Rapport de laboratoire TP3

INF2010

STRUCTURES DE DONNÉES ET ALGORITHMES

27 octobre 2022

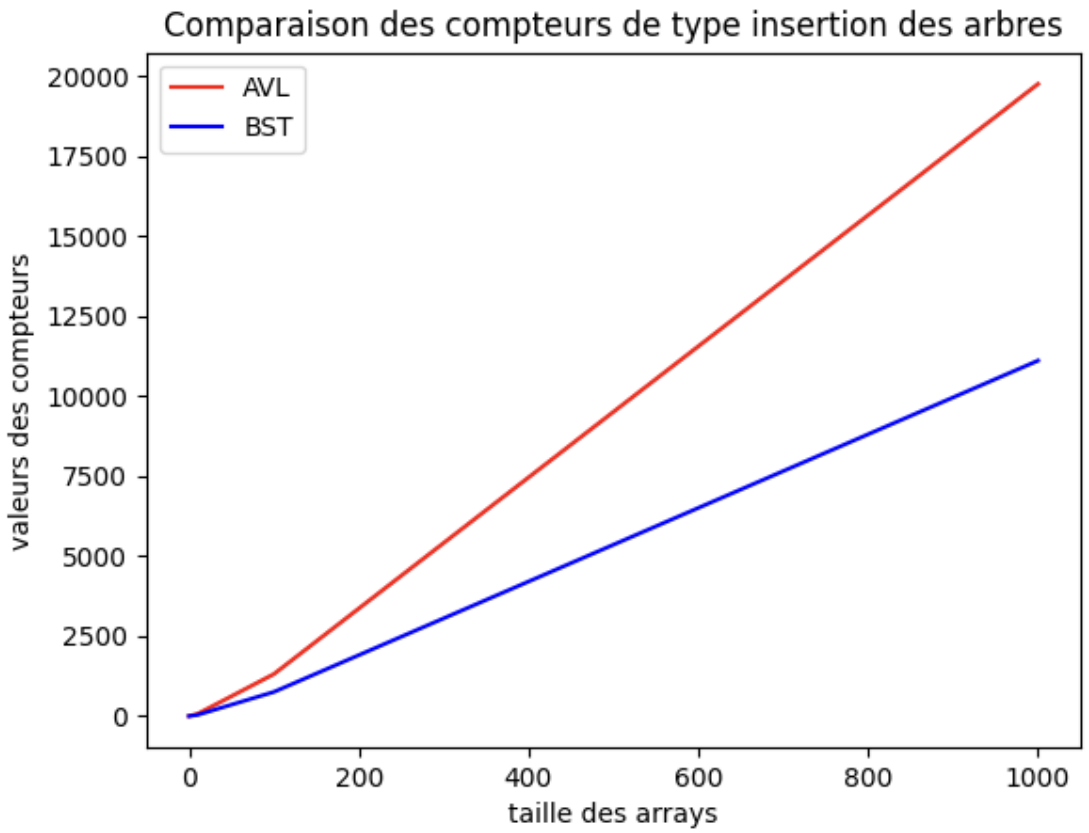
Skander Hannachi 2088988
Rayan Fellah 2147523

Partie 2: Analyse en cas moyen

Tableau 1 : Compteur du nombre d'opérations d'insertion pour les arbres BST et AVL selon la taille du tableau entré

Taille du tableau entré	BST	AVL
10	36	66
100	755	1314
1000	11117	19770

Graphique de comparaison des compteurs de type insertion des arbres:



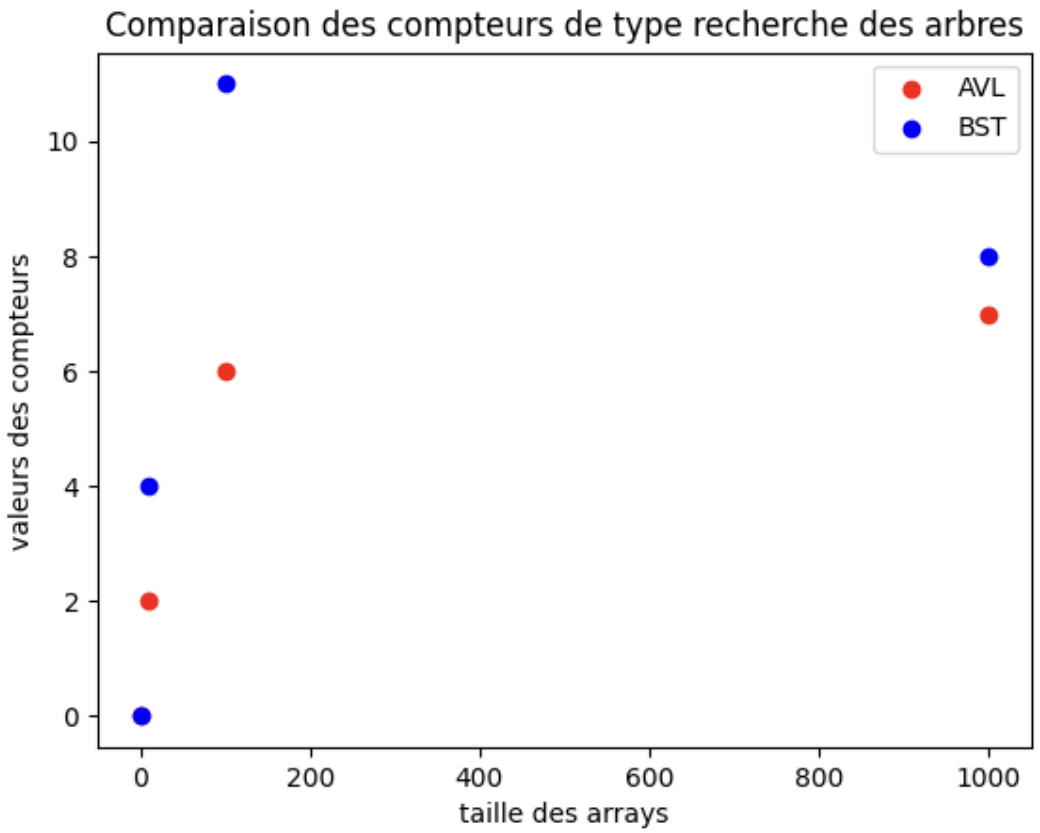
Analyse des résultats:

Pour l'insertion des données, il est possible de constater que logiquement, plus la taille du tableau augmente, plus le nombre d'opérations d'insertion augmente pour les deux types d'arbre. On peut voir que l'arbre AVL effectue plus d'opérations en général pour l'insertion , mais que la tendance asymptotique reste la même que l'arbre binaire de recherche. En effet, on observe que les deux types d'arbres semblent se diriger vers une complexité asymptotique de O(n log(n)). Cela va parfaitement en accord avec la théorie puisque l'opération d'insertion se fait en ca moyen en O(log(n)) pour un arbre binaire de recherche et pour un arbre AVL. Si l'on veut insérer un tableau de taille quelconque (prenons taille n), la complexité totale devient alors n * (O(log n)) donc **O(n log(n))**.

Tableau 2 : Compteur du nombre d'opérations de recherche pour les arbres BST et AVL selon la taille du tableau entré

Taille du tableau entré	BST	AVL
10	4	2
100	11	6
1000	8	7

Graphique de comparaison des compteurs de type recherche des arbres:



Analyse des résultats:

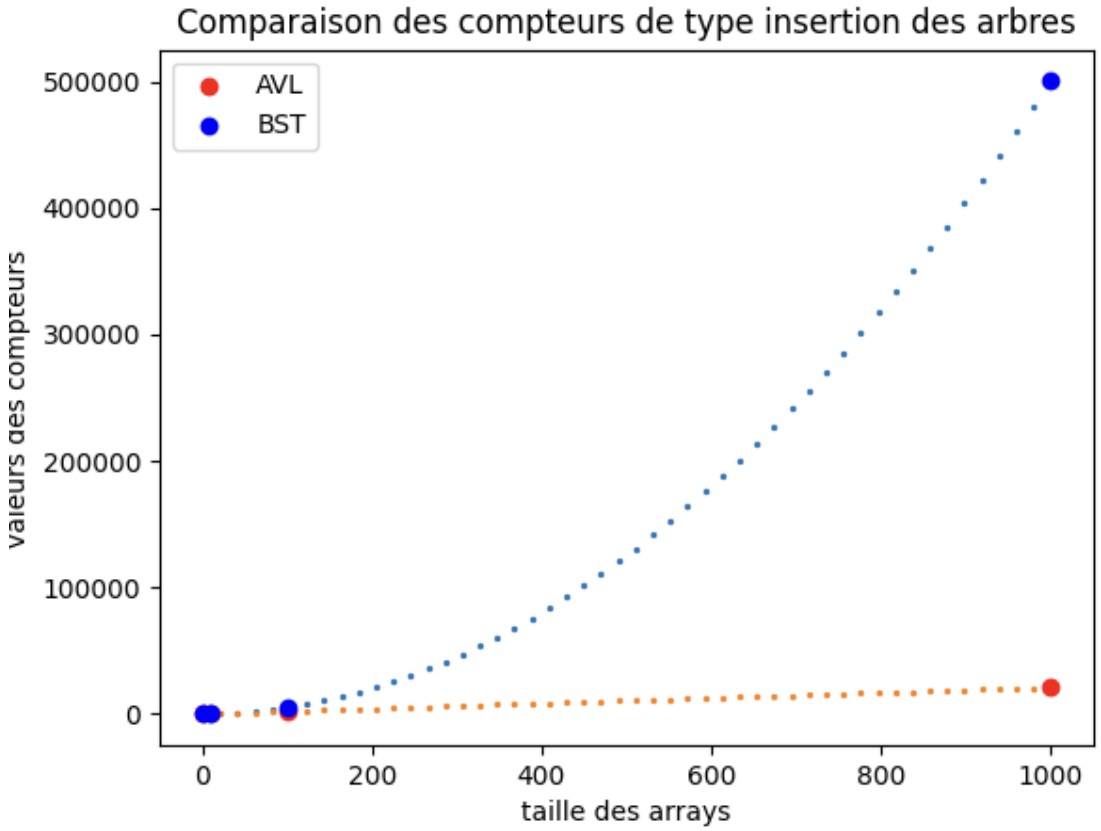
Comme vu dans la théorie, les arbres nous montrent une tendance logarithmique pour la recherche. En effet, ils ont un comportement asymptotique qui tend vers le **O(log(n))**. Cependant, puisqu'un arbre AVL est balancé lors des opérations d'insertion, la recherche se fera plus rapidement, mais cela n'optimise pas nécessairement la complexité algorithmique, comme nous le démontre le graphique ci-haut.

Partie 3: Analyse en pire cas

Tableau 3: Compteur du nombre d'opérations d'insertion pour les arbres BST et AVL selon la taille du tableau entré

Taille du tableau entré	BST	AVL
10	55	76
100	5050	1439
1000	500500	20944

Graphique de comparaison des compteurs de type insertion des arbres:



Analyse des résultats:

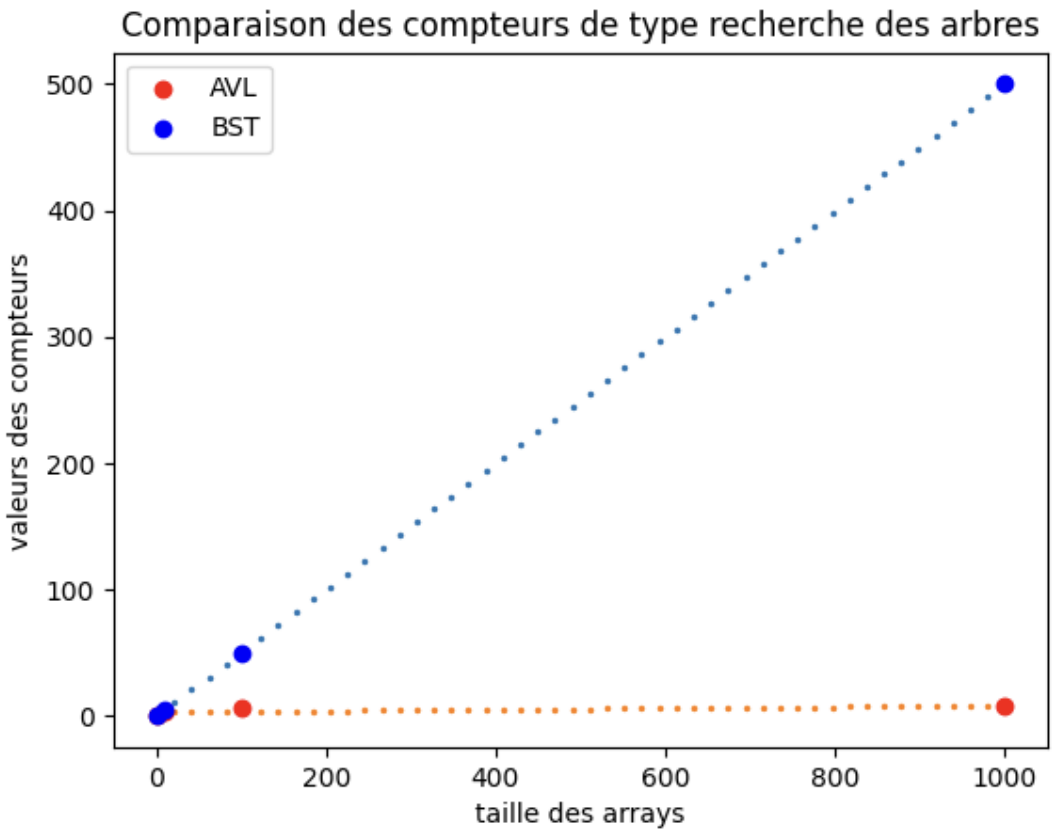
Dans l'analyse du pire cas, on observe que l'arbre de recherche standard semble se diriger vers une complexité asymptotique de O(n^2). Cela va parfaitement en accord avec la théorie puisque l'opération d'insertion se fait en O(n) pour le pire cas d'un arbre binaire de recherche. En effet, si l'on veut insérer un tableau de taille quelconque (prenons taille n), la complexité totale devient alors n * (O(n)) donc

O(n^2). On voit sur le graphique que la recette AVL démontre toute son utilité, puisque le balancement lors des insertions, on évite le pire cas qui serait une suite continuelle d'une des deux branches principales alors que l'autre est vide. En effet, la recette AVL assure une différence de niveau entre les branches de maximum 1. Cela résulte en la conservation de la complexité O(log(n)) pour une opération d'insertion et donc **O(nlog(n))** pour l'insertion d'un tableau de taille n.

***Tableau 4 :** Compteur du nombre d'opérations de recherche pour les arbres BST et AVL selon la taille du tableau entré

Taille du tableau entré	BST	AVL
10	5	4
100	50	6
1000	500	8

Graphique de comparaison des compteurs de type recherche des arbres:



Analyse des résultats:

Encore ici, on voit que la recherche en pire cas pour un arbre binaire de recherche (BST) est exactement de **O(n)**. En effet, pour une disposition où tous les éléments sont alignés à droite ou bien à gauche, on devra traverser l'ensemble des éléments contenu dans l'arbre pour trouver l'élément désiré. Pour ce qui est de l'AVL, on voit ici la conservation de la complexité du cas moyen de recherche malgré l'insertion

effectuée en pire cas (où tous les éléments se suivraient ex : 1,2,3,4,5…). En effet, encore, ici le graphique démontre une complexité asymptotique de **O(log(n))**.