Numérique et science informatique
Classe de Terminale

Lycée hoche

année scolaire 2024-2025

Contents

On se limite aux types abstraits de données qui stockent les données sous forme de collection unidimensionnelle ; ce sont les types linéaires, dont nous présentons les listes, les piles et les files.

1 Liste

Définition

Une liste est une collection finie de données.Elle permet de stocker des données et d'y accéder directement.C'est un **type abstrait de données** qui est:

- linéaire: Les données sont stockées dans une structure unidimensionnelle.
- indexé: Chaque donnée est associée à une valeur.
- ordonné: Les données sont présentées les unes après les autres.

Remarque: il ne faut pas confondre le type abstrait `liste` ci-dessus avec le même mot qui désigne en Python (en français) la structure de donnée `list` vue l'an dernier.

On appelle "tête" le premier élément de la liste et "queue" la liste privée de son premier élément.

on peut noter une liste sous la forme (a , Lst1) où a est la tête de la liste et Lst1 la queue.

A noter que dans l'écriture précédente , Lst1 est elle-même une liste qui peut s'écrire (b , Lst2) où Lst2 est une liste etc.

Il est seulement possible d'ajouter et de lire une donnée en tête de liste .

Cinq `primitives` (ou fonctions) permettent de définir le type abstrait de données `liste`:

- listeCree() : crée la liste vide.
- listeAjout(liste,element) :ajoute un élément en tête de liste.
- listeTete(liste) :renvoie la valeur de l'élément en tête de liste.
- listeQueue(liste) :renvoie la liste privée de son premier élément.
- listeEstVide(liste) :renvoie vrai si la liste est vide ,faux sinon.

Remarque : Le type abstrait de données liste est **non mutable**.

De nombreux autres algorithmes sur les listes peuvent être ajoutés en fonction des besoins :

- Renvoyer une liste dont l'ordre des éléments a été inversé ;
- Renvoyer la longueur, c'est-à-dire le nombre des éléments de la liste ;
- Accéder au ième élément d'une liste ;
- Rechercher un élément dans une liste en renvoyant au choix :
  - "Vrai" si l'élément figure dans la liste , " Faux " sinon ;
  - La position de la première occurrence de l'élément recherché ;
  - La liste des positions de toutes ses occurrences ;
- supprimer la tête d'une liste et renvoyer cette tête.
- Renvoyer une nouvelle liste correspondant à la liste d'origine à laquelle un élément a été ajouté à la fin .

2 Pile

2.1 présentation

La pile, comme la liste, permet de stocker des données et d'y accéder. La différence se situe au niveau de l'ajout et du retrait d'éléments.

On parle de mode LIFO (Last In, First Out, donc, dernier arrivé, premier sorti), c'est-à-dire que le dernier élément ajouté à la structure sera le prochain élément auquel on accèdera.

Les premiers éléments ayant été ajoutés devront « attendre » que tous les éléments qui ont été ajoutés après eux soient sortis de la pile.

Contrairement aux listes, on ne peut donc pas accéder à n'importe quelle valeur de la structure (pas d'index). Pour gérer cette contrainte, on définit alors le sommet de la pile qui caractérise l'emplacement pour ajouter ou retirer des éléments.
On peut s'imaginer une pile d'assiettes pour mieux se représenter mentalement cette structure. On ajoute des nouvelles assiettes au sommet de la pile, et quand on veut en retirer une, on est obligé de prendre celle située au sommet.

Définition

Une Pile est une collection finie de données.
Le **sommet** de la pile est le dernier élément ajouté qui est aussi le seul élément auquel on peut accéder, ou qu'on peut retirer.Quand un élément est ajouté à la pile, il devient le nouveau sommet, et l'ancien sommet est alors son élément **suivant**.Quand un élément est retiré de la pile ,le nouveau sommet est l'élément qui suivait le sommet avant le retrait.

Six primitives permettent de définir le type abstrait de données:

- pileCree() : crée une pile vide.
- pileTaille(P) : renvoie le nombre d'éléments contenus dans la pile.
- pileEstVide(P) :renvoie vrai si la pile est vide , faux sinon.
- pileEmpiler(P,element) :ajoute un élément au sommet de la pile (qui devient le nouveau sommet)
- piledepiler(P) :retire et renvoie le sommet de la pile.
- pileSommet(P) :renvoie l'élément qui se trouve au sommet de la pile sans le retirer.

