Corrigé de l'exercice 1 (Série 3)

1. $g(n)=g(n-1) + 2n-1$ si $n>0$ et $g(0)=0$

$=g(n-2)+2(n-1)-1 + 2n-1$

$=g(n-3)+2(n-2)-1 + 2(n-1)-1 + 2n-1$

$=g(n-4)+ 2(n-3)-1 + 2(n-2)-1 + 2(n-1)-1 + 2n-1$

….

$= g(0)+2(\sum_{i=1}^{n} i)-n$

$= 0 + 2(n(n+1)/2) - n$

$= n^2$

2. $T(m,n)=2*T(m/2, n/2) + m*n$ si $m>1, n>1, m<=n$
$T(m,n)=n$ si $m=1$
$T(m,n)=m$ si $n=1$

Nous pouvons résoudre cette équation de récurrence en utilisant les itérations comme suit:

Sachant $m <= n$. alors :

$T(m,n) = 2*T(m/2,n/2) + m*n$

$= 2^2*T(m/2^2,n/2^2) + 2*(m*n/4) + m*n$

$= 2^2*T(m/2^2,n/2^2) + m*n/2 + m*n$

$= 2^3*T(m/2^3,n/2^3) + m*n/2^2 + m*n/2 + m*n$

...

$= 2^i*T(m/2^i,n/2^i) + m*n/2^{(i-1)} +...+ m*n/2^2 + m*n/2 + m*n$

Posons $k = \log_2 m$. Alors nous aurons

$T(m,n) = 2^k*T(m/2^k,n/2^k) + m*n/2^{(k-1)} +...+ m*n/2^2 + m*n/2 + m*n$

$= m*T(m/m,n/2^k) + m*n/2^{(k-1)} +...+ m*n/2^2 + m*n/2 + m*n$

$= m*T(1,n/2^k) + m*n/2^{(k-1)} +...+ m*n/2^2 + m*n/2 + m*n$

$= m*n/2^k + m*n/2^{(k-1)} +...+ m*n/2^2 + m*n/2 + m*n$

$= m*n*\sum_{i=0}^{k} \frac{1}{2^k}$

$= m*n*2$

$= Theta(m*n)$

Exercice 2

4. A la fin de l'exécution de la fonction Partition_Deb(T, 1, n), tous les éléments de T[1..k] sont strictement inférieurs à tous les éléments de T[k+1..n]. On en déduit immédiatement que tous les éléments de T[1..k] sont de rang au plus égal à k, alors que tous les éléments de T[k+1 ..n] sont de rang au moins égal à k+1.

5. L'algorithme consiste à réarranger le tableau T[i..j] en deus sous-tableaux T[i..k] et T[k+1..j] à l'aide de la procédure Partition_Deb, où tous les éléments du premier sont inférieurs au pivot T[i]. Après l'échange de T[i] et de T[k], le tableau T est donc découpé en trois parties, le sous-tableau T[i..k] contenant des valeurs strictement inférieurs à T[k], l'élément pivot T[k] et le sous-tableau T[k+1..j] contenant des valeurs supérieurs à T[k]. Trois cas de figures peuvent alors se présenter pour p$\in${i…j}. Si p=k, l'élément de rang p se trouve dans le sous-tableau T[i..k-1] et la fonction Rang se rappelle récursivement sur ce sous-tableau. finalement, si p>k, l'élément de rang p se trouve dans le sous-tableau T[k+1..j] et la fonction Rang se rappelle récursivement sur ce dernier sous-tableau.

```
fonction  Rang(T :tableau ; p,i,j :entier) :entier ;

k :entier ;

si i=j   alors   retourner(T[i])

sinon

   k<- Partition_Deb(T,i,j) ;

   echanger T[i] et T[k] ;

   si  p=k     alors    retourner(T[k])

   sinon     si p<k     alors  retourner(Rang(T,p,i,k-1))

                 sinon  retourner(Rang(T,p,k+1,j)) ;

                 fsi ;

    fsi ;

fsi ;
```

L'appel initial de la fonction est Rang(T,p,1,n)

6. Dans le pire des cas, à chaque appel récursif de la fonction Rang, l'entier k retourné par l'appel à la fonction Partition_Deb(T,i,j) est toujours égal à i. Le tableau T[i..j] est alors réarrangé en deux sous-tableaux T[i..j] de taille 1 et T[i+1..j] d'une taille diminuée de 1 par rapport à celle du tableau initial. Pour peu que p soit égal à n, le processus va repartir sur le plus grand des deux tableaux (T[i+1..j], et se répéter jusqu'à tomber sur le sous-tableau T[n..n].

On peut se convaincre aisément que le pire cas correspond à un tableau trié en ordre croissant. L'algorithme Rang doit alors effectuer n appels récursifs (Rang(T,p,i,n) pour i=1,…,n) et pour chacun,appeler la procédure Partition_Deb(T,i,n). On en déduit la complexité (en termes de comparaisons de clés de la fonction Partition_Deb) :

$$c(n) = \sum_{i=1}^{n} \mathrm{O}(n-i)$$

L'algorithme est donc en $O(n^2)$

Exercice 3

1. $\cup$ si A=∅ (resp B=∅) on a A∩B=∅ donc A∆B=B (resp A∆B=A)

2. supposons que $a_1=b_1$. On a $a_1 \notin$ A∆B et $b_1 \notin$ A∆B. Il en résulte que

   A∆B= A'∆B'.

   Supposons que $a_1<b_1$. On a alors $a_1 \in$ A∆B. Si x $\in$ A'∆B, on a aussi x $\in$ A∆B. D'où $\{a1\} \cup$ (A'∆B) $\subset$ A∆B. Soit x $\in$ A∆B. Si x≠a_1, alors on a soit x $\in$ A' et x $\notin$ B et donc x $\in$ A'∆B, soit x $\in$ B et x $\notin$ A' et donc x $\in$ A'∆B. Si x=a_1 alors x $\in$ $\{a1\} \cup$ (A'∆B) . D'où A∆B $\subset$ $\{a1\} \cup$ (A'∆B). Il en résulte que
   A∆B = $\{a1\} \cup$ (A'∆B).
   Si $b_1<a_1$, un raisonnement analogue à celui du cas précédent montre que :
   A∆B = $\{b_1\} \cup$ (A∆B')

3. fonction Diff_Sym(LA, LB :tle) :tle

```
var L :tle ;
Debut
si (LA=nul) alors retourner (LB)
sinon si (LB=nul) alors retourner (LA)
        sinon si (LA→valeur = LB→valeur) alors
                                        retourner(Diff_Sym(LA→svt, LB→svt)
                sinon si (LA→valeur <LB→valeur) alors
                                            retourner(Diff_Sym(LA→svt,LB) ;
                                            ajoutTete(LA→valeur,L)
                        sinon retourner(Diff_Sym(LA,LB→svt) ;
                                ajoutTete(LB→valeur,L) ;
                        fsi ;
                  fsi ;
           fsi ;
fsi ;
Fin ;
```

4. La complexité de la fonction Diff_Sym est en O(n+m) avec n la longueur de LA et m la longueur de LB.