Hachage

Algorithmes de suppression

D.E ZEGOUR

École Supérieure d'Informatique

ESI

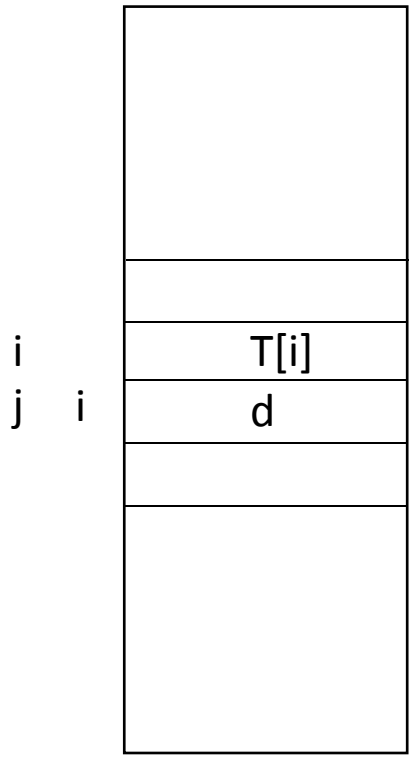
Hachage

Essai linéaire : suppression

Soit i l'adresse de l'élément à supprimer (d).

Adresse primaire de d : toutes les adresses possibles

h(d) = i
h(d) < i
h(d) > i



1. Rendre T(i) vide. Poser j := i.

2. i := i - 1; Si i < 0 : i := i + M Fsi

3. Si T(i) vide l'algorithme se termine.

4. Soit r := h (T(i))

Hachage

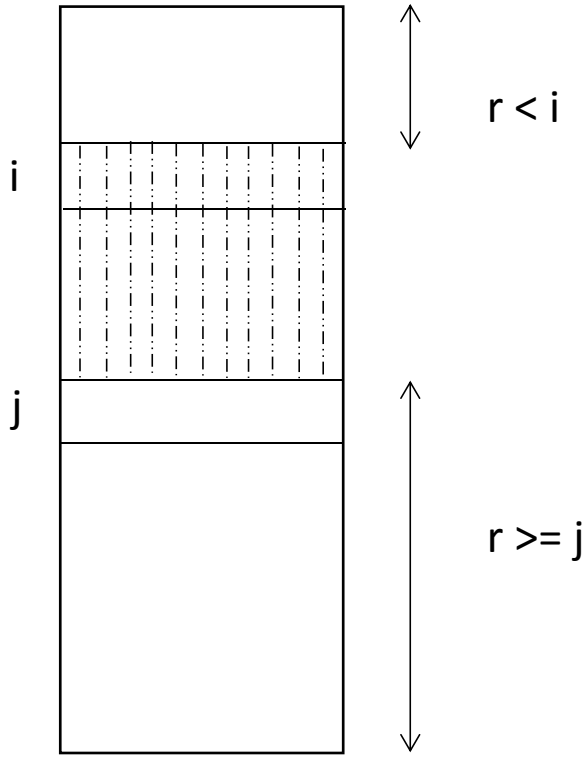
Essai linéaire : suppression

1. Rendre T(i) vide. Poser j := i.

2. i := i - 1; Si i < 0 : i := i + M Fsi

3. Si T(i) vide l'algorithme se termine.

4. Soit r := h (T(i))
CAS i < j
Si r < i ou r ≥ j :
    Déplacer l'élément, c'est à dire T(j) := T(i)
    Allera 1
Sinon
    Allera 2
Fsi



