



Initiation à l'informatique (MI-1003)

Université Bordeaux

Année 2015-2016,

Licence semestre 1



Plan du cours

- 1. Présentation et organisation**
- 2. Algorithmes**
- 3. Programmes**
- 4. Manipulation d'images**
- 5. Introduction aux graphes**
- 6. Graphes : définition**
- 7. Degré**
- 8. Chaînes**
- 9. Connexité**
- 10. Coloration**



1- Présentation et organisation

- Objectif et contenu
- Faut-il des connaissances préalables?
- Organisation et site web
- Support de cours
- Modalités de contrôle
- Comptes et tutorat

Objectifs et contenu

■ Objectif :

- Initiation à la programmation et l'algorithmique.

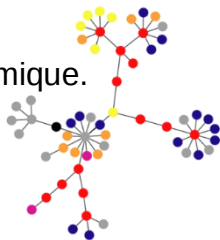
■ Thèmes :

- Manipulation d'images
- Étude d'un objet appelé *graphe*.

■ Organisation :

- Généralités, temps de calcul
- Notions théorique et algorithmes
- Programmation : TP

■ 5 notions abordées : images, graphes, algorithmes, programme, temps de calcul.





Faut-il des connaissances préalables?

■ Non prérequis

- Connaissance d'un langage, d'un système d'exploitation,
- Connaissance de la programmation,
- Connaissance de logiciels destinés au grand public.

■ Prérequis

- Il sera nécessaire de pouvoir comprendre un raisonnement mathématique pour les preuves des théorèmes.



Organisation et site web

- Responsable : Samuel Thibault.

- Site Web :

<http://dept-info.labri.fr/initinfo/>

- Supports de cours.
- Textes des TD, TP.
- Annales d'examens.

- Planning :

- 14 séances de cours intégré (1h20)
- 14 séances de TP (1h20) dont un TP noté.
- travail individuel



Modalités de contrôle

Epreuve	Durée	Coefficient
CC (Tests)	3 x 20mn	0.2
1 TP noté	1h20	0.2
1 DS semaine 47	1h20	0.2
1 examen semaine 2	1h20	0.4



Comptes sur machines - Tutorat

■ Tutorat pour :

- Activation de comptes,
- Prise en main de l'environnement informatique,
- Soutien pour les cours d'informatique,
- Lundi-Jeudi de 12h45 à 13h45 (Rez-de-Chaussée bât. A22) à partir du 15 Septembre, à confirmer



2- Algorithmes

- Qu'est-ce qu'un algorithme?
- Efficacité des algorithmes



Qu'est-ce qu'un algorithme?

- Un algorithme est une **méthode systématique** (comme une recette) pour résoudre un problème donné.
- Il se compose d'une suite **d'opérations simples** à effectuer pour résoudre un problème.
- Exemple : faire n tasses de café

mettre un filtre

Tant que niveau_réservoir < n **faire**

mettre une dose d'eau dans le réservoir

Fin tant que

nb_doses = 0

Tant que nb_doses < n **faire**

mettre une dose de café dans le filtre

augmenter nb_doses de 1

Fin tant que

allumer la cafetière



Qu'est-ce qu'un algorithme?

- Un algorithme est une méthode systématique (comme une recette) pour résoudre un problème donné.
- Il se compose d'une suite d'opérations simples à effectuer pour résoudre un problème.
- En informatique cette méthode doit être applicable par un ordinateur.

Exemple : calculer la somme des diviseurs de l'entier n

```
somme = 0
```

```
si  $n > 0$  alors
```

```
    pour tout entier  $i$  entre 1 et  $n$  faire
```

```
        si  $n$  est divisible par  $i$  alors
```

```
            ajouter  $i$  à somme
```

```
        Fin si
```

```
    Fin pour
```

```
Fin si
```



Importance de l'algorithmique

- Pour **un problème** donné, il y a **plusieurs algorithmes**.
- Il est facile d'écrire des **algorithmes faux ou inefficaces**.
- Une erreur peut faire la différence entre **plusieurs années**
et **quelques minutes de calculs** sur une même machine.
- C'est souvent une question d'utilisation de **structures de données ou d'algorithmes** connus dans la littérature.
- Une **structure de données** est une façon particulière d'organiser les données.

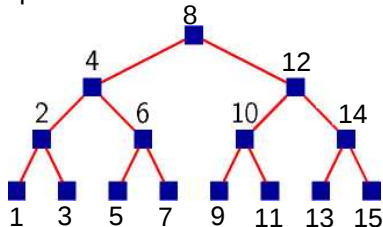
Représenter et organiser les données

Exemple : Construire une ville de 15 maisons en évitant aux livreurs de pizzas qui suivent les rues un trajet trop long depuis la pizzeria.

Organisation 1 : Linéaire. Numéros croissants. Pizzeria au numéro 1.



Organisation 2 : Embranchements. À l'ouest de la maison k , $n^\circ < k$, et à l'est, $n^\circ > k$. La pizzeria est au numéro 8.



