

# Algorithmique Avancée - TD 1 : corrigé - 4 octobre 2011

Université Paris 7 — BioInformatique — M1

## 1 L'algorithme d'Euclide

1. On remarque que le deuxième tiret n'est appelé qu'une fois au plus et le premier tiret doit être suivi d'un appel au deuxième ou au troisième tiret. On constate que l'appel au troisième tiret fait baisser la quantité  $x + y$  qui ne peut devenir négative. Donc l'algorithme termine.
2. Soit  $p$  le PGCD. **Invariant** : tout nombre qui divise  $x$  et  $y$  à une étape divise  $x$  et  $y$  à l'étape suivante. Trivial pour cas 1, non à prouver pour cas 2 (terminal). Pour cas 3, si  $d$  est un tel nombre alors il existe  $x'$  et  $y'$  tels que  $x = dx'$  et  $y = dy'$  et comme  $x \geq y$  on a  $x - y = d(x' - y') \geq 0$ . On conclut en observant que  $PGCD(x, 0) = x$ .

3.

$$PGCD(x, y) = \begin{cases} PGCD(y, x) & \text{si } y > x \\ x & \text{si } y = 1 \\ PGCD(y, x \% y) & \text{sinon} \end{cases}$$

où “%” est le modulo (reste de la division entière).

4. G Lamé, mathématicien mort en 1870, a prouvé le théorème suivant : le nombre de divisions à faire est au plus la moitié du nombre de chiffres décimaux. En fait, le **pire des cas** est pour deux éléments consécutifs de la *suite de Fibonacci*. Par exemple  $PGCD(34, 21) = PGCD(21, 13) = PGCD(13, 8) = PGCD(8, 5) = PGCD(5, 3) = PGCD(3, 2) = PGCD(2, 1) = 1$ . Les preuves sont un peu “matheuses” et omises ici, voir par exemple [http://www.acrypta.com/telechargements/fichencrypto\\_112.pdf](http://www.acrypta.com/telechargements/fichencrypto_112.pdf)  
Aucune raison pour que cela soit optimal.

## 2 Des pièces et une balance

1. Deux pesées suffisent dans tous ces cas
2. La balance possède trois états : penche à gauche, penche à droite ou équilibré.  
Si on met deux tas  $A$  et  $B$  de poids différents sur les plateaux, on n'apprend rien. Par contre si  $|A| = |B|$  alors
  - Si la balance penche vers  $A$  la fausse pièce est dans  $B$
  - Si la balance penche vers  $B$  la fausse pièce est dans  $A$
  - Sinon (équilibre) la fausse pièce n'est ni en  $A$  ni en  $B$On ne peut avoir d'autre information ! Un algorithme efficace consiste à couper le tas de pièces en TROIS tas  $A$ ,  $B$  et  $C$  de même taille (très exactement deux tas  $A$  et  $B$  de taille  $\lceil n/3 \rceil$  et  $C$  le reste). Puis on recommence.
3. Si l'algorithme dit que la pièce  $P$  est fausse alors on vient de faire une pesée à 2 pièces seulement, on vérifie facilement les 3 cas possibles. Dans le sens inverse (montrer que si  $P$  est fausse alors l'algorithme sort bien  $P$ ) on remarque que l'invariant “la fausse pièce est dans l'un des trois tas  $A$ ,  $B$  ou  $C$  considérés” est maintenu.

