Les structures de données

Yann Strozecki
yann.strozecki@uvsq.fr

Septembre 2016

Les structures de données

On va utiliser dans nos algorithmes de nombreuses structures de données présentées ici par ordre de complexité.

- Structures de base : entier, flottant, caractère
- Tableaux : structure linéaire de taille fixée en adressage direct
- Liste, file, pile . . . : structures linéaires de taille variable
- Structure hybride : table de hachage
- Structures arborescentes : tas, arbre binaire de recherche
- Graphe
- Base de données

Structures Linéaires

Eléments d'un même type stockés dans :

- **un tableau**
- **une liste**

Deux cas possibles :

- Eléments triés (l'ordre doit être maintenu)
- L'ordre n'a aucune importance.

Structures de Données Abstraites

- Mise en œuvre d'un ensemble dynamique.
- Définition de la structure par les opérations pour **manipuler** les données.
- Utilisation de librairies.
- On donne les propriétés des opérations et leurs signatures (programme donné par son interface).

Ce qu'on va faire

On doit se servir d'une structure abstraite de donnée uniquement par les fonctions qui la manipule et pas en utilisant sa représentation concrète.
Utile pour programmer simplement.

On va aussi expliquer l'implémentation concrète pour :

- calculer la complexité d'une opération et faire le choix de la meilleure structure pour un problème
- comprendre la quantité de mémoire utilisée
- être capable de définir ses propres structures de données

Ce qu'on va faire

On doit se servir d'une structure abstraite de donnée uniquement par les fonctions qui la manipule et pas en utilisant sa représentation concrète.
Utile pour programmer simplement.

On va aussi expliquer l'implémentation concrète pour :

- calculer la complexité d'une opération et faire le choix de la meilleure structure pour un problème
- comprendre la quantité de mémoire utilisée
- être capable de définir ses propres structures de données

La Pile

Définition

Analogie avec une pile d'assiette :
LIFO (Last In First Out ou **Dernier Arrivé Premier Servi**)

- On ajoute les éléments au dessus de la pile
- On a accès uniquement à l'élément qui est au dessus de la pile (élément le plus récemment inséré).

La Pile : type abstrait

- liste créerPileVide()
- booléen estPileVide(p pile)
- pile push(p pile, n entier) *ajoute la valeur n en haut de la pile p et renvoie la pile*
- entier pop(p pile) *retire l'élément du dessus de la pile et le renvoie*

Propriétés :

- pop(push(p,x)) = x
- estPileVide(créerPileVide()) = vrai

La Pile : type abstrait

- liste créerPileVide()
- booléen estPileVide(p pile)
- pile push(p pile, n entier) *ajoute la valeur n en haut de la pile p et renvoie la pile*
- entier pop(p pile) *retire l'élément du dessus de la pile et le renvoie*

Propriétés :

- pop(push(p,x)) = x
- estPileVide(créerPileVide()) = vrai

Utilisation d'une pile

Exemple

Pile p ; entier x ;

- p = Inserer(p,12) ;

Utilisation d'une pile

Exemple

Pile p ; entier x ;

- p = Inserer(p,12) ;
- p = Inserer(p,34) ;

12	Sommet

Utilisation d'une pile

Exemple

Pile p ; entier x ;

- p = Inserer(p,12) ;
- p = Inserer(p,34) ;
- p = Inserer(p,23) ;

34	Sommet
12	

Utilisation d'une pile

Exemple

Pile p ; entier x ;

- p = Inserer(p,12) ;
- p = Inserer(p,34) ;
- p = Inserer(p,23) ;
- x = Suppression(p) ;

23	Sommet
34	
12	

Utilisation d'une pile

Exemple

Pile p ; entier x ;

- p = Inserer(p,12) ;
- p = Inserer(p,34) ;
- p = Inserer(p,23) ;
- x = Suppression(p) ;
- x = Suppression(p) ;

34	Sommet
12	

Utilisation d'une pile

Exemple

Pile p ; entier x ;

- p = Inserer(p,12) ;
- p = Inserer(p,34) ;
- p = Inserer(p,23) ;
- x = Suppression(p) ;
- x = Suppression(p) ;

12	Sommet

Algorithmes sur des piles

Proposer des algorithmes pour :

- Renvoyer le dernier élément d'une pile
- Retourner un pile.

