Cours n°5 : Structures de données : Tableaux et listes

Christophe Gonzales

HUGO3 — Algorithmique

But de la partie « Types et Structures »

Programmes informatiques

But de la partie « Types et Structures »

Programmes informatiques :

$\Longrightarrow$ traitements complexes sur des données complexes

But de la partie « Types et Structures »

Programmes informatiques :

$\Longrightarrow$ traitements complexes sur des données complexes

$\Longrightarrow$ nécessité d'utiliser des algorithmes efficaces

But de la partie « Types et Structures »

Programmes informatiques :

$\Longrightarrow$ traitements complexes sur des données complexes

$\Longrightarrow$ nécessité d'utiliser des algorithmes efficaces

$\Longrightarrow$ nécessité d'utiliser une « bonne » représentation des données

But de la partie « Types et Structures »

Programmes informatiques :

$\Longrightarrow$ traitements complexes sur des données complexes

$\Longrightarrow$ nécessité d'utiliser des algorithmes efficaces

$\Longrightarrow$ nécessité d'utiliser une « bonne » représentation des données

But de la partie « Types et Structures »

1. présenter les structures de données les plus classiquement utilisées en informatique.
2. faire appréhender les situations dans lesquelles telle ou telle structure est plus adaptée qu'une autre.

But de la partie « Types et Structures »

Programmes informatiques :

$\Longrightarrow$ traitements complexes sur des données complexes

$\Longrightarrow$ nécessité d'utiliser des algorithmes efficaces

$\Longrightarrow$ nécessité d'utiliser une « bonne » représentation des données

But de la partie « Types et Structures »

1. présenter les structures de données les plus classiquement utilisées en informatique.
2. faire appréhender les situations dans lesquelles telle ou telle structure est plus adaptée qu'une autre.

Objectif des structures de données

Organiser les informations pour y accéder, les créer, les détruire, le plus rapidement possible.

Les tableaux

Définition

- Structure regroupant un **nombre fixé** d'éléments **de même type**.
- Chaque élément identifié via un **index** (T[i]).
- Tableau stocké en mémoire de telle sorte que T[i] **accessible rapidement** (O(1)) à partir de i.

Les tableaux

Définition

- Structure regroupant un **nombre fixé** d'éléments **de même type**.
- Chaque élément identifié via un **index** (T[i]).
- Tableau stocké en mémoire de telle sorte que T[i] **accessible rapidement** (O(1)) à partir de i.

- **Exemple :** T = | 3 | 4 | -1 | 5 | 22 | 7 |

Les tableaux

Définition

- Structure regroupant un **nombre fixé** d'éléments **de même type**.
- Chaque élément identifié via un **index** (T[i]).
- Tableau stocké en mémoire de telle sorte que T[i] **accessible rapidement** (O(1)) à partir de i.

- **Exemple :** T = | 3 | 4 | -1 | 5 | 22 | 7 |

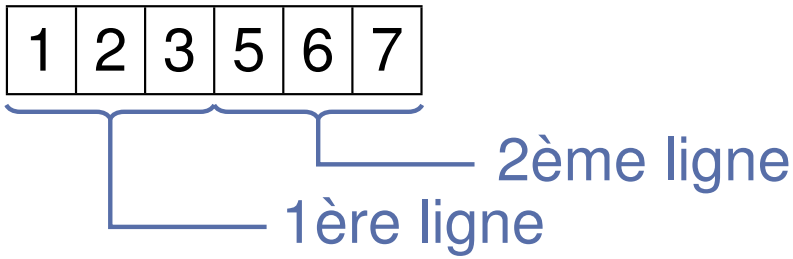
Tableaux 1D : éléments contigus en mémoire

Les tableaux 2D

En mathématiques :

- Matrice = $\begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \end{pmatrix}$

En informatique :

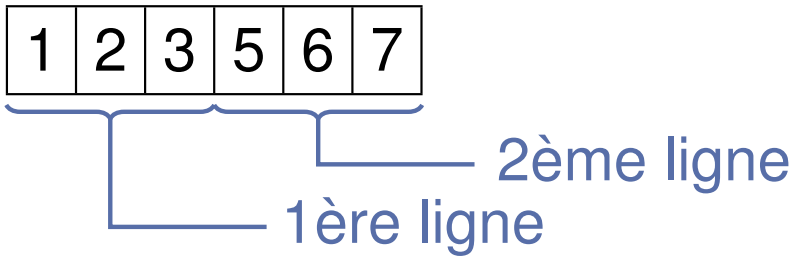
- T = 

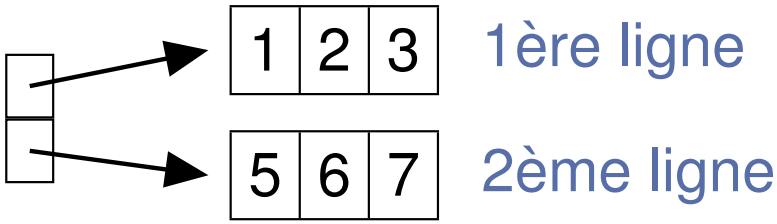
Les tableaux 2D

En mathématiques :

- Matrice = $\begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \end{pmatrix}$

En informatique :

- T = 

- T = 

$\Longrightarrow$ plusieurs représentations pour une même structure !

Caractéristiques d'une structure de données

Caractéristiques

- Opérations sur la structure ✔
- Implantation de la structure ✖

Caractéristiques d'une structure de données

Caractéristiques

- Opérations sur la structure ✔
- Implantation de la structure ✖

Opérations typiques

- Rajouter un élément dans la structure
- Supprimer un élément
- Chercher si un élément donné existe
- Accéder à l'élément le plus petit/le plus grand
- Accéder à un élément précis

Caractéristiques d'une structure de données

Caractéristiques

- Opérations sur la structure ✔
- Implantation de la structure ✖

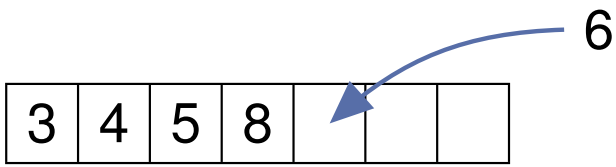
Opérations typiques

- Rajouter un élément dans la structure
- Supprimer un élément
- Chercher si un élément donné existe
- Accéder à l'élément le plus petit/le plus grand
- Accéder à un élément précis

