Algorithmique et Structures de Données

Master Biologie structurale, bioinformatique et biotechnologies

Contôle continu 2

Année 2017 - 2018

Jean-Michel Dischler

1. Plus grande sous-chaîne commune

1.1 Ecrire un algorithme prenant en entrée deux chaînes de caractères et renvoyant la sous-chaîne commune la plus grande :

largestcommon(s1 : String, s2 : String) -> String

On ne considère que des lettres majuscules. Par exemple pour les deux chaînes suivantes : « ABAB » et « BABA », la fonction renvoie : « ABA » (ou « BAB » selon l'algorithme). Ecrire un algorithme naïf consistant à produire toutes les sous-chaînes possibles de *s1* et à vérifier si elle existe dans *s2*. Quelle est la complexité de votre algorithme ?

1.2 On se propose maintenant d'utiliser la formulation suivante :

largestcommon(s1, s2) = s1[i..j] tel que j = position du MAX de plsuff(s1[1..p], s2[1..q]) pour tout p,q.

Ici on suppose que les indices comment à 1. La fonction *plsuff* calcule **la longueur** de toutes les sous-chaines communes possibles. Elle est définie récursivement par :

plsuff(s1[1:p], s2[1:q]) = si s1[p]=s2[q] alors renvoyer 1+ plsuff(s1[1:p-1], s2[1:q-1])
sinon renvoyer 0

Avec l'exemple précédent « ABAB » et « BABA », le tableau suivant représente tous les valeurs possibles de *plsuff* (les lignes représentent *q*, les colonnes *p*) :

		A	**B**	**A**	**B**
	0	0	0	0	0
B	0	0	1	0	1
A	0	1	0	2	0
B	0	0	2	0	3
A	0	1	0	3	0

On voit que le maximum est obtenu pour la valeur 3 : notez, qu'il y a deux fois la valeur 3 dans ce tableau. La position du maximum dans le tableau indique par ailleurs la dernière lettre respective de la sous-chaîne dans les deux chaînes d'entrée *s1* (horizontale) et *s2* (verticale).

Ecrire un algorithme qui calcule toutes les valeurs possibles de *plsuff*. Il n'est pas nécessaire que l'algorithme soit récursif ! On peut remplir directement un tableau 2D (comme dans l'illustration), ce qui correspondra à de la *programmation dynamique*. Mais attention à l'ordre de parcours du tableau. Utiliser ce résultat pour réécrire la fonction *largestcommon*.

Quelle est la complexité ?

2. Calculer une racine carrée.

On considère la suite définie par récurrence :

$$\begin{cases} x_0 = a/2 \\ x_{n+1} = \dfrac{x_n + \dfrac{a}{x_n}}{2} \end{cases}$$

où *a* est un nombre flottant. Ecrire une fonction prenant en paramètres *a* et *n* et permettant de calculer le nième therme de cette suite (càd renvoyant x_n) en utilisant une récursivité simple (un seul appel à x_{n-1}).

On peut montrer que cette suite converge vers la limite : $\sqrt{a}$

Modifier la fonction de façon à ne plus avoir de paramètre entier *n*. Le paramètre *n* est remplacé par un paramètre réel ε, de sorte que la fonction renvoie x_n ssi $| x_n - x_{n-1} | < \varepsilon$. Le paramètre ε permet en fait de contrôler la précision du calcul.