info803 : informatique
Cours / TD : programmation dynamique

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22
téléphone : 04 79 75 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/

Exercice 1 : nombres de Fibonacci

Avec un manque d'originalité flagrant, on définit la suite suivante

$$F_0 = 0 \qquad F_1 = 1 \qquad F_{n+2} = F_{n+1} + F_n$$

F_n est le n-ème nombre de Fibonacci.

Question 1 : écrivez un petit programme récursif naïf pour calculer F_n. Donner une relation de récurrence donnant le nombre d'additions A_n utilisées par cet algorithme pour le calcul de F_n.

Question 2 : calculez les 11 premières valeurs de F_n et de A_n ; conjecturez et prouvez ce qui ressort de cette expérimentation.

Question 3 : en sachant que l'on a la formule exacte

$$F_n \quad = \quad \left\lfloor \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n + \frac{1}{2} \right\rfloor$$

que concluez vous sur le temps de calcul de F_n utilisant l'algorithme naïf ? Peut-on espérer un jour obtenir la valeur de F_{100} ?

Question 4 : en re-regardant la question 2, estimez le nombre d'additions que vous avez fait lors du calcul de F_{10}. Déduisez-en un algorithme itératif pour le calcul de F_n et estimez sa complexité (nombre d'additions effectuées).

Laïus... Ceci est un exemple trivial d'utilisation de la *programmation dynamique* : on cherche à résoudre un problème, et pour faire ça il nous faut résoudre de nombreux sous-problèmes qui ne sont pas indépendants. Nous allons donc garder en mémoire le résultat de chaque sous problème pour n'avoir à le calculer qu'une seule fois. Dans le cas de Fibonacci, on peut faire le calcul en partant du bas, mais il est parfois nécessaire de garder l'ordre naturel de calcul du problème. Il faut alors stocker les valeurs dans un tableau et remplacer les appels à la fonction par un test :

```
si la valeur existe déjà, utiliser la valeur stockée
sinon, appliquer la définition récursive
```

Exercice 2 : l'exemple classique, la multiplication d'un chaîne de matrices

Le but est d'essayer de trouver la manière optimale de multiplier n matrices de tailles compatibles entre elles.

Question 1 : en supposant que l'algorithme utilisé pour multiplier deux matrices est l'algorithme naïf, donnez le nombre de multiplications scalaires nécessaire pour multiplier une matrice $p \times q$ par une matrice $q \times r$.

Question 2 : on veut calculer le produit

$$M_1 \times M_2 \times M_3$$

où les tailles respectives des matrices sont 300×10, 10×400 et 400×15. Combien faut-il de multiplications pour calculer le produit ?

Pour la suite, on considère une suite de matrices $M_1 \times \ldots \times M_n$ où la taille de M_i est $t_{i-1} \times t_i$.

Question 3 : le nombre de parenthésages correct sur une chaîne de $n+1$ matrices est égal au nombre de Catalan c_n :

$$\frac{(2n)!}{n!(n+1)!}$$

En utilisant la formule de Stirling, donnez un équivalent asymptotique (un Ω) de ce nombre. Est-il envisageable de tester tous les parenthésages possibles ?

> Remarque : prouver que les nombres de Catalan sont bien la solution à notre problème de parenthésage est un exercice non trivial intéressant...

Question 4 : le problème de la multiplication d'une chaîne de matrices revient donc à trouver un parenthésage optimal. Montrez que si

$$(M_1 \times \ldots \times M_i) \times (M_{i+1} \times \ldots \times M_n)$$

est un parenthésage optimal, alors $M_1 \times \ldots \times M_i$ et $M_{i+1} \times \ldots \times M_n$ sont nécessairement des parenthésages optimaux.

Déduisez-en une relation de récurrence sur le nombre optimal de multiplications nécessaires pour calculer $M_i \times \ldots \times M_j$, et donnez une ébauche d'algorithme qui permet de calculer ce nombre. Estimez la complexité de votre algorithme...
Utilisez cet algorithme (ou un autre) pour calculer le nombre de multiplications optimal pour le calcul du produit de 5 matrices avec les tailles

$$t_0 = 3 \quad t_1 = 4 \quad t_2 = 2 \quad t_3 = 1 \quad t_4 = 2 \quad t_5 = 5$$

Laïus : remarquez que la généralisation de u_n à $u_{i,j}$ est nécessaire.

Question 5 : dans votre algorithme, rajoutez une matrice qui garde en mémoire l'indice k pour lequel le min de la définition est obtenu. Utilisez ce tableau supplémentaire pour trouver un parenthésage optimal.