Algorithmique TD3

Magistère Informatique 1ère Année

30 Septembre 2010

Exercice 1 On souhaite implémenter une pile à l'aide d'un tableau. Si on fixe la taille du tableau a priori, on perd en efficacité pour deux raisons : lorsqu'il y a trop peu de données, beaucoup de place est perdue ; lorsqu'il y a trop de données, on ne peut pas tout rentrer dans le tableau. Pour résoudre ce problème, on utilise des piles dynamiques dont la taille peut changer de telle sorte qu'au moins la moitié du tableau est toujours rempli. L'opération qu'on veut implémenter est l'empilement : elle consiste à insérer l'élément à la première place libre lorsque le tableau n'est pas plein ; sinon, on crée un nouveau tableau de taille double du précédent, on copie dedans tous les éléments du premier tableau puis on empile finalement l'élément dedans.

1. Implémenter la fonction `empiler`(x).
2. Quelle est la complexité dans le pire cas de l'appel `empiler`(x) ?
3. Trouver la complexité amortie de la fonction `empiler`(x) lorsqu'on commence par une pile vide.
4. On considère une procédure différente `empiler2`(x) qui consiste à augmenter la taille du tableau d'une constante k plutôt que de la doubler, lorsque le tableau est plein. Quelle est la complexité amortie de cette procdure ?
5. Même question dans le cas où on élève au carré la taille du tableau lorsqu'il est plein.
6. Supposons qu'on veuille également une opération de dépilement `depiler`(), qui consiste à effacer le dernier élément du tableau. Si la suppression du dernier élément résulte en un tableau à moitié plein, on crée un tableau de taille divisée par deux et on copie les éléments du tableau dedans. Quel est le coût amorti des opérations `empiler`(x) et `depiler`().

Exercice 2 *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 3 *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) telle que la taille maximale de la pile est inférieur à $\log n$

Exercice 4 *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 fructueuse et infructueuse dans un ABR construit aléatoirement.

Exercice 5 *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.