Université du Québec en Outaouais **15 novembre 2020**

Département d'informatique et d'ingénierie

INF4063/ Devoir2

C'est un travail de groupe de **2 étudiants** (binôme), et il compte **pour 7 points** avec un barème fourni à titre indicatif sur 8 points (Vous pouvez avoir une note de 8 sur 7 si votre solution est complète).

À soumettre en ligne sur Moodle au plus tard le 13 décembre à 23 :55 sous forme d'un fichier zip. Le nom du fichier zip doit être une concaténation des noms de famille et prénoms des membres du binôme séparés par des « - » et suivi de « -Devoir ». Exemple de nom de fichier «CoutuJean-CruiseTom-Devoir2.zip ». Les dossiers extraits du fichier zip auront des noms qui commencent par «noms-prénoms » des étudiants.

Exercice 1 (4 pts)

Un arbre binaire A est défini récursivement comme suit :

(i) Soit A est l'arbre vide, qu'on dénote par φ

ou

(ii) A est composé d'un nœud, d'un sous-arbre gauche et d'un sous-arbre droit.

Un nœud d'un arbre est appelé feuille si ses sous-arbres gauche et droit sont vides (voir Figure 1). On dit d'un arbre qu'il est équilibré si, pour chacun de ses noeuds, le sous-arbre gauche et le sous-arbre droit ont des hauteurs qui diffèrent d'au plus 1 (voir Figure 1). Dans les deux questions qui suivent, si A n'est pas un arbre vide, alors A.gauche( ) retourne son sous-arbre gauche et A.droit( ) retourne son sous-arbre droit. De plus, vous pouvez supposer que l'expression hauteur (A) retourne la hauteur de l'arbre binaire A.

(a) (2 pts) Écrivez un algorithme récursif qui calcule le nombre de feuilles présentes dans un arbre binaire (Une description de votre algorithme est demandée et votre algorithme doit être écrit sous-forme d'une fonction pour laquelle vous précisez les paramètres d'entrée, leurs types et le type de la valeur de retour).

(b) (2 pts) Écrivez un algorithme récursif qui vérifie si un arbre binaire est équilibré (Une description de votre algorithme est demandée et votre algorithme doit être écrit sous-forme d'une fonction pour laquelle vous précisez les paramètres d'entrée, leurs types et le type de la valeur de retour).

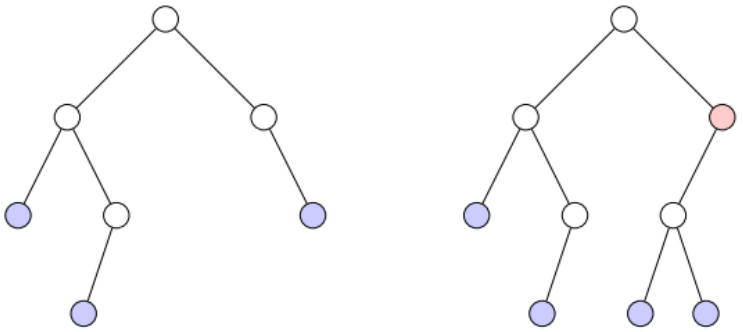

FIGURE 1 – Deux arbres binaires. Les feuilles sont colorées en bleu. L'arbre de gauche est équilibré alors que l'arbre de droite ne l'est pas : il y a un déséquilibre au noeud en rouge, puisque le sous-arbre gauche est de hauteur 2 et le sous-arbre droit est de hauteur 0 (différence de $2 > 1$.

Exercice 2 (4 pts) :

On considère une liste L_N d'entiers allant de zéro à N-1. On considère pour l'exécution du programme demandé dans cette exercice d'adopter une valeur de N= 1000 .

On vous demande de créer un programme en Java dont l'exécution fait :

1. Inviter l'utilisateur à choisir le nombre d'éléments N mentionné ci-haut. (Choix par défaut 1000)
2. stocker les éléments de L_N (sans besoin de créer L_N) dans une instance qu'on nomme LinkL de la classe LinkedList
3. Supprimer les éléments de LinkL un à un en exécutant une boucle avec l'appel de la méthode removeFirst() de LinkL à chaque itération de la boucle.
4. Le programme doit afficher le temps en Nanosecondes qu'a pris l'exécution de la boucle du point numéro 3 mentionnée ci-haut. L'affichage doit être similaire à : « Suppression des éléments de LinkL avec la méthode removeFirst( ) de LinkedList, avec une durée duréeLinkedList = ….. nanosecondes »
5. Stocker les éléments de L_N dans une instance nommée ArrL de la classe ArrayList.
6. Supprimer les éléments de ArrL un à un en exécutant une boucle avec un appel à chaque itération à la méthode remove(0) de ArrL. Le programme doit afficher le temps en nanosecondes qu'a pris l'exécution de la boucle. L'affichage de la durée doit être similaire à : « Suppression des éléments de ArrL avec remove(0) d'un ArrayList, durée duréeArrayDirect = ….. nanosecondes »

7. Recréer de nouveau une instance ArrL de ArrayList et qui consigne les éléments de L_N. Supprimer les éléments de ArrL un à un en exécutant une boucle avec l'appel de la méthode remove(i) de LL, avec i allant de N-1 jusqu'à zéro; ainsi on supprime le dernier élément à chaque itération. Le programme doit afficher le temps en nanosecondes qu'a pris l'exécution de la boucle. L'affichage la durée doit être similaire à : « Suppression des éléments de ArrL avec remove(i) ciblant les derniers éléments d'un ArrayList, avec une durée de duréeArrayInverse = ….. nanosecondes »

Pour déterminer le temps d'exécution d'un ensemble de lignes de code on vous suggère d'utiliser des instructions comme :

starttime = System.nanoTime( )

…. Des lignes de codes….

endittime = System.nanoTime( )

duration = endtime - starttime

Le code Java de votre programme doit former un seul fichier Java.

Veuillez exécuter votre programme avec N=1000 et veuillez inclure l'affichage sortie de votre programme dans votre réponse à cet exercice. (2 pts)

On attend que les durées affichées soient différentes selon les trois choix utilisés pour supprimer les éléments. Veuillez interpréter les trois comparaisons des durées, en comparant chaque durée aux deux autres durées (2 pts). Vos comparaisons doivent se baser sur la structure des données (SD) sous-jacentes implémentées par chacune des deux classes ArrayList et LinkedList; vous devez vous baser aussi sur les coûts d'accès et de modifications qui caractérisent chacune des SD.

Pour avoir une idée des SD implémentées pour chacune des classe ArrayList et LinkedList vous pouvez consulter les documentations de ces deux classes.

Votre réponse à cette question de comparaison est mieux d'être sous-forme tabulaire en remplissant une copie du tableau suivant :

Couple de durées	Comparaison interprétée
duréeLinkedList vs duréeArrayDirect	
duréeLinkedList vs duréeArrayInverse	
duréeArrayDirect vs duréeArrayInverse	

Livrables

Un fichier zip contenant :

1- Un fichier pdf nommé noms-prénomsDesÉtudains-Devoir2.pdf incluant vos réponses aux exercices 1 et 2. Rappel : La réponse de l'exercice 2 comprend une illustration d'une exécution de votre programme ainsi que les comparaisons interprétées.
2- Un **seul** fichier java nommé noms-prénomsDesÉtudains-Devoir2.java comprenant la méthode main et des commentaires concises permettant une meilleure lisibilité de votre code.

Bon travail