



8 - Traitement séquentiel

Schémas de parcours d'une séquence

Schémas de recherche dans une séquence

Jean-Michel Adam - UGA - UFR SHS

Rappel: notion de séquence

- Une séquence est une suite d'éléments de même type

○ ○ ○ ○ ... ○ ○ ○

- Ces séquences peuvent être représentées de diverses manières sur le plan informatique.
- Un algorithme itératif consiste à énumérer les éléments d'une séquence.

Enumération des éléments d'une séquence (rappel)

- Il y a en fait deux grandes raisons pour énumérer les éléments d'une séquence :
 1. Appliquer **un même traitement à tous les éléments** de la séquence = **parcours séquentiel**
 2. Rechercher dans la séquence un élément vérifiant une propriété donnée = **recherche séquentielle**
- Tous les algorithmes itératifs consistent
 - soit en un parcours d'une séquence
 - soit en une recherche dans une séquence
 - soit en une combinaison de parcours et de recherches



Séquence représentée sur un support externe

Le fichier séquentiel

Consultation du polycopié décrivant les fichiers séquentiels

Caractérisation d'une séquence

- Pour effectuer un parcours ou une recherche, il faut énumérer les éléments de la séquence
- 4 choses caractérisent la séquence à énumérer
 1. La représentation de **l'élément courant**
 2. La manière dont on va repérer **la fin de la séquence**
 3. La manière dont on va se positionner sur le **premier élément**
 4. La manière de laquelle on va **passer de l'élément courant au suivant**

Énumération des éléments d'une séquence : notations

1. **élémentCourant** (EC) représente l'élément courant de la séquence
2. **findeSéquence** (FDS) représente l'expression logique qui caractérise la fin de la séquence
3. **démarrer** représente la partie algorithmique consistant à se positionner sur le premier élément de la séquence
4. **avancer** représente la partie algorithmique consistant à passer de l'élément courant au suivant

Exemple : caractérisation de la séquence des caractères d'une chaîne x

Il faut un indice de parcours de x
soit i cet indice

élément Courant : $\text{nième}(x,i)$

fin de Séquence : $i = \text{longueur}(x)$

avancer : $i \leftarrow i + 1$

démarrer : $i \leftarrow 0$

1-Traitement de la séquence

- Le **parcours** d'une séquence consiste à énumérer les éléments de la séquence et à **appliquer un même traitement** à chacun d'eux
- Le traitement de la séquence se compose de 4 parties algorithmiques distinctes
 1. Le **traitement de l'élément courant**
 2. L'**initialisation** du traitement : instructions d'initialisation du processus de traitement
 3. La **terminaison** du traitement : instructions à effectuer une fois la séquence parcourue
 4. Le **traitement** à effectuer dans le cas où la **séquence est vide** (s'il y a lieu)

Parcours avec traitement intégré de la séquence vide

démarrer

initialisationDuTraitement

tantque non **findeSéquence** faire

traitement **élémentCourant**

avancer

ftantque

terminaisonDuTraitement

Bleu : énumération des éléments de la séquence

Rouge : traitement de la séquence

Si la séquence est vide il n'y a pas de traitement spécifique,

on effectue : *initialisationDuTraitement*

terminaisonDuTraitement

Exemple

fonction nba(x : chaîne) → entier ≥ 0

// nba(x) renvoie le nombre de 'a' présents dans la chaîne x

lexique de nba

x : chaîne // paramètre : séquence examinée

na : entier ≥ 0 // intermédiaire : nombre de 'a' rencontrés dans x

i : entier ≥ 0 // intermédiaire : indice de parcours de x

algorithme de nba

i ← 0 démarrer

na ← 0 initialisationTraitement

tantque i ≠ longueur(x) faire non findeSéquence

si nième(x,i) = 'a' ou nième(x,i) = 'A' elementCourant

alors

na ← na + 1 traitementElementCourant

fsi

i ← i + 1 Avancer

fintantque

renvoyer(na) terminaisonTraitement

Parcours avec traitement spécifique de la séquence vide

démarrer

selon **findeSéquence**

findeSéquence :

traitementSéquenceVide

non **findeSéquence** :

initialisationDuTraitement

répéter

traitementélémentCourant

avancer

jusqu'à **findeSéquence**

terminaisonDuTraitement

fselon

Bleu : énumération des éléments de la séquence

Rouge : traitement de la séquence

Exemple : calcul de la somme des éléments d'une séquence d'entiers représentée dans un fichier

lexique principal

f : fichier d'entiers // donnée : séquence examinée
e : écran // périphérique de sortie
s : entier // résultat inter.: somme des entiers énumérés

Algorithme principal

```
f.lirePremier           démarrer
selon f
  f.fdf : e.afficher("séquence vide")           traitementSéquenceVide
  non f.fdf :
    s ← 0                                       initialisationTraitement
    répéter
      s ← s + f.ec                               traitementElementCourant
      f.lireSuivant                             avancer
    jusqu'à f.fdf                               findeSéquence
    e.afficher("somme = ",s)                   terminaisonTraitement
fselon
```

