Implémentations et applications des piles et des files

I - Les piles

A - Définition

Def: Une pile est une structure de données abstraite dans laquelle les insertions et les suppressions d'éléments se font à une seule extrémité, appelé sommet de pile. On appelle aussi une pile LIFO (Last In, First Out).

Ex: Une pile d'assiette est une pile

Opérations sur une pile :
- insérer (ou empiler) : ajout d'un élément
- supprimer (ou dépiler) : suppression et renvoie du sommet de pile
- pile_vide : crée une nouvelle pile.

On peut aussi avoir les opérations suivantes :
- est_vide : teste si la pile est vide
- sommet : renvoie l'élément au sommet de la pile, sans la modifier

B - Exemples d'implémentation :

1 - Représentation contiguë

- Les données sont stockées dans un tableau, avec un indice du sommet de pile en plus.

Ex: (Python), pour une pile de type ~~int~~

```
def pile_vide (tailleMax : int) → (Array [int], int)
    return (numpy.zeros((tailleMax)), 0)
```

```
def empiler (p: Array [int], i: int, e : int) → (Array [int], int)
    tab[i] = e
    return tab, i+1
```

```
def depiler (p: Array [int], i: int) → (Array [int], int, int)
    e = tab[i]
    return (tab, i-1, e)
```

2 - Représentation chaînée

Les éléments sont chaînés entre eux, le sommet d'une pile non vide est le premier élément

Ex (avec des pointeurs)

Ex: Liste en Ocaml

```
let pile_vide () = []

let empile (l: int list) (e: int) : int list = e :: l

let depile (l: int list) : int list x int = (tl l, hd l)
```

3 - Comparaisons des deux méthodes

- Dans les deux cas, les opérations se font en complexité constante $\mathcal{O}(1)$.
- La représentation par tableau est plus compacte en mémoire, mais nécessite de fixer une taille maximale dès la création, contrairement à la représentation en liste chaînée.

C - Applications :

Voici quelques exemples d'utilisation de piles

- Exécution de processus :
  - pile et tas dans la mémoire (déf modifiée : on peut lire les valeurs ailleurs qu'au sommet)
  - pile d'appels de fonctions
- Algorithmes :
  - parcours de graphes (ex : DFS)
  - balayage de Graham

  DEV 1 : Implémentation (recherche d'enveloppe convexe d'un nuage de points)
- Langages formels :
  - automates à pile : reconnaissent les langage Hors Contexte.

II - Les files

A - Définition

Def : Une file est une structure analogue à une pile, à la différence près que les ajouts sont effectués à une extrémité, et les accès et suppressions à l'autre extrémité.
On appelle aussi une file FIFO pour First In, First Out.

Ex : Une file d'attente est une file

Opérations sur une pile :
- insérer (ou enfiler) : ajout d'un élément
- supprimer (ou dépiler) : suppression et renvoi de l'élément en début de file
- file_vide : crée une nouvelle file

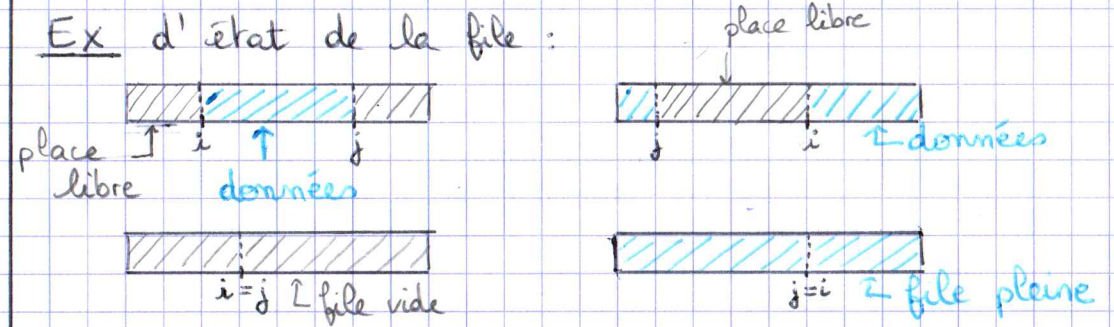
De manière analogue à la pile on peut aussi avoir les opérations : est_vide et premier

B - Exemples d'implémentations :

1 - Représentation contiguë :

Les données sont stockées dans un tableau. On conserve de plus les indices i et j du début et de la fin de la file.