Choisir la structure en fonction des opérations que l'on va utiliser

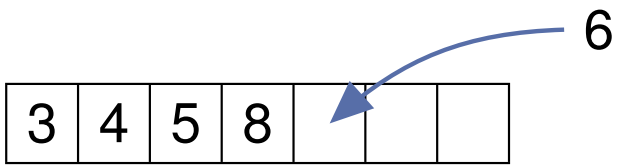
Tableaux : insertions/suppressions

✔ **Insertion en fin de tableau :** O(1)

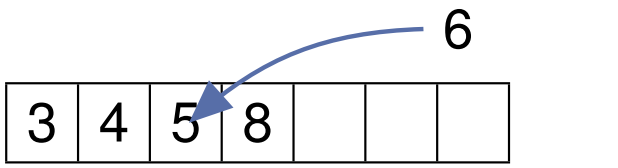


Tableaux : insertions/suppressions

✔ **Insertion en fin de tableau :** O(1)

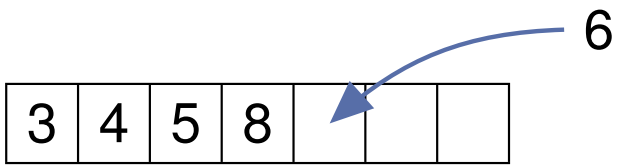


✖ **Insertion dans le courant du tableau :** O(n)

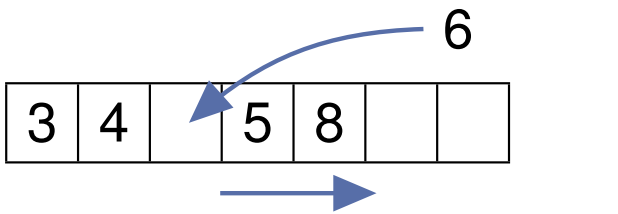


Tableaux : insertions/suppressions

✔ **Insertion en fin de tableau :** O(1)

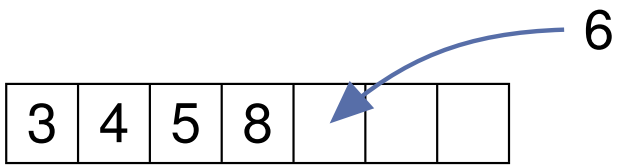


✖ **Insertion dans le courant du tableau :** O(n)

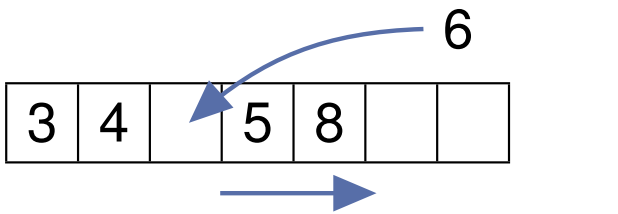


Tableaux : insertions/suppressions

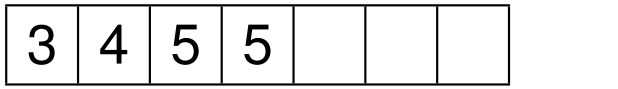
✔ **Insertion en fin de tableau :** O(1)



✖ **Insertion dans le courant du tableau :** O(n)

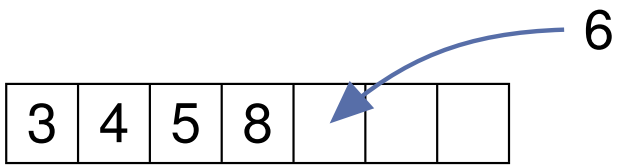


✔ **Suppression en fin de tableau :** O(1)

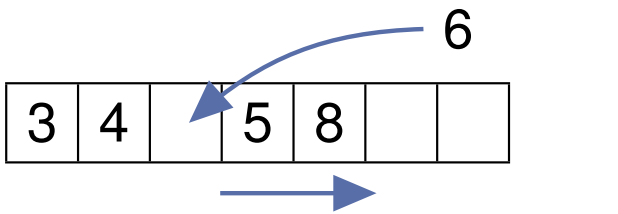


Tableaux : insertions/suppressions

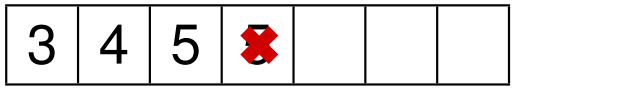
✔ **Insertion en fin de tableau :** O(1)



✖ **Insertion dans le courant du tableau :** O(n)

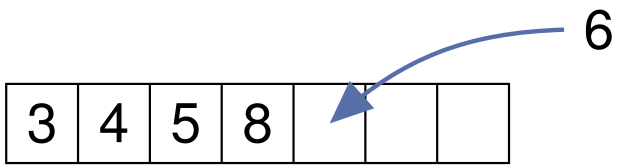


✔ **Suppression en fin de tableau :** O(1)

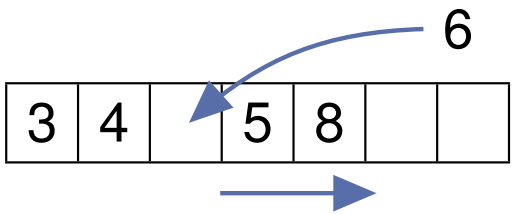


Tableaux : insertions/suppressions

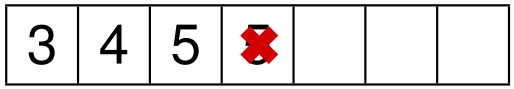
✔ **Insertion en fin de tableau :** O(1)



✖ **Insertion dans le courant du tableau :** O(n)



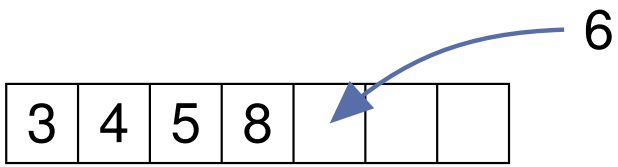
✔ **Suppression en fin de tableau :** O(1)



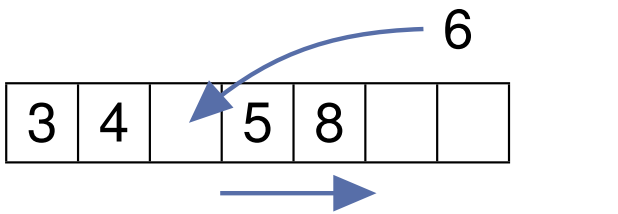
✖ **Suppression dans le courant du tableau :** O(n)

