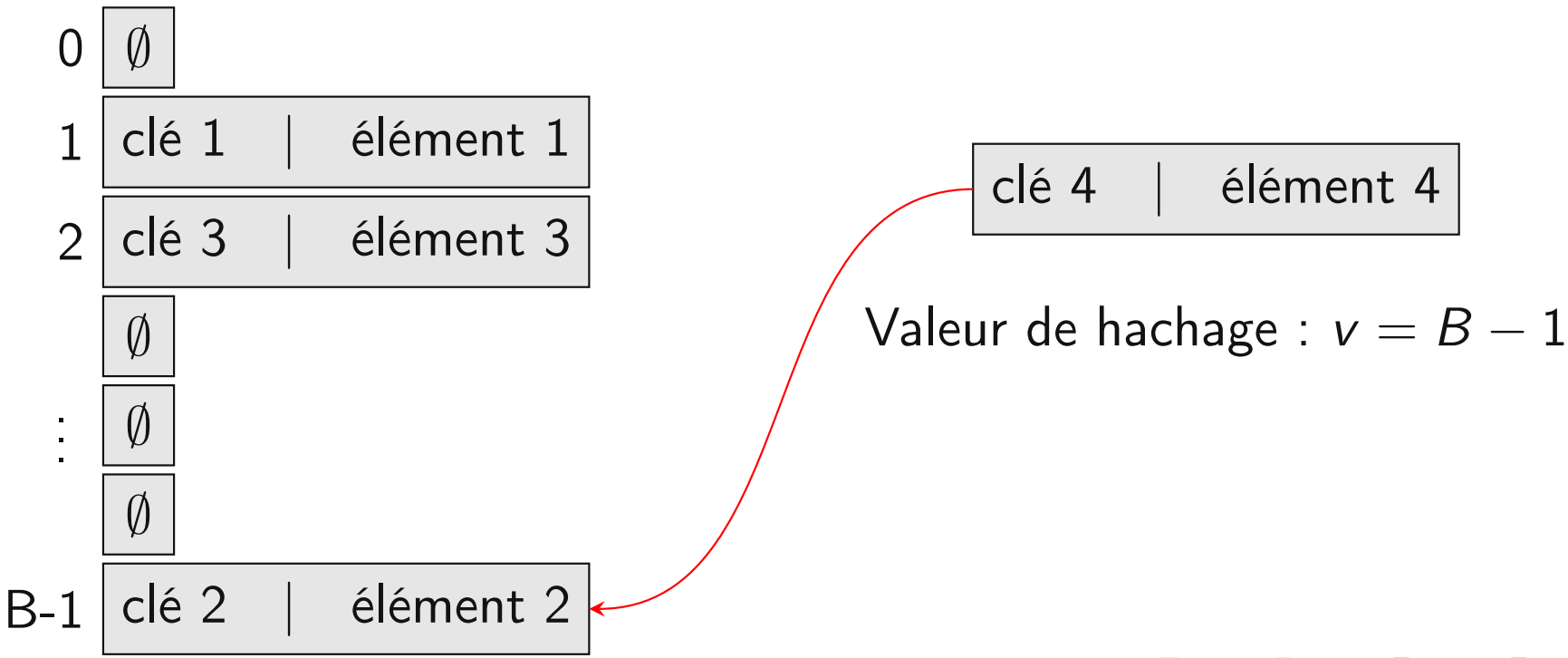
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

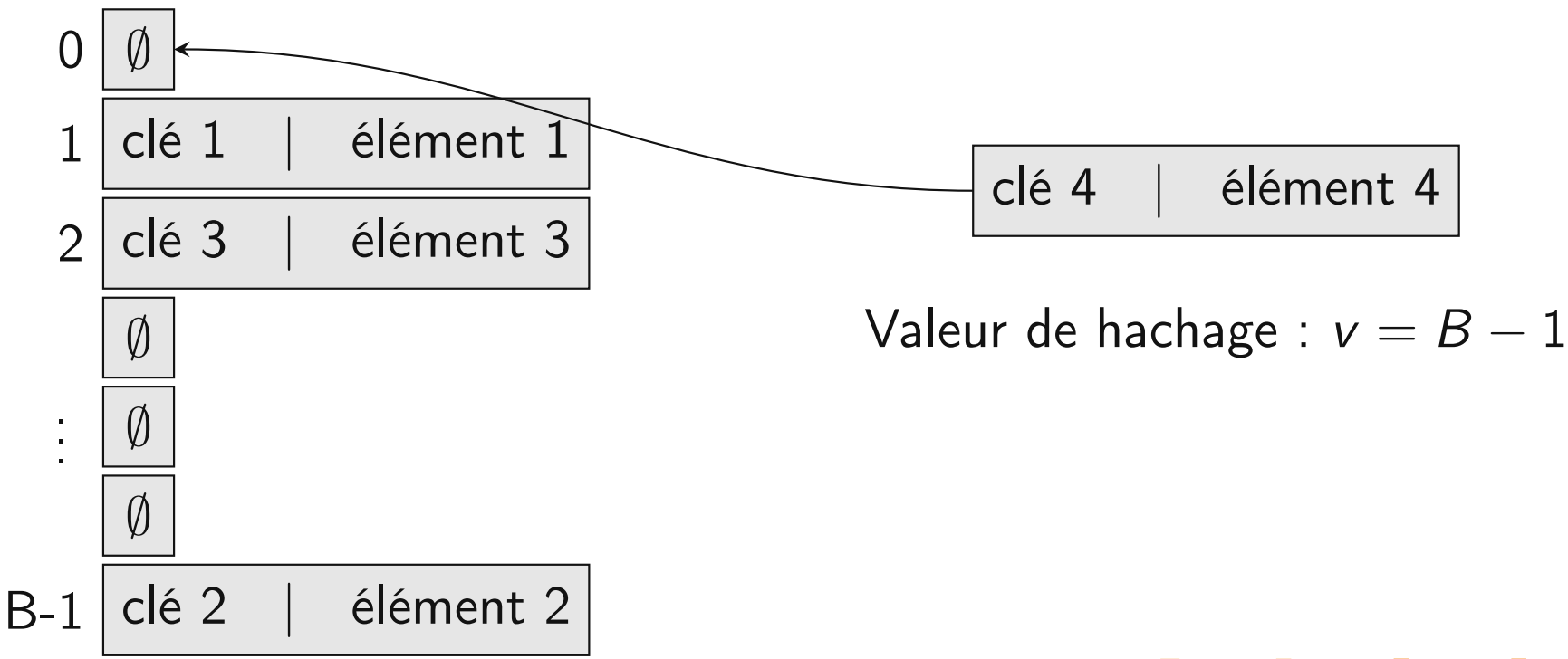
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

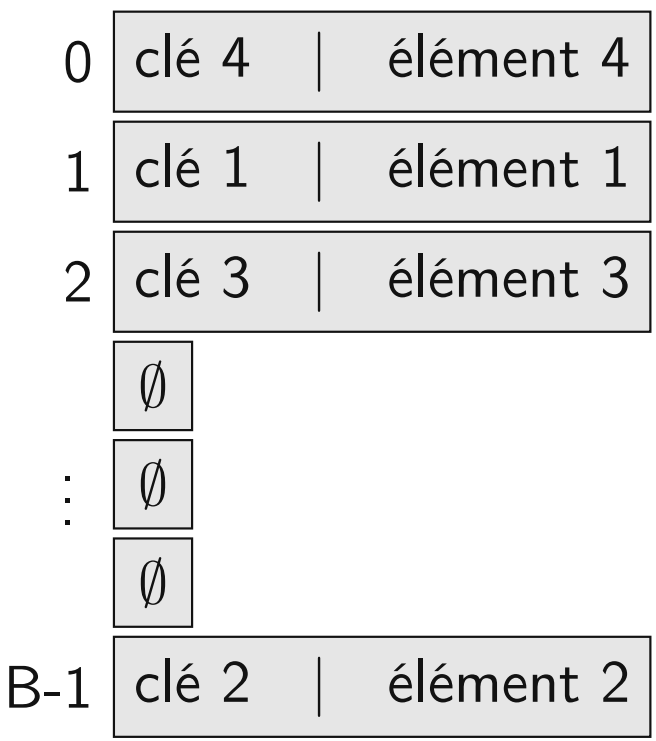
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

