

But de ces séances

PRÉREQUIS : avoir **vu la vidéo**

Buts :

- ▶ Améliorer/renforcer votre apprentissage
 - ▶ Répondre à vos questions
 - ▶ Approfondir des sujets (à votre demande)
- ☞ Vous faire **gagner du temps** de révision / de mise en pratique sur les exercices

Déroulement

- ▶ Qu'avez-vous retenu ? / Points importants
(rapide car pas le but de redire le cours)
- ▶ Questions / approfondissements / points difficiles
- ▶ Propositions d'« études de cas »
- ▶ Pratique libre (exercices) + réponse (libre) aux questions
(= commencer ensemble la série d'exercices)

Les conseils de Carla (PH-BA3)

- ▶ Bien **organiser son temps** :
 - ▶ s'organiser des créneaux fixes dans la semaine pour la classe inversée
 - ▶ regarder les vidéos suffisamment tôt dans la semaine en prenant des notes
 - ▶ prévoir un créneau pour la programmation et **se fixer une durée de travail**
- ▶ Utiliser la méthode de la page blanche pour vérifier ce dont on se souvient réellement
- ▶ Choisir les exercices **adaptés à son niveau pour progresser** et ne pas se laisser submerger par toutes les ressources
- ▶ Ne pas hésiter à **demandeur aux assistants** des indices, du feedback, des explications ou des méthodes de travail
- ▶ Utiliser les séances de cours pour approfondir les notions et **poser des questions**

Leçon I.1b (algorithmes, complexité) – Points clés

2 grandes parties :

- ① comment écrire un algorithme
- ② complexité d'un algorithme

Leçon I.1b (algorithmes, complexité) – Points clés ①

① comment écrire un algorithme :

► « méthodologie » :

1. **comprendre** le problème : question ? entrée(s) ? sortie(s) ?
(écriture de l'algorithme : ne pas oublier de bien spécifier l'/les entrée(s) et la/les sortie(s))
2. trouver *un* algorithme **correct** (vérifier qu'il est bien correct ; penser à vérifier les cas limites)
3. trouver un algorithme *efficace*
 ☞ savoir calculer la complexité d'un algorithme

► « boîte à outils » :

- affectation ($x \leftarrow 18$) et autres opérations usuelles
- 3 structures de contrôle : branchements, boucles, itérations *repetier*
- mot-clé « **Sortir** : » pour terminer et indiquer quelle est la sortie de l'algorithme
- *appel* d'autres (sous-)algorithmes *taille()*

► trois problèmes « clés » : recherche, tri, plus court chemin

Leçon I.1b (algorithmes, complexité) – Points clés ②

② complexité d'un algorithme :

- ▶ définition :
 - ▶ est une *fonction* de la taille de l'entrée
 - ▶ compte le nombre d'opérations **élémentaires** nécessaires
 - ▶ dans le pire des cas  l'identifier !
 - ▶ on s'intéresse à son comportement à l'infini (c'est-à-dire quand la taille de l'entrée augmente) :
on utilise donc une notation d'ensemble : $\Theta(f(n))$:
ensemble des fonctions qui « croissent à l'infini comme $f(n)$ »
- ▶ meilleure complexité des algorithmes résolvant les trois problèmes « clés » :
 - ▶ recherche : liste ordonnée : $\Theta(\log n)$; liste quelconque : $\Theta(n)$
 - ▶ tri : $\Theta(n \log n)$
 - ▶ plus court chemin : ? (semaine prochaine)

Leçon I.1b (algorithmes, complexité) – Question(s)

QUESTIONS?

Rappel semaine passée :

1. Quelle est la meilleure complexité ?

A. $\Theta(\log(n))$

B. $\Theta(n)$

C. $\Theta(n \cdot \log(n))$

D. $\Theta(n^2)$

La *meilleure* complexité est la plus petite :
on préfère un algorithme qui prend *moins* de temps

Question fréquente : *dans quelle base est le log ?*

$$\log_b n = \frac{\log_a n}{\log_a b} \text{ de}$$

Leçon I.1b (algorithmes, complexité) – Ecrire un algorithme

Ecrire (formellement cette fois) un algorithme pour :

