Algorithme et Structures de Données (ASD) Structures linéaires

Dr Ibrahima Gaye

May 13, 2022

Outline

Introduction

Les listes

- Une liste est la forme la plus simple d'organisation de données que l'on puisse rencontrer.
- Les données sont stockées les unes à la suite des autres dans des places et permettent divers traitements séquentiels.
- L'ordre des éléments dans une liste ne dépend pas des éléments eux-mêmes, mais de la place de ceux-ci dans la liste.

Introduction

Description des listes

- Elle consiste la façon d'accéder aux éléments de la liste.
- Deux de façons de décrire une liste :
  - Itérativement
  - Récursivement

Introduction

Implémentation des listes

- Plusieurs représentions en mémoire :
  - Statique : à l'aide des tableaux (accès direct)
  - Dynamique : à l'aide des listes chainées (accès séquentiel)
  - Hybride : une combinaison des deux représentations

Description des listes

Description récursivement

TAD

Types

liste, place

Utilise

élément

Opérations

$listevide : \rightarrow liste$

$cons : liste \times élément \rightarrow liste$

$fin : liste \rightarrow liste$

$tête : liste \rightarrow place$

$contenu : place \rightarrow élément$

$premier : liste \rightarrow élément$

$succ : place \rightarrow place$

Description récursivement

TAD

Préconditions

fin(l) est-défini-ssi l $\neq$ *listevide*
tête(l) est-défini-ssi l $\neq$ *listevide*
premier(l) est-défini-ssi l $\neq$ *listevide*

Axiomes

premier(l) = contenu(tête(l))
fin(cons(e,l)) = l
premier(cons(e,l)) = e
succ(tête(l)) = tête(fin(l))

Avec

liste l
élément e

Description récursivement

Opérations : interne et observateur

- Liste et Place comme types définis
- élément comme type prédéfini
- tête et succ comme opérations internes de place
- contenu comme observateur de place
- listevide, fin et cons comme opérations internes de liste
- premier comme observateur de liste

Description récursivement

- Ces opérations n'étant pas définies partout, il y bien sur des préconditions
- Les rôles :
  - listevide crée une liste sans éléments (une sorte de "constructeur")
  - tête permet de récupérer la première place (celle de tête)
  - contenu permet d'obtenir l'élément d'une place
  - premier permet d'obtenir le premier élément d'une liste (sans place intermédiaire)

Description récursivement

- ☛ fin permet de détruire l'élément de tête et de récupérer la liste restante
- ☛ cons permet d'ajouter un élément en première place (en l'insérant devant la liste existante)
- ☛ succ permet de passer à la place suivante

Description des listes

Description itérativement

TAD

types

liste, place

utilise

entier, élément

opérations

$listevide : \rightarrow liste$

$accès : liste \times entier \rightarrow place$

$contenu : place \rightarrow élément$

$ième : liste \times entier \rightarrow élément$

$longueur : liste \rightarrow entier$

$supprimer : liste \times entier \rightarrow liste$

$insérer : liste\ x\ entier\ x\ élément \rightarrow liste$

$succ : place \rightarrow place$

Description itérativement

TAD

préconditions

accès(l,k) est-défini-ssi $l \neq$ *listevide &* $1 \preceq k \preceq$ *longueur(l)*

supprimer(l,k) est-défini-ssi $l \neq$ *listevide &* $1 \preceq k \preceq$ *longueur(l)*

insérer(l,k,e) est-défini-ssi $1 \preceq k \preceq$ *longueur(l)+1*

Description itérativement

TAD

axiomes

$$longueur(listevide) = 0$$
$$longueur(supprimer(l,k)) = longueur(l)\text{-}1$$
$$longueur(insérer(l,k,e)) = longueur(l)+1$$
$$1 \preceq i \prec k \Rightarrow ième(supprimer(l,k),i) = ième(l,i)$$
$$k \preceq i \preceq longueur(l)\text{-}1 \Rightarrow ième(insérer(l,k,e),i) = ième(l,i\text{-}1)$$
$$1 \preceq i \prec k \Rightarrow ième(insérer(l,k,e),i) = ième(l,i)$$
$$k = i \Rightarrow ième(insérer(l,k,e),i) = e$$
$$k \prec i \preceq longueur(l)+1 \Rightarrow ième(supprimer(l,k),i) = ième(l,i+1)$$
$$contenu(accès(l,k)) = ième(l,k)$$
$$succ(accès(l,k)) = accès(l,k+1)$$