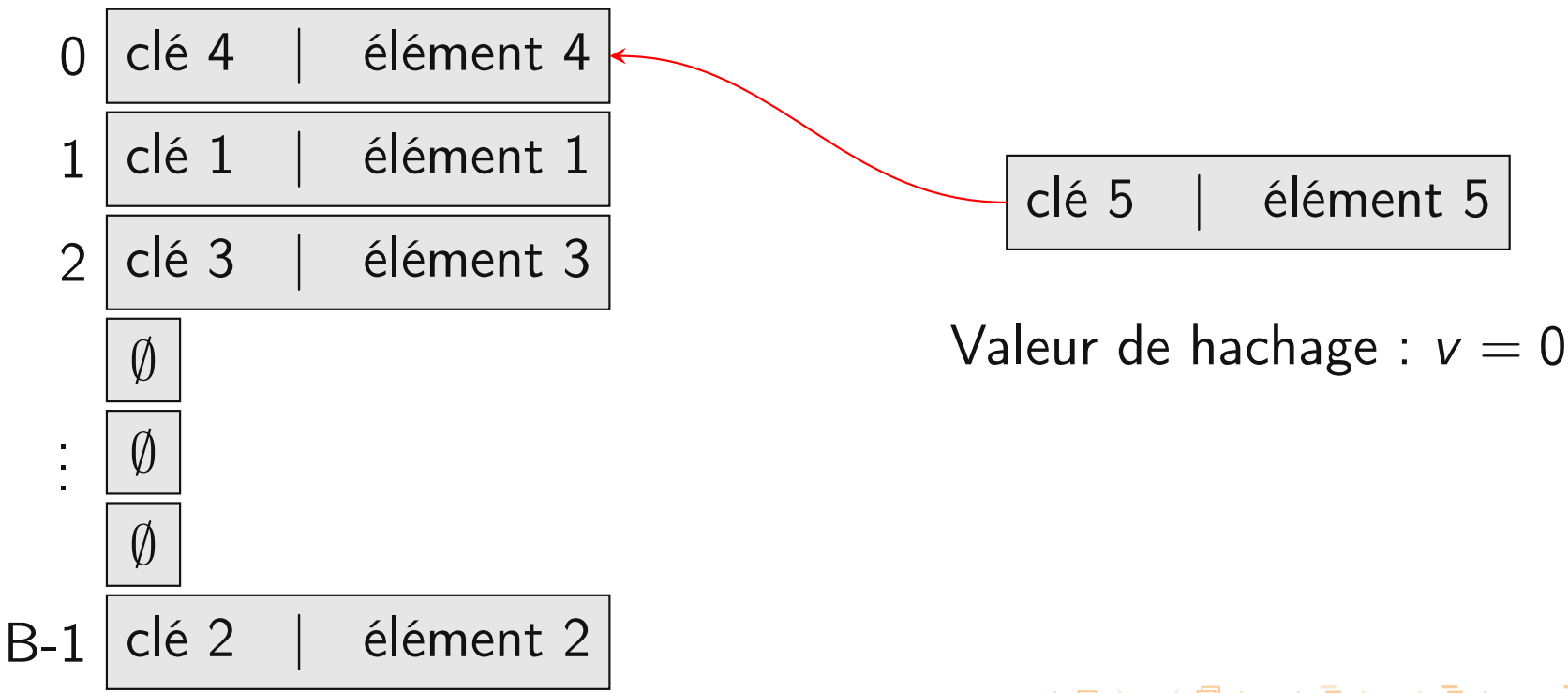
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

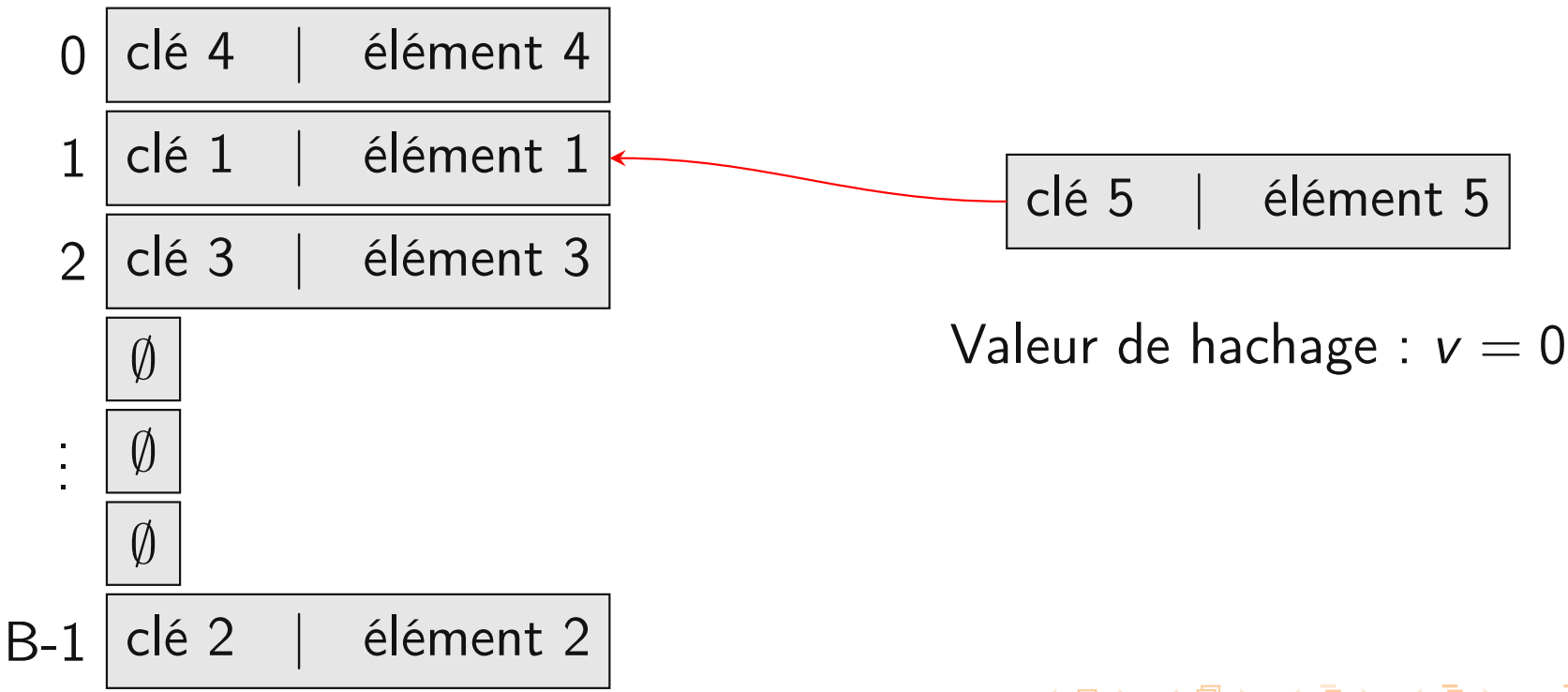
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

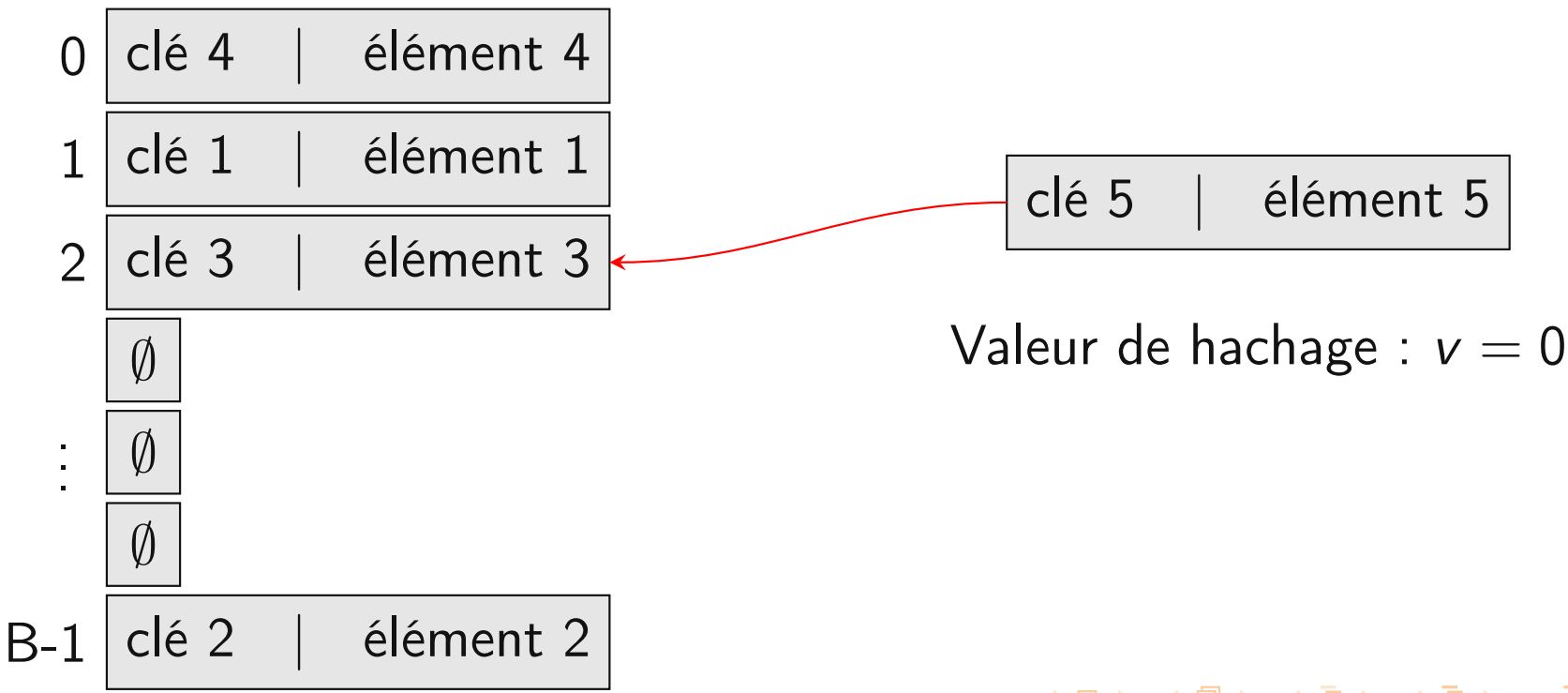
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