Tableaux : insertions/suppressions

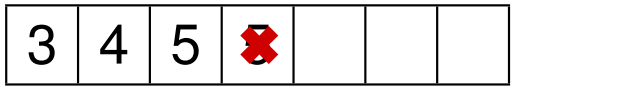
✔ **Insertion en fin de tableau :** O(1)



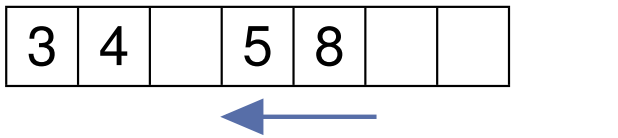
✖ **Insertion dans le courant du tableau :** O(n)



✔ **Suppression en fin de tableau :** O(1)

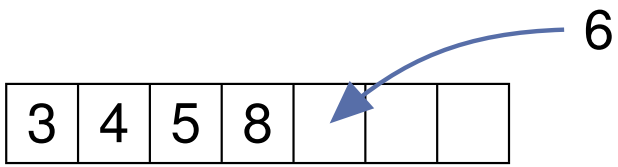


✖ **Suppression dans le courant du tableau :** O(n)

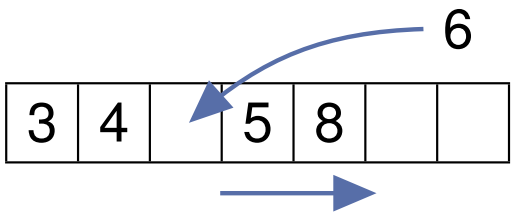


Tableaux : insertions/suppressions

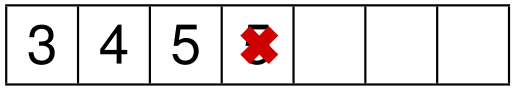
✔ **Insertion en fin de tableau :** O(1)



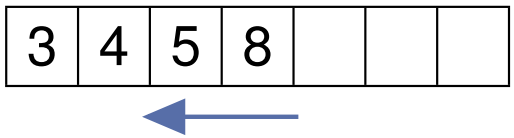
✖ **Insertion dans le courant du tableau :** O(n)



✔ **Suppression en fin de tableau :** O(1)

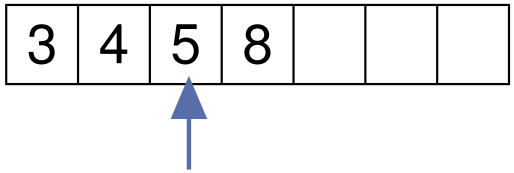


✖ **Suppression dans le courant du tableau :** O(n)



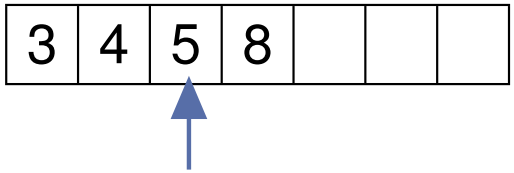
Tableaux : Recherche/Accès

✔ **Accéder au ième élément :** O(1)



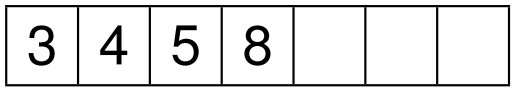
Tableaux : Recherche/Accès

✔ **Accéder au ième élément :** O(1)



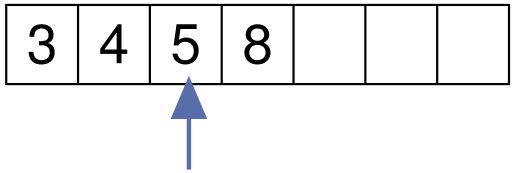
✖ **Accéder à l'élément le plus petit/plus grand :** O(n)

(mais O(1) si trié)



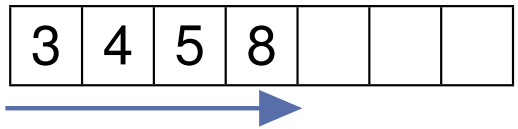
Tableaux : Recherche/Accès

✔ **Accéder au ième élément :** O(1)



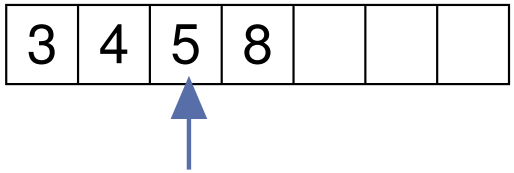
✖ **Accéder à l'élément le plus petit/plus grand :** O(n)

(mais O(1) si trié)



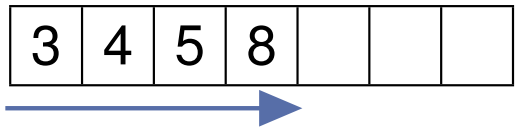
Tableaux : Recherche/Accès

✔ **Accéder au ième élément :** O(1)



✖ **Accéder à l'élément le plus petit/plus grand :** O(n)

(mais O(1) si trié)

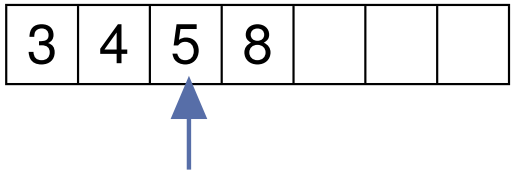


✖ **Chercher si un élément existe :** O(n)

(mais O(log n) si trié)

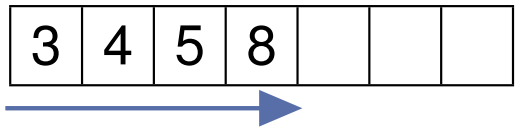
Tableaux : Recherche/Accès

✔ **Accéder au ième élément :** O(1)



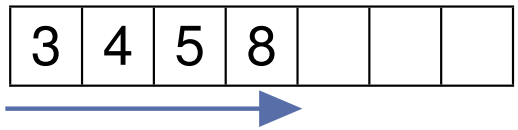
✖ **Accéder à l'élément le plus petit/plus grand :** O(n)

(mais O(1) si trié)



✖ **Chercher si un élément existe :** O(n)

(mais O(log n) si trié)



Tableaux : synthèse

Opération	Complexité	Intérêt
Insertion en fin de tableau	O(1)	✔
Insertion dans le courant du tableau	O(n)	✖
Suppression en fin de tableau	O(1)	✔
Suppression dans le courant du tableau	O(n)	✖
Accéder au ième élément	O(1)	✔
Accéder à l'élément le plus petit/le plus grand	O(n)	✖
Chercher si un élément existe	O(n)	✖

