

Piles et files

S2-3-2 GRAPHERS Piles & Files

Germain Gondor

LYCÉE CARNOT - DIJON, 2021 - 2022

Sommaire

- 1 Introduction
- 2 Les piles et les files

Sommaire

1 Introduction

2 Les piles et les files

Introduction

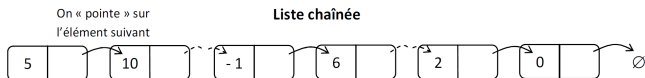
Les listes (pas au sens python) sont des données composites. Elles rassemblent dans une seule structure un ensemble d'entités plus simples qui peuvent être de types différents. En général, on traite plutôt de listes d'éléments de même type, par exemple des listes de nombres.

La liste est un objet de la catégorie des séquences, lesquelles sont des collections ordonnées d'éléments. Cela signifie simplement que les éléments d'une liste sont toujours disposés dans un certain ordre. Contrairement aux chaînes de caractères, les listes sont des séquences modifiables.

On peut implémenter une liste de deux manières :

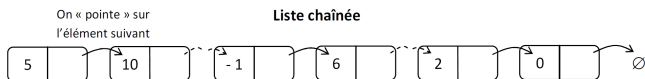
On peut implémenter une liste de deux manières :

- **par une liste chaînée** : Chaque maillon est un objet dans lequel on stocke l'élément et une information pour trouver le suivant (voire aussi le précédent). C'est une structure de données récursive. La liste chaînée n'a pas de taille fixée à l'avance. Seul l'objet « maillon » a une certaine dimension.

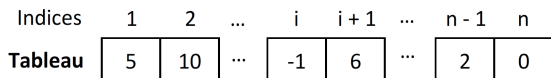


On peut implémenter une liste de deux manières :

- **par une liste chaînée** : Chaque maillon est un objet dans lequel on stocke l'élément et une information pour trouver le suivant (voire aussi le précédent). C'est une structure de données récursive. La liste chaînée n'a pas de taille fixée à l'avance. Seul l'objet « maillon » a une certaine dimension.



- **par un tableau** : Chaque élément de la liste peut être désigné par sa place dans la séquence, à l'aide d'un index. Pour accéder à un élément déterminé, on utilise le nom de la variable qui contient la liste et on lui accole entre deux crochets, l'index numérique qui correspond à sa position dans la chaîne. Le tableau a une dimension fixe. Certains langages (comme Python) permettent de modifier sa taille au cours de son utilisation.



Sommaire

1 Introduction

2 Les piles et les files

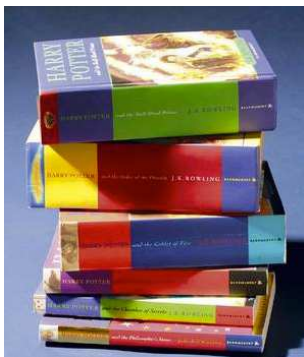
- Définitions
- Principales fonctions associées
- Piles et files définies par listes chaînées
- Pile définie par un tableau P de n éléments
- File définie par un tableau F de n éléments

Définitions

Les piles et les files sont des listes particulières : on accède aux éléments par les extrémités, c'est-à-dire au début ou à la fin.

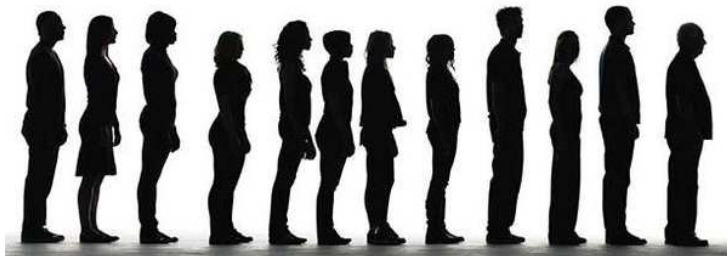
les piles (*stacks*)

Le premier empilé est le dernier à être dépilé
LIFO (Last In first Out)



les files

le premier entré est le premier à sortir
FIFO (First In First Out)



Principales fonctions associées

Pour les piles, les principales fonctions associées aux piles sont :

Principales fonctions associées

Pour les piles, les principales fonctions associées aux piles sont :

- ajouter un sommer un élément x : **empiler** `Pile.append(x)`

Principales fonctions associées

Pour les piles, les principales fonctions associées aux piles sont :

- ajouter un sommer un élément x : **empiler** `Pile.append(x)`
- supprimer du sommet le dernier élément de la pile : **dépiler** `Pile.pop()`.

