IFT2015 H09 — Examen Final

Miklós Csűrös

20 avril 2009

Aucune documentation n'est permise à l'exception d'une feuille de $8\frac{1}{2}'' \times 11''$.
L'examen vaut 150 points.
Répondez à toutes les questions dans les cahiers d'examen.

0 Votre nom (1 point)

►Écrivez votre nom et code permanent sur tous les cahiers soumis.

1 Table de symboles $3 \times 3 \times 3$ (39 points)

(a) Implémentations (27 points) On a vu plusieurs structures de données qui peuvent servir à implémenter le type abstrait de la table de symboles. L'efficacité des implémentations n'est pas la même : ici vous devez comparer le temps de calcul pour trois opérations fondamentales : insertion, recherche fructueuse, et recherche infructueuse. ►Donnez le temps de calcul des trois opérations avec les trois structures de données considerées suivantes : une liste chaînée d'éléments non-triés, un arbre rouge et noir, et un tableau de hachage avec adressage ouvert (hachage double), dont la facteur de remplissage $\alpha < 0.75$. Spécifiez le temps de calcul comme une fonction du nombre des éléments n, en utilisant la notation asymptotique, dans trois cas : le pire cas, le meilleur cas, et en moyenne. (Donc, cela fait 27 résultats de complexité temporelle au total.) Il ne faut pas justifier vos réponses (sauf trois, voir (b) ci-dessous).

(b) Justifications (12 points) ►Élaborez *trois* de vos réponses (votre choix lesquels) en **(a)** : expliquez comment la structure assure un tel temps de calcul.

2 Fusion (20 points)

Lors du tri par fusion on a utilisé l'algorithme de fusionner deux listes triées en un temps linéaire. ►Donnez un algorithme pour fusionner k listes triées dans un temps $O(\ell \log k)$ quand la longueur totale des listes est ℓ (donc ℓ est la longueur du résultat). **Indice :** utilisez un tas de taille k (il existe d'autres solutions aussi).

3 Tableau de hachage (30 points)

Chaînage séparé (15 points) Dans une impléméntation typique du chaînage séparé, on utilise une liste chaînée pour chaque case du tableau de hachage : une liste contient tous les éléments avec la même valeur de hachage. Quand on cherche une clé x dans le tableau, on doit vérifier si x se trouve sur la liste associée avec la valeur de hachage $h(x)$, ce qui prend un temps linéaire. Par contre, la recherche prendrait un temps logarithmique avec un arbre binaire de recherche (ABR). ►Expliquez pourquoi on utilise rarement le chaînage séparé avec des ABRs à chaque case au lieu de listes.

Grappe forte (15 points) ►Expliquez le phénomène de grappe forte associé avec le sondage linéaire.

4 Successeur (30 points)

Considérez un arbre binaire de recherche. Soit SUCCESSEUR(x) l'algorithme usuel qui retourne le nœud successeur d'u nœud x (ou null si x a la clé maximale).

Pire cas (15 points) ▶Démontrez que l'algorithme SUCCESSEUR prend $O(h)$ temps dans le pire cas, où h est la hauteur de l'arbre. (**Indice** : illustrez explicitement le nœud x du pire cas.)

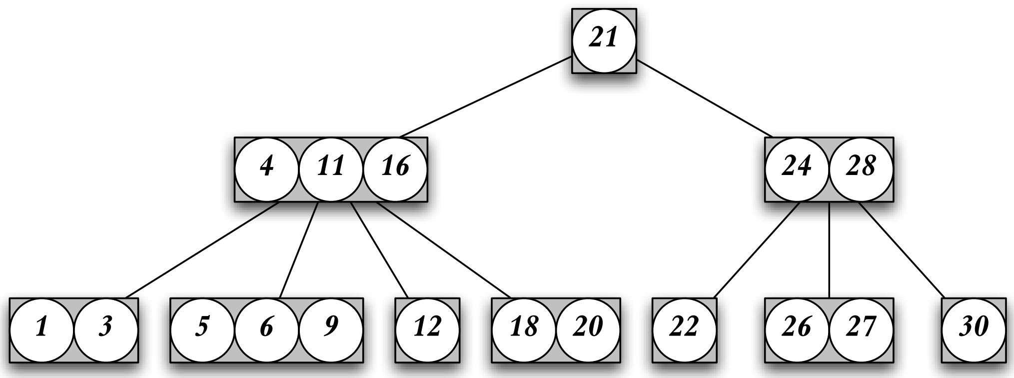
Coût amorti (15 points) On peut énumérer les nœuds de l'arbre en utilisant l'affectation $x \leftarrow$ SUCESSEUR(x) dans une boucle, à partir du nœud avec la clé minimale. ▶Démontrez qu'une série de $(n-1)$ exécutions consécutives de $x \leftarrow$ SUCCESSEUR(x) sur un arbre à n nœuds prend $O(n)$ temps au total. (**Indice** : examinez comment SUCCESSEUR visite les arêtes de l'arbre.) ▶Quel est le temps de calcul amortisé d'un appel à SUCCESSEUR dans une telle série ?

Indice. Le code de l'algorithme est le suivant.

SUCCESSEUR(x) *// trouve le successeur de x dans l'arbre*
si droit$(x) \neq$ null
    alors $y \leftarrow$ droit(x) *// successeur est le min dans le sous-arbre droit*
        tandis que gauche$(y) \neq$ null **faire** $y \leftarrow$ gauche(y)
        retourner y
    sinon $y \leftarrow$ parent(x) *// x est le max dans le sous-arbre gauche du successeur*
        tandis que $y \neq$ null et $x =$ droit(y)
            faire $x \leftarrow y; y \leftarrow$ parent(y)
        retourner y

5 Arbre 2-3-4 (15 points)

Considérez l'arbre 2-3-4 de l'illustration ci-dessous.



►Montrez la séquence de transformations dans la structure, ainsi que la structure résultante quand on insère la clé «8» dans le 4-nœud avec les clés «5», «6», et «9» (on performe l'éclatement/découpage en ascendant).

6 Algorithme de Kruskal (15 points)

L'algorithme de Kruskal construit un arbre couvrant minimal sur un graphe connexe à n sommets et m arêtes dans un temps de $O(m \log m)$ quand le graphe est representé par des listes d'adjacence. ►Quel est le temps de calcul de l'algorithme quand le graphe est representé par une matrice d'adjacence ? Justifiez votre réponse.

BONNE CHANCE !