

Projet FORMATION DIU Enseigner l'informatique au lycée

Objet : Etudier la complexité en temps des algorithmes de tris

Sujet du projet

Etude de la complexité de tris en Python

- **Les concepts techniques et aspects transverses**

Le but de ce projet est d'étudier la complexité en temps (au pire et dans des cas quelconques) des algorithmes de tris en exécutant sur des tableaux de tailles différentes (petites ou grandes), avec pour une taille donnée n des configurations de tableaux variées (aléatoires ou non – trié ordre croissant, trié ordre décroissant), et en les comparant en termes d'efficacité. On pourra regarder le temps d'exécution pour les différents tris et les comparer.

On s'intéressera aux opérations fondamentales « comparaisons entre éléments de tableaux » et « déplacements d'éléments de tableaux » on pourra ajouter dans les algorithmes de tris des compteurs d'opérations fondamentales), et en confrontant avec les résultats théoriques. On fera également attention à la complexité en place. Les tris choisis sont : tri insertion, tri bulles, tri rapide, tri fusion. Les coûts de différentes exécutions des différents tris seront stockés et affichés de façon graphique.

- **Les prérequis**

Ils auront reçu un enseignement Python assez complet. Ils devront avoir une maîtrise :

- des boucles ;
- de la notion de tableau ;
- des fichiers en lecture, en écriture ;
- de l'utilisation de fonctions ;
- d'appel aux bibliothèques.

- **Nombre d'élèves**

Il est prévu que les élèves soient par groupe de 3.

- **Volume horaire (heures classe + heures maison)**

Il est prévu 7 séances de 2 heures en classe avec du travail à terminer à la maison en fin de chaque séance.

Au sein du groupe, l'organisation se fera selon trois modalités :

- une séance d'introduction commune à tous sur les tris. Une séance commune sur le tri rapide (démonstration et programmation)
- Trois séances où chaque élève étudie un tri choisi
- deux séances consacrées aux comparaisons et aux rendus de chacun des tris en fonction de leurs configurations.

- Modalités de rendu et d'évaluation

Le groupe devra soutenir son projet lors d'une présentation orale de dix minutes en s'appuyant sur un outil numérique. Cette présentation sera suivie d'une démonstration fonctionnelle du projet. Associé à cette présentation, un compte rendu écrit de suivi, de projet au format pdf (4 à 6 pages), retraçant les étapes du projet et mettant en évidence les choix, les difficultés rencontrées, devra être transmis.

- Déroulement d'une séance de projet en classe

Au début de chaque séance, les élèves devront évaluer la réussite de leurs objectifs. Ils devront travailler en autonomie, rechercher des informations, prendre des décisions et avancer dans leur projets...

Ils devront compléter un cahier de suivi et définir les objectifs à atteindre pour la prochaine séance ; bien évidemment en fonction des difficultés rencontrées et de l'avancement du projet. Ils devront terminer la séance en ayant défini le travail maison et sa répartition.

Au cours de la séance, ils devront rendre compte de l'avancée du projet, des difficultés rencontrées avec le professeur qui servira de chef de projet.

- Plan du cours

Introduction : Qu'est-ce que trier ?

I- Comment fait Python ?

- 1) sort
- 2) sorted

II- Qu'est ce que le tri rapide ?