Description itérativement

TAD

avec

liste l

entier i,k

élément e

Description itérativement

- Une autre forme d'implémentation des listes linéaires.
- l'opération de base n'est plus l'accès à la première place d'une liste mais l'opération accès qui renvoie la *Kième* place de cette liste.
- Le type abstrait liste itérative bien que plus adapté à d'autres fonctionnements, permet aussi de décrire des traitements récursifs.
- Les même sont décrites par les deux descriptions.
- Seule la manière de s'en servir diffère.

Description itérativement

- Inversement, nous pouvons passer des opérations itératives en récursives. Exemple : **insérer (l,i,e)** va devenir :

  l2 $\leftarrow$ liste-vide
  On répète le bloc suivants i-1 fois

  $l2 \leftarrow cons\ (premier\ (l), l2)$
  $l \leftarrow fin\ (l)$

  l $\leftarrow$ cons(e,l)
  On répète le bloc suivants i-1 fois

  $l \leftarrow cons\ (premier\ (l2), l)$
  $l2 \leftarrow fin\ (l2)$

Extensions du type liste

Quelques opérations

- Souvent, on a besoin d'opérations complémentaires sur les listes comme :
  - la concaténation de deux listes
  - la recherche d'un élément dans une liste.
- on déclare ce que l'on appelle des extensions au type

Extensions du type liste

Concaténation dans une description itérativement

- ☞ Concaténer deux listes est l'opération qui permet de les rassembler en les mettant bout à bout.
- ☞ Les éléments de chacune conservent leur place d'origine au sein de leur propre liste.
- ☞ La deuxième liste étant accrochée à la suite de la première.

Concaténation dans une description itérativement

opération

$$concaténer : liste \times liste \rightarrow liste$$

axiomes (Description récursivement)

$$concaténer(listevide,l) = l$$
$$concaténer(cons(e,l),l2) = cons(e,concaténer(l,l2))$$

axiomes (Description itérativement)

$$longueur(concaténer(l,l2)) = longueur(l)+longueur(l2)$$
$$1 \prec i \prec longueur(l) \Rightarrow ième(concaténer(l,l2),i) = ième(l,i)$$
$$longueur(l)+1 \prec i \prec longueur(l)+longueur(l2) \Rightarrow$$
$$ième(concaténer(l,l2),i) = ième(l2,i\text{-}longueur(l))$$

avec

$$liste\ l,l2\ ;\ entier\ i\ ;\ élément\ e$$

Extensions du type liste

Recherche d'un élément

- La recherche consiste à trouver un élément dans une liste et à retourner sa place si celui-ci existe dans la liste.
- Le problème est que la recherche n'est pas définie pour un élément non présent.
- Il faut donc une précondition sur la recherche
  - L'opération auxiliaire existe sera utilisée.

Recherche d'un élément

opérations

$$rechercher : élément \times liste \rightarrow place$$
$$existe : élément \times liste \rightarrow booléen$$

préconditions

$$rechercher(e,l)\ est\text{-}défini\text{-}ssi\ existe(e,l) = vrai$$

axiomes (Liste récursive)

$$existe(e,listevide) = faux$$
$$e = e2 \Rightarrow existe(e2,cons(e,l)) = vrai$$
$$e \neq e2 \Rightarrow existe(e2,cons(e,l)) = existe(e2,l)$$
$$existe(e,l) = vrai \Rightarrow contenu(rechercher(e,l)) = e$$

