Clément Riou L2G2

Rendu projet algo tri

Différents fichiers du projet :

- Main.py : Ce fichier contient les tests des différents algos, leur temps d'exécution et leurs écart type.
- Algo.py : Ce fichier contient tous les algorithmes de tri ainsi que les fonctions intermédiaires pour leur fonctionnement.

Différents algos du projet :

Tri par insertion :

Complexité dans le meilleur cas : O(n)

Complexité dans le pire cas : O(n^2)

Complexité en moyenne : O(n^2)

Tri à bulles :

Complexité dans le meilleur cas : O(n)

Complexité dans le pire cas : O(n^2)

Complexité en moyenne : O(n^2)

Tri par éclatement (ou tri fusion) :

Complexité dans le meilleur cas : O(n log n)

Complexité dans le pire cas : O(n log n)

Complexité en moyenne : O(n log n)

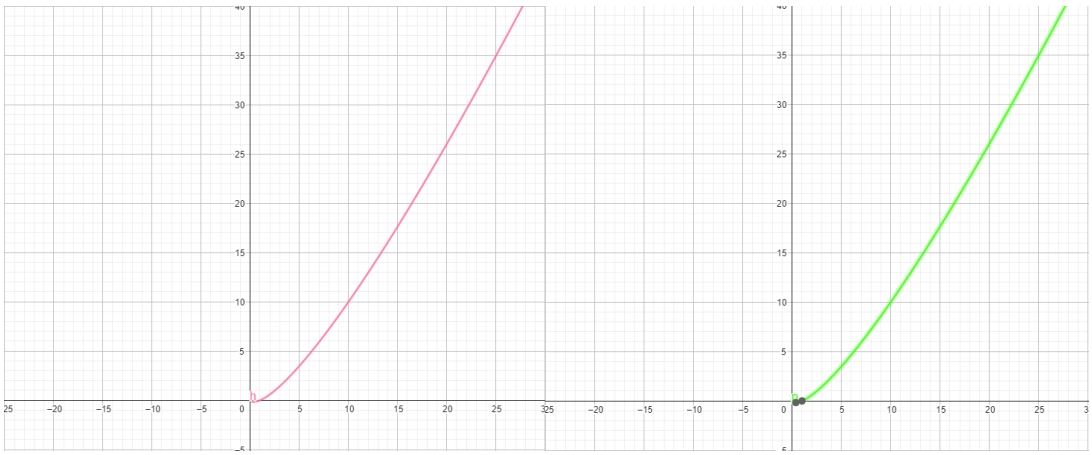
Tri rapide (quicksort) :

Complexité dans le meilleur cas : O(n log n)

Complexité dans le pire cas : O(n^2)

Complexité en moyenne : O(n log n)

Différentes courbes des algos pour la complexité moyenne :



Quels Algorithmes privilégier en fonction de la taille de la liste :

Pour de petites listes : les algorithmes de tri avec une complexité quadratique, tels que le tri par insertion et le tri à bulles, peuvent être appropriés. Malgré qu'ils aient une complexité pire cas de O(n^2), leurs performances sont souvent acceptables pour de petites tailles de données.

Pour des listes de taille moyenne à grande : les algorithmes avec une complexité de O(n log n) tels que le tri par éclatement (ou tri fusion) et le tri rapide sont généralement préférés. Ces algorithmes offrent de meilleures performances pour des tailles de données plus importantes. Le tri fusion est stable et peut être plus adapté lorsque la stabilité est requise. Le tri rapide est en général plus rapide en moyenne, mais il est non stable.

Pour de très grandes listes ou des situations nécessitant une grande efficacité : Lorsque les performances sont une priorité absolue et que la taille de la liste est très grande, des algorithmes plus avancés et spécialisés peuvent être envisagés.

Expérimentations :

Mes algorithmes n'étant pas correctement faits pour tri_rapide et eclate_fusion, je n'ai pas pu visualiser les résultats. Cependant, les algorithmes marchent pour des listes très petites et pour des listes plus grandes avec les deux autres algorithmes.

Résultats pour les algorithmes qui fonctionnent :

```
Python Console
Liste chainée générée aléatoirement avec 500 éléments:
     (9,(28,(77,(75,(38,(27,(27,(57,(52,(33,(67,(48,(55,(96,(42,(69,(15,(5,(8,(77,(60,(16,(90,(39,(57,(1,(6,(81,(2,(62,(64,(56,(83,(65,(2,(98,(26,(13,(77,(35,(81,(63,(81,(78,(37,(96,(79,(66,(67,(18,
     28.069600923418907
Liste croissante générée aléatoirement avec 500 éléments:
     (88,(96,(98,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,
     0.5715802655795565
Liste décroissante générée aléatoirement avec 500 éléments:
     (87,(7,(1,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,
     3.898961400168032

TEST DES ALGOS AVEC DES LISTES ALEA

 Test de la fonction : tri_par_insertion
     (1,(1,(1,(1,(2,(2,(2,(2,(2,(2,(3,(3,(3,(3,(3,(3,(3,(4,(4,(5,(5,(5,(5,(5,(5,(6,(6,(7,(7,(7,(8,(8,(8,(9,(9,(9,(9,(9,(9,(9,(9,(10,(10,(10,(11,(11,(11,(11,(11,(12,(12,(12,(12,(12,(12,(12,(13,(13,(
     Temps d'execution : 0.022938013076782227

 Test de la fonction : tri_bulles_rec
     (1,(1,(1,(1,(2,(2,(2,(2,(2,(2,(3,(3,(3,(3,(3,(3,(3,(4,(4,(5,(5,(5,(5,(5,(5,(6,(6,(7,(7,(7,(8,(8,(8,(9,(9,(9,(9,(9,(9,(9,(9,(10,(10,(10,(11,(11,(11,(11,(11,(12,(12,(12,(12,(12,(12,(12,(13,(13,(
     Temps d'execution : 0.025931835174560547

TEST DES ALGOS AVEC DES LISTES CROISSANTES

 Test de la fonction : tri_par_insertion
     (88,(96,(98,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,
     Temps d'execution : 0.052880048751831055

 Test de la fonction : tri_bulles_rec
     (88,(96,(98,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,(100,
     Temps d'execution : 0.017956972122192383
TEST DES ALGOS AVEC DES LISTES DÉCROISSANTES

 Test de la fonction : tri_par_insertion
     (0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,
     Temps d'execution : 0.05272030830383301

 Test de la fonction : tri_bulles_rec
     (0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,(0,
     Temps d'execution : 0.013962268829345703
```