- 1) Explications, démonstration
- 2) codage

III- Les autres tris

- 1) Explications, démonstration
- 2) codage

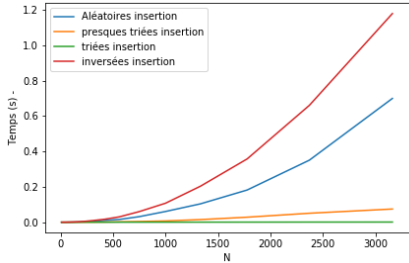
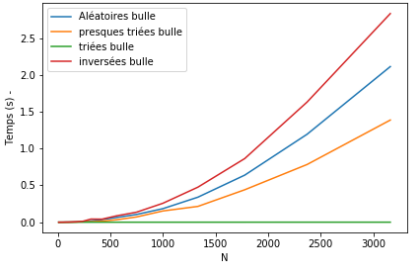
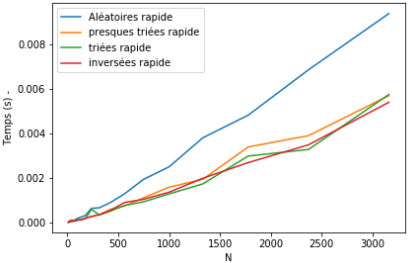
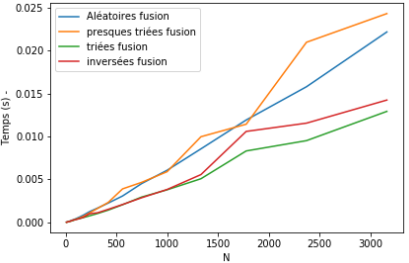
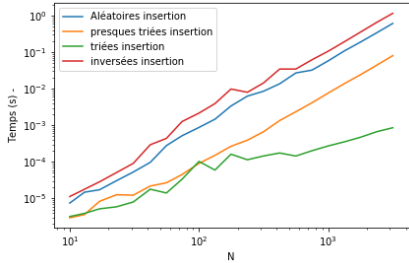
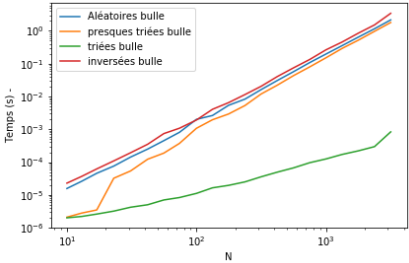
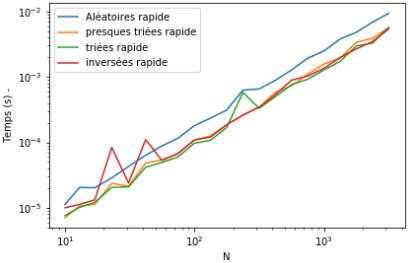
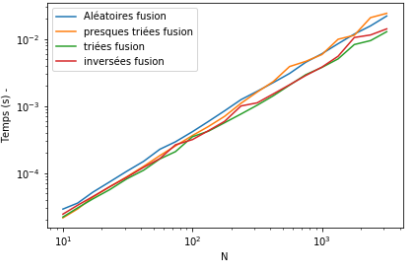
IV- Comment comparer les algorithmes ?

- 1) Comparaison d'éléments
- 2) déplacements d'éléments

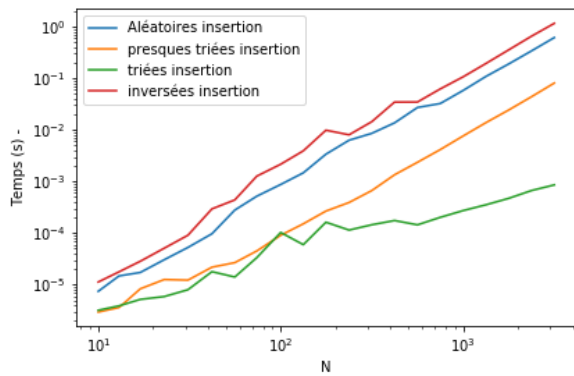
annexe

tri rapide:écriture très compacte du code grâce à la définition de tableau par compréhension
def triRapide(liste):

```
    if not liste:
        return []
    else:
        pivot = liste[-1]
        plusPetit = [x for x in liste if x < pivot]
        plusGrand = [x for x in liste[:-1] if x >= pivot]
        return triRapide(plusPetit) + [pivot] + triRapide(plusGrand)
```

	Tri insertion	Tri bulles	Tri rapide	Tri fusion
Cas favorable	Déjà trié Complexité temporelle $\Theta(n)$	Déjà trié Complexité temporelle $\Theta(n^2)$	Les deux sous-listes sont « équilibrées » à chaque appel récursif Complexité temporelle $\Theta(n \cdot \log(n))$	
Cas défavorable	Trié à l'envers Complexité temporelle $\Theta(n^2)$	Trié à l'envers Complexité temporelle $\Theta(n^2)$	Les deux sous-listes sont de taille 0 et n-1 à chaque appel récursif Complexité temporelle $\Theta(n^2)$	
Cas moyen	Répartition uniforme Complexité temporelle $\Theta(n^2)$	Répartition uniforme Complexité temporelle $\Theta(n^2)$	Complexité temporelle $\Theta(n \cdot \log(n))$	Complexité temporelle $\Theta(n \cdot \log(n))$
Graphique Echelle linéaire (Moyenne sur 50 listes)				
Graphique Echelle log-log (Moyenne sur 50 listes)				

Tri par insertion :

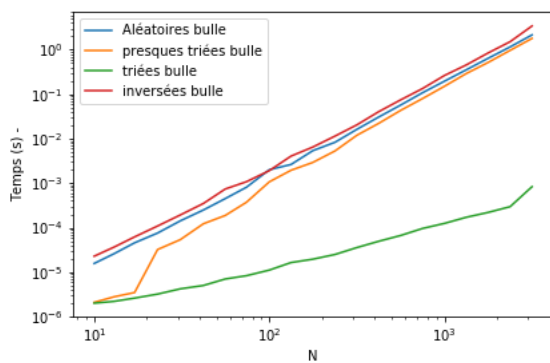


On obtient une pente de 1 pour les listes triées (diagramme log/log) : Complexité temporelle $\Theta(n)$

