# TD de Algorithmique n° 4

**27 octobre 2011**
**(Correction)**

## a) Le tri par comptage

Le *tri par comptage* ou *tri par dénombrement* (*counting sort*, an anglais) est un algorithme de tri qui s'applique à un tableau $A$ d'entiers compris entre 0 et $k$. Pour chaque élément $x$, l'algorithme détermine le nombre d'éléments qui lui sont inférieurs et on place $x$ dans la bonne position dans un autre tableau $B$.

L'algorithme suivant implémente une version simplifiée du tri par comptage : il marche sous l'hypothèse que dans le tableau $A$ il n'y a pas de répétitions (et donc que $k \geq length[A]$).

SimpleTriComptage$(A, B, k)$
1 **for** $i \leftarrow 1$ **to** $k$
2 **do** $C[i] \leftarrow 0$
3 **for** $j \leftarrow 1$ **to** $length[A]$
4 **do** $C[A[j]] \leftarrow 1$
5 **for** $i \leftarrow 1$ **to** $k$
6 **do** $C[i] \leftarrow C[i] + C[i-1]$
7 **for** $j \leftarrow 1$ **to** $length[A]$
8 **do** $B[C[A[j]]] \leftarrow A[j]$

**Exercice 1 :**

1. Modifier l'algorithme précédent pour pouvoir l'appliquer aussi à des tableaux qui contiennent des répétitions. A partir de maintenant on considère la version non simplifié du tri par comptage.
2. Comment peut-on exécuter l'algorithme de tri par comptage sur un tableau $A$ si on ne connait pas la valeur de $k$ ?
3. Simuler l'exécution du tri par comptage sur les tableaux suivants et déterminer les valeurs de $k$ :

$$A = \langle 3, 0, 7, 5, 3, 5, 8, 0, 9, 5, 3, 8 \rangle \quad B = \langle 6, 2, 1, 3, 11, 14, 13, 1, 4, 11, 15, 5, 6, 9, 2 \rangle$$

4. Est-ce que le tri par comptage est un tri stable ? Est-ce qu'il trie sur place ?

Rappel : un tri est *stable* s'il conserve l'ordre initial de deux éléments lorsque ceux-ci sont considérés comme "égaux" par la relation d'ordre choisie.

**Correction :**

1. L'algorithme général est le suivant :

SIMPLETRICOMPTAGE$(A, B, k)$
1 **for** $i \leftarrow 1$ **to** $k$
2 **do** $C[i] \leftarrow 0$
3 **for** $j \leftarrow 1$ **to** $length[A]$
4 **do** $C[A[j]] \leftarrow C[A[j]] + 1$
5 **for** $i \leftarrow 1$ **to** $k$
6 **do** $C[i] \leftarrow C[i] + C[i-1]$
7 **for** $j \leftarrow length[A]$ **downto** 1
8 **do** $B[C[A[j]]] \leftarrow A[j]$
9 $C[A[j]] \leftarrow C[A[j]] - 1$

2. Il faut d'abord trouver le maximum du tableau, et après l'utiliser comme valeur $k$.
3. A faire.
4. Le tri par comptage est un tri stable mais il ne trie pas sur place.

## b) Le tri par base

On rappelle le principe du *tri par base* (ou *radix-sort*, an anglais) : on a un tableau de nombres à trier et on effectue le tri par rapport à l'unité d'abord, en parcourant le tableau dans l'ordre et en "rangeant dans la file" correspondante au chiffre le nombre concerné. On reprend ensuite les nombres et on les classe de la même façon en suivant leur chiffre des dizaines, etc.

La procédure suivante, qui implémente cet algorithme, suppose que chaque élément du tableau $A$ a $d$ chiffres $(x_d \dots x_1)$ :

TRI-BASE$(A, d)$
1 **for** $i \leftarrow 1$ **to** $d$
2 **do** utiliser un tri stable pour trier $A$ par rapport au $i$-ème chiffre.

**Exercice 2 :**

1. Trier par base les tableaux suivants :

$$A = \langle 5998, 1445, 2114, 9479, 6566, 8680, 2680, 6016, 6592, 2422 \rangle;$$

$$B = \langle 49053, 70527, 15434, 50039, 68443, 34974 \rangle.$$

2. Est-ce que l'algorithme marche encore si on utilise un tri qui n'est pas stable ?
3. Comment peut-on le modifier pour trier un tableau de mots (à la place de nombre) ? Quel ordre doit-on considérer ?

**Correction :**

1. A faire.
2. Non.
3. L'algorithme est le même, mais il faut utiliser l'ordre lexographique sur les mots.

# I) exponentiation rapide

On concidère l'algorithme :

```
INCONNU(a,b)
1  i ← a
2  j ← b
3  p ← 1
4  while j > 0
5        do if j % 2 = 1
6              then p ← p * i
7           i ← i * i
8  j ← j/2
9  return p
```

**Exercice 3 :**

1. Que vaut INCONNU(7,10) ? Que calcule INCONNU ?
2. Quel invariant cette boucle vérifie elle ?
3. Quelle est la complexité de l'algorithme ?

**Correction :**

1. $7^{10}$ et $a^b$
2. $a^b = p * i^j$
3. $\lfloor \log b \rfloor$