Recherche d'un élément

axiomes (Liste itérative)

$$existe(e,listevide) = faux$$
$$e = e2 \Rightarrow existe(e2,insérer(l,i,e)) = vrai$$
$$e \neq e2 \Rightarrow existe(e2,insérer(l,i,e)) = existe(e2,l)$$
$$existe(e,l) = vrai \Rightarrow contenu(rechercher(e,l)) = e$$

avec

$liste\ l$
$entier\ i$
$élément\ e,e2$

Recherche d'un élément

- Les listes comme tous les autres types de données sont implémentables de différentes manières.
- Deux formes basiques sont la représentation
  - Statique : à l'aide de tableaux ;
  - Dynamique : à l'aide de pointeurs et d'enregistrements ;
  - Hybrides : statique + dynamique

Représentation (Implémentation) des Listes

Déclaration de la liste

Constantes

Nbmax = 20

Types

*TElement = . . . /*Définition du type des éléments */*
Tliste = Enregistrement / Définition du type Tliste */*
elts : Tableau[Nbmax] de TElément
longueur : Entier
Fin Enregistrement

Variable

liste : Tliste

Représentation de la liste statique

☞ Ce qui correspondrait à la structure de la figure ci-dessous

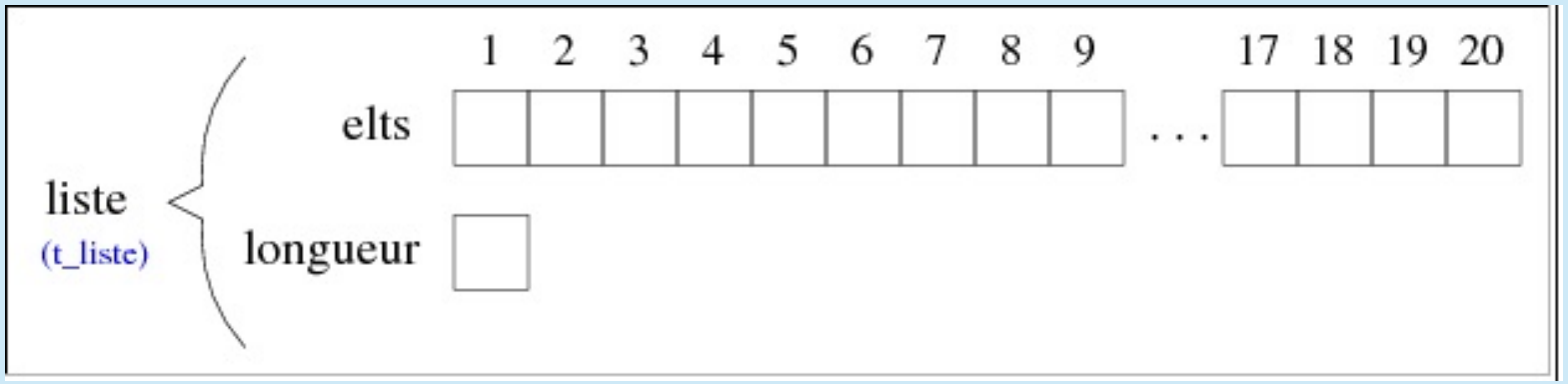

Figure: Représentation statique d'une liste

Représentation de la liste statique

- ☞ Le problème posé par les tableaux est la nécessité d'un surdimensionnement.
- ☞ Une taille supérieure à celle de la liste pour un éventuel ajout.
- ☞ C'est l'utilité de la variable longueur qui contiendra toujours la taille de votre liste.

Représentation de la liste statique

Remarques

- Pour insérer ou supprimer une donnée, vous devrez décaler dans un sens où dans l'autre tous les éléments se trouvant entre celle-ci et la fin de votre liste, ce qui ne rend pas cette représentation très performante en cas de modifications fréquentes des éléments.
- Contrairement à ce que l'on pourrait croire, cette représentation est parfaitement adaptée aux listes récursives
  - Longueur fait office de place de tête et
  - Il n'a aucun transfert de valeur a effectuer au milieu du tableau.