On obtient une pente de 2 pour les listes inversées (triées par ordre décroissant) : Complexité temporelle $\Theta(n^2)$

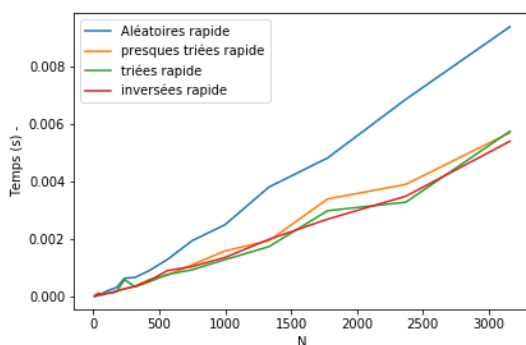
Légèrement en dessous, avec une pente de 2, les listes aléatoires donne aussi une complexité temporelle $\Theta(n^2)$.

Tri bulles :



On obtient une pente de 2 pour toutes les listes : Complexité temporelle $\Theta(n^2)$. Remarque, ce n'est pas flagrant ici pour les listes triées, plus rapides.

Tri rapide



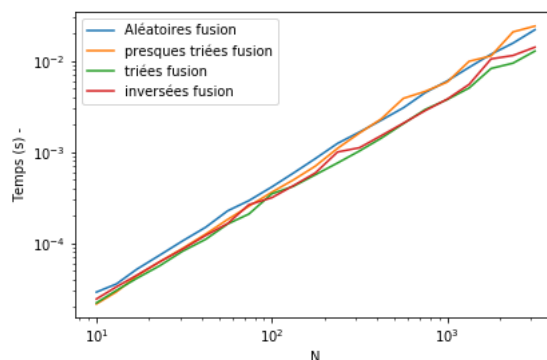
Le cas aléatoire dégrade les performances de ce tri, avec une pente de 2, soit une complexité temporelle $\Theta(n^2)$.

On a une allure $n \log n$.

Tri fusion

Une première remarque, les performances ne dépendent pas du type de liste.

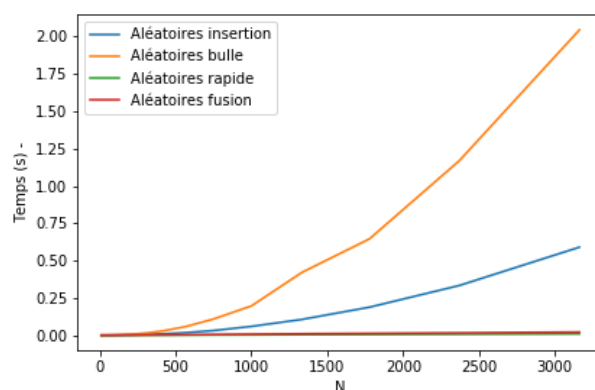
On a une allure $n \log n$.



Comparaison des tris.

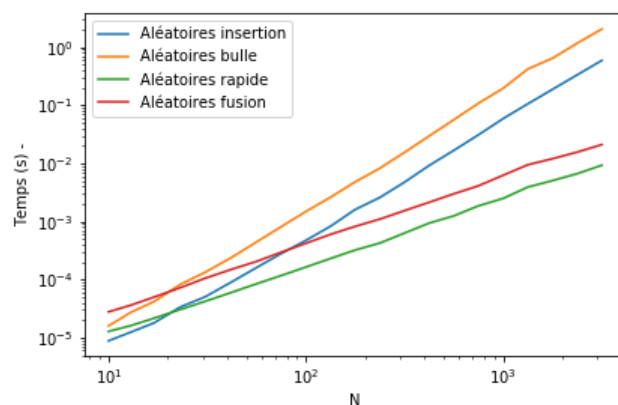
Le tri bulle est le plus lent, quel que soit la configuration des listes, dès que l'on dépasse 20.

Pour des listes grandes, le tri rapide et le tri fusion sont beaucoup plus rapides (ils ont un temps relativement très faibles et sont confondus avec l'axe des abscisses).

