Listes chaînées

Christophe Viroulaud Terminale - NSI

Archi 03

Listes chaînées

Christophe Viroulaud

Terminale - NSI

Archi 03

Listes chaînées

vrai en C, un peu différent en Python (haut-niveau)

Quand un tableau est créé, le système lui alloue un espace contigu en mémoire.

h	е	I	Ι	0	!					
	3						9			
								6		
h	е	У	8	5	3	9	1	0	2	!
	3	4								

FIGURE 1 – Le tableau est enregistré dans un espace libre

On accède à chaque élément du tableau **en temps constant**. Cependant insérer un nouvel élément devient problématique.

Listes chaînées

iste chaînée

Comparaison avec un

mplémentation

Manipuler une liste

aille de la liste l-ième élément nsertion

qui pallie les limitations d'un tableau?

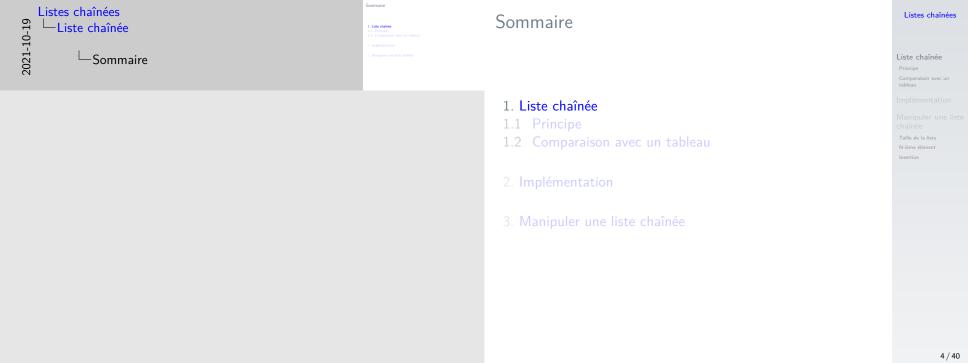
Listes chaînées

rincipe

nparaison avec un eau

.:...lan ....a liata

aînée lle de la liste ème élément





Liste chaînée - principe

Chaque élément :

Liste chaînée - principe

prend une place libre quelconque en mémoire.

Chaque élément :

prend une place libre quelconque en mémoire.

Principe Comparaison avec un tableau

Listes chaînées

Implémentation

puler une liste lée de la liste e élément

Liste chaînée - principe

Chaque élément :

Liste chaînée - principe

prend une place libre quelconque en mémoire. ► connaît l'emplacement de l'élément suivant.

pointe vers

Chaque élément :

- prend une place libre quelconque en mémoire.
- connaît l'emplacement de l'élément suivant.

Listes chaînées

Principe



tête de liste = dernier élément ajouté (ici 2)

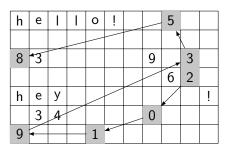


FIGURE 2 – Chaque élément pointe vers le suivant.

Listes chaînées

iste chaînée

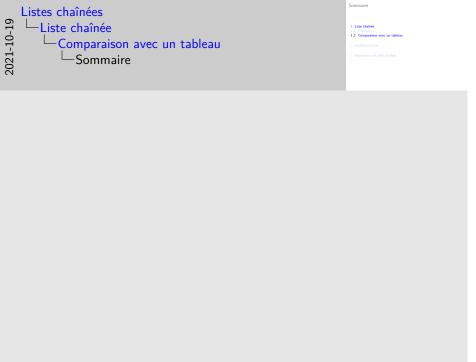
Principe

Comparaison avec un tableau

implementation

Manipuler une liste chaînée

aille de la liste -ième élément



# Sommaire

1. Liste chaînée

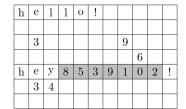
1.2 Comparaison avec un tableau

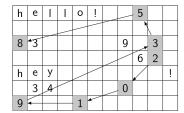
3. Manipuler une liste chaînée

Listes chaînées

Comparaison avec un tableau

## Comparaison avec un tableau





**Activité 1 :** Comparer l'efficacité des deux structures lors de :

- ► l'ajout d'un élément,
- ► l'accès à un élément,
- ► l'insertion d'un élément.

