Université Lille1 Sciences et Technologies
UFR IEEA

Algorithmique Avancée et Complexité

2012–2013
Master d'Informatique-S1
TD 1

**Complexité des algorithmes**

## Comparer les complexités

**Exercice 1 : Vrai ou Faux ?**

Pour chacune des affirmations suivantes, dire si elle est vraie ou fausse :

**Q 1.** *Histoire d'O :*

**Q 1.1.** $n \in O(n^2)$ ?
**Q 1.2.** $n^2 \in O(n)$ ?
**Q 1.3.** $\log n \in O(n)$ ?
**Q 1.4.** $n \log n \in O(n)$ ?
**Q 1.5.** $n \in O(n \log n)$ ?
**Q 1.6.** $n \in O(n + \log n)$ ?
**Q 1.7.** $n + \log n \in O(n)$ ?
**Q 1.8.** $4n^2 \in O(n)$ ?
**Q 1.9.** $100n^2 + n + 5 \in O(n^2)$ ?
**Q 1.10.** $(\log n)^2 \in O(n)$ ?
**Q 1.11.** $2^n \in O(3^n)$ ?
**Q 1.12.** $3^n \in O(2^n)$ ?
**Q 1.13.** $n! \in O(2^n)$ ?
**Q 1.14.** $2^n \in O(n!)$ ?

**Q 2.**... *et de* $\Theta$

**Q 2.1.** $n \in \Theta(n^2)$ ?
**Q 2.2.** $\log n \in \Theta(n)$ ?
**Q 2.3.** $n \log n \in \Theta(n)$ ?
**Q 2.4.** $n + \log n \in \Theta(n)$ ?
**Q 2.5.** $n^2 + n + n \log n \in \Theta(n^2)$ ?
**Q 2.6.** $(\log n)^2 \in \Theta(n)$ ?
**Q 2.7.** $\log(n^2) \in \Theta(\log n)$ ?

**Q 3.** *Vrai ou Faux* Dans cette question, $f$, $g$ et $h$ sont des fonctions de $\mathcal{N}$ dans $\mathcal{R}^+$. Quelles affirmations sont justes ?

**Q 3.1.** $5f \in \Theta(f)$ ?
**Q 3.2.** $5f \in O(f)$ ?
**Q 3.3.** $f \in O(g) \Rightarrow g \in O(f)$ ?
**Q 3.4.** $f \in O(g) \Rightarrow f \in \Theta(g)$ ?
**Q 3.5.** $f \in \Theta(g) \Rightarrow f \in O(g)$ ?
**Q 3.6.** $f \in \Theta(g) \Rightarrow g \in O(f)$ ?
**Q 3.7.** $f \in \Theta(g) \Rightarrow g \in \Theta(f)$ ?
**Q 3.8.** $f \in \Theta(g) \Rightarrow f + g \in \Theta(g)$ ?
**Q 3.9.** $f \in \Theta(n) \Rightarrow f^2 \in \Theta(n^2)$ ?
**Q 3.10.** $f \in O(g), g \in O(h) \Rightarrow f \in O(h)$ ?

**Exercice 2 : D'un algorithme à l'autre**

**Q 1.** *Pour le meilleur et pour le pire*

**Q 1.1.** La complexité exacte dans le pire des cas de $A$ est $n^2$, celle de B est en $n$. Peut-on en déduire que $B$ est plus rapide que $A$ sur certaines données ? Sur toutes les données ?

**Q 1.2.** Mêmes questions si la complexité exacte dans le meilleur des cas de $A$ est $n^2$, celle dans le pire des cas de $B$ est $n$.

**Q 1.3.** Mêmes questions si la complexité exacte dans le meilleur des cas de $A$ est $n$, celle dans le pire des cas de $B$ est $n^2$.

**Q 1.4.** Si un algorithme $A$ a une complexité dans le meilleur des cas en $\Theta(n)$ et une complexité dans le pire des cas en $\Theta(n)$, que puis-je en déduire pour sa complexité en moyenne ?

**Q 1.5.** La complexité dans le pire des cas de l'algorithme $A$ est en $\Theta(n \log n)$, celle de l'algorithme $B$ est en $\Theta(n^2)$. Peut-on en déduire que l'algorithme $A$ est préférable à $B$ ?

Avez-vous un exemple de telles situations ?

**Q 2.** *Ne négligeons pas trop vite les constantes !* On suppose que la complexité exacte de l'algorithme $A$ est $10^6 n^2$, celle de $B$ $n^3$.

**Q 2.1.** A partir de quelle taille de données $A$ est-il plus rapide que $B$ ?

**Q 2.2.** Si on suppose qu'une instruction s'exécute en $1\mu s$, cela correspond approximativement à quel temps d'exécution ? Même question pour $10ns$ ?

