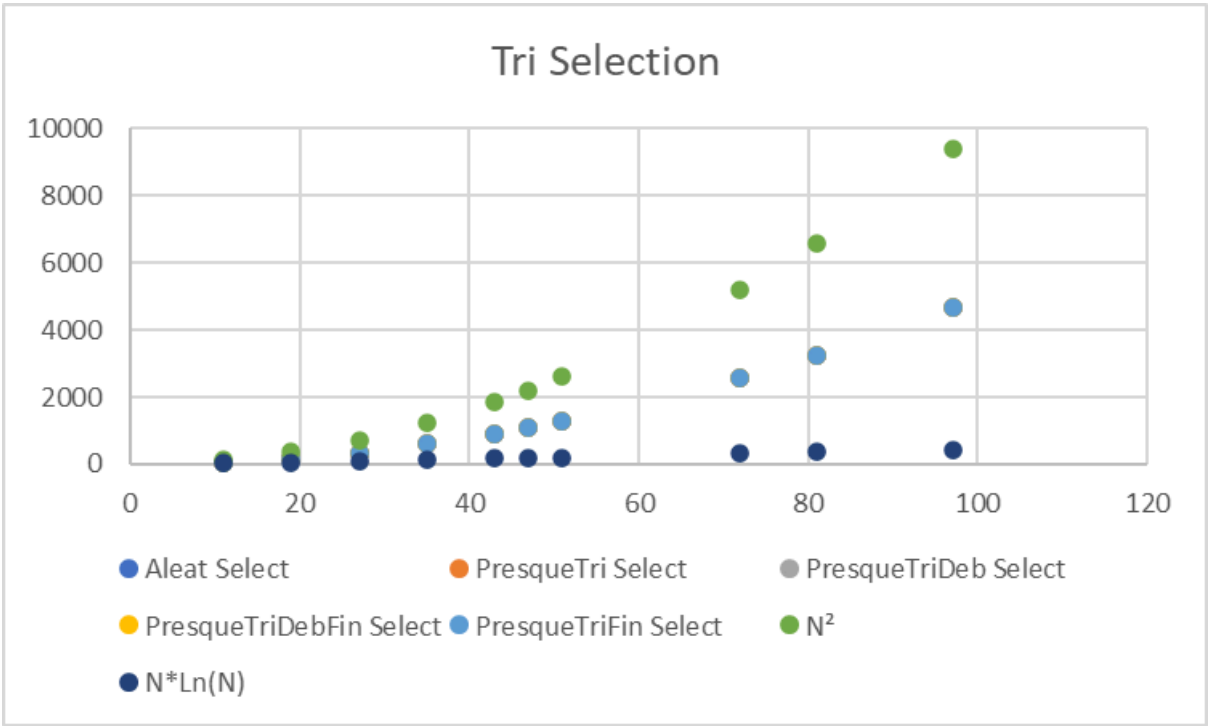
De Robert De Bousquet Emilien
Haen Rémi

SAE S1.02 - Comparaison d'approches algorithmiques

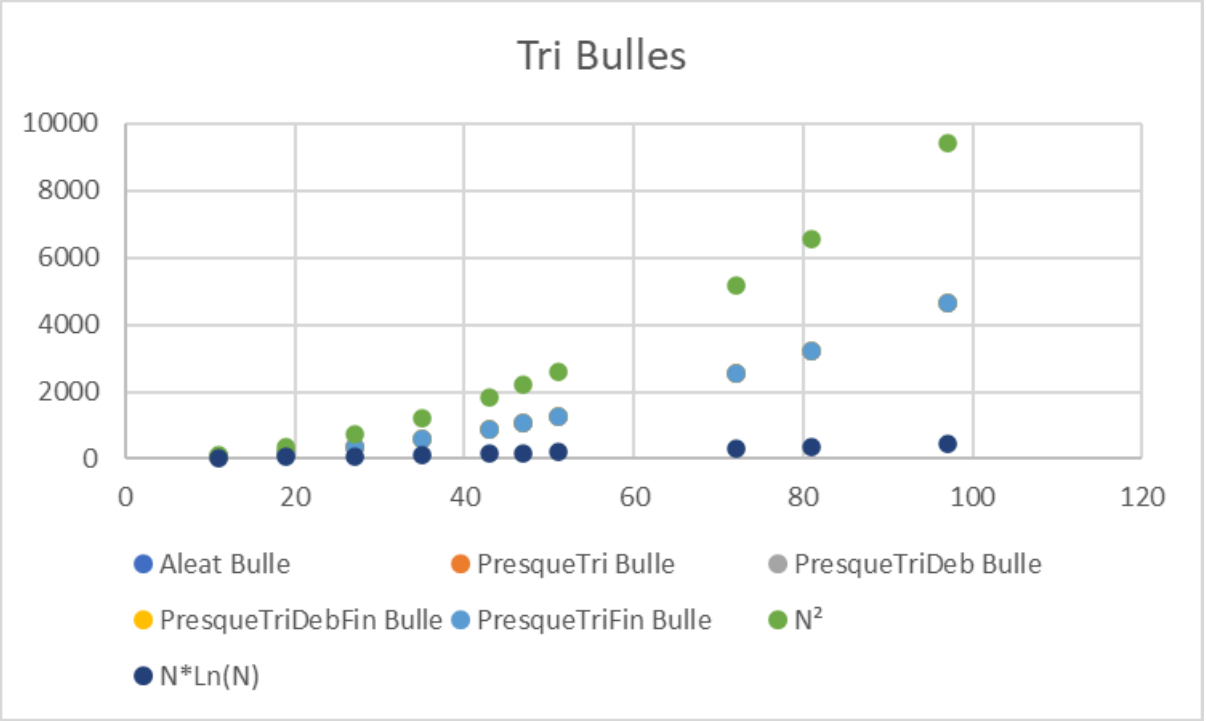
Lors de cette SAE nous avons testé bon nombre d'algorithmes de tri aux fonctionnement divers et aux efficacité différentes. Dans ce document, nous allons observer les différents algorithmes étudiés.
Pour classifier ces algorithmes, nous allons nous pencher sur le nombre de comparaison nécessaire au tri du tableau d'entier. Le nombre de de comparaisons est généralement croissant avec le nombre d'entrées dans le tableau, de fait nous allons mettre en parallèle de l'efficacité de chaque algorithme avec la fonction N^2 et la fonction N*Ln(N).
Les échantillons que nous prenons sont ceux générés sur des tableaux contenant un nombre aléatoire d'éléments entre 2 et 100. Les éléments sont eux aussi aléatoires.

Dans le premier cas, étudions les résultats obtenus lors d'un tri avec **l'algorithme de sélection**. Voici un tableau présentant le nombre de comparaisons en fonction du nombre d'éléments dans le tableau. Le tri par Sélection est stable, peu importe le nombre ou les éléments à trier, il faudra le même nombre de comparaisons pour un même nombre de données (c'est pourquoi toutes les données sont superposées dans le tableau). Cependant, nous pouvons remarquer que pour un nombre important de données, l'algorithme nécessite un grand nombre de comparaisons. Ainsi en comparant les résultats obtenus lors du Tri par sélection avec un grand nombre d'éléments (N^2) avec une valeur optimale (N*Ln(N)), nous pouvons en déduire que cet algorithme n'est pas très performant sur un grand nombre de données.

