

# Fondements informatiques I

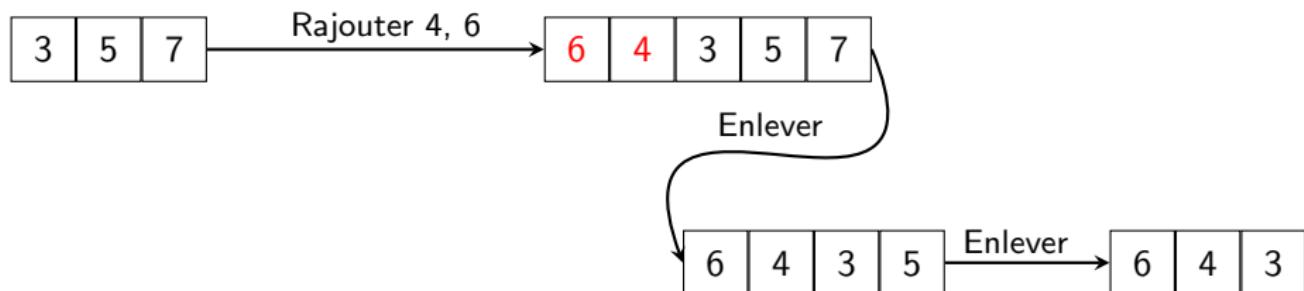
## Cours 9: Algorithmes : piles et files

Sorina Ionica [sorina.ionica@uvsq.fr](mailto:sorina.ionica@uvsq.fr)

Sandrine Vial [sandrine.vial@uvsq.fr](mailto:sandrine.vial@uvsq.fr)

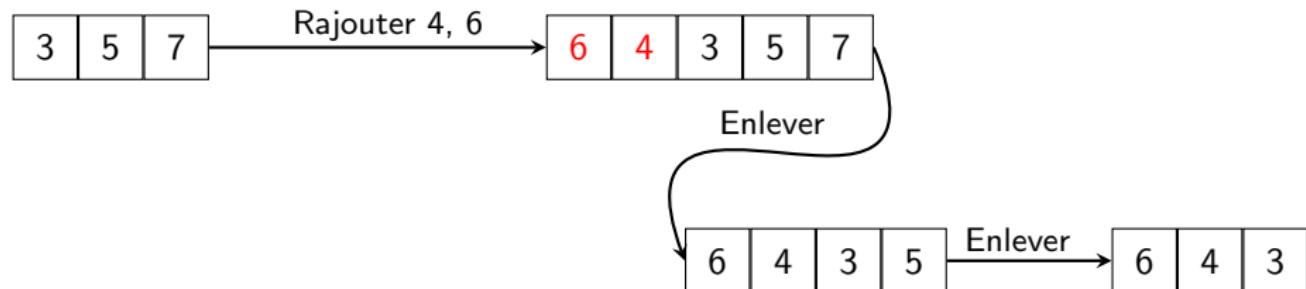
# File

Une file (en anglais queue) est une structure de données basée sur le principe que les premiers éléments ajoutés dans la file seront les premiers à en être retirés.



# File

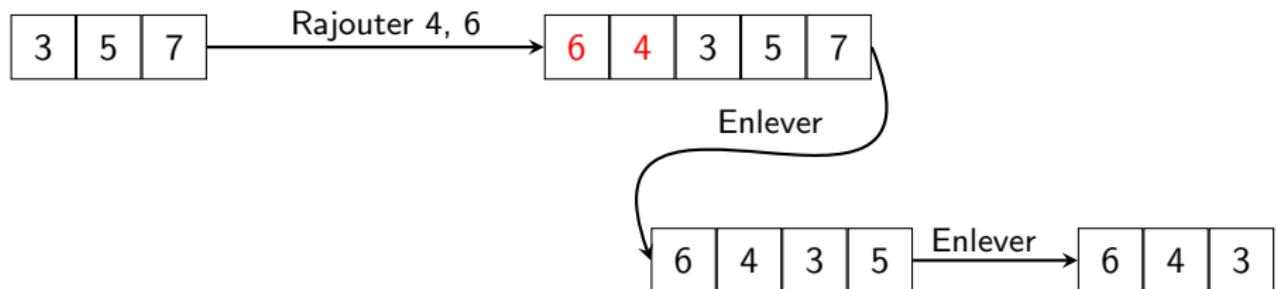
Une file (en anglais queue) est une structure de données basée sur le principe que les premiers éléments ajoutés dans la file seront les premiers à en être retirés.



