Université Claude Bernard Lyon 1

LIF5 - Algorithmique et programmation procédurale

Carole Knibbe

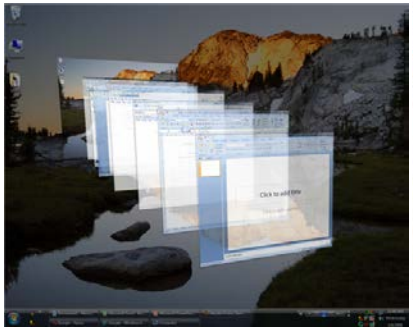
Samir Akkouche

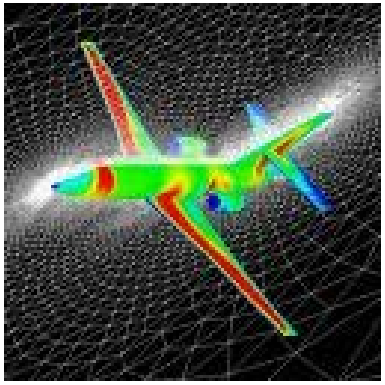
LIRIS

Automne 2013

Présentation de l’UE

La place de l'UE dans les domaines de l'informatique

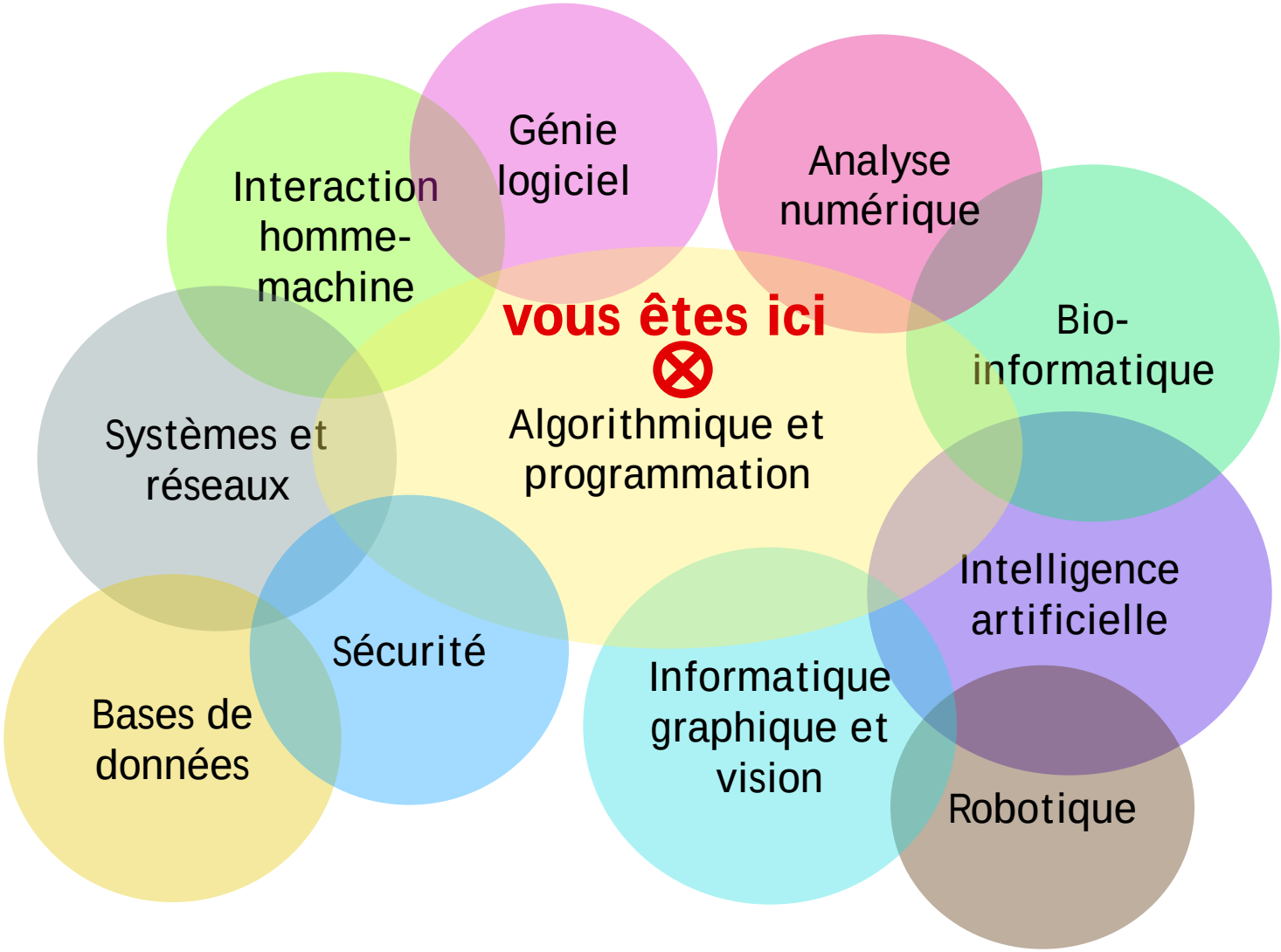




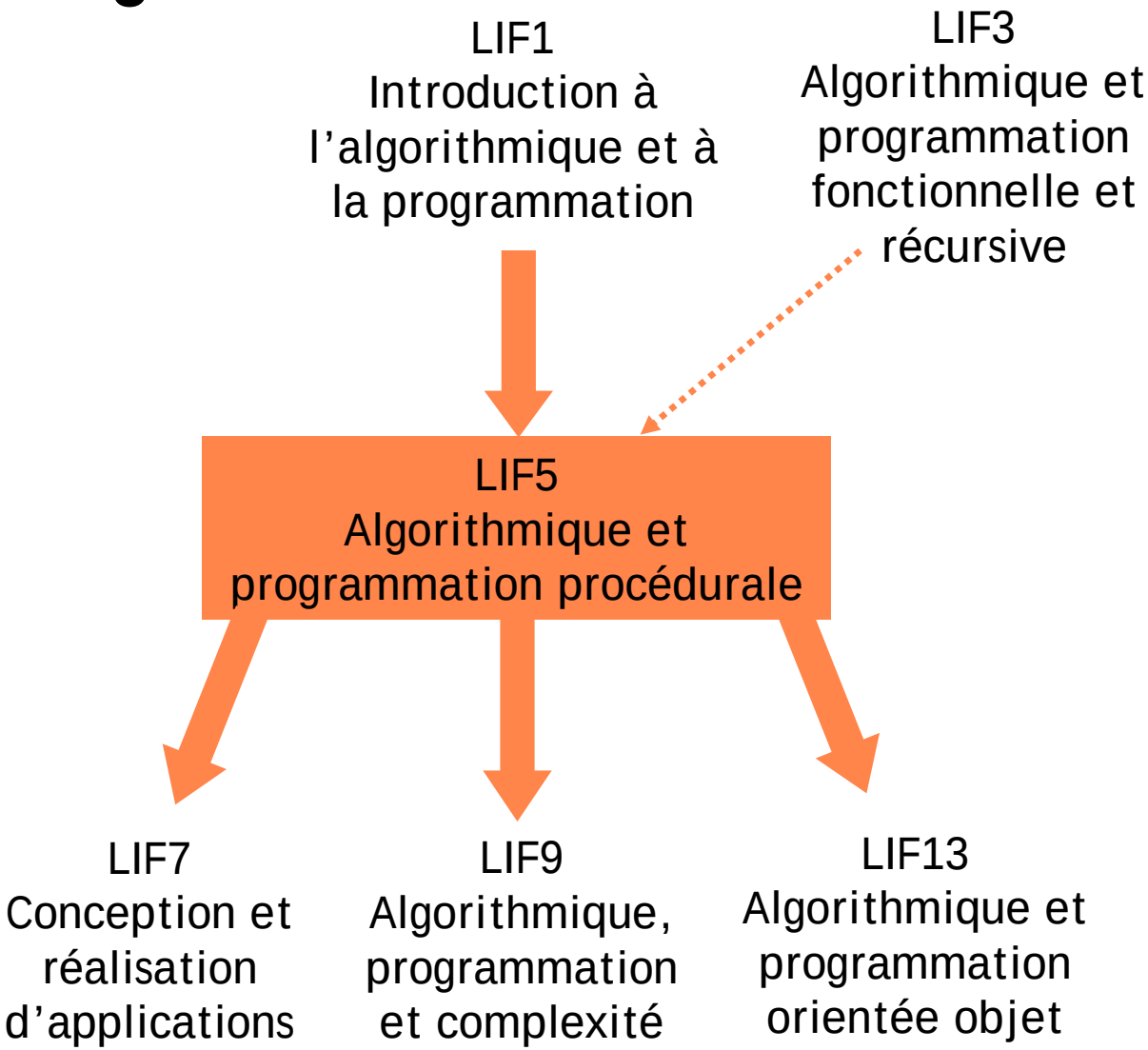
vous êtes ici

Google™

La place de l'UE dans les domaines de l'informatique



La place de l'UE dans votre parcours d'algo-prog.



