Girardon Jules
Seitz Nathanaël

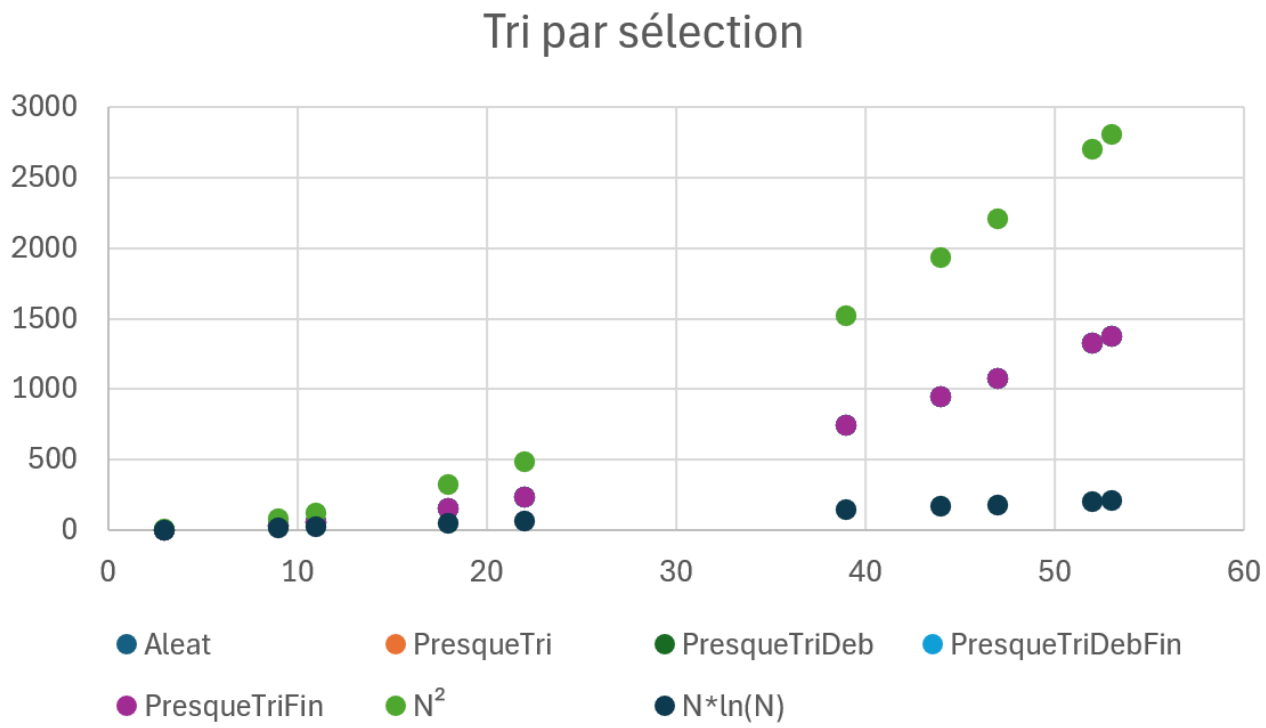
SAE S1.02
Comparaison d'approches algorithmiques

Table des matières

Premier algorithme : tri par sélection

Un algorithme simple qui effectuera dans tous les cas le même nombre de comparaisons, mais considéré comme non stable.
Sur un tableau de n éléments, on va rechercher le plus petit élément puis l'échanger avec le premier élément. Puis on va chercher le deuxième plus petit élément pour le mettre à la deuxième place, et ainsi de place jusqu'à ce que le plus grand élément soit à la place n.

On note sa complexité de cette façon : [n *(n-1)] / 2
C'est donc un tri pratique, mais pas très efficace car les éléments sont déplaçables un par un dans la structure uniquement.