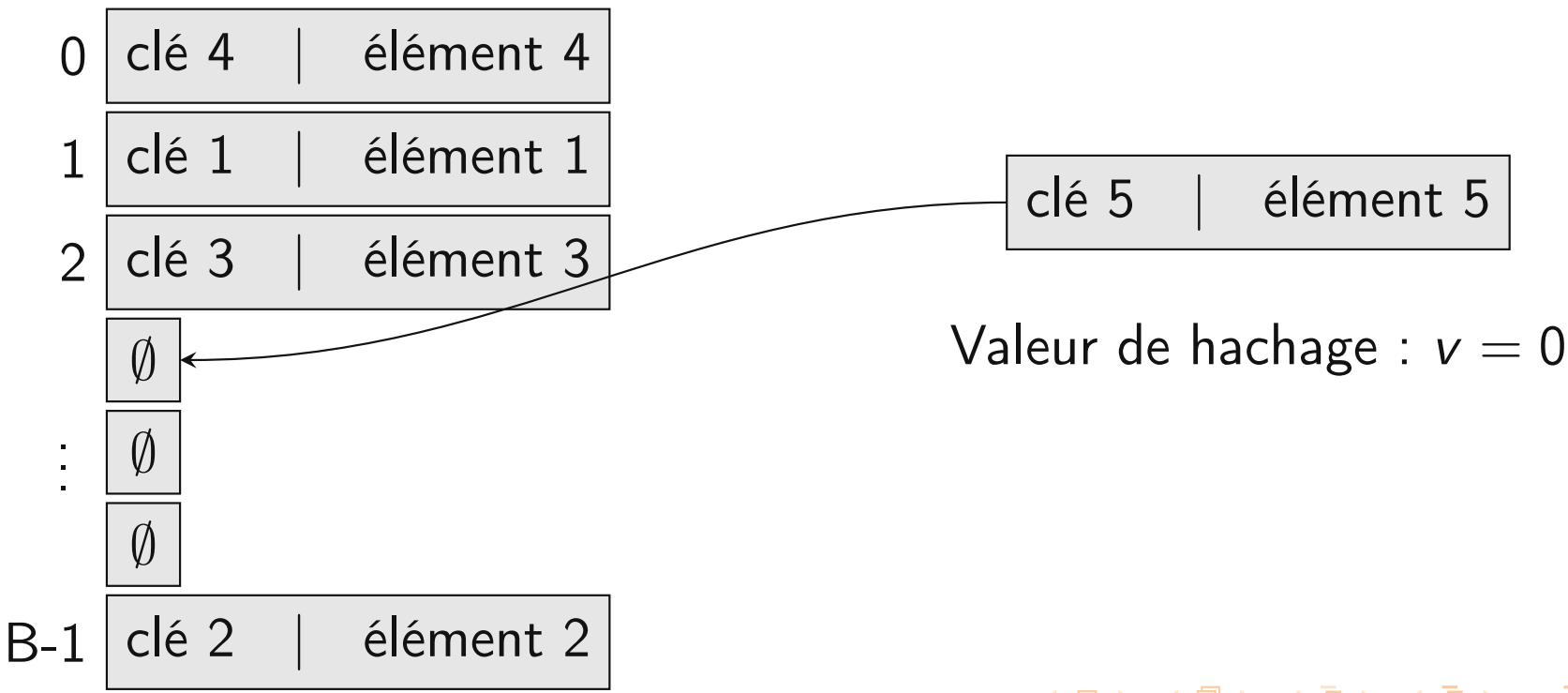
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

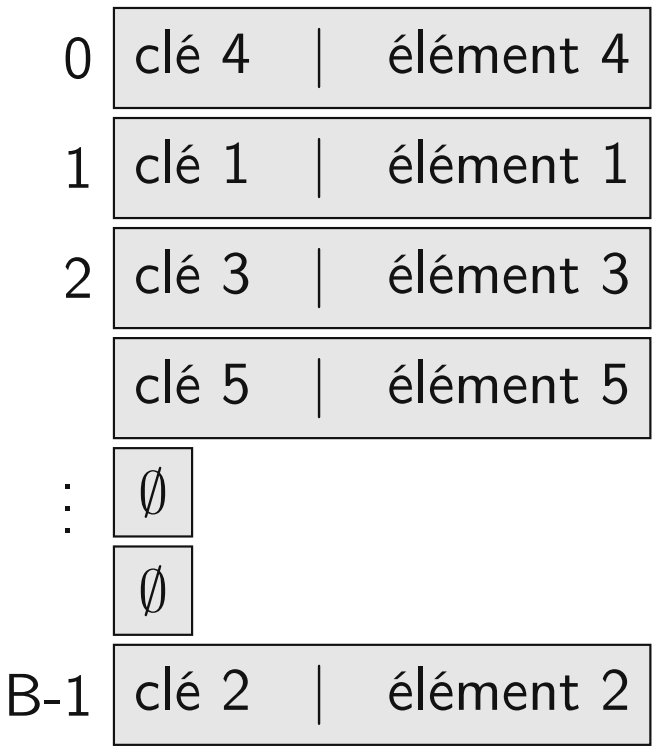
- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...



# Hachage ouvert

## Principe

- Un seul élément par case du tableau.
- **Insertion :** On calcule la valeur de hachage $\rightarrow v$.
  Si la case $v$ est déjà prise, on cherche la première case libre parmi $v+1$, $v+2$, ...




Valeur de hachage : $v = 0$

# Exercice 4 – Hachage ouvert

| Clé | Élément |
|---|---|
| 233 | David |
| 311 | Matt |
| 674 | Chris |
| 649 | Peter |
| 452 | Jodie |
| 152 | Tom |
| 324 | Jon |
| 963 | Bill |

1. Calculer la table de hachage ouvert obtenue après insertion des éléments ci-contre, dans l'ordre indiqué.

2. En comptant 1 opération pour calculer une valeur de hachage et 1 opération pour comparer deux clés, combien d'opérations sont nécessaires pour créer la table ? (On suppose que pour chaque insertion, on vérifie au préalable que la clé est effectivement nouvelle.)

# Recherche et suppression

- **Recherche d'un élément**
  - Calculer la valeur de hachage $v$
  - Tant que $T[V] \neq \emptyset$ :
    - Si la clé recherchée = la clé dans $T[v]$ : Retourner $T[v]$
    - $v = (v + 1)\%B$
  - Retourner "non trouvée"

# Recherche et suppression

- **Recherche d'un élément**
  - Calculer la valeur de hachage $v$
  - Tant que $T[V] \neq \emptyset$ :
    - Si la clé recherchée = la clé dans $T[v]$ : Retourner $T[v]$
    - $v = (v + 1)\%B$
  - Retourner "non trouvée"
- **Suppression d'un élément**
  - Calculer la valeur de hachage
  - Tant que $T[v] \neq \emptyset$ :
    - Si la clé recherchée = la clé dans $T[v]$ : $T[v] \leftarrow \emptyset$
    - $v = (v + 1)\%B$

