# Algos de tri

3 types d'algos de tri:

- tri par selection
- tri par propagation
- tri par insertion

tout les algorithmes reposent sur une méthode d'echange d'items basé sur leurs indice:

```
def swap(T, i, j):
   a = T[i]
   T[i] = T[j]
   T[j] = a
```

## tri par selection

bien que simple, l'algorithme est considere comme inefficace a cause de son temps d'execution quadratique

$$\Omega(n) = n^2$$
$$O(n) = n^2$$
$$\Theta(n) = n^2$$

il consiste a parcourir une liste, et echanger le plus petit element par le premier, puis d'avencer l'indice du premier jusqu'a finir de parcourir la liste

```
i ← 1
while i < length(A)
    j ← i
    while j > 0 and A[j-1] > A[j]
        swap A[j], A[j-1]
        j ← j - 1
    end while
    i ← i + 1
end while
```

## tri par propagation (Bubble sort)

il a une complexite de $n^2$, sauf pour le meilleur cas, ou $\Omega(n) = n$ il consiste a echanger les elements qui sont dans le désordre ($n + 1 < n$), a la fin de chaque iteration, le dernier element est le plus grand, donc l'indice est soustre a chaque fois

```
tri_à_bulles(Tableau T)
   pour i allant de (taille de T)-1 à 1
       pour j allant de 0 à i-1
           si T[j+1] < T[j]
               (T[j+1], T[j]) ← (T[j], T[j+1])
```

## tri par insertion

il parcours la liste, et a chaque fois que l'algo trouve un nombre inferieur au precedent, il revient en arriere pour le placer correctement

```
  procédure tri_insertion(tableau T)

       pour i de 1 à taille(T) - 1

            # mémoriser T[i] dans x
            x ← T[i]
```

```
            # décaler les éléments T[0]..T[i-1] qui sont plus grands que x, en
partant de T[i-1]
            j ← i
            tant que j > 0 et T[j - 1] > x
                     T[j] ← T[j - 1]
                     j ← j - 1

            # placer x dans le "trou" laissé par le décalage
            T[j] ← x
```