Les structures de données avancées
Piles & Files

Anis SAIED
anis_saied@hotmail.com

IPEIN

2025/26

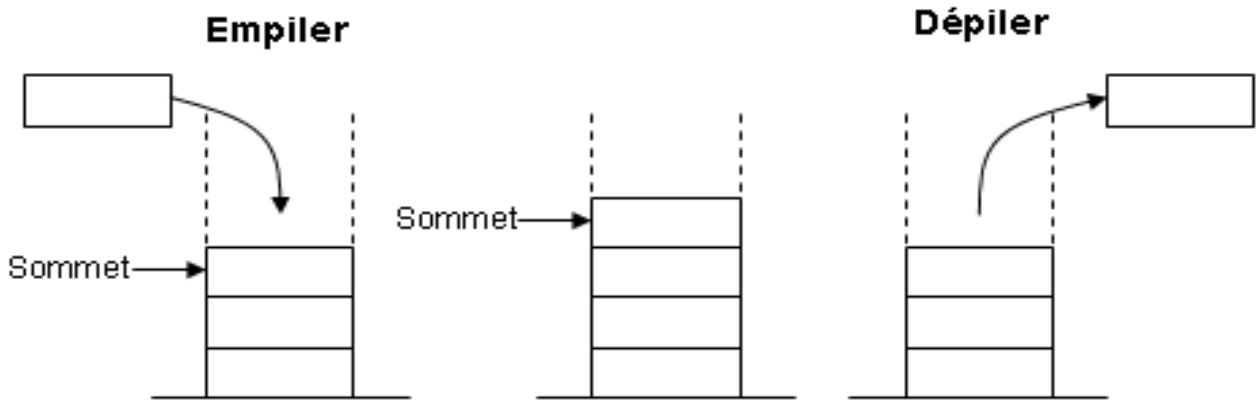
Les Piles

Définition

Une Pile (**stack**) est une structure de données qui met en oeuvre le principe : **dernier entré, premier sorti** ou **LIFO** (**L**ast - **I**n - **F**irst - **O**ut).

Cette structure permet de "stocker" provisoirement des éléments et de récupérer les données les plus récentes.

Les seules opérations dont on a besoin sont EMPILER (INSERER ou **push**en anglais) et DEPILER (SUPPRIMER ou **pop** en anglais).



Le sommet de la pile est le dernier élément empilé. C'est celui qui sera le premier dépilé.

Les Piles

Exemples de cas d'utilisations

- Gestion de l'historique dans un navigateur web:
  Les deux boutons « page suivante » et « page précédente» de votre navigateur, chacun utilise une pile.
  Les différentes pages visitées sont stockées dans une pile. Lorsqu'on clique sur le bouton de retour en arrière, on dépile la pile associée au bouton précédent.
- Exécution d'une fonction récursive. Chaque appel récursif est stocké dans la pile d'exécution jusqu'à atteindre un cas terminal, puis les fonctions sont dépilées.

Les Piles I

Exemples: Pile d'exécution d'une fonction récursive

Une pile d'exécution est une structure de données qui stocke les appels de fonctions en cours d'exécution.

Chaque appel de fonction crée un nouveau cadre (frame) dans la pile, contenant les **variables locales** et l'**adresse de retour**.

Les cadres sont empilés lors des appels et dépilés lorsque les fonctions se terminent.