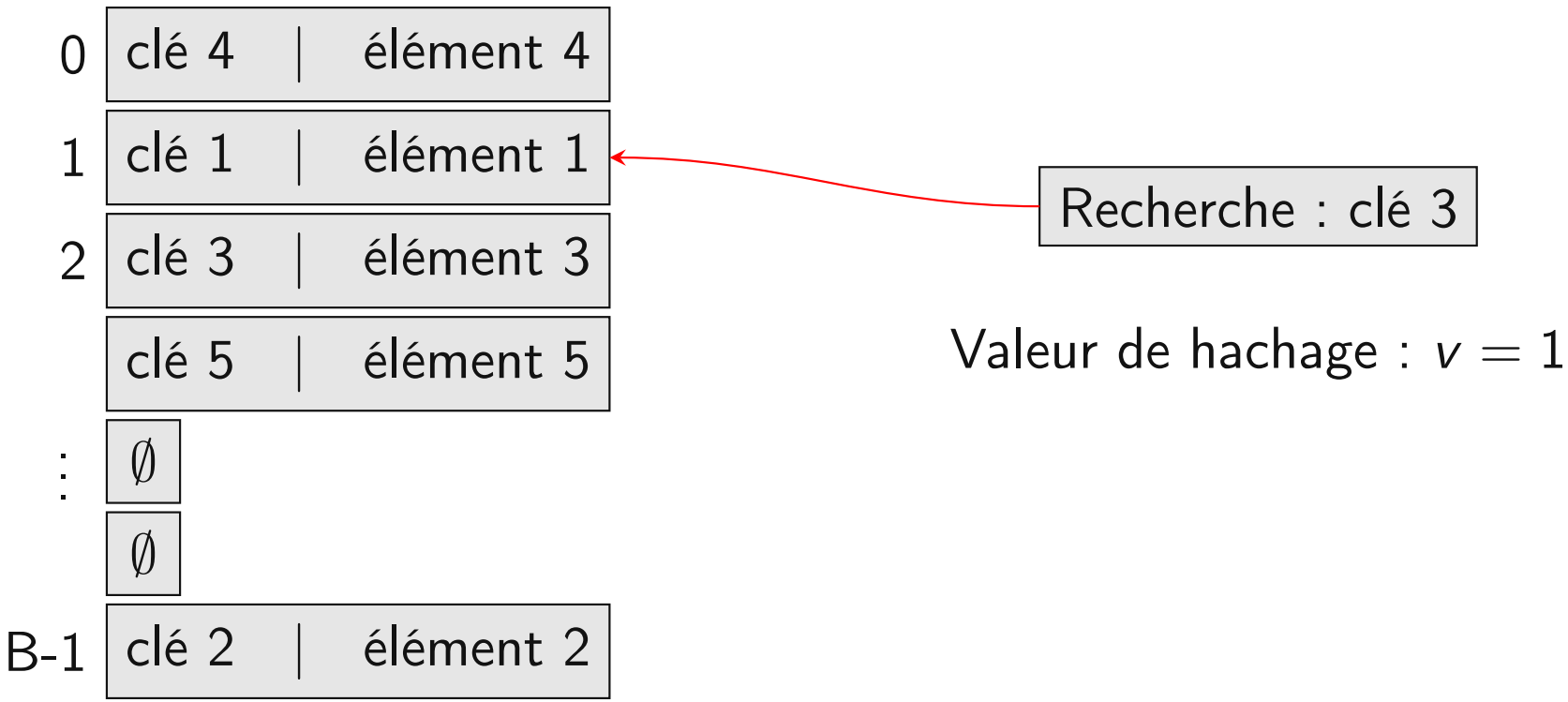
# Recherche et suppression

- **Recherche d'un élément**
  - Calculer la valeur de hachage $v$
  - Tant que $T[V] \neq \emptyset$ :
    - Si la clé recherchée = la clé dans $T[v]$ : Retourner $T[v]$
    - $v = (v + 1)\%B$
  - Retourner "non trouvée"
- **Suppression d'un élément**
  - Calculer la valeur de hachage
  - Tant que $T[v] \neq \emptyset$ :
    - Si la clé recherchée = la clé dans $T[v]$ : $T[v] \leftarrow$ *"supprimé"*
    - $v = (v + 1)\%B$

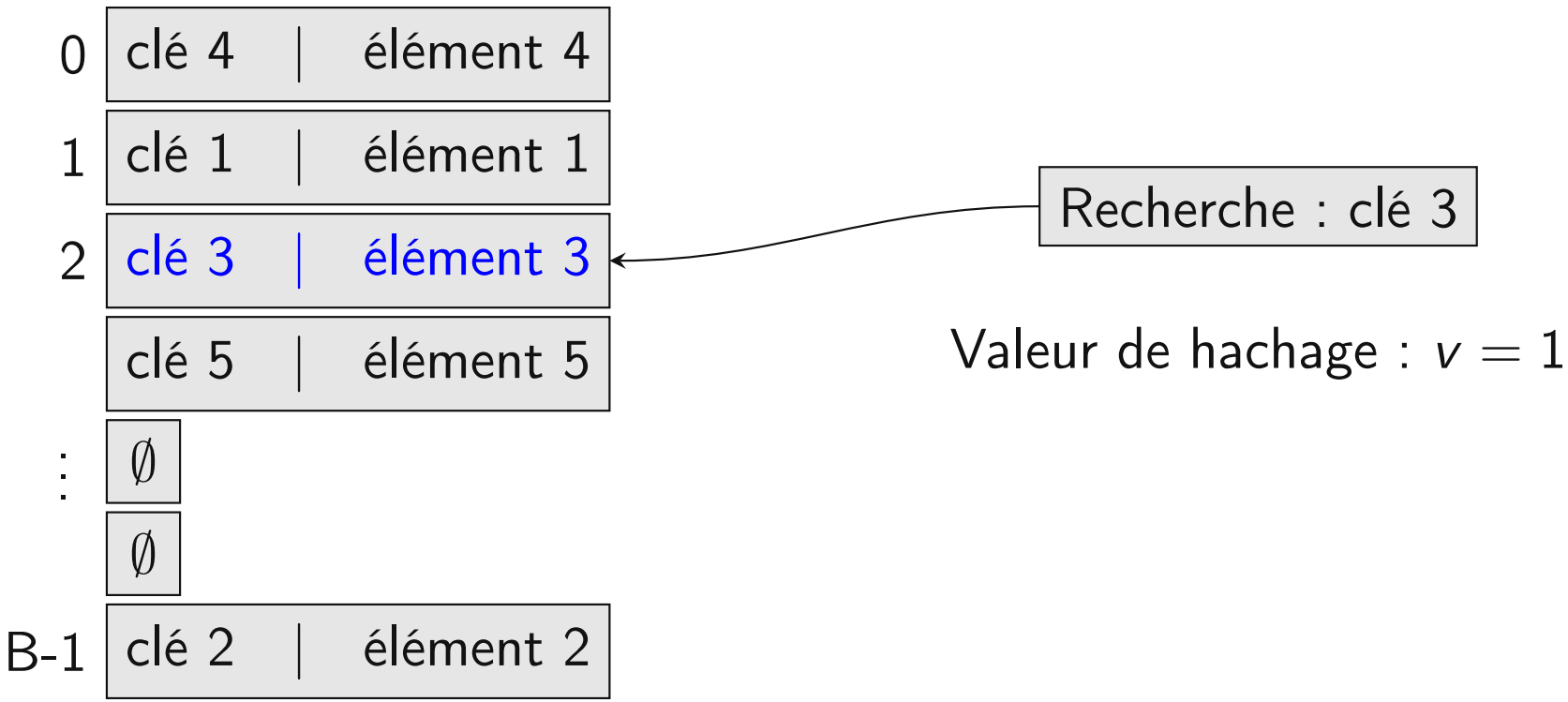
**Suppression**

On ne peut pas simplement vider une case, sinon la recherche ne fonctionnerait plus.