2 -Recherche dans une séquence

- La **recherche** consiste à déterminer si la séquence comporte au moins un élément vérifiant une propriété **P**.
- Il y a 2 possibilités :
 - soit on trouve un élément qui vérifie P
 - soit aucun élément de la séquence ne vérifie P
- On a donc 2 causes d'arrêt de la recherche
 - soit on a atteint la fin de séquence
 - soit on a trouvé un élément qui vérifie P
- Il n'y a pas de traitement : on ne fait qu'énumérer les éléments de la séquence

Recherche d'un élément vérifiant une propriété P

démarrer

tantque non **findeSéquence** etpuis non P (EC) faire
avancer

ftantque

// FindeSequence oualors P(EC)

selon **findeSéquence**

non **findeSéquence**: *// élément trouvé*

findeSéquence: *// élément non trouvé*

fselon

Avec etpuis on impose un ordre dans l'évaluation des conditions

- On regarde d'abord si on n'a pas atteint la fin de séquence
 - Puis on regarde si la propriété est vérifiée pour l'élément courant
- Cet ordre d'évaluation est nécessaire**, car si on a atteint la fin de séquence, l'élément courant n'est pas déterminé et la condition P(EC) n'est donc pas évaluable.

Exemple de recherche

- On considère un texte formé de lettres et d'espaces représenté dans un fichier
- Ecrire un algorithme qui affiche un message indiquant si le fichier comporte au-moins un mot.

Recherche de l'existence d'un mot

lexique principal

f : fichier de caractères // donnée : texte examiné
e : écran // périphérique de sortie

Algorithme principal

f.lirePremier

tantque non f.fdf et puis f.ec = ' ' faire
f.lireSuivant

ftantque

// f.fdf oualors f.ec ≠ ' '

selon f

f.fdf : e.afficher("aucun mot !")

non f.fdf : // f.ec = 1^{er} caractère du 1^{er} mot
e.afficher("au-moins un mot !")

fselon

Recherche du premier élément vérifiant la propriété P, on sait qu'un tel élément existe

démarrer

tantque non P (ElementCourant)

avancer

ftantque

// ElementCourant est le premier élément de la séquence vérifiant P

Il y a de nombreuses situations de recherche pour lesquelles on est sûr que l'élément recherché existe. L'algorithme consiste alors à se positionner sur le premier élément vérifiant la propriété.

Exemple : se positionner sur le premier mot d'un texte non vide

lexique principal

f : fichier de caractères // donnée : texte examiné
e : écran // périphérique de sortie

algorithme principal

f.lirePremier

tantque f.ec = ' ' faire
 f.lireSuivant

ftantque

// f.ec = premier caractère du premier mot du texte

Calcul de la somme d'une séquence d'entiers représentée dans un fichier (le retour)

lexique principal

f : fichier d'entiers // donnée : séquence examinée
e : écran // périphérique de sortie
s : entier // intermédiaire : somme des entiers énumérés

Algorithme principal

f.lirePremier

selon f

f.fdf : e.afficher("séquence vide")

non f.fdf :

s ← 0

répéter

s ← s + f.ec

f.lireSuivant

jusqu'à f.fdf

e.afficher("somme = ",s)

fselon

Que faudrait-il modifier à cet algorithme si la séquence était mémorisée dans un tableau ?

Représentation de la séquence dans un tableau d'entiers

t

12	-3	71	34	12	...	11	56	8	-28	76		
0											lg	LMAX-1

t : tableau sur [0...LMAX-1] d'entiers

lg : entier ≥ 0 // longueur de la séquence

On suppose que le tableau contient déjà la séquence de lg entiers.

Il faut un indice de parcours de t :

k : entier entre 0 et LMAX // **indice de parcours de t**

Caractérisation de la séquence

Élément Courant : $t[k]$

Fin de Séquence : $k = \lg$

Avancer : $k \leftarrow k + 1$

Démarrer : $k \leftarrow 0$

Calcul de la somme d'une séquence d'entiers représentée dans un tableau

lexique principal

t : tableau sur [0..LMAX-1] d'entiers // séquence examinée
lg : réel ≥ 0 // longueur du texte examiné
e : écran // périphérique de sortie
s : entier // intermédiaire : somme des entiers énumérés
k : entier entre 0 et LMAX // indice de parcours de T

Algorithme principal

k \leftarrow 0

selon k,lg

k = lg : e.afficher("séquence vide")

k < lg :

s \leftarrow 0

répéter

s \leftarrow s + t[k]

k \leftarrow k + 1

jusqu'à k = lg

e.afficher("somme = ",s)

fselon

Exercice : nombre d'entiers pairs

Ecrire un algorithme qui compte le nombre d'entiers pairs présents dans une séquence d'entiers représentée dans un fichier

lexique principal

f : fichier d'entiers // donnée : séquence examinée

e : écran // périphérique de sortie

nbp : entier // intermédiaire : nombre d'entiers pairs de la pp de f

Algorithme principal

f.LirePremier

selon f

 f.fdf : e.afficher("séquence vide")

non f.fdf :

 nbp ← 0

répéter

 si f.ec mod 2 = 0 alors nbp ← nbp + 1 fsi

 f.LireSuivant

jusqu'à f.fdf

 e.afficher nombre d'entiers pairs : ",nbp)

fselon