La File

- Comment modéliser une file d'attente ?
- Comment modéliser un buffer qui stocke les entrées claviers ?

Définition

Analogie avec une file d'attente :
FIFO (First In First Out ou **Premier Arrivé Premier Servi**)

- On rajoute un élément à la fin de la file
- On supprime l'élément qui est en tête de file.

La File

- Comment modéliser une file d'attente ?
- Comment modéliser un buffer qui stocke les entrées claviers ?

Définition

Analogie avec une file d'attente :
FIFO (First In First Out ou **Premier Arrivé Premier Servi**)

- On rajoute un élément à la fin de la file
- On supprime l'élément qui est en tête de file.

La File : type abstrait

- liste créerFileVide()
- booléen estFileVide(f file)
- file enqueue(f file, n entier) *ajoute la valeur n à la fin de la file f et renvoie la file*
- entier dequeue(f file) *retire l'élément au début de la file et le renvoie*

Propriétés :

- f=créerFileVide() ; f = enqueue(f,x) ;f = enqueue(f,y) ; dequeue(f) == x
- estFileVide(créerFileVide()) == vrai

La File : type abstrait

- liste créerFileVide()
- booléen estFileVide(f file)
- file enqueue(f file, n entier) *ajoute la valeur n à la fin de la file f et renvoie la file*
- entier dequeue(f file) *retire l'élément au début de la file et le renvoie*

Propriétés :

- f=créerFileVide() ; f = enqueue(f,x) ;f = enqueue(f,y) ; dequeue(f) == x
- estFileVide(créerFileVide()) == vrai

Utilisation d'une file

Exemple

File f ; entier x ;

- f = Inserer(f,12) ;

Utilisation d'une file

Exemple

File f ; entier x ;

- f = Inserer(f,12) ;
- f = Inserer(f,34) ;

12
Début
Fin

Utilisation d'une file

Exemple

File f ; entier x ;

- f = Inserer(f,12) ;
- f = Inserer(f,34) ;
- f = Inserer(f,23) ;

12 34
Début
Fin

Utilisation d'une file

Exemple

File f ; entier x ;

- f = Inserer(f,12) ;
- f = Inserer(f,34) ;
- f = Inserer(f,23) ;
- x = Suppression(f) ;

12 34 23

Début

Fin

Utilisation d'une file

Exemple

File f ; entier x ;

- f = Inserer(f,12) ;
- f = Inserer(f,34) ;
- f = Inserer(f,23) ;
- x = Suppression(f) ;
- x = Suppression(f) ;

34 23

Début

Fin

Utilisation d'une file

Exemple

File f ; entier x ;

- f = Inserer(f,12) ;
- f = Inserer(f,34) ;
- f = Inserer(f,23) ;
- x = Suppression(f) ;
- x = Suppression(f) ;

23
Début
Fin

Les listes comme type abstrait

On peut manipuler un objet de type liste si on a les opérations suivantes :

- liste créerListeVide()
- booléen estListeVide(l liste)
- entier hd(l liste) *renvoie la valeur du premier élément de la liste l*
- liste tl(l liste) *renvoie la liste l moins son premier élément*
- liste insère(l liste, e entier) *revoie la liste l avec un élément de valeur e inséré au début*

Ces fonctions peuvent être implémentées de nombreuses manières, seule la complexité des opérations changera.

Exercices

Implémentez les fonctions suivantes sur le type abstrait liste.

- Afficher les éléments de la liste.
- Rechercher un élément dans la liste.
- Inverser l'ordre des éléments d'une liste.
- Supprimer un élément sur deux de la liste.

Digression sur les fonctions récursives

Un algorithme est dit récursif quand il s'appelle lui-même.

