Complexité des algorithmes
M1 MIAGE – TD 1

1 Test d'égalité

Étudier la complexité, en nombre de comparaisons de bits (ou d'octets ou de caractères), d'algorithmes simples pour distinguer des :

1. objets, exactement *par leur adresse*,
2. objets ayant exactement les mêmes attributs,
3. objets ayant les mêmes attributs, selon leur propre méthode,
4. chaînes de caractères, et
5. séquences d'ADN (alphabet {`A`, `C`, `G`, `T`}), avec des « séquences équivalentes ».

2 Quelques algorithmes sur les nombres

1. Donner un algorithme (pseudo-code) pour calculer $n!$, la factorielle de n, à partir du nombre n. Donner sa complexité en fonction de n et en fonction du nombre de bits du nombre n, ce qui n'est pas la mẹme chose.
2. Donner un algorithme qui vérifie si un nombre n est premier. Donner sa complexité en fonction de n et du nombre de bits de n.

3 Grands entiers

La cryptographie utilise souvent des entiers dont les valeurs dépassent largement la capacité du format `int`.

1. Quelle structure de donnée pourrait-on utiliser pour stocker ces entiers ? Quelle serait la taille de la structure pour un entier valant n ? Inversement, pour une taille donnée t, quel est le plus grand entier que l'on pourrait y stocker ?
2. Proposer un algorithme pour additionner deux gros entiers. Combien d'*opérations élémentaires* faut-il faire en fonction de t ? de n ?
3. Que se passe-t-il pour la multiplication ? la division entière ? le modulo ?
4. Qu'en est-il en terme de place occupée ?

4 Autour des notations asymptotiques

1. Montrer que la relation $f(x) \in \mathcal{O}(g(x))$ est réflexive et transitive.
2. Montrer que la relation Θ est une relation d'équivalence.
3. Montrer que si $f \in \mathcal{O}(g)$ alors $a \cdot f \in \mathcal{O}(g)$ pour tout entier a strictement positif.
4. Montrer que si $f_1 \in \mathcal{O}(g_1)$ et $f_2 \in \mathcal{O}(g_2)$ alors $f_1 + f_2 \in \mathcal{O}(g_1 + g_2)$.
5. Montrer que si $f_1 \in \mathcal{O}(g_1)$ et $f_2 = \mathcal{O}(g_2)$ alors $f_1 + f_2 \in \mathcal{O}(\max(g_1, g_2))$.
6. Pourquoi peut-on écrire $\Theta(\log n)$ sans préciser la base du logarithme ?

5 Comparaison de croissances

1. Classer les complexités suivantes, de la plus petite à la plus grande : n^2, 2^n, $n \log n$, n, n^3, $\sqrt{n}$, $\log n$.
2. Trouver une valeur entière pour n telle que

$$1\,000\,000 \log_{10}(n) \leq n \ ; \leq n^2 \ ; \leq n^3 \ .$$

   Que se passe-t-il pour les valeurs supérieures ?
3. La première colonne indique le temps d'exécution en microsecondes d'un programme pour une entrée de taille n. Pour chaque colonne, donner le temps d'exécution —ordre de grandeur ou une valeur exacte— de ce programme pour une entrée de la taille donnée.

	5	**10**	**20**	**40**	**100**	**500**
$\log_2(n)$						
n	5	10	20	40	100	500
$n \log_2(n)$						
n^2						
n^3						
n^5						
2^n						
3^n						
n^n						
2^{2^n}						

   À l'aide de, par exemple, `gnuplot`[1], tracer les coures correspondantes. Sans ordinateur, essayer de donner leurs allures générales grâce aux points calculés puis réaliser une impression pour la prochaine fois.

Aide `gunplot`

(version 4.2)

Pour afficher la fonctions $x \to 2^x$ entre 1 et 100 :

```
plot [ x = 1 : 100]  2**x
```

Pour afficher plusieurs fonctions simultanément, utiliser la virgule pour les séparer :

```
plot [ t = 1 : 100] [ 0 : 400 ]  log(t) , t , t**3
```

Le premier crochet limite les valeurs de `t`, le second l'affichage des valeurs.

Pour tracer une courbe donnée par une suite de points dans un fichier :

```
plot 'carre.dat'  with  line
```

La clause `with` sert à dire comment dessiner : ici avec des lignes entre les points. On peut aussi utiliser des arguments tels que `smooth csplines` directement (voir ci-dessous).

On peut mixer fonctions et fichiers (entre `'` ou `"`). Si le fichier a plusieurs colonnes, on peut préciser lesquelles considérer avec une clause `using a:b`

Exemple de rendu complexe :

[1] Chargement sous ubuntu : paquets `gnuplot` et `gnuplot-x11`.

```
plot [x=1:5] [-10:100] sin(100*x)*x**x/10, 'ce.dat' using 1:3 with boxes, "ce.dat" using 1:4 smooth csplines
```

Pour passer en échelle logarithmique (conseillé pour les fonctions à tracer) :

```
set logscale xy
```

(utiliser `unset` pour revenir en échelle normale).

Sortie sur un fichier en `.gif` :

```
set terminal gif transparent medium size 448,336
set output "trace.gif"
replot
```

Pour revenir à un affichage à l'écran :

```
set terminal X11
```

Pour une très courte introduction :
`http://aristote.biophy.jussieu.fr/phynum/cours/logi/logi2.html`
et pour une documentation[2] (il y a aussi une aide en ligne !) :
`http://www.gnuplot.info/docs/gnuplot.html`

[2] `file:///usr/share/doc/gnuplot-doc/html/gnuplot.html` avec le paquet `gnuplot-doc`.