Les listes

Définition

- Collection ordonnée d'éléments $\langle x_1 ; \cdots ; x_n \rangle$
- x_1 accessible en O(1)

Les listes

Définition

- Collection ordonnée d'éléments $\langle x_1 ; \cdots ; x_n \rangle$
- x_1 accessible en O(1)
- Si on accède à x_i, x_{i+1} accessible en O(1)
- Dans certaines listes, si on accède à x_i, x_{i-1} accessible en O(1)

Les listes

Définition

- Collection ordonnée d'éléments $\langle x_1 ; \cdots ; x_n \rangle$
- x_1 accessible en O(1)
- Si on accède à x_i, x_{i+1} accessible en O(1)
- Dans certaines listes, si on accède à x_i, x_{i-1} accessible en O(1)

Cette définition ne présuppose rien sur la représentation en mémoire.

Les listes

Définition

- Collection ordonnée d'éléments $\langle x_1 ; \cdots ; x_n \rangle$
- x_1 accessible en O(1)
- Si on accède à x_i, x_{i+1} accessible en O(1)
- Dans certaines listes, si on accède à x_i, x_{i-1} accessible en O(1)

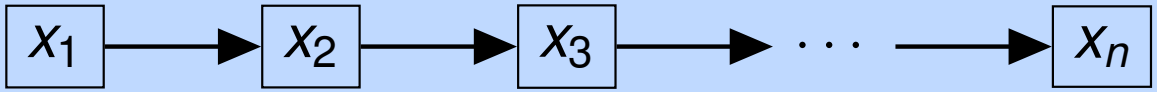
Cette définition ne présuppose rien sur la représentation en mémoire.

- En principe, pas de contrainte sur le nombre d'éléments dans la liste.

Les listes simplement chaînées

Définition

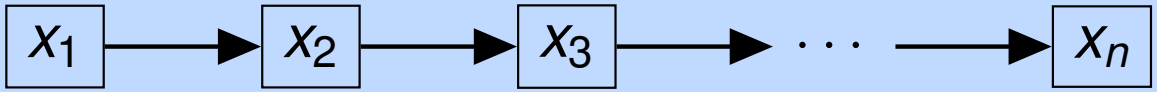
- Définition récursive : une liste est composée d'un élément (*la tête*) suivi par la liste des éléments suivants (*la queue*).
- Illustration graphique :



Les listes simplement chaînées

Définition

- Définition récursive : une liste est composée d'un élément (*la tête*) suivi par la liste des éléments suivants (*la queue*).
- Illustration graphique :



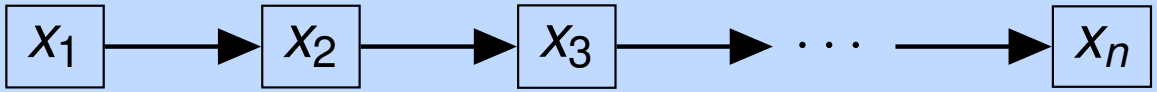
- *Les seuls opérateurs :*
  Un opérateur pour **accéder à la tête** :
  `car` en LISP/Scheme, `hd` en ocaml.
  Un opérateur pour **accéder à la queue** :
  `cdr` en LISP/Scheme, `tl` en ocaml

toujours faire un dessin indiquant comment sont chaînés les éléments.

Les listes simplement chaînées

Définition

- Définition récursive : une liste est composée d'un élément (*la tête*) suivi par la liste des éléments suivants (*la queue*).
- Illustration graphique :



- *Les seuls opérateurs :*
  Un opérateur pour **accéder à la tête** :
  `car` en LISP/Scheme, `hd` en ocaml.
  Un opérateur pour **accéder à la queue** :
  `cdr` en LISP/Scheme, `tl` en ocaml

toujours faire un dessin indiquant comment sont chaînés les éléments.

Vous pensez pouvoir vous en passer ? OK, vous aurez des bugs. . .

Listes chaînées : insertions

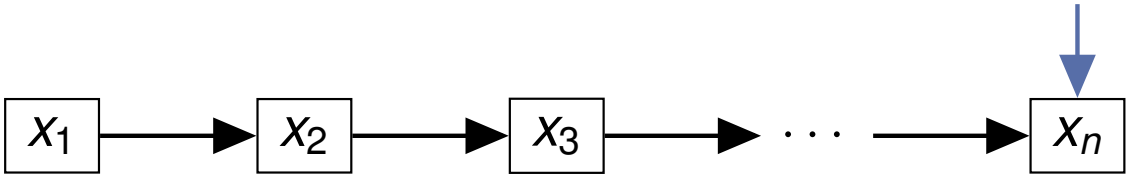
Souvent les flèches sont des pointeurs
$\Longrightarrow$ en mémoire, les boites ne sont pas placées les unes à côté des autres

Listes chaînées : insertions

Souvent les flèches sont des pointeurs
$\Longrightarrow$ en mémoire, les boites ne sont pas placées les unes à côté des autres

✖ **Insertion en fin de liste :** O(n)

O(1) si on a accès au dernier elt

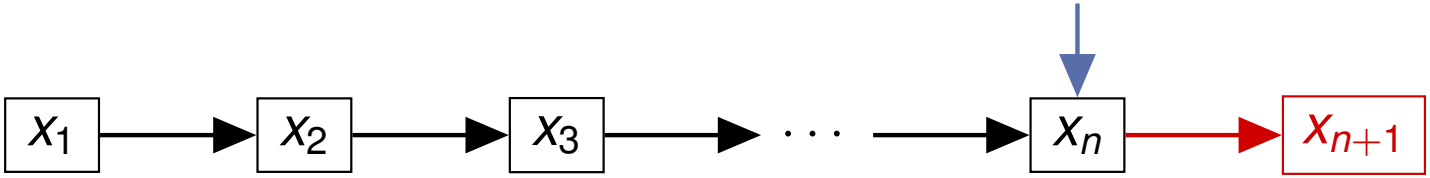


Listes chaînées : insertions

Souvent les flèches sont des pointeurs
$\Longrightarrow$ en mémoire, les boites ne sont pas placées les unes à côté des autres

✖ **Insertion en fin de liste :** O(n)

