École Polytechnique de Montréal
Département de génie informatique et génie logiciel

INF2010
Structures de données et algorithmes

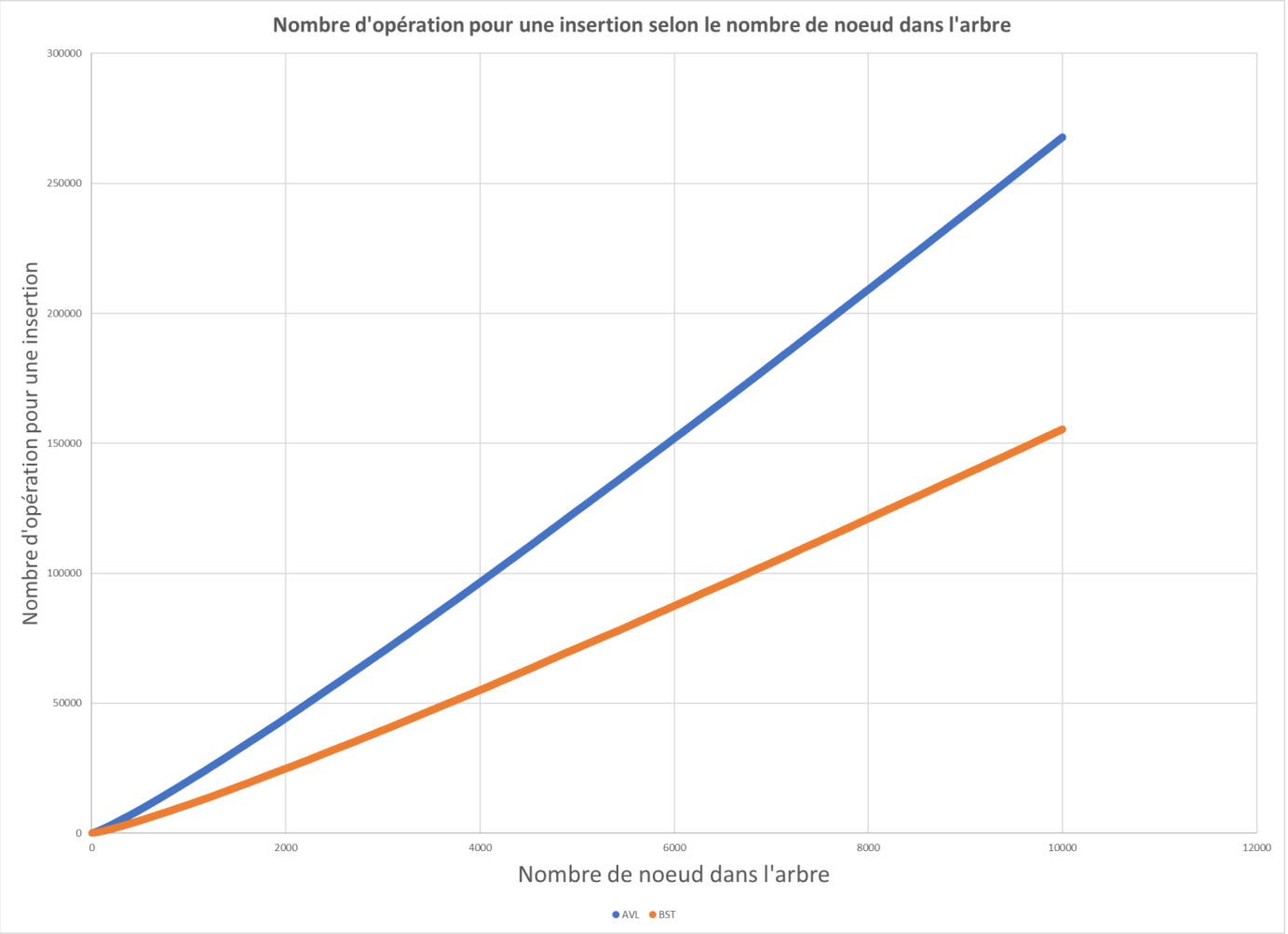
Soumis par
Elouan Guyon 2141829
Antoine Soldati 2147297

