INTRODUCTION AUX STRUCTURES DE DONNÉES

DONNÉE, INFORMATION, MÉMOIRE, ALGORITHME, COMPLEXITE

ISABELLE WATTIAU

DONNÉE VS. INFORMATION

- Une donnée est le résultat d'une mesure
  - Exemple : 27°C
- Une information est une donnée interprétée :
  - 27°C est la température de la pièce où se trouve le serveur
- Souvent les deux termes sont utilisés indifféremment

LES CONCEPTS DE BASE

- Donnée : représentation d'une information
- Type de donnée :
  - définit la nature du codage et les opérations autorisées sur une donnée
- Donnée simple : non décomposable
- Donnée complexe : composée de données simples ou complexes
- Exemples :
  - entier : donnée simple
  - chaîne de caractère : simple ou complexe
  - donnée multimédia : simple ou complexe

UNITES DE MESURE

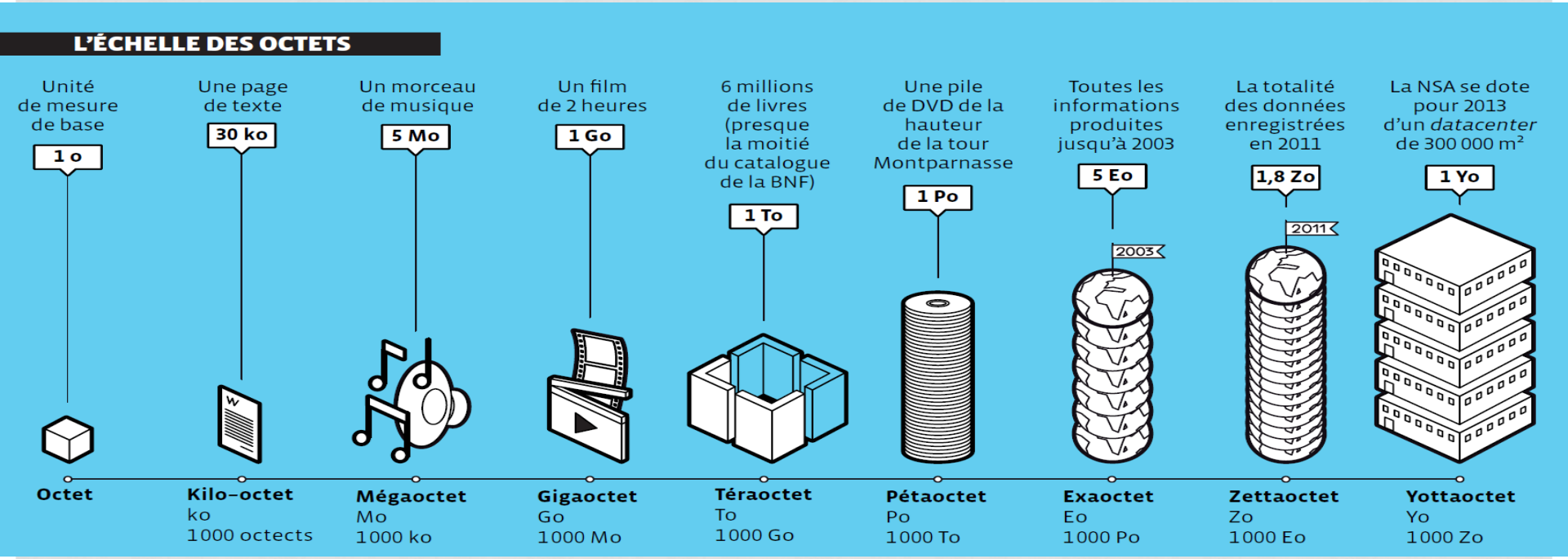
- 1 bit = 1 binary digit = 1 valeur binaire (0 ou 1)
- 1 octet = 8 bits
- 1 K-octet = 2^{10} octets = 1024 octets = environ 1000
- 1 M-octet = 2^{20} octets = 1024*1024 octets = environ 1 million

- 1 mot = unité de base manipulée par un processeur
  - Un mot peut être égal à 8, 16, 32 ou 64 bits
  - On dit que le processeur est 8 bits ou 16 bits, etc.

