Preuve et Analyse des Algorithmes

9 janvier 2014

Q 1. Soit n un entier positif. Soit $q \neq 1$ un nombre complexe.

1. Que vaut la somme de la série géométrique $\sum_{i=0}^{n-1} q^i$?

$$\sum_{i=0}^{n-1} q^i = \frac{q^n - 1}{q - 1}$$

2. Montrer que si $M_n = 2^n - 1$ est premier alors n est premier.
   Par la contraposée, si $n = rs$ alors

$$2^n - 1 = (2^{rs} - 1) = (2^r - 1) \sum_{i=0}^{s-1} 2^{ri}$$

   les deux facteurs sont supérieurs à 1 quand r et s le sont.

3. La réciproque est-elle vraie ?
   D'après l'examen de travaux-pratiques : non. On a $2^2 - 1 = 3$ premier, $2^3 - 1 = 7$ premier, $2^5 - 1 = 31$ premier, $2^7 - 1 = 127$ premier. Le plus petit contre exemple est :

$$2^{11} - 1 = 2047 = 23 \times 89.$$

Q 2. Soit n un entier positif.

1. Que vaut la somme de la série arithmétique $\sum_{k=0}^{n-1} k$?

$$\sum_{k=0}^{n-1} k = \frac{1}{2} n(n-1)$$

2. Cette somme apparaît naturellement dans l'analyse du temps de calcul d'un algorithme du cours. Lequel ?
   tri par selection.

Q 3. On note $I(a, b)$ le nombre d'itérations de l'algorithme d'Euclide pour calculer le PGCD des entiers a et b.

1. Quelle suite numérique est impliquée dans l'analyse de $I(a, b)$?
   La suite de Fibonacci. D'aprés le cours, pour $a > b$:

$$I(a, b) = k \Longrightarrow b \geq F_k, a \geq F_{k+1}.$$

2. Donner une instance (a, b) pour laquelle $I(a, b) = 11$.

$$(F_{12}, F_{11})$$

Q 4.

```
[drm@obelix] $ lscpu
Architecture: i686   Mode: 32-bit Vitesse proc. en MHz: 3000.000
[drm@obelix] $ free -b
             total       used       free     shared
Mem:    1582931968  837398528  745533440          0
-/+ buffers/cache:  389427200 1193504768
Swap:   3187666944          0 3187666944
[drm@obelix] $ echo 'l(1582931968)/l(2)'  | bc -l
30.559952105962841097 56
```

FIG. 1 – capacités de la machine `obelix`

1. Déterminer une solution (entière) de $127u + 107v = 1$.
   On applique l'algorithme du cours pour résoudre le problème de Bâchet-Bézout :

127	107	
1	0	127
0	1	107
1	-1	20
-5	6	7
11	-13	6
-16	19	1

$$127 \times (-16) + 107 \times (19) = 1$$

2. Quel est l'inverse de 107 modulo 127 ?
   C'est 19.
3. Que vaut 107^{126} modulo 127 ?
   Comme 127 est premier, le petit théorème de Fermat affirme que c'est 1.

Q 5. Une implantation du tri linéaire en langage C

```
typedef unsigned long long ullong;
void trilin( ullong *t , ullong n );
```

sur la machine `obelix` trie une instance de taille 2^{20} en 1 microseconde.

1. Rappeler les préconditions d'utilisation du tri linéaire.
   L'algorithme du tri linéaire s'applique pour trier n valeurs comparable à n. Il uilise beaucoup de mémoire.
2. Estimer le temps de calcul d'une instance de taille 2^{24} sur la machine `obelix`.
   C'est 16 μsec.
3. A quoi faut-il s'attendre pour la taille 2^{26} ?
   En théorie 64 μsec probablement beaucoup plus car les accès à la mémoire vont être ralentis par le swap.

Q 6. Soit m un entier positif. On rappelle que le coefficient de Fourier-Hadamard en a d'une table t de taille 2^m est donné par : $\hat{t}(a) = \sum_{0 \leq x < 2^m} t(x)(-1)^{a.x}$.

1. Rappeler l'algorithme récursif vu en cours.

```
FOURIER( table t )
    i, j : indice;
     tmp : valeur
debut
  si ( n = 1 ) alors retourner fsi
```

```
  soit g la partie gauche de t
  soit d la partie droite de t
  FOURIER( g );
  FOURIER( d );
  pour i dans [0,n/2] faire
     j    := i + n /2
     tmp  := t[j]
     t[i] := t[i] -  tmp
     t[j] := t [j] - tmp
  fip
fin
```

2. Calculer la tranformée de Hadamard-Fourier de la table

					$0,1,2,3,4,5,6,7$		
0	1	2	3	4	5	6	7
1	-1	5	-1	9	-1	13	-1
6	-2	-4	0	22	-2	-4	0
28	-4	-8	0	-16	0	0	0

Q 7. La figure [2] rapporte sur le temps de calcul d'un test de primalité de Fermat appliqué au n-ième nombre de Fermat $F_n = 2^{2^n} + 1$.