REMARQUE : on peut récupérer cet élément en l'affectant à une variable : `x=Pile.pop()`

Principales fonctions associées

Pour les piles, les principales fonctions associées aux piles sont :

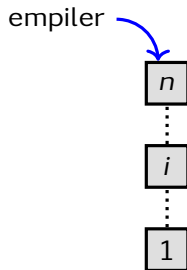
- ajouter un sommer un élément x : **empiler** `Pile.append(x)`
- supprimer du sommet le dernier élément de la pile : **dépiler**
`Pile.pop()`.
REMARQUE : on peut récupérer cet élément en l'affectant à une variable : `x=Pile.pop()`
- tester si la pile est vide : **est_vide** `Pile==[]`

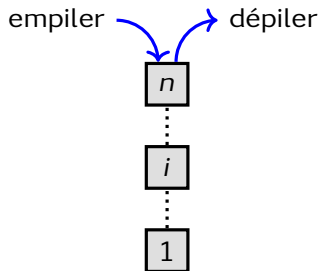
Principales fonctions associées

Pour les piles, les principales fonctions associées aux piles sont :

- ajouter un sommer un élément x : **empiler** `Pile.append(x)`
- supprimer du sommet le dernier élément de la pile : **dépiler**
`Pile.pop()`.
REMARQUE : on peut récupérer cet élément en l'affectant à une variable : `x=Pile.pop()`
- tester si la pile est vide : **est_vide** `Pile==[]`
- tester si la pile est pleine : **est_pleine** avertissement par `overflow` ou `stackoverflow`







Pour les files, les principales fonctions associées aux piles sont :

Pour les files, les principales fonctions associées aux piles sont :

- ajouter en queue : **enfiler**

Pour les files, les principales fonctions associées aux piles sont :

- ajouter en queue : **enfiler**
- supprimer en tête : **défiler**

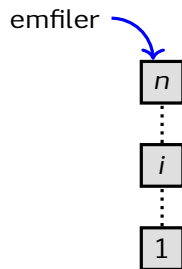
Pour les files, les principales fonctions associées aux piles sont :

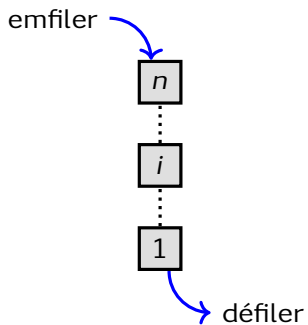
- ajouter en queue : **enfiler**
- supprimer en tête : **défiler**
- tester si la pile est vide : **est_vide** `File==[]`

Pour les files, les principales fonctions associées aux piles sont :

- ajouter en queue : **enfiler**
- supprimer en tête : **défiler**
- tester si la pile est vide : **est_vide** `File==[]`
- tester si la pile est plein : **est_pleine** avertissement par `overflow` ou `stackoverflow`







Piles et files définies par listes chaînées

C'est l'objet du Tp GRAPHE-Tp-1 pour lequel on utilise une classe `Pile` possédant uniquement les méthodes `empiler`, `depiler`, `est-vide` et `sommet` (retournant la valeur du sommet de la pile).