4. Grâce à l'algorithme ci-dessus, on se ramène à un problème trois fois plus petit ( $n$  est divisé par 3 à chaque fois).  
Il faut  $\lceil \log_3(n) \rceil = O(\log n)$  pesées en tout ! 3 pour  $n \leq 27$  etc.
5. On ne peut faire moins : en effet il y a  $n$  possibilités. Pour tout algorithme qui fait  $k$  pesées il y a  $3^k$  exécutions possibles. Si  $3^k < n$  nécessairement deux possibilités tombent dans la même exécution de l'algorithme, qui n'arrive donc pas à les distinguer.
6. La première fois que la balance penche, vers  $X$  ( $X = A$  ou  $B$ ), on compare  $X$  avec autant de pièces du tas  $C$  (dont on sait donc qu'elles sont bonnes).
  - Si penche vers  $X$  alors la fausse pièce est plus lourde et est dans  $X$ .
  - Si penche vers  $C$  alors la fausse pièce est plus légère et est dans  $X$ .
  - sinon (équilibre) la fausse pièce est dans l'autre tas que  $X$  ( $B$  ou  $A$ ) et est plus légère
 Il faut  $1 + \lceil \log_3(n) \rceil$  pesées en tout. Notez que s'il n'y a que deux pièces on ne peut pas conclure ! (on a besoin du tas  $C$ ).
7. Méthode simple : on divise le problème en deux tas  $E$  et  $F$  de taille  $n/2$  (plus une pièce si  $n$  impair)
  - Si la balance penche vers  $E$  les deux fausses pièces sont dans  $F$  (ou la pièce de côté) → recommencer sur cet ensemble
  - Si la balance penche vers  $F$  les deux fausses pièces sont dans  $E$  (ou la pièce de côté) → recommencer sur cet ensemble
  - Sinon (équilibre) une fausse pièce dans chaque ensemble → on applique l'algorithme précédent (tiret 2) à chacun des deux ensembles.
 Le nombre de pesées est :  $x$  pesées binaires (avec  $x \leq \lceil \log_2(n) \rceil$ ) puis deux fois  $y$  pesées ternaires avec  $y = \lceil \log_3(\frac{n}{2^x}) \rceil$   
 Le pire cas est quand les deux pièces restent ensemble jusqu'au bout (car  $\log_2(n) > 2 \log_3(n)$ )  
 Par contre cet algorithme n'est pas optimal...

### 3 Autour de $X^n$

1. Calculer  $X_2 = X.X$ ,  $X_3 = X_2.X \dots X_n = X_{n-1}.X$ . Comme  $X_i = X^i$  c'est OK.  $n$  multiplications.
2. Calculer  $X_2 = X.X$ ,  $X_3 = X_2.X_2 \dots X_a = X_{a-1}.X_{a-1}$ . Comme  $X_i = X^{2^i}$  c'est OK.  $a$  multiplications.
3. Formulation 1 : soit  $k = n/2$  (quotient de la division). Si  $n$  pair  $X^n = X^k.X^k$  sinon  $X^n = X^k.X^k.X$ . A chaque fois on divise  $n$  en deux →  $\log_2(n)$  étapes → au plus  $2 \log_2(n)$  multiplications.  
 Formulation 2 :
 

```

A = 1 // NB code X^k
B = X // NB code X^{2^a}
Tant que n > 0
  si n impair
    A = A.B
  B = B.B
  n = n / 2
Fin tant que
Retourner A
      
```
4. Invariant 1 :  $B = X^{2^i}$  à l'étape  $i$ .  
 Invariant 2 :  $X^{n_0} = X^n.A$  où  $n_0$  est le  $n$  au début de l'algo et  $n$  le  $n$  courant. Comme  $n$  diminue et on termine à  $n = 0$  cela prouve l'algorithme : à la fin  $X^{n_0} = A$ .
5. Pour  $X^{15}$  l'algorithme calcule  $X.X^2.X^4.X^8$  en 6 multiplications. Mais on peut le faire en 4 multiplications seulement :  
 $X^2 = X.X$  ;  $X^3 = X.X^2$  ;  $X^5 = X^3.X^2$  ;  $X^{15} = X^3.X^5$ .  
 Il existe un algorithme qui calcule la décomposition optimale (hors sujet ici !). Une borne inférieure du nombre de multiplications est  $\lceil \log_2(n) \rceil$ , et notre algorithme fait au plus le double, ce qui est une *garantie de performance*.

