Université de Nice – Sophia Antipolis — Licence 2 Informatique
Outils Formels Pour l'Informatique — 20012–2013

TD 1

Complexité des algorithmes

1 Echauffement

Exercice 1) Réviser l'écriture d'un nombre entier sous forme décimale, sous forme binaire, sous forme hexadécimale. Ecrire les nombres 189 (en base 10), 10101011 (en base 2), A1B2 (en base 16) dans les différentes bases.

Exercice 2) Vous connaissez certainement les algorithmes qui permettent de décider si un nombre entier est divisible par 2, par 3, par 5, en n'utilisant que l'addition d'entiers et la comparaison de deux entiers. Quelle est la complexité de chacun de ces algorithmes, en raisonnant sur le nombre de chiffres nécessaires à l'écriture décimale du nombre et en prenant comme opérations élémentaires prises l'addition et la comparaison.

Exercice 3) Adapter l'algorithme de multiplication *à la russe* aux nombre binaires.

2 Exercices d'entrainement

Exercice 4) Si une instruction est traitée en 10^{-7} seconde, quelle est la taille maximale que peut traiter un algorithme en $\log(n)$, n, $n\log(n)$, n^2, 2^n si on dispose de une seconde / une minute / une heure / une journÈe ? Même questions si chaque instruction est traitée en $10^{-20} seconde$

Exercice 5)

1. Prouver que $n^2 - n = O(n^2)$.
2. Prouver que $n^2 + n = O(n^2)$.
3. Donner l'ordre de grandeur des expressions suivantes:
   (a) $\dfrac{n^2 + 3n + 1}{n + 1}$
   (b) $\dfrac{n\log(n) + n^2 + \log(n)^2}{n + 1}$
4. Si $f(n)$ en une fonction en $O(n)$, les affirmations suivantes sont-elles vraies ?
   (a) $(f(n))^2 = O(n^2)$;
   (b) $2^{f(n)} = O(2^n)$.

Exercice 6) Itérations. Calculer la complexité des programmes suivants. Vous donnerez une borne supérieure avec un $O()$ dans un premier temps et affinerez votre calcul en utilisant $\Theta()$.

listing 1

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j = j++) {
    x = x + 3; }
  }
```

listing 2

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j = j++) {
    x = x + 3; }
  }
```

listing 3

```
for (int i = 0; i < n; i++) {
    for (int j = i; j < n-i; j++) {
    x = x + 3; }
  }
```

listing 4

```
for (int i = 5; i < n; i++) {
    for (int j = i-5; j < i+5; j++) {
    x = x + 3; }
  }
```

listing 5

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i*n; j = j++) {
    x = x + 3; }
  }
```

listing 6

```
for (int i = n; i > 1; i=i/2) {
    for (int j = 0; j < i; j++) {
    x = x + 3; }
  }
```

3 Pour aller plus loin

Exercice 7) Écrire un algorithme qui calcule, pour un tableau T de taille n donné en paramètre, la somme :

$$\sum_{i,j=1}^{n} (T(i) - T(j))^2$$

1. Écrire l'algorithme "naïf" quadratique qui vient en premier à l'esprit.
2. Est-il possible de linéariser cet algorithme ?

Exercice 8) Trouver un algorithme qui teste si un tableau de taille n représente une permutation (c'est-à-dire si tous ses éléments sont distincts et compris entre 1 et n).

1. Donner un premier algorithme naïf qui soit quadratique.
2. Donner un second algorithme linéaire utilisant un tableau auxiliaire.

Exercice 9) Ecrire un algorithme pour calculer p^n, qui soit de complexité $O(\log_2 n)$ (indication: considérer n écrit en binaire).

Exercice 10) Etudiez le nombre d'additions réalisées par l'algorithme suivant dans le meilleur des cas, dans le pire des cas et enfin en moyenne en supposant que chaque test a une probabilité $\frac{1}{2}$ d'être vrai.

```
for (int i = 0; i < n; i++) {
    if(T[i]>a,s=s+T[i]);  }
```