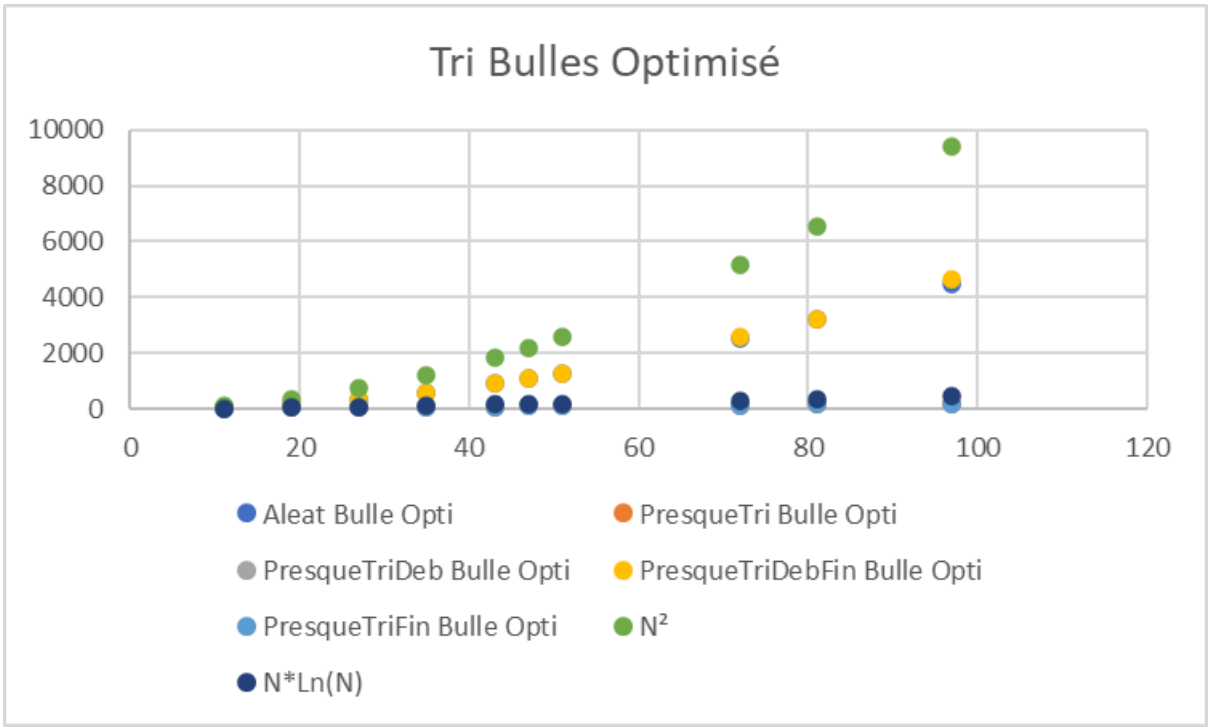


De Robert De Bousquet Emilien
Haen Rémi

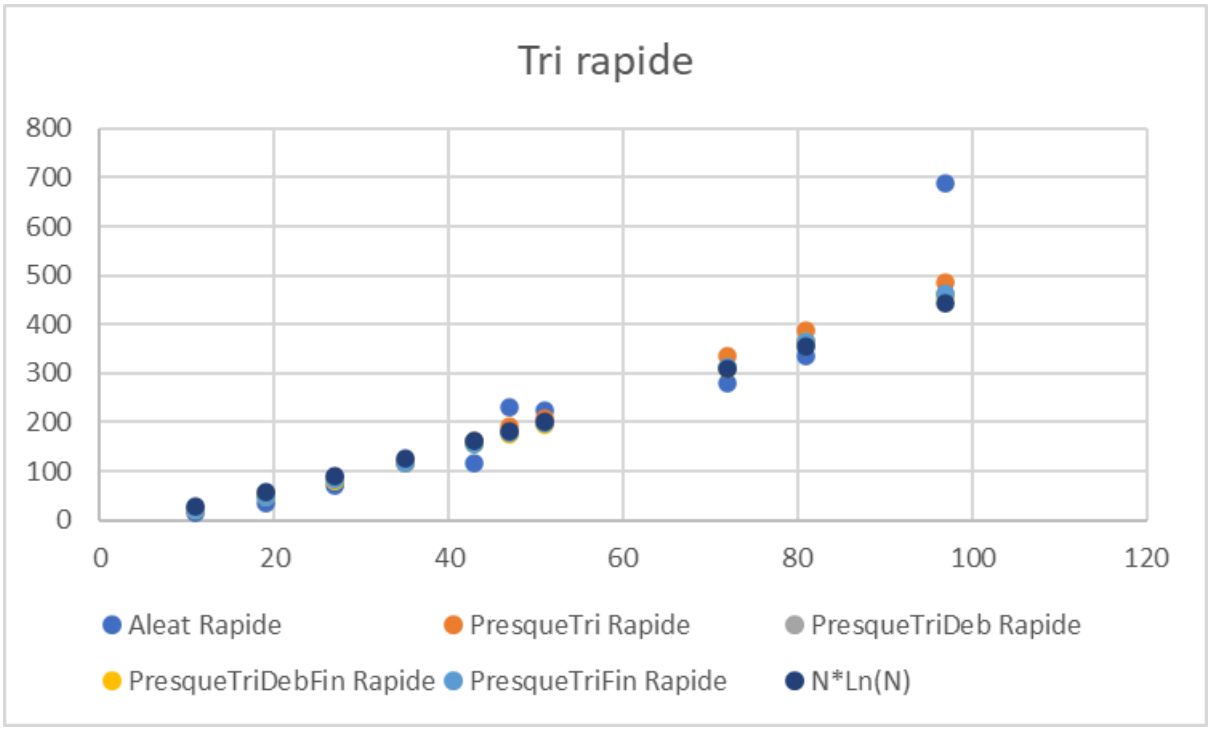
Étudions dans un second cas, les résultats du tri avec l'algorithme du **tri à bulles**. Les résultats obtenus sont les mêmes qu'avec le tri sélection. Ainsi nous pouvons conclure que cet algorithme est aussi peu performant que l'algorithme de sélection sur des grands tableaux.

