Algorithmique TD3

Magistère Informatique 1ère Année

1er Octobre 2009

Exercice 1 *Files*

Une file est une liste linéaire où les insertions se font toutes d'un même côté et les suppressions toutes de l'autre côté (à l'inverse des piles dans lesquelles insertions et suppressions sont faites du même côté).

1. Définir le type abstrait file
2. Implémenter une file à l'aide de deux piles et calculer le coût amorti d'une opération.

Exercice 2 *Tri par Comparaisons*

Montrer que le nombre moyen de comparaisons effectuées par un tri par comparaisons est en $\Omega(n \log n)$.

Exercice 3 *Hauteur moyenne d'un ABR*

1. Calculer la hauteur moyenne d'un ABR construit aléatoirement.

   Univers: Permutations de taille n avec distribution uniforme.
   Hauteur: Pour une permutation σ, hauteur de l'arbre obtenu par insertions successives de $\sigma(1), ..., \sigma(n)$.

2. Calculer le coût moyen d'une recherche dans un ABR construit aléatoirement.

Exercice 4 *Arbres Binaires de Recherche*

1. Ecrire une fonction $SUPPRIMER(ABR)$ qui prend en argument une clé k et un arbre binaire de recherche t, et qui supprime k de t.
2. Ecrire une fonction $SCINDER(t, x)$ qui prend en argument un arbre binaire de recherche t et une clé k et qui retourne une décomposition (t_1, t_2) de t telle que :

$$\begin{aligned} c(t) &= c(t_1) \cup c(t_2) \\ c(t_1) &= \{c \in c(t) | c \leq k\} \\ c(t_2) &= \{c \in c(t) | c > k\} \end{aligned}$$

3. Ecrire une fonction

$$CONCATENER(t_1, t_2)$$

qui prend en argument deux arbres binaires de recherche t_1 et t_2, tels que:

$$max(c(t_1)) < min(c(t_2))$$

et qui retourne un arbre binaire de recherche t tel que:

$$c(t) = c(t1) \cup c(t2)$$

Exercice 5 *Arbres Binaires de Recherche Balisés*

Dans un ABR balisé, les clés ne se trouvent qu'aux feuilles et les noeuds internes ont tous exactement 2 fils et contiennent des balises qui permettent d'orienter la recherche. Si x est un noeud interne, alors:

$$\{c(x.g)\} < x.balise \leq \{c(x.d)\}$$

1. Montrer que $n(a) = 2f(a) - 1$ si a est un ABR balisé.
2. Ecrire les procédures d'insertion et de suppression dans un ABR balisé.
3. Ecrire une procédure pour transformer en temps linéaire un ABR balisé en un ABR classique ayant la même structure.
4. Ecrire la procédure inverse.

Exercice 6 *Récursivité et Piles*

1. Écrire une fonction récursive pour évaluer une expression préfixe.

   $$/ + 8 \times 3\ 4 - 5\ 3$$

2. Écrire une version itérative de cette fonction (utiliser les piles).
3. Écrire une version itérative du tri rapide (utiliser les piles).

Exercice 7 *Expressions arithmétiques*

1. La grammaire des expressions arithmétiques en notation préfixe est:

   ```
   EXP := REEL | OP_BIN EXP EXP | OP_UN EXP | (EXP)
   ```

   - Ecrire une fonction récursive $EVAL$ pour évaluer une expression en notation préfixe.
   - Donner une version itérative de la fonction $EVAL$.

2. Considérons maintenant les expressions en notation infixe données par la grammaire:

   ```
   EXP := REEL | EXP OP_BIN EXP | OP_UN EXP | EXP
   ```

   Ecrire une fonction qui construit l'arbre représentant une expression infixe en respectant les règles de priorité et d'associativité.

Exercice 8 Etant donné un alphabet Σ ordonné, on définit l'ordre lexicographique sur Σ^* comme suit: Soit $w = a_0a_1...a_p$ et $w' = b_0b_1...b_q$, $w < w'$ si et seulement si w est un prefixe de w' ou $\exists j$ tel que $\forall i < j$ $a_i = b_i$ et $a_j < b_j$.

La structure d'arbre de base peut contenir des mots écrits sur un alphabet à deux lettres. Par exemple l'arbre ci-dessous contient les mots bab, ba, baa, bbb et a. Soit $\mathcal{L}$ un langage fini sur un alphabet à deux lettres dont la somme des longueurs des mots vaut n. Montrer qu'on peut utiliser la structure d'arbre de base pour trier le langage lexicographiquement en temps $O(n)$.