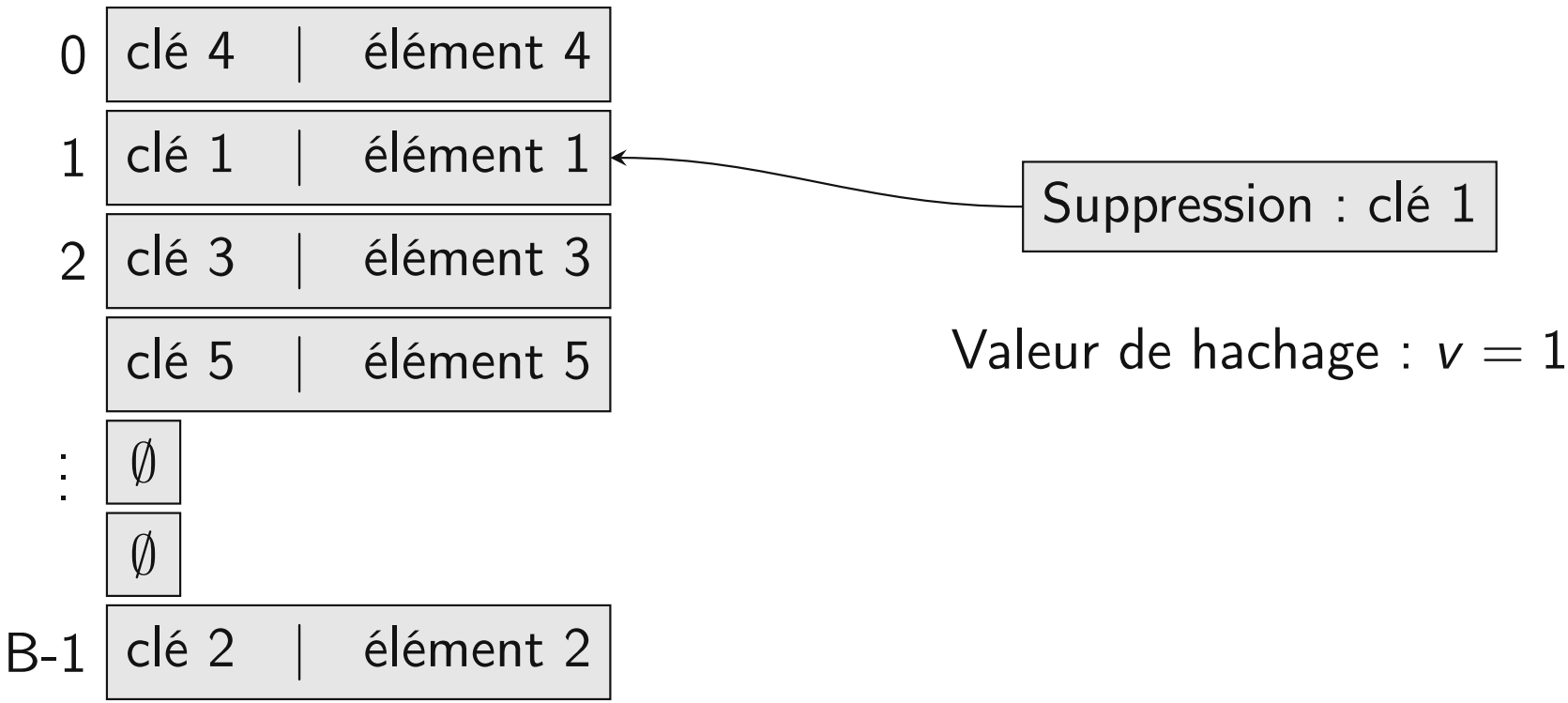
# Recherche et suppression



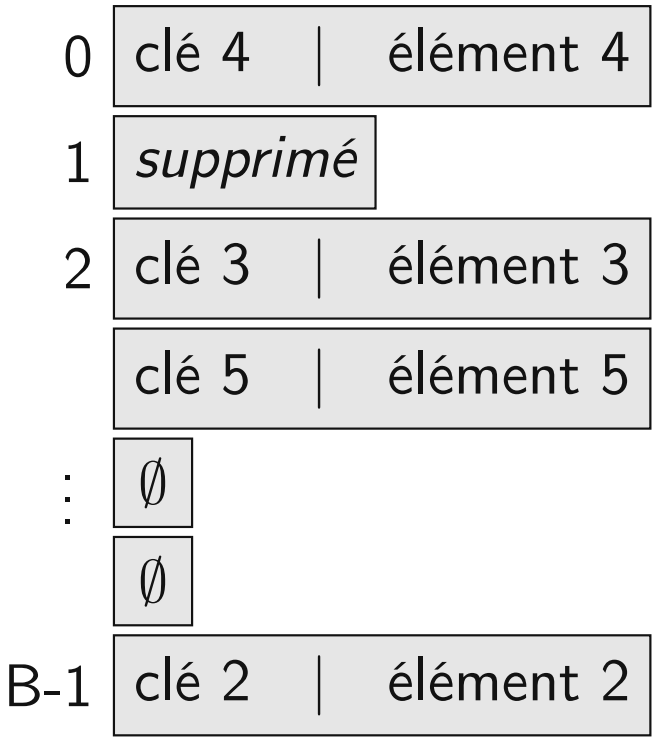
# Recherche et suppression



# Recherche et suppression



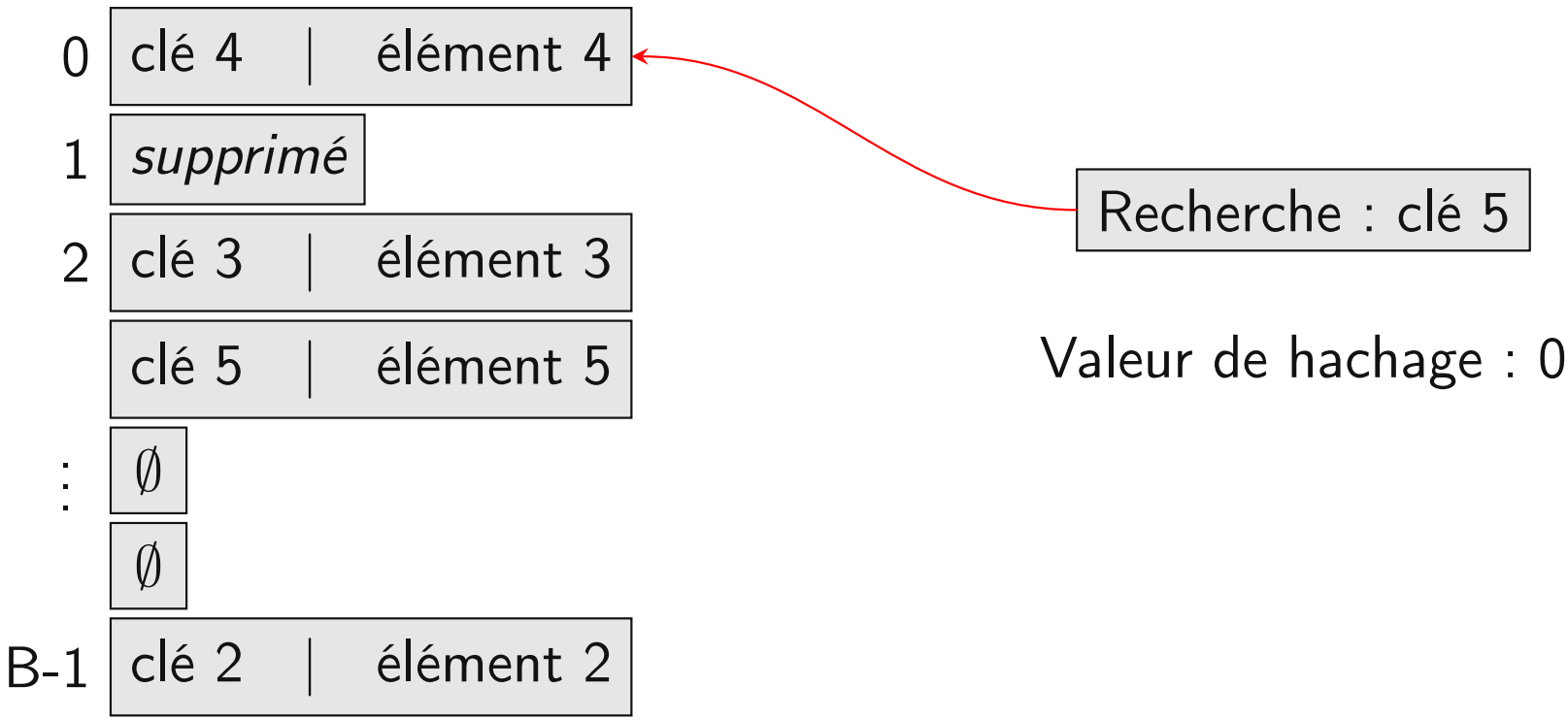
# Recherche et suppression



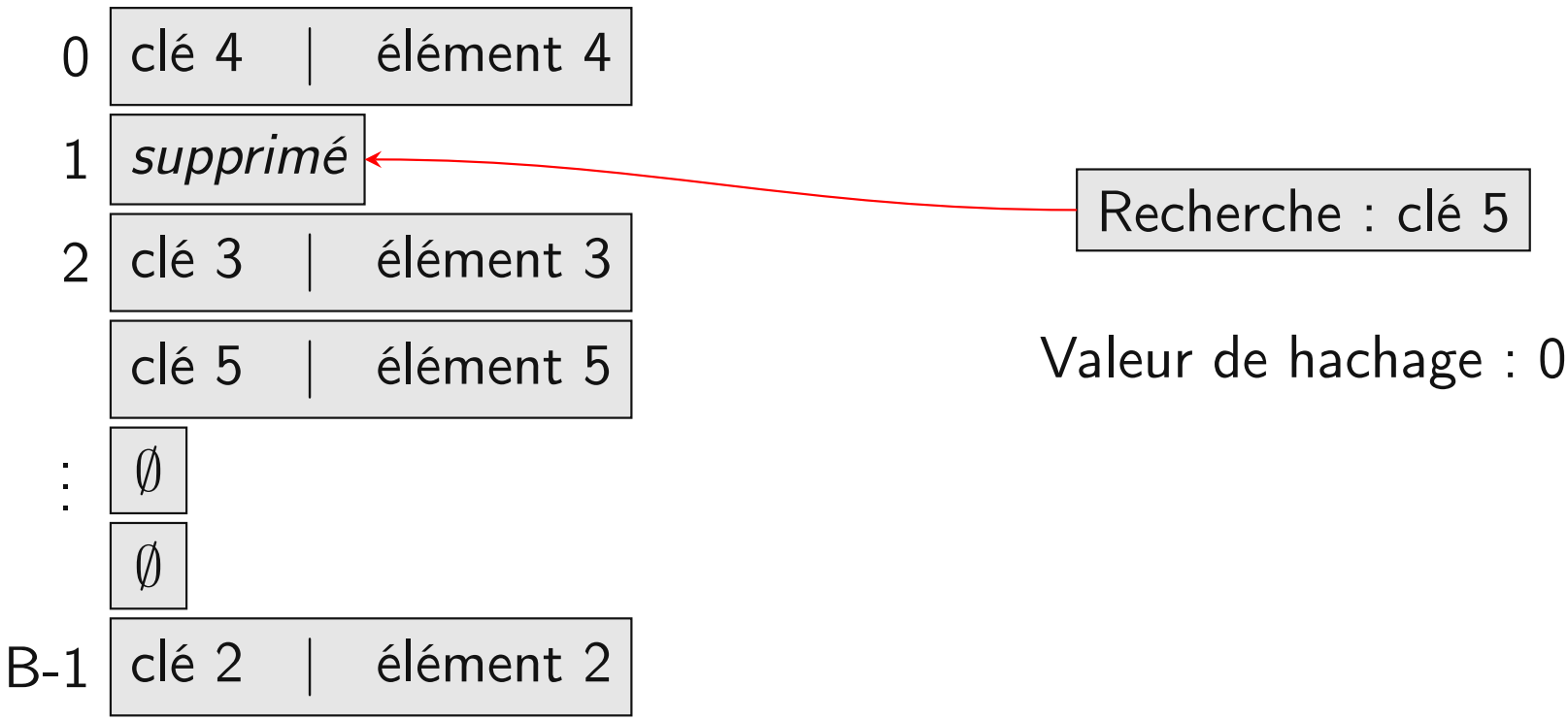

Suppression : clé 1

Valeur de hachage : $v = 1$

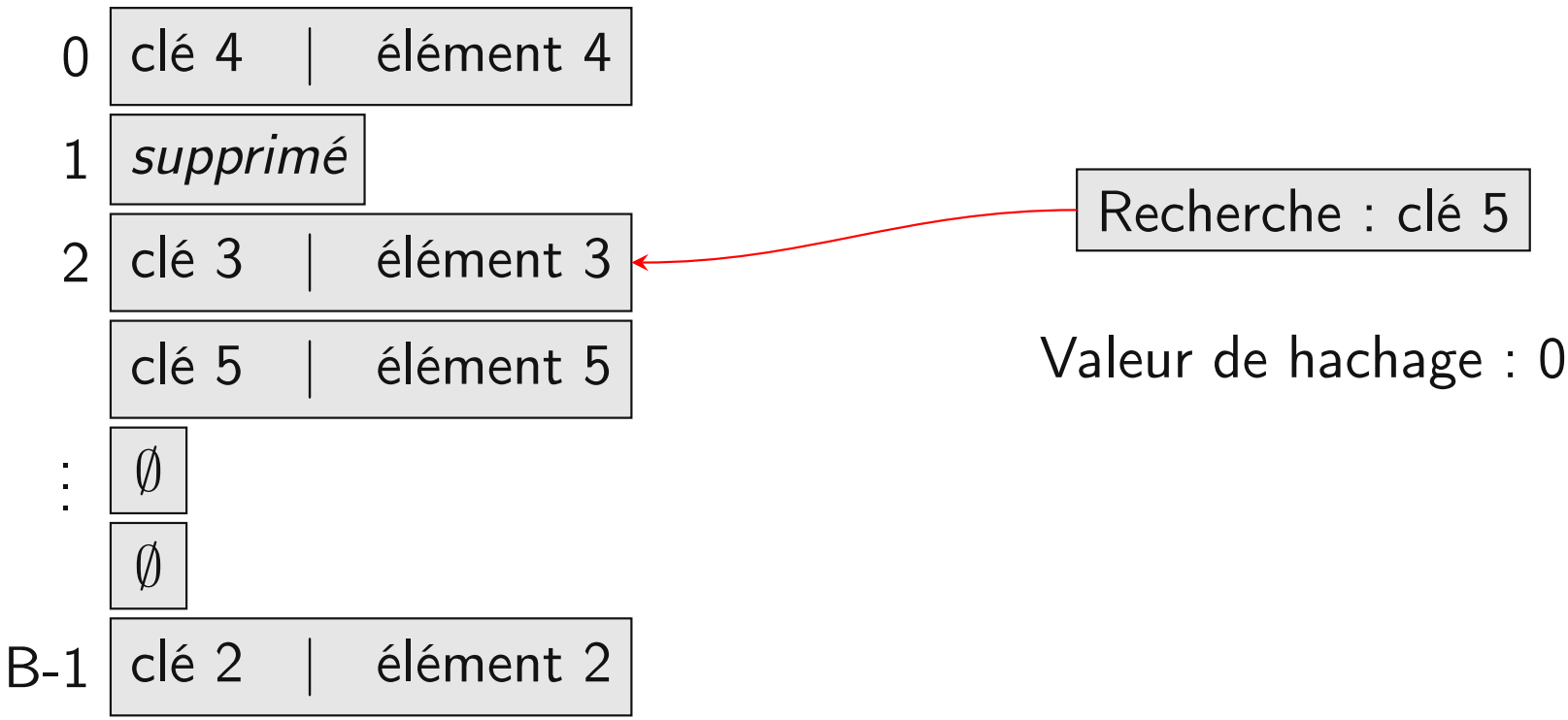
# Recherche et suppression



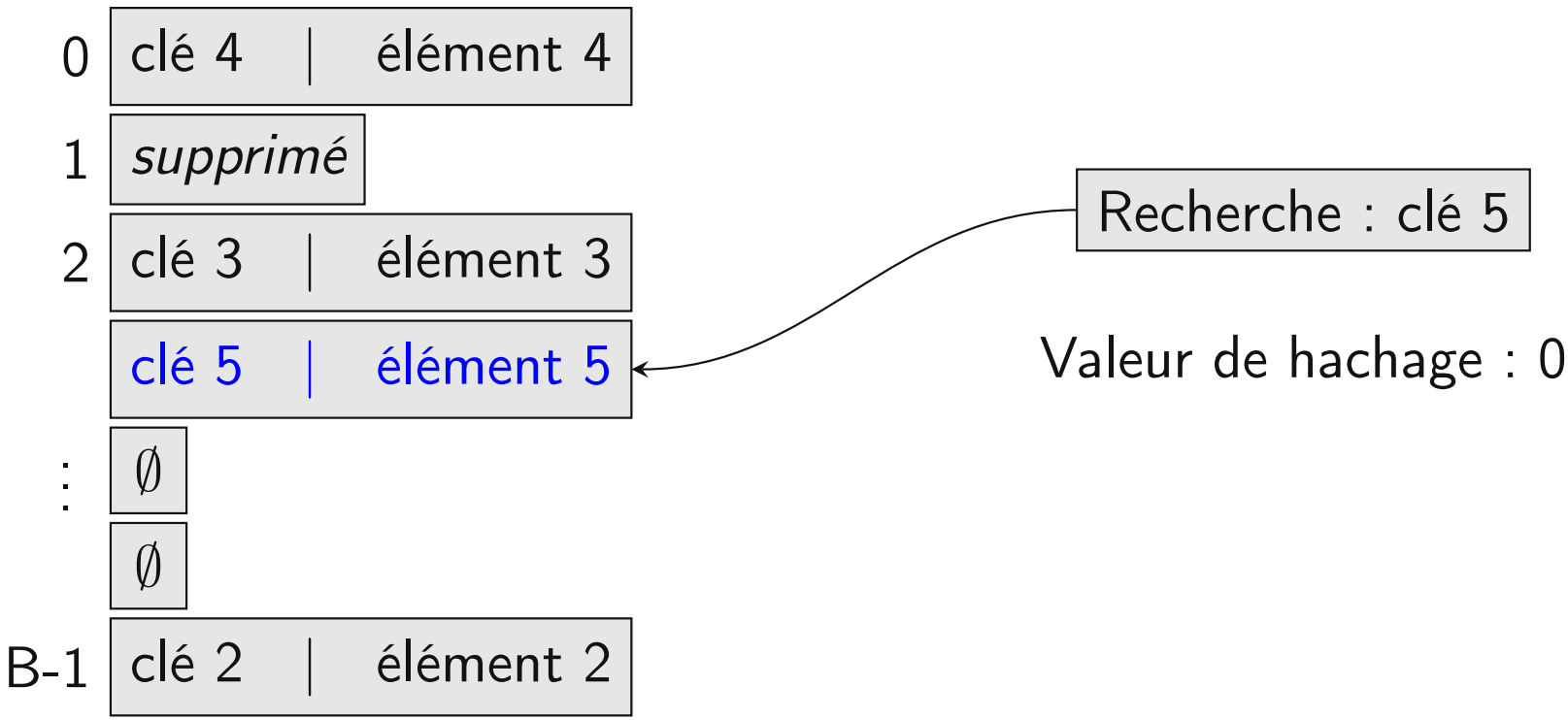
# Recherche et suppression



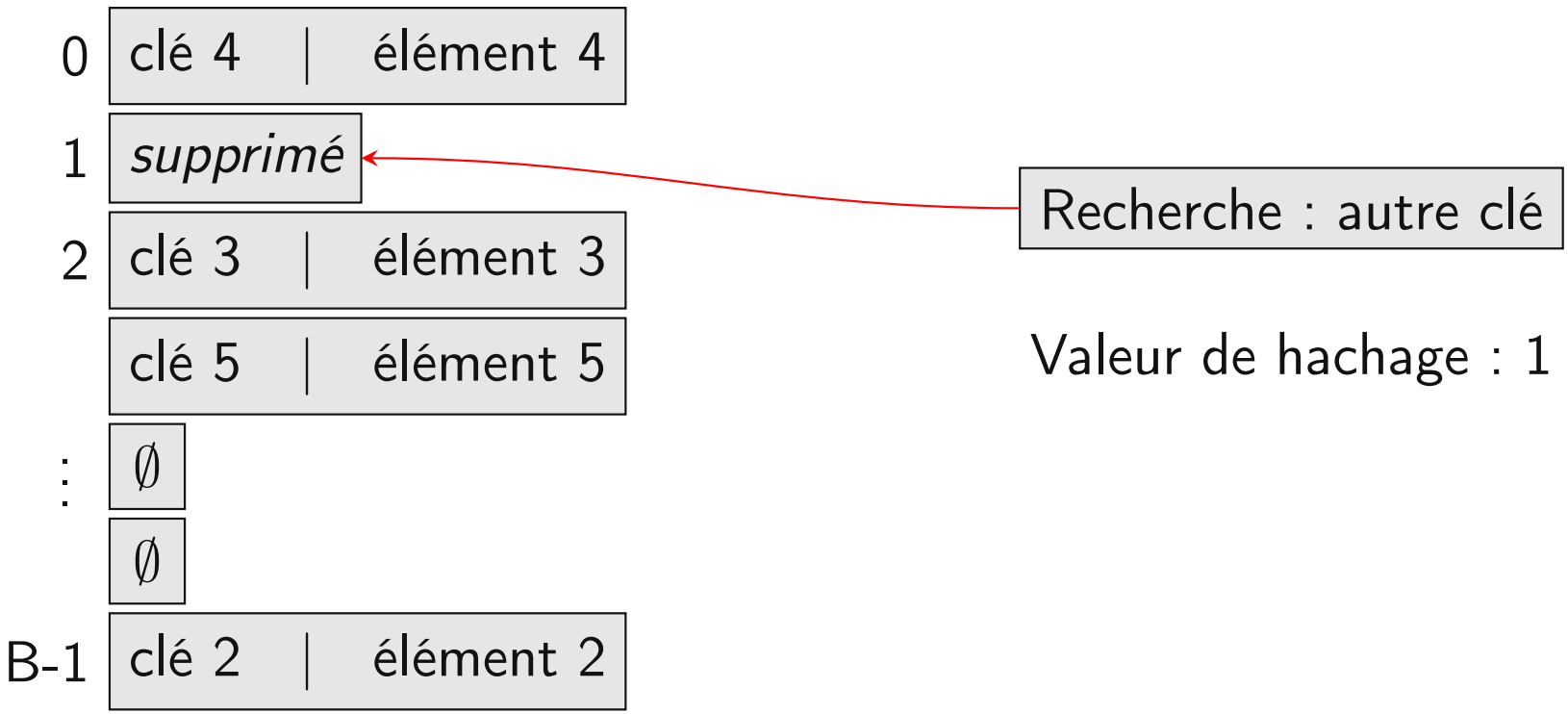