Pile définie par un tableau P de n éléments

Pile définie par un tableau P de n éléments

- l'ordre est déterminé par les indices des éléments du tableau $P[1:n]$

Pile définie par un tableau P de n éléments

- l'ordre est déterminé par les indices des éléments du tableau $P[1:n]$
- le tableau possède un attribut `sommet [P]` qui indexe l'élément le plus récemment inséré

Pile définie par un tableau P de n éléments

- l'ordre est déterminé par les indices des éléments du tableau $P[1:n]$
- le tableau possède un attribut `sommet [P]` qui indexe l'élément le plus récemment inséré
- la pile se constitue des éléments $P[1 : \text{sommet}[P]]$ où $P[1]$ est l'élément de base de la pile (le plus ancien) et $P[\text{sommet}[P]]$ est l'élément situé au sommet (le plus récent)

Pile définie par un tableau P de n éléments

- l'ordre est déterminé par les indices des éléments du tableau $P[1:n]$
- le tableau possède un attribut `sommet [P]` qui indexe l'élément le plus récemment inséré
- la pile se constitue des éléments $P[1 : \text{sommet}[P]]$ où $P[1]$ est l'élément de base de la pile (le plus ancien) et $P[\text{sommet}[P]]$ est l'élément situé au sommet (le plus récent)
- quand `sommet [P] = 0`, la pile est vide

Pile définie par un tableau P de n éléments

- l'ordre est déterminé par les indices des éléments du tableau $P[1:n]$
 - le tableau possède un attribut `sommet [P]` qui indexe l'élément le plus récemment inséré
 - la pile se constitue des éléments $P[1 : \text{sommet}[P]]$ où $P[1]$ est l'élément de base de la pile (le plus ancien) et $P[\text{sommet}[P]]$ est l'élément situé au sommet (le plus récent)
 - quand $\text{sommet}[P] = 0$, la pile est vide
 - Si $\text{sommet}[P]$ est supérieur à n , on dit que la pile déborde
- REMARQUE :** Dépiler une pile vide provoque une erreur ; on parle de débordement négatif.

Fonction Pile_vider

PileVide (P)

- 1: **si** sommet[P] = 0 **alors**
- 2: **renvoi:** Vrai
- 3: **sinon**
- 4: **renvoi:** Faux
- 5: **fin si**

Fonction empiler

Empiler (P,x)

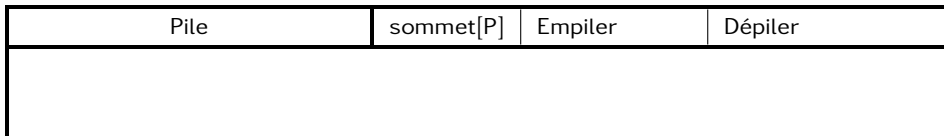
1: $S[P] \leftarrow \text{sommet}[P]+1$

2: $P[\text{Sommet}[P]] \leftarrow x$

Fonction déplier

Dépiler (P)

- 1: **si** PileVide(P) **alors**
- 2: "débordement négatif" (Erreur)
- 3: **sinon**
- 4: sommet[P] \leftarrow sommet[P]-1
- 5: **renvoi:** P[sommet[P]+1]
- 6: **fin si**



Pile							sommet[P]	Empiler	Dépiler	
	1	2	3	4	5	6	7			
P	15	6	2	9				4	on place en 5	on récupère 9

Pile		sommet[P]	Empiler	Dépiler							
	1 2 3 4 5 6 7										
P	<table border="1"> <tr> <td>15</td> <td>6</td> <td>2</td> <td>9</td> <td></td> <td></td> <td></td> </tr> </table>	15	6	2	9				4	on place en 5	on récupère 9
15	6	2	9								
	1 2 3 4 5 6 7										
P	<table border="1"> <tr> <td>15</td> <td>6</td> <td>2</td> <td>9</td> <td>17</td> <td>3</td> <td>12</td> </tr> </table>	15	6	2	9	17	3	12	7	débordement !	on récupère 12
15	6	2	9	17	3	12					
	1 2 3 4 5 6 7										
P	<table border="1"> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>								0	on place en 1	débordement négatif

File définie par un tableau F de n éléments

File définie par un tableau F de n éléments

- la file a un attribut tête [F] qui repère sa tête

File définie par un tableau F de n éléments

- la file a un attribut `tête [F]` qui repère sa tête
- la file a un attribut `queue [F]` qui indexe le prochain emplacement pouvant conserver un nouvel élément.

