<u>*ANALYSE ASYMPTOTIQUE*</u>

Exercice 1 : Tours de Hanoi (algorithme récursif)

```java
    public static void hanoi(int n, List<Integer> tourDepart, List<Integer>
tourArrivee, List<Integer> tourIntermediaire) {
        if (n == 1) {
            tourArrivee.add(tourDepart.remove(tourDepart.size() - 1));      T(1)
        } else {
            hanoi(n - 1, tourDepart, tourIntermediaire, tourArrivee);       T(n-1)
            tourArrivee.add(tourDepart.remove(tourDepart.size() - 1));      T(1)
            hanoi(n - 1, tourIntermediaire, tourArrivee, tourDepart);       T(n-1)
        }
        compteur++;
    }
```

Si on teste la valeur de n pour plusieurs valeurs, on obtient :

Nombre de disques	1	2	3	4	5	6	...	n
Nombre de déplacements	1	3	7	15	31	63	...	$2^n - 1$

$\times 2$ $+1$ $\times 2$ $+1$ $\times 2$ $+1$ $\times 2$ $+1$ $\times 2$ $+1$ $\times 2$ $+1$

On a deux déplacements simples, plus deux appels à la fonction récursive :

T(n)=1+(n-1)+ 1+(n-1)= 2 + 2 (n-1)

= 2^(n-1)

<u>On a donc une complexité de O(2^n)</u>

Exercice 2 : Tri sélectif (algorithme récursif)

```java
public static void tri(int[] tab, int n) {
    if (n == 1) {
        return;
    }
    tri(tab, n - 1);
    int dernier = tab[n - 1];
    int j = n - 2;
    while (j >= 0 && tab[j] > dernier) {
        tab[j + 1] = tab[j];
        j--;
    }
    tab[j + 1] = dernier;
}
```

Dans le pire des cas on aura une complexité de $O(n^2)$ avec n la taille de l'entrée (en réalité n-1).