Compte rendu - Projet programmation fonctionnelle

Dorine Descamps et Marwan Ait Addi

6 mai 2019

Table des matières

Afin de faire un choix de liste cohérent, nous avons testé le temps que mettaient nos fonctions à trier des listes en faisant changer plusieurs variables. L'ordre étant une variable possible nous gardons l'ordre croissant pour tous les autres tests. Nous arrondirons les temps en fonctions des résultats, l'unité étant la seconde.

1 Expérience

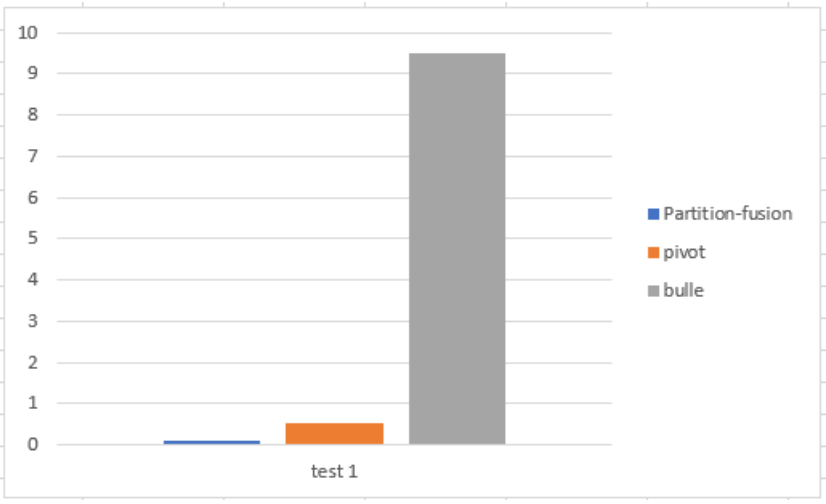
1.1 Comparaison du temps des fonctions de tri en augmentant la taille des listes

Nous nous plaçons dans une fourchette d'entier de 0 à 100 pour le choix des éléments de la liste et augmentons la taille de la liste.

nombre d'éléments de la liste	10	100	1000	10 000	25 000	50 000
temps tri_partition_fusion	0	0	0.003	0.043	0.13	0.29
temps tri_pivot	0	0	0.002	0.051	0.36	1.83
temps tri_bulle	0	0.002	0.2	28.34	cf remarque	

Remarque : Nous ne traitons plus la fonction tri à bulle pour les listes de 25 000 et 50 000 éléments,2 sachant que les temps seront énormes (compté en minute voire dizaine de minutes).

Plus visuellement :



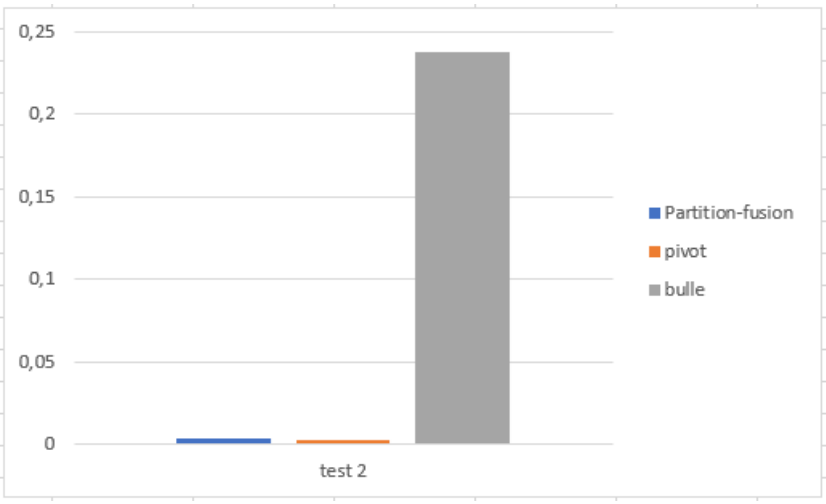
Il apparaît déjà que le tri à bulle est la manière de trier la moins optimisé sur de grande liste mais aussi que la fonction la plus rapide semble être le tri par partition-fusion.

