Julien Lefevre
Téo Itsweire-Krebs
Antoine Promis

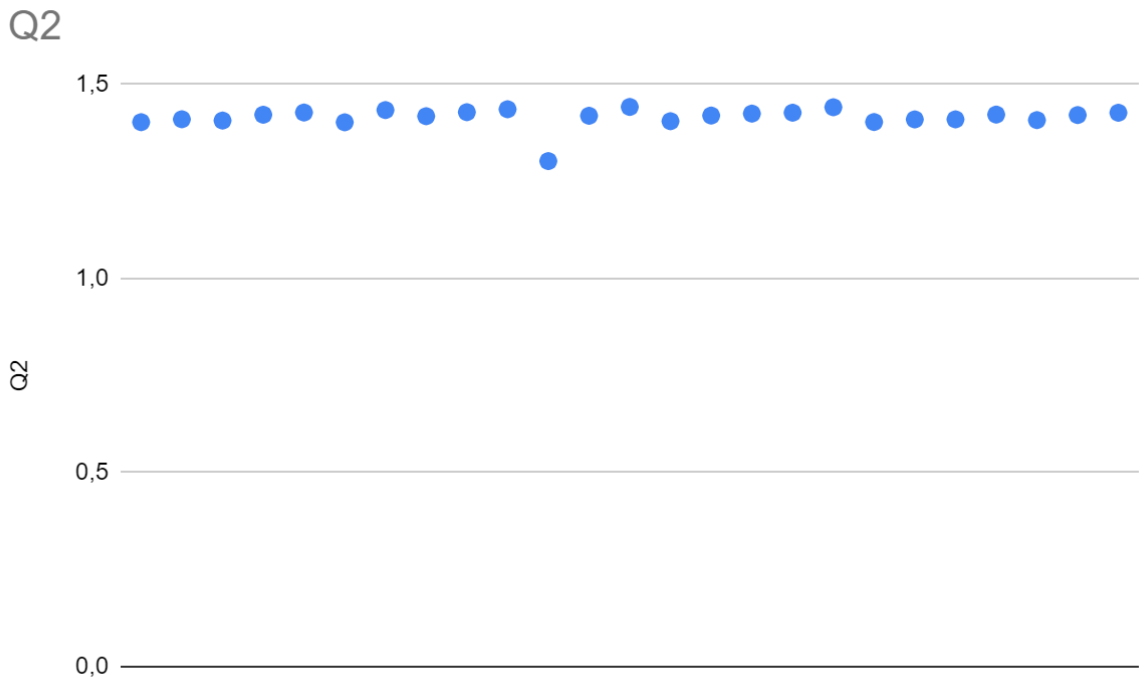
L3 Informatique

# Compte-rendu Projet Algorithmes et Programmation 3

## 1. Arbres binaires de recherche

### Question 2 :

Dans cette question, il nous était demandé de réaliser l'expérience suivante : calculer la moyenne de déséquilibre d'un arbre binaire de recherche. Pour cela, nous utilisons la fonction *bst_rnd_create()* que nous avons programmé dans la question 1. Ensuite, nous mesurons le déséquilibre en chaque nœud de l'arbre, ce qui nous permet d'obtenir la moyenne de déséquilibre de l'arbre binaire de recherche. Nous avons appliqué cette expérimentation sur 10 000 arbres binaires de recherche de taille 100. Voici sous forme de graphique les résultats que nous avons obtenu :