Détail des pré-requis

- Savoir écrire un algorithme simple en langage algorithmique
  - manipuler des variables de type booléen, entier, réel, caractère
  - manipuler des tableaux et chaînes de caractères
  - connaître les structures de contrôle (tests, boucles, …)
  - savoir découper un programme en fonctions et procédures
  - connaître les modes de passage des paramètres

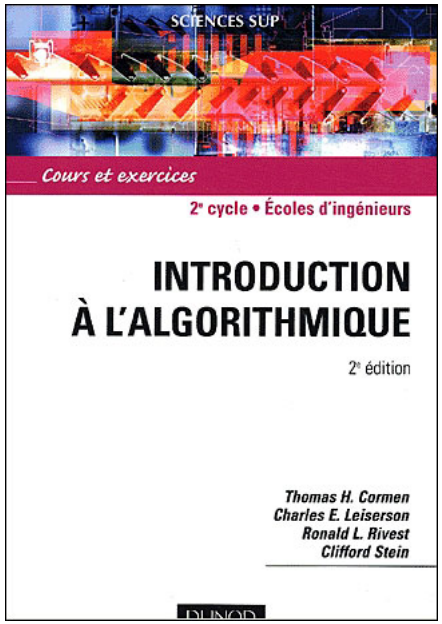
- Savoir l'implanter en langage C

Contenu de l'UE

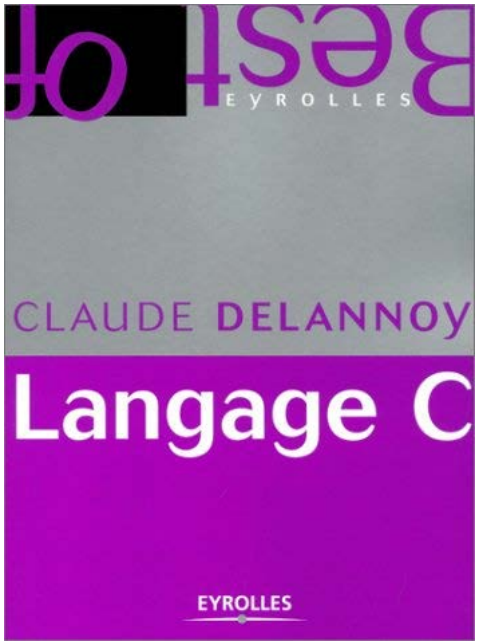
- Principales structures de données utilisées en informatique
  - types primitifs : entiers, réels, caractères, booléens, pointeurs
  - types agrégés : tableaux et structures
  - structures dynamiques : piles, files, listes, arbres, fichiers…
- Algorithmes s'appuyant sur ces structures
  - algorithmes de tri
  - introduction à la notion de complexité algorithmique
- Mise en œuvre en C
  - gestion des entrées-sorties
  - utilisation de fichiers
  - gestion dynamique de la mémoire
  - type de données abstrait et programmation modulaire

Ressources disponibles à la BU

- T. H. Cormen et al., *Introduction à l'algorithmique*, éditions Dunod (parties 1 à 3)



- C. Delannoy, *Langage C*, éditions Eyrolles



Modalités d'évaluation

Contrôle continu intégral (pas de seconde session) :

- 3 mini évaluations en TP — 3x10 % = 30%
- CC mi-parcours, en amphi, anonyme — 35 %
  - le lundi 4 novembre 2013 à 10h 45
- CC final, en amphi, anonyme — 35 %
  - semaine du 7 janvier 2014, date fixée par le BAL