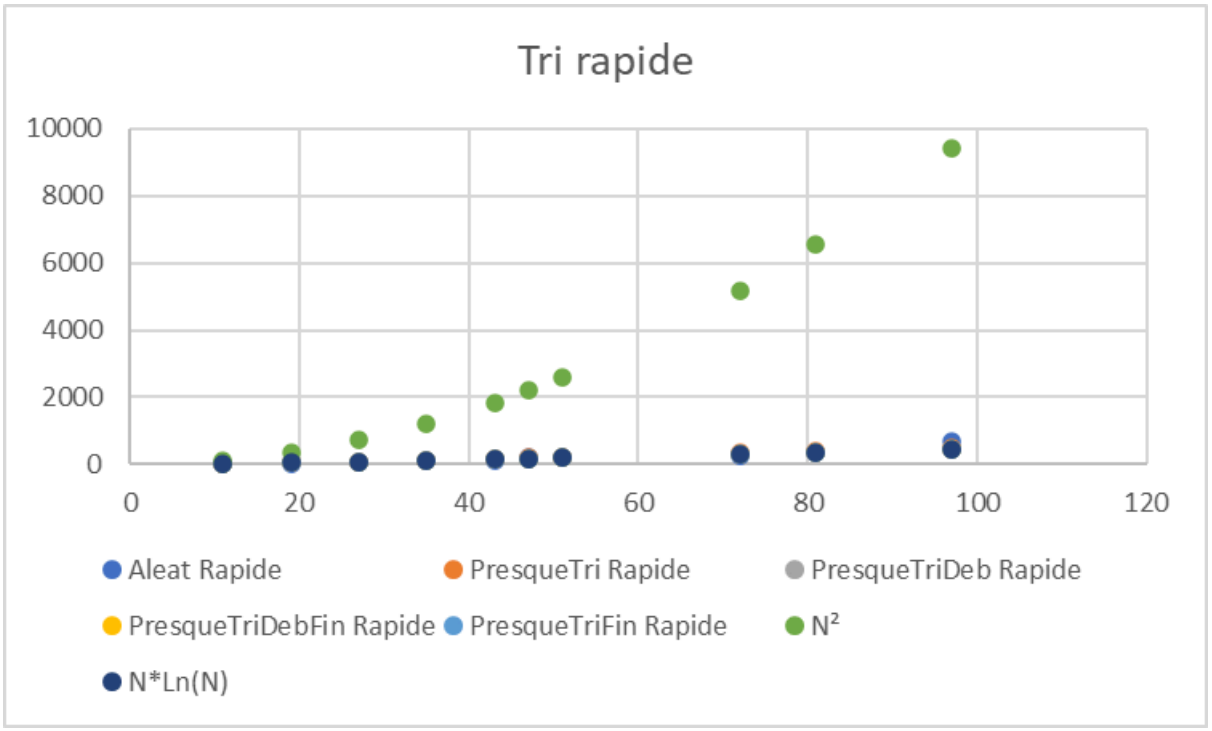


Le **tri à bulles** possède une version **optimisée**, les résultats diffèrent légèrement sur la plupart des tableaux, or on observe une variation du comportement sur les tableaux dit "presque trié", dans le cas où peu de données sont à trier, ce tri devient plus efficace.

