

Institut Préparatoire aux Études d'Ingénieurs de Tunis



Leila Ben Othman

Deuxième année de classes préparatoires

SM - SP

Année universitaire: 2017 - 2018

Les piles et les files

Plan

① Introduction

② Les piles

- Définition

- Opérations de base

- Réalisation

③ Les files

- Définition

- Opérations de base

- Réalisation

Structure de données

- En informatique, une structure de données définit la manière avec laquelle les données sont stockées et organisées.
- Cette organisation des données est faite grâce à des opérations. Chaque opération permet de préciser la façon d'accéder aux données.
- Exemples d'opérations:
 - récupérer le premier élément
 - supprimer le dernier élément
 - connaître le nombre d'éléments
 - *etc*
- Exemples de structures: liste, tableau, pile, file, arbre, graphe, etc

Les piles - Exemples de la vie courante

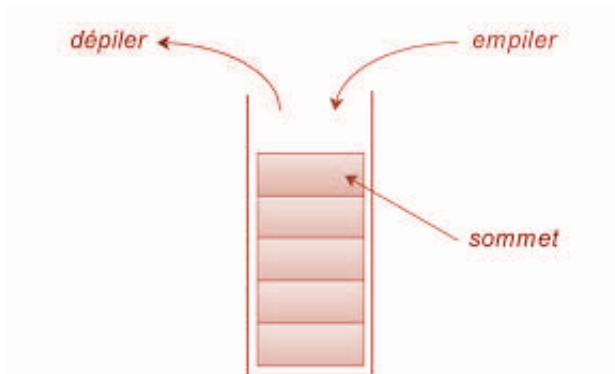


Les piles

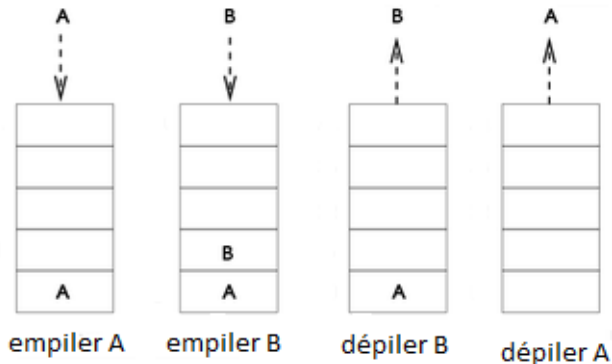
- Une **pile** (**stack** en anglais) est une séquence d'éléments (structure linéaire) dans laquelle on ne manipule qu'un élément particulier, qui est l'extrémité appelé **sommet** de la pile:
 - un élément ne peut être ajouté qu'au niveau du sommet de la pile
 - un élément ne peut être retiré qu'au niveau du sommet de la pile
- Une pile est une structure de données qui se base sur le principe **LIFO** pour **Last In, First Out** ou **dernier arrivé premier servi**, *c'est-à-dire* les derniers éléments ajoutés à la pile seront les premiers à être retirés.

Les piles - Empiler/Dépiler

- L'opération consistant à ajouter un nouvel élément au sommet de la pile s'appelle **empiler**
- L'opération consistant à retirer l'élément situé au sommet de la pile s'appelle **dépiler**



Les piles - Empiler/Dépiler



Les piles - Applications

Les Piles sont utilisées dans de nombreuses applications :

- Dans un navigateur web, une pile sert à mémoriser les pages Web visitées: À chaque page visitée, l'adresse de la page est empilée. En appuyant sur le bouton « Afficher la page précédente », l'utilisateur désempile l'adresse de la page.
- La fonction « Annuler » (en anglais Undo) d'un traitement de texte mémorise les modifications apportées au texte dans une pile.
- Dans la compilation des programmes: par exemple, dans le cas de fonctions récursives, les appels récursifs sont mémorisés dans une pile.

Les piles - Réalisation

- La réalisation d'une pile nécessite l'implémentation de certaines opérations de base telles que:
 - créer pile (vide)
 - empiler (push)
 - dépiler (pop)
- On peut également réaliser d'autres opérations telles que:
 - tester si une pile est vide
 - retourner la taille de la pile
 - vider une pile

Les piles - Réalisation

- Il existe plusieurs façons de concevoir une pile: On peut réaliser une pile de taille limitée ou une pile de taille non limitée (pile bornée/pile non bornée).
- Nous allons implémenter une pile dont la taille n'est pas limitée : on peut y empiler autant d'éléments que l'on veut (dans la limite de la mémoire de l'ordinateur).
- Pour cela, nous utiliserons une liste (class «list») PYTHON (structure dynamique) avec **append(e)** pour empiler et **pop()** pour dépiler, ce qui garantit un moindre coût par rapport aux méthodes **insert(0,e)** et **pop(0) /del L[0]**.

