Feuille n°2

1) Comparaison

Classer ces fonctions selon leur vitesse de croissance. Les regrouper par ordre de croissance.

a. n^2,
b. 2^n,
c. $\ln(n)$,
d. $n - n^3 + 7n^5$,
e. $\log_2(n) + \sqrt{n}$,
f. $2^{\frac{n}{2}}$,
g. $n^2 + \log_2(n)$,
h. n^3,
i. $n \log_2(n)$,
j. $n^2 \log_2(n)$,
k. $n(\log_2(n))^2$,
l. $(1,23)^n + n^{1,23}$,
m. $n^{\frac{7}{4}} + n$,
n. $n^3 + n^2 \log_2^{10}(n)$, et
o. $\log_2(n) + \log_2(\log_2(n))$.

où ln représente le logarithme népérien (base e) et $\log_2$ représente le logarithme en base 2.

2) Boucles imbriquées

Calculer la complexité du programme ci-dessous en nombre d'appel de la fonction `f` en fonction de n dans les deux cas suivants :

```
for i = 1 to n-1 do
    for j = i+1 to n do
        for k = 1 to j do
            t[i,j,k] := f(t[i,j,k])
        end do
    end do
end do
```

```
for i = 1 to n do
    for j = 1 to i do
        for k = j to i do
            t[i,j,k] := f(t[i,j,k])
        end do
    end do
end do
```

3) Rechercher dans un tableau

On considère l'algorithme de recherche séquentielle d'un élément x dans un tableau $t[1..n]$. On fait l'hypothèse que l'élément est toujours dans le tableau avec une probabilité d'être dans la case i proportionnelle à i.

a) Calculer cette probabilité.

b) Calculer la complexité en moyenne de l'algorithme.

c) Reprendre les deux questions précédentes en considérant que cette probabilité n'est plus proportionnelle à i mais à $n + 1 - i$.

4) Étude expérimentale sur la suite de Fibonacci

Celle-ci se définit grâce à l'équation récursive suivante :

$$\begin{aligned} \mathcal{F}_0 &= 0 \ , \\ \mathcal{F}_1 &= 1 \ , \text{ et} \\ \mathcal{F}_{n+2} &= \mathcal{F}_{n+1} + \mathcal{F}_n \ . \end{aligned}$$

Il faut implanter en Java différents algorithmes pour la calculer, effectuer des mesures, tracer des courbes et essayer de trouver les ordres de croissances correspondants.

5) Programmation récursive.

À partir de quand ne peut-on plus le faire ?

Comparer la croissance à celle de la suite elle-même.

6) Implanter le calcul de cette suite avec une boucle simple (et deux variables pour avoir $\mathcal{F}_{n+1}$ et $\mathcal{F}_n$ en permanence). Étudier théoriquement et expérimentalement la complexité.

7) On connaît une expression close de la suite :

$$\mathcal{F}_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right) \ .$$

L'utiliser et comparer (erreur relative, erreur absolue) avec la valeur obtenue par la méthode précédente.

Jusqu'où peut-on aller ?

8) À partir de quel rang, la suite de Fibonacci dépasse-t-elle les capacités des `long` ?

Refaire toute l'étude avec des `BigInteger` pour pousser beaucoup plus loin l'expérimentation.

Complément `gnuplot`

Il est possible, non seulement de tracer des courbes à partir de valeurs contenues dans un fichier, mais également de les retraiter pour obtenir des courbes ! Avec `using`, on ajoute des parenthèses et on utilise `$i` pour accéder à la colonne i.

Exemple :

```
plot 'f.dat' using ($1):($2/$1) smooth csplines title "long / n"   \
   , 'f.dat' using ($1):($3/$1) smooth csplines title "BigInteger / n"  \
   , 'f.dat' using ($1):($3/($1*$1)) smooth csplines title "BigInteger / n2"
```

et le fichier `f.dat` suivant :

```
1        2       1        1       2.71
2        4       4        8       7.38
3        6       9       27      20.08
4        8      16       64      54.59
5       10      25      125     148.41
```