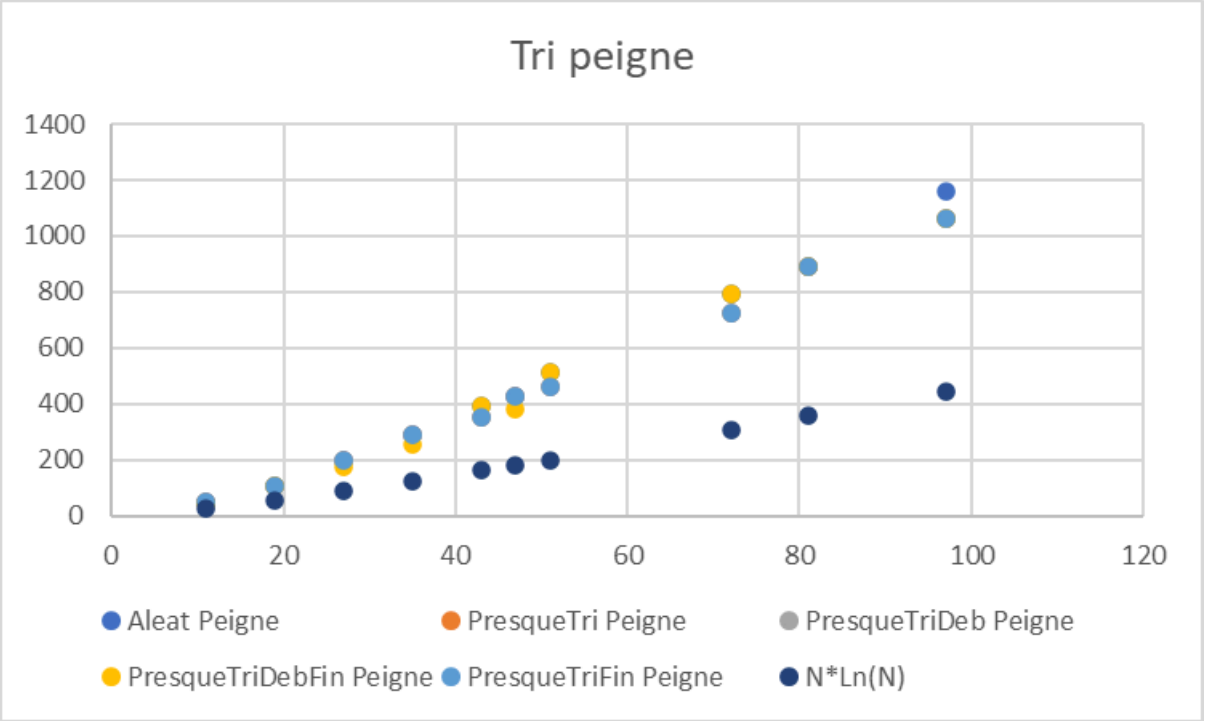


Étudions maintenant les résultats obtenus par l'algorithme de **Tri Rapide**, On observe que cet algorithme (et plusieurs autres) s'éloigne de plus en plus de la complexité N², ce qui signifie qu'ils sont plus efficaces en nombre de comparaisons. On observe que les résultats obtenus se rapproche de la fonction N*Ln(N), ainsi, pour un grand nombre de données à traiter, l'algorithme de Tri Rapide se montrera plutôt efficace.

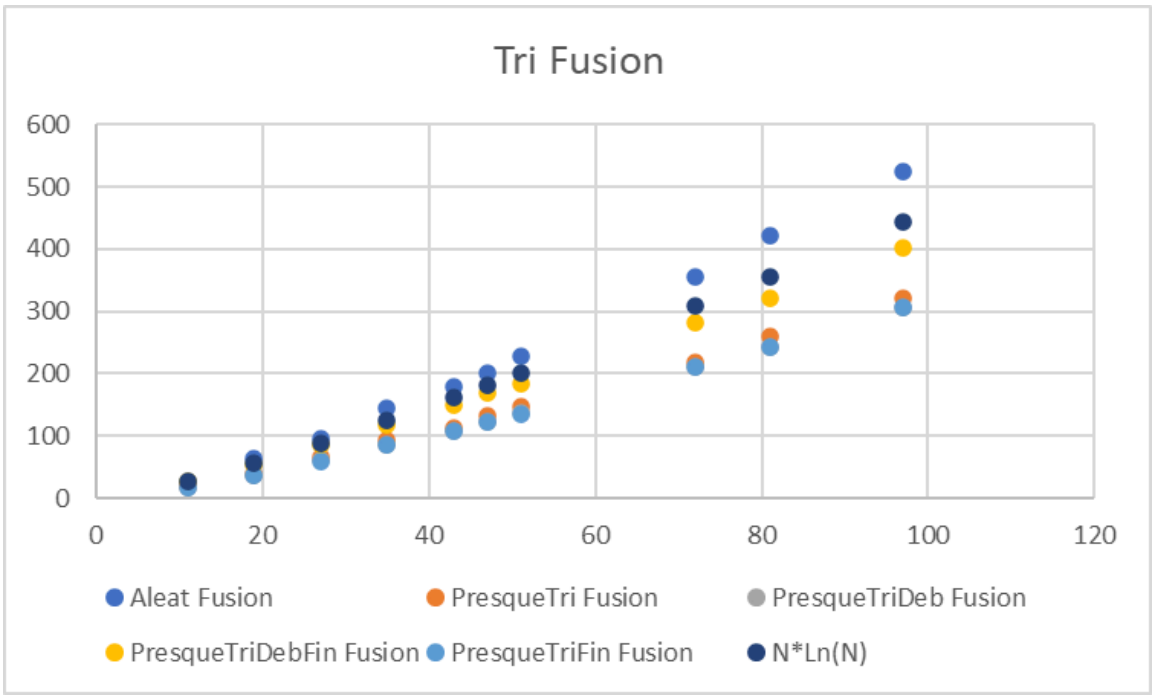




Le **tri à peigne** est également rapproché de la fonction N*Ln(N) et est plus éloigné de N² que les premières fonctions. De fait, cet algorithme possède une bonne optimisation par rapport au tri selection, cependant, on observe qu'il est près de deux fois moins efficace que le tri rapide, notamment lorsque le nombre d'entrées augmente. On observe aussi que les résultats se chevauchent la plupart du temps ce qui en fait un tri efficace linéairement aux nombre d'entrées du tableau.

