PC 4
Listes chaînées

- Listes
- Classes et objets
- Listes chaînées
- Piles
- Files

Listes

But : représenter des suites $(a, b, c, \ldots)$ d'éléments d'un ensemble de façon à pouvoir facilement insérer un nouvel élément (ou en supprimer un).

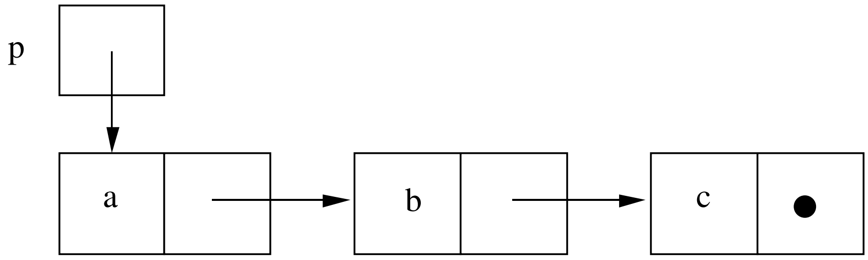


Figure 1: Liste chainée (a, b, c)

Exemples d'application

1. Le *hachage ouvert* associe à chaque clé $h(x)$ une liste.

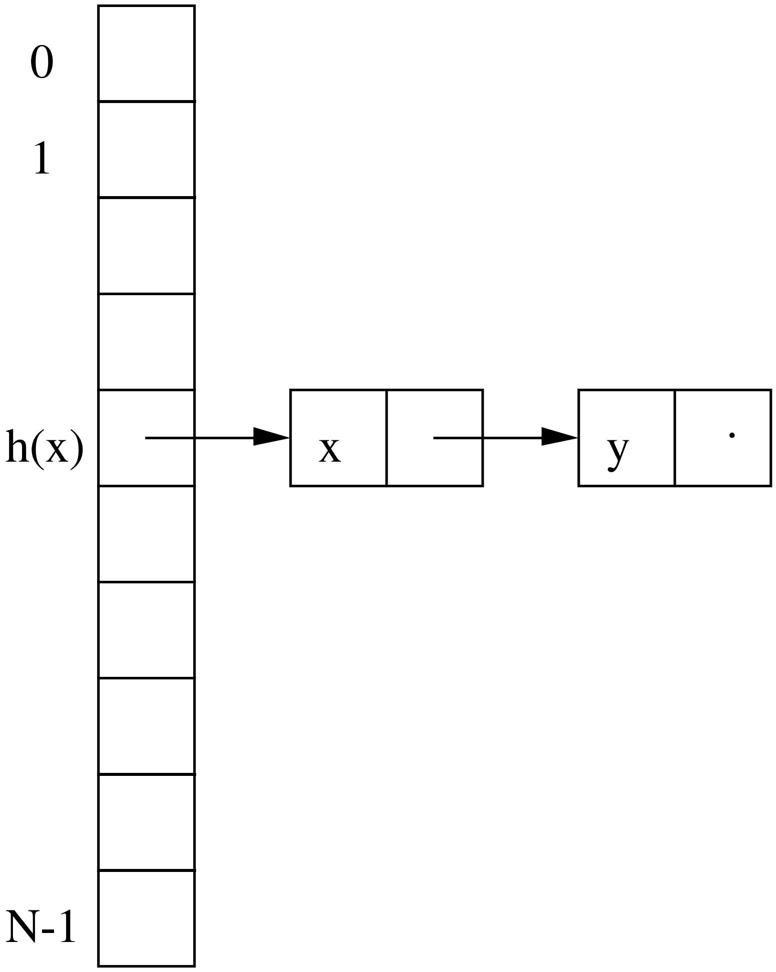


Figure 2: Hachage ouvert

2. L'espace mémoire sur disque est attribué par blocs de taille fixe et chaque fichier est représenté par une liste de blocs.

Classes et objets

Cahque objet en Java est une *instance* d'une classe. Par exemple, un objet de la classe

```
class Point {
  public double x,y;
}
```

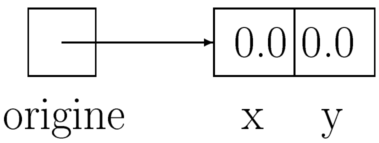
peut être créé par

```
Point origine = new Point();
```

et ses coordonnées initialisées par

```
origine.x = 0.0;
origine.y = 0.0;
```

Le nom de l'objet est une *référence* de cet objet.



Si un nom ne référence aucun objet, sa valeur est **`null`**.

Constructeurs

On crée des objets en utilisant des *constructeurs*. La déclaration

```
Point p;
```

ne crée pas d'objet, seulement une référence. On utilise un constructeur pour créer l'objet. Par exemple, si la classe Point contient le constructeur

```
Point(double a, double b) {
  x= a;
  y= b;
}
```

l'instruction

```
p=new Point(5.2,3);
```

crée un point de coordonnées $(5.2, 3)$.

Listes chaînées

```
class Liste {
  int contenu;
  Liste suivant;

  Liste (int x, Liste a) {
    contenu = x;
    suivant = a;
  }
}
```

Les instructions

```
 p=new Liste(5,null);
 p.suivant= new Liste(7,null);
```

créent la liste $(5, 7)$.