- trouver tous les éléments maximaux dans une liste

elements_max

entrée : une liste L

sortie : une liste L' de positions

$$L = (7, 8, 4, -1, 8, 3)$$

1 2 3 4 5 6

$\rightarrow L' = (2, 5)$

Trouver tous les éléments maximaux dans une liste

Éléments maximaux

entrée : L une liste (non vide ??) de nombres

sortie : (liste des positions des) éléments maximaux de la liste

$n \leftarrow \text{taille}(L)$

$L \leftarrow \text{tri}(L)$

$L' \leftarrow (n)$ // liste à 1 seul élément

Tant que $n > 1$ et $L(n-1) = L(n)$

$n \leftarrow n - 1$

$L' \leftarrow L' \oplus (n)$ // ajout de la valeur n à la liste

Sortir : L'

^{1 2 3 4 5 6}
(-1, 3, 4, 7, 8, 8)

↑ ↑
 $n-1$ n

↓

(6, 5)

Trouver tous les éléments maximaux dans une liste

Éléments maximaux

entrée : L une liste (non vide ??) de nombres

sortie : (liste des positions des) éléments maximaux de la liste

$n \leftarrow \text{taille}(L)$ $\rightarrow \Theta(n)$

$L \leftarrow \text{tri}(L)$ $\rightarrow P(n)$

$L' \leftarrow (n)$ // liste à 1 seul élément $\rightarrow 1$

Tant que $n > 1$ **et** $L(n-1) = L(n)$ $\rightarrow 2$

$n \leftarrow n-1$ $\rightarrow 1$

$L' \leftarrow L' \oplus (n)$ // ajout de la valeur n à la liste $\rightarrow 1$

Sortir : L'

$\in \Theta(n \lg n)$

propositions:

Complexité ?

$\Theta(n \lg n)$

~~$\Theta(n)$~~

$\Theta(n + P(n))$ $P(n)$ complexité de tri

Peut-on faire mieux ?

$\Theta(n^2 + n \lg n)$

$\in \Theta(n^2)$

- 1) pire cas: tous égaux
- 2) on compte

$$\Theta(n) + P(n) + 2n + 1$$

$$= \Theta(n) + \Theta(n \lg n) + 2n + 1$$

$$\in \Theta(n \lg n)$$

Trouver tous les éléments maximaux dans une liste

Éléments maximaux

entrée : L une liste (non vide ??) de nombres

sortie : (liste des positions des) éléments maximaux de la liste

$n \leftarrow \text{taille}(L)$

$L \leftarrow \text{tri}(L)$

$L' \leftarrow (n)$ // liste à 1 seul élément

Tant que $n > 1$ **et** $L(n-1) = L(n)$

$n \leftarrow n - 1$

$L' \leftarrow L' \oplus (n)$ // ajout de la valeur n à la liste

Sortir : L'

Complexité ?

☞ $\Theta(n \log n)$, mais...
...il est FAUX !!

Peut-on faire mieux ?

☞ **oui :**
non seulement *juste*,
mais aussi
à moindre coût

Trouver tous les éléments maximaux dans une liste

Éléments maximaux v2

entrée : L une liste (non vide??) de nombres

sortie : (liste des positions des) éléments maximaux de la liste

$n \leftarrow \text{taille}(L)$

$L' \leftarrow ()$

// liste vide



$x_{\max} \leftarrow -\infty$

Pour i allant de 1 à n

Si $L(i) > x_{\max}$

$x_{\max} \leftarrow L(i)$

$L' \leftarrow (i)$

// liste à 1 seul élément

Sinon, si $L(i) = x_{\max}$

$L' \leftarrow L' \oplus (i)$

// ajout de la valeur i à la liste

Sortir : L'

(7, 8, 4, -1, 8, 3)
1 2 3 4 5 6

n	L'	$x_{\max}$	i
6	()	$-\infty$	
			1 2 ... 5
	(1) (2)	7 8	
	(2, 5)		

Trouver tous les éléments maximaux dans une liste

Éléments maximaux v2

entrée : L une liste (non vide ??) de nombres

sortie : (liste des positions des) éléments maximaux de la liste

```
 $n \leftarrow \text{taille}(L)$   $\rightarrow \Theta(n)$   
 $L' \leftarrow ()$  // liste vide  $\rightarrow 1$   
 $x_{\max} \leftarrow -\infty$   $[1]$   $\rightarrow 1$   
Pour  $i$  allant de 1 à  $n$   $\rightarrow 3$   
  Si  $L(i) > x_{\max}$   $\rightarrow 1$   
     $x_{\max} \leftarrow L(i)$   
     $L' \leftarrow (i)$  // liste à 1 seul élément  $\rightarrow 1$   
  Sinon, si  $L(i) = x_{\max}$   $\rightarrow 1$   
     $L' \leftarrow L' \oplus (i)$  // ajout de la valeur  $i$  à la liste  $\rightarrow 1$   
Sortir :  $L'$ 
```

Complexité ? $\Theta(n)$

$\rightarrow$ pire cas

Peut-on faire mieux ?