O(1) si on a accès au dernier elt

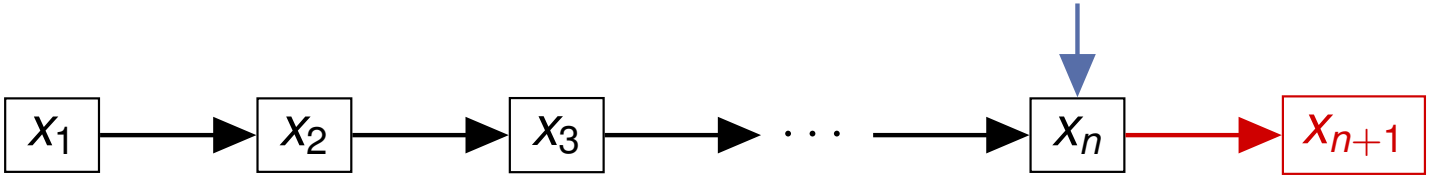


Listes chaînées : insertions

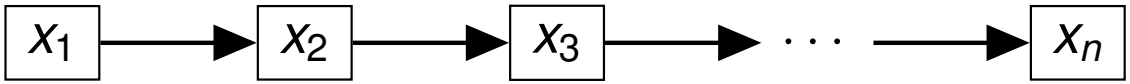
Souvent les flèches sont des pointeurs
$\Longrightarrow$ en mémoire, les boites ne sont pas placées les unes à côté des autres

✖ **Insertion en fin de liste :** O(n)

O(1) si on a accès au dernier elt



✔ **Insertion en début de liste :** O(1)

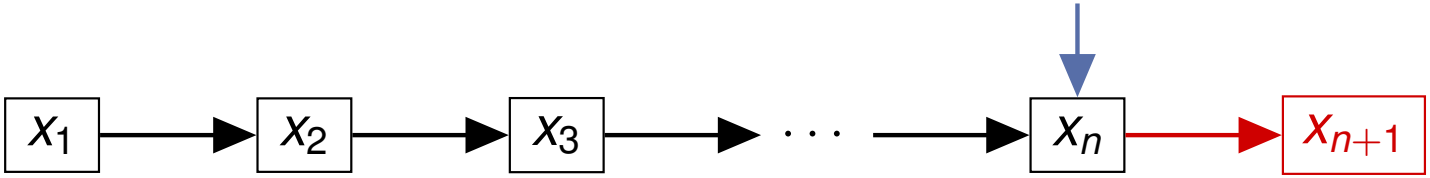


Listes chaînées : insertions

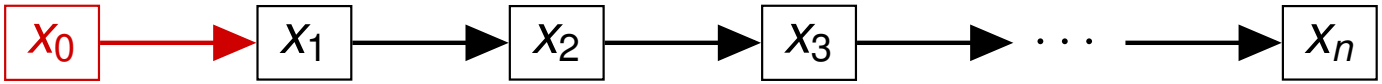
Souvent les flèches sont des pointeurs
$\Longrightarrow$ en mémoire, les boites ne sont pas placées les unes à côté des autres

✖ **Insertion en fin de liste :** O(n)

O(1) si on a accès au dernier elt



✔ **Insertion en début de liste :** O(1)

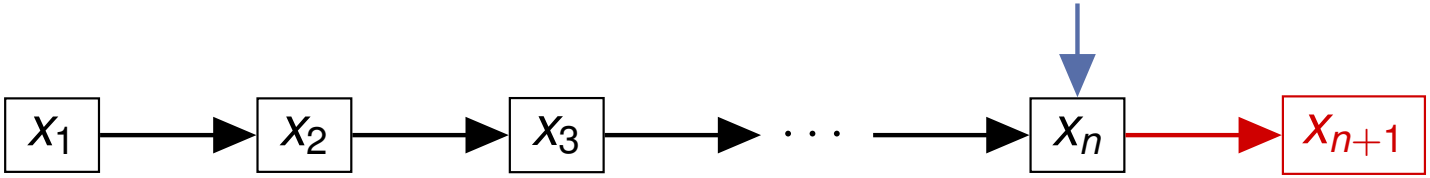


Listes chaînées : insertions

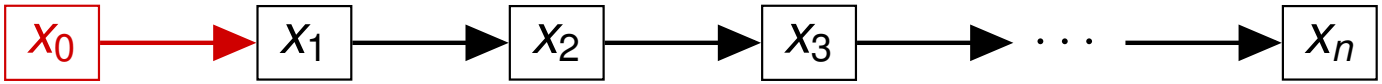
Souvent les flèches sont des pointeurs
$\Longrightarrow$ en mémoire, les boites ne sont pas placées les unes à côté des autres

✖ **Insertion en fin de liste :** O(n)

O(1) si on a accès au dernier elt

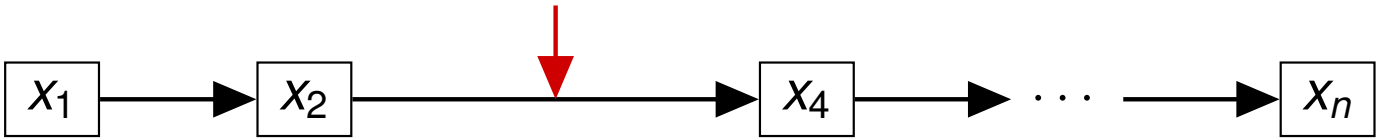


✔ **Insertion en début de liste :** O(1)



✖ **Insertion en $i$ème position :** O(n)

O(1) si on a accès au $(i-1)$ème elt

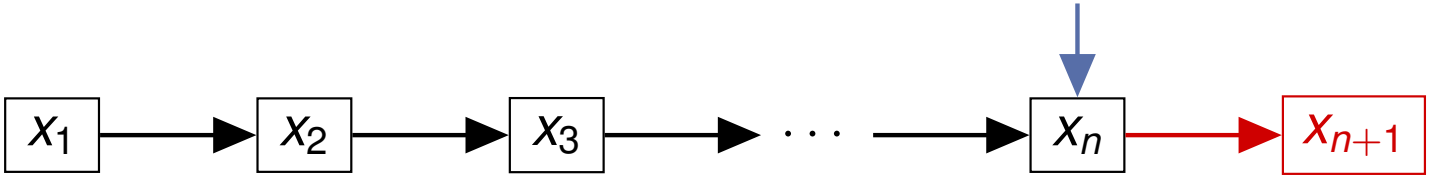


Listes chaînées : insertions

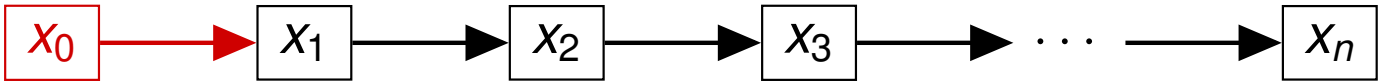
Souvent les flèches sont des pointeurs
$\Longrightarrow$ en mémoire, les boites ne sont pas placées les unes à côté des autres