Organisation des enseignements de LIF5

- 12 séances de CM de 1h30 chacune, mardi de 17h30 à 19h
- 9 séances de TD de 2h chacune, lundi ou Mardi
  - premier TD le 16 septembre 2013
- 2 séances de « TD spécifiques » de 1h30 : révisions, annales…
  - une avant le CC mi-parcours
  - une avant le CC final
- 8 séances de TP de 3h chacune
  - lundi 10h-13h ou mardi 14h-17h selon les groupes
  - premier TP la semaine du 23 septembre
- 4 séances de soutien, « sur invitation » : cf Tomuss
  - le mardi 15 octobre, 17h30- 18h30
  - le mardi 16 novembre, 17h30- 18h30
  - le mardi 10 décembre, 17h30- 18h30
  - le mardi 17 décembre, 17h30- 18h30

Informations pratiques

http://samir.akkouche.perso.univ-lyon1.fr/LIF5/

- Planning des séances de CM, TD, TP, soutien
- Ressources pour certains TP
- Diapositives des cours (+ diapos LIF1)

Dispenses d'assiduité

- Principe général :
  - éligibles = étudiants salariés, sportifs de haut niveau, double cursus, etc.
  - dispense des CM et des TD
  - la présence en TP reste obligatoire
  - l'accord du responsable d'UE est nécessaire
  - démarche administrative complète sur www.univ-lyon1.fr, rubrique Scolarité / Scolarité en ligne

- En LIF5 :
  - venez en discuter à la fin du CM (pas de mail): engagement de présence au CC mi-parcours et au CC final
  - la dispense sera refusée sinon

Plagiat

- Différence plagiat / collaboration ?
- En LIF7, travail collaboratif
- En LIF5, évaluation de votre capacité à travailler individuellement, à développer vos propres idées
  => 0 si plagiat
- Ce qui est important, c'est que vous passiez par le processus d'essais d'écriture de programmes
- Faites de votre mieux, rendez ce que vous pouvez, mais n'abandonnez pas en copiant !
- Vidéo…

Chapitre 1
Introduction générale

Les algorithmes et les structures de données comme technologies

Qu'est-ce qu'un programme ?

- Déclarations et définitions de données
  - Structurées en "structures de données"
  - Décrivent, spécifient les données à manipuler

```
tab : tableau [1..15] d'entiers
i : entier
moy : réel
```

```
int tab[15];
int i;
double moy;
```

- Instructions
  - Structurées par l'algorithme
  - Décrivent, spécifient les actions à effectuer sur les données

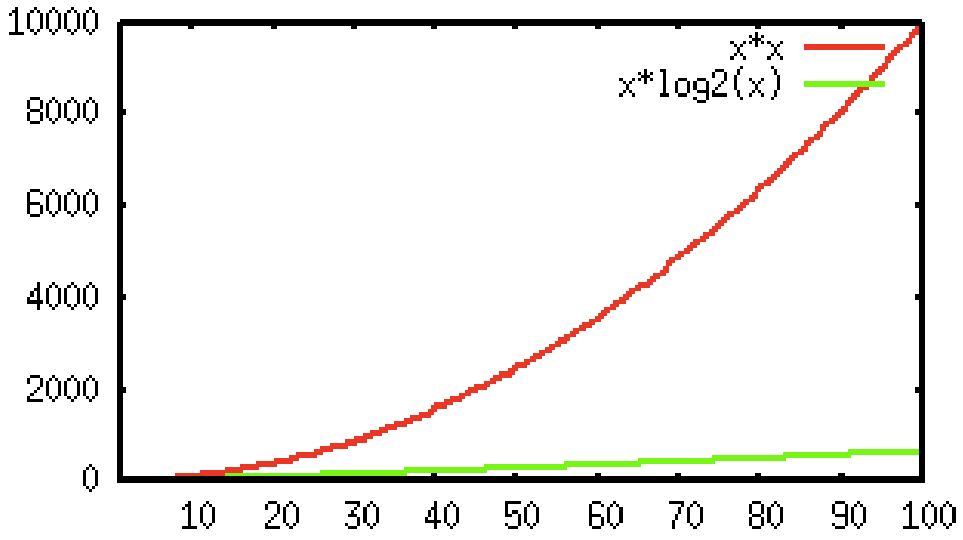
```
moy ← 0.0
Pour i allant de 1 à 15 par pas de 1
Faire
        moy ← moy + tab[i]
Fin Pour
moy ← moy / 15.0
```

```
moy = 0.0;
for (i=0; i<15; i++)
{
        moy = moy + tab[i];
}
moy = moy / 15.0;
```