Listes chaînées

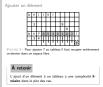
\_iste chaînée

Comparaison avec un tableau

mplémentation

Manipuler une liste

Taille de la liste N-ième élément Insertion Listes chaînées
Liste chaînée
Comparaison avec un tableau
Ajouter un élément



## Ajouter un élément

h	е	1	Ι	0	!					
	3						9			
								6		
h	е	у	8	5	3	9	1	0	2	!
	3	4								
			+	+	+	+	+	+	+	7

FIGURE 3 – Pour ajouter 7 au tableau il faut recopier entièrement ce-dernier dans un espace libre.

# À retenir

L'ajout d'un élément à un tableau a une complexité **linéaire** dans le pire des cas. Listes chaînées

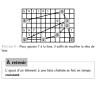
ste chaînée

Comparaison avec un tableau

Implementation

chaînée Taille de la liste N-ième élément Insertion

# Listes chaînées Liste chaînée Comparaison avec un tableau



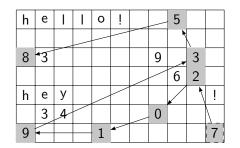


FIGURE 4 – Pour ajouter 7 à la liste, il suffit de modifier la tête de liste.

# À retenir

L'ajout d'un élément à une liste chaînée se fait en temps constant.

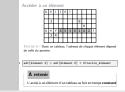
Listes chaînées

iste chaînée

Comparaison avec un tableau

mplémentation

chaînée Taille de la liste N-ième élément



#### Accéder à un élément

h	е		I	0	!					
	3						9			
								6		
h	е	У	8	5	3	9	1	0	2	!
	3	4								

FIGURE 5 – Dans un tableau, l'adresse de chaque élément dépend de celle du premier.

adr[élément 2] = adr[élément 0] + 2\*taille\_élément

À retenir

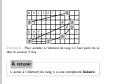
L'accès à un élément d'un tableau se fait en temps **constant**.

Listes chaînées

Comparaison avec un

tableau

# Listes chaînées Liste chaînée Comparaison avec un tableau



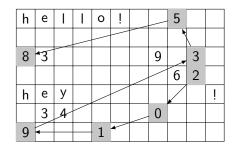


FIGURE 6 – Pour accéder à l'élément de rang n il faut partir de la tête et avancer 5 fois.

# À retenir

L'accès à l'élément de rang n a une complexité **linéaire**.

Listes chaînées

iste chaînée

Comparaison avec un tableau

mplémentation

Vlanipuler une liste chaînée Taille de la liste

# Listes chaînées Liste chaînée Comparaison avec un tableau Insérer un élément



L'insertion d'un élément dans un tableau a une com-

plexité linéaire dans le pire des cas.

## Insérer un élément

h	е	1	Ι	0	!					
	3						9			
								6		
h	е	у	8	5	3	9	1	0	2	!
	3	4								
			+	+	+	7	¥	¥	À	¥

FIGURE 7 – Pour insérer 7 dans le tableau il faut recopier entièrement ce-dernier dans un espace libre.

## À retenir

L'insertion d'un élément dans un tableau a une complexité **linéaire** dans le pire des cas.

Listes chaînées

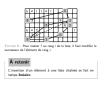
ste chaînée

Comparaison avec un tableau

Implementation

chaînée Taille de la liste

# Listes chaînées Liste chaînée Comparaison avec un tableau



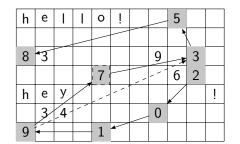


FIGURE 8 – Pour insérer 7 au rang i de la liste, il faut modifier le successeur de l'élément de rang i.

# À retenir

L'insertion d'un élément à une liste chaînée se fait en temps **linéaire**.

Listes chaînées

iste chaînée

Comparaison avec un tableau

Implementation

chaînée Taille de la liste N-ième élément



tableau liste
ajout linéaire constant
accès constant linéaire
insertion linéaire linéaire

Bilan

## Bilan

	tableau	liste		
ajout	linéaire	constant		
accès	constant	linéaire		
insertion	linéaire	linéaire		