✖ **Insertion en fin de liste :** O(n)

O(1) si on a accès au dernier elt

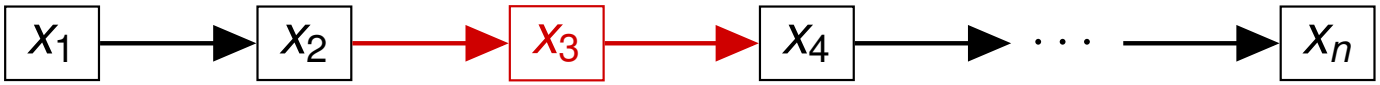


✔ **Insertion en début de liste :** O(1)



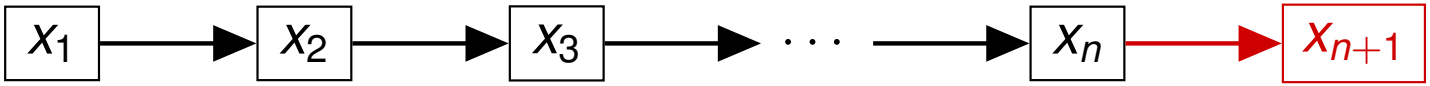
✖ **Insertion en *i*ème position :** O(n)

O(1) si on a accès au $(i-1)$ème elt



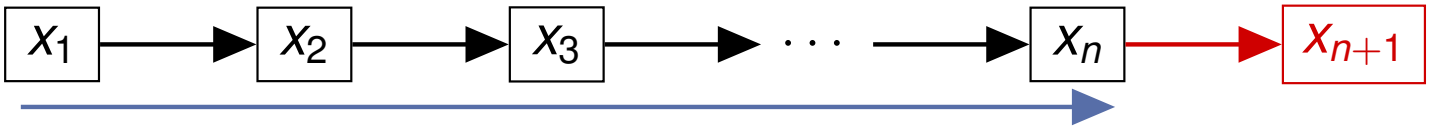
Listes chaînées : suppressions

- **Suppression en fin de liste :** O(n)



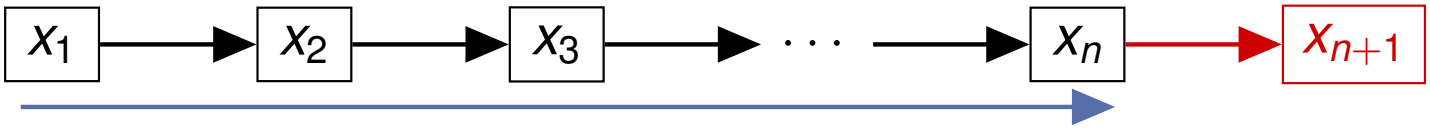
Listes chaînées : suppressions

- **Suppression en fin de liste :** O(n)

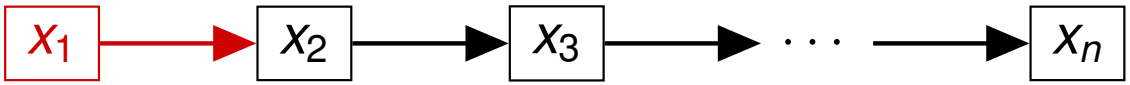


Listes chaînées : suppressions

✖ **Suppression en fin de liste :** O(n)

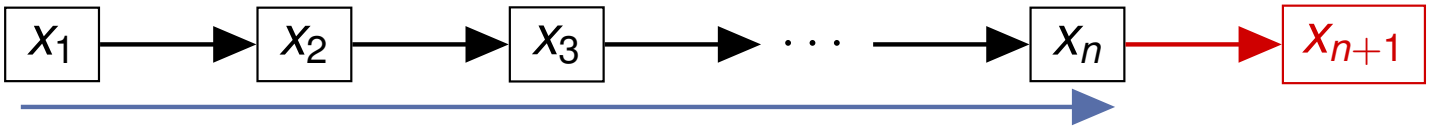


✔ **Suppression en début de liste :** O(1)

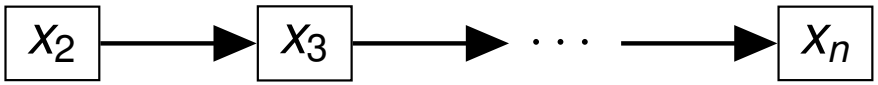


Listes chaînées : suppressions

✖ **Suppression en fin de liste :** O(n)

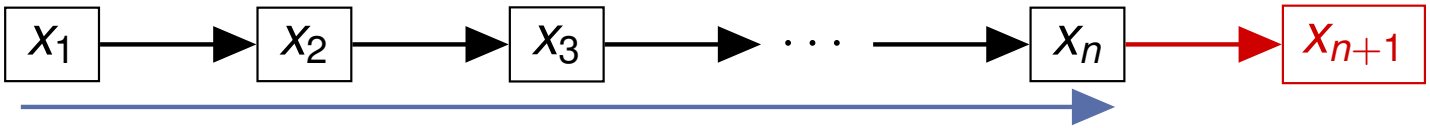


✔ **Suppression en début de liste :** O(1)

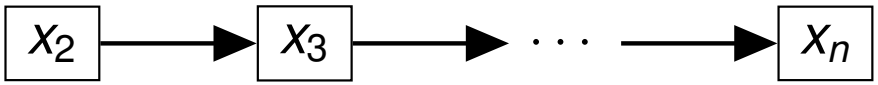


Listes chaînées : suppressions

✖ **Suppression en fin de liste :** O(n)

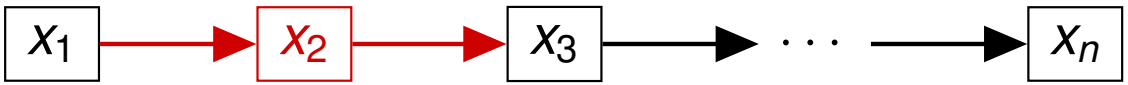


✔ **Suppression en début de liste :** O(1)



