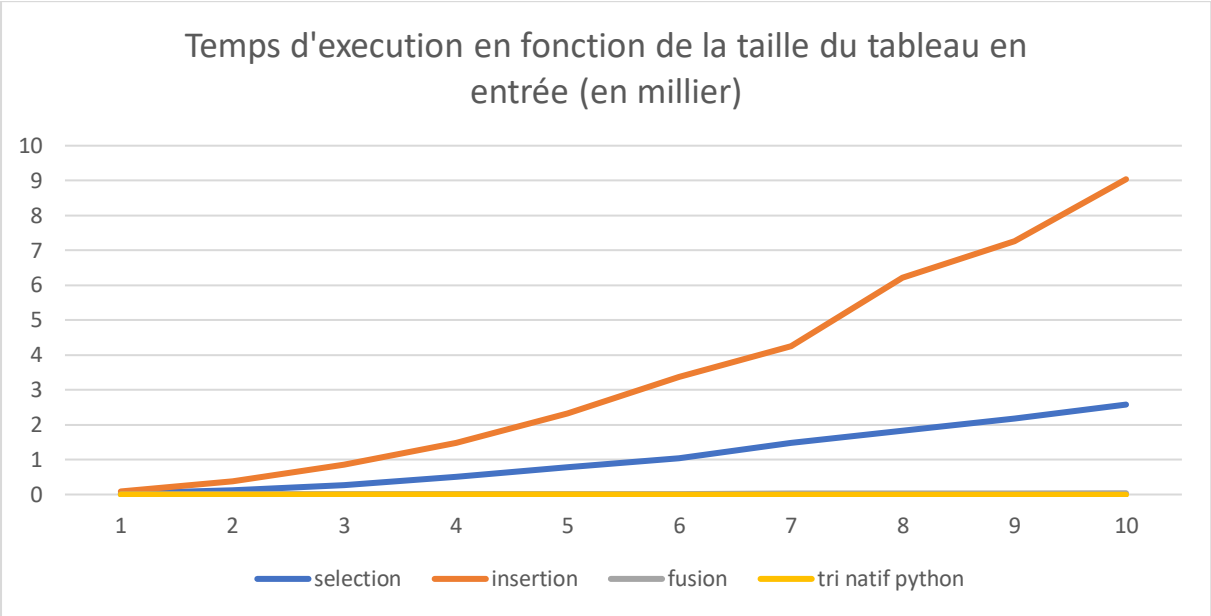
Exercice préliminaire

Comment vous semble évoluer la courbe ?

La courbe évolue de manière linéaire, avec un niveau de complexité de O(n)

Résultats



Tri par sélection

- Écrivez en français classique ce que vous voyez. Quel est le fonctionnement ? Comment l'expliqueriez-vous à quelqu'un ?

Cette méthode est linéaire et semble être une optimisation de la méthode de tri par insertion, car il y a trois curseurs avec le même principe que le tri par insertion, mais un avec index supplémentaire.

Tri par insertion

- Écrivez en français classique ce que vous voyez. Quel est le fonctionnement ? Comment l'expliqueriez-vous à quelqu'un ?

Cette méthode est linéaire, avec une approche assez classique. Il s'agit juste d'incrémenter deux curseurs, puis de les faire évoluer d'une part de l'intervalle 0 à n pour le premier, et de c+1 à n pour le deuxième, c représentant dans notre cadre le premier curseur.

Ces deux curseurs vont servir de point de comparaison pour une éventuelle permutation de deux index.

Tri par fusion

Cet algorithme est assez complexe car il mélange deux approches : la récursion ainsi qu'un principe de dichotomie simplifié à l'extrême.

En effet, l'idée est ici de diviser la liste sous forme de listes individuelles, c'est-à-dire ne comprenant qu'un seul élément. On ajoute à cette idée deux fonctions : l'une, récursive, s'appliquant à subdiviser la liste pour la propulser vers la seconde, dédiée à la fusion de deux listes prétriées.

- Question bonus : Y a-t-il des tailles de tableaux pour lesquelles le tri par fusion n'est pas aussi rapide que les précédents tris abordés ?

Oui pour les tableaux de très petites tailles, le tri par fusion n'est pas aussi bien optimisé car sa courbe de complexité algorithmique n'est pas adaptée.

Tri natif Python

- Une dernière fois, analysez le temps d'exécution et découvrez si python fait mieux que nos implémentations rudimentaires ;)

L'algorithme sort bat tous les records. J'imagine que l'implémentation à Python profite de chacune des fonctionnalités les plus up-to-date de Python !

Le mot de la fin

Si je devais créer ma propre implémentation d'un algorithme de tri, je me focaliserais sur la taille et les dimensions des données entrantes. En l'occurrence, si le tableau est petit, je m'appliquerais à utiliser un algorithme de sélection. En revanche, pour les tailles de tableau plus importantes, je choisirais un algorithme de complexité O(n) tel que l'algorithme de tri par fusion.

Enfin, pour aller plus loin, je pense qu'une implémentation sous forme d'arbre serait la solution la plus valable sur le long terme, car dans le cadre du tri de plusieurs millions d'entier (par exemple dans le cadre d'une base de données). Car en possédant un index dynamique qui se tri au fur et à mesure de l'ajout de nouvelles données, on peut imaginer un temps d'exécution quasi instantanné.