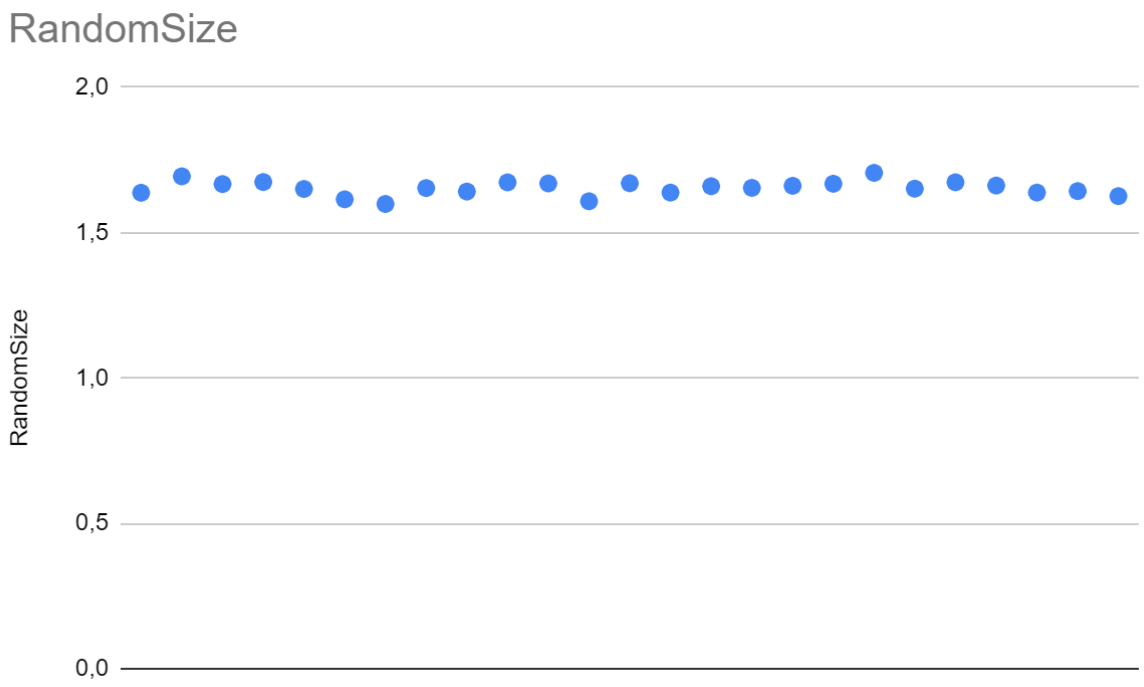



*"Graphe représentant 25 moyennes de déséquilibre de chacunes 400 arbres binaire de recherche"*

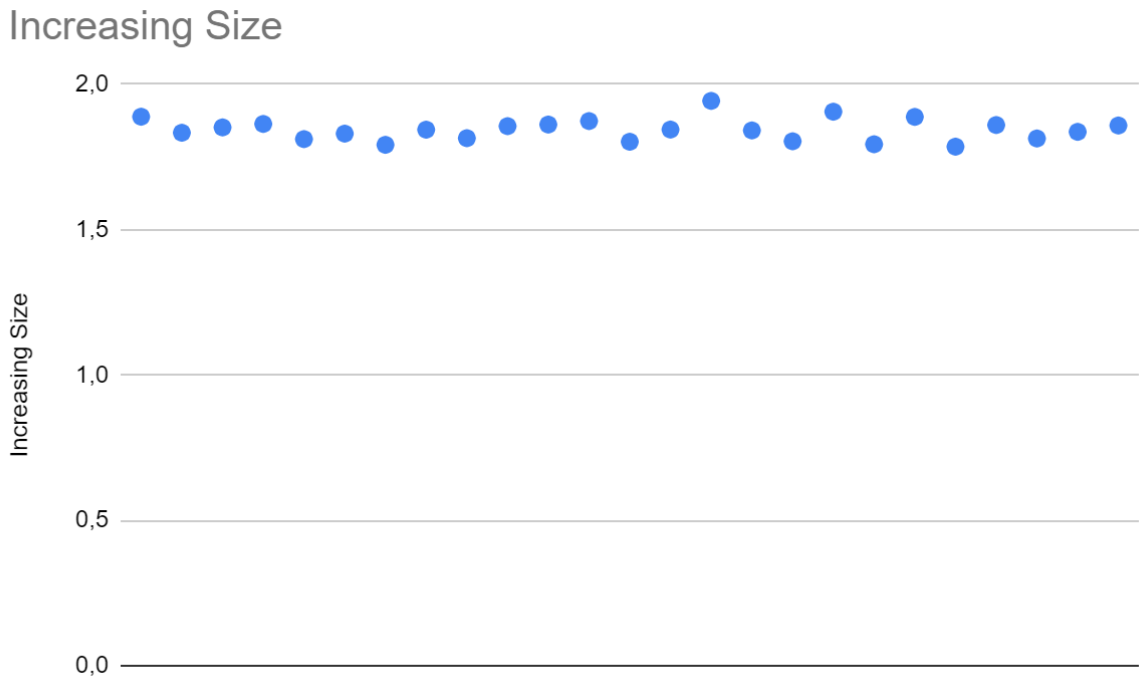
### Question 3 :