**Q 3.** Un algorithme dont la complexité exacte est $k * n^2$ , est remplacé par un algorithme de complexité exacte $k * n * \log n$. On suppose ici que, pour les deux algorithmes, la complexité dans le meilleur des cas est la même que celle dans le pire des cas, c'est-à-dire que la complexité ne dépend que de la taille de la donnée.

Par combien le temps d'exécution est divisé pour les données de taille $n = 1000$ ? pour les données de taille $n = 10000$ ?

**Exercice 3 : A vos chronos**

**Q 1.** Un algorithme de tri prend 1 seconde sur votre machine pour trier 1000 éléments. Combien de temps prendra-t-il pour en trier 10.000 si

. vous supposez que son temps d'exécution est proportionnel à $n^2$ ?

. vous supposez que son temps d'exécution est proportionnel à $n \log n$ ?

**Q 2.**Un algorithme prend 1 seconde sur votre machine pour traiter une donnée de taille $n_0$. Vous échangez votre machine contre une machine dix fois plus rapide. Quelle est la taille maximale d'une donnée pouvant maintenant être traitée en 1 seconde si

. vous supposez que le temps d'exécution de l'algorithme est proportionnel à $n$ ?

. vous supposez que son temps d'exécution est proportionnel à $n^2$ ?

. vous supposez que son temps d'exécution est proportionnel à $\log n$ ?

. vous supposez que son temps d'exécution est proportionnel à $2^n$ ?

## Analyse d'algorithmes : quelques rappels..

**Exercice 4 :** *Quelques basiques*

**Q 1.** *Recherche basique*

Calculer la complexité dans le pire des cas et dans le meilleur des cas de l'algorithme :

```
//n entier >0 , t tableau de n entiers
 boolean recherche(int x){
int j=0;
   while ((j<n) && (t[j]!=x))) j++;
return (j<n);
}
```

Comment pourrait-on calculer la complexité en moyenne ?

**Q 2.** Que retourne la fonction `myst` ci-dessous ?
Evaluer l'ordre de grandeur de sa complexité.

```
// n entier >=0
int myst (Int n) {
  int   r=0;
  for (i=1; i<=n-1; i++)
     for (j=i+1; j <=n; j++) r++;
  return r; }
```

**Q 3.** Même question pour :

```
// n entier >=0
int myst2 (Int n) {
  res=0;
  for (i=n; i>1; i=i div 2) res=res+1;
  return res;}
```

**Q 4.** Pour chacun des algorithmes suivants, dire si sa complexité (temporelle) dans le pire des cas est en $\Theta(log\ n)$, $\Theta(n)$, $\Theta(n\ log\ n)$ ou $\Theta(n^2)$, en justifiant brièvement :

```
int T1(int n){
  int c=0;
  for (int j=0;j<3;j++)
    for (int i=n;i>0;i=i-4)  c++;
  return c;}
```

```
int T2(int n){
  int c=0;
  for (int i=1;i<n;i=4*i) c++;
  return c;}

int T3(int n){
  int c=0;
  for (int i=0;i<n;i++)
    for (int j=1;j<i;j=j+3) c++;
  return c;}
```

```
int T4(int n){
  int c=0;
  for (int i=n;i>0;i--)
    for (int j=i; j<i+3;j++) c++;
  return c;}

int T5(int n){
  int c=0;
  for (int i=n;i>0; i=i div 3) c++;
  return c;}
```

**Q 5.** *Vision globale...*
Evaluer l'ordre de grandeur de la complexité de la fonction ci-dessous :

```
//n entier >0, t tableau  de n entiers
void balayage_inutile(){
  int j=0;
  for (int i=0; i<=n-1; i++){
     while ((j<n) && (t[j]==t[i])) j++;
  }
}
```

**Q 6.** *Aanalyse d'algorithmes récursifs* Pour chacun des algorithmes récursifs suivants, donner l'ordre de grandeur de la complexité temporelle en évaluanr l'ordre de grandeur du nombre d'appels récursifs effectués :

```
int A1(int n){
  if (n>1) return A1(n-1)
  else return 1; }

int A2(int n){
  if (n>1)  return A2(n-2) ;
  else return 1;}
```

```
int A3(int n){
  if (n>1)  return A3(n div 2 )
  else return 1;}

int A4(int n){
  if (n>2) return A4(n-1) + A4(n-3);
  else return 1;}
```

**Exercice 5 :** *Multiplication et Exponentiation*

Soit le problème suivant :
*Entrée :* $n$ et $k$ deux entiers positifs
*Sortie :* $n^k$

**Q 1.** Quelle est la taille de l'entrée en fonction de $n$ et $k$ ?

**Q 2.** *D'un coût à l'autre...* Soit l'algorithme suivant :

```
product=1;
for (i=1;i<=k;i++) product=product *n;
```

**Q 2.1.** On suppose qu'on est en coût uniforme, i.e. que le coût d'une multiplication est 1, quelle que soit la taille de ses opérandes. Quelle est la complexité de l'algorithme ci-dessus ? Est-il polynomial ?

