Informatique tronc commun
Exercices d'algorithmique

Sujet

25 novembre 2005

1 Calculs récursifs : quelques fonctions

Donner une version récursive simple d'un algorithme calculant la quantité voulue dans chaque cas suivant. On justifiera l'arrêt et la validité de l'algorithme, puis on calculera sa complexité.

1. Calcul de la factorielle $n!$
2. Calcul de la dérivée d'ordre n :$f^{(n)}$
3. Calcul du terme d'ordre n de la suite u définie par : $u_0 = a$ et $\forall n \geq 1,\ u_n = f(u_{n-1})$
4. Calcul de la somme des n premiers termes d'un tableau T.
5. Calcul de la somme des n premiers termes d'une liste L.
6. Calcul de l'évaluation d'un polynôme P en x :$P(x)$, dans chaque cas suivant :
   - P est représenté par le tableau de ses coefficients.
   - P est représenté par la liste des couples (puissance, coefficient). Dans ce dernier cas, $P(x) = 2x + 5x^6$ est représenté par `L=[{p=1 ;c=2} ;{p=6 ;c=5}]`

2 Itération, récursion : calcul de terme de suite

Donner une version itérative puis récursive multiple puis récursive avec stockage d'un algrithme calculant le terme d'ordre n de la suite u définie par $u_0 = a$, $u_1 = b$, et $\forall n \geq 2,\ u_n = f(u_{n-1}, u_{n-2})$. On justifiera l'arrêt et la validité de l'algorihtme, puis on calculera sa complexité dans chaque version.

3 Récursion simple, diviser pour régner

Donner une version itérative puis récursive simple, puis diviser-pour régner d'un algorithme calculant la quantité voulue dans chaque cas suivant. On justifiera l'arrêt et la validité de l'algorithme, puis on calculera sa complexité.

- Calcul de x^n avec x réel.
- Calcul de $n * x$ avec x réel.

Le *Diviser pour régner*[a]
Ici, l'on coupe le problème en deux (en général) parties ayant à peu près la même taille (en général). Cela donne souvent une bonne complexité : l'exemple canonique est celui où les algorithmes de division et de fusion sont linéaires, on obtient alors du $\Theta(n \ln n)$. En effet, au $k^{\text{ème}}$ niveau de récursion, on travaille sur 2^k problèmes de taille $n/2^k$, au total cela fait du $\Theta(n)$ sur ce niveau. Or il y a $\log_2 n$ niveaux, donc au total on obtient bien $\Theta(n \ln n)$.
Exemple : tri-fusion. On coupe le tableau en deux morceaux de même taille (à 1 près), on trie les deux morceaux en appliquant la même méthode, puis l'on combine les résultats via la procédure `fus` de la section précédente. Cela donne un tri en $\Theta(n \ln n)$.

[a] Le diviser pour régner est censé être hors programme, mais il est plusieurs fois mentionné des « exemples de tris récursifs », ce qui fait allusion au tri rapide et au tri fusion qui sont des diviser pour régner.

4 Produits avancés

Donner une version diviser-pour régner d'un algorithme calculant la quantité voulue dans chaque cas suivant. on justifiera l'arrêt et la validité de l'algorithme, puis on calculera sa complexité.

- Calcul du produit de deux matrices carrées de taille 2^n, représentées chacune par un tableau.
- Calcul du produit de polynômes de degré 2^n, représentées chacun par un tableau.

5 Récursion mutuelle

Soient u et v les deux suites définies par : $\begin{cases} u_0 = a \\ v_0 = b \end{cases}$ et $\forall n \geq 1$ $\begin{cases} u_n = u_{n-1}^2 + 2v_{n-1} \\ v_n = v_{n-1} + 3u_{n-1} \end{cases}$

- Programmer les deux fonctions récursives `calcul`$_u$`(a,b,n)` et `calcul`$_v$`(a,b,n)` qui renvoient u_n et v_n.
- Claculer leur compelxité en nombre d'appels récursifs.

6 Translation de polynômes

Soit P un polynôme à coefficients entiers représenté par le tableau p de ses coefficients. Soit a un entier. En vous inspirant de l'algorithme de Horner, écrire une fonction `translate(p,a)` qui renvoie le tableau des coefficients du polynôme Q tel que $\forall x,\ Q(x) = P(x + a)$.