Ici, l'expérience que nous devions réaliser est la suivante : nous devions estimer le déséquilibre moyen d'un arbre binaire de recherche construit à l'aide d'une suite de nombres

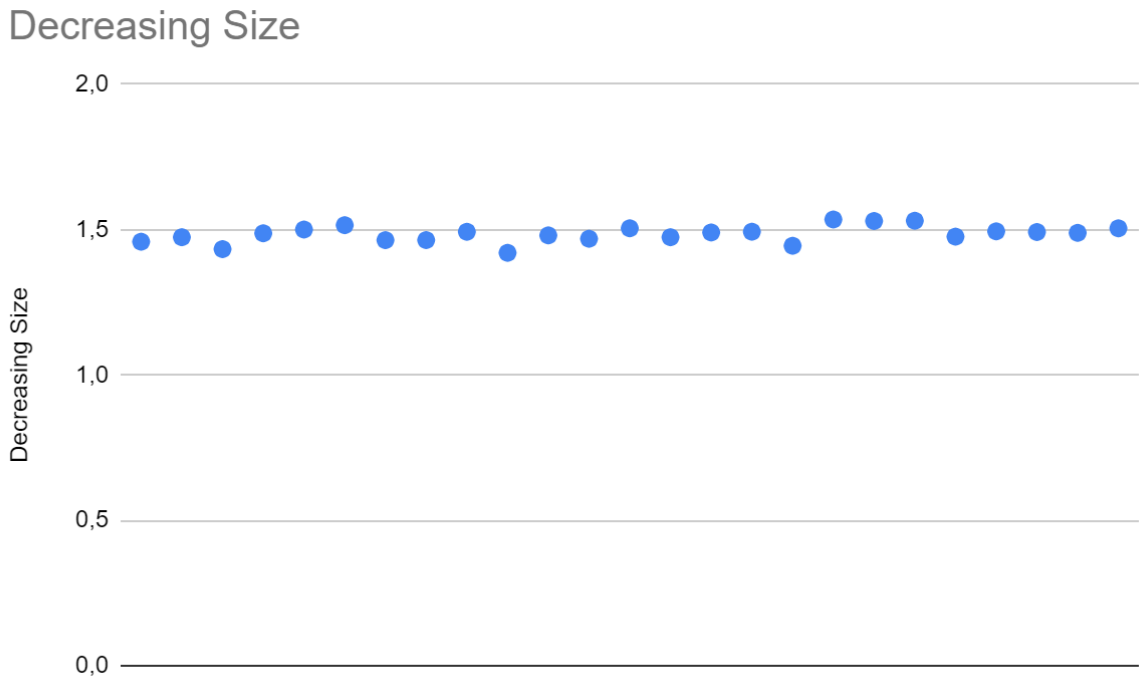
entiers qui contient des sous-suites ordonnées de longueurs variables. Nous avons mené cette expérience avec des sous-suites de longueurs croissantes, décroissantes, aléatoires et fixes, voici sous forme de graphiques les résultats que nous avons obtenu :