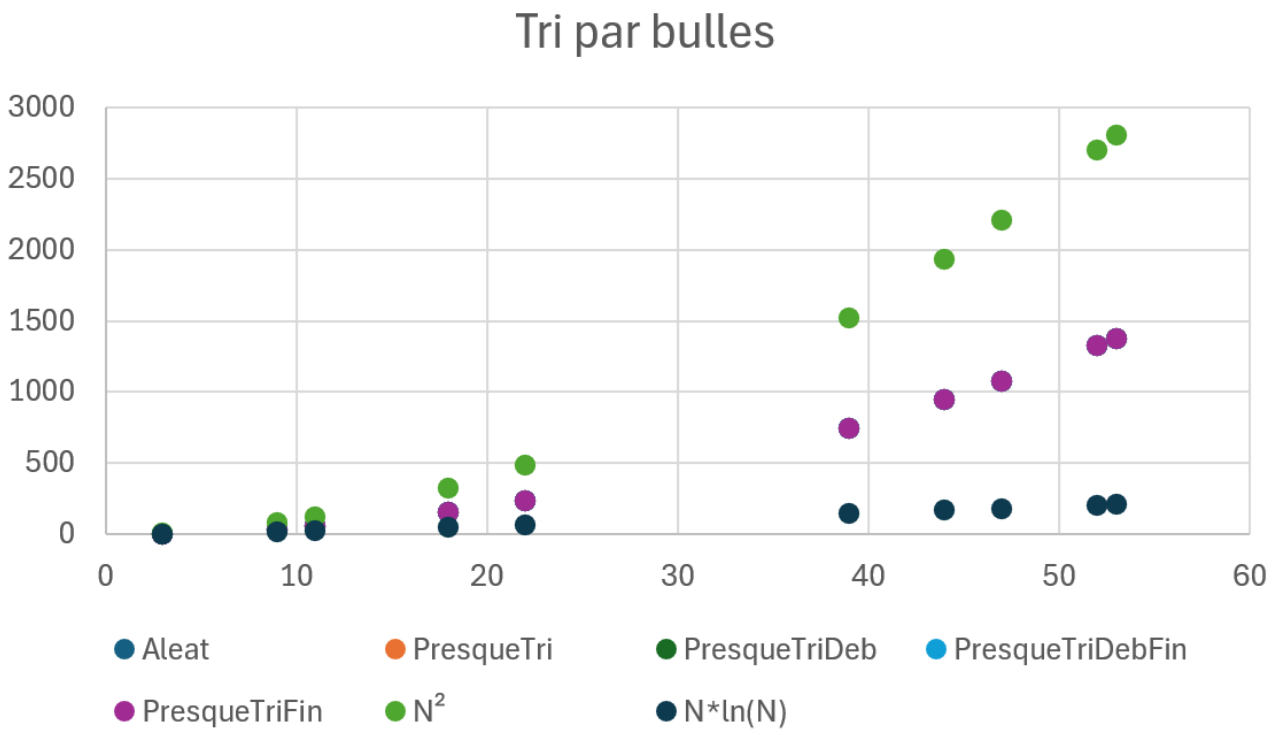


Deuxième algorithme : tri à bulles

Cet algorithme stable va comparer toutes les valeurs puis les déplacer lorsqu'elles rencontrent une valeur supérieure. Ainsi, le tri sera fait du plus grand au plus petit : comme des bulles dans une boisson gazeuse, les valeurs vont « remonter ».

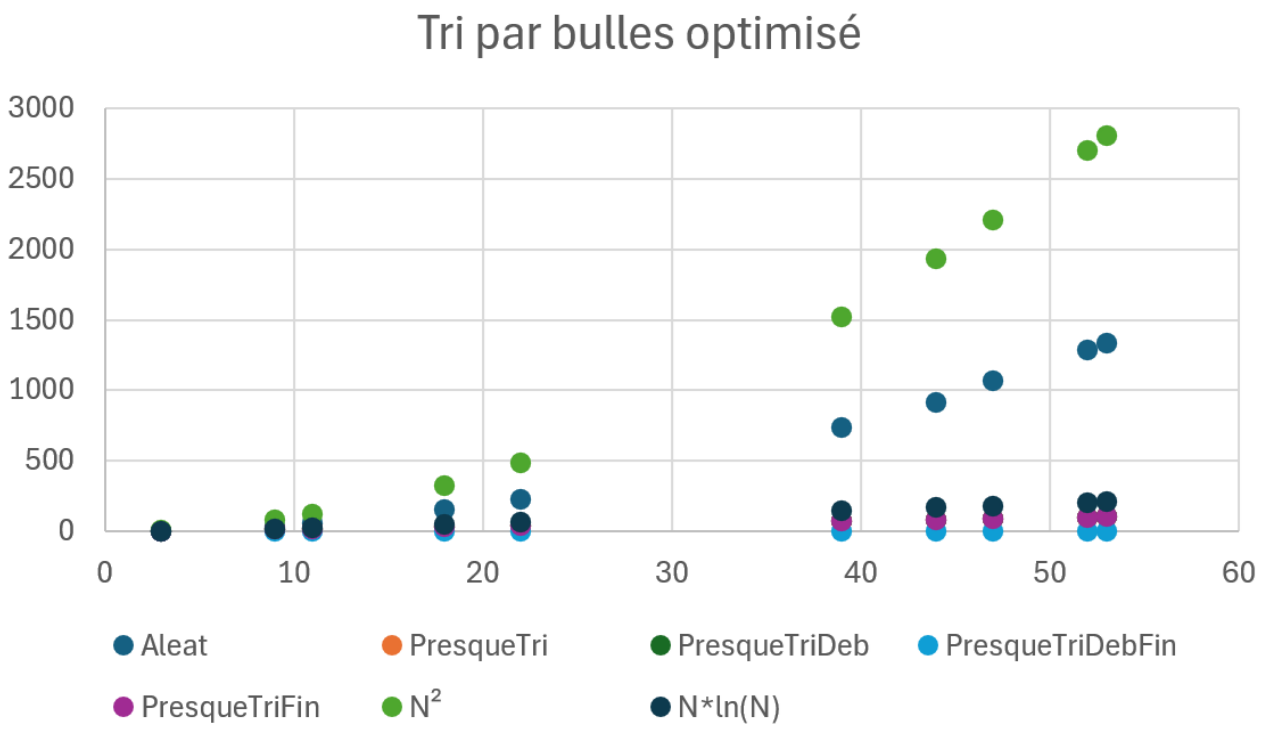
Sa complexité est la même que celle du tri par sélection : [n *(n-1)] / 2

Ce tri est très simple à comprendre, mais c'est l'un des plus lents de tous. Ainsi, il est très travaillé en théorie, mais quasiment jamais mis en pratique.