Hachage

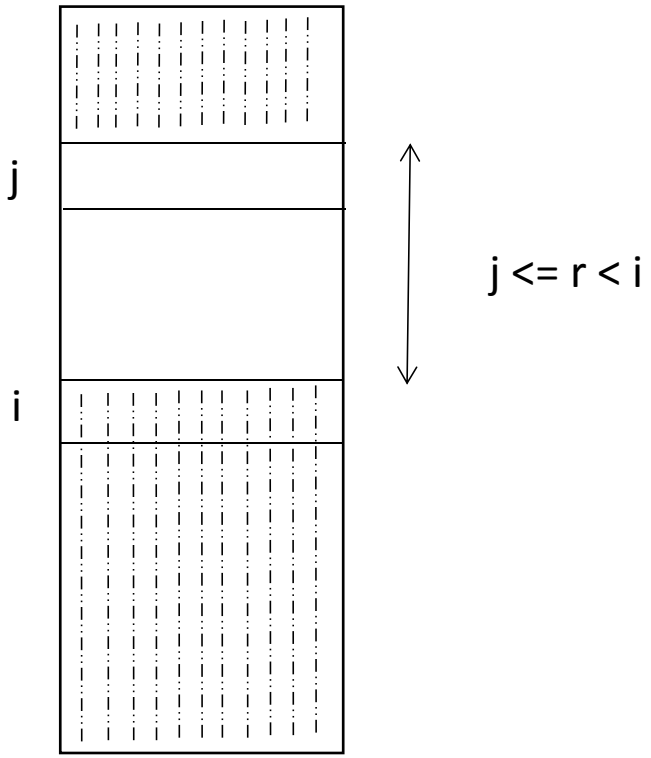
Essai linéaire : suppression

1. Rendre T(i) vide. Poser j := i.

2. i := i - 1; Si i < 0 : i := i + M Fsi

3. Si T(i) vide l'algorithme se termine.

4. Soit r := h (T(i))
CAS i > j
Si j ≤ r < i :
    Déplacer l'élément, c'est à dire T(j) := T(i)
    Allera 1
sinon
    Allera 2
Fsi



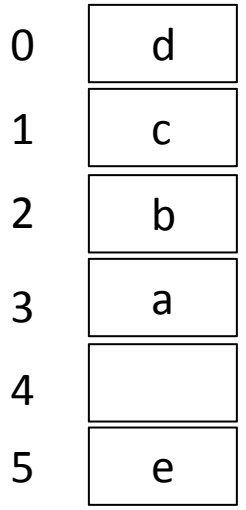
Hachage

Essai linéaire : suppression

Suppression de la donnée e
Adresses primaires : a(3), b(2), c(3), d(2), e(1)

Rendre la case vide

Case précédente vide : l'algorithme s'arête



Hachage

Essai linéaire : suppression

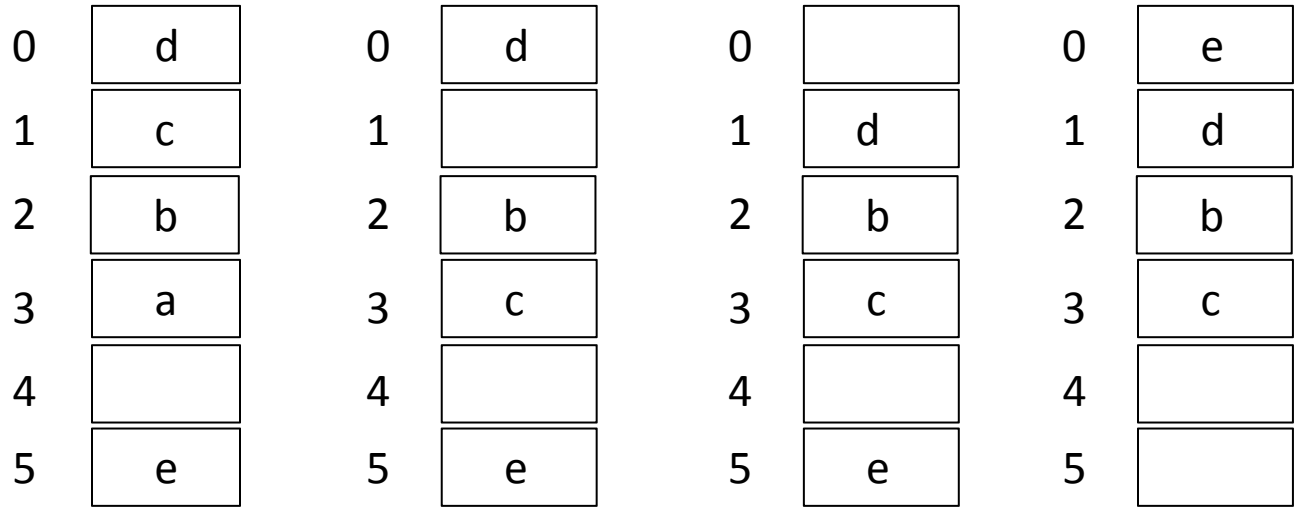
Suppression de la donnée a
Adresses primaires : a(3), b(2), c(3), d(2), e(1)