Représentation (Implémentation) des Listes

Déclaration de la liste

Types

TElement = ... / Définition du type des éléments */*
*TPliste = *Tliste /* Définition du type pointeur TPliste */*
TListe = Enregistrement / Définition du type TListe */*
elt : TElement
suiv : TPliste
FinEenregistrement

Variables

liste : TPliste

Représentation de la liste dynamique

- Ce qui correspondrait à la structure de la figure ci-dessous

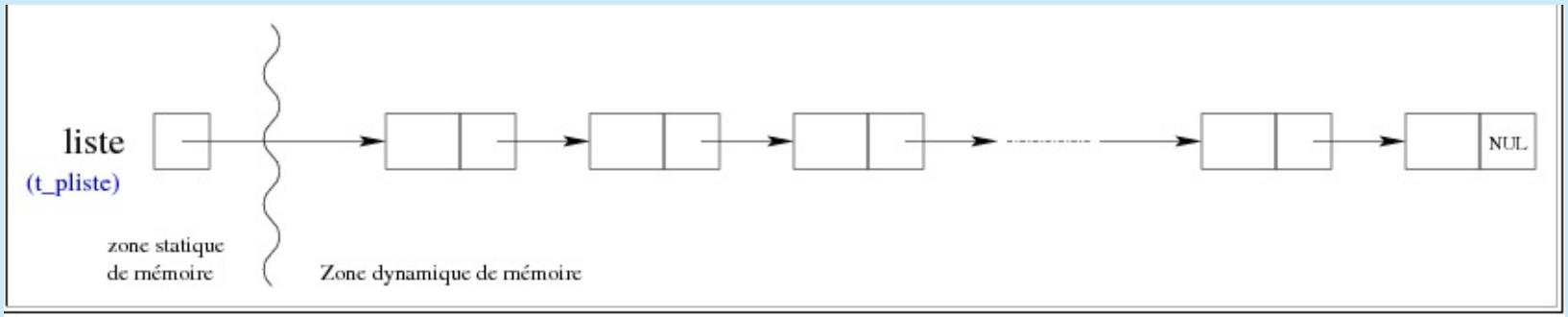

Figure: Représentation dynamique d'une liste

- Le pointeur NUL représente la fin de liste (listevide)

Représentation de la liste dynamique

Remarques

- Nécessite plus de places de la représentation statique
  - on doit stocker l'élément et un (des) pointeur(s)
- Mais le nombre d'éléments est toujours celui de la liste, ni plus ni moins. Contrairement à l'implémentation statique pour laquelle il faut surdimensionner le tableau.
- L'inconvénient majeur est de ne pas pouvoir accéder au Kième élément directement.
- Mais il est facile de concaténer deux chaînes, d'ajouter ou de supprimer un élément sans avoir à tout décaler.
- cette représentation est de plus très bien adaptée aux traitements récursifs.

Représentation (Implémentation) des Listes

Représentation hybride des listes

Sentinelle

- ☞ Une possibilité est de ne pas utiliser un pointeur sur l'enregistrement, mais directement un enregistrement pour générer la tête de liste.
- ☞ L'avantage est de ne pas avoir besoin de traitement particulier en insertion devant le premier élément.

Variable

TListe liste

Représentation hybride des listes

Liste circulaire

- Le dernier pointeur n'est pas NUL, mais il pointe sur le premier élément de la liste.
- Le pointeur principal de liste référence le dernier élément et non pas le premier.
- Pour obtenir l'élément de tête, il suffit d'avancer d'un lien.
- Si la liste n'est composée que d'un élément, celui-ci pointe sur lui-même.
- La déclaration est alors la même que pour la représentation dynamique de base.

