

Fondements informatiques I

Cours 9: Algorithmes : manipuler des tableaux

Sorina Ionica sorina.ionica@uvsq.fr

Sandrine Vial sandrine.vial@uvsq.fr

Qu'est-ce que c'est qu'un algorithme ?

Rappel (premier cours)

Un algorithme est une suite finie d'instructions et opérations permettant de résoudre un problème. Pour donner un algorithme :

- Input : les données en entrée du problème
- Output : la solution du problème
- Les étapes qu'on doit effectuer pour résoudre

Exemple concret : Dans un tableau de notes des étudiants de L1, on veut calculer la note la plus petite et la plus grande.

Ensuite on voudrait faire afficher le tableau (la liste) des notes à un examen, triée dans l'ordre croissante.

Pour quoi ?

La bonne pratique : avant de commencer à coder, on réfléchit sur papier !

En langage humain, on précise les étapes de notre algorithme (pseudocode).

On réfléchit également à la structure du programme :

Quelles variables ? De quel type ? Quelles fonctions ?

Pour chaque fonction on précise les paramètres en entrée et en sortie.

Minimum d'un tableau

On veut calculer le minimum d'un tableau et renvoyer son indice.

Algorithme

Input: un tableau L d'entiers

Output: l'indice du plus petit entier dans la liste

Au début on initialise $m=0$ (premier indice du tableau)

Pour tout i allant de 1 à longueur(L)-1

 Si $L[j] < L[m]$ alors

$m \leftarrow j$

On renvoie m

Minimum d'un tableau

En Python cela donne :

```
def minimum(L):
    m = 0
    for j in range(1, len(L)):
        if L[j] < L[m]:
            m = j
    return m
```

Et pour calculer le maximum ?

Tri par sélection

Algorithme

Input: le tableau L

Output: le tableau L trié

Pour tout i allant de 0 à longueur(L)-1:

 On calcule l'indice m du plus petit élément dans L[i:]

 On échange L[i] et L[m]

Tri par sélection

Algorithme

Input: le tableau L

Output: le tableau L trié

Pour tout i allant de 0 à longueur(L)-1:

 On calcule l'indice m du plus petit élément dans L[i:]

 On échange L[i] et L[m]

Exécution sur le tableau $L = [5, 1, -8, 2, -4, 7]$:

- $i = 0$ $\text{minimum}(L)=2$ alors $L = [-8, 1, 5, 2, -4, 7]$

Tri par sélection

Algorithme

Input: le tableau L

Output: le tableau L trié

Pour tout i allant de 0 à longueur(L)-1:

 On calcule l'indice m du plus petit élément dans L[i:]

 On échange L[i] et L[m]

Exécution sur le tableau $L = [5, 1, -8, 2, -4, 7]$:

- $i = 0$ $\text{minimum}(L)=2$ alors $L = [-8, 1, 5, 2, -4, 7]$
- $i = 1$ $\text{minimum}(L[1 :])=4$ alors $L = [-8, -4, 5, 2, 1, 7]$

Tri par sélection

Algorithme

Input: le tableau L

Output: le tableau L trié

Pour tout i allant de 0 à longueur(L)-1:

 On calcule l'indice m du plus petit élément dans L[i:]

 On échange L[i] et L[m]

Exécution sur le tableau $L = [5, 1, -8, 2, -4, 7]$:

- $i = 0$ $\text{minimum}(L)=2$ alors $L = [-8, 1, 5, 2, -4, 7]$
- $i = 1$ $\text{minimum}(L[1 :])=4$ alors $L = [-8, -4, 5, 2, 1, 7]$
- $i = 2$ $\text{minimum}(L[2 :])=2$ alors $L = [-8, -4, 1, 2, 5, 7]$

Tri par sélection

Algorithme

Input: le tableau L

Output: le tableau L trié

Pour tout i allant de 0 à longueur(L)-1:

 On calcule l'indice m du plus petit élément dans L[i:]

 On échange L[i] et L[m]

Exécution sur le tableau $L = [5, 1, -8, 2, -4, 7]$:

- $i = 0$ $\text{minimum}(L)=2$ alors $L = [-8, 1, 5, 2, -4, 7]$
- $i = 1$ $\text{minimum}(L[1 :])=4$ alors $L = [-8, -4, 5, 2, 1, 7]$
- $i = 2$ $\text{minimum}(L[2 :])=2$ alors $L = [-8, -4, 1, 2, 5, 7]$
- $i = 3$ $\text{minimum}(L[3 :])=3$ alors $L = [-8, -4, 1, 2, 5, 7]$

Tri par sélection

Algorithme

Input: le tableau L

Output: le tableau L trié

Pour tout i allant de 0 à longueur(L)-1:

 On calcule l'indice m du plus petit élément dans L[i:]

 On échange L[i] et L[m]

Exécution sur le tableau $L = [5, 1, -8, 2, -4, 7]$:

- $i = 0$ $\text{minimum}(L)=2$ alors $L = [-8, 1, 5, 2, -4, 7]$
- $i = 1$ $\text{minimum}(L[1 :])=4$ alors $L = [-8, -4, 5, 2, 1, 7]$
- $i = 2$ $\text{minimum}(L[2 :])=2$ alors $L = [-8, -4, 1, 2, 5, 7]$
- $i = 3$ $\text{minimum}(L[3 :])=3$ alors $L = [-8, -4, 1, 2, 5, 7]$
- $i = 4$ $\text{minimum}(L[4 :])=4$ alors $L = [-8, -4, 1, 2, 5, 7]$

Tri par sélection en Python

```
def tri_selection(L):
    for i in range(len(L)):
        m = minimum(L[i:])
        L[i], L[m+i] = L[m+i], L[i] # échanger L[i] et L[m]
    return L

L = [5, 1, -8, 2, -4, 7]
print(tri_selection(L))
```

Comment créer un tableau trié directement

Si le tableau est créé au fur et à mesure, on insère les éléments dans le bon ordre.

Algorithme

Input : un tableau Tab trié, un élément t

Output: le tableau trié Tab modifiée après rajout de t

On initialise i<-0.

taille<-longueur(Tab)

Tant que i<taille and t>l[i]

 On avance dans la liste i=i+1

On décale les éléments Tab[i], ..., Tab[taille-1] d'un cran

On place l'élément t à l'indice i dans tab (Tab[i]=t)

On renvoie Tab

Comment créer un tableau trié directement

Algorithme

On initialise $i=0$.

$\text{taille} \leftarrow \text{longueur}(\text{Tab})$

Tant que $i < \text{taille}$ and $t > \text{Tab}[i]$

 On avance dans la liste $i = i + 1$

On décale les éléments $\text{Tab}[i], \dots, \text{Tab}[\text{taille}-1]$ d'un cran

On place l'élément t à l'indice i dans Tab ($\text{Tab}[i] = t$)

On renvoie Tab

Exécution pour construire un tableau trié à partie de la séquence 5, 1, -8, 2, -4, 7 :

Comment créer un tableau trié directement

Algorithme

On initialise $i=0$.

$\text{taille} \leftarrow \text{longueur}(\text{Tab})$

Tant que $i < \text{taille}$ and $t > \text{Tab}[i]$

 On avance dans la liste $i = i + 1$

On décale les éléments $\text{Tab}[i], \dots, \text{Tab}[\text{taille}-1]$ d'un cran

On place l'élément t à l'indice i dans Tab ($\text{Tab}[i] = t$)

On renvoie Tab

Exécution pour construire un tableau trié à partie de la séquence 5, 1, -8, 2, -4, 7 :

- $i = 0$ alors $\text{Tab} = [5]$

Comment créer un tableau trié directement

Algorithme

On initialise $i=0$.

$\text{taille} \leftarrow \text{longueur}(\text{Tab})$

Tant que $i < \text{taille}$ and $t > \text{Tab}[i]$

 On avance dans la liste $i = i + 1$

On décale les éléments $\text{Tab}[i], \dots, \text{Tab}[\text{taille}-1]$ d'un cran

On place l'élément t à l'indice i dans Tab ($\text{Tab}[i] = t$)

On renvoie Tab

Exécution pour construire un tableau trié à partie de la séquence 5, 1, -8, 2, -4, 7 :

- $i = 0$ alors $\text{Tab} = [5]$
- $i = 1$ alors $\text{Tab} = [1, 5]$

Comment créer un tableau trié directement

Algorithme

On initialise $i=0$.

$\text{taille} \leftarrow \text{longueur}(\text{Tab})$

Tant que $i < \text{taille}$ and $t > \text{Tab}[i]$

 On avance dans la liste $i = i + 1$

On décale les éléments $\text{Tab}[i], \dots, \text{Tab}[\text{taille}-1]$ d'un cran

On place l'élément t à l'indice i dans Tab ($\text{Tab}[i] = t$)

On renvoie Tab

Exécution pour construire un tableau trié à partie de la séquence 5, 1, -8, 2, -4, 7 :

- $i = 0$ alors $\text{Tab} = [5]$
- $i = 1$ alors $\text{Tab} = [1, 5]$
- $i = 2$ alors $\text{Tab} = [-8, 1, 5]$

Comment créer un tableau trié directement

Algorithme

On initialise $i=0$.

