informatique

DUT INFO 2 / 2015-2016

Mis à jour le 24 janvier 2016 à 22:26

Compléments de mathématiques

Guillaume CONNAN

IUT NANTES Université de Nantes

UNIVERSITÉ DE NANTES

CHAPITRE 1

Complexité

THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

Recherche 1 - 1 CCP MP 2015

1. Donnez la décomposition en binaire (base 2) de l'entier 21.

   On considère la fonction `mystere` suivante :

```
def mystere(n, b) :
    """Données : n > 0 un entier et b > 0 un entier
       Résultat : ......."""
    t = [] # tableau vide
    while n > 0 :
        c = n % b
        t.append(c)
        n = n // b
    return t
```

   On rappelle que la méthode append rajoute un élément en fin de liste. Si l'on choisit par exemple `t = [4,5,6]`, alors, après avoir exécuté **`t.append(12)`**, la liste a pour valeur `[4,5,6,12]`.

   Pour $k \in \mathbb{N}^*$, on note c_k, t_k et n_k les valeurs prises par les variables `c`, `t` et `n` à la sortie de la k-ème itération de la boucle « while ».

2. Soit $n > 0$ un entier. On exécute **`mystere(n,10)`**. On pose $n_0 = n$.

   i. Justifier la terminaison de la boucle `while`.

   ii. On note p le nombre d'itérations lors de l'exécution de **`mystere(n, 10)`**. Justifiez que pour tout $k \in [\![0, p]\!]$, on a $n_k \leqslant \frac{n}{10^k}$. Déduisez-en une majoration de p en fonction de n.

3. En vous aidant du script de la fonction **`mystere`**, écrivez une fonction **`somme_chiffres`** qui prend en argument un entier naturel et renvoie la somme de ses chiffres. Par exemple, **`somme_chiffres(256)`** devra renvoyer 13.

4. Écrivez une version récursive de la fonction **`somme_chiffres`** que vous nommerez **`somme_rec`**. Est-elle plus efficace ?

5. Écrivez une version en Haskell de cette fonction : est-elle plus efficace ?

Recherche 1 - 2 Maths II CCP MP 2015 : sujet 0

1. Écrire une fonction factorielle qui prend en argument un entier naturel n et renvoie $n!$ (on n'accepetra pas bien sûr de réponse utilisant la propre fonction factorielle du module math de Python ou Scilab).

2. Écrire une fonction seuil qui prend en argument un entier M et renvoie le plus petit entier naturel n tel que $n! > M$.

3. Écrire une fonction booléenne nommée est_divisible, qui pend en argument un entier naturel n et renvoie True si $n!$ est divisible par $n+1$ et False sinon.

4. On considère la fonction suivante nommée mystere :

```
def mystere(n):
    s = 0
    for k in range(1,n+1):
        s = s + factorielle(k)
    return s
```

   i. Quelle valeur renvoie mystere(4) ?

   ii. Déterminer le nombre de multiplications qu'effectue mystere(n).

   iii. Proposer une amélioration du script de la fonction mystere afin d'obtenir une complexité linéaire.

Recherche 1 - 3

Donnez l'ordre de complexité temporelle des fragments de code Java suivants :

```
int sum = 0;
for (int n = N; n > 0; n /= 2)
   for (int i = 0; i < n; i++)
      sum++;
```

```
int sum = 0;
for (int i = 1; i < N; i *= 2)
   for(int j = 0; j < i; j++)
      sum++;
```

```
int sum = 0;
for (int i = 1; i < N; i *= 2)
   for (int j = 0; j < N; j++)
      sum++;
```

Recherche 1 - 4 Tri selectif

On parcourt la liste, on cherche le plus grand et on l'échange avec l'élément le plus à droite et on recommence avec la liste privée du plus grand élément.
On commence par chercher l'indice du maximum d'une liste. On part de 0 et on compare à chaque élément de la liste en faisant évoluer l'indice du maximum si nécessaire.

```
def ind_maxi(xs):
    ind_tmp = 0
    n       = len(xs)
    for i in range(n):
        if xs[i] > xs[ind_tmp]:
            ind_tmp = i
    return ind_tmp
```