Les algorithmes sont des technologies

Exemple : tri d'un tableau de *n* nombres par ordre croissant

- tri par insertion : nombre d'instructions = $k_{ins}\, n^2$
- tri par fusion : nombre d'instructions = $k_{fus}\, n\, log_2(n)$



Les algorithmes sont des technologies

Exemple : tri d'un tableau de $n = 10^6$ nombres par ordre croissant

Ordinateur rapide : 1 milliard d'instructions par seconde

Très bon programmeur, langage de bas niveau… :
$k_{ins} = 2$

Tri par insertion

⇩

2000 secondes

Ordinateur 100 fois plus lent : 10 millions d'instructions par seconde

Programmeur débutant, langage de haut niveau, compilateur peu performant :
$k_{fus} = 50$

Tri par fusion

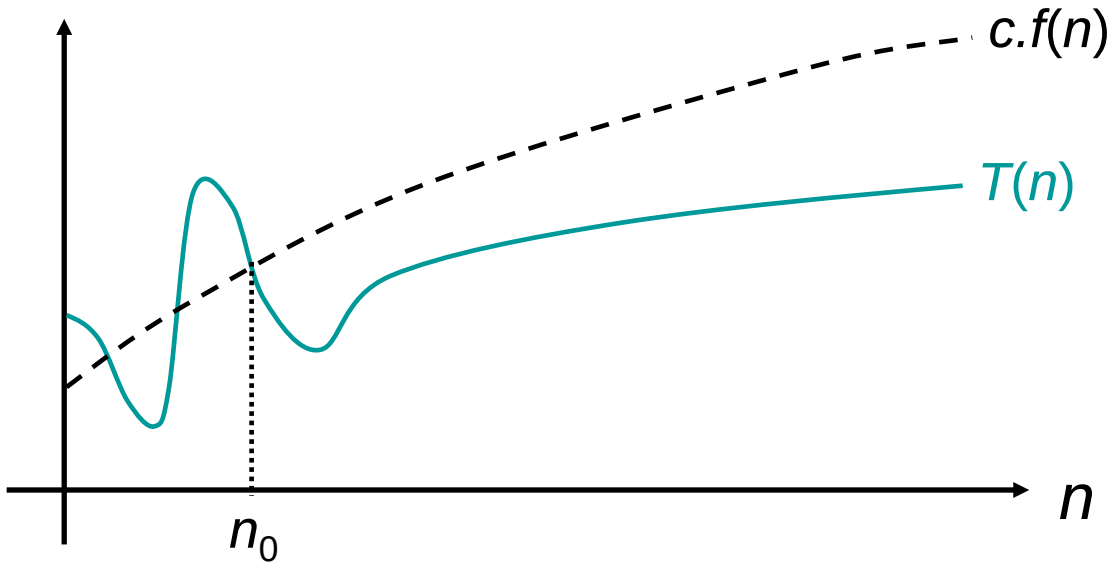
⇩

100 secondes seulement !

PhD comics cartoons by Jorge Cham

Analyse des performances d'un algorithme

- n = nombre d'objets à traiter
- $T(n)$ = nb d'opérations nécessaires = **complexité** de l'algorithme
- Notation $\mathcal{O}()$: un algorithme est $\mathcal{O}(f(n))$ s'il existe deux constantes positives c et n_0 t.q. $T(n) \leq c.f(n)$ quand $n \geq n_0$



- Exemples : tri par insertion $\mathcal{O}(n^2)$, tri par fusion $\mathcal{O}(n.\log_2(n))$

Analyse des performances d'un algorithme

- Règles d'analyse
  - énoncés qui se suivent : somme des nb d'opérations
  - boucle : nb de passages x nb d'opérations par passage
  - si … alors … sinon … : nb d'opérations dans la branche qui en contient le plus (cas le pire)