Nombreuses applications :

- traiter dans l'ordre une liste de transactions, la gestion d'un stock
- gestion des tâches dans l'ordre d'arrivée par exemple sur un système d'exploitation
- transmission des données dans l'ordre sur un réseau

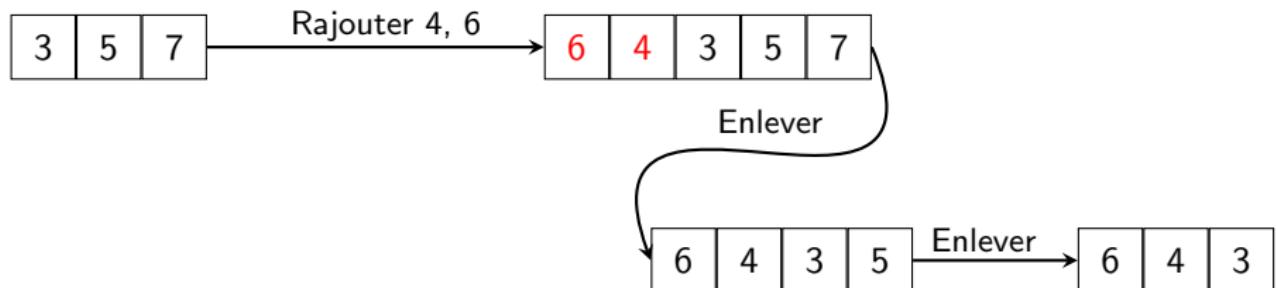
# File



Pour mettre en place une file nous avons besoin de deux procédures (fonctions en Python) :

- une pour ajouter en tête de file
- une pour enlever de la file.

# File



Pour mettre en place une file nous avons besoin de deux procédures (fonctions en Python) :

- une pour ajouter en tête de file
- une pour enlever de la file.

```
file=[3,5,7]
ajouter(file,4)
ajouter(file,6)
enlever(file)
enlever(file)
```

# Ajouter dans le file

On suppose que notre pile est représentée par un tableau dynamique.

## Procédure ajouter

Input: La file f et l'élément e à ajouter

Output : La file modifiée

$l \leftarrow \text{longueur}(f)$

On agrandit f d'une case (c.a.d. on alloue de la mémoire,

$\text{longueur}(f) \leftarrow \text{longueur}(f) + 1$

Pour tout i allant de  $l-1$  à 0

$f[i+1] \leftarrow f[i]$

$f[0] = e$

Return f

# Enlever de la file

On suppose que notre pile est représentée par un tableau dynamique.

## Procédure enlever

Input: La file f

Output : Le dernier élément de la file et la file modifiée

`l<-longueur(f)`

Si (`l>0`) alors:

`e<-f[l-1]`

On rétrécit f d'une case (désallouer mémoire,  
`longueur(f)<-l-1`)

Return e,f

# Une file en Python

En Python, on utilisera les listes pour représenter des tableaux dynamiques.

```
def ajouter(f,e):
    l=len(f)
    f.append(0)
    for i in range(l):
        f[l-i]=f[l-i-1]
    f[0]=e
```

# Une file en Python

En Python, on utilisera les listes pour représenter des tableaux dynamiques.

```
def ajouter(f,e):
    l=len(f)
    f.append(0)
    for i in range(l):
        f[l-i]=f[l-i-1]
    f[0]=e

def enlever(f):
    if (len(f)>0):
        return f.pop()
```

# Une file en Python

En Python, on utilisera les listes pour représenter des tableaux dynamiques.

```
def ajouter(f,e):
    l=len(f)
    f.append(0)
    for i in range(l):
        f[l-i]=f[l-i-1]
    f[0]=e

def enlever(f):
    if (len(f)>0):
        return f.pop()
```

La fonction `pop(i)` enlève l'élément d'indice `i` dans la liste.

L'appel `pop()` enlève le dernier élément de la liste.

Serait-il possible d'utiliser la fonction `remove` au lieu de `pop` ?

# Une file en Python

```
def ajouter(f,e):                def enlever(f):  
    l=len(f)                      if (len(f)>0):  
        f.append(0)                  return f.pop()  
    for i in range(l):  
        f[l-i]=f[l-i-1]  
    f[0]=e
```

Pour quoi ces fonctions ne renvoient pas `f` comme demandé ?

