# NSI 1ère - Algorithmique - Tris 4

QK

## Quatrième partie : complexité

### Rappel : tri par insertion

```
i = 1
Tant que i < longueur(A)
    j = i
    Tant que j > 0 et A[j-1] > A[j]
        echanger A[j] et A[j-1]
        j = j - 1
    fin tant que
    i = i + 1
fin tant que
```

### Tri par insertion : meilleur cas, pire cas

- *meilleur cas* : liste déjà triée
- *pire des cas* : liste triée à l'envers.

### petit TP : Mesurer la complexité du tri par insertion.

- Pour des listes de plus en plus grandes, on mesure les temps d'exécution
- On trace une courbre présentant les résultats
- Conjecture sur la complexité du tri par sélection

### Correction du TP : tri par sélection

- Le tri par insertion est en $O(n^2)$
- Pire des cas : la liste est triée par ordre contraire

Quelques exemples de vitesses en fonction de la taille de la liste d'entrée pour le tri par insertion

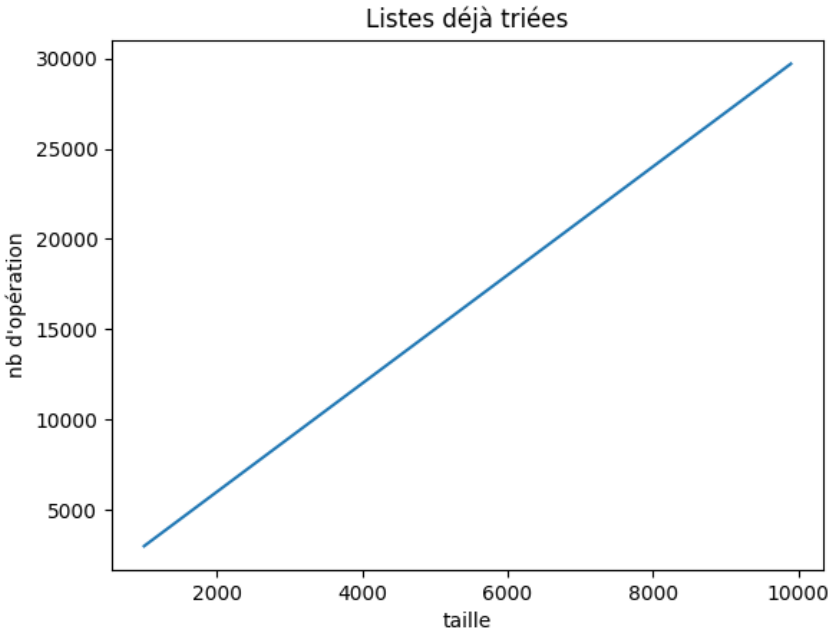



Figure 1: meilleur cas

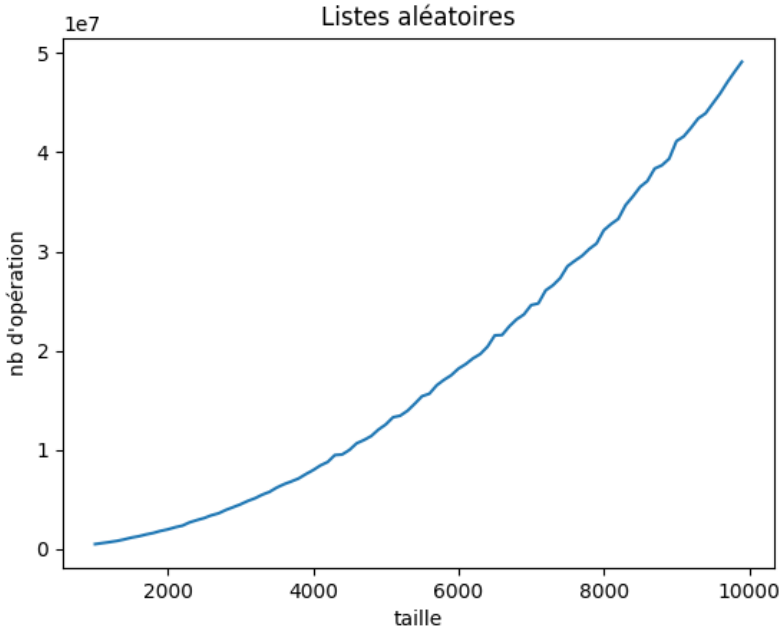



Figure 2: random

### Meilleur cas

### Cas random

### Pire des cas

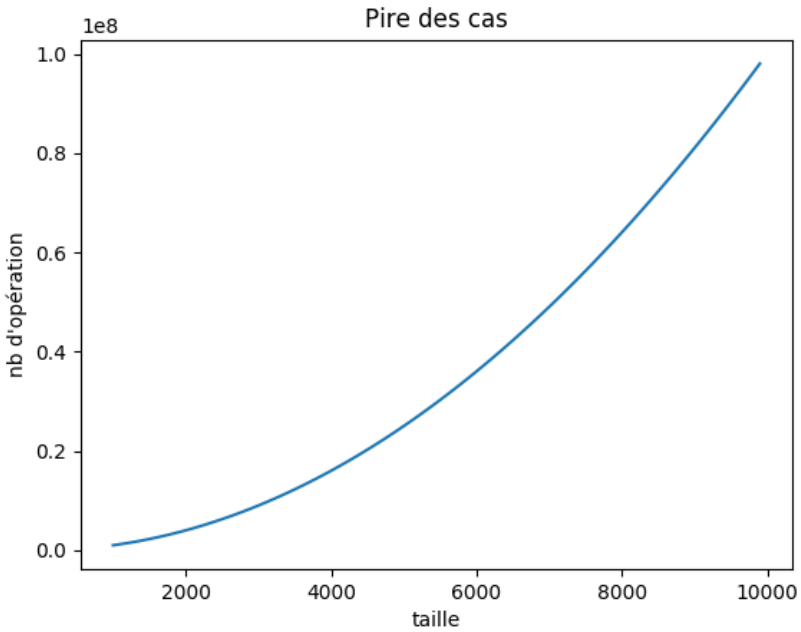



Figure 3: pire cas

## insertion : preuve de la complexité

### Complexité : calcul à la main

- Pour simpliflier, on ne compte que les échanges.