Temps de calcul

- Dans les deux organisations le livreur a une méthode simple pour trouver une maison en partant de la pizzeria.
- On suppose qu'il faut une unité de temps pour passer d'une maison à une autre (en suivant une rue).
- Quel est, dans le cas le pire, le temps mis par un livreur pour aller jusqu'à une maison depuis la pizzeria?

Nombre de maisons	Temps organisation 1	Temps organisation 2
15	14	3
1023	1022	9
1073741823	1073741822	29
n	$n-1$	$\sim \log_2(n)$

- **Note** une organisation en étoile avec la pizzeria au milieu permet des trajets très courts, mais **choisir** la bonne rue prend du temps.



Temps de calcul

- Le **temps de calcul** (ou **complexité**) d'un algorithme est la fonction qui à un entier n associe le **nombre maximal d'instructions élémentaires** que l'algorithme effectue, lorsqu'on travaille **sur des objets de taille n** .
- En pratique, on se contente d'un ordre de grandeur.
- **Exemples d'opérations élémentaires** :
 - **additionner, soustraire, multiplier** ou **diviser** deux nombres,
 - **tester** si une valeur est égale à une autre valeur,
 - **affecter** une valeur à une variable.

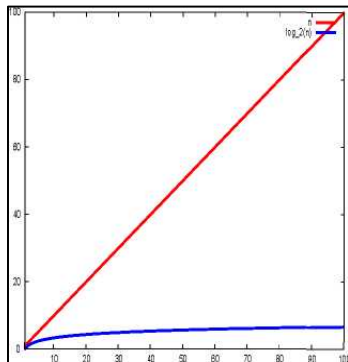


Temps de calcul

- Pour déterminer si un algorithme est efficace, on compte le **nombre d'opérations** nécessaires à effectuer **dans le pire des cas** et *en fonction de la taille de la donnée*.
- Le temps de calcul d'un algorithme est une évaluation du nombre d'opérations élémentaires (opérations arithmétiques) qu'il effectue sur une donnée de taille n .
- Exemple
 - avec l'organisation 1 de la ville, de taille n maisons, l'algorithme naturel pour trouver une maison a une complexité $O(n)$.
 - avec l'organisation 2 d'une ville de taille n maisons, l'algorithme naturel pour trouver une maison a une complexité $O(\log_2(n))$, ce qui est bien inférieur.

Différence entre n et $\log n$

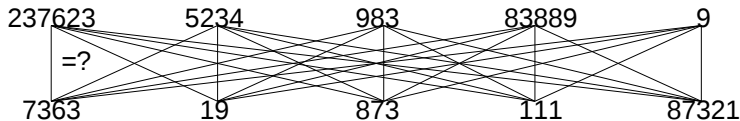
- Pour notre livreur de pizza
 - Si $n = 10^6$, alors $\log_2 n \approx 20$
 - Il fait 50 000 fois moins de déplacements si les maisons sont organisés par « embranchements »
 - Si $n = 10^9$, alors $\log_2 n \approx 30$, il fait alors 30 000 000 fois moins de déplacements.



Temps de calcul : 2ème exemple

Problème : déterminer si 2 ensembles $E1$, $E2$ de n entiers ont une valeur commune.

Algorithme 1 : comparer successivement chaque élément de $E1$ avec chaque élément de $E2 \rightarrow n^2$ comparaisons.



On peut résoudre le problème avec environ $n \cdot \log(n)$ comparaisons!

$ E1 = E2 $	Algorithme 1	Algorithme 2
n	n^2	$n \cdot \log(n)$
10	100	10
1000	1000000	3000
100000	10000000000	500000



Différence entre $O(n^2)$ et $O(n * \log(n))$

Sur un ordinateur exécutant une instruction élémentaire en 10^{-9} sec

- Si les ensembles $E1$ et $E2$ ont $n = 1\,000\,000 = 10^6$ éléments
 - Exécuter $n * \log n$ instructions élémentaires nécessite 0,006s.
 - Exécuter n^2 instructions élémentaires nécessite 10^3 s soit environ 16mn40.
- Si les ensembles $E1$ et $E2$ ont $n = 10\,000\,000 = 10^7$ éléments
 - Exécuter $n * \log n$ instructions élémentaires nécessite 0,07s.
 - Exécuter n^2 instructions élémentaires nécessite 10^5 s soit plus d'une journée.
- En informatique, on manipule parfois des ensembles énormes
 - Google indexe plusieurs milliards de pages web,
 - Google reçoit près de 200 millions de requêtes/jour.