- 1 byte = un octet ou un mot selon les cas

CAPACITES DE STOCKAGE

- Source CNRS



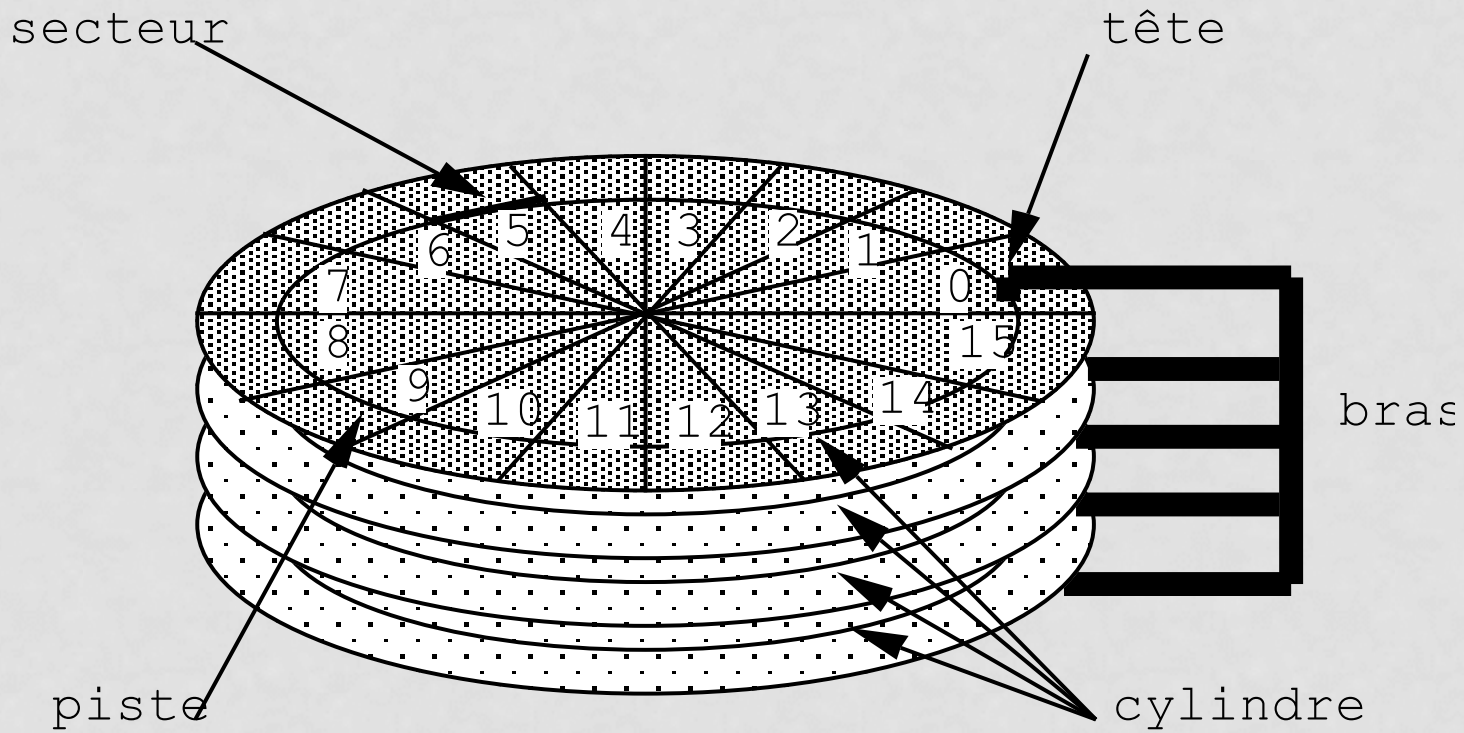
MÉMOIRE, MÉMOIRES

- Mémoire centrale ou mémoire vive ou mémoire RAM (Random Access Memory) ou mémoire interne
  - Mémoire volatile
  - Rapide d'accès pour le processeur
  - Temps d'accès : en nanosecondes
- Mémoire externe ou mémoire secondaire ou mémoire disque ou mémoire auxiliaire ou mémoire de masse
  - Non volatile
  - Pour la sauvegarde
  - Temps d'accès : en millisecondes
- Mémoire volatile : son contenu est perdu en l'absence d'alimentation électrique

SUPPORT DE MEMORISATION

- Dispositif de stockage de l'information
- Mémoire de masse :
  - De grande capacité
  - Non volatile
  - Fonctionne en lecture et écriture
- Technologies :
  - Passées : ruban perforé, cassette audio, tambour magnétique, disquette
  - Actuelles : bande magnétique, disque dur, disque SSD, disque optique, disque magnéto-optique, mémoire flash
  - Futures : mémoire holographique, memristor, etc.

DISQUE MAGNETIQUE OU DISQUE DUR



Extrait de C. Carrez, Dunod, 1997

DISQUE ET TEMPS D'ACCES

Temps de déplacement du bras	Délai rotationnel	Temps de transfert	Volume de 1 K	Volume de 20 K
5 ms	6 ms	5 Mo/s	12 ms	15 ms
25 ms	8,3 ms	250 Ko/s	37 ms	113 ms

Extrait de C. Carrez, Dunod, 1997

- Support privilégié pour des informations à retraiter
- Économique : 0,04 euro / Go