Ensuite, on copie la liste donnée en argument pour ne pas l'écraser. On parcourt la liste et on procède aux échanges éventuels entre le maximum et l'élément de droite.

```
def tri_select(xs):
    cs = xs.copy()
    n  = len(cs)
    for i in range(n - 1,0,-1):
        i_m = ind_maxi(cs[:i + 1])
        if i_m != i:
            cs[i_m],cs[i] = cs[i],cs[i_m]
        # print(cs) : pour suivre l'évolution
    return cs
```

On va utiliser `permutation` de la bibliothèque `numpy.random` qui renvoie une permutation uniformément choisie par les permutations de $\mathfrak{S}_n$.

```
In [5]: ls = list(permutation(range(10)))

In [6]: ls
Out[6]: [8, 0, 4, 3, 5, 2, 7, 1, 9, 6]

In [7]: tri_select(ls)
[8, 0, 4, 3, 5, 2, 7, 1, 6, 9]
[6, 0, 4, 3, 5, 2, 7, 1, 8, 9]
[6, 0, 4, 3, 5, 2, 1, 7, 8, 9]
[1, 0, 4, 3, 5, 2, 6, 7, 8, 9]
[1, 0, 4, 3, 2, 5, 6, 7, 8, 9]
[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 0, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Out[7]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

1. Il s'agit de démontrer la **correction** de ces deux fonctions.
   Démontrez que pour la première, l'invariant de boucle est : « ind_temp est l'indice du maximum des i + 1 premiers termes de la liste en argument ».
   Quel est celui de la seconde ? Démontrez-le.
2. Nous mesurerons la complexité temporelle en nombre de comparaisons.
   Expérimentalement, nous pouvons en avoir une idée :

```
In [10]: ls = list(permutation(100))

In [11]: %timeit tri_select(ls)
1000 loops, best of 3: 561 µs per loop

In [12]: ls = list(permutation(200))

In [13]: %timeit tri_select(ls)
100 loops, best of 3: 2.03 ms per loop

In [14]: ls = list(permutation(400))

In [15]: %timeit tri_select(ls)
100 loops, best of 3: 7.89 ms per loop
```

   Démontrez-le.

Recherche 1 - 5 Tri fusion

Cette fois, si le tableau a au plus une valeur, il est trié, sinon on coupe le tableau en deux, on trie ces deux moitiés et on fusionne.

```
def tri_fusion(xs):
    t = len(xs)
    if t < 2:
        return xs
    return fusion(tri_fusion(xs[:t//2]), tri_fusion(xs[t//2:]))
```

Il reste à définir la fusion :

```
def fusion(xs,ys):
    if xs == Vide or ys == Vide:
        return xs + ys
    if tete(xs) < tete(ys):
        return [tete(xs)] + fusion(queue(xs),ys)
    return [tete(ys)] + fusion(xs,queue(ys))
```

1. Déterminez une version impérative de la fusion.
2. Étudiez la terminaison, la correction et la complexité de cet algorithme.

Recherche 1 - 6 Tri rapide

Observez et commentez :

```
def partition(pivot,seq):
    p,m,g = [],[],[]
    for item in seq:
        (p if item < pivot else (g if item > pivot  else m)).append(item)
    return p,m,g

def tri_rapide(xs):
    if estVide(xs):
        return Vide
    else:
```

```
        pivot = tete(xs)
        p,m,g = partition(pivot, xs)
        return (tri_rapide(p)) + m + (tri_rapide(g))
```

Pour la complexité en moyenne, démontrez que

$$K(n) = n - 1 + \frac{1}{n}\sum_{i=0}^{n-1}(K(i) + K(n-1-i))$$

et en déduire que

$$\frac{K(n)}{n+1} = 2\sum_{i=2}^{n}\frac{i-1}{i(i+1)}$$

Que se passe-t-il dans le pire des cas ?
Dans quel cas est-on sûr que le tri en $n \log n$?

Recherche 1 - 7 Tri sportif

Que pensez-vous de la complexité de cet algorithme de tri ?
(source : http://unclecode.blogspot.tw/2012/02/stupid-sort_2390.html) :