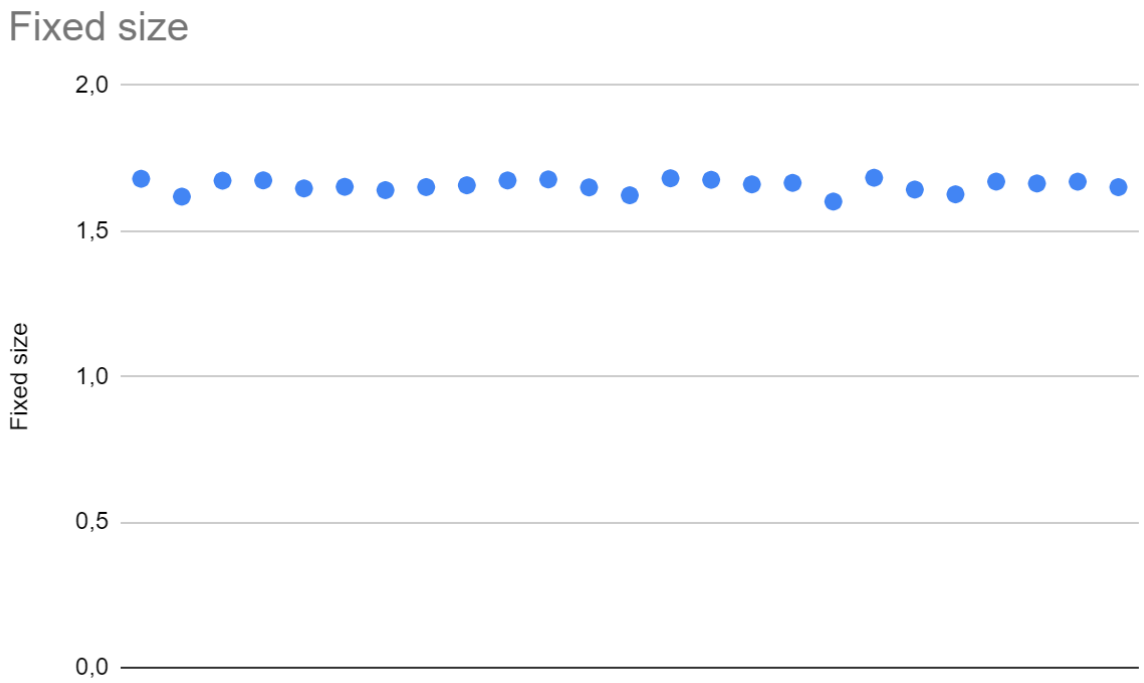

*“Graphe représentant 25 moyennes de déséquilibre de chacunes 100 arbres binaire de recherche (sous suites de tailles aléatoires)”*




*“Graphe représentant 25 moyennes de déséquilibre de chacunes 100 arbres binaire de recherche (sous suites de tailles croissantes)”*




*“Graphe représentant 25 moyennes de déséquilibre de chacunes 100 arbres binaire de recherche (sous suites de tailles décroissantes)”*