Insertion en tête de liste

```
static Liste ajouter (int x, Liste p) {
 return new Liste(x,p);
}
```

L'instruction `q = ajouter(x,p);` insère x en tête de la liste p.

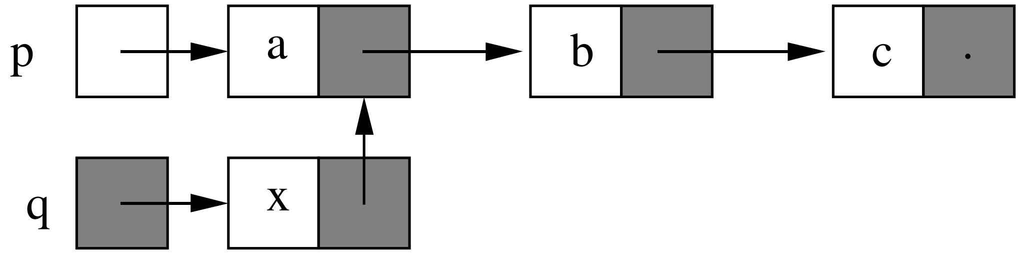


Figure 3:

Recherche d'un élément

```
static boolean recherche(int x, Liste p) {
  if (p == null)
    return false;
  else if (p.contenu == x)
    return true;
  else
    return recherche (x, p.suivant);
}
```

ou itérativement:

```
static boolean rechercheIter (int x, Liste p) {
  while (p != null) {
    if (p.contenu == x)
      return true;
    p = p.suivant;
  }
  return false;
}
```

Suppression d'un élément

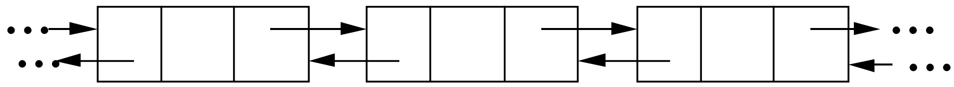
```
static Liste supprimer(int x, Liste a) {

  if (a != null)
    if (a.contenu == x)
      return a.suivant;
    else
      a.suivant= supprimer (x, a.suivant);
  return a;
}
```

Variantes

1. Liste avec cellule vide en tête: évite de traiter de façon spéciale l'insertion et la suppression en tête.

2. Liste doublement chaînée.



Possibilité de chaînage circulaire.

Piles

Liste avec accès restreint utilisée dans de très nombreux algorithmes. On insère et supprime toujours du même côté (lifo). La tête s'appelle traditionnellement le *sommet* de pile. Opérations de base:

- `Empiler(x,S)`: ajoute x en sommet de pile.
- `Depiler(S)`: supprime le sommet de pile.

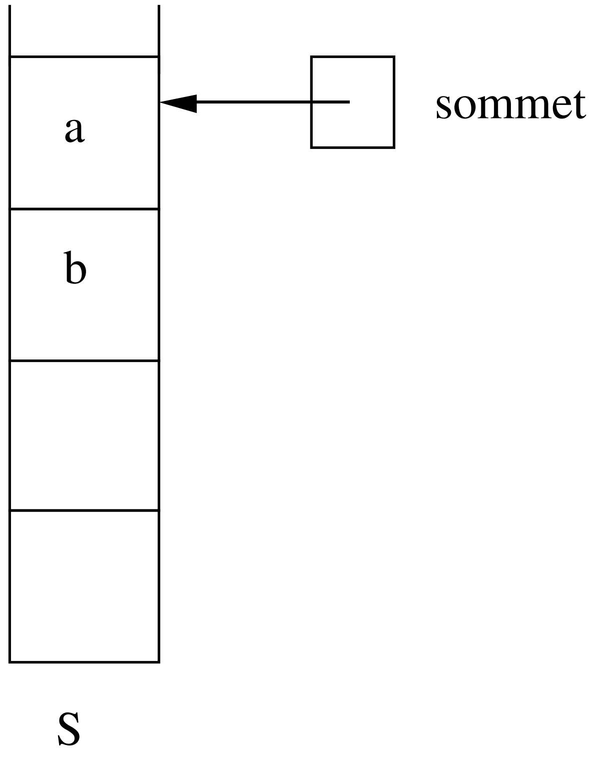


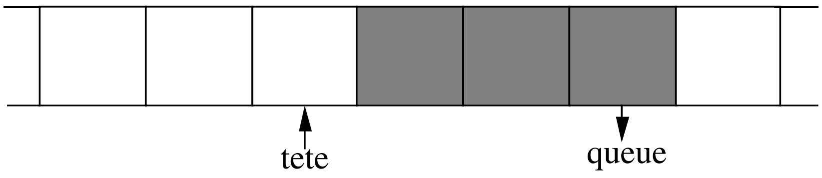
Figure 4:

Implémentation usuelle en tableau.

Files

Cette fois on insère d'un côté (la *queue*) et on supprime toujours de l'autre (la *tête*): stratégie *fifo*.

Implémentation possible: Dans un tableau avec gestion circulaire des indices.



File vide:

`queue = tete mod N`

File pleine:

`tete = queue + 1 mod N`