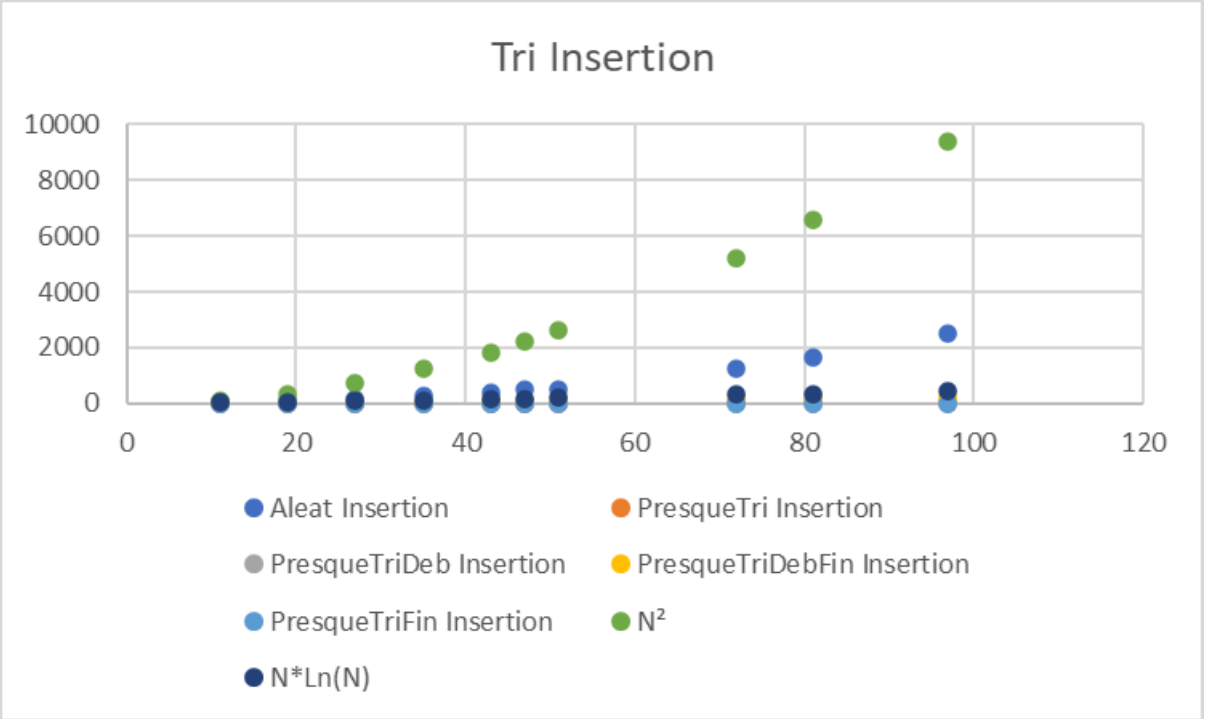


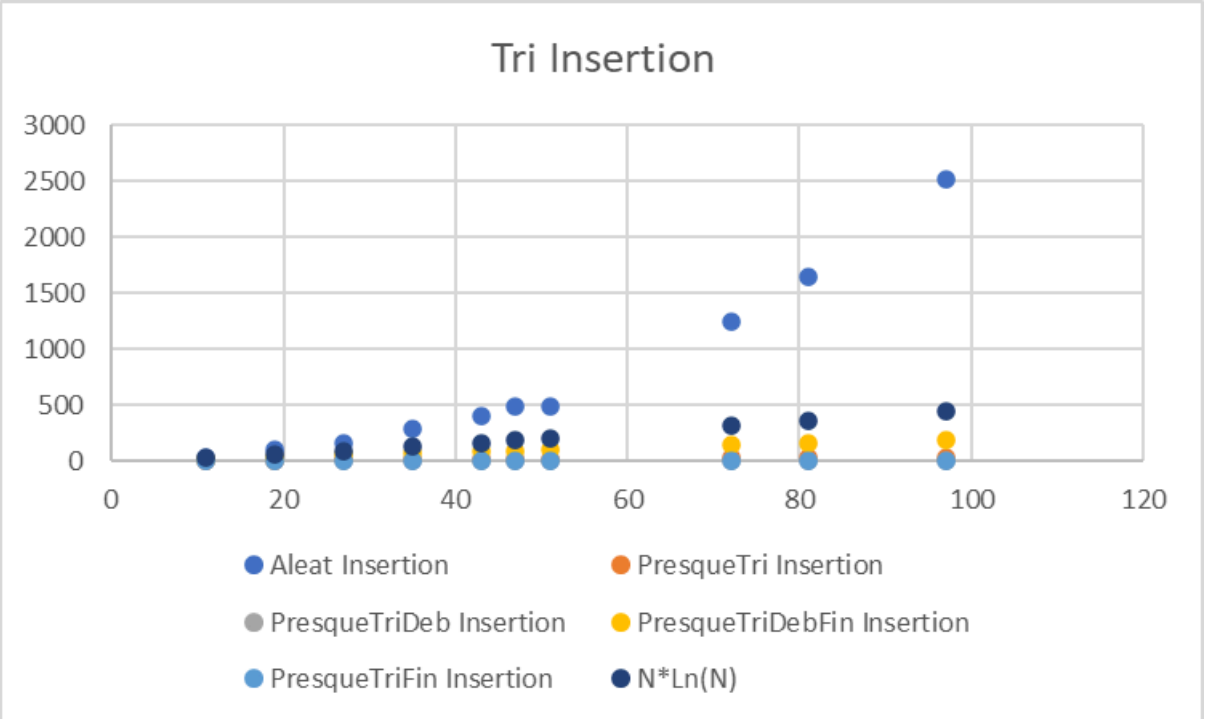
Prenons maintenant le cas du **tri Fusion**, l'allure de la courbe du nombre de comparaisons est très proche dans tous les cas de figure de l'allure de la courbe de N*Ln(N). On peut également notifier que le tri fusions est plus de 2 fois plus rapide que le tri peigne et légèrement plus efficace que le tri rapide.



