DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE

SIGLE DU COURS: IFT 2015 (E21)

NOM DU PROFESSEUR: Neil Stewart

TITRE DU COURS: Structures de Données

EXAMEN FINAL

Date : Mardi-mercredi 20-21 juillet, 2021
Heure : Examen disponible: le 20 juillet 9:00; à remettre mercredi le 21 juillet 12:00.
Lieu : À la maison

DIRECTIVES PÉDAGOGIQUES:

- Vous disposez de 27 heures pour compléter cet examen.
- Toute documentation permise. Mettez les références. Pas de bonus LaTeX.
- Vous devez remettre *exactement* cinq pages en format pdf.
  La première page servira à identifier l'examen, mais elle doit surtout avoir une boîte rectangulaire avec votre nom de famille en majuscules, votre prénom, et votre matricule, style: STEWART, Neil. Matricule 235711.
- La réponse à la question i doit être mise sur la page $i+1$, $i = 1, 2, 3, 4$. Il faut respecter rigoreusement ce format.

PLAGIAT. Constitue un plagiat:

- faire exécuter son travail par un autre
- utiliser, sans le mentionner, le travail d'autrui
- ÉCHANGER DES INFORMATIONS LORS D'UN EXAMEN
- falsifier des documents

Le plagiat est passible de sanctions allant jusqu'à l'exclusion du programme.

Figure 1: Style de la boîte exigée.

1. Question 1 *(10 points)*

   Dans la Figure 2 nous voyons un exemple de SkipList déterministe (1-2-3 deterministic SkipList, telle que vue au cours). Il y a quatre niveaux, 1, 2, 3 et 4. L'entête E est reliée au noeud terminal T au niveau 3. Le noeud 28 vient d'être inséré, mais la personne qui a implanté l'algorithme d'insertion a oublié d'augmenter le niveau de certains noeuds en faisant l'insertion:

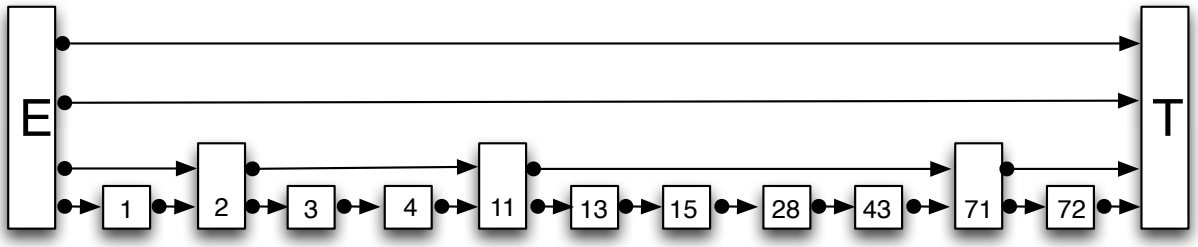

Figure 2: "1-2-3 Deterministic Skiplist"

   (a) En descendant depuis le niveau 3, pour insérer le noeud 28, il y avait un écart de 3. La hauteur de quel noeud aurait dû avoir été augmentée à cette étape?

   Ensuite, le noeud dont la hauteur vient d'être augmenté est maintenant relié au noeud 71 au niveau 2, et en descendant depuis le niveau 2, pour insérer le noeud 28, il y avait encore un autre noeud dont la hauteur aurait dû avoir été augmentée. Lequel? (Il n'est pas nécessaire de faire un dessin: dites-moi en une ou deux courtes phrases quels noeuds sont maintenant inclus dans quelles listes.) Nous avons maintenant un noeud de hauteur 3, et trois noeuds de hauteur 2.

   (b) Donnez un dessin de l'arbre 2-4 qui correspond à la SkipList après avoir inséré le noeud 28 en augmentant la hauteur de noeuds comme il fallait. (Il n'est pas nécessaire de faire un beau dessin par ordinateur: si vous préférez, vous pouvez par exemple faire cela soigneusement à la main, et inclure une photo du dessin sur la page de réponse.)

   (c) Nous avons mentionné l'idée de transformer l'algorithme top-down de retrait en algorithme de retrait pour les arbres 2-4.

   Supposons que nous voulions retirer la clef 72 de l'arbre 2-4 que vous venez de dessiner. Donnez-moi les règles qui doivent être suivies à chaque étape pour faire cela (en s'inspirant de l'algorithme top-down SkipList), mais *exprimez-vous uniquement par référence à l'arbre 2-4.* Je ne veux rien entendre de la SkipList maintenant! Je cherche l'énoncé de *la règle,* et non pas uniquement une description de ce qui s'est passé dans l'exemple actuel.

   Pour ce faire, il faudrait énoncer une règle que je pourrais noter "(a)", une règle "G-à-D" ou "D-à-G" (choisissez le cas qui est approprié pour l'exemple actuel), et une règle "(b-1)" ou "(b-2)" (choisissez le cas qui est approprié pour l'exemple actuel).