Représentation hybride des listes

Liste circulaire

☞ Ce qui correspondrait à la structure de la figure ci-après

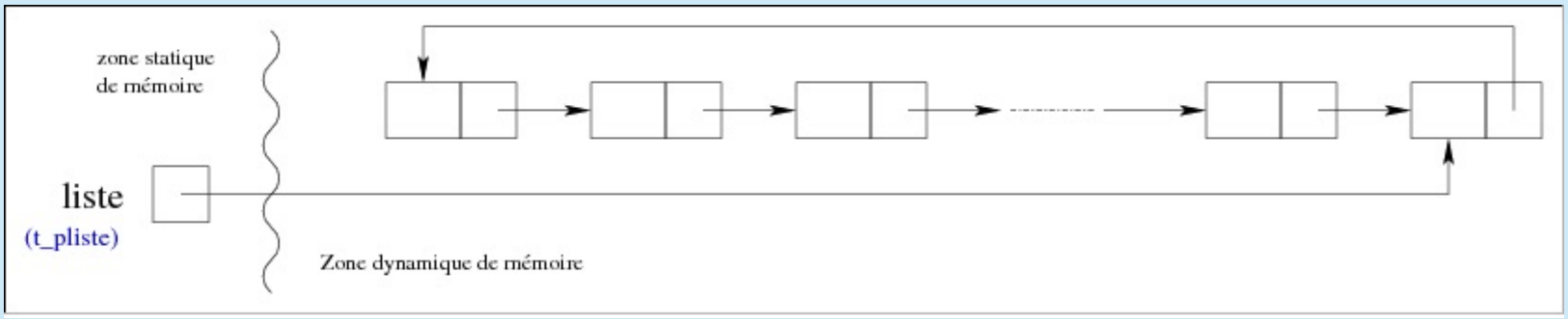

Figure: Représentation d'une liste dynamique circulaire

Représentation hybride des listes

Représentation bi-directionnelle

- Le problème des représentations précédentes est de ne pouvoir aller que dans un sens.
- Il peut être intéressant de revenir sur l'élément précédent, or cette possibilité n'existe pas.
- Il suffit de rajouter un lien en sens inverse.
- Il faut posséder non seulement un pointeur de tête, mais aussi un pointeur de queue.

Représentation hybride des listes

Représentation bi-directionnelle : Déclaration

Types

TElement = ... / Définition du type des éléments */*
*TPenreg = *Tenreg /* Définition du type pointeur *Tenreg */*
TEnreg = Enregistrement / Définition du type TEnreg */*
elt : TElement
suiv,prec : TPenreg
FinEnregistrement

Représentation hybride des listes

Représentation bi-directionnelle : Déclaration

Types

.....

TListe = Enregistrement / Définition du type TListe */*

premier,dernier : TPenreg

FinEnregistrement

Variable

liste : TListe

Représentation hybride des listes

Représentation bi-directionnelle : Déclaration

Ce qui correspondrait à la structure de la figure ci-après.

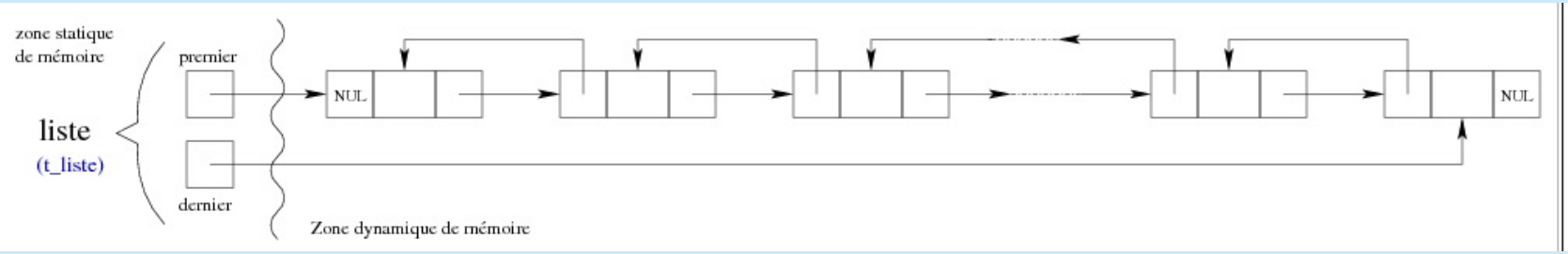

Figure: Représentation d'une liste dynamique doublement chaînée

Représentation hybride des listes

Autres variantes

Il est toujours possible de construire d'autres structures comme par exemple :

une liste circulaire doublement chaînée