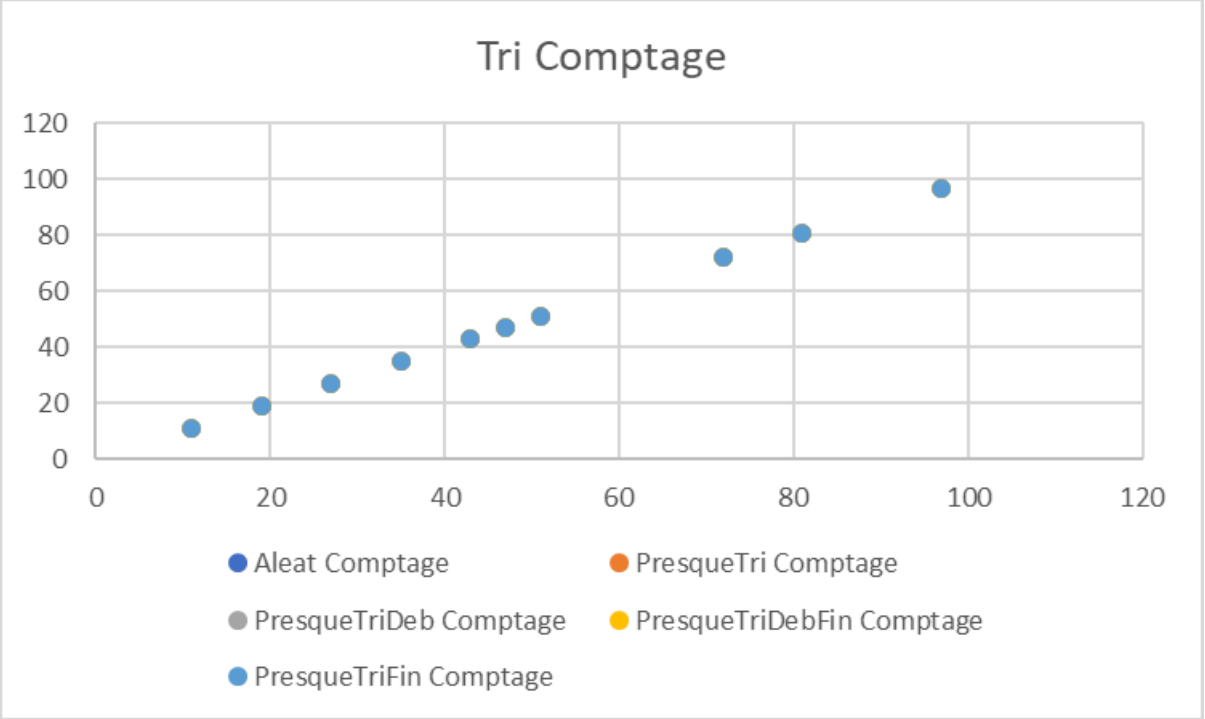
De Robert De Bousquet Emilien
Haen Rémi

Le tri par insertion est quant à lui efficace avec peu de valeurs à comparer, cependant, on observe une augmentation rapide lorsque le nombre de données augmente, le nombre de comparaisons augmente dans le cas où le tableau est généré aléatoirement. Néanmoins, il reste plutôt efficace pour les tableaux partiellement triés.