$\rightarrow$ pire cas :

n fois : $n \times \Theta(n)$
 $\in \Theta(n)$

$\rightarrow \Theta(n) + 2 + \Theta(n) \in \Theta(n)$

Trouver tous les éléments maximaux dans une liste

Éléments maximaux v2

entrée : L une liste (non vide ??) de nombres

sortie : (liste des positions des) éléments maximaux de la liste

$n \leftarrow \text{taille}(L)$

$L' \leftarrow ()$ // liste vide

$x_{\max} \leftarrow -\infty$

Pour i allant de 1 à n

Si $L(i) > x_{\max}$

$x_{\max} \leftarrow L(i)$

$L' \leftarrow (i)$ // liste à 1 seul élément

Sinon, si $L(i) = x_{\max}$

$L' \leftarrow L' \oplus (i)$ // ajout de la valeur i à la liste

Sortir : L'

Complexité ?

☞ $\Theta(n)$

Peut-on faire mieux ?

☞ **non**

Questions fréquentes

- Comment ajouter un élément à une liste ?

$L \leftarrow L \oplus (x)$ // ajout en fin

$L \leftarrow (x) \oplus L$ // ajout au début

$$L \leftarrow L \oplus L'$$

- Comment ajouter un élément x à une position i dans une liste ?

$L \leftarrow (L[1], \dots, L[i-1], x, L[i], \dots, L[n])$

- Comment vider une liste ?

$L \leftarrow ()$ // affectation de la liste vide

- Comment supprimer un élément d'une liste ?

$L \leftarrow (L[1], \dots, L[i-1], L[i+1], \dots, L[n])$

(où la taille de la liste aura préalablement été affecté à n)

Leçon I.1b (algorithmes, complexité) – Recherche

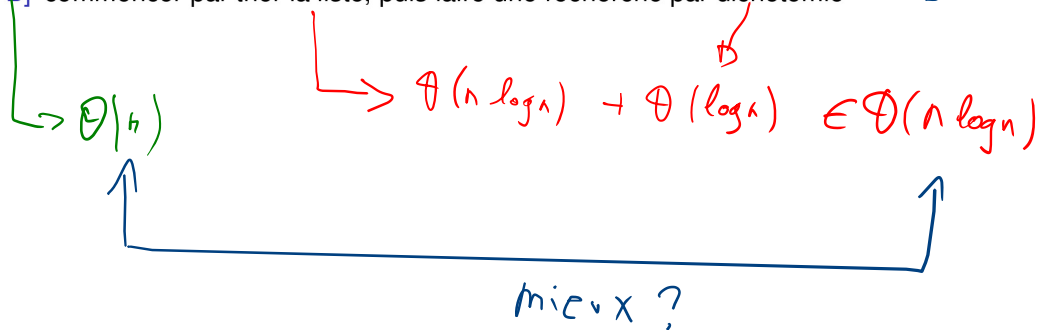
Recherche dans une liste : qu'est-ce qui est le mieux ?

$$n = \text{taille}(L)$$

A] faire une recherche linéaire

$$\approx 50$$

B] commencer par trier la liste, puis faire une recherche par dichotomie $\approx 15?$



Leçon I.1b (algorithmes, complexité) – Recherche

Recherche dans une liste : qu'est-ce qui est le mieux ?

A] faire une recherche linéaire

B] commencer par trier la liste, puis faire une recherche par dichotomie

Cela dépend de ce que l'on veut :

soit K le nombre de fois que l'on fait la recherche,
et soit n taille de la liste ;

A : $K \cdot f(n)$, avec $f \in \Theta(n)$

B : $g(n) + K \cdot h(n)$, avec $g \in \Theta(n \log n)$ et $h \in \Theta(\log n)$