```
void StupidSort()
{
    int i = 0;
    while(i < (size - 1))
    {
           if(data[i] > data[i+1])
          {
                int tmp   = data[i];
                data[i]   = data[i+1];
                data[i+1] = tmp;
                i = 0;
          }
          else
          {
               i++;
          }
     }
}
```

Recherche 1 - 8 Yet another sort of sort

```
Pour i de 1 à n Faire
    Pour j de n à i+1 parPasDe -1 Faire
        Si t[j]<t[j-1] Alors
            Échange t[j] et t[j-1]
        FinSi
    FinPour
FinPour
```

Qu'est-ce que c'est ? Qu'est-ce que ça fait ? Comment ça marche ? Complexité ? En python ? Donnez un nom à ce tri.

Recherche 1 - 9 Multiplication du paysan russe

Voici ce qu'apprenaient les petits soviétiques pour multiplier deux entiers. Une variante de cet algorithme a été retrouvée sur le papyrus de Rhind datant de 1650 avant JC, le scribe Ahmes affirmant que cet algorithme était à

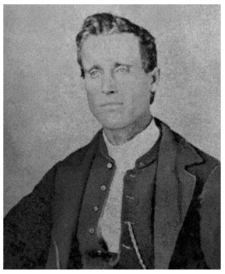
l'époque vieux de 350 ans. Il a survécu en Europe occidentale jusqu'aux travaux de Fibonacci.