- Temps d'accès : en ms

ALGORITHME

- **Suite de règles opératoires** dont l'application permet de résoudre un problème en un **nombre fini** d'opérations
- Écriture informelle d'un futur programme
- L'écriture d'un algorithme :
  - est une étape préalable au codage (écriture du programme)
  - facilite la définition des étapes du programme
- Analogie : une recette de cuisine
- L'écriture de l'algorithme permet de :
  - préciser les actions sans les détails des opérations
  - vérifier la correction de l'algorithme
  - étudier son efficacité en temps et en espace mémoire

- L'étape suivante :
  - implantation de l'algorithme, ou mise en œuvre, ou codage dans un langage de programmation

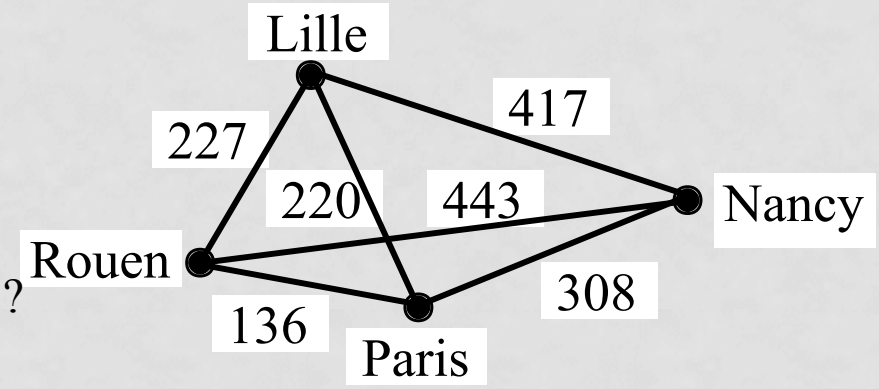
EXEMPLE D'ALGORITHME

- Réponse à la question : n est-il un nombre premier ?
- Ecriture intuitive :
  - On teste tous les diviseurs compris entre 2 et n-1
  - On s'arrête dès qu'un diviseur est trouvé
- Algorithme :

```
Début
d:=2;
trouvé:=faux;
Tant que non trouvé et d<=n-1 faire
        si n modulo d = 0 alors trouvé:=vrai sinon d:=d+1 finsi
 Fin tant que;
 Si trouvé=vrai
        alors écrire(n,'n est pas premier')
        sinon écrire (n,'est premier')
 Finsi
 Fin
```

Le problème du voyageur de commerce

Comment se rendre successivement dans toutes ces villes en parcourant la distance la plus courte possible ?