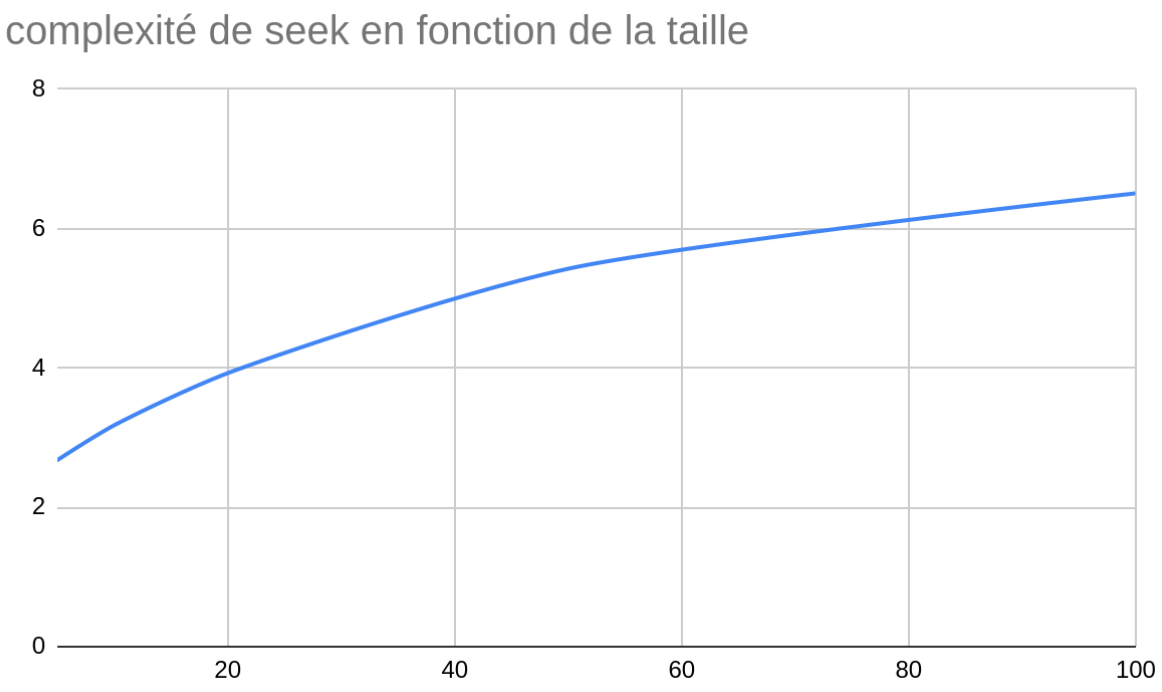

*“Graphe représentant 25 moyennes de déséquilibre de chacunes 100 arbres binaire de recherche (sous suites de tailles fixés à 10)”*

## 2. Arbres AVL
### 2.2. Expérimentations avec les arbres AVL

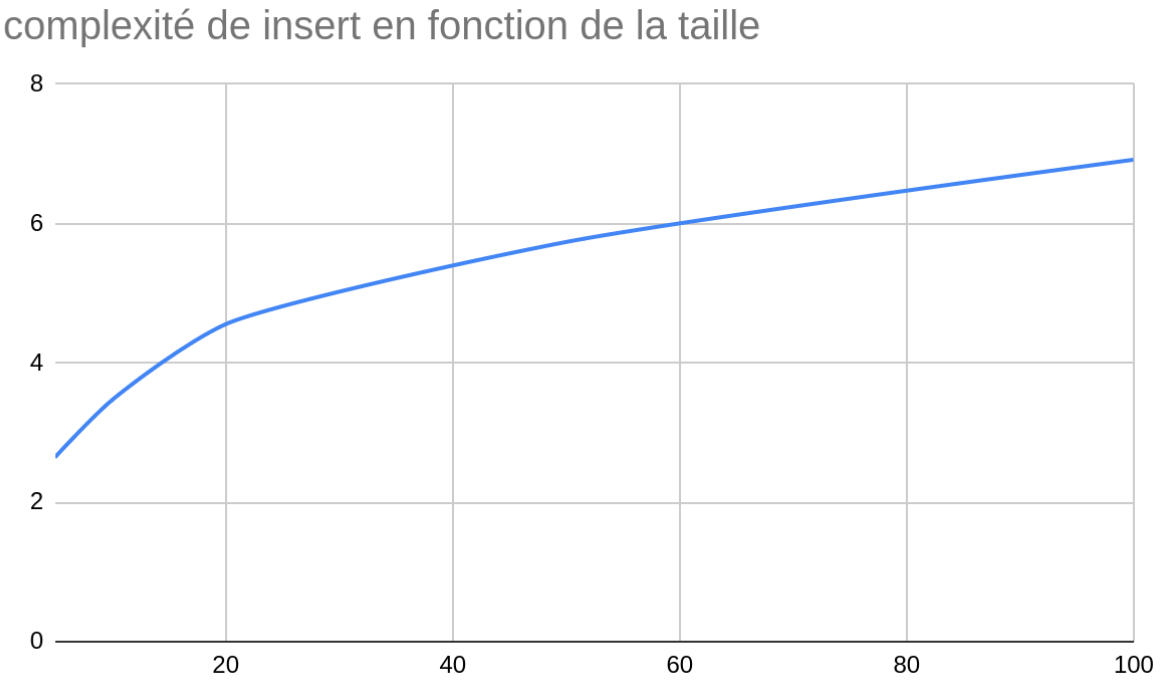
Nous avons remarqué au cours de ces expérimentations, que une de nos fonctions à un problème. En effet, lors de la création d'arbre de grande taille, une erreur apparait dans certains cas et le programme s'arrête (l'arbre AVL n'est donc pas construit). Après de longues recherches, nous avons pu identifier la source du problème : notre fonction rebalance_aux() (qui se trouve dans le fichier experimentations_AVL.ml). Malheureusement, nous n'avons pas réussi à identifier plus précisément ce problème et n'avons donc pas été en mesure de le résoudre. Cependant, nous pouvons affirmer que lorsque nous n'entrons pas dans le cas qui arrête le programme, et que donc notre arbre AVL est construit, ce dernier est bien un Arbre AVL. Ainsi, nous avons pu tant bien que mal effectuer les expérimentations demandées.