**Q 2.2.** On suppose maintenant que le modèle est celui du coût logarithmique, i.e. que les opérations élémentaires sont les opérations bit à bit.

Si l'algorithme utilisé pour la multiplication est « l'algorithme de l'école primaire », quel est le coût de la multiplication d'un nombre à $x$ bits par un nombre à $y$ bits ?

Calculer alors le coût de l'algorithme.

**Q 3.** Quels algorithmes pour la multiplication de deux entiers connaissez-vous ?

**Q 4.** Connaissez-vous un algorithme plus efficace que celui-ci pour l'exponentiation ?



**Exercice 6 :** *Halte au gaspillage*

Soient les deux fonctions sur les entiers naturels définies comme suit :
*rec1*(n)= si n $<$ 1 alors 1 sinon 2 $\times$ *rec1*(n-1)
*rec2*(n)= si n $<$ 1 alors 1 sinon *rec2*(n-1) + *rec2*(n-1)
Calculer et comparer la complexité en nombre d'appels récursifs des fonctions rec1 et rec2.

## Quelques algorithmes à trouver... et à prouver

**Exercice 7 : Un tableau bien rangé**

On a un tableau de nombres à $n$ lignes et $m$ colonnes ; on recherche si un élément $x$ est dans ce tableau.

**Q 1.** On suppose d'abord que chaque ligne est triée, par exemple dans l'ordre croissant. Quel algorithme proposez-vous ? Quelle sera sa complexité ? Justifiez sa correction.

**Q 2.** On suppose maintenant qu'à la fois les lignes et les colonnes sont triées dans l'ordre croissant. Proposer un algorithme en $O(n+m)$ qui recherche un élément $x$ dans ce tableau. Prouver que votre algorithme est correct.

**Exercice 8 : Atteindre la cible**

Soit $S$ un ensemble de $n$ nombres et un entier cible $c$. Proposez un algorithme en $0(n \log n)$ qui détermine si il existe deux éléments $x$ et $y$ de $S$ tels que $x + y = c$. Justifiez sa correction.

**Exercice 9 : Majorité absolue ?**

On veut tester si il existe une valeur majoritaire (absolue) dans une liste, et connaître cette valeur si elle existe. Le problème est donc :
*Entrée :* $x_1, \ldots, x_n$
*Sortie :*
$a$ si $x_i = a$ pour au moins $n/2 + 1$ indices $i$.
Non, si il n'existe pas de tel $a$.

Proposer un algorithme linéaire (en $O(n)$) pour ce problème.

*Aide :* On pourra utiliser, en plus de la liste $L$ des éléments de départ, « la valeur majoritaire dans $L$ si elle existe, est aussi majoritaire dans $L_{aux}$ ».

**Exercice 10 : L'intrus**

Soit un tableau d'entiers. On sait que chaque élément du tableau est présent exactement deux fois dans le tableau sauf un qui est présent une seule fois. On cherche cet élément. Par exemple, si le tableau contient 12, 45, 7, 45, 8, 12, 8, le résultat attendu est 7. Proposer un algorithme en $\Theta(n)$ pour ce problème.

On se placera comme d'habitude dans le modèle de coût uniforme, supposant donc que les opérations de base sur les entiers sont en coût constant.

**Exercice 11 : Championnat**

On cherche à déterminer la meilleure de $n$ équipes ;

**Q 1.** *Coupe ?* On organise d'abord une coupe : à chaque tour, chaque équipe restante rencontre une et une seule équipe. Seules les équipes gagnantes participent au tour suivant. (Il n'y a pas de match nul.)

**Q 1.1.** Combien faut-il de tours ? Combien de matchs en tout seront joués ? Peut-on faire mieux ?

**Q 1.2.** Dans quelle mesure peut-on dire que « le meilleur gagne » ? On voudrait maintenant récompenser le vice-champion. Comment le détermineriez-vous ? Pourquoi ?

**Q 2.**... *ou championnat ?* On organise maintenant la compétition différemment. Chaque équipe rencontre toutes les autres équipes une et une seule fois. Le gagnant sera celui qui aura marqué le plus de points selon un calcul que nous n'évoquerons pas ici. Le championnat est donc organisé en $n - 1$ journées.

Proposez un algorithme qui organise les matchs :
*Donnée :* $n$ le nombre d'équipes
*Sortie :* le planning des $n - 1$ journées, i.e. pour chaque journée, la liste des matchs.