- parcours (Ville$_1$,...,Ville$_n$) = Σ distance(Ville$_{i-1}$,Ville$_i$)
  calculer le parcours de toutes les permutations prendre et choisir
- Additions : n * n!
  - 5 villes => 600 additions
  - 20 villes => 5 .10 ^{19} additions, soit 1,5 millions d'années
- Note : résolution par des heuristiques

EVALUATION DE LA COMPLEXITE

- On n'évalue pas le temps d'exécution
- Mais le nombre d'opérations (ex : additions) qu'effectue l'algorithme *en fonction* du nombre de données (ex : villes) à traiter

OPERATION FONDAMENTALE

- Une opération fondamentale peut être complexe, mais elle a une durée indépendante des données
- Si plusieurs opérations fondamentales sont à considérer :
  - On fait un décompte séparé
  - Et, éventuellement, on applique un coefficient
- Les opérations de détail sont prises en compte par absorption
- On peut comparer des algorithmes s'ils utilisent les mêmes opérations
- Exemples d'opérations :
  - addition des distances dans voyageur de commerce
  - comparaison de valeurs dans une recherche ou un tri
  - déplacements de valeurs dans un tri
  - accès à la mémoire secondaire

CALCUL DE LA COMPLEXITE

- Consiste à évaluer le nombre d'exécutions de l'opération fondamentale
- Exemple : recherche de la présence de x dans le tableau T

```
var T: tableau [1..N] de T_Elem;  {rangé par valeurs croissantes}
        k : entier;
début k := 1;
tant que k <= N et T[k] < x   faire k := k+1; fin tant que;
retourner k;
fin
```

- Opération fondamentale : test de la condition de boucle
- Complexité : au mieux 1, au pire N, en moyenne N/2

  {1, ou 2, ou 3,..., N} si k≤N et N si k=N+1

COMPLEXITE : DEFINITIONS

- Soit D_n l'ensemble des données de taille n
  et $coût_A$ (d) le coût de l'algorithme sur la donnée d
- *complexité au mieux* $Min_A(n)=min\{coût_A(d) \mid d \in D_n\}$
- *complexité au pire* $Max_A(n)=max\{coût_A(d) \mid d \in D_n\}$
- *complexité en moyenne* $Moy_A(n)=\Sigma\{p(d)*coût_A(d) \mid d \in D_n\}$
  où p(d) est la probabilité d'avoir la donnée d

$$Min_A(n) \leq Moy_A(n) \leq Max_A(n)$$

- En général on évalue la complexité en moyenne et au pire

ALGORITHME DE RECHERCHE DICHOTOMIQUE

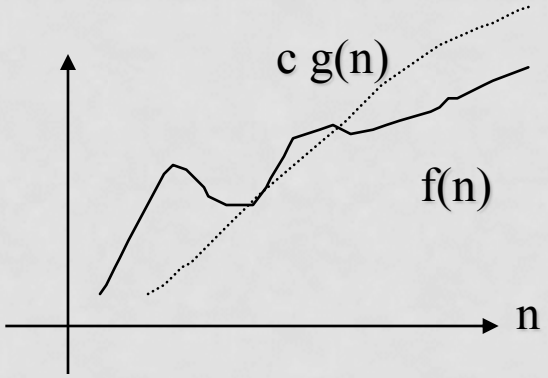
```
var T: tableau [1..N] de T_Elem;
      i, j, k: entier;
début
   i := 1; j := N;
   tant que i < j faire
        k := (i+j) / 2;
        si   T[k] < x    alors i := k+1;
                          sinon j := k;
   finsi;
   fin tant que;
   si i = N et T[i] < x alors
         i := N+1;
   finsi;
   retourner i;
fin;
```