2. Question 2 *(10 points)*

   (a) Dans la méthode de chaînage externe, nous pouvons imaginer garder les listes triées. Quels sont les coûts (coût moyen) en fonction de la longueur de la liste $\lambda = N/M$ dans les deux cas (triées et non-triées) pour les opérations:

   - Insertion dans le cas où nous savons *a priori* que la clef est absente.
   - Recherche quand nous ne savons pas si la clef est présente. La réponse ici varie peut-être, dépendant du cas: *a posteriori* la clef est présente, ou *a posteriori* la clef n'est pas présente.

   Quand je dis "en fonction de λ" j'entends des fonctions du style 2λ, $\lambda/4$, 17, λ^2, $e^{17\lambda}$, ...

   (b) Dans les arbres cousus ("Threaded tree") , nous avons remplacé les références nulles par des ficelles vers des prédecesseurs et des successeurs. S'il y a N noeuds avec clef dans l'arbre cousu, il y a $N + 1$ ficelles, y compris les pointeurs nuls au début et à la fin.

   Démontrez formellement que la complexité d'un parcours in-order (GRD) d'un arbre avec ficelles est $O(N)$.

3. Question 3 *(10 points)*

   (a) Dans le pseudo-code pour l'algorithme de Prim il y avait une boucle **pour**, boucle qui a dû être faite pour tout w non-déjà visité et adjacent à v: **faire** ...

   Quelles sont les modifications nécessaires (à la place des trois petit points ...) pour transformer l'algorithme en algorithme de Dijkstra?

   (b) Le champs p dans l'algorithme de Prim a pu être interprété comme "point d'attache". Comment interpréter le champs p dans le cas de Dijkstra? (Peut-être regardez la partie (c) avant de répondre.)

   (c) Supposons que nous ayons appliqué l'algorithme de Dijkstra, en se servant d'une table T avec le format utilisé dans le cours pour l'algorithme de Prim. Écrivez du pseudo-code (en utilisant la notation du cours, style $T[v].d$, ..., **pour**, **faire, si**, **sinon**, ...) pour une procédure récursive qui imprime le chemin pour se rendre du noeud d'origine au noeud donné. Par exemple, si le noeud d'origine était v_4, et le graphe était celui de l'exemple utilisé dans le cours pour illustrer l'algorithme de Prim, alors *Imprimer*(v_6, T) va imprimer v_4 à v_5 à v_7 à v_6. Supposez disponible une fonction *Écrire*$(\ldots)$ capable d'écrire des chaînes de caractères et des sommets particuliers.

4. Question 4 *(10 points)*

   (a) Pour l'insertion top-down dans un arbre rouge-noir, nous avons dit que si nous insérons rouge, et le parent a déjà un enfant rouge (nous avons donc maintenant deux sibling rouges), ce qu'il faut faire est de colorer noirs les deux sibling, et de colorer rouge le parent. Ensuite il faut regarder le grandparent pour peut-être faire autre chose.

   Mais qu'est-ce qu'il faut faire dans le cas spécial où le parent est la racine (il n'y a pas de grandparent)? Nous avons surtout parlé de 2 des règles qui doivent être satisfaites pour les arbres rouge-noir, mais en fait il y avait 4 règles au total: expliquez en fonction de ces quatre règles.

   (b) Un ami vous dit que le Splay Tree est une idée terrible parce que, par exemple:

      i. pour insérez les N clefs dans l'ordre cela prend $O(N^2)$ opérations pour créer ce qui est finalement une liste linéaire ($1+2+3+\ldots+N = N(N+1)/2 = O(N^2)$) et ensuite ...

      ii. ... ensuite, pour accéder à la clef 1, en faisant les rotations subséquentes, il faut faire $O(N)$ opérations, tandis que cela serait seulement $O(\log N)$ pour l'arbre rouge-noir.

   Vous donnez une réponse courte et précise pour la première des deux remarques. C'est quoi?

   (c) Pour la remarque ii) votre réponse est plus nuancée. C'est quoi?

   Pour commencer, est-ce que votre ami a raison? Peut-on lui indiquer un résultat qui va quand même rassurer? Est-ce que ce résultat est plus fort ou moins fort que celui que nous pouvons donner pour l'arbre rouge-noir dans le contexte d'une analyse amortie? Peut-on en effet donner un résultat pour les arbres rouge-noir, dans le style d'une analyse amortie? Peut-on donner un résultat comme l'analyse ordinaire dans le cas de l'arbre rouge-noir, mais cette fois-ci pour les Splay Tree?

   Je fais exprès pour être un peu vague dans le paragraphe que je viens d'écrire, c'est simplement pour guider votre réponse. Vous pouvez vous exprimer partout avec la notation $O(\ldots)$.

Fin de l'examen.

Neil Stewart