Représentation graphique : réseaux

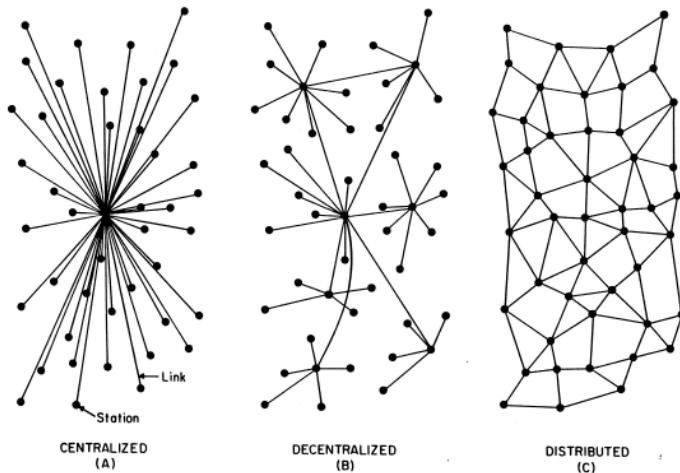
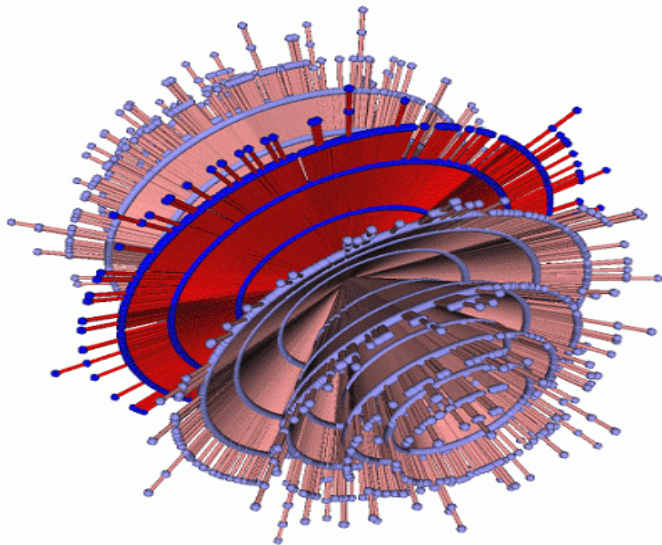


FIG. 1 – Centralized, Decentralized and Distributed Networks

Représentation graphique : réseaux





Qu'est-ce que l'informatique?

- L'informatique même pour non informaticiens
- Quelques domaines de l'informatique



Qu'est-ce que l'informatique?

- Dans la vie quotidienne : ordinateur avec logiciels.
- En entreprise : un outil de communication et de production.
- À l'université : une discipline scientifique.
 - Une partie pratique (par exemple, autour de la programmation).
 - Une partie théorique similaire aux maths (objets abstraits).
 - Les objets en mathématiques : nombres, relations, fonction, transformations, *etc.*
 - Les objets en informatique : algorithmes, programmes, preuves, systèmes de réécriture, images numériques, graphes, *etc.*



L'informatique pour non informaticiens

- Le travail d'un scientifique ou d'un ingénieur nécessite de plus en plus la manipulation de logiciels.
- Ces logiciels sont de plus en plus sophistiqués.
- Souvent, ces logiciels nécessitent de la programmation.
- Il faut des connaissances informatiques (algorithmique et programmation) pour
 - programmer efficacement,
 - maintenir les programmes.

Exemples de domaines en informatique

■ Les bases de données

- 1.070.000.000 internautes en 2005
- 42 298 371 sites web en 2003
- 100 millions transactions FedEx / jour
- 150 millions transactions VISA / jour
- 300 millions appels longue distance / jour sur le réseau ATT's
- 35 milliards e-mails / jour dans le monde

- Trouver rapidement un billet d'avion, un trajet, une page web,...
- Traçabilité des transactions en agro-alimentaire, dans le domaine financier, ...
- Croiser les informations des corps policiers au niveau européen, ...
- Systèmes d'informations géographiques



Exemples de domaines en informatique

■ La sécurité

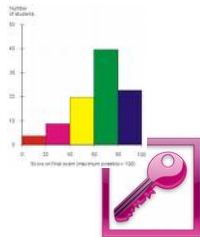
- Transports
- Médecine,
- Finance
- Communications
- Énergie
- Systèmes embarqués



Exemples de domaines en informatique

■ Les logiciels

- Navigateurs internet
- Anti-virus
- Pare-feu ou passerelle
- Clients de messagerie (mail)
- Jeux
- ...



■ Les langages de programmation

- Les langages de programmation sont souvent utilisés dans des domaines spécifiques.
- HTML, php, javascript pour la création de pages web,
- SQL pour les bases de données,
- Java pour les applications embarquées, les serveurs, + ...
- C pour les systèmes d'exploitation (Windows, Unix), +...
- **Python** est relativement générique : sites web (youtube, instagram, ...) traitement de données (gmail, google maps, dropbox, ...) scripts de jeux (Civilization IV, EVE Online, ...),
...

Exemples de domaines en informatique

■ Image et son

- MP3, JPEG, MPEG : codage et compression.
- Voix par IP, numérisation et transformation.
- Image 3D, jeux vidéos...