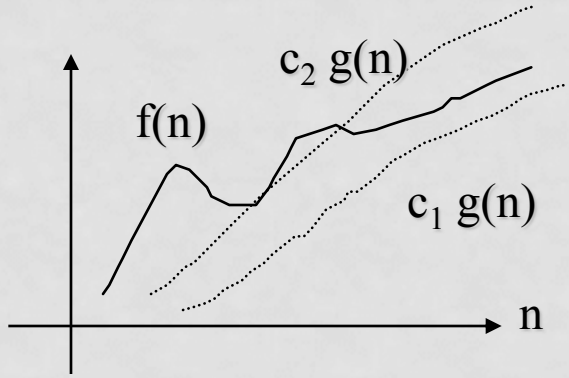
- Valeurs rangées en ordre croissant
- Invariants :
  - $m < i \Rightarrow T[m] < x$
  - $j \neq N \Rightarrow x \leq T[j]$
  - $N \neq 0 \Rightarrow i \leq j$
- arrêt:
  - nouveau(j-i) = ancien(j-i) / 2
- complexité:
  - $\lceil \log_2 N \rceil$

⇨ meilleur que précédent sauf si début

ORDRE DE GRANDEUR

- L'important est l'ordre de grandeur, et l'évolution en fonction de la taille
- Définition : f est *dominée* par g, f = O(g) s'il existe n_0 et c tels que pour tout $n > n_0$, on ait $f(n) \leq c\ g(n)$
- définition: f et g sont *du même ordre de grandeur*, f = Θ(g) s'il existe n_0 , c_1 et c_2 tels que pour tout $n > n_0$, on ait $c_1\ g(n) \leq f(n) \leq c_2\ g(n)$



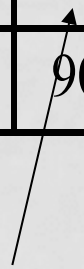


TEMPS D'EXECUTION SELON LA TAILLE

	Constante		Linéaire		Quadratique		Exponentielle
Taille	1	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
10^2	1µs	7µs	0.1 ms	0.7 ms	10 ms	1 s	$4\,10^7$Ga
10^3	1µs	10µs	1 ms	10 ms	1 s	17 mn	□
10^4	1µs	13µs	10 ms	130 ms	1.7 mn	12 j	□
10^5	1µs	17µs	0.1 s	1.7 s	2.8 h	32 a	□
10^6	1µs	20µs	1 s	20 s	12 j	32 Ka	□
		Logarithmique		n-Logarithmique		Cubique	

TAILLE POSSIBLE SELON LA COMPLEXITE

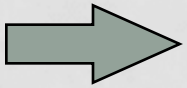
	1	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
1 s	□	□	1 M.	63 K.	1 K.	100	20
1 mn	□	□	60 M.	3 M.	7 K.	400	26
1 h	□	□	4 G.	130 M.	60 K.	1.5 K.	32
1 j	□	□	90 G.	3 G.	300 K.	4.4 K.	36



- Exemple : si l'algorithme est de complexité linéaire, il peut traiter 4 Go de donnée en une heure

COMPLEXITE EN PLACE MEMOIRE

- Évaluer la quantité de mémoire nécessaire en fonction de la taille des données

conservation des résultats intermédiaires  diminution de complexité en temps

- S'assurer que la mémoire est suffisante

compromis espace-temps

CONCLUSION

- Algorithme :
  - 1) correct
  - 2) temps d'exécution raisonnable
- Complexité relative à une ou des opérations fondamentales
  - dépend de la taille des données et de leur configuration
    - complexité < n^2 => on peut traiter un problème de taille quelconque
    - entre n^2 et n^3 => taille moyenne
    - > n^3 => petite taille
- L'amélioration de la performance des machines ne change pas l'efficacité
- Compromis espace-temps
- Problématique des accès disque