1.2 Comparaison du temps des fonctions de tri en augmentant la fourchette d'éléments possible

Cette fois-ci, nous gardons la taille de la liste constante à 1000 éléments et augmentons la fourchette d'entier possible.

fourchette d'éléments de la liste	50	500	5000	10 000
temps tri_partition_fusion	0.003	0.003	0.003	0.006
temps tri_pivot	0.002	0.002	0.002	0.004
temps tri_bulle	0.22	0.25	0.23	0.25

Plus visuellement :



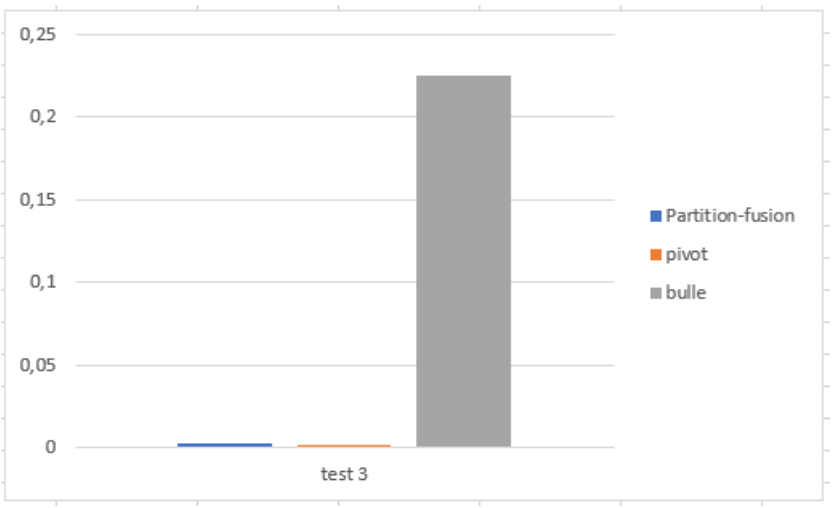
Nous pouvons remarqué que la fourchette d'éléments n'est pas une variable très important, les résultats sont relativement stables même si le pivot est le plus rapide.

1.3 Comparaison du temps des fonctions de tri en fonction de l'ordre

Pour cette partie nous prenons une liste de 1000 éléments compris entre 0 et 5000 et changeons l'ordre de tri.

ordre	<	>	<=	>=
temps tri_partition_fusion	0.004	0.003	0.0030	0.002
temps tri_pivot	0.002	0.002	0.0020	0.001
temps tri_bulle	0.23	0.21	0.229	0.23

Plus visuellement :



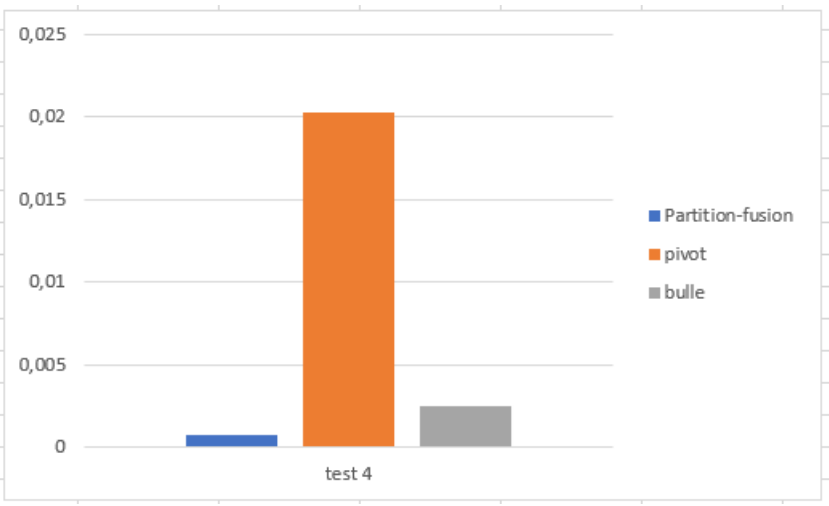
Là aussi, nous remarquons que l'ordre n'impacte pas beaucoup le temps de résolution, les résultats restent plutôt constants et le tri par pivot est toujours plus rapide.

1.4 Comparaison du temps des fonctions de tri en fonction du nombre de doublons

Ici, nous utilisons des listes de 100 éléments entre 0 et 100 modifiés à la main pour avoir la quantité de doublons voulue.

Quantité de doublons	100%	environ 50%	25%	aucun
temps tri_partition_fusion	0.001	0	0.001	0.001
temps tri_pivot	0.08	0	0.001	0
temps tri_bulle	0	0.004	0.005	0.001

Plus visuellement :



Les chiffres de cette expérience ne donnent pas de résultats flagrants, si ce n'est que le tri à bulle est toujours à la traine.

1.5 Comparaison du temps des fonctions de tri avec des listes déjà triées

Pour ces tests nous utilisons des listes déjà triées de 1000 éléments entre 0 et 1000 et nous étudions le réaction des fonctions en fonction de l'ordre donné en paramètre.