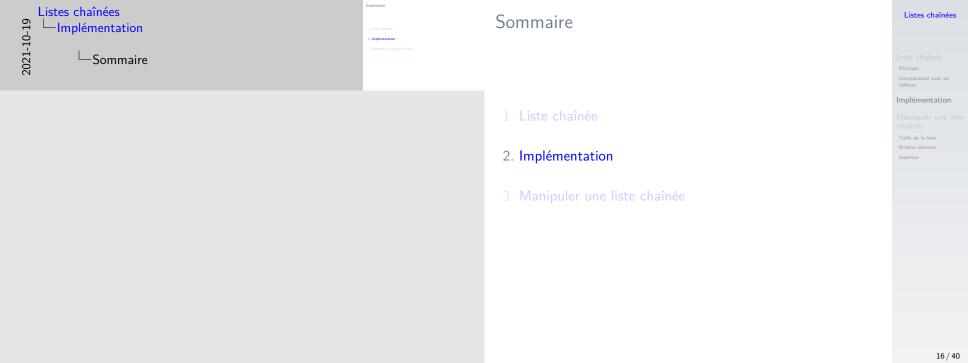
Listes chaînées

Liste chaînée

Comparaison avec un tableau

mplémentation

inée
le de la liste
eme élément



Implémentation - le maillon FIGURE 9 - Chaque maillon contient 2 informations

lazz Maillon: def \_\_init\_\_(melf, val: int, muiv: object): Implémentation - le maillon



FIGURE 9 – Chaque maillon contient 2 informations

```
class Maillon:
    Crée un maillon de la liste chaînée
    11 11 11
    def __init__(self, val: int, suiv: object):
        self.valeur = val
        self.suivant = suiv
```

Code 1 - Un objet Maillon

self.suivant = suiv Code 1 - Un objet Maillon

Listes chaînées

La liste

La liste

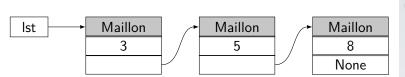


FIGURE 10 – La liste est une succession de maillons

Listes chaînées

Listes chaînées

lst = Maillon(3, Maillon(5, Maillon(8, None)))

Code 2 – Chaque maillon est le suivant d'un autre.

19 / 40

Motion Motion Military Motion 3 S New Process 11 - L'objet Lieue contient une réference à la titu.

La liste - seconde approche

La liste - seconde approche

Liste

lst

Listes chaînées

ste chaînée

Comparaison avec un

Implémentation

Manipuler une liste

Taille de la liste N-ième élément



FIGURE 11 – L'objet Liste contient une référence à la tête.

- 1. L'attribut tete représente le premier Maillon.
- 2. Une liste vide renvoie alors None.

```
class Liste:
    """

Crée une liste chaînée

"""

def __init__(self):
    self.tete: Maillon = None
```

 ${\sf Code}\ {\sf 3-Objet}\ {\bf Liste}$ 

Listes chaînées

Principe

Implémentation

Vaninuler une list

ille de la liste ième élément

#### Activité 2 :

- 1. Écrire la méthode  $est\_vide(self) \rightarrow bool$  qui renvoie True si la liste est vide, False sinon.
- 2. Écrire la méthode ajoute(self, val: int)  $\rightarrow$ None qui ajoute un Maillon en tête de la liste.
- 3. Créer la liste contenant les éléments 8, 5, 3.

Listes chaînées

## Listes chaînées ☐ Implémentation

```
def ext_vide(zelf) -> bool:
    return melf.tete is None
 def ajoute(self, val: int) => None:
    self.tete = Maillon(val, self.tete)
```

```
lst.ajoute(5)
lst.ajoute(5)
lst.ajoute(3)
                      Code 4 - Création de la liste
```

3

4

```
def est_vide(self) -> bool:
    return self.tete is None
def ajoute(self, val: int) -> None:
    self.tete = Maillon(val, self.tete)
```

```
lst = Liste()
lst.ajoute(8)
lst.ajoute(5)
lst.ajoute(3)
```

Code 4 – Création de la liste

Listes chaînées

La méthode \_\_str\_\_ permettra de visualiser l'état de la liste.

```
def __str__(self):
    m = self.tete
    s = ""

while m is not None:
    s += str(m.valeur)+" - "
    m = m.suivant

else:
    s += "fin"

return s
```

```
1 print(lst)
```

Code 5 – Afficher la liste

Listes chaînées



# Listes chaînées Manipuler une liste chaînée Taille de la liste Manipuler une liste chaînée - taille

```
Manipuler une liste chaînée - taille
Peur consière la zallé de la litte if fant la percourir
entièment.

Activité 3:

- Entre la entième de la commanda del commanda de la commanda de la commanda del commanda de la commanda de la commanda de la commanda del command
```

Pour les plus avancés : Écrire la méthode native (itérative) len (self) → int qui redéfinit la

## Manipuler une liste chaînée - taille

Pour connaître la taille de la liste il faut la parcourir entièrement.