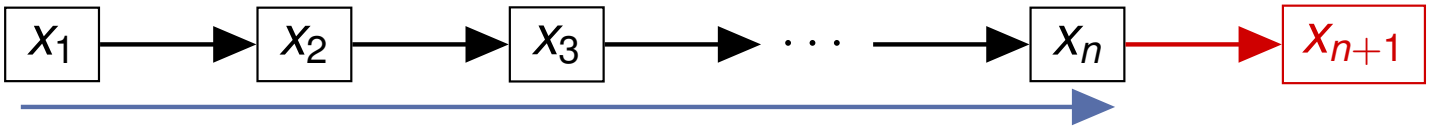
✖ **Suppression du ième élément de la liste :** O(n)

O(1) si on accède à l'elt précédent

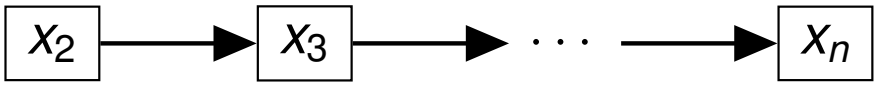


Listes chaînées : suppressions

✖ **Suppression en fin de liste :** O(n)

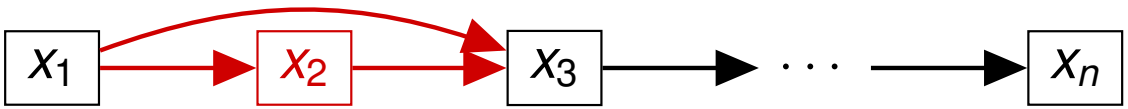


✔ **Suppression en début de liste :** O(1)



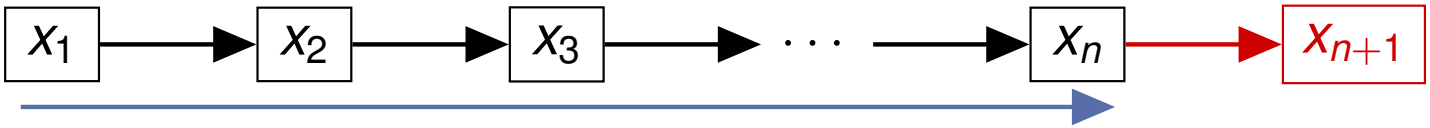
✖ **Suppression du ième élément de la liste :** O(n)

O(1) si on accède à l'elt précédent

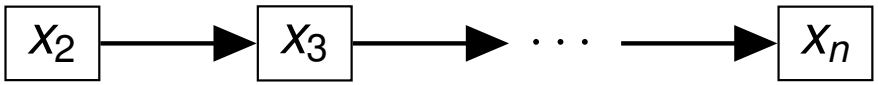


Listes chaînées : suppressions

✖ **Suppression en fin de liste :** O(n)

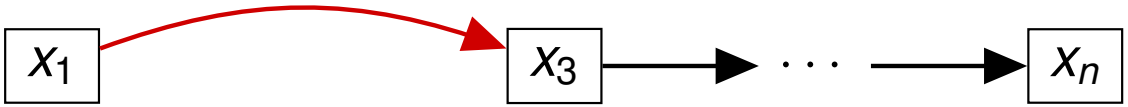


✔ **Suppression en début de liste :** O(1)



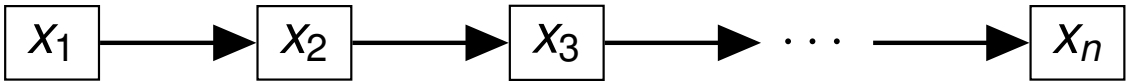
✖ **Suppression du ième élément de la liste :** O(n)

O(1) si on accède à l'elt précédent



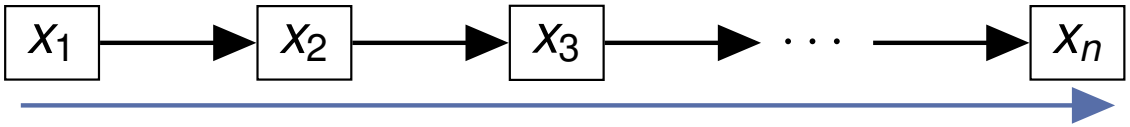
Listes chaînées : recherche/accès

- **Accéder au ième élément :** O(n)



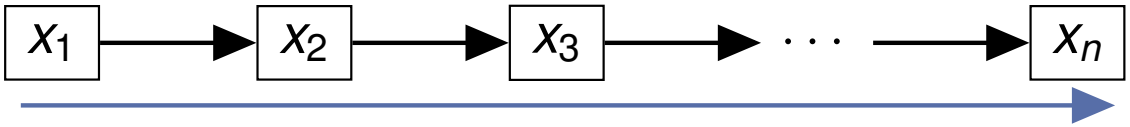
Listes chaînées : recherche/accès

- **Accéder au ième élément :** O(n)

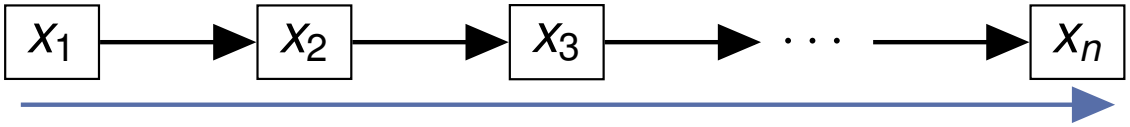


Listes chaînées : recherche/accès

- **Accéder au ième élément :** O(n)

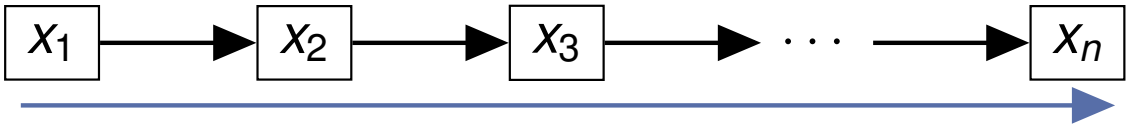


- **Accéder à l'élément le plus petit/plus grand :** O(n)

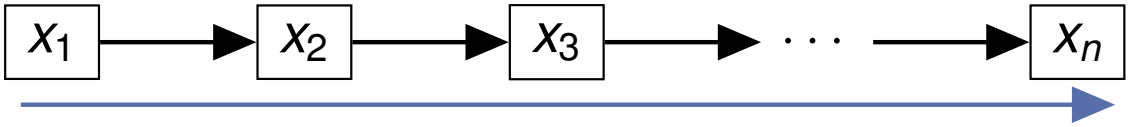


Listes chaînées : recherche/accès

- **Accéder au ième élément :** O(n)

