TD 10 : complexité

Dans tous les exercices, lorsque l'on demande de déterminer une complexité, bien préciser le modèle de calcul : taille du problème, opérations élémentaires, etc. En revanche, on pourra se contenter de déterminer l'*ordre de grandeur* de la complexité et non sa valeur exacte. On pourra utiliser la notation $O(\ldots)$, ou se contenter de calculer « avec les mains ».

Exercice 1.

(1) Proposer une implantation de la fonction suivante :

```
/** Trouve la dernière apparition d'un entier dans un tableau
 * @param v un tableau d'entiers
 * @param x un entier
 * @return la position de la dernière occurrence de x dans v,
 *         ou -1 s'il n'y a pas de telle occurrence
 **/
int dernierePosition( vector<int> v, int x ) {
```

(2) Quelle est la complexité au pire de votre algorithme ?

(3) (♣) Quelle est la complexité en moyenne de votre algorithme, sous l'hypothèse que x apparaît exactement une fois dans v ?

Exercice 2.

On considère les deux fragments de programmes suivants :

```
for ( int i = 0; i < n; i++ )
    for ( int j = 0; j < n; j++ )
        if ( i == j ) drawPixel( i, j, rouge );
```

```
for ( int i = 0; i < n; i++ )
    drawPixel( i, i, rouge );
```

Que font-ils ? Quelle est leur complexité ? Lequel est meilleur ?

Exercice 3.

On considère deux algorithmes A et B de complexité respective $1\,000 \log_2 n + 50$ et $4n$.

(1) Tracer les deux courbes sur le même graphique, pour n entre 1 et 10 000.

(2) Pour des petites et des grandes valeurs de n, quel algorithme est le plus performant ?

(3) Quelle est la valeur du seuil pour passer de l'un à l'autre ?

Exercice 4.

Le graphique ci-dessous (figure 1) représente les temps de calcul d'un ordinateur faisant un milliard d'opérations par seconde sur différentes complexités.

(1) En vous appuyant sur la figure 1, déterminer combien de temps il faudrait pour exécuter un algorithme de complexité n^3 sur une entrée de taille $n = 1\,000$?

(2) Même chose pour $n = 10^6$ et $n = 10^9$.

(3) On considère un algorithme dont la complexité est de n^2. En vous appuyant encore sur la figure 1, évaluer l'ordre de grandeur de la taille maximale d'un problème qui peut être traité en moins d'une heure par un humain et par un ordinateur.

(4) Même chose pour des algorithmes de complexité $\log n$, n, n^3 et 2^n.

(5) (♣) Connaissez vous des algorithmes pour chacune de ces complexités ?

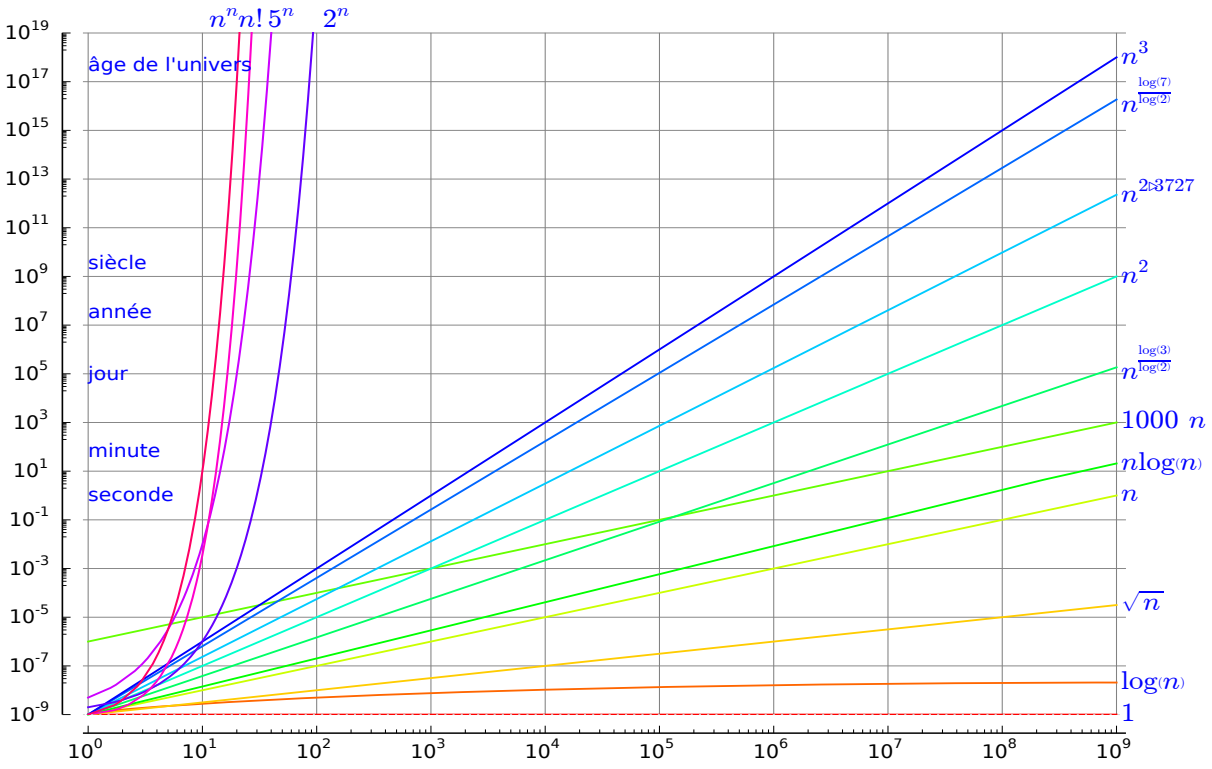

FIGURE 1. Quelques courbes de complexité : en abscisse n ; en ordonnée, le temps en secondes

Exercice 5.
Quelle est la complexité des algorithmes d'addition et de multiplication tels que vous les avez appris à l'école primaire ?

Exercice ♣ 6.
Rappeler les différentes implantations de la fonction `Fibonacci` vues lors des TDs précédents. Quelle est, dans chaque cas, la complexité en temps et en mémoire ?

Exercice 7.
Pour chacun des problèmes suivants, donner un algorithme, borner sa complexité, et essayer de déterminer s'il est optimal ou non :

(1) Compter les entiers plus grand que 10 dans un tableau ;

(2) Insérer un élément au début d'un tableau ;

(3) Tester si un nombre est premier ;

(4) Rechercher un élément dans un tableau trié ;

(5) Compter les entiers plus grand que 10 dans un tableau trié ;

(6) Calculer l'intersection de deux ensembles représentés par des tableaux d'entiers ;

(7) Calculer la puissance n-ième d'un nombre ;

(8) (♣) Calculer le pgcd de deux nombres ;