Section de Laboratoire #2

18 octobre 2022

Analyse des cas moyens

1. Insertion



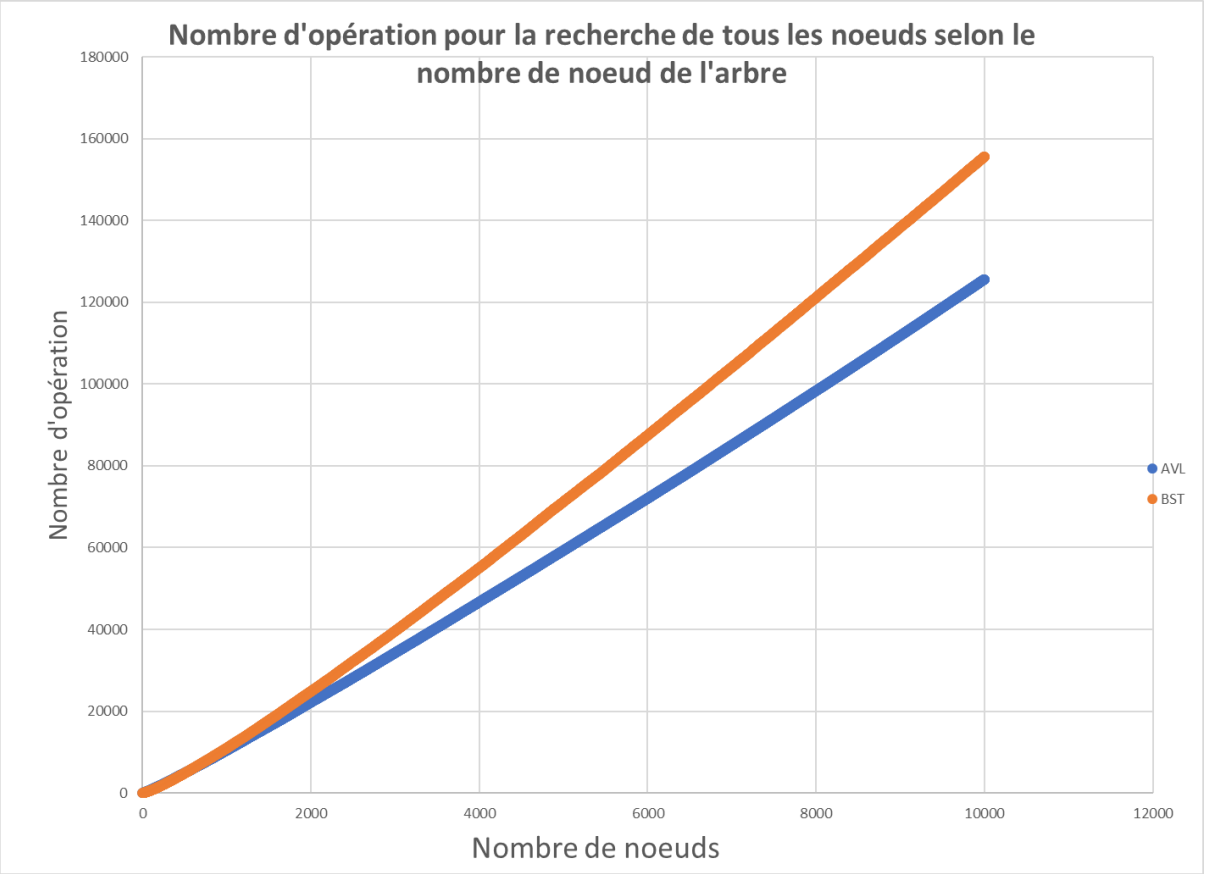
On peut voir sur le graphique que les courbes présentent une complexité O(nlogn). On voit au début que la courbe est légèrement courbée vers le haut. Vers la fin, la relation donne l'impression d'être linéaire, mais elle est bien O(nlogn).

On peut également voir que l'arbre BST est plus rapide. En effet, le nombre d'opérations pour un arbre de n noeuds est toujours plus grand avec un arbre AVL parce que les balancement et les rotations augmentent le nombre d'opérations de ces derniers.

La complexité de chaque courbe est conforme à la théorie. Pour trouver l'emplacement d'insertion, il faut chercher la position à insérer, cela se fait en O(logn). Pour insérer, c'est un temps constant. Vu qu'on insère n noeuds dans un arbre de taille n, il va de soit que la courbe soit O(nlogn). L'insertion pour AVL prend plus d'opération que pour BST puisque AVL doit balancer l'arbre régulièrement.