```
1  Fonction MULRUSSE(x:entier ,y: entier, acc:entier):entier
2  Si x==0 Alors
3  |  Retourner acc
4  Sinon
5  |  Si x est pair Alors
6  |  |  Retourner MULRUSSE(x/2,y*2,acc)
7  |  Sinon
8  |  |  Retourner MULRUSSE((x-1)/2,y*2,acc+y)
9  |  FinSi
10 FinSi
```

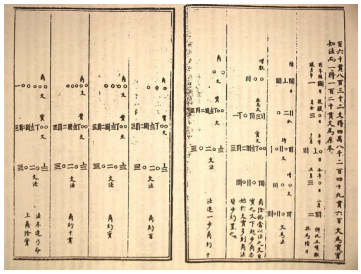
Que vaut acc au départ ?
Écrivez une version récursive de cet algorithme en évitant l'alternative des lignes 5 à 9.
Écrivez une version impérative de cet algorithme.
Prouvez la correction de cet algorithme.
Étudiez sa complexité.
En python, `x >> 1` décale l'entier x d'un bit vers la doite et `x << 1` décale x d'un bit vers la gauche en complétant par un zéro à droite, `x& y` renvoie l'entier obtenu en faisant la conjonction logique bit à bit des représentations binaires de x et y et `~x` renvoie le complément à 1 de x.
Ré-écrivez la multiplication russe en utilisant que ces opérations bit à bit (pas de division ni de multiplication).

Recherche 1 - 10 Schéma de Horner-Ruffini-Holdred-Newton-Al-Tusi-Liu-Hui...

Paolo Ruffini
(1765 - 1822)



William Horner
(1765 - 1822)



Neuf chapitres sur l'art mathématique

La méthode que nous allons voir porte le nom du britannique William George Horner (1786 - 1837) mais en fait elle fut publiée presque 10 ans auparavant par un horloger londonien, Theophilius Holdred et simultanément par l'italien Paolo Ruffini (1765 - 1822) mais fut déjà utilisée par Newton 150 ans auparavant et par le chinois Zhu Shije cinq siècles plus tôt (vers 1300) et avant lui par le Persan Sharaf al-Din al-Muzaffar ibn Muhammad ibn al-Muzaffar al-Tusi vers (1100) et avant lui par le Chinois Liu Hui (vers 200) révisant un des résultats présent dans *Les Neuf Chapitres sur l'art mathématique* publié avant la naissance de JC...
Il faut cependant noter que Ruffini l'avait employée en fait comme un moyen de calculer rapidement le quotient et le reste d'un polynôme par $(X-\alpha)$. C'est ce que nous allons (re)découvrir aujourd'hui...
Dans toute la suite, un polynôme de dégré n sera représenté par le vecteur de ses coefficients. Par exemple, `[1 2 3]` correspond au polynôme $1+2x+3x^2$.
Prenons l'exemple de $P(x)=3x^5-2x^4+7x^3+2x^2+5x-3$. Le calcul classique nécessite 5 additions et 15 multiplications. On peut faire pas mal d'économies de calcul en suivant le schéma suivant :

$$\begin{aligned}
P(x) &= \underbrace{a_n x^n + \cdots + a_2 x^2 + a_1 x}_{\text{on met } x \text{ en facteur}} + a_0 \\
&= \left(\underbrace{a_n x^{n-1} + \cdots + a_2 x}_{\text{on met } x \text{ en facteur}} + a_1 \right) x + a_0 \\
&= \cdots \\
&= \left(\cdots \left(\left((a_n x + a_{n-1}) x + a_{n-2} \right) x + a_{n-3} \right) x + \cdots \right) x + a_0
\end{aligned}$$

Ici cela donne $P(x) = (((((3x) - 2)x + 7)x + 2)x + 5)x - 3$ c'est-à-dire 5 multiplications et 5 additions.
Comparez les complexités au pire du calcul de $P(t)$ pour $t \in \mathbb{K}$. Vous prendrez comme « unité de complexité » les opérations arithmétiques de base : `+ - *`.
Déterminez une fonction `horner(P,t)` qui évalue le polynôme `P` en `t` selon le schéma de HORNER.

Recherche 1 - 11 Complexité de l'algorithme d'Euclide et nombre d'or...

On suppose dans toute la suite que a et b sont deux entiers naturels tels que $a \geqslant b$.

1. Rappeler le principe de l'algorithme et de sa preuve.
2. Soit $r_n = a \wedge b$ avec les notations habituelles : $r_0 = b$ puis $r_1 = a \bmod b$, $r_2 = r_1 \bmod r_0$ et plus généralement :

$$r_{k+1} = r_k q_k + r_{k-1}, \quad 0 \leqslant r_{k-1} < r_k$$

   Montrez que pour tout entier $k \in \{1, 2, \ldots, n\}$, on a $q_k \geqslant 1$ puis que $q_n \geqslant 2$.
3. Notons $\Phi = \frac{1+\sqrt{5}}{2}$. Comparez Φ et $1 + \frac{1}{\Phi}$.
4. Montrez par récurrence que $r_k \geqslant \Phi^{n-k}$ pour tout entier $k \in \{0, 1, \ldots, n\}$.
5. Déduisez-en que $b \geqslant \Phi^n$ puis que le nombre d'appels récursifs de l'algorithme d'Euclide est au maximum de $\frac{\ln b}{\ln \Phi}$.

In english :-)

Recherche 1 - 12 Local min

Write a program that, given an array a[] of N distinct integers, finds a local minimum : an index i such that both `a[i] < a[i-1]` and `a[i] < a[i+1]` (assuming the neighboring entry is in bounds). Your program should use 2 lg N compares in the worst case.

Recherche 1 - 13 Bitonic search

An array is bitonic if it is comprised of an increasing sequence of integers followed immediately by a decreasing sequence of integers. Write a program that, given a bitonic array of N distinct `int` values, determines whether a given integer is in the array. Your program should use 3 lg N compares in the worst case.

Recherche 1 - 14 Floor and ceiling

Given a set of comparable elements, the ceiling of x is the smallest element in the set greater than or equal to x, and the floor is the largest element less than or equal to x. Suppose you have an array of N items in ascending order. Give an O(log N) algorithm to find the floor and ceiling of x.

Recherche 1 - 15 Common elements

Given two arrays of N 64-bit integers, design an algorithm to print out all elements that appear in both lists. The output should be in sorted order. Your algorithm should run in N log N. Hint : mergesort, mergesort, merge. Remark : not possible to do better than N log N in comparison based model.

Recherche 1 - 16 Mumixam

Given an array a[] of N real numbers, design a linear-time algorithm to find the maximum value of a[j] - a[i] where j ⩾ i.

Recherche 1 - 17 DÉFI

Résolvez le problème suivant : http://introcs.cs.princeton.edu/java/assignments/collinear.html