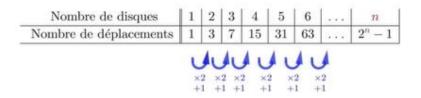
## **ANALYSE ASYMPTOTIQUE**

## Exercice 1 : Tours de Hanoi (algorithme récursif)

```
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 :



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)

On a donc une complexité de O(2^n)

## Exercice 2 : Tri sélectif (algorithme récursif)

```
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(n²) avec n la taille de l'entrée (en réalité n-1).