L’économie en opération devient évidente dans la graphique de recherche AVL versus BST.

2. Recherche

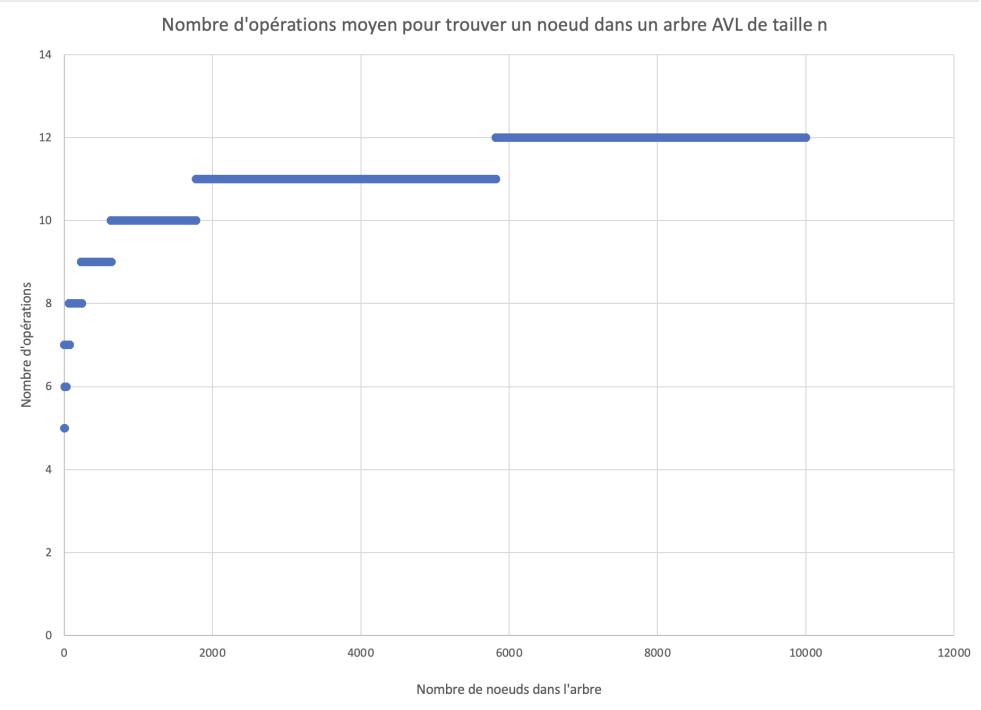


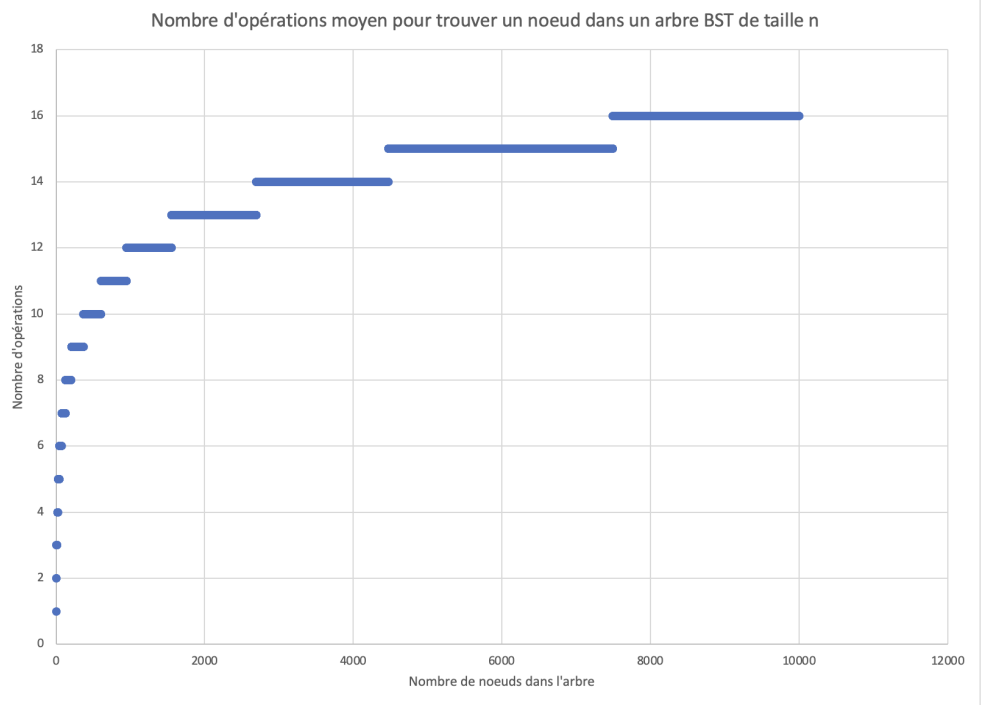
On peut voir sur le graphique que les courbes présentent une complexité O(nlogn) (difficile à voir à l'œil nu, mais se vérifie bien avec une règle).