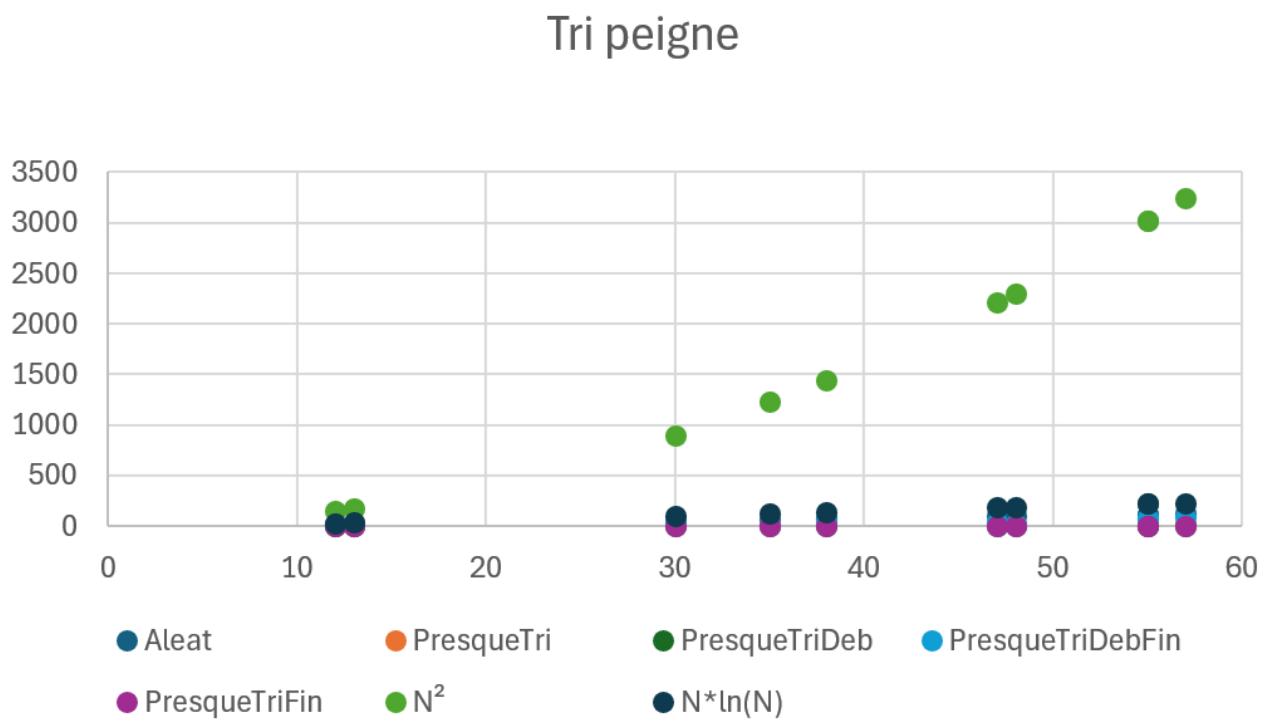
Troisième algorithme : tri à bulles optimisé

Celui-ci est une variante du tri à bulles : on l'optimisera de sorte qu'il s'interrompe dès qu'un élément parcourt la totalité de la liste sans permutation, le rendant ainsi non stable. Dès lors, la liste sera entièrement triée. Sa complexité reste la même : [n *(n-1)] / 2
Cependant, lorsque l'ordre initial des éléments est un aléatoire « parfait », sa complexité sera d'environ [n *(n-1)] / 4
C'est lié au fait que l'espérance sera de 1/2, grâce à l'aléatoire « parfait ».



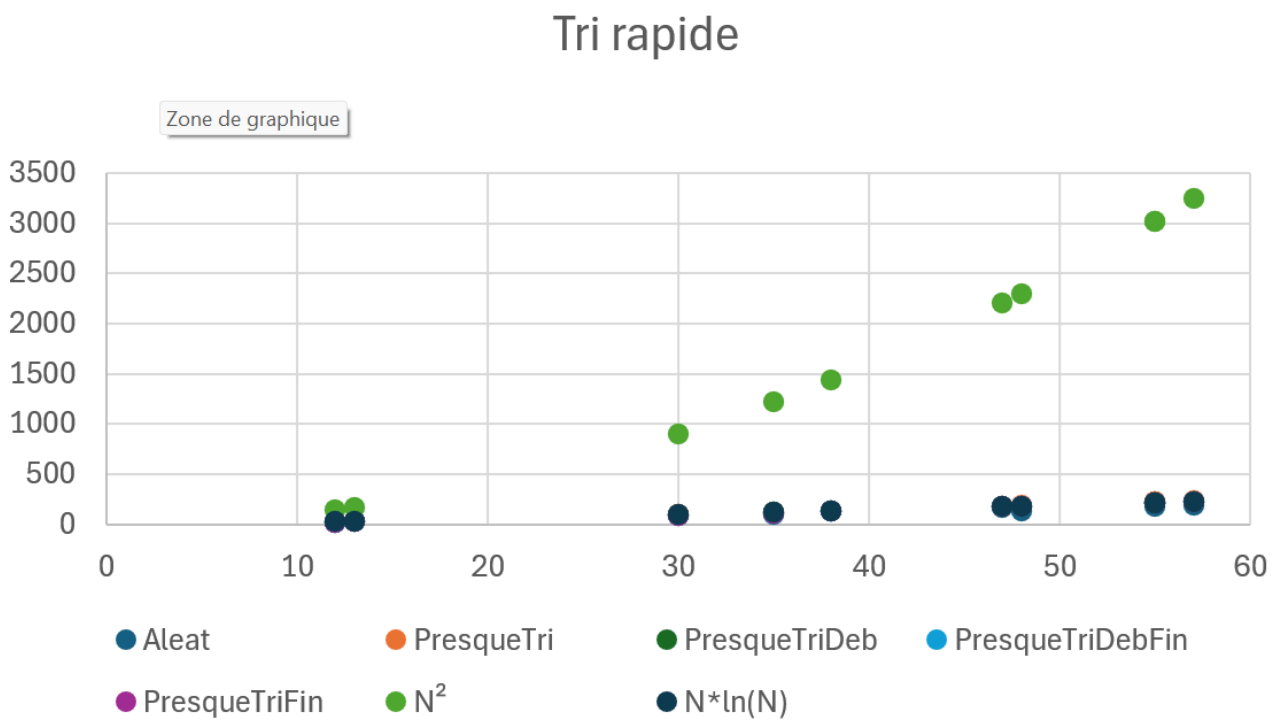
Quatrième algorithme : tri à peigne

Inspiré du tri à bulles, son fonctionnement global reste dans la même idée, mais considéré comme non stable. Cependant, la comparaison n'est pas faite avec la totalité des autres valeurs, mais par intervalles décroissants pour que la vitesse de déplacement des valeurs soit beaucoup plus importante.