# Recherche et suppression



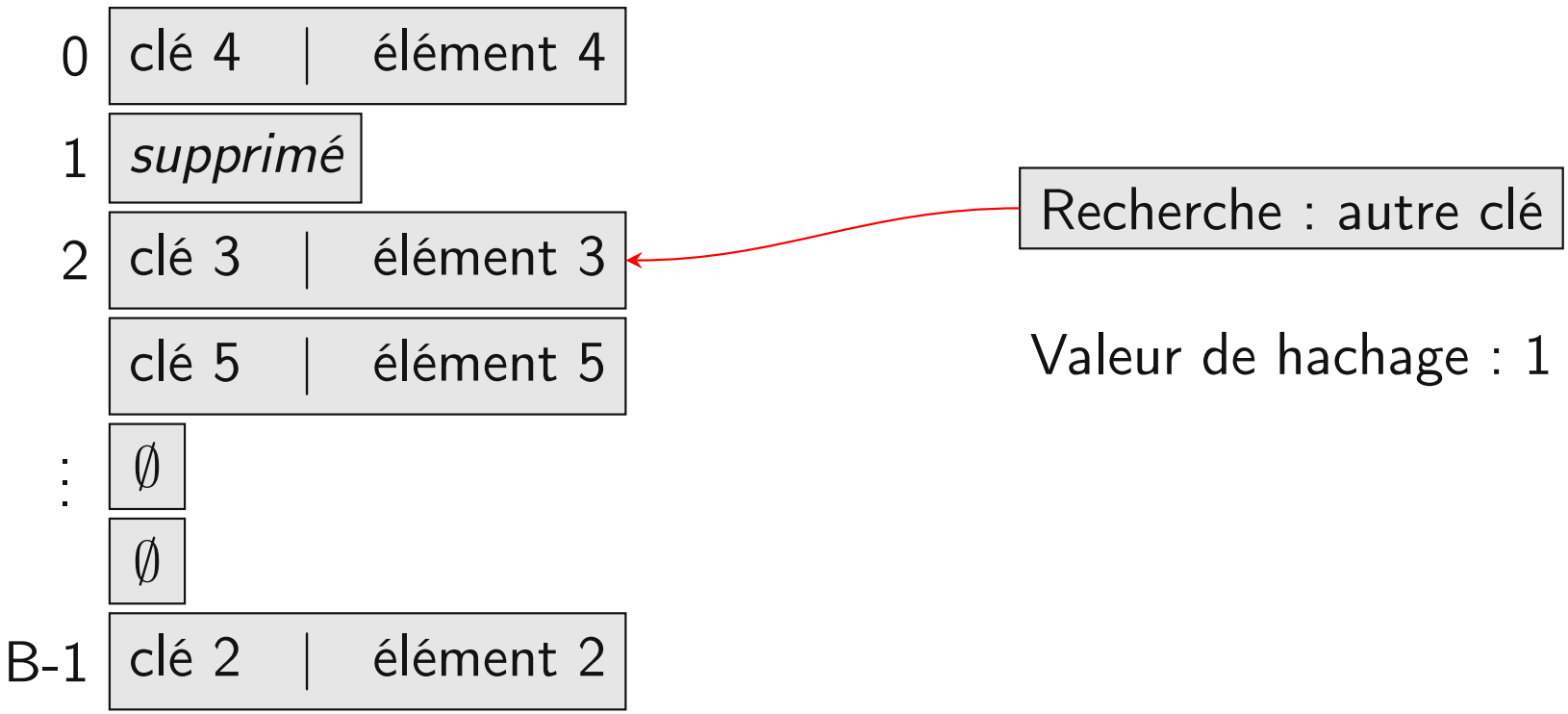
# Recherche et suppression



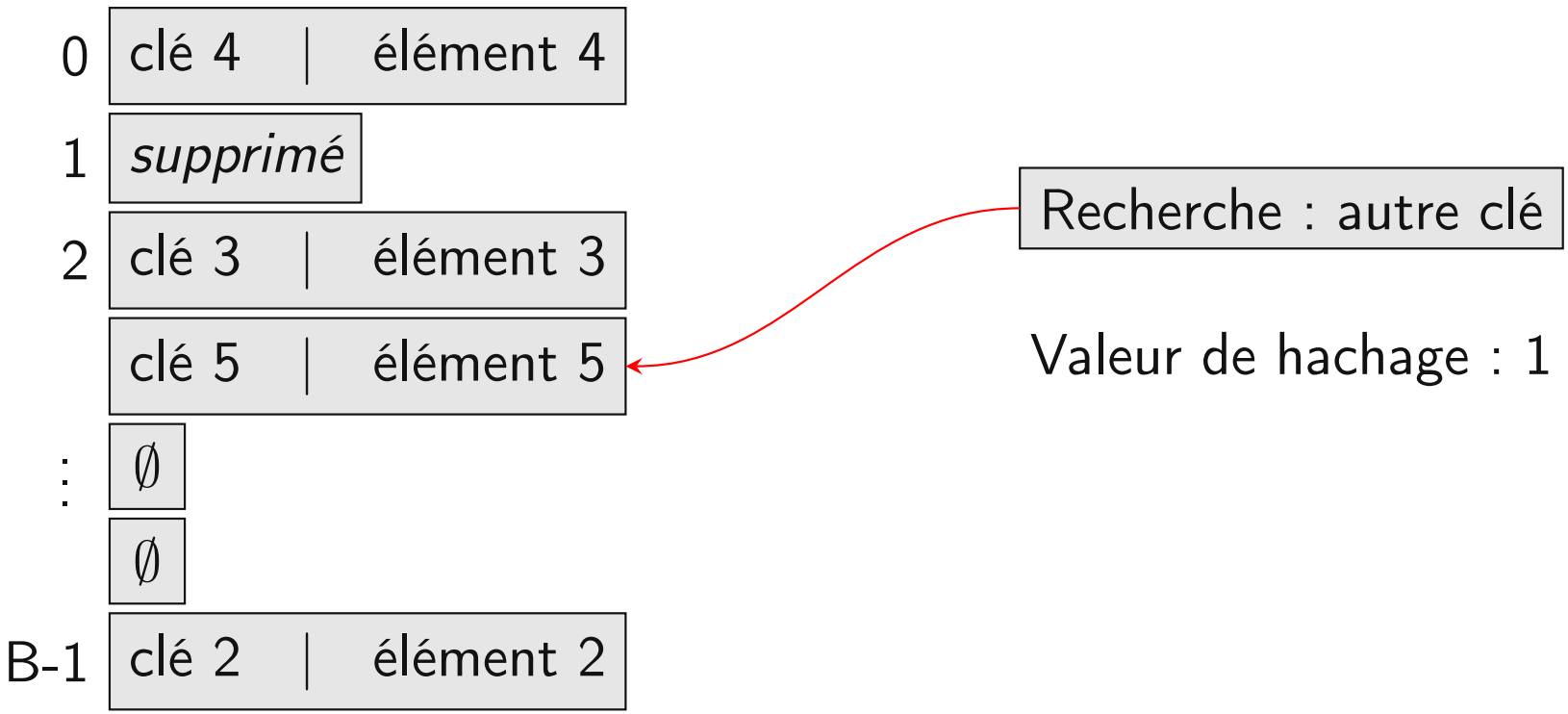
# Recherche et suppression



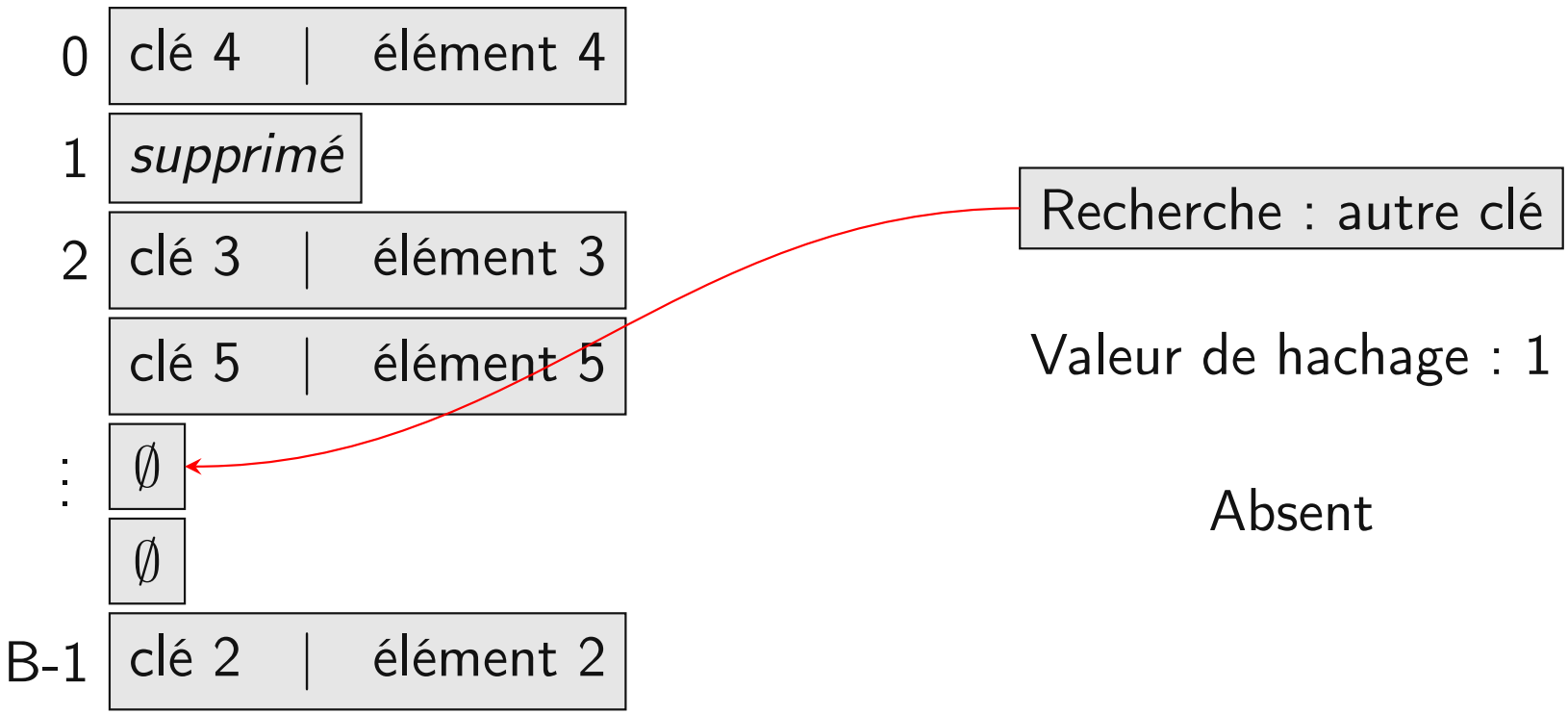
# Recherche et suppression



# Recherche et suppression



# Recherche et suppression



# Exercice 5 – Recherche et Suppression

Dans la table calculée précédemment, décrire les comparaisons de clés et les modifications effectuées lors des opérations suivantes :

1. Recherche de la clé **860**.
2. Recherche de la clé **452**.
3. Suppression de la clé **649**.
