

Algorithmique Avancée - TD 1 - 4 octobre 2011

Université Paris 7 — BioInformatique — M1

Prouver un algorithme

La *preuve* d'un algorithme vise à montrer deux choses

1. L'algorithme **calcule ce qu'on veut**. En général on fait une *preuve par invariant* : on trouve une propriété qui
 - est vraie initialement
 - reste vraie d'une étape à l'autre de l'algorithme
 - permet de prouver, à la fin, que le résultat est correct
2. Il **le fait dans le temps qu'on veut**. Cela peut vouloir dire :
 - Il le fait en un temps fini (preuve de **terminaison**) - nécessaire : un "algorithme" qui ne termine pas toujours n'est pas correct. Cela se fait en général en exhibant une quantité entière qui décroît et ne peut devenir négative.
 - Ou, mieux, on peut borner le temps d'exécution en fonction de la taille des données (analyse de complexité **au pire** des cas) — facultatif — implique la terminaison
 - Ou borner son temps d'exécution **en moyenne** pour des données prises dans un certain ensemble. Facultatif et plus dur qu'au pire
 - Parfois (et là c'est souvent très dur) on peut prouver l'**optimalité** : impossible de faire un temps plus court

1 L'algorithme d'Euclide

C'est le plus vieil algorithme connu qui est écrit correctement, c'est à dire sous forme d'une fonction récursive et non d'une "recette de cuisine". Son auteur est mort en -265 d'après Wikipedia.

On rappelle que le PGCD de deux nombres x et y est le plus grand nombre qui divise x et y . L'algorithme est :

$$PGCD(x, y) = \begin{cases} PGCD(y, x) & \text{si } y > x \\ x & \text{si } y = 0 \\ PGCD(x - y, y) & \text{sinon} \end{cases}$$

1. Montrez que l'algorithme termine
2. Montrer par invariant que l'algorithme est correct.
3. Pouvez-vous l'améliorer pour aller plus vite ?
4. Que dire de sa complexité au pire ? Est-ce optimal ?

2 Des pièces et une balance

On a n pièces. On sait que l'une d'entre elle est fautive (elle est plus légère que les autres). On dispose d'une balance à plateaux.

1. Combien de pesées faut-il faire pour trouver la fautive parmi 4 pièces ? 8 ? 9 ?

2. Proposez un algorithme résolvant le problème
3. Montrez sa correction
4. Donnez sa complexité
5. Prouvez son optimalité
6. Combien de pesées ajouter si on ne sait pas si la pièce fautive est plus lourde ou plus légère ?
7. Même questions si on sait que deux pièces sont fautes (et plus légères).

3 Autour de X^n

Étant donnés deux entiers positifs non nuls X et n , on recherche le moyen le plus rapide de trouver X^n .

1. Donner un algorithme de complexité $O(n)$ (en nombre de multiplications) qui prend en argument X et n et qui retourne X^n .
2. Donner un algorithme de calcul de X^{2^a} . Quelle est sa complexité (en nombre de multiplications) ?
3. Donner un algorithme de calcul de X^n de complexité $O(\log(n))$. *Indice : s'aider de la représentation binaire de n .*
4. Prouvez votre algorithme par invariant
5. Montrez grâce au calcul de X^{15} que votre algorithme n'est pas optimal

4 Tas de crêpes

On dispose d'une pile de n crêpes dont les diamètres sont tous différents. Le problème consiste à trier la pile de manière à obtenir la crêpe la plus large en bas de la pile et la crêpe la moins large en haut de la pile. Pour éviter de trop manipuler les crêpes, on ne s'autorise qu'une seule opération : retourner les p premières crêpes du haut de la pile ($1 \leq p \leq n$ et p peut changer d'un retournement à l'autre).

1. Proposez un algorithme qui trie les crêpes en au plus kn retournements. Avec k petit !
2. On suppose maintenant que les crêpes ont une face brûlée et une face non brûlée. Modifiez l'algorithme ci-dessus pour que les crêpes soient triées et aient toutes leur face brûlée vers le bas de la pile. Évaluez le nombre de retournements de votre algorithme.

5 La fonction d'Ackerman

Soit $A_i(a, b)$ une fonction $\mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ définie par

$$A_i(a, b) = \begin{cases} b + 1 & \text{si } i = 0 \\ a & \text{si } b = 0 \text{ et } i = 1, \text{ ou si } b = 1 \text{ et } i > 1 \\ A_{i-1}(a, A_i(a, b-1)) & \text{sinon} \end{cases}$$

1. 0- Identifiez et donnez un nom à la fonction $A_0(a, b)$
 - 1- Identifiez et donnez un nom à la fonction $A_1(a, b)$
 - 2- Identifiez et donnez un nom à la fonction $A_2(a, b)$
 - 3- Identifiez et donnez un nom à la fonction $A_3(a, b)$
 - 4- Identifiez et donnez un nom à la fonction $A_4(a, b)$
2. Combien de chiffre possède $A_0(10, 10)$? $A_1(10, 10)$? $A_2(10, 10)$? $A_3(10, 10)$? $A_4(10, 10)$? $A_5(10, 10)$?
3. Combien d'appels récursif fait $A_i(a, b)$? D'après la question précédente, pour quelles valeurs un ordinateur *échouera* certainement à faire le calcul ?