On peut également voir que l’arbre AVL est plus rapide. En effet, on peut voir que la courbe AVL monte moins rapidement que la courbe BST.

La complexité de chaque courbe est conforme à la théorie. Pour trouver le noeud, il faut chercher la position de celui-ci, cela se fait en O(logn) (voir plus bas). Vu qu’on cherche n noeuds, il va de soit que la courbe soit O(nlogn). On voit également que le balancement d’un arbre permet de chercher un noeud plus rapidement (effet plus prononcé plus on a un grand arbre).

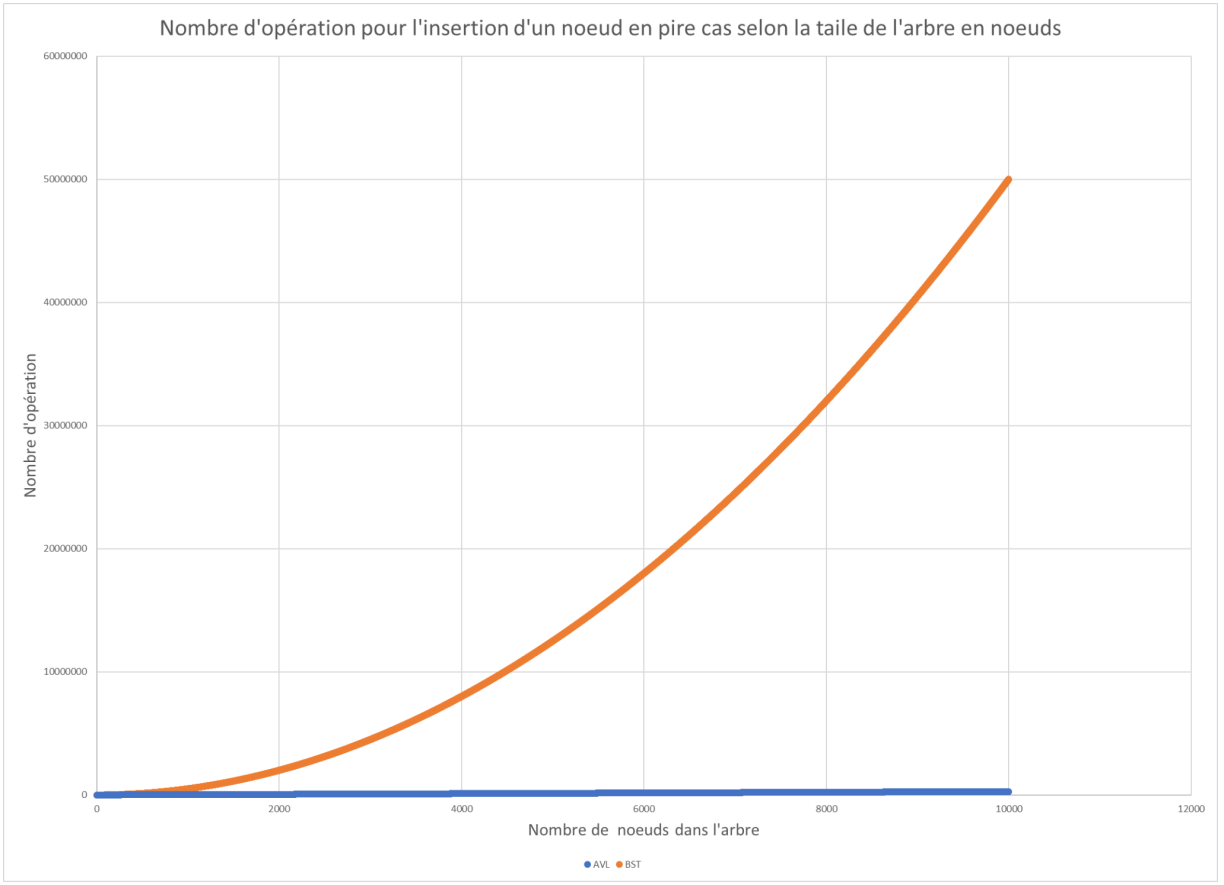
Ci-dessous, nous avons le nombre moyen d'opérations pour chercher un nombre quelconque dans un arbre BST ou AVL de n noeuds:



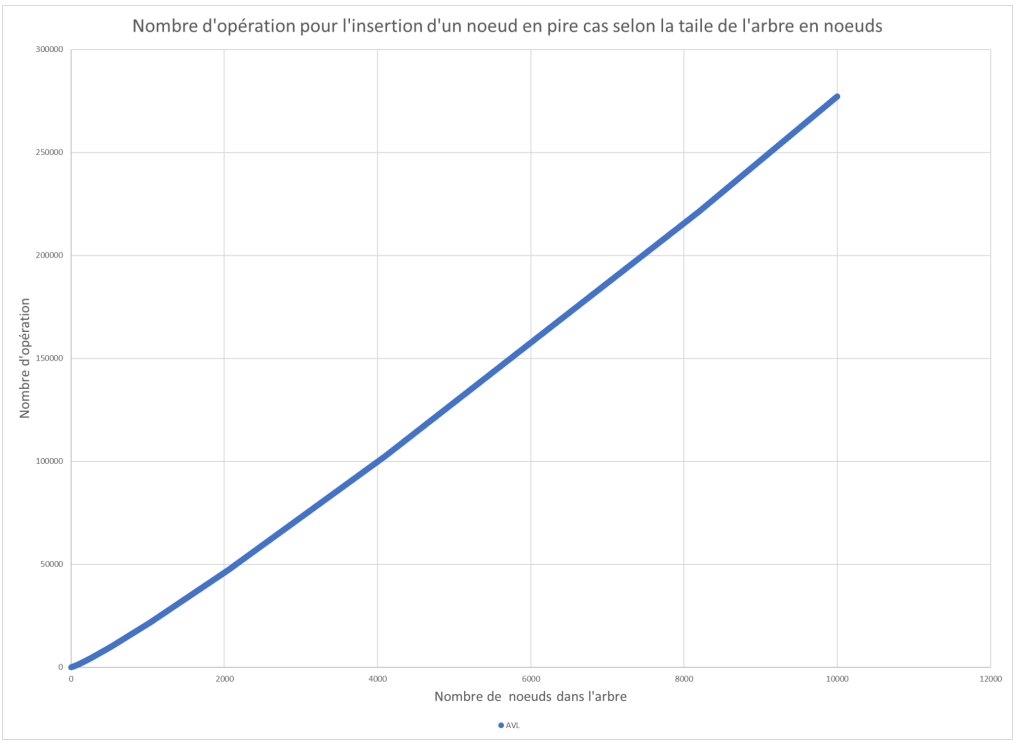


Dans les deux graphiques, on peut très bien voir que la complexité de la recherche d'un élément est O(logn), ce qui est conforme à la théorie.

3. Pire cas pour l'insertion



On peut voir sur le graphique que la courbe de BST présente une complexité $O(n^2)$. La courbe AVL présente aussi une complexité O(nlogn), mais il faut faire une graphique juste pour AVL pour voir la complexité car le nombre d'opérations pour insérer dans un arbre BST augmente bien plus rapidement que pour un arbre AVL.



On peut également voir que l'arbre AVL est bien plus rapide. Cela est logique, parce que même si on est dans le pire cas, l'arbre AVL va se balancer ce qui va grandement réduire le temps moyen d'une insertion.

La complexité de chaque courbe est conforme à la théorie.

Pour l'arbre BST de pire cas, chaque nœud n'a qu'un seul parent et un seul enfant (hormis le premier et le dernier nœud). On va donc trouver l'emplacement d'insertion d'un nœud en O(n). Insérer (se fait en O(1)) n nœuds en dans un arbre BST de pire cas va donc se faire en $O(n^2)$, ce qui est conforme à la théorie.

Trouver l'emplacement d'insertion dans l'arbre AVL de pire cas se fait en O(logn) puisque l'arbre est balancé. Pour insérer un nœud, c'est un temps constant. Vu qu'on insère n noeuds, il va de soit que la courbe soit O(nlogn).