Partie 2 : Analyse des différents arbres binaires de recherche

Compréhension de l'arbre Splay

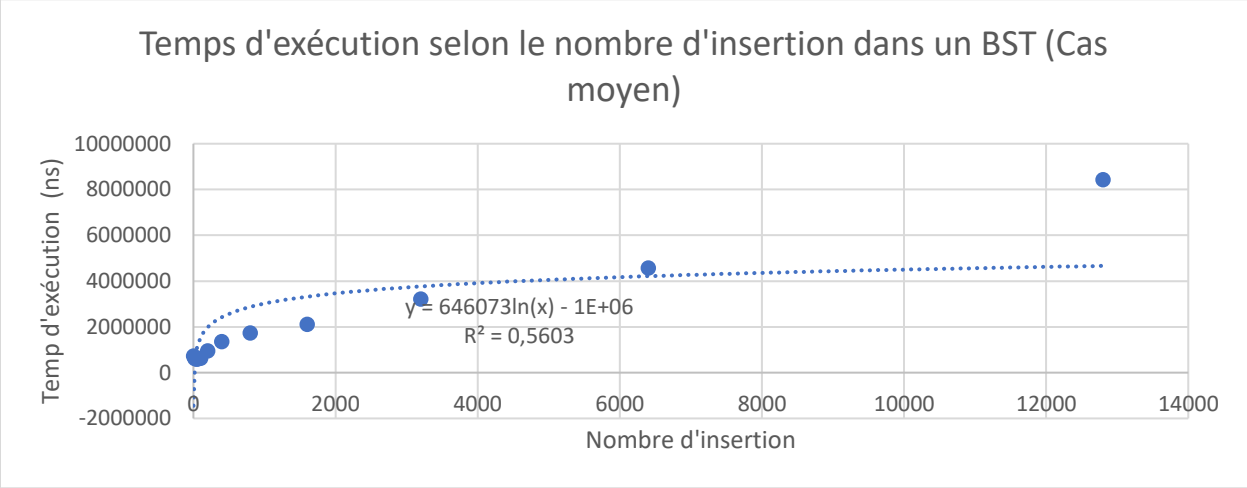
L'arbre Splay utilise une fonction appelée splay et qui prend un nœud en paramètre lors de l'insertion, de la recherche et de la suppression d'un nœud, selon la littérature, l'arbre est restructuré après un get d'un nœud de l'arbre [1]. Cette fonction permet de faire en sorte que le nœud devienne un parent et change de niveaux pour être du début de l'arbre comme l'indique le 'while (x.parent != null)'. On fait donc des échanges avec les parents pour permettre de garder un arbre équilibré. La littérature précise que cette manière de restructurer l'arbre en effectuant des rotations pour permettre au nœud de devenir la racine de l'arbre Splay et que cette méthode permet de réduire la hauteur des nœuds qui se trouvent sur le chemin entre la racine et le nœud à accéder [1]. La fonction splay est appelée à la suite de l'insertion d'un nœud, après avoir trouvé un nœud à l'aide de la fonction findNode et avant de faire un remove. Lors de la suppression, c'est le nœud de gauche qui devient la racine si le nœud à enlever possède un enfant de droite et un enfant de gauche. Selon la littérature, après la restructuration de l'arbre c'est le nœud possédant l'élément minimal de la branche de droite qui vient prendre la place de la racine [1].

