MAPLE / 0

Problèmes d'introduction

Exercice 1 . CALCUL FORMEL ET NUMÉRIQUE, ÉTUDE D'UNE SUITE –

Ecrire deux fonctions calculant le n-ème terme de la suite définie par

$$\begin{cases} u_0 &= 11/2, \\ u_1 &= 61/11, \\ u_{n+1} &= 111 - 1130/u_n + 3000/(u_n u_{n-1}), \quad \text{pour } n \geqslant 1. \end{cases}$$

La première par un calcul exact ; la deuxième en calculant avec d chiffres significatifs. Comparer les résultats obtenus avec $d = 10$, 20, 30 et pour n variant entre 5 et 40 par exemple. Que donne l'utilisation d'une fonction récursive ? [*utiliser* `time` *pour chronométrer*]. Quelle est la limite de la suite (u_n) ? [*pour chercher des points fixes utiliser* `solve`*, ou* `fsolve` *pour une solution approchée*]

Exercice 2 . LISTE DE NOMBRES PREMIERS (crible d'Erathostène) –

Construire la liste $\mathcal{P}(n)$ des nombres premiers inférieurs à n.

1) en utilisant `isprime` (dans un premier temps, se passer de `select`, l'utiliser ensuite).

2) en criblant la suite des entiers x, $2 \leqslant x \leqslant n$: soit k le premier entier non criblé (initialement $k = 2$) ; alors on rajoute k à la liste et on supprime dans la suite tous les multiples de k. Montrer que cette procédure produit bien la liste voulue [*En pratique, remplir de marqueurs un* `array` *de taille fixe plutôt que des listes de tailles variables dont on supprime les éléments*]. Comparer avec la méthode précédente.

Exercice 3 . NOMBRES DE FIBONACCI –

Soit la suite de Fibonacci définie par $F_0 = 1$, $F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$ pour $n \geqslant 0$.

1) Montrer que $F_{2n} = F_n^2 + F_{n-1}^2$ et $F_{2n-1} = 2F_nF_{n-1} - F_{n-1}^2$ [*par récurrence.* MAPLE *peut faire les calculs !*]

2) Ecrire une procédure récursive de calcul de F_n utilisant ces formules [*on pourra utiliser* `irem` *ou* `type(n,even)`]. Comparer avec celle obtenue à partir de la relation de définition.

3) On pose $\mathcal{F}_n := \begin{pmatrix} F_{n+2} & F_{n+1} \\ F_{n+1} & F_n \end{pmatrix}$ et $A := \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$. Montrer que $\mathcal{F}_{n+1} = A\mathcal{F}_n$. En déduire une expression de $\mathcal{F}_n$ en fonction de A [*introduire* $\mathcal{F}_{-1}$]. En déduire que $\mathcal{F}_{2n} = \mathcal{F}_{n-1}^2$, puis les formules de la question précédente. Ecrire une fonction calculant F_n utilisant l'exponentiation matricielle [*utiliser* `matrix`*. La fonction* `evalm` *peut être utile pour visualiser certains résultats*]

4) Trouver des relations de récurrence dépendant de la classe de n modulo 3 [*calculer* $\mathcal{F}_{3n+1}$ *en fonction de* $\mathcal{F}_{n-1}$]. Ecrire la procédure correspondante et comparer les temps de calcul.

★ **5)** Montrer que $F_n = a\phi^n + b(-1/\phi)^n$, ou ϕ est le nombre d'or, c'est-à-dire la racine > 1 de l'équation caractéristique $x^2 = x + 1$ [*diagonaliser* A *ou utiliser un résultat général sur les suites définies par récurrence linéaire*]. Calculer a et b. En déduire que $F_n = \lfloor a\phi^n \rceil$, ou $\lfloor x \rceil$ désigne l'entier le plus proche de x [`round` *en* MAPLE]. Ecrire la fonction correspondante et comparer.

★ **Exercice 4** . DIVISER POUR RÉGNER –

Ecrire une fonction calculant $n!$ (sans utiliser `n !`).

1) Que donne l'utilisation d'une procédure récursive ? itérative ?

2) Ecrire une fonction qui fait le produit des éléments d'une liste $[x_1, \ldots, x_n]$ en calculant d'abord les produits 2 à 2 : x_1x_2, x_3x_4, ... puis en itérant cette opération. [*Ce type d'idée (décomposer un problème de taille n en sous-problèmes de taille inférieure, puis itérer) est baptisé "diviser pour régner" (divide and conquer). La fonction correspondante est souvent récursive ; pas la peine ici*].

3) Modifier la procédure précédente pour qu'elle fasse les multiplications dans l'ordre x_1x_n, x_2x_{n-1}, ...

4) Appliquer a $n!$ et comparer avec la fonction MAPLE `n !`

5) Soient deux entiers en base x,

$$A = \sum_{i=0}^{n-1} a_i x^i \quad \text{et} \quad B = \sum_{i=0}^{n-1} b_i x^i$$

d'au plus n chiffres. Combien de multiplications de chiffres requiert le calcul (naïf) $A \times B$? Supposons maintenant que $n = 2$ et écrivons

$$AB = (a_1x + a_0)(b_1x + b_0) = \alpha_1 x^2 + ((a_0 + b_0)(a_1 + b_1) - \alpha_1 - \alpha_0)x + \alpha_0,$$

avec $\alpha_i := a_ib_i$ soit 3 multiplications élémentaires (et quelques additions et retenues) au lieu de 4. Montrer qu'en itérant ce procédé de type "diviser pour régner", on calcule AB en $O(n^{\log_2 3})$ multiplications [*on pourra dans un premier temps supposer que* $n = 2^k$]. Expliquer l'intérêt pour le calcul de 2 en 2 de $n!$

Note : Ce procédé est dû à Karatsuba (1963). Il est aussi valide pour calculer des produits de polynômes sur un anneau commutatif (c'est même plus simple : pas de retenues). Des généralisations de ces techniques, appelées FFT [1], permettent de calculer le produit de deux polynômes de degré n en $O(n \log n)$ opérations élémentaires, donc en temps essentiellement linéaire [*on ne peut pas faire mieux que* $n+1$ *puisqu'il faut au*

[1] car inspirées de la "Fast Fourier Transform" ou transformée de Fourier (discrète) rapide. Intuitivement, on remplace les polynômes par leurs valeurs en $2n+1$ racines de l'unité (calculées en divisant pour régner), on multiplie ces valeurs, et on reconstruit le produit par interpolation de Lagrange.

moins lire les coefficients]. Diverses astuces, en particulier dues à Schönage et Strassen, permettent d'appliquer ce résultat aux multiplications d'entiers de n chiffres, en $O(n \log n \log \log n)$ opérations. C'est un résultat asymptotique ! Ces méthodes ne sont utiles que pour des n élevés [*de l'ordre de 2000 (FFT), resp. 150 (Karatsuba)*].