UNIVERSITÉ DE MONTRÉAL
Département IRO

Été 2018
Neil Stewart

IFT 2015 E18

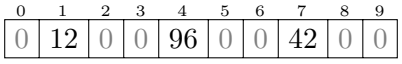
Devoir 1.

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

Les 10 points "Partie pratique" pour le cours E18 seront probablement distribués de la façon suivante : Devoir #1 (1 point), Devoir #2 (3 points), Devoir #3 (6 points).

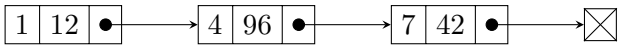
1 Partie Pratique (1 point)

Un vecteur est un tableau unidimensionnel d'éléments. Dans plusieurs applications, les éléments du vecteur peuvent être nuls. Un tel vecteur est dit creux.



Il est inefficace d'utiliser un tableau pour le stockage d'un vecteur creux. Conséquemment, il faut choisir une meilleure représentation.

Une approche, parmi d'autres, est d'utiliser une liste chaînée. Chaque élément non nul du vecteur creux est associé à un nœud dans la liste chaînée, contenant son *index*, sa *valeur* et un *pointeur* vers le prochain nœud, comme suit :



Vous devez implanter cette structure de données en *Java*. Un fichier squelette nommé `SparseVector.java` vous est déjà fourni. Veuillez remettre votre version complétée du fichier sur StudiUM, avec les noms et matricules des auteurs en commentaire au début du fichier.

2 Partie Théorique (9 points)

1. (1 point)
   Les bornes de style $O(\ldots)$ et $\Omega(\ldots)$ sont surtout utiles dans un contexte où on ne connaît pas la valeur de la fonction $T(N)$: le processus qui nous intéresse est normalement compliqué, et on n'est pas capable de dire exactement la valeur de la fonction $T(N)$. Alors au moment de l'analyse, dans des cas fastidieux, on se dit "Ouf, dans tous les cas $T(N)$ n'est jamais plus grand que ...", ou bien "Ouf, dans tous les cas $T(N)$ n'est jamais plus petit que ...", et on obtient des bornes $O(\ldots)$ et $\Omega(\ldots)$. Dans le cas où les approximations n'impliquent que, par exemple, le choix de constantes multiplicatives plus grandes ou plus petites, nous aurons aussi une borne $\Theta(\ldots)$.

Ceci étant dit, voici un fait simple qui mérite d'être remarqué. Supposons que $T(N)$ est connue explicitement : n'importe quelle fonction non-négative, par exemple

$$T(N) = \exp(N^3)/lg(N), \quad 2 \leq N.$$

Trouvez une fonction $g(N)$ telle que $T(N) = O(g(N))$, $T(N) = \Omega(g(N))$, et $T(N) = \Theta(g(N))$. Donnez les preuves.

2. (1 point)
   (a) Supposons une structure de données *queue*. Dans la suite d'entrée nous balayons de gauche à droite : une lettre veut dire "en-queue" (ajouter la lettre à la queue), et une étoile * veut dire "dé-queue" (retirer une lettre de la queue). Donnez la séquence de valeurs retournée si on applique la séquence suivante d'opérations, en supposant la queue vide au départ.

   QUE*ST*I*ON***FAC***ILE*****

   Montrez aussi, dans un seul dessin, le contenu de la queue : rayer la lettre au moment de faire dé-queue.

   (b) Imaginons maintenant des séquences d'entrée différentes de celle-ci. Supposons que le nombre lettres est égal au nombre d'étoiles. Énoncez une condition sur la séquence qui garantit que la séquence va produire une réponse analogue à la réponse produite par la séquence ci-dessus.

3. (1 point)
   (a) Weiss, Exercice 2.2, page 50, partie (c) seulement. Donnez une preuve ou donnez un contre-exemple simple.
   (b) Weiss, Exercice 2.5, page 50.

4. (2 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.
   Évidemment $T_{avg}(N) \leq T_{worst}(N)$: en effet

$$T_{avg}(N) = \frac{1}{n} \sum_{i=1}^{n} T_i(m_j, N) \leq \frac{1}{n} \sum_{i=1}^{n} \max_{k=1,\ldots,n} T_k(m_j, N) = \max_{k=1,\ldots,n} T_k(m_j, N) = T_{worst}(N).$$

   Il n'est pas possible que $T_{worst} = O(N \log N)$ et $T_{worst} = \Omega(N)$ en même temps que $T_{avg} = O(N \log^2 N)$ et $T_{avg} = \Omega(N \log^2 N)$. Donnez une preuve formelle que c'est impossible.

5. (1 point)
   Weiss, Exercice 2.7, page 50, partie (a) seulement, et seulement les cas (2), (3), (4) et (5).
6. (1 point)
   Nous avons vu dans l'Introduction du cours comment utiliser un arbre binaire de recherche, avec clefs organisées pour satisfaire l'ordre in-order (Gauche-Racine-Droite), pour chercher une clef dans un temps normalement inférieur à $O(N)$. Comment s'arranger dans la pratique pour que l'arbre satisfasse à la condition Gauche-Racine-Droite ? *Indice :* Exercice 4.9 (a), Weiss page 161.
7. (2 points)
   Weiss, Exercice 4.7, page 161. (Une preuve n'est pas nécessaire pour la deuxième partie "determine when the equality is true.")

À réaliser en équipes de 1 ou 2. À remettre le 30 mai, 2018, avant `9:00`. Les solutions seront affichées le 30 mai. Les devoirs en retard ne seront pas acceptés.