- Rendre la case 3 vide
- Comme la case 2 n'est pas vide, l'algo continue
- Comme b est dans son adresse primaire (h(b)=2)) il ne sera pas déplacé, l'algo continue

- L'adresse primaire de c est 3, c sera déplacé vers la position 3,l'algo continue puisque la case précédant c n'est pas vide

d sera déplacé vers 1

e sera déplacé vers 0



Hachage

Double hachage : suppression

On ne peut trouver un algorithme analogue à celui de l'essai linéaire.

Un moyen simple : une suppression logique (Ajouter un bit d'effacement)

Hachage

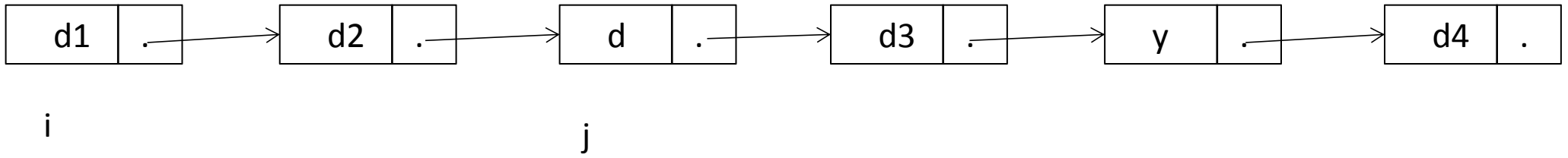
Chaînage interne : suppression

Soit à supprimer l'élément d.

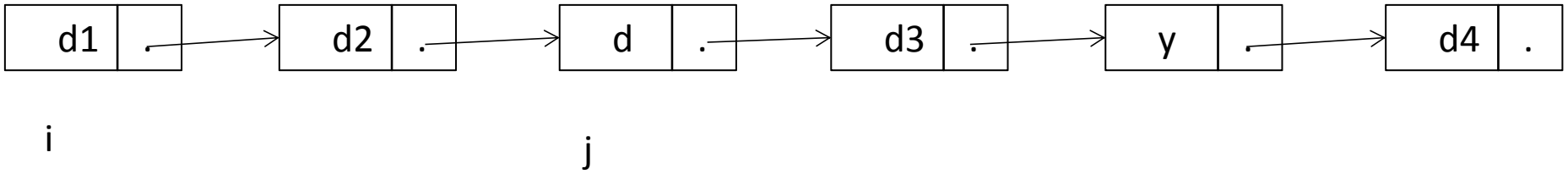
Rechercher l'élément.

Soit i son adresse primaire, et j l'indice de l'élément d. (i peut être égal à j)

i est donc la liste qui contient d.