1. Commenter le grahique.
   Le temps de calcul est cubique en la taille des entiers.
2. Estimer le temps de calcul pour F_{13}.
   Notons T le temps de calcul en fonction de la taille, F_{13} est de taille 2^13, d'aprés le graphique :
   $$T(2^{13}) = 512 \times T(2^{10}) \sim 10240s$$
   soit de l'ordre de 3 heures.

Q 8. Pour un entier n, on note $\pi(n)$ le nombre de nombres premiers strictement inférieurs à n. L'algorithme du crible d'Eratostène est un procédé efficace pour déterminer tous les nombres premiers inférieurs à un entier donné n. On utilise une table t de n booléens initialisés à 0, sauf $t[0] = 1$ et $t[1] = 1$. L'algorithme procède par élimination en $\sqrt{n}$ étapes. A l'étape i, si $t[i] = 0$ c'est que i est premier, et, on marque "composé" les entiers multiple de i : $t[2i] = 1$, $t[3i] = 1$, ..., $t[ij] = 1$. A la fin de ce processus, les les entiers premiers correspondent aux cellules marquées 0.

On rappelle que la série harmonique : $H_n = \sum_{i=1}^{n} \frac{1}{i}$ vérifie la relation $H_n \sim \ln n$.

1. Utiliser le crible pour déterminé $\pi(100)$.
   $\pi(100) = 25$.
2. Ecrire une fonction `int pi( int n )` qui retourne $\pi(n)$.

```
int pi( int n )
{ int r = 0;
  int i, j, *t = calloc( n, 1 );
  for( i = 2; i*i <=n; i++ )
     if ( ! t[i] )
        for( j = i*2; j < n; j+=i )
                 t[j] = 1;
  for( i = 2; i < n; i++ )
     t += t[i];
  return n-r;
}
```

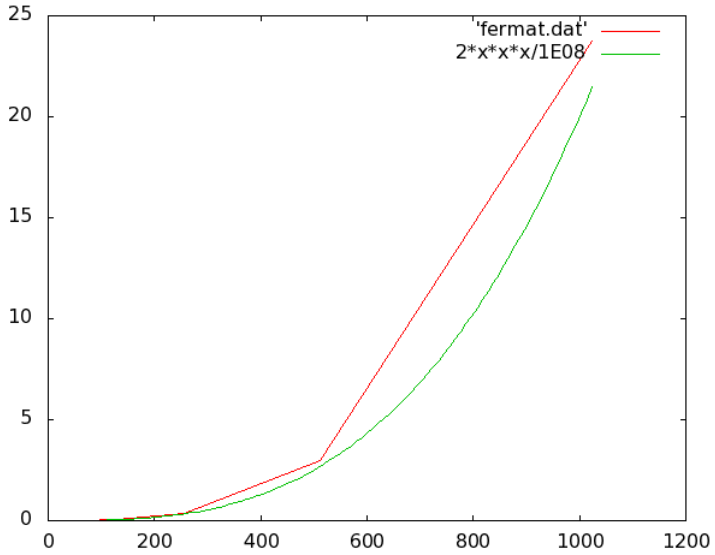

FIG. 2 – Temps de calcul en secondes du test de primalité de Fermat appliqué à F_n. En abscisse la taille binaire des nombres.

3. Etablir que le temps de calcul est proportionnel à $\sum_{p\leq\sqrt{n}} \frac{n}{p}$, où la somme porte sur des entiers premiers.
   Chaque nombre premier p coute n/p opérations de biffage. Le reste du travail est de coût linéaire.
4. Montrer que le temps de calcul est majoré par $nH_{\sqrt{n}}$.
   On néglige la contribution linéaire.

$$\sum_{p\leq\sqrt{n}} \frac{n}{p} \leq \sum_{k\leq\sqrt{n}} \frac{n}{k} = nH_{\sqrt{n}}$$

5. Montrer que le temps de calcul est $\mathrm{O}(n \log n)$.

$$nH_{\sqrt{n}} \sim n \log \sqrt{n} \leq \frac{1}{2} n \log n$$

   La notation O absorbe les facteurs multiplicatifs.