#### Question 1 :

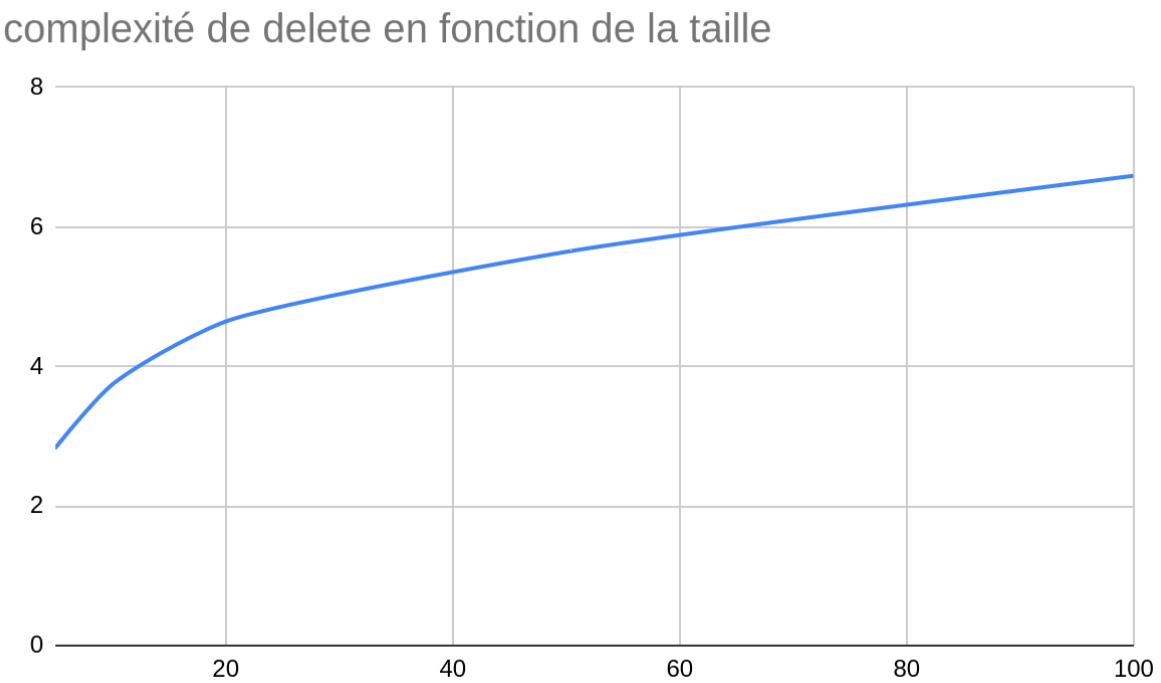
Dans cette question, il nous a été demandé de vérifier expérimentalement que nos fonction de recherche, d'insertion et de suppression ont bien une complexité en Θ(log n) avec n la taille de l'arbre. Pour chacune de ces fonctions, nous avons donc calculé le nombre d'appel récursif nécessaire pour chaque fonction, voici les résultats que nous avons obtenu :




*"Graphe représentant la complexité de la fonction de recherche d'un élément en fonction de la taille de l'arbre AVL"*




*“Graphe représentant la complexité de la fonction d’insertion d’un élément en fonction de la taille de l’arbre AVL”*




*“Graphe représentant la complexité de la fonction de la suppression d’un élément en fonction de la taille de l’arbre AVL”*

Question 2 :
Par manque de temps, nous n’avons pu répondre à cette question, mais la création d’un arbre AVL à l’aide des suites de sous-suites fonctionne.