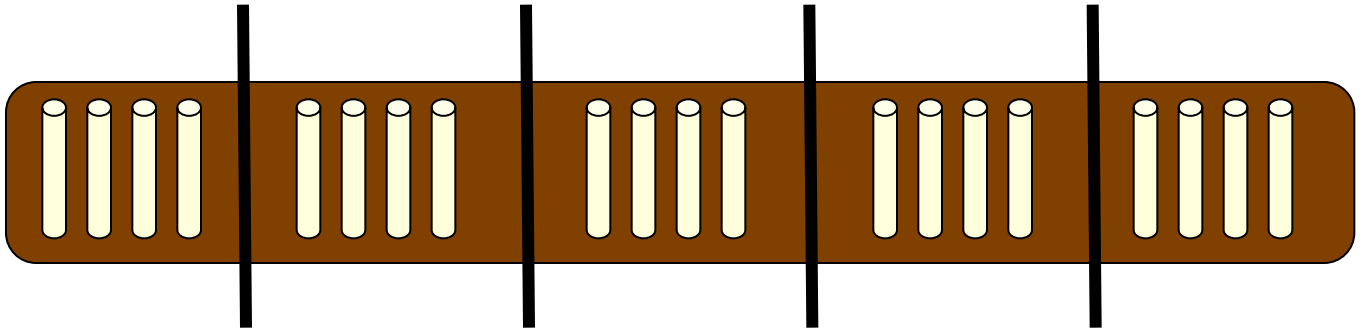
- Règles de simplification pour un polynôme
  - ne garder que le terme de plus haut degré
  - éliminer son facteur constant
  - exemple : $T(n) = 50n^3 + 8n^2 + 78n + 10000$ est $\mathcal{O}(n^3)$

Les structures de données sont des technologies

- Exemple : le jeu de Nim
- Vidéo…

Les structures de données sont des technologies

Stratégie : utiliser une structuration des données qui rend l'algorithme de choix plus simple et plus efficace



- Structuration : 5 cases de 4 bâtonnets chacune.
- Algorithme : Jouer de sorte à laisser 1 bâtonnet dans la case.

Stratégie gagnante à coup sûr si l'on joue en premier.

A retenir :

Les algorithmes peuvent s'appuyer sur une structuration particulière des données pour être plus efficaces

Les structures de données sont des technologies

Autre exemple : le jeu « Spit Not So »

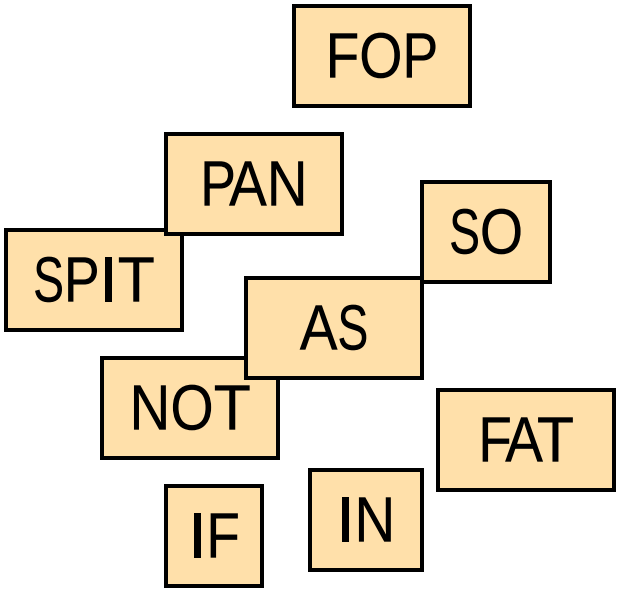
- 2 joueurs
- 9 cartes, faces visibles, avec les mots suivants :

SPIT	NOT	SO	FAT	FOP	AS	IF	IN	PAN

- Chaque joueur prend une carte à chaque tour
- But du jeu : posséder toutes les cartes ayant la lettre « … » quelle que soit cette lettre

Les structures de données sont des technologies

Exemple :



PhD comics cartoons by Jorge Cham

Les structures de données sont des technologies

Exemple :

SPIT

FOP

PAN

SO

AS

NOT

FAT

IF

IN

Les structures de données sont des technologies

Exemple :

SPIT

FOP

PAN

AS

NOT

FAT

IF

IN

SO

Les structures de données sont des technologies

Exemple :

FOP

PAN

AS

NOT

IF

IN

SPIT

FAT

SO

Les structures de données sont des technologies

Exemple :

FOP

PAN

AS

IF

IN

SPIT

FAT

SO

NOT

Les structures de données sont des technologies

Exemple :

PAN

AS

IF

IN

SPIT

FAT

FOP

SO

NOT

Les structures de données sont des technologies

Exemple :

PAN

AS

IN

SPIT

FAT

FOP

SO

NOT

IF

Les structures de données sont des technologies

Exemple :

Gagné !