#### Activité 3:

- Écrire la méthode récursive taille\_rec(self, m: Maillon) → int qui renvoie la taille de la liste démarrant à m
- Écrire la méthode taille(self) → int qui renvoie la taille de la liste. Cette méthode utilisera taille\_rec.
- 3. Pour les plus avancés : Écrire la méthode native (itérative) \_\_len\_\_(self) → int qui redéfinit la fonction len pour la classe Liste.

#### Listes chaînées

\_iste chaînée

Comparaison avec un tableau

nplémentation

lanipuler une liste naînée

Taille de la liste N-ième élément

```
def taille_rec(self, m: Maillon) -> int:
    """
    méthode interne pour calculer la taille
    de la chaîne
    """
    if m is None:
        return 0
    else:
        return 1 + self.taille_rec(m.suivant)
```

ste chaînée

mparaison avec un oleau

anipuler une liste

Taille de la liste N-ième élément

```
Listes chaînées
 -Manipuler une liste chaînée
    ☐ Taille de la liste
```

```
def taille(self) -> int:
    return self.taille rec(self.tete)
      Code 6 - Affichage de la taille de la liste
```

```
def taille(self) -> int:
    appel principal de la méthode récursive
   pour mesurer
    la taille de la chaîne
    11 11 11
    return self.taille_rec(self.tete)
```

```
print(lst.taille())
```

Code 6 – Affichage de la taille de la liste

Listes chaînées

Taille de la liste

```
Listes chaînées

Manipuler une liste chaînée
Taille de la liste
```

```
1 def __len__(self) -> int:
2    maillon = self.tete
3    taille = 0
4    while maillon is not None:
5         maillon = maillon.suivant
```

```
6 taille += 1
7 return taille
```

print(len(lst))

```
Code 7 - Appel de la fonction len
```

Listes chaînées

Taille de la liste



# Sommaire

- 3. Manipuler une liste chaînée
- 3.2 N-ième élément

Listes chaînées

N-ième élément

Listes chaînées -Manipuler une liste chaînée └─N-ième élément –N-ième élément

N-ième élément Il faut parcourir la liste iusqu'au rang n pour trouver Activité 4 : L. Écrire la méthode récursive get\_element\_rec(self, n: int, n: Maillon) -> int qui renvoie la valeur du n-ième élément de la liste démarrant à m Écrire la méthode get element(self. n: int) int qui renvoie la valeur du n-ième élément Cette méthode utilisera met element rec. Pour les plus avancés : Écrire la méthode native (itérative) getitem (self. n: int) → int qui redéfinit la structure à crochets (lat [n]) pour

la classe Liste.

#### N-ième élément

Il faut parcourir la liste jusqu'au rang n pour trouver l'élément.

#### Activité 4:

- Écrire la méthode récursive get\_element\_rec(self, n: int, m: Maillon)  $\rightarrow$  int qui renvoie la valeur du n-iéme élément de la liste démarrant à m
- 2. Écrire la méthode get\_element(self, n: int) → int qui renvoie la valeur du n-iéme élément. Cette méthode utilisera get\_element\_rec.
- 3. Pour les plus avancés : Écrire la méthode native (itérative) getitem (self, n: int)  $\rightarrow$  int qui redéfinit la structure à crochets (lst[n]) pour la classe Liste.

Listes chaînées

N-ième élément

```
def get_element_rec(self, n: int, m: Maillon) -> int:
    """
    méthode interne pour renvoyer le n-ième élément.
    """
    if n == 0:
        return m.valeur
    else:
```

return self.get\_element\_rec(n-1, m.suivant)

Listes chaînées

ste chaînée

omparaison avec un Ibleau

plémentation

chaînée Taille de la liste N-ième élément

```
def get_element_rec(self, n: int, m: Maillon) -> int:
    méthode interne pour renvoyer le n-ième élément.
    # n est plus grand que la taille de la liste
    if m is None:
        raise IndexError("indice invalide")
    if n == 0:
        return m.valeur
    else:
        return self.get_element_rec(n-1, m.suivant)
```

Code 8 – Avec gestion du dépassement de taille

Listes chaînées

N-ième élément

```
def get_element(self, n: int) -> int:
    """

appel principal de la méthode récursive pour renvoyer le n-ième élément
    """

return self.get_element_rec(n, self.tete)
```

```
print(lst.get_element(3))
```