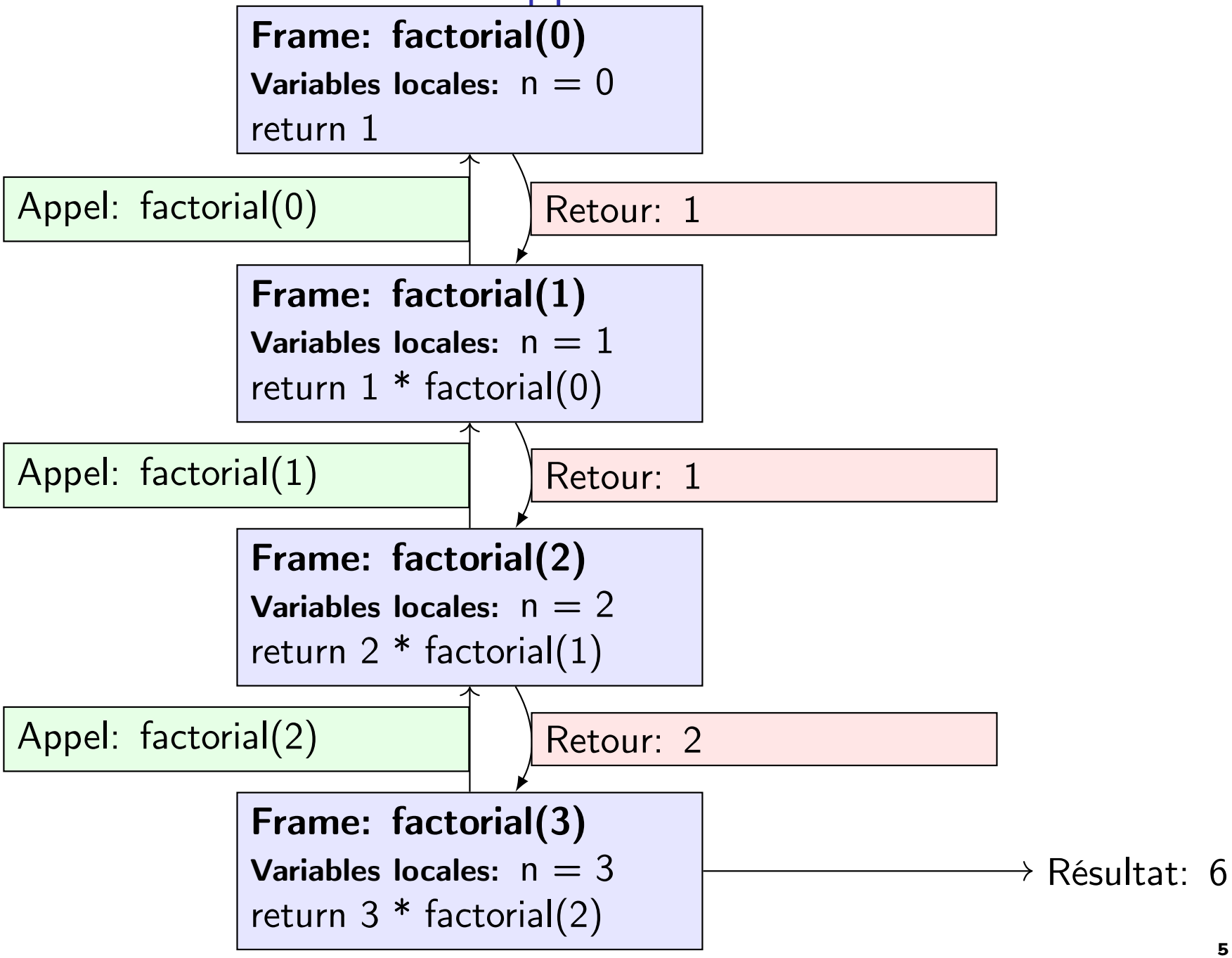
Le premier appel est empilé au bas de la pile, et chaque appel successif est ajouté au-dessus.

La pile d'exécution fonctionne de bas en haut, où le fond de la pile représente le premier appel.

Une fois le cas terminal est atteint, les fonctions sont dépilées en sens inverse, avec les valeurs de retour propagées du haut vers le bas.

Exemple : voir schéma d'exécution de `factorial(3)`

Les Piles: Illustration des appels récursifs



Les Piles

Mise en œuvre d'une pile avec une liste

Il y a beaucoup de façons de concevoir une pile : listes, tableaux,.... La plus simple consiste à utiliser un conteneur de type **list**.

Programmer une classe nommée `Pile` contenant les méthodes suivantes :

- `__init__(self)` : initialise une pile vide non bornée (dont la taille maximale n'est pas définie).
- `pile_vide(self)` : retourne `True` si la pile p est vide et `False` sinon.
- `sommet(self)` : renvoie l'élément du sommet de la pile p, sans le supprimer.
- `taille(self)` : renvoie la taille de la pile p.
- `empiler(self,x)` : ajoute l'élément x au sommet de la pile p.
- `depiler(self)` : supprime le sommet de la pile p et le retourne. Que se passe-t-il quand on essaye de dépiler une pile vide ?

Les Piles

Exercices d'application

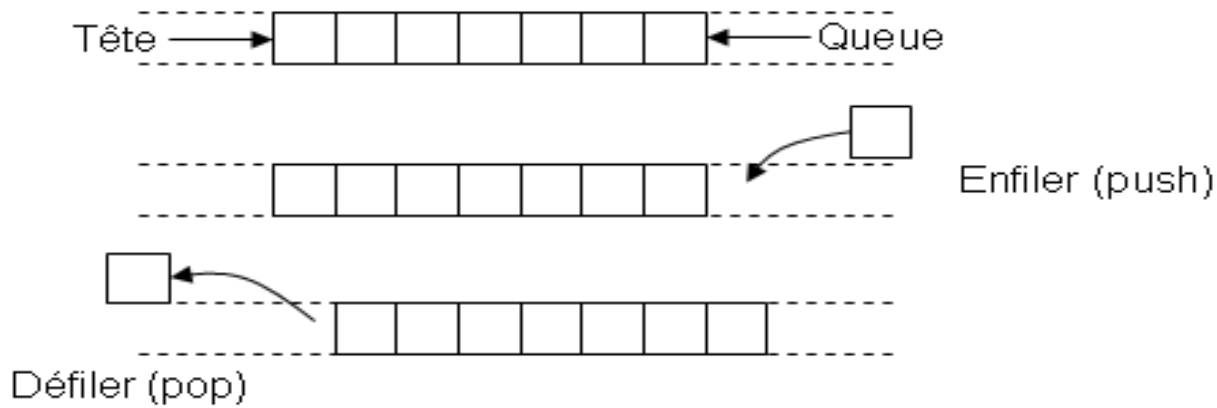
Exercice 1: conversion de base 10 en base 2
Écrire une fonction `conversion(n)` qui retourne la représentation en base 2 du nombre n représenté en base 10 à l'aide de piles.
Indication : La fonction python `bin(10)` retourne `'0b1010'`.