1.5.1 Liste trié dans l'ordre croissant stricte (<)

ordre	<	>	<=	>=
temps tri_partition_fusion	0.002	0.003	0.007	0.003
temps tri_pivot	0.08	0.11	0.07	0.1
temps tri_bulle	0	0.22	0	0.24

1.5.2 Liste trié dans l'ordre décroissant stricte (>)

ordre	<	>	<=	>=
temps tri_partition_fusion	0.002	0.003	0.003	0.003
temps tri_pivot	0.07	0.09	0.07	0.07
temps tri_bulle	0.03	0	0.23	0

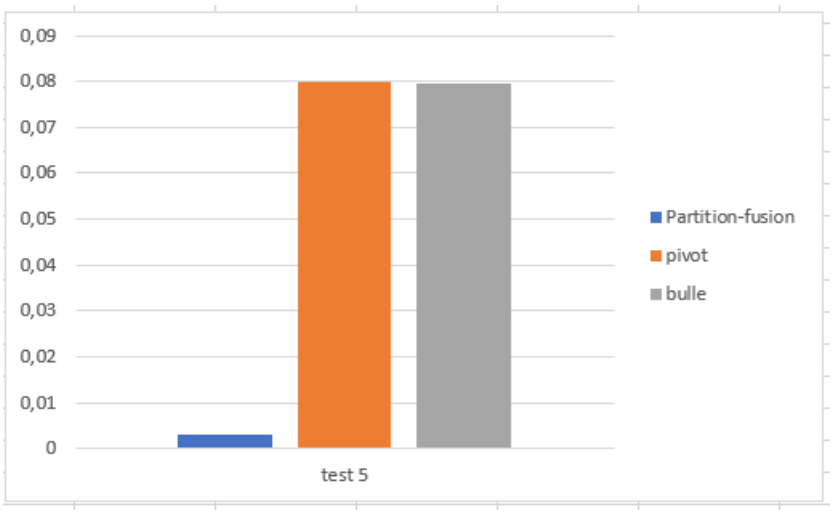
1.5.3 Liste trié dans l'ordre croissant non stricte (<=)

ordre	<	>	<=	>=
temps tri_partition_fusion	0.002	0.002	0.003	0.006
temps tri_pivot	0.09	0.09	0.07	0.1
temps tri_bulle	0	0.001	0	0.25

1.5.4 Liste trié dans l'ordre décroissant non stricte (>=)

ordre	<	>	<=	>=
temps tri_partition_fusion	0.006	0.006	0.003	0.002
temps tri_pivot	0.12	0.07	0.067	0.08
temps tri_bulle	0.2	0.24	0.26	0

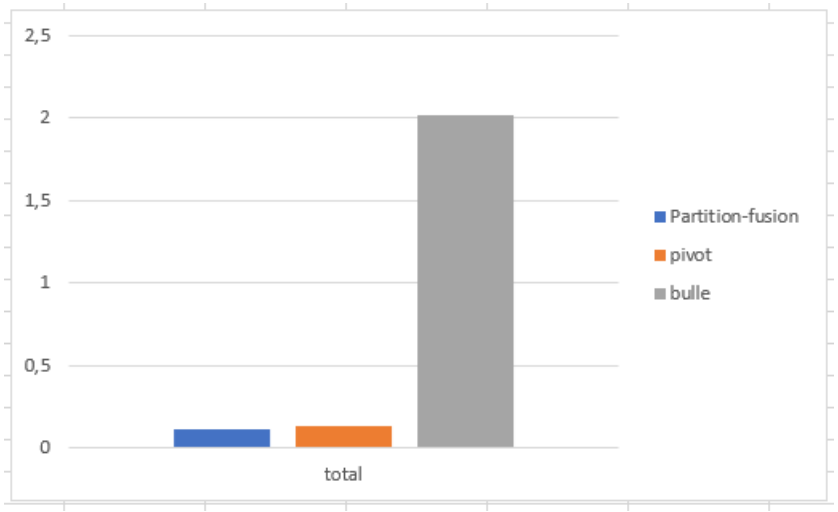
Plus visuellement :



Il apparaît que la fonction bulle est la plus efficace pour trié une liste déjà trié dans le même ordre, ce qui est tout a fait logique mais pour le reste c'est la fonction tri partition fusion qui est la plus rapide.

2 Résultats

En faisant la moyenne de tous les tests on obtient donc :



Il apparaît clairement que le tri a bulle n'est pas optimisé, mais le tri par partition fusion reste légèrement plus rapide que le tri pivot, c'est donc le tri par partition-fusion que nous garderons pour le module de tri des listes.