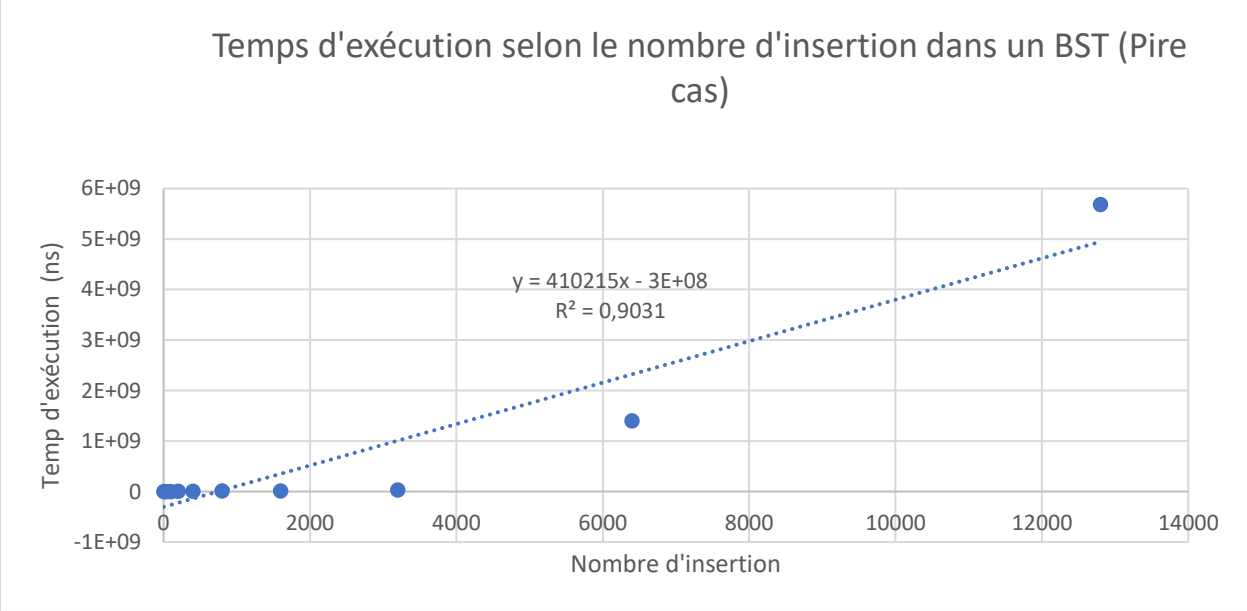
Évaluation de la performance

Pour les différentes opérations, les tailles de population utilisées sont toutes respectivement 1, 20, 50, 100, 200, 400, 800, 1600, 3200, 6400 et de 128000. Pour les cas moyens, les arbres sont remplis avec des éléments aléatoires et pour les pires cas, les éléments sont remplis de 0 jusqu'à la taille de l'arbre de façon croissante. Ainsi, un arbre contenant 5 nœuds sera rempli par l'élément 0 puis 1, puis 2 et 3 et finalement par 4. De telle façon de procéder ont été choisies car le cas moyen correspond au cas ou les données sont aléatoires et le pire cas consiste à avoir des éléments qui ne sont pas bien uniformisés et qui oblige les arbres binaires à prendre des structures moins équilibrées, un nouveau maximum étant toujours ajouté, les arbres prennent donc une structure plus linéaire.

Ainsi, la complexité pour l'insertion dans le cas moyen pour une BST est, selon la littérature de O(logn), et est de O(n) dans le pire cas.

Les résultats obtenus expérimentalement sont les suivants :