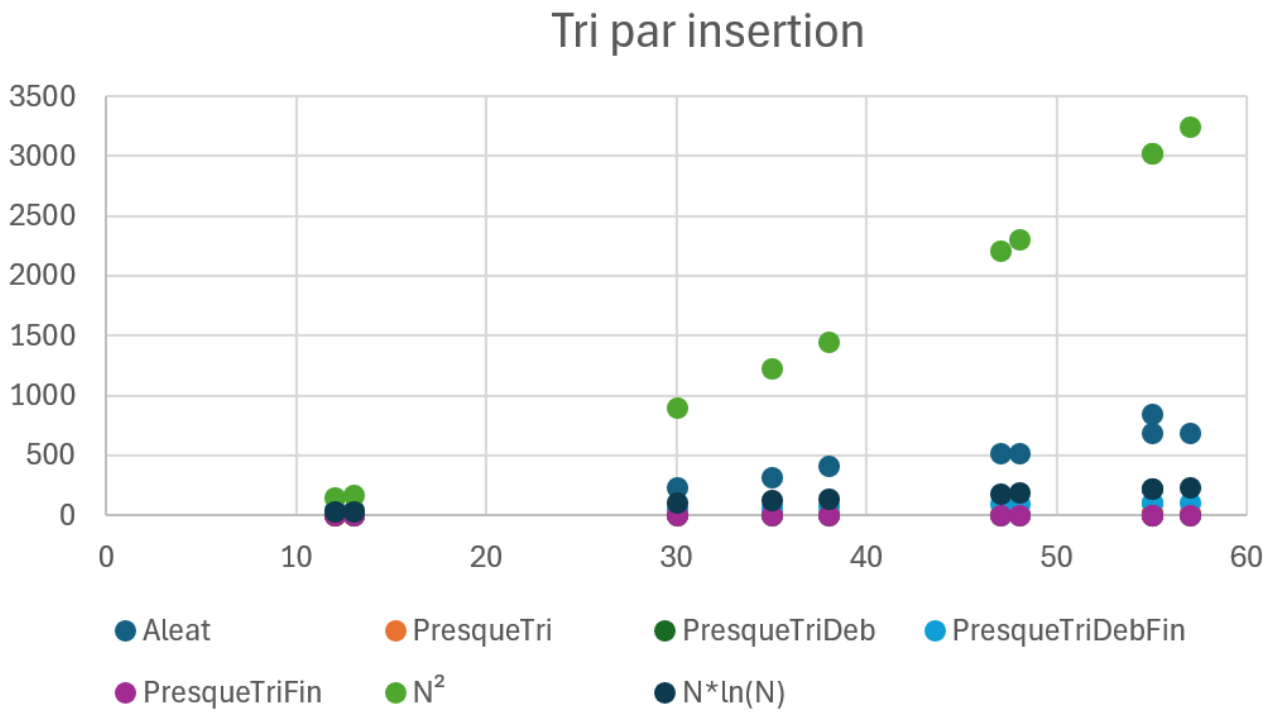
Cinquième algorithme : tri rapide

Cet algorithme non stable a une appellation décrivant son efficacité mais pas sa méthode. On cherche à placer un élément de la liste à sa place définitive avec des permutations, et cet élément sera appelé « pivot ». Puis on permutera toutes les valeurs inférieures à ce pivot à sa gauche, et toutes les valeurs supérieures à droite ; c'est le partitionnement. Puis pour chaque côté, on redéfinira un nouveau pivot et on rappliquera la même méthode encore et encore jusqu'à ce que tous les éléments soient triés.
On note sa complexité de cette façon : O(n*log(n))