Code 9 – Appel de la fonction

Listes chaînées

Principe

Comparaison avec un

plémentation

ivianipuier une liste chaînée Taille de la liste N-ième élément

```
| find __mexism__(mifs, no line) == into
| command | Vidence for eng no. Les indice
| command | vidence for eng no. Les indice
| command | vidence for eng no. Les indice
| command | vidence for eng no les indice
| command | vidence for eng no les indice
| command | vidence for eng no les indice
| command | vidence for eng no les indice
| command | comman
```

```
def __getitem__(self, n: int) -> int:
       renvoie l'élément de rang n. Les indices
      commencent à 0.
        11 11 11
       maillon = self.tete
       i = 0
        while i < n and maillon is not None:
           maillon = maillon.suivant
            i += 1
10
       if maillon is None:
11
           raise IndexError("indice invalide")
12
13
       return maillon.valeur
14
```

Code 10 – Appel de la fonction

print(lst[3])

Listes chaînées

N-ième élément



# Sommaire

- 1. Liste chaînée
- 2. Implémentation
- 3. Manipuler une liste chaînée 3.1 Taille de la liste
  - élément
- 3.3 Insertion

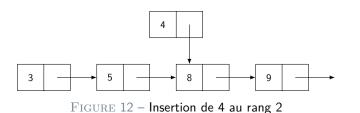
Listes chaînées

Insertion

Intention Co. Description of the delicate service of the delicate service of the delicate service of the delicate service of the delicate of t

#### Insertion

L'insertion d'un élément au rang n peut être réalisée sur le même principe. On prendra le parti d'insérer l'élément en fin si la valeur de n dépasse la taille de la liste.



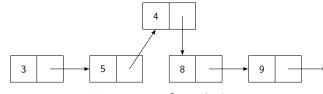


FIGURE 13 – Seconde étape

Listes chaînées

iste chaînée

Principe Comparaison avec

nnlémentation

anipuler une liste

Taille de la liste N-ième élément Insertion Listes chaînées

Manipuler une liste chaînée
Insertion

Activité 5 :

1. Écrie la fonction récuritive lansarer\_rec(self, val: ist, s: ist, s: ist, s: chject) — Hose qui inden' l'idément au rang n.

Ecric la fonction lansarer (self, val: ist, s: i

Il faut remarquer que la fonction inserer\_rec place en réalité l'élément au rang m+1.

#### Activité 5 :

- Écrire la fonction récursive inserer\_rec(self, val: int, n: int, m: object) → None qui insère l'élément au rang n.
- Écrire la fonction inserer(self, val: int, n: int) → None qui insère l'élément val au rang n.
   Cette fonction gérera le cas ou n = 0.

### Remarque

Il faut remarquer que la fonction inserer\_rec place en réalité l'élément au rang n+1.

Listes chaînées

Liste chaînée

Comparaison avec u

plémentation

chaînée Taille de la liste N-ième élément

Insertion

```
of inserer_rectail, not inst, no inst, no depth of Bear in the property of the control of the co
```

```
1 def inserer_rec(self, val: int, n: int, m:
      object) -> None:
       méthode interne pour placer val au rang n
       si n est trop grand, place l'élément en
      fin de liste
       11 11 11
       if m.suivant is None or n == 0:
           nouveau = Maillon(val, m.suivant)
           m.suivant = nouveau
       else:
           self.inserer_rec(val, n-1, m.suivant)
10
```

Listes chaînées

Liste chaînée

Comparaison avec u

plémentation

nipuler une liste aînée

Taille de la liste N-ième élément Insertion

# Listes chaînées Manipuler une liste chaînée Insertion

```
of innercial, vd: us, u us) > hum-
property of the formation pure
place vd: us

# opportunity of a 'innertian pure
place vd: us

# opportunit on an particular of i'
intertian act on data

# opportunit on a particular of i'
intertian act on data

# opportunity of intertian act of intertian act on data

# opportunity of intertian act of intertion act of intertian act of intertion act of intertian act of intertian
```

```
def inserer(self, val: int, n: int) -> None:
       appel principal de l'insertion pour
      placer val en n
       # gestion du cas particulier où l'
      insertion est en début
       if n == 0:
           nouveau = Maillon(val, self.tete)
           self.tete = nouveau
       else:
           # n-1 pour ajuster la position
10
           self.inserer_rec(val, n-1, self.tete)
11
```

ste chaînée

Comparaison avec un

nplémentation

chaînée
Taille de la liste
N-ième élément
Insertion