7 Décodage de nombres entiers

On représente un entier n par une chaîne de caractères. Par exemple, $n = 125$ est représenté par `s="125"`.

- Écrire une fonction `valeur(s)` qui renvoie l'entier associé à `s`.
- Écrire une fonction `compare(s,t)` qui renvoie la valeur `true` quand `s` $\geq$ `t`.
  On ne calculera pas les entiers associés à `s` et `t` car cette démarche est caduque si les entiers manipulés sont trop grands pour le type entier.

8 Suite de Fibonnaci

La suite de Fibonnaci est définie par $f_0 = f_1 = 1$ et $\forall n \geq 2,\ f_n = f_{n-1} + f_{n-1}$.

- Écrire une fonction récursive calculant f_n et évaluer sa complexité.
- Pour améliorer la complexité, écrire une fonction récursive calculant (f_{n-1}, f_n) et évaluer sa complexité.

– Montrer que pour $n, p \geq 1$ on a $f_{n+p} = f_n f_p + f_{p-1} f_{n-1}$. En déduire un algorithme de calcul de f_n selon la méthode diviser pour régner.

9 Une fonction mystérieuse

Soit `f` la fonciton définie par :

```
f := proc(x)
  if (x<=1) then 1
  else
    if (x mod 2) = 0 then
      2*f(x/2) ;
    else
      1+f(x+1) ;
    fi ;
  fi ;
end ; ;
```

– Montrer que cette fonction est bien définie pour tout entier naturel x.
– Montrer que `f(x)+x` est uen puissance de 2.
– Pouvez-vous décrire la fonction `f` ?

10 Calcul du reste et du quotient dans la division euclidienne

Soient n et p deux entiers naturels avec $p \geq 1$? on veut calculer le quaotient q et le reste r dans la division euclidienne de n par p. On introduit l'environnement Env $= a; b :$ `int` et la propriété $H =$"$a \geq 0$ et $b \geq 0$ et $n = ap + b$".

– Préciser une valeur initiale des variables a,b pour que H soit vraie.
– On suppose H vraie et $b \geq p$. Comment faut-il modifier a et b pour que H reste vraie et pour se *rapprocher* de la condition précédente.
– En déduire une écriture itérative puis récursive multiple de la fonction `DE(n,p)`. Justifier son arrêt et sa validité. Cette fonction devra renvoyer le couple `(q,r)`.

11 Calcul du PGCD

Soient a et b deux entiers naturels avec $a \geq b$ On veut écrire une fonction entière `pgcd(a,b)` qui renvoie le PGCD de a et b. On impose une contrainte : utiliser la fonction `n mod p` qui renvoie le reste dans la division euclidienne de `n` par `p`.

On introduit l'environnement suivant : Env $= n, p :$ `int`, et la propriété $H =$"$n \geq p \geq 0$ et $n \wedge p = a \wedge b$".

– Quelle valeur suffit-il de donner initialement à n et p pour avoir H vraie ?
– On suppose $p = 0$. Quel est, dans chaque cas, le PGCD de n et p ?
– On suppose que p est un entier strictement positif.
  On sait que si $n \geq p$ alors $n \wedge p = p \wedge (n \bmod p)$. en déduire une modification de l'environnement permettant de se *rapprocher* de la condition $p = 0$ et laissant la propriété H invariante.
– En déduire une écriture itérative puis récursive multiple de la fonction `pgcd(a,b)`. Justifier de son arrêt et de sa validité.
– On étudie dans cette question la version récursive. On introduit la suite de Fibonnaci (f_n) définie à la question 8.
  – On suppose que $a \geq b > 0$ Établir que si `pgcd(a,b)` fait n appels récursifs avec $n \geq 1$ alors $a \geq f_n$ et $b \geq f_{n-1}$.
  – Montrer que $f_n \geq (\frac{1+\sqrt{5}}{2})^{n-1}$
  – En déduire la complexité de la version récursive est en $O(\ln b)$. Que dire de la complexité de la version itérative ?