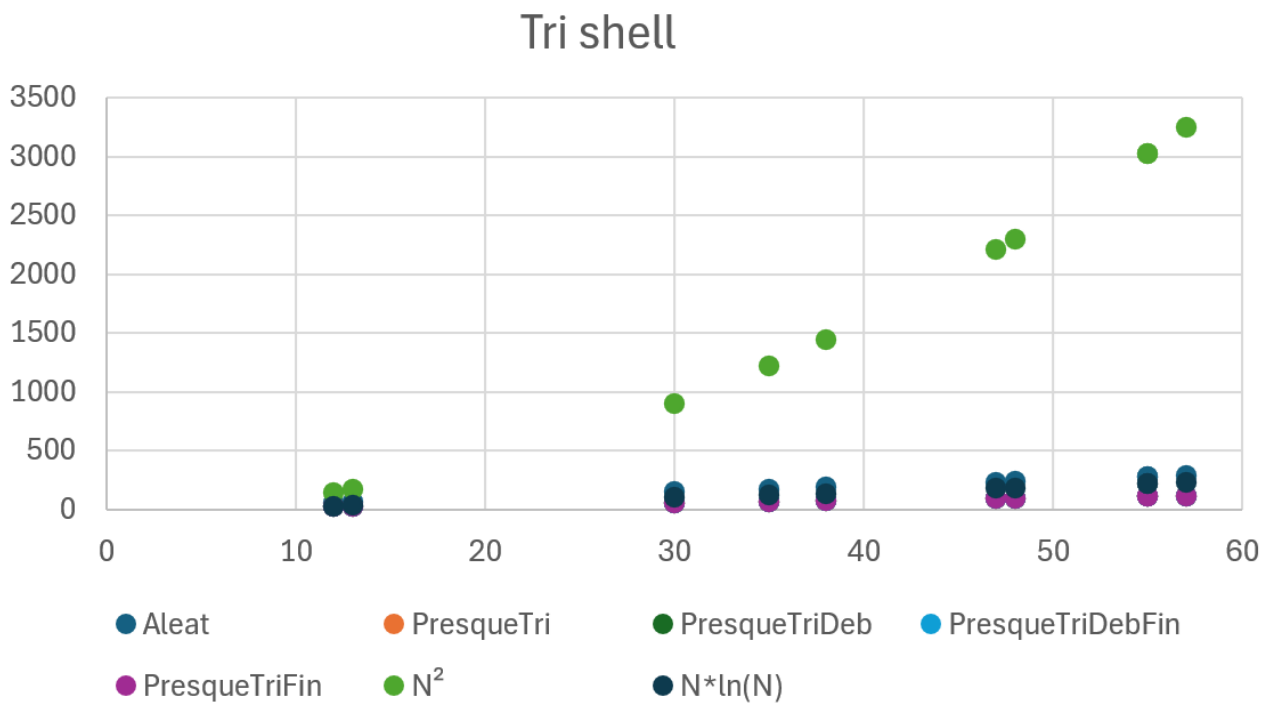
Sixième algorithme : tri par insertion

C'est l'algorithme stable considéré comme le plus efficace pour des entrées de taille minime, par exemple pour le tri d'un jeu de cartes.
Il fonctionne de sorte que chaque élément soit comparé aux éléments précédents, qui seront triés auparavant. On démarrera en comparant la deuxième valeur à la première, et chaque décalage laissera un emplacement vacant pour qu'il puisse être utilisé par la valeur correspondante (ex : emplacement 8 vide jusqu'à ce que la 8ème valeur soit déplacée).



Septième algorithme : tri de Shell

Grosse amélioration du tri par insertion, il tire son nom de son inventeur Donald Shell. Sa méthode est de trier chaque liste d'éléments séparés de n positions avec la méthode du tri par insertion en réduisant n à chaque fois jusqu'à n=1. De cette manière, tous les éléments seront triés, mais l'algorithme n'est pas stable.
Commencer avec l'espacement des éléments permet d'éviter l'inconvénient du tri par insertion, qui est de ne déplacer les valeurs qu'une par une. La dernière itération à n=1 est exactement un tri par insertion.
Sa complexité est la suivante : O(n*log(n))

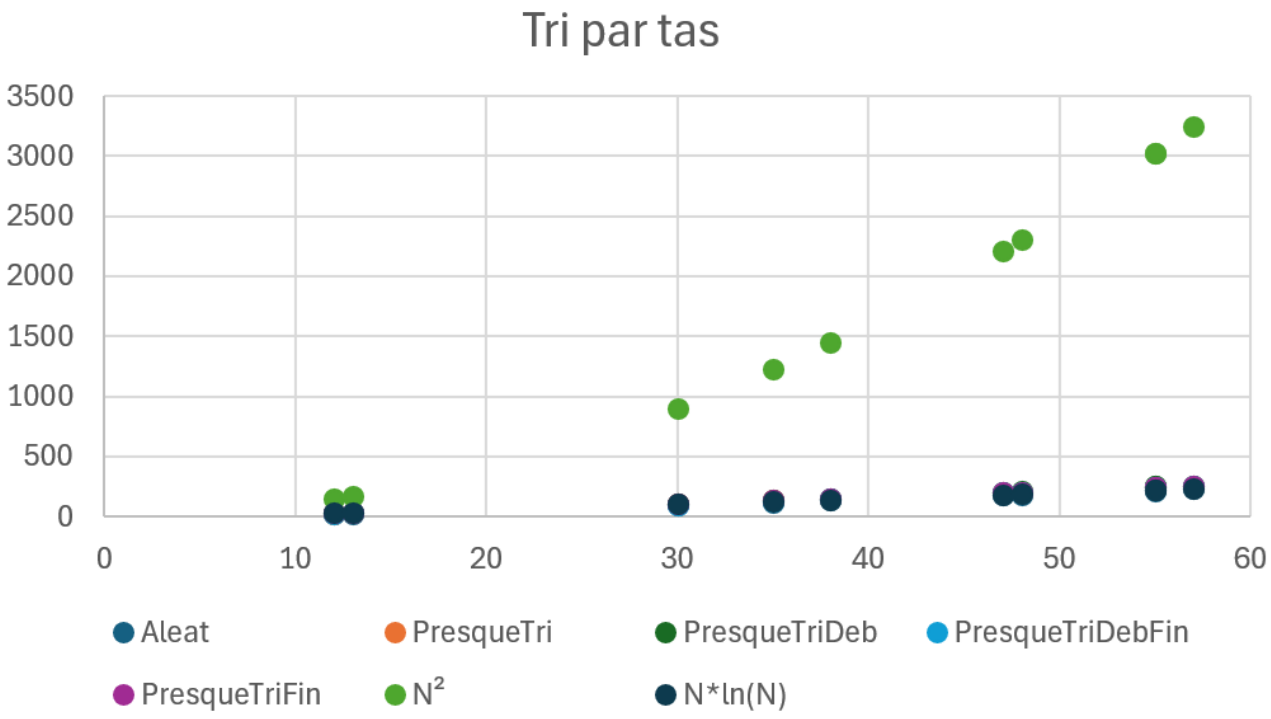


Huitième algorithme : le tri stupide

Très connu pour sa méthode inefficace, le tri stupide consiste à mélanger aléatoirement la totalité des valeurs tant qu'elles ne sont pas toutes ordonnées. Il n'est quasiment jamais utilisé en pratique pour des raisons évidentes d'inefficacité à moins, soyons clairs, d'un gigantesque coup de chance, surtout si la taille de la liste est importante. Sa complexité est de O((n + 1)!) pour n éléments.