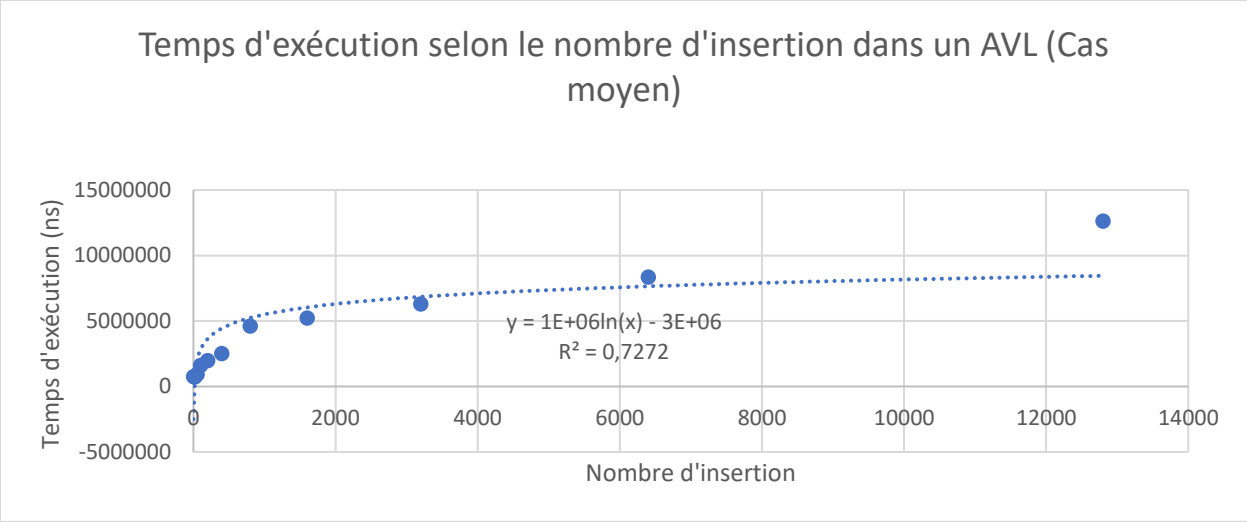


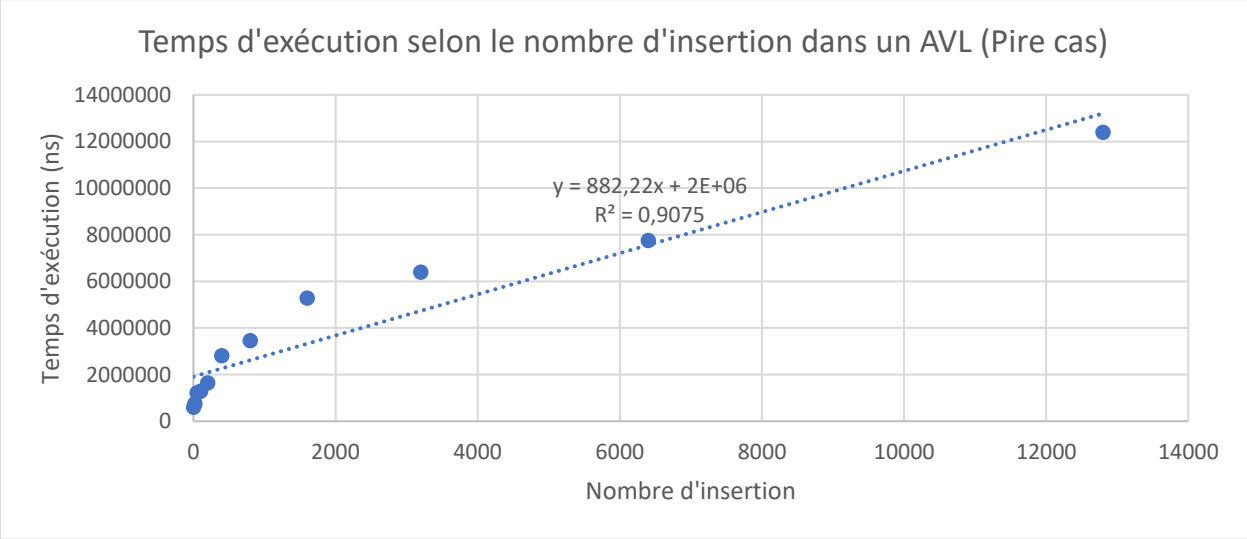


Les deux graphiques ne représentent pas totalement la réalité. Une hypothèse pour expliquer cette différence est que le nombre de données prises n'est pas assez élevé.

La complexité pour l'insertion dans une arbre AVL est de O(logn) pour le cas moyen et de O(n) dans le pire cas.

Les résultats obtenus par l'expérience sont les suivants :