AS

IN

SPIT

PAN

FAT

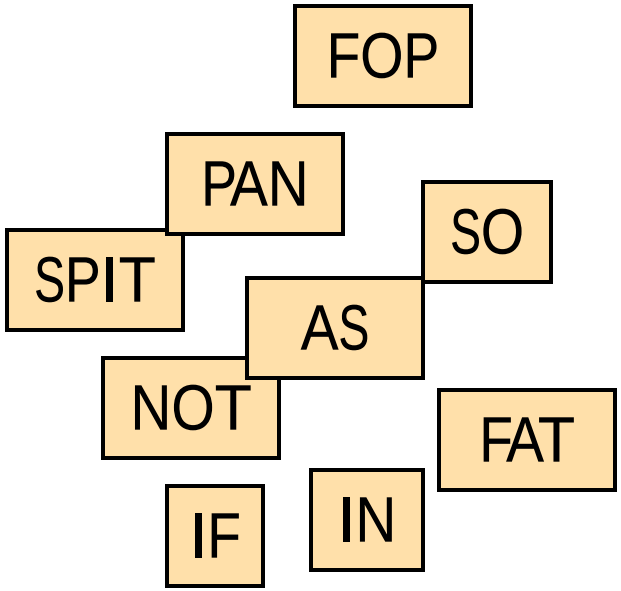
FOP

SO

NOT

IF

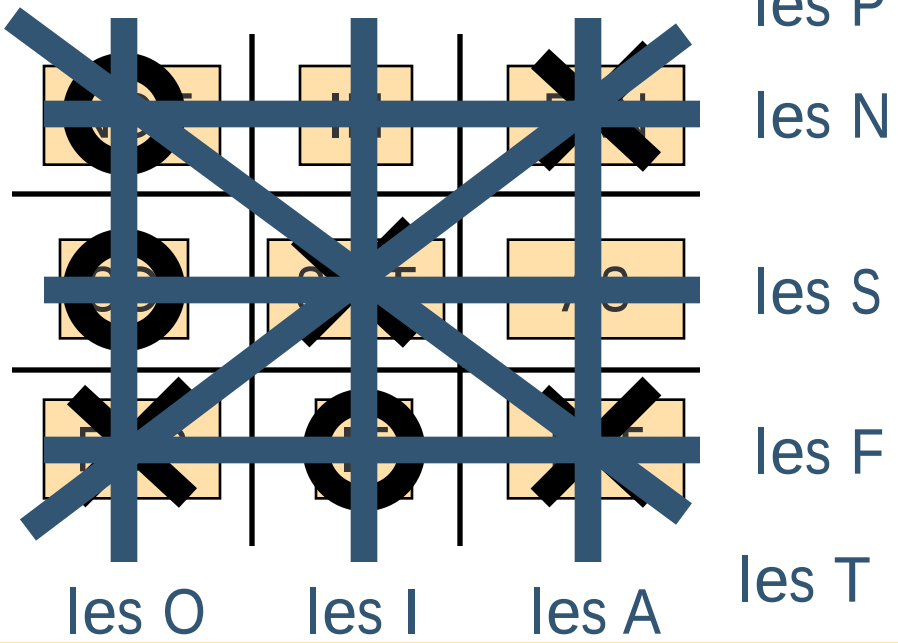
Les structures de données sont des technologies



A votre tour !

Les structures de données sont des technologies

Stratégie du vainqueur : utiliser une structuration des données qui rend l'algorithme de choix plus simple et plus efficace



A retenir :

Les algorithmes peuvent s'appuyer sur une structuration particulière des données pour être plus efficaces

PhD comics cartoons by Jorge Cham

Les structures de données sont des technologies

3ème exemple : le tri par tas d'un tableau de n éléments (sera vu en fin de semestre)

- Réorganiser toutes les données sous forme d'un « tas binaire »
- Une fois réorganisé de cette façon, le tableau peut être trié plus rapidement
- Coût total dans le pire des cas : $\mathcal{O}(n.\log_2(n))$, contre $\mathcal{O}(n^2)$ pour un tri « naïf » comme le tri par insertion