2.2 Remarques

La pile est utile dans différents types de problèmes:

- algorithme d'un navigateur pour pouvoir mémoriser les pages web et revenir en arrière (ou réavancer) sur certaines pages.
- stocker des actions et les annuler (ou les réappliquer) , sur l'ordinateur (CTRL+Z et CTRL+Y).
- Coder une calculatrice en notation polonaise inversée.
- algorithme du parcours en profondeur pour les arbres et les graghes par exemple,pour résoudre un labyrinthe ,trouver un trajet sur une carte.
- Ecrire des versions itératives de certains algorithmes récursifs.
- illustration du fonctionnement de la pile d'appels des fonctions lors de l'exécution d'un programme.

3 File

3.1 Présentation

La file, comme la liste, permet de stocker des données et d'y accéder. La différence se situe au niveau de l'ajout et du retrait d'éléments.
On parle de mode FIFO (First in, First out, donc, premier arrivé, premier sorti), c'est-à-dire que le premier élément ayant été ajouté à la structure sera le prochain élément auquel on accèdera.

Les derniers éléments ajoutés devront « attendre » que tous les éléments ayant été ajoutés avant eux soient sortis de la file. Contrairement aux listes, on ne peut donc pas accéder à n'importe quelle valeur de la structure (pas d'index).
Pour gérer cette contrainte, la file est caractérisée par deux « emplacements » :

- la tête de file, sortie de la file (début de la structure), où les éléments sont retirés ;
- le bout de file, entrée de la file (fin de la structure), où les éléments sont ajoutés.

On peut s'imaginer une file d'attente, dans un cinéma par exemple. Les premières personnes à pouvoir acheter leur place sont les premières arrivées, et les nouveaux arrivants se placent au bout de la file.

Définition

Une file est une collection de données.On appelle **tête de file** le premier élément de la structure et **bout de file** le dernier élément.Quand un élément est ajouté à la fin de la file, on l'ajoute en bout de file et il devient le nouveau bout de file, c'est-à-dire l'élément suivant l'élément précédemment situé en bout de file.

Quand un élément est retiré de la file,on le sélectionne à la tête de la file et la nouvelle tête est l'élément qui suivait l'ancienne tête.Lorsqu'on ajoute un élément à une file vide,celui-ci est donc à la fois tête et bout de file.

Six primitives permettent de définir le type abstrait de données:

- fileCree() : crée une pile vide.
- fileTaille(F) : renvoie le nombre d'éléments contenus dans la file.
- fileEstVide(F) :renvoie vrai si la file est vide , faux sinon.
- fileAjouterFin(F) :ajoute un élément au bout de la file (qui devient le nouveau bout de fil)
- FileretirerTete(F) :retire et renvoie à la tête de la file.
- fileTete(P) :renvoie l'élément qui se trouve à la tête de la file (sans le retirer).

3.2 Remarques

La file est utile dans différents types de problèmes :
• pour une imprimante, gestion de la file d'attente des documents à imprimer ;

• modélisation du jeu de la bataille (on révèle la carte au-dessus du paquet et on place celles gagnées en dessous...) ;

• gestion de mémoires tampon, pour gérer les flux de lecture et d'écriture dans un fichier, par exemple ;
• matérialisation d'une file d'attente, pour un logiciel (visioconférence par exemple) ou un jeu (gestion des connexions des utilisateurs, des tours de jeu...),...
• algorithme du parcours en largeur pour les arbres et les graphes, par exemple, pour trouver le plus court trajet sur une carte, ou récupérer les valeurs d'une structure

4 Implantation de structures de données

4.1 Implantation du type abstrait de données liste avec les listes Python

```
def listeCree() :
    '''
    crée une liste vide en s appuyant sur les listes Python
    '''
    return []


def listeAjout(liste, element) :
    ''' Paramètres
    liste : une liste à laquelle on souhaite ajouter un élément
    element : lélément à ajouter, de type quelconque
    '''

    # ajoute un élément en tête de liste
    liste.append(element)


def listeTete(liste) :
    ''' Paramètres
    liste : une liste
    '''
    #renvoie la valeur de lélément en tête de liste
    if not listeEstVide(liste) :
        return liste[-1]
    else :
        return None


def listeQueue(liste) :
    ''' Paramètres
    liste : une liste    :   renvoie la queue de la liste
    '''
    return liste[:-1]
    # il peut être pertinent d utiliser la syntaxe liste.pop()
    # afin d éviter le slicing

def listeEstVide(liste) :
    ''' Paramètres
    liste : une liste
    renvoie ``True`` si la liste est vide, ``False`` sinon
    '''
    return len(liste)==0
```