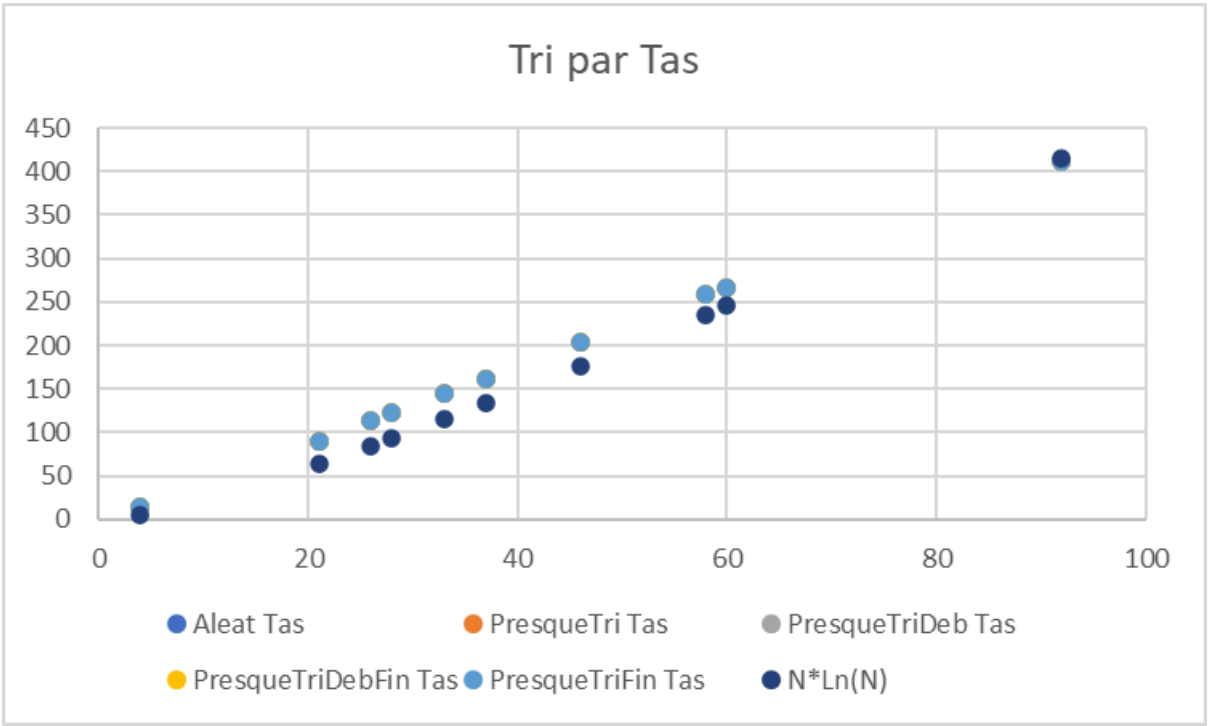




L'algorithme de comptage est linéaire voit son nombre de comparaisons augmenter de manière linéaire. Cependant le nombre de comparaison reste assez faible pour un grand nombre de données. L'absence de la fonction N*Ln(N) est ici justifiée par le fait que ce tri possède un nombre de comparaison bien inférieur.



Finalement, Nous pouvons remarquer sur le **tri par Tas**, la linéarité des résultats, ainsi, plus il y aura de données à traiter, moins celà sera efficace. Cependant, il faut noter que les résultats sont assez bon, comme peut en témoigner la fonction N*Ln(N) qui prouve une progression du nombre de comparaisons assez lente.



Pour conclure: Après avoir analysé les différents résultats obtenus par les différents algorithmes de tri utilisés, nous pouvons remarquer que l'algorithme de Tri Rapide est l'un des plus efficaces pour traiter un grand nombre de données efficacement. parmi les algorithmes que nous avons eut à analyser en premier. Cependant, sur la taille de l'échantillon que nous avons prit, nous avons pu observer le tri par comptage se démarquer de manière assez impressionnante. Néanmoins la linéarité du nombre de comparaison démontre que pour une très grande quantité de données, ce tri deviendra complexe à utiliser.