Exemples classiques :

- Factorielle
- Fibonnaci

Digression sur les fonctions récursives

Un algorithme est dit récursif quand il s'appelle lui-même.

Exemples classiques :

- Factorielle
- Fibonnaci
- Un autre exemple vu en cours ?

Digression sur les fonctions récursives

Un algorithme est dit récursif quand il s'appelle lui-même.

Exemples classiques :

- Factorielle
- Fibonnaci
- Un autre exemple vu en cours ?

Deux conditions nécessaires pour que les fonctions terminent :

- Un des arguments de la fonction décroit.
- Un cas spécial, le cas de base qui ne fait pas d'appel récursif, est prévu quand cet argument atteint une certaine valeur.

Digression sur les fonctions récursives

Un algorithme est dit récursif quand il s'appelle lui-même.

Exemples classiques :

- Factorielle
- Fibonnaci
- Un autre exemple vu en cours ?

Deux conditions nécessaires pour que les fonctions terminent :

- Un des arguments de la fonction décroit.
- Un cas spécial, le cas de base qui ne fait pas d'appel récursif, est prévu quand cet argument atteint une certaine valeur.

Digression sur les fonctions récursives (2)

Calculer la complexité d'un programme récursif :

- On la note $C(n)$ ou n est la taille de l'entrée.
- On analyse le programme pour écrire une équation qui fait dépendre $C(n)$ de $C(k)$ pour $k < n$.
- On résout l'équation.

Par exemple pour Fibonnaci on a
$C(n) = C(n-1) + C(n-2) + 1$

Digression sur les fonctions récursives (2)

Calculer la complexité d'un programme récursif :

- On la note $C(n)$ ou n est la taille de l'entrée.
- On analyse le programme pour écrire une équation qui fait dépendre $C(n)$ de $C(k)$ pour $k < n$.
- On résout l'équation.

Par exemple pour Fibonnaci on a
$C(n) = C(n-1) + C(n-2) + 1$

En modifiant le programme on obtient : $C(n) = C(n-1) + 5$

Digression sur les fonctions récursives (2)

Calculer la complexité d'un programme récursif :

- On la note $C(n)$ ou n est la taille de l'entrée.
- On analyse le programme pour écrire une équation qui fait dépendre $C(n)$ de $C(k)$ pour $k < n$.
- On résout l'équation.

Par exemple pour Fibonnaci on a
$C(n) = C(n-1) + C(n-2) + 1$

En modifiant le programme on obtient : $C(n) = C(n-1) + 5$

Aspect pratique : implémenté avec une pile.
On parle de récursivité terminale quand l'appel récursif est à la fin du code. Cela permet des optimisations par le compilateur.

Digression sur les fonctions récursives (2)

Calculer la complexité d'un programme récursif :

- On la note $C(n)$ ou n est la taille de l'entrée.
- On analyse le programme pour écrire une équation qui fait dépendre $C(n)$ de $C(k)$ pour $k < n$.
- On résout l'équation.

Par exemple pour Fibonnaci on a
$C(n) = C(n-1) + C(n-2) + 1$

En modifiant le programme on obtient : $C(n) = C(n-1) + 5$

Aspect pratique : implémenté avec une pile.
On parle de récursivité terminale quand l'appel récursif est à la fin du code. Cela permet des optimisations par le compilateur.

Liste et récursivité

Une liste a une définition récursive.

Une liste est :

- vide
- un élément de tête et un suite qui est une liste

Cela donne lieu à des algorithmes qui suivent ce schéma :

- traiter le cas de la liste vide (éventuellement à un élément aussi)
- faire un calcul qui utilise le résultat d'un appel récursif sur la liste moins son premier élément

Liste et récursivité

Une liste a une définition récursive.

