UNIVERSITÉ DE MONTRÉAL
Département IRO

Été 2017
Neil Stewart

IFT 2015 E17

Devoir 1.

10/10, soit 10% de la note finale.

Les 10 points "Partie pratique" pour le cours E17 seront probablement distribués de la façon suivante : Devoir #1 (1 point), Devoir #2 (3 points), Devoir #3 (6 points).

1 Partie Pratique (1 point)

Pour cette question, vous devez implanter la méthode d'insertion d'un vecteur (*dynamic array*, *array list*, et autres appellations). Un fichier squelette `ArrayList.java` vous est fourni, et vous devez écrire le code de la méthode `insert` de telle manière que :

1. l'insertion ne cause un agrandissement du tableau qu'au besoin,
2. le nombre de réallocations du tableau est minimisé,
3. la taille en mémoire du tableau est plus petite que celle d'une liste simplement chaînée de même longueur.

Vous ne pouvez modifier le code des autres méthodes.

Veuillez remettre votre version complétée de `ArrayList.java` sur Studium avec les noms et matricules des auteurs en commentaire au début du fichier.

2 Partie Théorique (9 points)

1. ($1\frac{1}{2}$ points)
   Dans la méthode *Quicksort*, à chaque étape on divise la liste à triér (de longueur N) en deux sous-listes ; on met la sous-liste la plus longue sur une pile, et on s'occupe de l'autre. Nous pouvons représenter le processus en forme de hiérarchie : chaque fois que nous sous-divisons une sous-liste, nous descendons un niveau dans la hiérarchie.
   (a) Quelle est la relation entre la profondeur de l'hiérachie et le nombre d'éléments dans la pile ?
   (b) Démontrez par induction que la profondeur maximale de l'hiérachie est $O(\log N)$.
2. (1 point)
   (a) Weiss, Exercice 2.2, page 50, parties (c) et (d) seulement. Dans les deux cas, donnez une preuve ou donnez un contre-exemple simple.
   (b) Weiss, Exercice 2.25, page 53. Expliquez votre réponse, ou expliquez pourquoi vous n'avez pas pu répondre. (Il ne suffit pas de dire que vous êtes un peu fatigué ce soir !)

3. (2 points)
Supposons qu'un monceau est représenté dans un tableau linéaire, avec cases numérotées de 1 à N. Dans chacun des cas suivants, donnez l'expresssion mathématique appropriée, ainsi que les limites sur la variable i. Par exemple, si la question était

$$\text{parent}(i) = ?, \text{ où } ?$$

la réponse serait

$$\text{parent}(i) = \lfloor i/2 \rfloor, \text{ où } 2 \leq i \leq N.$$

Par contre, si la question était

$$\text{sibling\_gauche}(i) = ?, \text{ où } ?$$

la réponse serait

$$\text{sibling\_gauche}(i) = i - 1, \text{ où } i \text{ est impair et } 3 \leq i \leq N.$$

(a) enfant_gauche(i) = ?, où ?
(b) enfant_droit(i) = ?, où ?
(c) sibling_droit(i) = ?, où ?
(d) i est feuille $\Leftrightarrow$?

4. (3 points)
Soit $T_i(m_j, N)$ le coût de résoudre un problème de taille N en utilisant la méthode m_j. Dans la notation du livre (et du cours), soit le coût moyen

$$T_{avg}(N) = \frac{1}{n} \sum_{i=1}^{n} T_i(m_j, N),$$

le coût dans le plus mauvais cas

$$T_{worst}(N) = \max_{i=1,\ldots,n} T_i(m_j, N),$$

et le coût dans le meilleur cas

$$T_{best}(N) = \min_{i=1,\ldots,n} T_i(m_j, N),$$

où n est le nombre de problèmes de taille N.
Voici deux énoncés. Dans les deux cas, donnez une preuve formelle, ou donnez un contre-exemple simple.
(a) $T_{best}(N) = O(f(N)) \Rightarrow T_{avg}(N) = \Omega(f(N))$
(b) $T_{avg}(N) = O(f(N)) \Rightarrow T_{best}(N) = O(f(N))$

5. ($1\frac{1}{2}$ points)
   Pour cet exercice vous pouvez prendre pour acquis le résultat démontré dans le cours : si un arbre binaire est "Full", le nombre de feuilles est égal au (nombre de noeuds internes) +1.

   Weiss, Exercice 4.4, Exercice 4.5, et Exercice 4.6, page 161.

À réaliser en équipes de 1 ou 2. À remettre le 31 mai, 2017, avant `9:00`. Les solutions seront affichées le 31 mai. Les devoirs en retard ne seront pas acceptés.