4.2 Implantation du type abstrait de données 'pile' avec les listes Python

```
def pileCree() :

    '''
    Crée une pile vide en s'appuyant sur les listes Python
    '''
    return []


def pileTaille(pile):


    '''
     Paramètres :   pile : Une pile, telle que créée dans ce module (
   utilisant une list Python)

    Description : La pile dont on veut connaître le nombre d'éléments
    Renvoie le nombre d'éléments contenus dans la pile.
    '''
    return len(pile)

def pileEstVide(pile) :

    '''
    Paramètres      pile : Une pile, telle que créée dans ce module (
   utilisant une list Python)

    Description : La pile dont on souhaite déterminer si elle est vide.
    Renvoie ``True`` si la pile est vide, et ``False`` sinon
    '''
    return pileTaille(pile) == 0


def pileEmpiler(pile, element) :

    '''
    Paramètres  -pile : Une pile, telle que créée dans ce module (
   utilisant une list Python)

    Description : La pile sur laquelle on souhaite empiler un élément
    element : N'importe quel type de données.
    Description : L'élément à empiler dans la pile.
    Empile un élément dans la pile passée en paramètres.
    '''
    pile.append(element)



def pileDepiler(pile) :
    '''
    Paramètres     pile : Une pile, telle que créée dans ce module (
   utilisant une list Python)
```

```
    Description : La pile sur laquelle on souhaite dépiler un élément
    Dépile (supprime de la pile) l'élément au sommet de la pile et le
    renvoie.
    '''
    if pileEstVide(pile) :
        return None
    else :
        return pile.pop()

def pileSommet(pile):
    '''
    Paramètres   pile : Une pile, telle que créée dans ce module (
    utilisant une list Python)

    Description : La pile dont on veut connaître le sommet.
    Renvoie le sommet de la pile (mais ne le dépile pas).
    '''
    if pileEstVide(pile) :
        return None
    else :
        return pile[len(pile) - 1]
```

4.3 Implantation du type abstrait de données 'file' avec les listes Python

```
# Implantation du type Abstrait de données file avec les listes Python.



def fileCree() :
    '''
    Crée une file vide en s'appuyant sur les listes Python.
    '''

    return []

def fileTaille(file) :
    '''
      Paramètres  -file : Une file, telle que créée dans ce module (
   utilisant donc une listes Python)
      Description : La file dont on veut connaître le nombre d'éléments
      ----  Renvoie le nombre d'éléments contenus dans la file

    '''
    return len(file)


def fileVide(file) :

    '''
    Paramètres  -file : Une file, telle que créée dans ce module (
   utilisant donc une liste Python)

    Description : La file qu'on veut vérifier.  Renvoie 'True' si la
   file est vide, et 'False' sinon
    '''

    return fileTaille(file) == 0


def fileAjouterFin(file, element) :

    '''
     Paramètres  file : Une file, telle que créée dans ce module (
   utilisant donc une liste Python)

    Description : La file à laquelle on souhaite ajouter un élément
   element : N'importe quel type de données.  Description : L'élé
   ment à ajouter au bout de la file.  Ajoute un élément au bout de
   la file passée en paramètres.
    '''
    file.append(element)
```

```
def fileRetirerTete(file) :
    '''
    Paramètres      file : Une file, telle que créée dans ce module (
    utilisant donc une liste Python)

    Description : La file à laquelle on souhaite retirer un élément
    Retirer (supprime de la file) et renvoie l'élément en tête (situé au
    début)      de la file.
    '''

    if fileVide(file) :
        return None
    else :
        return file.pop(0)


def fileTete(file) :
    '''
     Paramètres      file : Une file, telle que créée dans ce module (
    utilisant donc une liste Python)

     Description : La file dont on veut connaître la tête.      Renvoie l
    'élément en tête (situé au début) de la file (mais ne le supprime pas
    )
    '''

    if fileVide(file) :
        return None
    else :
        return file[0]
```