Hachage



Chaînage interne : suppression

L'algorithme est le suivant :

1. Voir s'il existe plus loin (à partir de l'élémént j) dans la liste une donnée y telle la liste h(y) passe par j.

2. Si y n'existe pas, on supprime d de la liste en ajustant le chaînage et l'algorithme se termine.

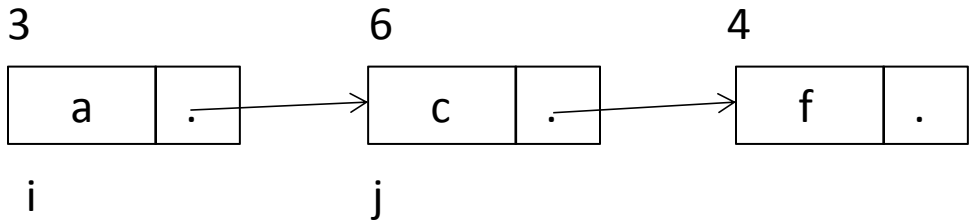
3. Si y existe, on le déplace à la position j. Poser j := indice de y et d := y et recommencer à partir de 1.

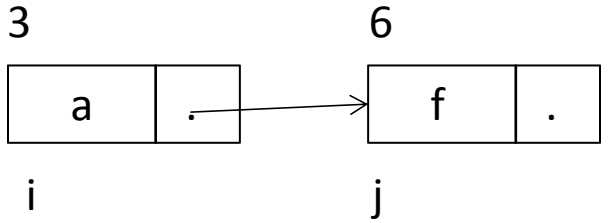
Dans les deux cas, on met à jour la variable R comme suit:
(k étant l'indice de l'élément supprimé) SI k > R : R := k + 1 FSI

Hachage

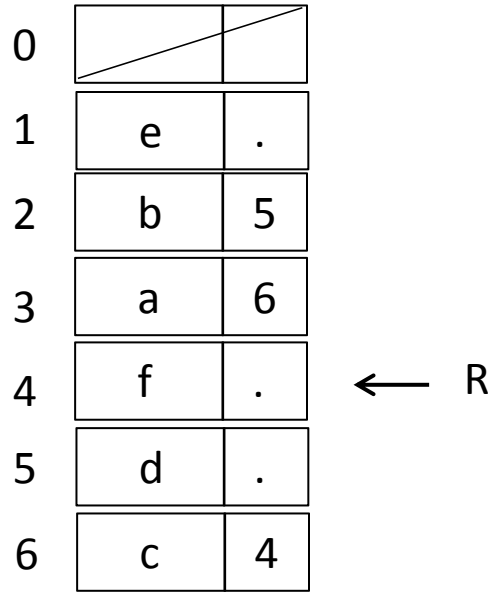
Chaînage interne

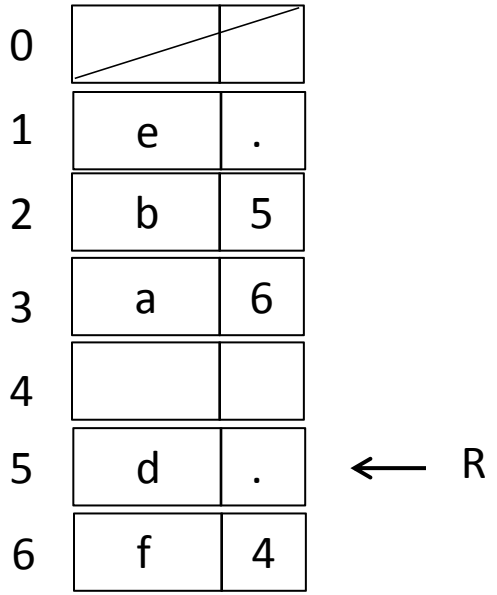
Suppression de c
Rappel : a(3), b(2), c(3), d(2), e(1), f(3)





- c est trouvé en j=6
- L'adresse primaire de c est i=3
- Il existe y tel que la liste h(y) passe par 6 (y=f)
- Comme la liste qui commence en 3 passe par j=6, f sera déplacé vers la position 6.





Hachage

Chaînage séparé

L'algorithme de suppression d'un élément est très simple. Ca consiste tout simplement à supprimer un élément d'une liste linéaire chaînée.