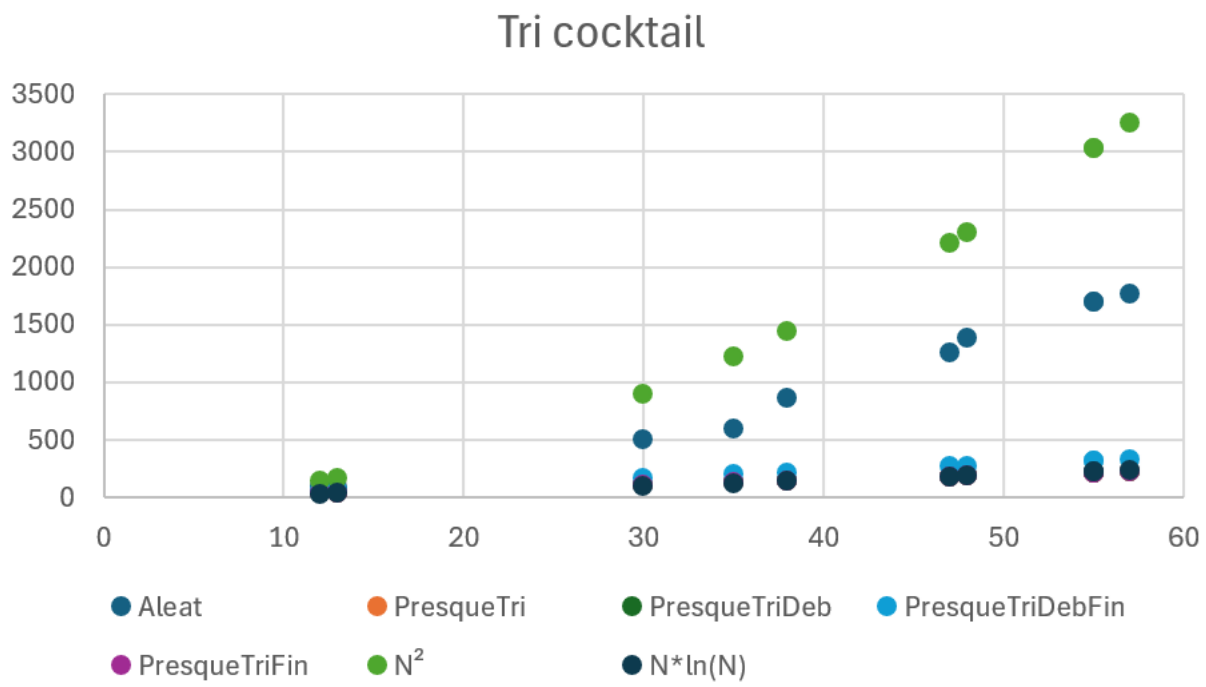
Neuvième algorithme : le tri par tas

Le tri par tas est un algorithme non stable de tri basé sur la structure de données. Chaque tas est un arbre binaire où chaque nœud aura une valeur inférieure ou égale à celle de ses enfants. On réalisera le tamisage des éléments, qui consiste à échanger chaque valeur avec la plus grande de ses fils de façon récursive jusqu'à ce que cette dernière soit à sa place.
Sa complexité est la suivante : O(n*log(n))



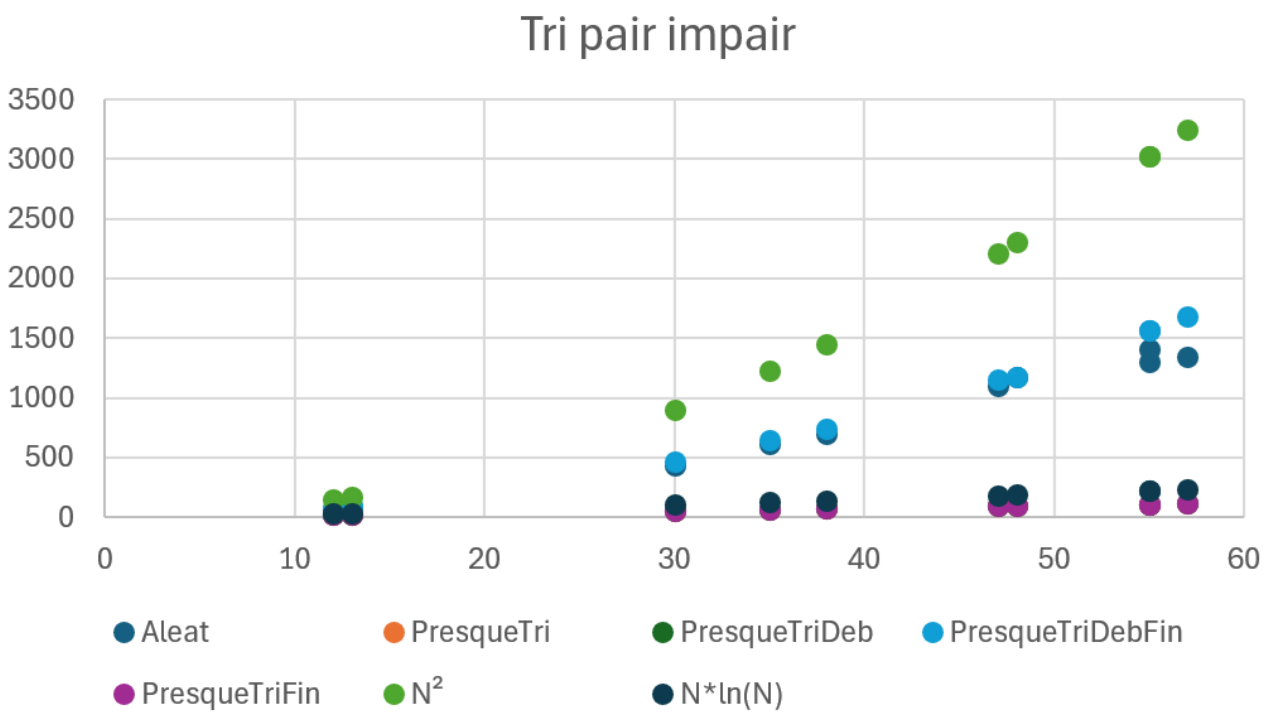
Dixième algorithme : le tri cocktail

C'est une variante stable du tri à bulles qui va avoir une méthode plus complète mais tout aussi simple à comprendre. On parcourra la liste de gauche à droite, en comparant les éléments voisins et les échangeant si nécessaire, pour retrouver le plus grand élément à la fin de la liste. Puis on fera la même méthode mais cette fois-ci de gauche à droite en déplaçant les éléments plus petits. On répétera ces deux étapes jusqu'à ce qu'aucune modification ne soit effectuée lors d'une itération totale. Sa complexité est de $O(n^2)$ avec n le nombre d'éléments.