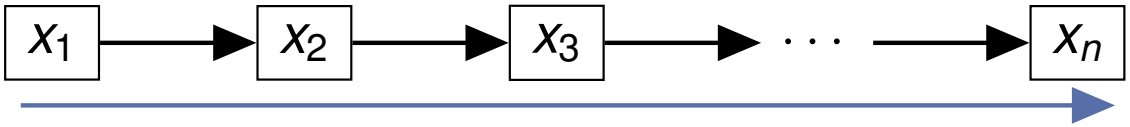


- **Accéder à l'élément le plus petit/plus grand :** O(n)

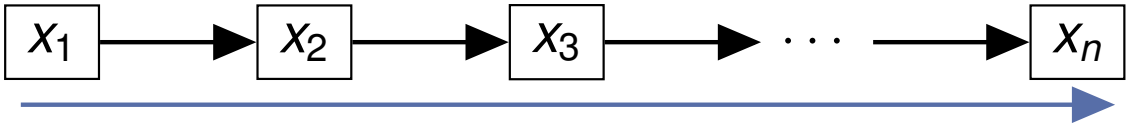


Listes chaînées : recherche/accès

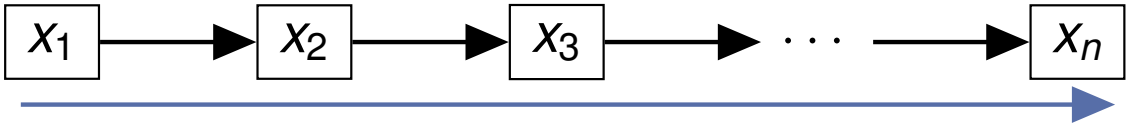
- **Accéder au ième élément :** O(n)



- **Accéder à l'élément le plus petit/plus grand :** O(n)



- **Chercher si un élément existe :** O(n)

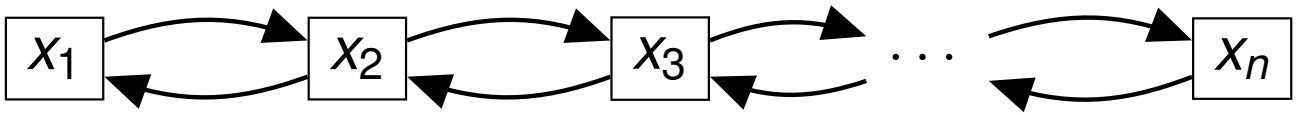


Listes : synthèse

Opération	Complexité	Intérêt
Insertion en début de liste	O(1)	✔
Insertion en fin de liste	O(n)/O(1)	✖
Insertion en cours de liste	O(n)/O(1)	✖
Suppression en début de liste	O(1)	✔
Suppression en fin de liste	O(n)	✖
Suppression en cours de liste	O(n)/O(1)	✖
Accéder au ième élément	O(n)	✖
Accéder au 1er élément	O(1)	✔
Accéder à l'élément le plus petit/le plus grand	O(n)	✖
Chercher si un élément existe	O(n)	✖

Listes particulières

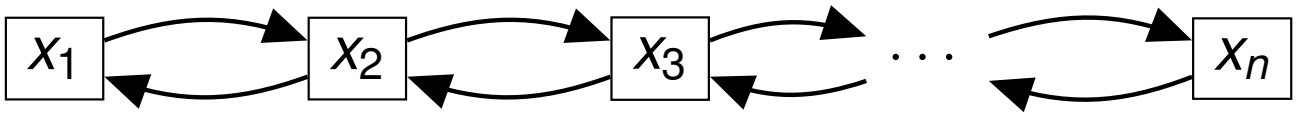
- **Listes doublement chaînées :**



$\Longrightarrow$ accès à l'élément précédent et au suivant

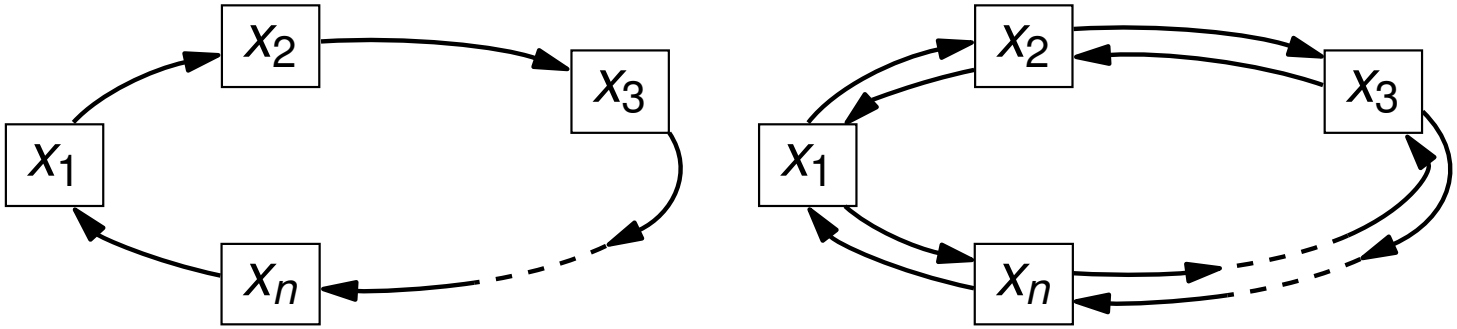
Listes particulières

- **Listes doublement chaînées :**



$\Longrightarrow$ accès à l'élément précédent et au suivant

- **Listes circulaires :**



Comparaisons tableaux – listes

	liste	tableau
nb d'éléments	arbitraire	borné
accès aux éléments	lent (parcours)	rapide
insertion au début	rapide	lente (décalages)
insertion en fin	lente (parcours)	rapide
insertion au milieu	lente (parcours)	lente (décalages)
suppression	rapide (rechaînage)	lente (décalages)

Comparaisons tableaux – listes

	liste	tableau
nb d'éléments	arbitraire	borné
accès aux éléments	lent (parcours)	rapide
insertion au début	rapide	lente (décalages)
insertion en fin	lente (parcours)	rapide
insertion au milieu	lente (parcours)	lente (décalages)
suppression	rapide (rechaînage)	lente (décalages)

Règle : Utiliser plutôt des tableaux sauf si :

- on doit faire beaucoup de suppressions
- on ne connaît pas le nombre d'éléments maximal a priori