Une liste est :

- vide
- un élément de tête et un suite qui est une liste

Cela donne lieu à des algorithmes qui suivent ce schéma :

- traiter le cas de la liste vide (éventuellement à un élément aussi)
- faire un calcul qui utilise le résultat d'un appel récursif sur la liste moins son premier élément

Liste non triée : Recherche

Algorithme 1 Recherche dans une liste non triée

```
Recherche(L : ↑ Elément, x : entier) : booléen
▷ Entrées : L (tête de la liste), x (élément recherché)
▷ Sortie : vrai si l'élément x a été trouvé dans la liste L, faux sinon.
 Début
     si (estListeVide(L))
         retourner  faux ;
     sinon  si (hd(l) = x)
             retourner  vrai ;
         sinon
             retourner Recherche(tl(L),x) ;
         fin si
     fin si
 Fin
```

Complexité ? Pouvez-vous écrire le même algo pour l'insertion dans une liste triée ?

Un problème entier

En cryptographie, on a besoin de calculer le pgcd de deux nombres.

Definition

Le pgcd de deux nombre a et b est l'entier c le plus grand tel que c divise a et c divise b.

Deux remarques utiles :

- pgcd(a,0) = a
- Pour $b \neq 0$, pgcd(a,b) = pgcd(b, a mod b)

Un problème entier

En cryptographie, on a besoin de calculer le pgcd de deux nombres.

Definition

Le pgcd de deux nombre a et b est l'entier c le plus grand tel que c divise a et c divise b.

Deux remarques utiles :

- pgcd(a,0) = a
- Pour $b \neq 0$, pgcd(a,b) = pgcd(b, a mod b)

Euclide faisait de l'algorithmique

On va faire un algorithme récursif à partir des deux idées précédentes. Au tableau, donnez-moi vos idées.

Le faire tourner sur 4122 et 1332.

Euclide faisait de l'algorithmique

On va faire un algorithme récursif à partir des deux idées précédentes. Au tableau, donnez-moi vos idées.

Le faire tourner sur 4122 et 1332.

Quelle est la complexité de l'algorithme ?

Euclide faisait de l'algorithmique

On va faire un algorithme récursif à partir des deux idées précédentes. Au tableau, donnez-moi vos idées.

Le faire tourner sur 4122 et 1332.

Quelle est la complexité de l'algorithme ?

On peut regarder deux étapes de suite et regarder de combien diminuent la somme de a et b. Les étapes : (a, b) puis (b, c) avec $c = a \mod b$ et (c, d) avec $d = b \mod c$.

Euclide faisait de l'algorithmique

On va faire un algorithme récursif à partir des deux idées précédentes. Au tableau, donnez-moi vos idées.

Le faire tourner sur 4122 et 1332.

Quelle est la complexité de l'algorithme ?

On peut regarder deux étapes de suite et regarder de combien diminuent la somme de a et b. Les étapes : (a, b) puis (b, c) avec $c = a \mod b$ et (c, d) avec $d = b \mod c$.
Par définition $b \geq d + c$. Comme $a > b > d + c$. Quand on fait la somme $(a + b) > 2(d + c)$.

Euclide faisait de l'algorithmique

On va faire un algorithme récursif à partir des deux idées précédentes. Au tableau, donnez-moi vos idées.

Le faire tourner sur 4122 et 1332.

Quelle est la complexité de l'algorithme ?

On peut regarder deux étapes de suite et regarder de combien diminuent la somme de a et b. Les étapes : (a, b) puis (b, c) avec $c = a \mod b$ et (c, d) avec $d = b \mod c$.
Par définition $b \geq d + c$. Comme $a > b > d + c$. Quand on fait la somme $(a + b) > 2(d + c)$.

Au final la complexité est en $O(\log_2(a + b))$.

Euclide faisait de l'algorithmique

On va faire un algorithme récursif à partir des deux idées précédentes. Au tableau, donnez-moi vos idées.

Le faire tourner sur 4122 et 1332.

Quelle est la complexité de l'algorithme ?

On peut regarder deux étapes de suite et regarder de combien diminuent la somme de a et b. Les étapes : (a, b) puis (b, c) avec $c = a \mod b$ et (c, d) avec $d = b \mod c$.
Par définition $b \geq d + c$. Comme $a > b > d + c$. Quand on fait la somme $(a + b) > 2(d + c)$.

Au final la complexité est en $O(\log_2(a + b))$.