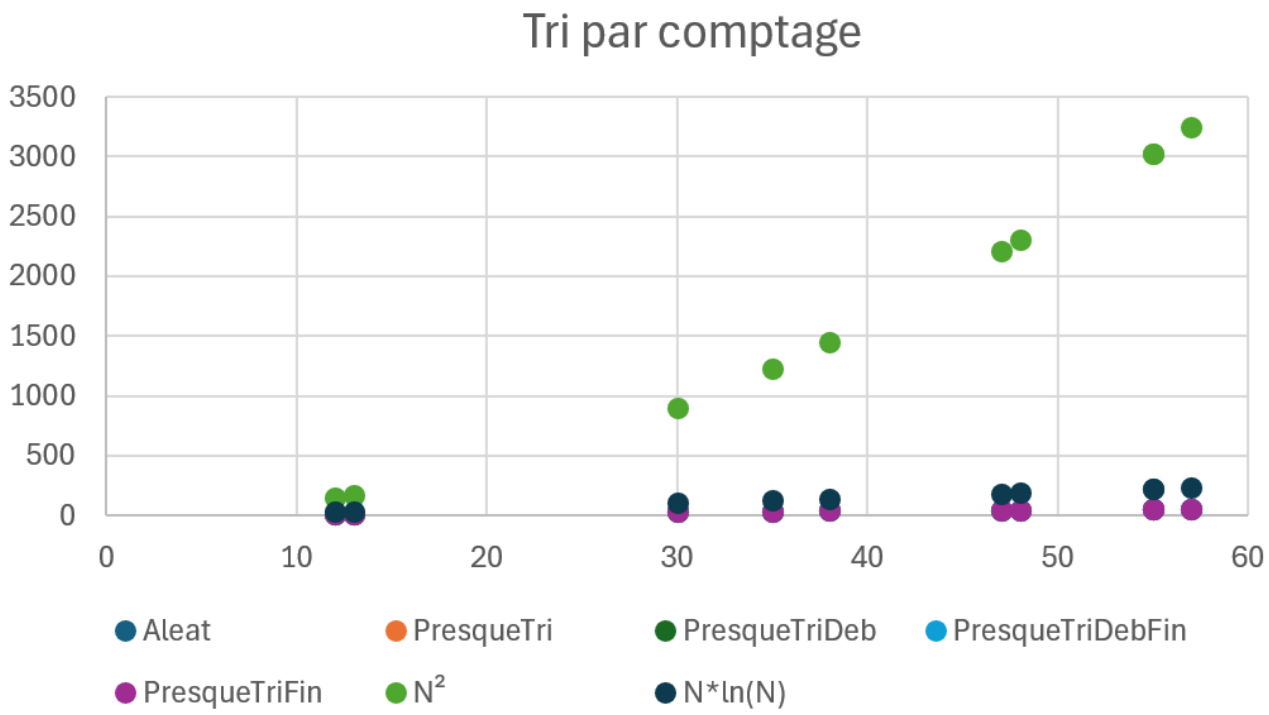
Onzième algorithme : le tri pair-impair

Cet algorithme non stable est une variante du tri à bulles basé sur l'idée de tri des éléments d'une liste en alternant entre les indices pairs et impairs. Il compare et échange les éléments voisins en alternant entre les indices de position pairs et impairs jusqu'à ce que la liste soit totalement triée. Sa complexité est de $O(n^2)$ avec n le nombre d'éléments. Cependant, ce n'est pas le plus efficace car d'autres méthodes comme le tri rapide et le tri fusion sont plus rapides.