File définie par un tableau F de n éléments

- la file a un attribut $tête[F]$ qui repère sa tête
- la file a un attribut $queue[F]$ qui indexe le prochain emplacement pouvant conserver un nouvel élément.
- les éléments de la file sont repérés par $tête[F]$, $tête[F+1]$, ..., $queue[F]-1$

File définie par un tableau F de n éléments

- la file a un attribut $tête[F]$ qui repère sa tête
- la file a un attribut $queue[F]$ qui indexe le prochain emplacement pouvant conserver un nouvel élément.
- les éléments de la file sont repérés par $tête[F]$, $tête[F+1]$, ..., $queue[F]-1$
- l'emplacement 1 suit l'emplacement n dans un ordre circulaire

File définie par un tableau F de n éléments

- la file a un attribut tête $[F]$ qui repère sa tête
 - la file a un attribut queue $[F]$ qui indexe le prochain emplacement pouvant conserver un nouvel élément.
 - les éléments de la file sont repérés par tête $[F]$, tête $[F+1]$, ..., queue $[F] - 1$
 - l'emplacement 1 suit l'emplacement n dans un ordre circulaire
 - quand tête $[F] =$ queue $[F]$, la file est vide
- REMARQUE :** au départ, on a tête $[F] =$ queue $[F] = 1$

File définie par un tableau F de n éléments

- la file a un attribut tête $[F]$ qui repère sa tête
- la file a un attribut queue $[F]$ qui indexe le prochain emplacement pouvant conserver un nouvel élément.
- les éléments de la file sont repérés par tête $[F]$, tête $[F+1]$, ..., queue $[F] - 1$
- l'emplacement 1 suit l'emplacement n dans un ordre circulaire
- quand tête $[F] =$ queue $[F]$, la file est vide
REMARQUE : au départ, on a tête $[F] =$ queue $[F] = 1$
- quand tête $[F] =$ queue $[F] + 1$, la file est pleine.

Fonction Enfiler

Enfiler(F,x)

- 1: $F[\text{queue}[F]] \leftarrow x$
- 2: **si** $\text{queue}[F] = \text{longueur}[F]$ **alors** $\text{queue}[F] \leftarrow 1$
- 3: **sinon** $\text{queue}[F] \leftarrow \text{queue}[F] + 1$
- 4: **fin si**

Fonction Défiler

Défiler(F)

- 1: $x \leftarrow F[\text{tête}[F]]$
- 2: **si** tête[F] = longueur[F] **alors** tête[F] $\leftarrow 1$
- 3: **sinon** tête[F] \leftarrow tête[F]+1
- 4: **fin sirenvoi:** x

File	t	q	action précédente
------	---	---	----------------------

File												t	q	action précédente	
P	1	2	3	4	5	6	7	8	9	10	11	12	7	12	Les éléments 15, 6, 2, 8 et 4 ont été enfilés

File												t	q	action précédente	
	1	2	3	4	5	6	7	8	9	10	11	12	7	12	Les éléments 15, 6, 2, 8 et 4 ont été enfilés
P							15	6	2	8	4				
	1	2	3	4	5	6	7	8	9	10	11	12	7	3	Les éléments 17, 3 et 5 ont été enfilés
P	3	5					15	6	2	8	4	17			

File												t	q	action précédente	
P	1	2	3	4	5	6	7	8	9	10	11	12	7	12	Les éléments 15, 6, 2, 8 et 4 ont été enfilés
P	3	5					15	6	2	8	4	17	7	3	Les éléments 17, 3 et 5 ont été enfilés
P	3	5					15	6	2	8	4	17	8	3	l'élément 15 à été défilé