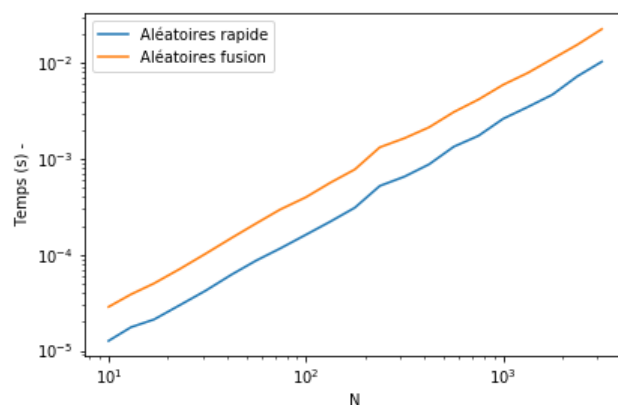


On retrouve la complexité temporelle $\Theta(n^2)$ pour le tri bulle et le tri insertion sur les listes aléatoires.

Le tri rapide et le tri fusion ont une allure $n \log n$.

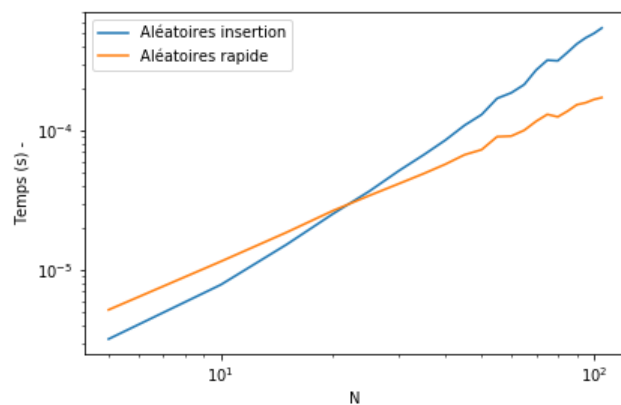


On constate que le tri rapide est plus efficace sur les listes générées (nombres entiers aléatoires)



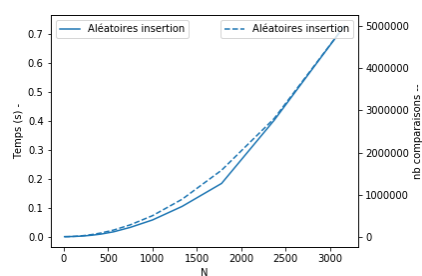
Remarque : pour des listes de petites tailles, le tri par insertion est plus efficace.

Environ pour une taille de 20, on a égalité entre les deux tris.

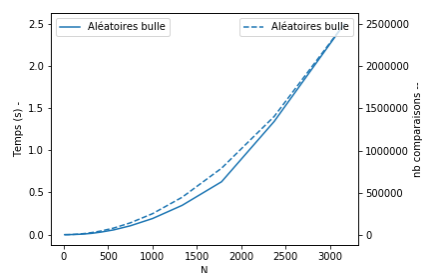


Pour les opérations fondamentales, on a une superposition des courbes du temps et du nombre de comparaisons :

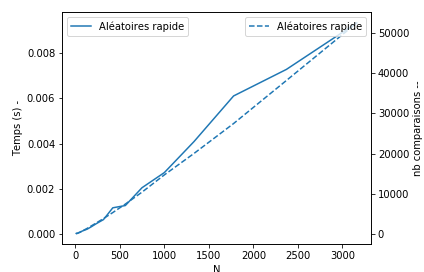
Pour le tri par insertion



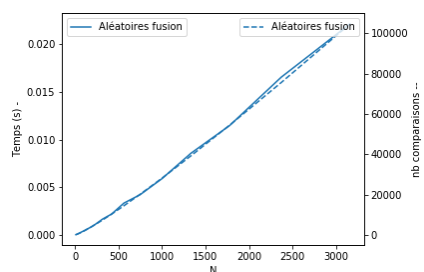
Pour le tri bulles



Pour le rapide

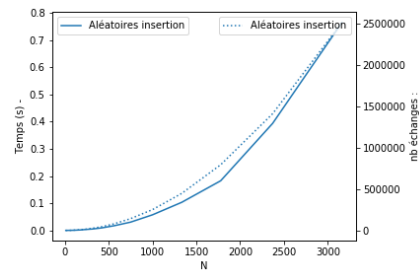


Et pour le tri fusion



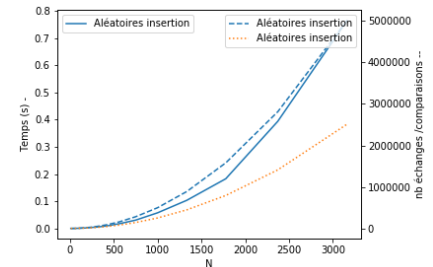
De même pour les échanges dans un tableau :

(un seul exemple , ici le tri insertion)



Lorsqu'on met sur le même graphique temps/ compteur échanges / compteurs de comparaisons :

On constate que le nombre de comparaisons est plus important que le nombre d'échanges.



Projets : deux versions :