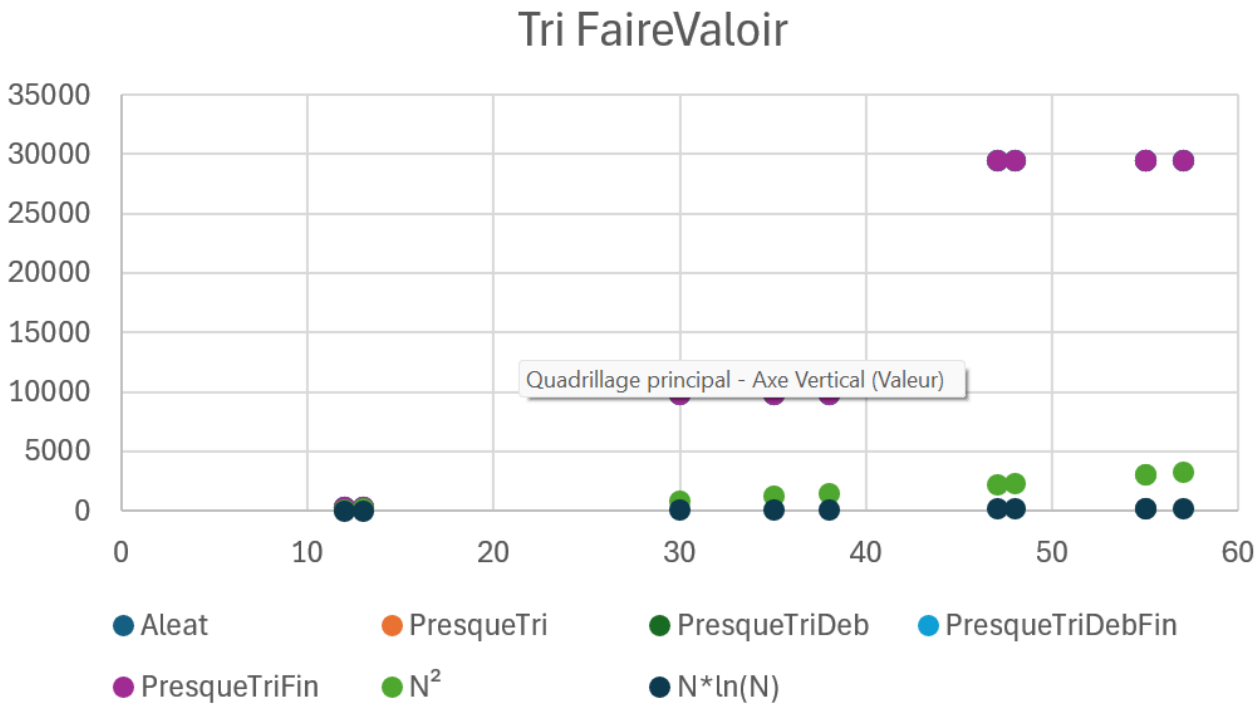
Douzième algorithme : le tri comptage

C’est un algorithme de tri qui fonctionne en donnant le nombre d’apparitions de chaque élément dans une liste puis en utilisant les quantités de valeurs pour reconstruire la liste.
Les trois étapes sont le comptage des apparitions, le calcul des positions finales, et la reconstruction de la liste finale.
Sa complexité dépend du nombre d’éléments n et de la plage des éléments k pour la formule O(n+k)



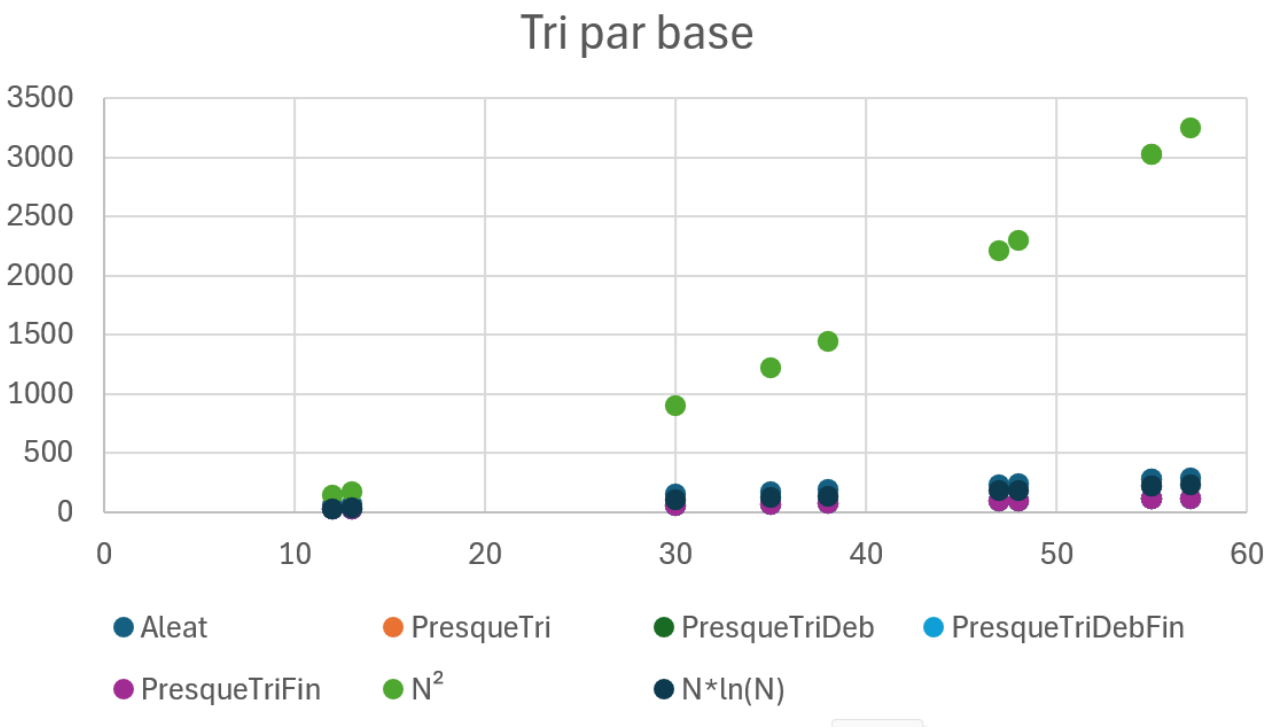
Treizième algorithme : le tri faire-valoir

Cet algorithme de tri par récursion se sépare en deux étapes : tout d'abord, on échange les valeurs externes de la liste (premier et dernier éléments) s'ils doivent l'être. Puis si la liste contient plus de trois éléments, on va trier d'abord les deux premiers tiers de la liste, puis on triera les deux derniers tiers et enfin les deux premiers à nouveau. Donc 1 2 puis 2 3 et à nouveau 1 2.
Cependant, il est très inefficace en comparaison avec le tri rapide ou le tri à bulles.



Quatorzième algorithme : le tri par base (ou tri radix)

Le principe de cet algorithme non stable est d'ordonner les éléments à l'aide de clefs uniques. On cherchera le chiffre le moins significatif de chaque clef, puis on triera la liste des éléments selon ce chiffre avec un algorithme de tri différent, et l'intérêt du tri stable est que l'ordre des éléments ayant le même chiffre sera conservé. On répétera ce tri en regardant le chiffre suivant de la clef à chaque fois.
Sa complexité dépend du nombre d'éléments n et de la plage des éléments k, donnant la formule O (n+k)



Choix de l'algorithme le plus efficace

L'algorithme le plus efficace parmi les 5 premiers est la méthode de tri rapide, avec sa complexité moyenne de O(n log n) et sa rapidité d'exécution.
Mais d'autres algorithmes sont très efficaces pour autant, comme le tri par tas, par insertion ou par sélection.

Sources

- Wikipédia
- cplusplusreference
- lwh.free.fr
- GitHub
- tutorialsgrey
- Cours des années précédentes