- Dans ce cas, à chaque itération de la boucle intérieure, on parcourt toute la liste triée et on échange tous les couples d'éléments avant d'insérer.
- Celle-ci contenant `i` éléments, le nombre d'échanges est :

  $C = 1 + 2 + 3 + \cdots + n$
- ...

## Complexité : calcul à la main

- Remarquons qu'on peut écrire $C$ à l'envers :

$$\begin{aligned} C &= 1 + 2 + \cdots + (n-1) + n \\ C &= n + (n-1) + \cdots + 2 + 1 \end{aligned}$$

- On ajoute en colonne :

$$\begin{aligned} 2C &= (n+1) + (n+1) + \cdots + (n+1) + (n+1) \\ 2C &= n(n+1) \\ C &= n(n+1)/2 \end{aligned}$$

  $C$ est toujours plus petit qu'un polynôme en $n^2$.

On note : $C = O(n^2)$

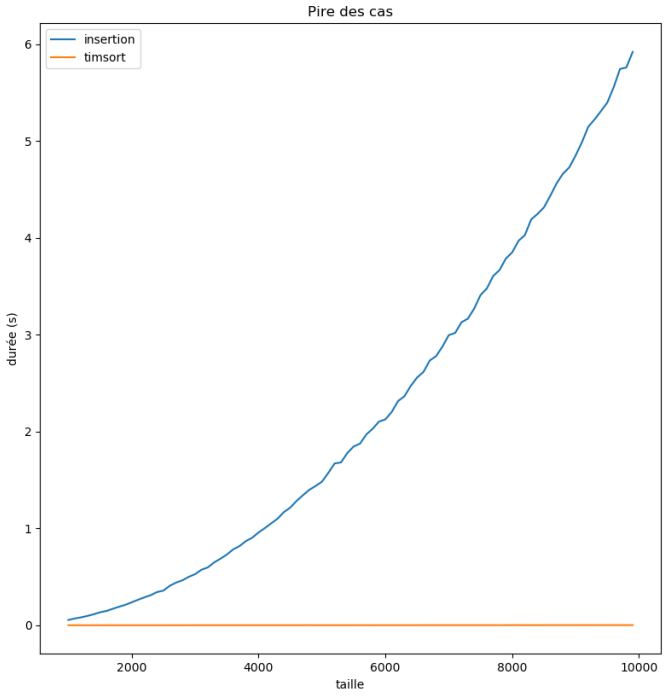
Le tri par insertion est de complexité **quadratique**

## $O(n^2)$

- Comme le tri par insertion, le tri par sélection est quadratique.
- Le nombre "d'opérations" réalisées dans un tri par sélection ou par insertion est de l'ordre du carré de la taille.
- On dit que ce sont des algorithmes en $O(n^2)$

# Comparaison à un autre tri

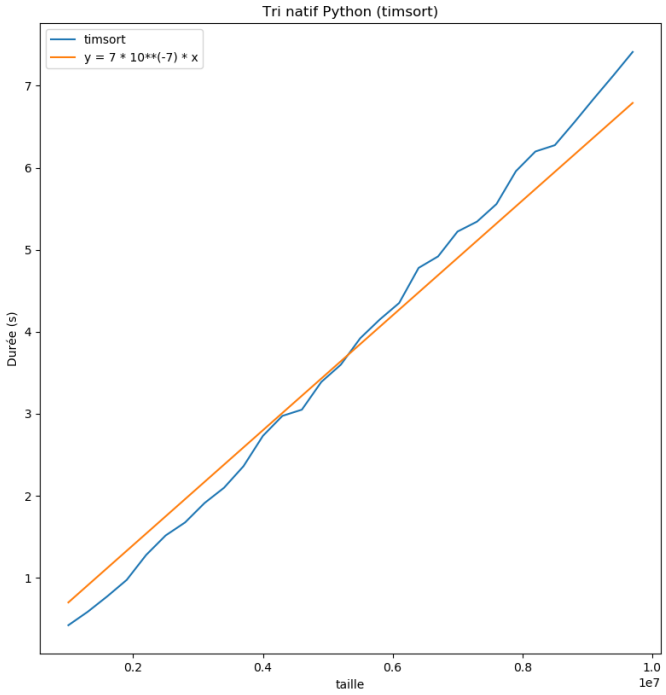
## Tri par insertion vs Tri natif en Python




.

Le tri natif est-il à temps constant ????

## Tri natif en Python seul




.

non... le tri natif (TimSort) est en $O(n \times log(n))$