Complexité d'un algorithme

Trois questions à se poser quand on fabrique un algorithme :
– est-ce qu'il donne un résultat ? $\rightsquigarrow$ terminaison ;
– est-ce qu'il donne le/un bon résultat ? $\rightsquigarrow$ validité ;
– est-ce qu'il donne le résultat en temps raisonnable ? $\rightsquigarrow$ complexité.

1 Calculs des nombres de Fibonacci

On cherche à calculer les nombres de Fbonacci $(F_n)_{n\geq 1}$ définis par :

$$F_0 = F_1 = 1, \quad \forall n \in \mathbb{N},\ F_{n+2} = F_{n+1} + F_n.$$

1.1 Calcul récursif

La méthode récursive est élégante, mais on va voir qu'elle n'est pas très efficace. Voici le code correspondant :

Xcas

```
f(n) := {
    if (n<2) return 1 ;
    return f(n-1) + f(n-2) ;
}
```

1. Tester le calcul sur quelques valeurs.
2. L'instruction `time(f(n))` calcule le temps $t(n)$ mis pour calculer $f(n)$. Dessiner les points de coordonnées $(n, t(n))$ pour n variant dans un intervalle « intéressant ». Faire un ajustement exponentiel.
3. On note a_n le nombre d'appels à la fonction `f` effectués pendant le calcul de $f(n)$ et s_n le nombre d'additions nécessaires. Par exemple, on a : $a_0 = a_1 = 0$, $a_2 = 1$, $s_0 = s_1 = 0$, $s_1 = 1$.
   Donner des relations de récurrences déterminant les suites (a_n) et (s_n). Préciser et interpréter les mesures du temps de calcul.

1.2 Calcul par boucle simple

1. Proposer une méthode par itération simple pour calculer les nombres de Fibonacci. À quel temps de calcul s'attend-on ?
2. Implémenter, mesurer le temps de calcul, dessiner un nuage de points et vérifier ou infirmer l'hypothèse précédente.

1.3 Calcul matriciel

1. Pour n entier naturel, on considère la matrice-ligne $X_n = (F_{n+1} \quad F_n)$. Donner une formule de récurrence matricielle et retrouver ainsi la formule de Binet exprimant F_n en fonction de n et du nombre d'or.

2. En déduire une méthode bien plus efficace que les précédentes pour le calcul de (F_n) par le calcul d'une puissance de matrice (sans utiliser les opérations matricielles prédéfinies).

2 Algorithme de Gauss

Évaluer le nombre d'opérations (additions/soustractions et multiplications/divisions) que l'on effectue en résolvant un système linéaire à n équations et n inconnues générique (ayant une unique solution) avec l'algorithme de Gauss.
Pourquoi est-il raisonnable de négliger les comparaisons (pour trouver le plus grand pivot dans chaque colonne) et les permutations de lignes (pour placer le pivot à la bonne place) ?

3 Algorithme d'Euclide (2)

Rappeler l'algorithme d'Euclide (on suppose disposer des opérations arithmétiques et de la partie entière)... Évaluer, en fonction des valeurs des paramètres entrés, le nombre *maximal* de divisions euclidiennes qu'il faut effectuer.