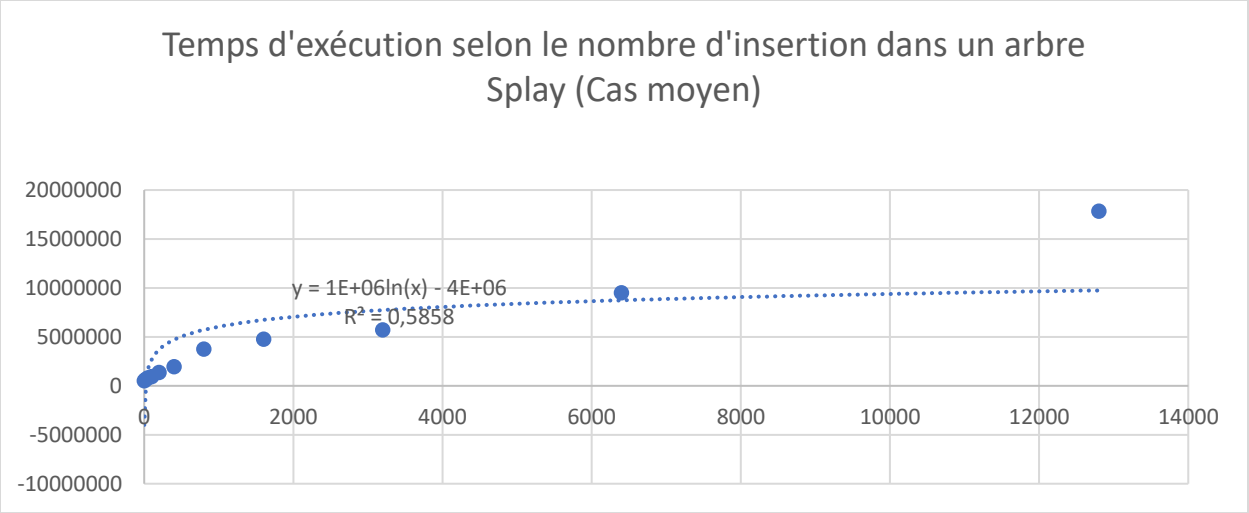


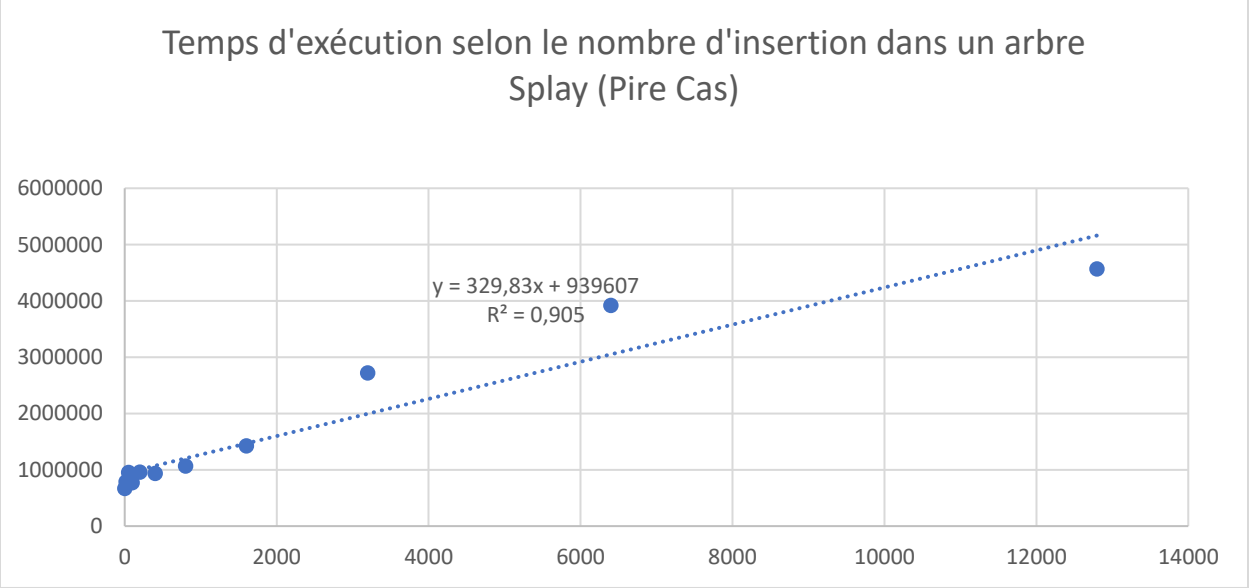


Encore une fois, les graphiques ne sont pas très représentatifs de la réalité. On peut toutefois noter que le temps d'exécution pour le pire cas est moins long que celui pour un cas similaire avec l'arbre BST. Ainsi, l'insertion semble plus rapide avec un BST car il n'y a nul besoin de rebalancer l'arbre après chaque insertion.

En ce qui concerne l'arbre Splay, la complexité pour un cas moyen est de O(logn) et de O(n) pour le pire cas.

Les résultats expérimentaux sont les suivants :





Les graphiques ne sont, encore une fois, pas représentatif de la réalité. Cependant, on remarque que les insertions sont plus rapides que l'arbre AVL, mais plus lent que le BST.

Pour les opérations de recherche, la complexité, dans un scénario moyen, pour un BST est de O(logn) et de O(n) dans le pire cas.

En ce qui concerne l'arbre AVL, la recherche dans le cas moyen et ans le pire cas a une complexité de O(logn).

Pour l’arbre Splay la complexité d’un cas moyen est de O(log n) et de O(n) dans le pire cas.

Finalement, pour les opérations de retrait. Le BST a une complexité de O(log n) dans le cas moyen et a une complexité de O(n) dans le pire cas.

L’arbre AVL a une complexité de O(log n) pour le retrait dans les 2 cas.

L’arbre Splay a une complexité de O(log n) pour le retrait dans un cas moyen et un complexité de O(n) dans le pire des cas.