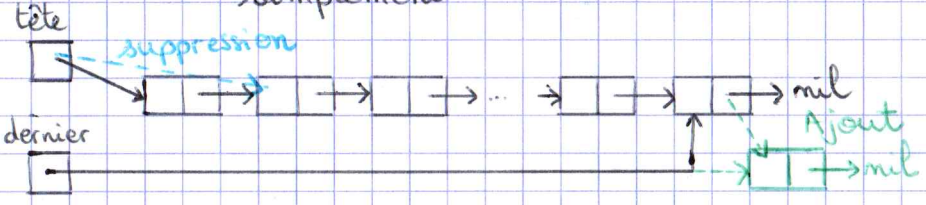
Ex d'état de la file :



2 - Représentations chaînées :

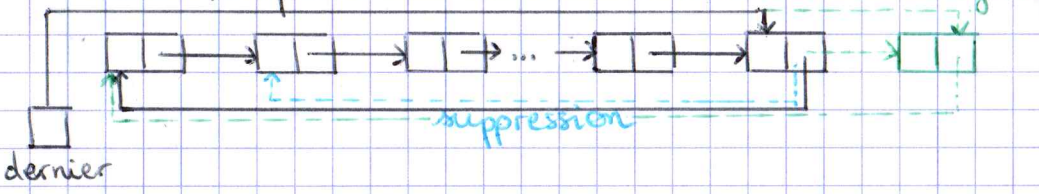
Plusieurs possibilités :
- Liste ~~doublement~~ chaînées : simplement

Ex :



- Liste simplement chaînée circulaire

Ex :



3- Comparaisons des représentations

Comme pour les piles, dans les différentes représentations, les opérations sont effectuées en un nombre constant d'étapes.

La représentation contiguë est plus compacte mais nécessite de fixer une taille dès la création.

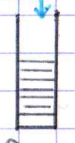
C- Applications :

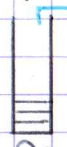
- Ordonnancement de tâches (ex: ordonnancement de processus)
- Zones tampons (buffers)
  ex: ~~routeurs~~ informations reçues par un routeur
  interface clavier - ordinateur
  etc.
- Parcours de graphes (ex: BFS).

III- Lien entre pile et file et extensions :

A- Représentation des piles en files et vice-versa

Propriété: Une file peut être représentée par deux piles





enfiler : $\mathcal{O}(1)$
defiler : $\mathcal{O}(1)$ (amorti)

Quand P_2 vide, on dépile intégralement P_1 dans P_2.

Propriété: Une pile peut être représentée par deux files

Deux méthodes: On pose n taille de la pile

* empiler efficace:
empiler elt → enfiler f_1 elt — complexité : $\mathcal{O}(1)$
depiler → tant que taille $f_1 > 1$: défiler f_1 dans f_2; échanger f_1 et f_2 — $\mathcal{O}(n)$

* depiler efficace :
empiler : → enfiler f_2 ; défiler f_2 dans f_1 intégralement
$\mathcal{O}(n)$ échanger f_1 et f_2.
depiler → défiler f_1. $\mathcal{O}(1)$

B- Structures dérivées :

1- Files à double entrée :

Def: Une file à double entrée (aussi appelée double-ended queue ou deque) est une structure dans laquelle les ajouts comme les suppressions peuvent être exécutés à chaque extrémité.

Implémentations : tableau avec indice de début et de fin ou liste doublement chaînée par ex.

Applications : • Peut remplacer une pile comme une file.
• Algorithmes de "work stealing" pour de la parallélisation

2- Files de priorité :

Def Une file de priorité est une structure de données qui permet de gérer un ensemble S de données, dont chacune à une valeur associée nommée clé.
Cette structure supporte les opérations suivantes :
- Inserer(S, x) insère l'élément x à S
- Minimum(S) renvoie l'élément de clé minimale
- ExtraireMin(S) renvoie et supprime l'élément de clé minimale
- DiminuerClé(S, x, k) modifier la clé de x pour k, supposée inférieure à la clé d'origine)

Implémentation : DEV2 : Implémentation d'une file de priorité avec un tas binomial

Applications : Ordonnancement avec gestion de priorité (par ex: ordonnancement de processus).