Les piles - Réalisation

Écrire les fonctions suivantes:

- `creer_pile()`: permet de créer et retourner une pile vide.
- `empiler(P,e)` : permet d'empiler l'élément `e` au sommet de la pile `P`.
- `depiler(P)`: permet de dépiler et retourner le sommet de la pile `P`.
- `est_vide(P)`: permet de tester si la pile `P` est vide (renvoie un booléen indiquant si la pile est vide).
- `taille_pile(P)`: permet de retourner le nombre d'éléments de la pile.
- `vider_pile(P)`: permet de vider la pile `P`.

NB: Il faut pas oublier le fait qu'on ne peut rien retirer à une pile vide. On peut également écrire une fonction qui permet de retourner le sommet de la pile sans le dépiler.

Les piles - Implémentation (1)

```
def creer_pile():  
    return []  
  
def empiler(P,e):  
    P.append(e)  
  
def est_vide(P):  
    return len(P)==0
```

Les piles - Implémentation (2)

```
def depiler(P):  
    assert est_vide(P) != True, "Attention pile vide"  
    return P.pop()  
  
def taille_pile(P):  
    return len(P)  
  
def vider_pile(P):  
    while (est_vide(P) != True):  
        s=depiler(P)
```

Tester votre pile

- Créer une pile
- Empiler plusieurs éléments
- Tester la fonction taille
- Dépiler et afficher le sommet
- Vider la pile et tester la fonction est_vide.

Les files

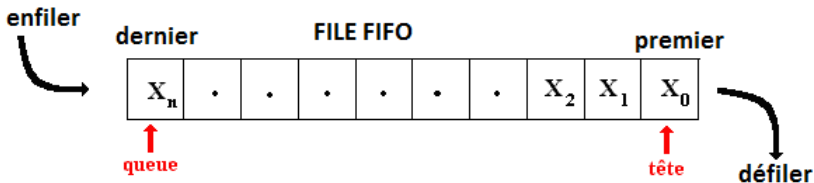


Les files

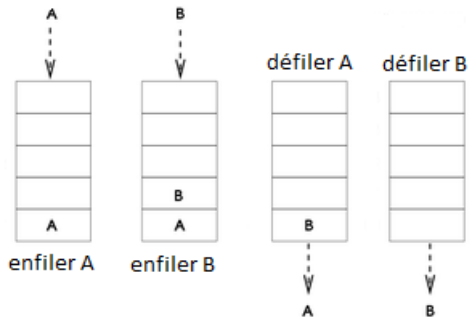
- Une file (**queue** en anglais) est une séquence d'éléments dans laquelle on manipule que deux éléments particuliers, qui sont à l'extrémité appelés **tête** et **queue** de la file:
 - un élément ne peut être ajouté qu'au niveau de la queue de la file.
 - un élément ne peut être retiré qu'au niveau de la tête de la file.
- Une file est une structure de données qui se base sur le principe **FIFO** pour **First In, First Out** ou **premier arrivé premier servi**, *c'est-à-dire* les premiers éléments ajoutés à la file seront les premiers à être retirés.
- C'est le principe des **files d'attente**: les premières personnes arrivées sont les premières personnes à sortir de la file.

Les files - Enfiler/Défiler

- L'opération consistant à ajouter un nouvel élément à la queue de la file s'appelle **enfiler**
- L'opération consistant à retirer l'élément situé à la tête de la file s'appelle **défiler**



Les files - Enfiler/Défiler



Les files - Réalisation

- La réalisation d'une file nécessite l'implémentation de certaines opérations de base telles que:
 - créer file (vide)
 - enfiler (enqueue)
 - défiler (dequeue)
- D'autres opérations peuvent également être réalisées, telles que:
 - tester si une file est vide
 - retourner la taille de la file
 - vider une file

Les files - Réalisation

Écrire les fonctions suivantes:

- `creer_file()`: permet de créer et retourner une file vide.
- `enfiler(F,e)` : permet d'enfiler l'élément `e` à la queue de la file `F`.
- `defiler(F)`: permet de défiler et retourner la tête de la file `F`.
- `est_vide(F)`: permet de tester si la file `F` est vide (renvoie un booléen indiquant si la file est vide).
- `taille_file(F)`: permet de retourner le nombre d'éléments de la file.
- `vider_file(F)`: permet de vider la file `F`.

Les files - Implémentation (1)

```
def creer_file():  
    return []  
  
def enfiler(F,e):  
    F.append(e)  
  
def defiler(F):  
    if est_vide(F)!=True:  
        return (F.pop(0))  
    else :  
        return ("file_vide")  
  
def est_vide(F):  
    return len(F)==0
```

Les files - Implémentation (2)

```
def taille_file(F):  
    return len(F)  
  
def vider_file(F):  
    while (est_vide(F) != True):  
        s=defiler(F)
```

Tester votre file

```
#script principal  
ma_file=creer_file()  
enfiler(ma_file,10)  
enfiler(ma_file,20)  
enfiler(ma_file,30)  
enfiler(ma_file,40)  
print('La taille est ', taille_file(ma_file))  
print("Tete de file est ", defiler(ma_file))  
print('La taille est ', taille_file(ma_file))
```