Exercice 2: Permutations circulaires
Écrire une fonction `permut_circ(p,n)` qui reçoit en argument une pile `p` et un entier `n` et effectue sur la pile `n` permutations circulaires successives.
Dans cet exercice, c'est la pile `p` elle-même qui sera modifiée.
Par exemple avec $n = 2$:
`p = [103, 2, 98, 11, 7]` $\longrightarrow$ `p = [11, 7, 103, 2, 98]`

Les Files

Définition

Une File (**queue**) est une structure de données qui met en œuvre le principe : **premier entré, premier sorti** ou **FIFO** (**F**irst - **I**n - **F**irst - **O**ut). La première valeur de la file est la tête ou sommet de la file et la dernière est la queue de la file.
Les opérations principales sont ENFILER (**enqueue/push** en anglais) et DEFILER (**dequeue/pop** en anglais).



Lorsque nous enfilons un élément, il est ajouté à la fin de la file.
Quand nous défilons un élément, le premier élément est supprimé de la file.
Ce processus continue, le premier élément ajouté étant toujours le premier à être retiré.

Les Files

Exemples de cas d'utilisation

En Informatique, les files sont utilisées pour :

- Gestion de la file d'attente d'impressions : Les documents sont enfilés dans une file et imprimés dans l'ordre où ils ont été ajoutés.
- Gestion des processus dans un système d'exploitation: Les tâches sont enfilées en attente de traitement par le processeur, chacune traitée dans l'ordre d'arrivée.
- Gestion des appels téléphoniques dans un centre d'appel: Les appels sont enfilés et traités dans l'ordre où ils ont été reçus.

Les Files

Mise en œuvre d'une file avec une liste et une taille maximale globale

Le type `list` intègre déjà toutes les méthodes pour réaliser cette structure de données.

Programmer une classe nommée `File` contenant les méthodes suivantes :

- TAILLE_MAX = 40 : attribut de classe définissant la taille maximale par défaut pour la file.
- __init__(self) : initialise une file vide bornée, en utilisant la taille maximale définie par l'attribut `TAILLE_MAX`.
- file_vide(self) : retourne `True` si la file f est vide et `False` sinon.
- enfiler(self, x) : ajoute l'élément x à la fin de la file f si celle-ci n'est pas pleine.
- defiler(self) : retire et retourne l'élément en début de file f.
- sommet(self) : renvoie l'élément au début de la file f sans le retirer.
- taille(self) : renvoie la taille de la file f.
- est_pleine(self) : retourne `True` si la file a atteint sa capacité maximale définie par `TAILLE_MAX`.

Exercices d'application

Exercice 3: Gestion d'une file de priorité
Écrire une fonction `gestion_priorite(f, obj)` qui insère un objet `obj` dans une file `f` en fonction de son attribut `priorite` (un nombre entier). Les objets ayant une priorité élevée doivent être en début de file pour être traités en premier. La file doit respecter la limite de taille définie par `TAILLE_MAX`.
Indication : Assurez-vous que la file n'est pas pleine avant d'ajouter l'objet, et placez chaque nouvel objet de manière à ce que les priorités les plus hautes soient toujours en début de file.

Exercice 4: Gestion d'une file circulaire
Écrire une fonction `rotation_file(f, n)` qui effectue une rotation circulaire de la file `f` par n positions. La rotation circulaire déplace les éléments en début de file vers la fin, et ainsi de suite. Ce processus doit modifier la file elle-même.

Solution - Exercice 3 : Gestion d'une file de priorité

```python
from file import *

def gestion_priorite(f, obj):
    """Insère un élément dans la file en fonction de sa priorité."""
    if est_plein(f):
        print("La file est pleine!")
        return
    # Insertion en fonction de la priorité
    index = 0
    while index < taille(f) and f[index].priorite >= obj.priorite:
        index += 1
    f.insert(index, obj)

# Exemple d'utilisation
class Tache:
    def __init__(self, nom, priorite):
        self.nom = nom
        self.priorite = priorite

tacheA = Tache("Tâche A", 3); tacheB = Tache("Tâche B", 1)
tacheC = Tache("Tâche C", 5)
file = creer_file()
gestion_priorite(file,tacheA); gestion_priorite(file, tacheB)
gestion_priorite(file, tacheC)
```

Solution - Exercice 4 : Gestion d'une file circulaire

```python
from file import *

def rotation_file(f, n):
    """Effectue une rotation circulaire de n positions sur la file."""
    if file_vide(f):
        print("La file est vide!")
        return

    n = n % len(f) # Gérer les rotations supérieures à la taille de la file
    for i in range(n):
        enfiler(f, defiler(f))

# Exemple d'utilisation
f = creer_file()
for i in range(5): # Remplir la file avec des objets
    f.enfiler(Tache("A", i))
rotation_file(file, 2)
for i in range(taille(f)): # Affiche la file après rotation
    elt = defiler(f); print(elt); enfiler(f, elt)
```

Questions ?