Les structures de données sont des technologies

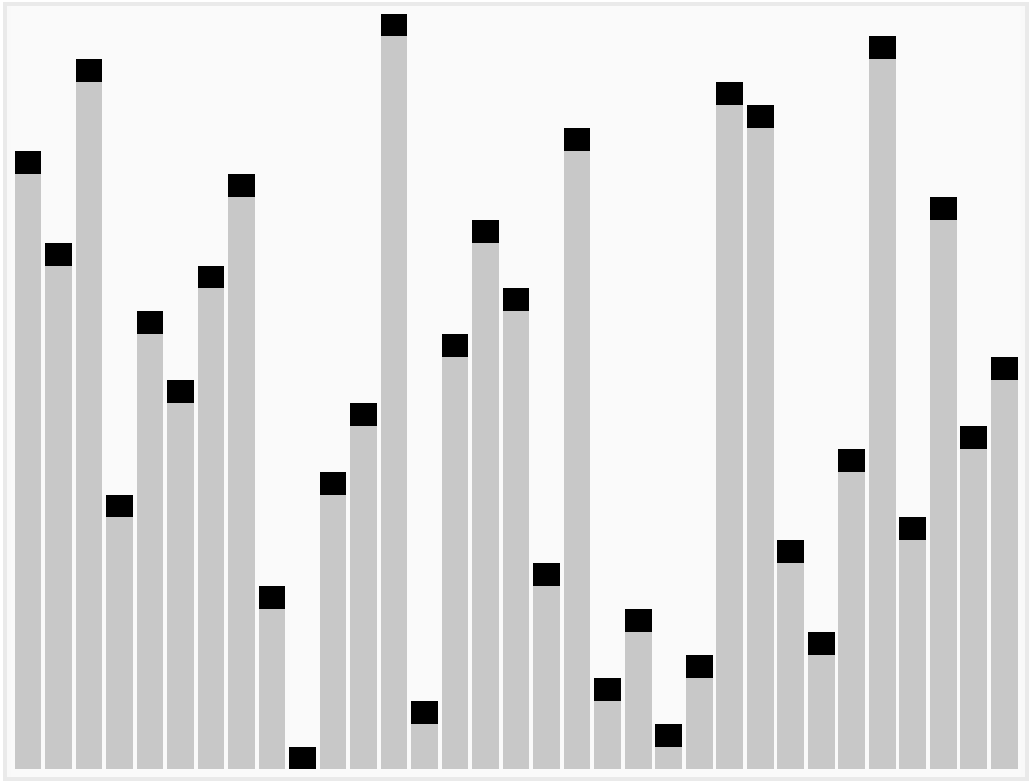


Illustration du tri par tas

Différentes structures de données

Niveau langage (évolué)	tables		arbres	graphes		fichiers
	listes chaînées			files de priorité		tableaux dynamiques
		piles	files			
			tableaux		structures	
	booléens	caractères	entiers	réels		pointeurs
Niveau machine	mots ⇧ bytes (octets) ⇧ bits					

Objectifs d'apprentissage

A la fin de cette UE, vous serez capables :

- d'expliquer pourquoi les algorithmes et les structures de données sont des technologies
- d'expliquer les forces et faiblesses des principales structures de données
- de construire une structure de donnée complexe à partir de structures de base
- de prévoir l'évolution des données manipulées par un algorithme
- de mesurer l'efficacité d'un algorithme simple
- de concevoir un algorithme s'appuyant sur une structure de donnée appropriée
- de le mettre en œuvre en C

Pourquoi le langage C ?

- Windows, Linux et Mac OS X sont écrits en C
  - assembleur : performant mais problèmes de portabilité
  - Java, Perl… : pas d'accès direct à la gestion de la mémoire

- Langage C : permet au programmeur de gérer la mémoire comme s'il avait utilisé l'assembleur (important pour écrire un programme de bas niveau)
  - ex. imprimer un document Word sur une imprimante réseau = envoyer des chaînes d'octets depuis la mémoire de l'ordinateur vers le buffer de la carte réseau…

- Le compilateur C produit un code rapide et performant

- Effet de domino : le langage C est également répandu dans des bibliothèques de plus haut niveau (ex. openGL)