Programmation avancée

Répétition 4: Tas, file à priorité et arbre

Jean-Michel Begon

novembre 2014

Exercice 1

(a) Combien y a-t-il au minimum (resp. maximum) d'éléments dans un tas de hauteur n ?

(b) Où l'élément le plus petit réside dans un tas-max ?

(c) Quelle est la relation entre un tas-min et un tableau trié (par ordre croissant) ?

(d) Est-ce que le tableau $[23, 17, 14, 6, 13, 10, 1, 5, 7, 12]$ est un tas-max ?

Exercice 2

Dessiner tous les tas-MIN possibles avec l'ensemble des clés $\{A, B, C, D, E\}$, où chaque clé n'apparaît qu'une seule fois.

Exercice 3

Soit un tas-MIN `H`. Illustrer l'exécution des opérations suivantes :

```
heapInsert(H, 5)
heapInsert(H, 4)
heapInsert(H, 7)
heapInsert(H, 1)
heapExtractMin(H)
heapInsert(H, 3)
heapInsert(H, 6)
heapExtractMin(H)
heapExtractMin(H)
heapInsert(H, 8)
heapExtractMin(H)
heapInsert(H, 2)
heapExtractMin(H)
heapExtractMin(H)
```

Exercice 4

(a) Comment implémenter une pile au moyen d'une file à priorité ?

(b) Comment implémenter une file au moyen d'une file à priorité ?

(c) Comment implémenter une file aléatoire au moyen d'une file à priorité ?

(d) Comment implémenter une file à priorité maximum avec un tas-min ?

(e) Comment implémenter une file à priorité qui soit $\Theta(1)$ pour l'insertion et $O(n)$ pour l'extraction ?

Exercice 5

Le parcours préfixe d'un arbre binaire donne :

```
A B C - - D - - E - F - -
```

En donner les parcours infixe et postfixe. Les lettres correspondent aux noeuds internes et les tirets à des feuilles de l'arbre.

Exercice 6

Implémenter les parcours en profondeur (infixe, préfixe, postfixe) ainsi qu'en largeur pour les tas.

Exercice 7

Lors d'un parcours en profondeur d'un arbre, on utilise (implicitement) une pile. Lors du parcours en largeur, on utilise une file. Que se passe-t-il si on utilise une file à priorité ? (On suppose que les éléments de l'arbre sont des entiers, et qu'il s'agit d'une file à priorité maximum.)

Exercice 8

Soit un arbre `T`.

(a) Ecrire une méthode `size(T)` qui détermine le nombre d'éléments dans l'arbre `T`.

(b) Ecrire une méthode `isComplete(T)` qui détermine si l'arbre `T` est complet ou non.

(c) Ecrire une méthode `mirror(T)` qui retourne l'arbre miroir de l'arbre `T`.

(d) Ecrire une méthode `printLowerThan(T, x)` qui imprime les valeurs contenues dans `T` inférieures ou égales à x. Quelle est la complexité de cette opération ?

Bonus : Pour chaque fonction, décrivez un algorithme a) récursif et b) non-récursif. *Bonus :* Pourquoi les implémentations récursives sont faciles ?

Exercice 9

Prouver que le nombre de feuilles d'un arbre binaire est égal au nombre de noeuds de degré 2, plus 1.

Bonus

Bonus 1

Les files à priorité implémentées sur base de tas offrent de bonnes performances mais supposent une capacité bornée *a priori*. On souhaiterait dépasser cette limitation en utilisant un tas extensible.

(a) Proposer le pseudo-code correspondant.

(b) A-t-on conservé les performances de l'implémentation à base de tas ?

Implémenter une file à priorité se basant sur un tas-max extensible.

Bonus 2

Un tas d'arité d ($d-ary$ heap), est similaire a un tas binaire mais où chaque noeud (interne) peut posséder jusqu'à d fils.

(a) Comment représenter un tel tas dans un tableau ?

(b) Dessiner l'arbre ternaire correspondant au tableau suivant :
$[25, 12, 24, 15, 8, 7, 4, 13, 6, 3, 5, 9, 10, 1, 2]$. S'agit-il d'un tas-max ?

(c) Quelle est la hauteur d'un tas d'arité d de n éléments en fonction de n et de d ?