# Une file en Python

```
def ajouter(f,e):                def enlever(f):  
    l=len(f)                      if (len(f)>0):  
        f.append(0)                  return f.pop()  
        for i in range(l):  
            f[l-i]=f[l-i-1]  
        f[0]=e
```

Pour quoi ces fonctions ne renvoient pas `f` comme demandé ?

```
f=[3,5,7]  
ajouter(f,4)  
ajouter(f,6)  
print("voici la file",f)  
enlever(f)  
print("voici la file",f)
```

## Exemple d'utilisation

Écrire un programme qui permet de traiter les transactions sur un compte bancaire pendant une journée. A la fin de la journée, le montant restant sur le compte est affiché.

En Python, on utilisera la fonction `enlever`.

```
montant=3498
f=[53,-78,198,-23,-243]

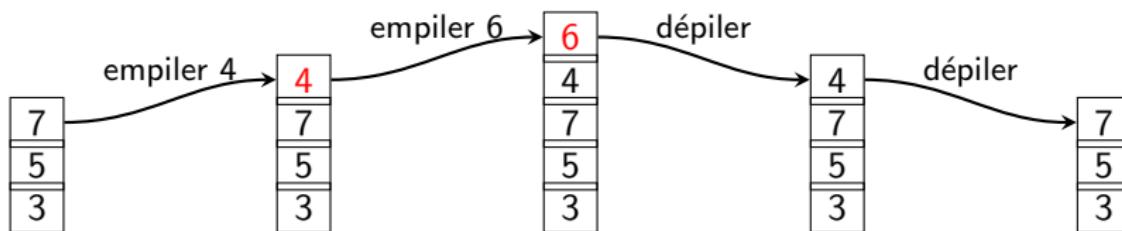
while len(f)>0:
    transaction=enlever(f)
    montant=montant+transaction

print("montant restant", montant)
```

# Pile

Une pile (en anglais stack) est une structure de données fondée sur le principe "dernier arrivé, premier sorti"

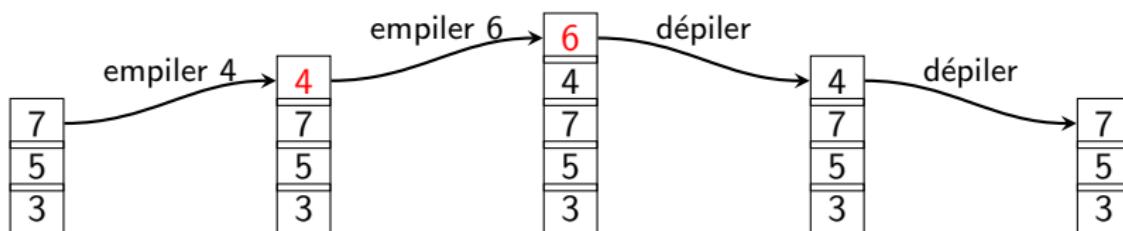
Le dernier élément ajouté à la pile est le premier à en sortir.



# Pile

Une pile (en anglais stack) est une structure de données fondée sur le principe "dernier arrivé, premier sorti"

Le dernier élément ajouté à la pile est le premier à en sortir.



Nombreuses applications :

- inverser une séquence
- évaluer des expressions arithmétiques (à voir plus tard)
- dans les navigateurs internet pour sauvegarder les pages visitées

# Représenter une pile, empiler

Par un tableau dynamique : sommet de pile = dernière case

- empiler = ajouter en dernière position
- dépiler = supprimer l'élément en dernière position

## Algorithme pour empiler

Input: Pile p et élément e

Output: Pile p modifiée

On agrandit p d'une case (allocation mémoire etc.)

On rajoute e.

Renvoyer p

# Dépiler

## Algorithme pour dépiler

Input: Pile p

Output: Pile p modifiée et la valeur dépilerée

l<-longueur(f)

Si (l>0) alors:

    e<-f[l-1]

On rétrécit p d'une case (désallouer mémoire etc.)

Renvoyer p,e

# Une pile en Python

On utilisera une liste pour représenter la pile.

```
def empiler(p,e):          def depiler(p):  
    p.append(e)            if len(p)>0:  
                        return p.pop()
```

```
p=[3,5,7]  
empiler(p,4)  
empiler(p,6)  
print("voici l'état de la pile", p) #p=[3,5,7,4,6]  
depiler(p)  
depiler(p)  
print("voici l'état de la pile", p) #p=[3,5,7]
```

# Exemple de programme utilisant une pile

Inverser un tableau ou une chaîne de caractères en utilisant une pile.  
En Python, on utilise les fonctions empiler et depiler.

## En Python

```
a = [3, 5, 7, 2]
inverse = []

while (len(a)>0):
    empiler(inverse, depiler(a))

print(inverse)
```