$\text{taille} \leftarrow \text{longueur}(\text{Tab})$

Tant que $i < \text{taille}$ and $t > \text{Tab}[i]$

 On avance dans la liste $i = i + 1$

On décale les éléments $\text{Tab}[i], \dots, \text{Tab}[\text{taille}-1]$ d'un cran

On place l'élément t à l'indice i dans Tab ($\text{Tab}[i] = t$)

On renvoie Tab

Exécution pour construire un tableau trié à partie de la séquence 5, 1, -8, 2, -4, 7 :

- $i = 0$ alors $\text{Tab} = [5]$
- $i = 1$ alors $\text{Tab} = [1, 5]$
- $i = 2$ alors $\text{Tab} = [-8, 1, 5]$
- $i = 3$ alors $\text{Tab} = [-8, 1, 2, 5]$

Comment créer un tableau trié directement

Algorithme

On initialise $i=0$.

$\text{taille} \leftarrow \text{longueur}(\text{Tab})$

Tant que $i < \text{taille}$ and $t > \text{Tab}[i]$

 On avance dans la liste $i = i + 1$

On décale les éléments $\text{Tab}[i], \dots, \text{Tab}[\text{taille}-1]$ d'un cran

On place l'élément t à l'indice i dans Tab ($\text{Tab}[i] = t$)

On renvoie Tab

Exécution pour construire un tableau trié à partie de la séquence 5, 1, -8, 2, -4, 7 :

- $i = 0$ alors $\text{Tab} = [5]$
- $i = 1$ alors $\text{Tab} = [1, 5]$
- $i = 2$ alors $\text{Tab} = [-8, 1, 5]$
- $i = 3$ alors $\text{Tab} = [-8, 1, 2, 5]$
- $i = 4$ alors $\text{Tab} = [-8, -4, 1, 2, 5]$

Comment créer un tableau trié directement

Algorithme

On initialise $i=0$.

$\text{taille} \leftarrow \text{longueur}(\text{Tab})$

Tant que $i < \text{taille}$ and $t > \text{Tab}[i]$

 On avance dans la liste $i = i + 1$

On décale les éléments $\text{Tab}[i], \dots, \text{Tab}[\text{taille}-1]$ d'un cran

On place l'élément t à l'indice i dans Tab ($\text{Tab}[i] = t$)

On renvoie Tab

Exécution pour construire un tableau trié à partie de la séquence 5, 1, -8, 2, -4, 7 :

- $i = 0$ alors $\text{Tab} = [5]$
- $i = 1$ alors $\text{Tab} = [1, 5]$
- $i = 2$ alors $\text{Tab} = [-8, 1, 5]$
- $i = 3$ alors $\text{Tab} = [-8, 1, 2, 5]$
- $i = 4$ alors $\text{Tab} = [-8, -4, 1, 2, 5]$
- $i = 5$ alors $\text{Tab} = [-8, -4, 1, 2, 5, 7]$

Insertion dans une liste triée en Python

```
def insertion(t,l):
    i=0
    taille=len(l)
    while (i<taille and t>l[i]):
        i+=1
    if (i<taille):
        l[i+1:]=l[i:]
        l[i]=t
    else :
        l.append(t)
```

```
insertion(5,l)
insertion(10,l)
print(l)
```

Suite de Fibonacci

La suite de Fibonacci est définie par la relation suivante :

$$F_0 = 0, \quad F_1 = 1, \quad \text{et pour tout } n \geq 2, \quad F_n = F_{n-1} + F_{n-2}.$$

On veut calculer toutes les éléments de la suite de Fibonacci, jusqu'au n -ième.

Algorithme

Input: n , valeurs initiales $f_0=0$, $f_1=1$

Output : Le tableau Fib contenant les nombres de Fibonacci

Fib=[0,1]

i=2

Tant que $(i \leq n)$ faire:

 On rajoute $\text{Fib}[i]=\text{Fib}[i-2]+\text{Fib}[i-1]$ dans le tableau

Renvoyer Fib

Suite de Fibonacci

On veut calculer le n -ième élément de la suite de Fibonacci.

On remarque qu'à chaque étape on a besoin de connaître que les deux valeurs précédentes.

Algorithme

Input: n , valeurs initiales $a=0$, $b=1$

Output : n -ième valeur dans la suite

compteur=0

Tant que (compteur < n) faire:

$a \leftarrow b$

$b \leftarrow a+b$

Renvoyer a

Suite de Fibonacci

En Python

```
def fibonacci(n):
    if n < 0:
        print("n doit être >= 0")
    a, b = 0, 1
    for _ in range(n):
        a, b = b, a + b
    return a
```

À retenir : Un programme correct n'est pas juste un programme qui fonctionne mais aussi :

- qui utilise le moins de mémoire possible
- qui résout le problème avec le moins d'opérations possibles !