4. Recherche de la clé **324**.

# Pour aller plus loin...

- Plus compacte en mémoire que le hachage fermé, mais de taille limitée : $n \leq B$
- Pour éviter les congestions :
  - hachage linéaire : $v$, $v+1$, $v+2$, $v+3$, ...
  - hachage quadratique : $v$, $v+1$, $v+3$, $v+6$, ...
  - hachage (pseudo-)aléatoire : séquence différente pour chaque point de départ
- Nombre moyen de comparaisons $\leq k$ tant que $n \leq \frac{k-1}{k} B$ (pour le hachage aléatoire)

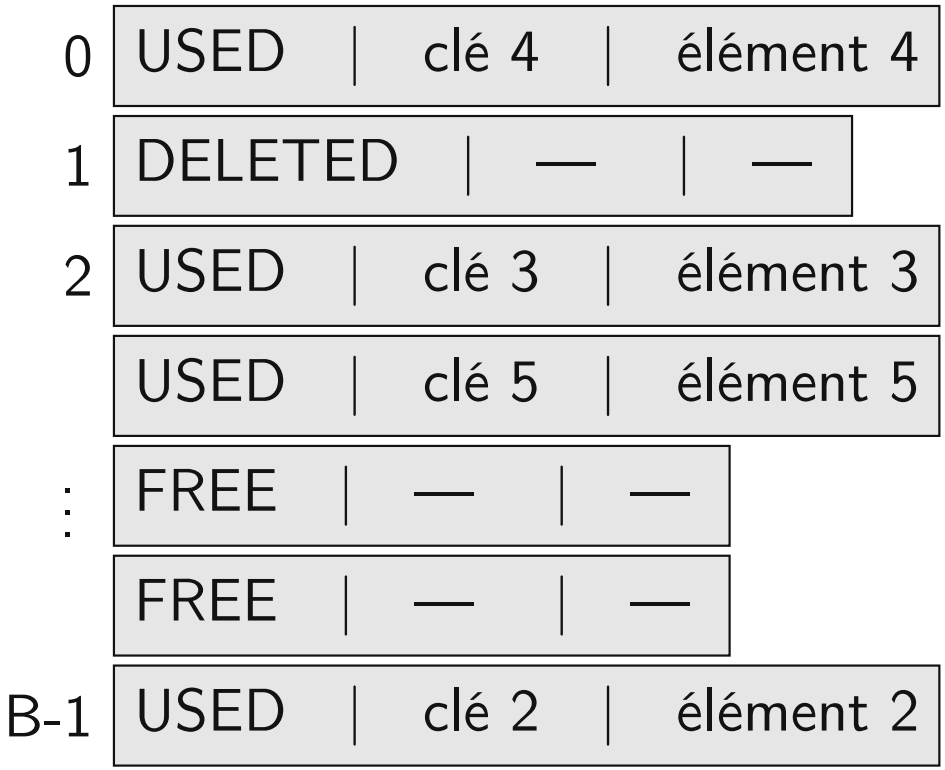
# Implémentation du hachage ouvert

```
#define table_size 9
typedef int Key_t;
typedef char* Value_t;

typedef enum {FREE, DELETED, USED} Status_t;
typedef struct {
    Status_t status;
    Key_t key;
    Value_t val;
} Cell;

typedef struct {
    int size;
    Cell tab[table_size];
} Hashtable;
```

# Recherche et suppression



# Exercice 6 – Implémentation du hachage ouvert

Implémenter les fonctions de manipulation sur les tables de hachage ouvert :

```
Value_t get(Hashtable* t, Key_t k);
void set(Hashtable* t, Key_t k, Value_t v);
void delete(Hashtable* t, Key_t k);
```

*À terminer en TP*